UNIDAD ACADÉMICA:

OFICINA DE POSTGRADOS

TEMA:

DESARROLLO DE UNA APLICACIÓN WEB PARA LA OPTIMIZACIÓN DE PROCESOS DE ATENCIÓN AL USUARIO DE UNA UNIDAD DE POSTGRADOS

Proyecto de Investigación y desarrollo previo a la obtención del título de

Magister en Gerencia Informática

Línea de Investigación, Innovación y Desarrollo principal:

Ingeniería de Software y/o Plataformas Educativas

Caracterización técnica del trabajo:

Desarrollo

Autor:

Ricardo Daniel Soria Zevallos

Director:

José Marcelo Balseca Manzano, Mg.

Ambato – Ecuador

Mayo 2018

Desarrollo de una Aplicación Web para la Optimización de Procesos de Atención al Usuario de una Unidad de Postgrados

Informe de Trabajo de Titulación
presentado ante la
Pontificia Universidad Católica del Ecuador
Sede Ambato

por

Ricardo Daniel Soria Zevallos

En cumplimiento parcial de
los requisitos para el Grado de
Magister en Gerencia Informática
con mención en Desarrollo De
Software y Redes

Oficina de Postgrados
Mayo 2018

Desarrollo de una Aplicación Web para la Optimización de Procesos de Atención al Usuario de una Unidad de Postgrados

Aprobado por:

María Fernanda San Lucas, Mg.
Presidente del Comité Calificador
Coordinadora de la oficina de Postgrados

Santiago Alejandro Acurio Maldonado, Mg.
Miembro Calificador

José Marcelo Balseca Manzano, Mg.
Miembro Calificador
Director de Proyecto

Hugo Rogelio Altamirano Villarroel, Dr.
Secretario General

Verónica Maribel Pailiacho Mena, Mg.
Miembro Calificador

Fecha de aprobación:
Mayo 2018

Ficha Técnica

Programa: Magister en Gerencia Informática

Tema: Desarrollo de una Aplicación Web para la Optimización de Procesos de Atención al Usuario de una Unidad de Postgrados

Tipo de trabajo: Proyecto de Investigación y Desarrollo

Clasificación técnica del trabajo: Desarrollo

Autor: Ricardo Daniel Soria Zevallos

Director: José Marcelo Balseca Manzano, Mg.

Líneas de Investigación, Innovación y Desarrollo

Principal: Ingeniería de Software y/o Plataformas Educativas

Resumen Ejecutivo

Este proyecto tiene por objeto optimizar los procesos de atención a usuarios de la Oficina de Investigación y Posgrados (OIP) de la Pontificia Universidad Católica del Ecuador Sede Ambato (PUCESA), mediante la implementación de una aplicación web que despliega en tiempo real la información referente al estado de los maestrantes y de sus proyectos de grado, tanto a nivel individual como colectivo. Actualmente, dicha información se maneja de manera manual y su procesamiento tarda demasiado, lo que ocasiona que la atención a los usuarios externos sea lenta y que la información gerencial requerida por la dirección de la OIP no se encuentre disponible de manera oportuna.

Mediante la implementación de tal aplicación web se pretende automatizar el procesamiento de la información interna de la OIP, lo que soluciona ambos problemas. Se emplea como métodos de recolección de información la observación directa y las entrevistas con el personal de la OIP, y como metodología de desarrollo de software se recurre a *Extreme Programming* (XP), que es la que mejor se adapta a posibles cambios en la estructura del proyecto y a la escasez de tiempo para su desarrollo.

Como resultado se obtiene una aplicación web que es debidamente validada tanto a nivel funcional como técnico, la misma que queda lista para ser implementada y puesta en producción en los servidores de la PUCESA, resolviendo efectivamente los problemas para los que fue creada.

Declaración y Autorización

Yo: RICARDO DANIEL SORIA ZEVALLOS, con CC. 180270892-3, autor del trabajo de graduación intitulado: "Desarrollo de una Aplicación Web para la Optimización de Procesos de Atención al Usuario de una Unidad de Postgrados", previa a la obtención del título profesional de Magister en Gerencia Informática, en la oficina de postgrados.

1.- Declaro tener pleno conocimiento de la obligación que tiene la Pontificia Universidad Católica del Ecuador, de conformidad con el artículo 144 de la Ley Orgánica de Educación Superior, de entregar a la SENESCYT en formato digital una copia del referido trabajo de graduación para que sea integrado al Sistema Nacional de Información de la Educación Superior del Ecuador para su difusión pública respetando los derechos de autor.

2.- Autorizo a la Pontificia Universidad Católica del Ecuador a difundir a través de sitio web de la Biblioteca de la PUCE Ambato, el referido trabajo de graduación, respetando las políticas de propiedad intelectual de Universidad

Ambato, mayo 2018

RICARDO DANIEL SORIA ZEVALLOS

CC. 180270892-3

Dedicatoria

En primer lugar a Dios, ya que por su inmensa voluntad puedo cumplir esta importante meta de mi vida.

A mis padres, ya que gracias su apoyo incondicional me ha sido posible cursar y culminar esta carrera.

A mi esposa Gabriela, por su constante apoyo, comprensión y asistencia durante el desarrollo del presente proyecto.

A mis hermanos, sobrinos y demás familiares, quienes de una u otra forma han contribuido siempre con mi desarrollo personal y profesional.

Reconocimientos

Agradezco de manera muy especial todo el apoyo y asistencia recibidos de parte de mi director de proyecto, el Ing. Mg. Marcelo Balseca Manzano, quien nunca dudó en compartir sus conocimientos y experiencia con mi persona a lo largo del desarrollo del presente proyecto.

De igual forma agradezco al personal del Departamento de Investigación y Posgrados de la PUCESA, de manera particular a su Directora la PhD. Varna Hernández Junco y, posteriormente, al Mtr. Diego Jiménez, ya que gracias a su valiosa ayuda he podido desarrollar y culminar exitosamente el presente proyecto.

Debo también agradecer al personal del Departamento de Informática de la PUCESA, y de manera particular al Ing. Mg. José Fabián Enríquez, puesto que su asistencia desde el punto de vista técnico fue muy valiosa y resultó de vital importancia para el desarrollo del presente proyecto.

Resumen

Este proyecto tiene por objeto optimizar los procesos de atención a usuarios de la Oficina de Investigación y Posgrados (OIP) de la Pontificia Universidad Católica del Ecuador Sede Ambato (PUCESA), mediante la implementación de una aplicación web que despliega en tiempo real la información referente al estado de los maestrantes y de sus proyectos de grado, tanto a nivel individual como colectivo. Actualmente, dicha información se maneja de manera manual y su procesamiento tarda demasiado, lo que ocasiona que la atención a los usuarios externos sea lenta y que la información gerencial requerida por la dirección de la OIP no se encuentre disponible de manera oportuna. Mediante la implementación de tal aplicación web se pretende automatizar el procesamiento de la información interna de la OIP, lo que soluciona ambos problemas. Se emplea como métodos de recolección de información la observación directa y las entrevistas con el personal de la OIP, y como metodología de desarrollo de software se recurre a Extreme Programming (XP), que es la que mejor se adapta a posibles cambios en la estructura del proyecto y a la escasez de tiempo para su desarrollo. Como resultado se obtiene una aplicación web que es debidamente validada tanto a nivel funcional como técnico, la misma que queda lista para ser implementada y puesta en producción en los servidores de la PUCESA, resolviendo efectivamente los problemas para los que fue creada.

Palabras clave: aplicación, web, optimización, procesos, atención, usuarios, posgrados

Abstract

This project aims to the improve customer service processes at the Research and Postgraduate Department (OIP) – Pontifical Catholic University of Ecuador in Ambato (PUCESA). To this end, a web application will provide information of the processes status in regard to postgraduate students´ thesis on both an individual and collective level. Direct observation and interviews directed to the OIP´s staff were applied to gather information. Also, Extreme Progamming (XP) was used as a software development methodology, as it is the one that best suits to the project structural changes and the lack of time for its development. As a result, this web application is validated at a functional and technical level, which will be ready to be implemented and launched by the servers at PUCESA thus, resolving effectively the problems for which it was created.

Keywords: application, web, optimization, processes, assistance, users, postgraduate

Tabla de Contenidos

Lista de Tablas

Lista de Figuras

Capítulo 1

Introducción

Sin lugar a dudas, la informática o ciencia de la información es una de las ramas del conocimiento humano que más rápidamente avanza, experimentando cambios o novedades a veces de un día para otro. No es de admirarse que los equipos informáticos de todo tipo se vuelvan obsoletos muy rápidamente, tal es así que un computador que hoy en día es nuevo, en pocos meses ya se puede considerar obsoleto, lo cual es incluso considerado a nivel contable de las empresas, para las cuales los equipos tecnológicos e informáticos son los que más rápidamente pierden su valor.

Este avance, sin embargo, no solo se da a nivel de hardware, es decir de equipos, sino también a nivel de software o programas, los cuales también avanzan a igual y, en determinadas ocasiones, a mayor velocidad de lo que lo hace el hardware. Si bien el software es un bien intangible del que contablemente no se suele considerar su devaluación, esto no determina que el aparecimiento de nuevas versiones de todo tipo de programas haga que sus versiones anteriores se vuelvan obsoletas con la misma velocidad que sucede con el hardware.

Por otro lado, desde que las organizaciones de todo tipo y tamaño comenzaron a automatizar e informatizar sus operaciones, lo que en países latinoamericanos tomó especial auge durante la década de 1990, las mismas se han visto obligadas a actuar en consecuencia de la evolución del hardware y el software expuestas en los párrafos anteriores, por lo que hoy en día es común que la mayor parte de organizaciones cuente con su departamento de informática el que, a su vez, está manejado por profesionales especializados que tienen a su cargo, entre otras cosas, el mantener el hardware y el software de su respectiva organización actualizados y correctamente estructurados en pro de ganar competitividad y agilizar sus procesos de todo tipo.

Parte de las responsabilidades del personal del departamento de informática de cualquier organización suele consistir en el desarrollo e implementación de software específicamente diseñado para responder a las necesidades de tal organización, con la finalidad de manejar adecuadamente sus flujos de información sin descuidar aspectos tan críticos como la seguridad, consistencia y disponibilidad de la misma, cuando ésta es requerida. Tal es así que hoy en día las empresas y

organizaciones suelen considerar su información como uno de sus activos más importantes y valiosos, llegando a valer muchas veces en la práctica más que muchos activos tangibles.

Las aplicaciones web han ido tomando más y más auge a lo largo del tiempo debido a que ofrecen múltiples ventajas sobre las aplicaciones tradicionales de escritorio, como se detalla más adelante en el presente documento. Muchos tipos de organizaciones optan por esta alternativa para la automatización de sus procesos y, entre ellas, las instituciones universitarias y sus diferentes departamentos son de las que más suelen recurrir a este tipo de aplicaciones, sobre todo debido a la diversidad de usuarios que requieren usar sus sistemas.

El presente trabajo consiste justamente en la elaboración de una aplicación web para la automatización y optimización de los procesos que se llevan a cabo en un departamento de posgrados universitario, con la finalidad de agilizar la atención a sus usuarios tanto internos como externos y, a la vez, de almacenar y administrar adecuadamente la información generada por tal departamento, la misma que resulta crucial para el correcto funcionamiento del mismo.

1.1. Presentación del trabajo

Para la elaboración de la aplicación web descrita en el párrafo anterior se toma como caso de estudio el departamento de posgrados de la PUCESA donde se ha identificado el problema de que la mayoría de sus procesos de atención a clientes tanto externos como internos se realiza hasta hoy en día de manera manual, utilizándose apenas algunas herramientas ofimáticas de uso general para asistir en alguna medida al desarrollo de tales procesos, pero que sin embargo no permiten automatizar el manejo de la información ni manejarla adecuadamente, creándose problemas como inconsistencias en dicha información y lentitud en prácticamente todos los procesos que demandan el interactuar con dicha información.

Si se realiza una búsqueda sobre trabajos similares llevados a cabo en otras instituciones tanto a nivel local, como nacional y regional, su escasez ([1], [2]) demuestra que éste no es un problema aislado de la PUCESA sino que, por el contrario es un problema muy común que aqueja a la gran mayoría de instituciones universitarias y de educación superior, si bien se evidencia que algunas de ellas ya se encuentran realizando los trabajos de investigación y desarrollos respectivos con la finalidad de automatizar sus procesos y manejo de información, aprovechando así las bondades de la tecnología de hardware y software existentes en la actualidad.

En el caso particular de la PUCESA, se encuentra que se ha realizado ya trabajos parciales de automatización de algunas de sus áreas y departamentos, sin embargo, la OIP no ha recibido la atención requerida en este sentido, razón por la cual los procesos se siguen realizando de manera manual hasta hoy en día. Se determina entonces la necesidad urgente de automatizar sus procesos, aprovechando la infraestructura informática existente en la institución tanto a nivel de hardware como de software que, de otro modo, está subutilizada.

Para logarlo se necesita realizar un análisis detallado sobre los procesos internos y externos que se llevan a cabo en la OIP de la institución, así como de las tecnologías e información ya existentes en la misma, para determinar la mejor manera de aprovechar éstas últimas con el fin de resolver los problemas detectados en el mencionado departamento. Cabe mencionar que se requiere la colaboración tanto del personal de la OIP como del departamento de informática institucional, ya que de otro modo sería imposible establecer los requerimientos y las bases necesarias para iniciar con el desarrollo del presente proyecto.

1.2. Descripción del documento

El presente documento consiste en el Informe Final del Proyecto (IFP), donde se detallan todos los pormenores del desarrollo del presente proyecto, desde el análisis de requerimientos hasta el del producto final que consiste en la aplicación web en funcionamiento. En este documento se pueden encontrar información sobre todas y cada una de las etapas del desarrollo del proyecto, y se espera por tanto que sirva de apoyo y de material de consulta para cualquier persona que se encuentre desarrollando un proyecto de similares características.

El documento se encuentra dividido en seis capítulos y tres apéndices, donde se aborda toda la temática expuesta en los párrafos anteriores, y en esta sección se da una idea general al lector sobre el contenido y objetivo principal de cada una de ellas.

Se inicia con una sección de preliminares donde se presenta claramente el tema del proyecto y su ficha técnica, la cual a su vez incluye un resumen ejecutivo que explica a breves rasgos la temática del proyecto. Se incluye también la declaratoria de originalidad y responsabilidad, la dedicatoria, los reconocimientos, el resumen completo del proyecto así como su traducción al idioma inglés, lo que se conoce como *Abstract*, y finaliza la sección con los índices general, de tablas y de figuras.

A continuación se pasa al capítulo 1 cuyo objetivo fundamental es el de brindar al lector una visión general acerca de la problemática detectada, los antecedentes que se encuentra para la misma, y el planteamiento de una solución a dichos problemas. En este caso se trata de la falta de automatización en los procesos que tienen lugar en la OIP de la PUCESA, llevándose a cabo un análisis comparativo con la situación de otras instituciones de similares características y, por otro lado, un análisis de las herramientas y tecnologías existentes en la PUCESA, las mismas que pueden ser aprovechadas para automatizar los procesos mencionados, ofreciéndose así una solución efectiva a múltiples problemas que hoy en día se dan en tal departamento.

En el capítulo 2 se exponen algunos de los lineamientos establecidos mediante un análisis previo de la problemática detectada, los mismos que sirvieron de base para la elaboración del plan para el presente proyecto de investigación y desarrollo. Se describe con más detalle el problema de estudio y se formula preguntas clave que ayudan a delimitarlo con mayor claridad, así como la meta que se persigue con el presente proyecto. Por último se plantea el objetivo general del proyecto así como sus objetivos específicos y se termina definiendo la planificación inicial sobre las funciones que va a llevar a cabo el producto final del presente trabajo, así como las que no va a llevar a cabo.

El capítulo 3 está enteramente dedicado a lo que son las bases teóricas donde se exponen conceptos, definiciones, antecedentes históricos y otros lineamientos generales que ayudan a comprender de mejor manera la naturaleza del problema así como la solución que se plantea al respecto. Los temas sobre los que se incluye información en este capítulo son, entre otros, los relativos a gestión documental, programación orientada a objetos, bases de datos, ingeniería de software y metodologías de desarrollo de software.

El capítulo 4, por otro lado, consiste en una descripción mucho más detallada de la metodología elegida para la elaboración del producto final del presente proyecto. Al tratarse del desarrollo de una aplicación web se elige eXtreme Programming como metodología de desarrollo de software, y se detalla las tareas realizadas para cada una de sus fases como son planeación, diseño, codificación y pruebas, a lo largo de múltiples iteraciones.

A continuación está el capítulo 5 cuyo objetivo es el de describir detalladamente el producto final del presente proyecto, es decir, la aplicación web finalizada con sus respectivos menús, secciones y funcionalidad. Se presenta múltiples capturas de pantalla que permiten entender de manera muy clara y rápida lo que se logra con cada página web que forma parte de la presente aplicación.

Finalmente, el capítulo 6 tiene por objeto el presentar las conclusiones y recomendaciones que se puede formular luego de haber concluido con el desarrollo del proyecto, que se espera que sean de utilidad para la institución.

A continuación se encuentra el apéndice A del documento, que describe la base de datos "Graduacion", una de las bases de datos con las que trabaja la presente aplicación, y las respectivas tablas y campos que resultan de interés y utilidad para el proyecto.

Sigue el apéndice B que, al igual que el apéndice anterior, describe las tablas y campos de una base de datos, en este caso la base de datos "AtencionUsuario", que fue diseñada e implementada específicamente para esta aplicación.

Luego se tiene el apéndice C, cuya finalidad es la de describir detalladamente los webservices proporcionados por el personal del departamento de informática de la institución, aquellos que resultan útiles para el presente proyecto. Se procede a describir cada uno de ellos con sus respectivos campos.

Finalmente, se tiene la lista de referencias con lo cual se termina el documento.

Capítulo 2

Planteamiento de la Propuesta de Trabajo

El presente proyecto tiene por objeto la creación de una aplicación web que permite automatizar y optimizar el proceso de atención al usuario de las unidades de postgrados de las universidades ecuatorianas. Se ha tomado como caso de estudio la OIP de la PUCESA, se realiza un análisis detenido de los procesos a los que actualmente se someten sus usuarios tanto externos, refiriéndose a los maestrantes, así como internos, que constituirían los docentes, el propio personal administrativo de dicho departamento incluyendo su director(a), y el personal de cualquier otro departamento que tuviera relación alguna con las actividades que se desarrollan en la OIP de la PUCESA.

De este análisis se han identificado los principales cuellos de botella, procesos que se realizan de manera manual y presencial demandando de extensos períodos de tiempo para su conclusión, llegando en determinados casos a tomar varios días y hasta semanas. De allí se desprende la necesidad de desarrollar la presente aplicación web, cuya finalidad es la de agilizar dichos procesos al dar a sus usuarios la posibilidad de ejecutarlos vía online, eliminando la necesidad de trasladarse personalmente y ahorrando recursos críticos como el tiempo y el papel.

El desarrollo de la presente aplicación web se lo realiza haciendo uso de la plataforma ASP.NET bajo el entorno Microsoft Visual Studio 2010, y de Microsoft SQL Server 2008 R2 como motor de base de datos. Esto se decide debido a que mucha de la información requerida para la presente aplicación ya se encuentra disponible en las bases de datos que actualmente maneja la PUCESA mediante su aplicación previamente desarrollada conocida como “Academics”. Esto determina que es necesario acceder a dicha información y hacer uso de la misma, con la finalidad de evitar inconsistencias y duplicidad de la información, por lo que lo más recomendable es crear una aplicación compatible con aquella ya mencionada, de donde surge a su vez la necesidad de usar las mismas herramientas y versiones en las que fue desarrollada y sobre la que aún trabaja “Academics”.

Finalmente, dentro de la metodología de desarrollo de software se ha elegido combinar la metodología de Cascada, desde un punto de vista general, debido a que ofrece fases bien definidas dentro de una secuencia lógica y fácil de seguir, y la metodología de desarrollo ágil Extreme

Programming (XP) desde un punto de vista más práctico, debido a que esta última ofrece notables ventajas como el desarrollo incremental e iterativo, el aseguramiento de la calidad del software, la flexibilidad a la hora de introducir cambios, el trabajo en equipo y la simplicidad en el código.

2.1. Información técnica básica

Tema: "Desarrollo de una Aplicación Web para la Optimización de Procesos de Atención al Usuario de una Unidad de Postgrados"

Tipo de trabajo: Proyecto de Investigación y Desarrollo

Clasificación técnica del trabajo: Desarrollo

Líneas de Investigación, Innovación y Desarrollo

Principal: Ingeniería de Software y/o Plataformas Educativas

2.2. Descripción del problema

Hasta hace no muchos años, eran muy pocas las instituciones de educación superior que ofrecían programas de posgrado al público en general en el Ecuador. Apenas algunas universidades de las grandes ciudades como Quito, Guayaquil y posteriormente Cuenca proporcionaban este servicio, por lo que para la gran mayoría de profesionales a nivel nacional era casi imposible acceder al mismo. No es sino hasta principios y en algunos casos mediados de la década del 2000 que las unidades de posgrados comenzaron a masificarse y popularizarse en las universidades de las demás ciudades del país, y en particular de Ambato.

Esta incursión relativamente reciente de los departamentos de investigación y las escuelas de posgrados en las universidades ha determinado que, en la mayoría de los casos, los procesos correspondientes a sus actividades académicas y de atención a sus usuarios se sigan realizando de forma manual y presencial, puesto que muy pocas instituciones han dedicado el tiempo y el esfuerzo necesarios para elaborar sistemas informáticos de automatización y optimización de los mencionados procesos. Si bien existen algunos casos en que ya se ha tomado acciones en este sentido, como son la Universidad de Cuenca [3] y la Escuela Superior Politécnica del Litoral [2], a modo de ejemplos, muchas de las instituciones a nivel nacional no cuenta todavía con una herramienta informática que permita

solventar este problema, por lo que sus procesos se siguen realizando de la misma manera que hace décadas atrás.

Como es de suponerse, el mencionado fenómeno ocasiona graves problemas como son la lentitud e ineficiencia en los procesos de atención a los usuarios, tanto internos como externos, así como el desperdicio de recursos, la duplicidad e inconsistencia en la información, la impresión innecesaria de documentos y los largos tiempos de espera a los que deben someterse los usuarios externos para ser atendidos, solo por mencionar algunos de los problemas más notables.

La OIP de la PUCESA no se escapa a la mencionada realidad por lo que, luego de haber realizado el correspondiente análisis de sus procesos y de la información disponible en la institución, se ha identificado la necesidad urgente de elaborar una aplicación que permita superar en la mayor medida posible los problemas mencionados, ahorrando tiempo y demás recursos a todas las personas que de una u otra manera requieren de los servicios de dicho departamento. Del mencionado análisis se ha decidido que la mejor solución sería la implementación de una aplicación web, ya que la misma es fácilmente accesible a todos sus usuarios y además permite utilizar adecuadamente la información ya disponible en la institución, la misma que actualmente forma parte de la ya existente aplicación conocida como “Academics” [4].

2.3. Preguntas básicas

¿Cómo aparece el problema que se pretende solucionar?

La creación de las diferentes Maestrías en la PUCESA ha hecho que se generen innumerables trámites que aún se realizan de forma manual, creando muchos cuellos de botella.

¿Qué lo origina?

La inexistencia de una aplicación que permita la adecuada atención a los usuarios de las unidades de postgrado.

2.4. Formulación de meta

Desarrollar una aplicación web que optimice los procesos de atención a usuarios de una unidad de postgrados.

2.5. Objetivos

Objetivo General

Desarrollar una aplicación web para la optimización de los procesos de atención al usuario de una unidad de postgrados.

Objetivos Específicos

1. Identificar detalladamente la totalidad de los procesos a los que deben someterse los usuarios de la unidad de postgrados de la PUCESA en la actualidad, con sus correspondientes flujos de información.
2. Determinar la estructura y el funcionamiento actual de la plataforma Academics [4] y sus correspondientes bases de datos.
3. Construir la aplicación web en cuestión, en base a la información y resultados obtenidos de las investigaciones realizadas en los puntos anteriores.
4. Implementar y evaluar la aplicación desarrollada dentro de la infraestructura de la PUCESA.

2.6. Delimitación funcional

Pregunta 1. ¿Qué será capaz de hacer el producto final del trabajo de titulación?

- Uno de los módulos de la aplicación web se encargará de permitir y validar la autentificación de los usuarios que tienen derechos para acceder al sistema.
- Otro módulo de la aplicación web a desarrollarse en el presente trabajo se encargará de ofrecer al usuario un listado de todas las solicitudes y oficios que éste puede ingresar al sistema, a fin de que pueda elaborar cualquiera de ellos en línea y enviarlo directamente a la OIP de la PUCESA.

- Otro de los módulos tendrá la función de dar seguimiento a los documentos electrónicos y/o físicos generados por los usuarios de la aplicación, encargándose de redirigirlos adecuadamente a los departamentos o personas que se requiera, guardándose el status de dichos documentos en todo momento.
- Para el caso de documentos impresos, otro módulo de la aplicación permitirá guardar registro de la ubicación física de dichos documentos con total exactitud, a fin de poder localizarlos de manera inmediata en caso de requerirse.
- Un siguiente módulo tendrá por función el permitir al usuario conocer el status exacto o respuesta a sus solicitudes, pudiendo también desplegarse el historial completo de sus trámites y actividades académicas, desde su alta como usuario de la aplicación. Este módulo guardará relación directa con la aplicación a desarrollarse en [1].
- Otro de los módulos se encargará de la recuperación de la información de los maestrantes, dentro de la aplicación web a desarrollarse. Este módulo guardará relación directa con la aplicación a desarrollarse en [1].
- Adicionalmente, otro módulo permitirá el acceso a profesores para la administración de su información pertinente incluyendo hojas de vida, certificaciones, syllabus, administración de horarios y disponibilidad de tiempos.
- Existirá también un módulo de administración que permitirá el correcto manejo de información dentro de la aplicación, atendiendo a niveles de autoridad y derechos de usuarios.
- Finalmente, un último módulo de la aplicación permitirá al usuario desplegar cualquier documento relativo a su actividad académica con la configuración adecuada en formato PDF, listo para la impresión.

Pregunta 2. ¿Qué no será capaz de hacer el producto final del trabajo de titulación?

- La aplicación a desarrollarse no se encargará de la gestión detallada de procesos de graduación de postgrados, puesto que se encuentra en desarrollo una aplicación específica para tal fin como parte de otro proyecto de investigación y desarrollo [5], con la que la presente aplicación se integrará complementándola y haciendo uso de sus datos.

Capítulo 3

Marco Teórico

3.1. Definiciones y conceptos

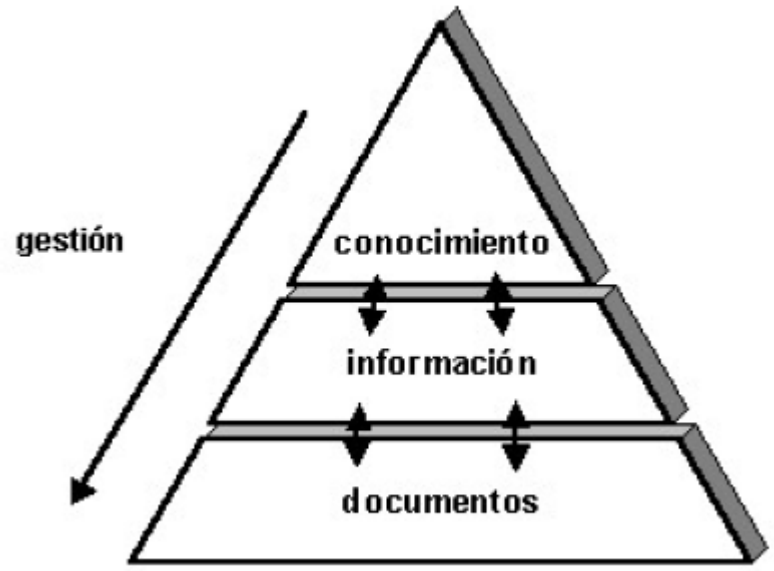
3.1.1. Gestión Documental

Se coincide íntegramente con la autora Gloria Ponjuán Dante, quien manifiesta que *"Los empleados de una organización necesitan, les resulta imprescindible, contar con sistemas de información eficaces y eficientes que puedan respaldar sus trabajos para percibir e interpretar también el ambiente en que coexisten; estos sistemas de información deberán estar respaldados por una gestión de información a nivel de toda la organización, considerando todas sus dimensiones. La gestión de información no podrá realizarse sin tener como base una adecuada gestión documental. Estos tres niveles de gestión, o esta cadena de gestión, permitirán integrar las fuerzas fundamentales que marcan el éxito en las organizaciones contemporáneas. Una gestión del conocimiento no existe sin gestión de información, una gestión de información no existe sin gestión documental"* [6]. De allí se deduce claramente que no se puede hablar de documentación sin considerar previamente los conceptos de conocimiento e información, en los que obviamente se basa todo documento, de cualquier tipo que éste sea.

Análogamente, para poder referirse a procesos de gestión documental, es necesario en primer lugar tener claras las nociones de gestión del conocimiento y de la información respectivamente, ya que las mismas se convierten en parte del contexto dentro del que tiene lugar la gestión documental, y por tanto, los tres procesos están íntimamente relacionados dentro de toda organización. Desde luego, se torna también de vital importancia definir el concepto de gestión, que según Ponjuán *"es un proceso mediante el cual se obtiene, despliega o utiliza una variedad de recursos básicos para apoyar los objetivos de la organización"*, con lo cual también se concuerda.

La mencionada autora propone el siguiente gráfico para clarificar la relación existente entre estos cuatro conceptos:

Figura 1: Integración del conocimiento con la información y los documentos



Fuente: Gestión documental, de información y del conocimiento... puntos de contacto y diferencias ([6])

3.1.1.1. Conocimiento y gestión del conocimiento

El conocimiento es básicamente un cúmulo de experiencias adquiridas por los individuos a través del tiempo, que les permiten a éstos resolver problemas de diversas índoles de manera más o menos directa y certera. El conocimiento puede estar expresado a través de datos registrados en base a las experiencias de los individuos, pero únicamente puede considerarse conocimiento cuando dichos datos son sistematizados, interpretados y utilizados por las personas con fines prácticos y de resolución de problemas, ya que de lo contrario son meros cúmulos o bancos de datos. El conocimiento puede ser compartido e intercambiado entre las personas, es decir, un individuo puede aprender a partir de las experiencias y datos registrados por otras personas.

En tal virtud, Hessen en 1970 ya explica que *"En el fenómeno del conocimiento se encuentran frente a frente la conciencia y el objeto: el sujeto y el objeto. Por ende, el conocimiento puede definirse como una determinación del sujeto por el objeto"* [7], de donde se puede deducir que sin la presencia de un sujeto,

no es posible que se dé el fenómeno del conocimiento, ya que el mismo en sí consiste en la búsqueda del objeto (en este caso una experiencia o dato) por parte de dicho sujeto.

Para entender de mejor manera el fenómeno del conocimiento, es muy importante diferenciar el conocimiento tácito del explícito, siendo el primero aquel de tipo subjetivo y que se genera puramente en base a la experiencia o al azar, es un conocimiento de tipo personal e intuitivo que muchas veces no queda expresado en ningún medio y por tanto no se transmite a otras personas, mientras que otras veces se transmite verbalmente, en especial de una generación a otra, sin llegar a ser un conocimiento de carácter público. El conocimiento explícito, por otro lado, es aquel que ha sido debidamente probado, documentado y publicado y por tanto es de aceptación oficial y científica, y suele ser transmitido al público a través de medios impresos o audiovisuales tales como libros, revistas, vídeos y páginas web.

En cuanto a la gestión del conocimiento, se debe indicar que es un proceso sobre todo ligado a las organizaciones de todo nivel, ya que son éstas las que principalmente requieren sistematizar el proceso de generación, documentación, socialización y puesta en producción de conocimientos tanto tácitos como explícitos, relacionados a sus actividades productivas y administrativas, ya que son precisamente estos conocimientos los que les van a permitir mejorar su productividad y competitividad, manteniéndose en el tiempo.

Así, Ponjuán manifiesta que *"Por gestión del conocimiento se entiende el proceso sistemático e integrador, de coordinación de las actividades de adquisición, creación, almacenaje y comunicación del conocimiento tácito y explícito por individuos y grupos con objeto de ser más efectivos y productivos en su trabajo, y cumplir los objetivos y metas de la organización"* [6], de donde conviene además resaltar que los conocimientos generados en las organizaciones deben estar alineados con las metas y objetivos estratégicos de las mismas.

3.1.1.2. Información y gestión de la información

Si bien el conocimiento puede ser considerado un conjunto sistemático de datos que se obtienen a través de la experiencia o el aprendizaje, la información puede también verse como un conjunto de datos relacionados entre sí, pero con la connotación de que dichos datos conciernen a un fenómeno o suceso en particular, y están delimitados dentro de un espacio y tiempo específicos. Esto es de vital importancia para entender el significado de información, y sobre todo la diferencia entre conocimiento

e información, ya que ésta última puede variar en diferentes tiempos o entre diferentes organizaciones, instituciones, países y demás. En este sentido, Ponjuán nos dice que *"Un grupo de datos tiene información si alguien es capaz de comprenderlos y para ello requiere conocimiento"* [6], con lo cual se coincide totalmente ya que la afirmación ilustra debidamente la diferencia y a su vez la relación existentes entre conocimiento e información.

Si bien la información es importante para todos tanto a nivel individual como colectivo, el concepto de información cobra especial importancia en las organizaciones, ya que la misma engloba datos que permiten conocer la situación exacta de una empresa, institución, gobierno u otro tipo de organización en un determinado momento y compararla con la información generada por la misma organización en otro momento, para así descubrir si se está cumpliendo con los objetivos estratégicos de dicha organización y en qué medida. Este concepto ha tomado tanta importancia que hoy en día es muy común escuchar términos tales como "era de la información" y "sociedad de la información", y también se dice que hoy en día la información es el activo más valioso de las empresas y organizaciones.

Tomando lo mencionado como antecedente, es fácil deducir que toda organización hoy en día se ve en la inevitable necesidad de implementar un sistema de gestión de su información que le permita recopilar datos, registrarlos, almacenarlos y poder consultarlos en el futuro. En este sentido, Aja asegura que *"La gestión de la Información comprende las actividades relacionadas con la obtención de la información adecuada, a un precio adecuado, en el tiempo y lugar adecuado, para tomar la decisión adecuada"* [8].

Por otro lado, Ponjuán [6] manifiesta que en toda organización moderna, la gestión de la información está basada en múltiples dimensiones como se lista a continuación:

- *Gestión de Servicios*
- *Gestión de Finanzas*
- *Gestión de Contenidos Informacionales*
- *Gestión de Recursos Humanos*
- *Gestión del Cambio*
- *Gestión de la Tecnología*
- *Gestión de las Arquitecturas Informacionales*

Las relaciones entre las mencionadas dimensiones se ilustran en el siguiente gráfico:

Figura 2: Modelo de gestión de la información

Fuente: Gestión documental, de información y del conocimiento... puntos de contacto y diferencias ([6])

3.1.1.3. Documentos y gestión de la documentación

Toda vez que una organización ha implementado su sistema, y por ende sus políticas de gestión de la información, ésta debe ser fácilmente transmisible a la persona o personas adecuadas para que las mismas puedan interpretar correctamente dicha información y puedan tomar las acciones y decisiones requeridas con total conocimiento de causa, a fin de garantizar el óptimo funcionamiento de la organización en su conjunto. De esta necesidad surge el documento, que es un medio que permite plasmar información en él y transmitirla de una persona A (emisor) a una persona B (receptor). Por otro lado, el documento permite también guardar adecuadamente registros de la información plasmada en el mismo con fines de archivo e históricos.

Según Codinas [9], *"un documento es una información registrada en un soporte material destinada a un ser humano",* mientras que Grau [10] manifiesta que *"Un documento es una información estructurada con el propósito de ser percibida por las personas de una organización".* Ambas corresponden a definiciones clásicas sobre lo que es un documento, que bien pueden aplicarse a los papiros usados por los antiguos egipcios para transmitir información, que son las formas más primitivas de documentos que se han registrado históricamente. Sin embargo, dado el aparecimiento y masificación de las tecnologías de la información, hoy en día existen documentos en formato electrónico que requieren un dispositivo especializado para ser visualizados, o incluso que pueden ser ilegibles para el ser humano ya que su destinatario es un computador.

Al igual que para el conocimiento y la información, los documentos generados dentro de una organización requieren ser clasificados, organizados, almacenados y debidamente asegurados, ya que mucha de la información que contienen es confidencial y puede poner en riesgo la normal operatividad de la organización en cuestión, si cae en manos equivocadas. Es muy importante asegurarse de que la información contenida en los documentos de una organización es accesible únicamente por la o las personas que requieren acceso a dicha información. Todos los procesos mencionados vienen a constituir lo que se conoce como la gestión de la documentación, o simplemente gestión documental, de una organización. La norma ISO 15489 – 1 [11] lo define de manera precisa al establecer que la gestión documental es *"el campo de la gestión responsable del control eficaz y sistemático de la creación, la recepción, el mantenimiento, el uso, la disposición y la preservación de registros, en la que son determinantes los procesos de captura y mantenimiento de la evidencia y la información sobre las transacciones y actividades de negocios de la organización"*.

3.1.1.3.1. Sistemas de gestión documental

Desde el aparecimiento de las primeras tecnologías de la información hasta la fecha, dichas tecnologías han ido ganando mayor campo y utilización dentro de las organizaciones, y a su vez han ido evolucionando permanentemente. Y la gestión documental no se ha quedado fuera de esta incursión masiva de la tecnología, tal es así que hoy en día existen sistemas de gestión documental especializados que prácticamente cualquier tipo de organización puede implementar y comenzar a usarla inmediatamente.

A nivel organizacional existen aplicaciones que permiten dar seguimiento detallado a la generación, envío/recepción, procesamiento y estado de los diferentes documentos que se requieren dentro de la

organización. Así, por ejemplo, se tiene Quipux [12], que es un sistema bastante utilizado por el gobierno ecuatoriano para las tareas de gestión documental de prácticamente todas sus dependencias, y de igual forma, es un sistema utilizado por otras instituciones y organizaciones. Quipux a su vez está basado en Orfeo [13], otra aplicación desarrollada originalmente por el gobierno colombiano para sus dependencias gubernamentales. La gran ventaja de estas aplicaciones es están desarrolladas bajo plataformas del sistema operativo Linux, lo cual permite la libre implementación de las mismas sin requerirse la compra de licencias de uso, si bien el proceso de implementación puede ser bastante largo y complicado, dependiendo principalmente del tamaño de las organizaciones.

Por otro lado, si bien a nivel personal no se suelen requerir de aplicaciones de gestión documental como las mencionadas en el párrafo anterior, existen programas informáticos para la generación, almacenamiento y edición de documentos de diferentes tipos incluyendo textos, hojas de cálculo y presentaciones. Estos programas especializados se suelen conocer bajo el nombre de "paquetes de ofimática", de entre los cuales el más utilizado a nivel mundial es Microsoft Office, pese a que es de pago y existen otras alternativas gratuitas pero de más difícil implementación y aprendizaje. Roberge [14] lo ilustra de manera muy simple mediante el siguiente gráfico:

Figura 3: Información y documentación en una organización

Fuente: Lo esencial de la gestión documental: sistemas integrales de gestión de los documentos electrónicos ([14])

3.1.2. Programación orientada a objetos

La Programación Orientada a Objetos (POO) es un paradigma de programación que permite encapsular datos y funciones dentro de una sola unidad lógica que cumple con un ciclo de vida determinado. Dado que este paradigma de programación está especialmente diseñado para representar y dar solución a problemas de la vida real, se ha convertido en el más utilizado dentro del mundo del desarrollo de software desde hace varias décadas, y constituye la base para los lenguajes de programación y entornos de desarrollo más populares a nivel mundial hoy en día.

Si bien la masificación en el uso de la POO se da relativamente en los últimos tiempos, sus orígenes se remontan a la década de 1960 cuando Kristen Nygaard y Ole-Johan Dahl crearon el lenguaje de programación denominado Simula en el centro conocido como "The Norwegian Computing Center". Este lenguaje introduce y utiliza por primera vez los conceptos de clase y objeto, los mismos que son básicos y fundamentales para entender la POO y sus lenguajes. Simula iría posteriormente evolucionando y diversificándose para dar lugar al aparecimiento de los lenguajes de programación modernos utilizados hoy en día, los mismos que continúan basándose en los mismos conceptos básicos aunque incluyen mucha más funcionalidad.

La POO toma especial impulso cuando comienzan a surgir las primeras plataformas visuales de desarrollo de software, las mismas que a su vez comenzaron a popularizarse cuando el sistema operativo Microsoft Windows pasó a ser un sistema completamente gráfico, dejando de lado las interfaces textuales, allá por la década de 1990. Dichas plataformas de desarrollo debieron forzosamente implementar la POO, ya que es el paradigma más adecuado para manejar los nuevos elementos visuales que se introducen dentro de esta nueva tendencia de desarrollo de software, tales como botones, cuadros de texto, casillas de verificación, botones de opción múltiple, etiquetas y demás. Este tipo de elementos requieren que la codificación se realice mediante eventos, ya que debe responder a acciones concretas de los usuarios, volviéndose muy poco práctico el uso de otras técnicas de programación como la estructurada, la funcional o la imperativa [15].

En la actualidad, debido a la masificación de dispositivos móviles tales como tabletas, celulares inteligentes y computadores de bolsillo, la POO se ha vuelto aún más útil y necesaria, ya que por obvias razones este tipo de dispositivos son enteramente visuales y funcionan también mediante la respuesta a eventos generados por el usuario. Hoy en día, todo programador o estudiante de programación tiene la necesidad cada vez más urgente de familiarizarse con lenguajes de programación orientados al

desarrollo de aplicaciones móviles así como de aplicaciones web, ya que las tecnologías web toman también cada vez mayor auge debido a su gran versatilidad, universalidad e independencia de plataformas específicas de software. Por supuesto, todos estos lenguajes de programación son orientados a objetos por lo que la formación de todo programador debe iniciar por el entendimiento y dominio de los conceptos relacionados con la POO.

3.1.2.1. Pilares fundamentales de la Programación Orientada a Objetos

La POO se basa en cuatro pilares fundamentales que son abstracción, herencia, polimorfismo y encapsulamiento, que se procede a describir a continuación.

3.1.2.1.1. Abstracción, clases y objetos

El concepto angular dentro de la POO es, muy probablemente, el de clase, ya que de él en sí parte la POO. Una clase se puede conceptualizar como la definición de un tipo de entidad específica del mundo real. Si bien es algo complicado dar un concepto específico de clase, Gallardo [15] lo define como *"una plantilla donde se define un conjunto de campos (también llamados en POO variables de instancia) y un conjunto de métodos (funciones) que podrán ser ejecutadas por los objetos de esa clase"*.

Para entenderlo mejor, resulta muy útil recurrir a un ejemplo de la vida real, como lo es un automóvil. Los campos o variables de instancia de un automóvil podrían ser su color, marca, año de fabricación, cilindraje del motor, tipo de automóvil, localización, velocidad y otros datos. Sin embargo, si se habla de automóviles en general, no se puede establecer ninguno de esos datos, simplemente se sabe que los mismos forman parte de y pertenecen a los automóviles. Eso es precisamente lo que hace una clase: establecer qué datos definirán la existencia de una entidad de la vida real. En cuanto a los métodos o funciones de una clase, se puede decir que son éstos quienes definen la forma de trabajar con sus datos y las relaciones entre ellos.

El proceso de analizar un elemento o fenómeno de la realidad y llegar a representarlo dentro de la POO como una clase con todos los datos que resulten interesantes según el problema que se desee resolver, así como los métodos requeridos para manipular tales datos, es el proceso que se conoce como abstracción.

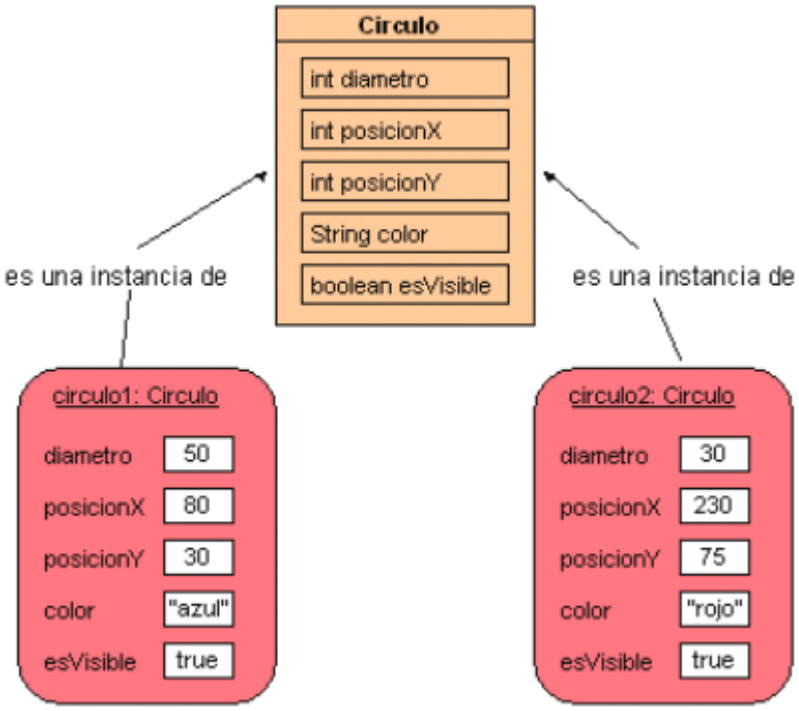
Un objeto, por otro lado, se considera una instancia específica de una clase. Volviendo al ejemplo de los automóviles, si se proporcionan valores específicos para cada campo, se puede decir que se está hablando de un automóvil en particular, y se puede definir tantos automóviles como sea necesario, cada uno con su conjunto específico de datos. Por ejemplo, el automóvil llamado auto1 puede ser de color celeste, marca Chevrolet, año 2011, motor de 1000cc e ir a 50Km/h de velocidad, el automóvil auto2 puede ser rojo, marca Nissan, año 2005, motor de 1600cc e ir a 0Km/h (estar estacionado), y un tercer automóvil llamado auto3 puede ser color habano, marca Toyota, año 2008, motor de 1800cc e ir a 30Km/h. Dentro del modelo de la POO, cada uno de estos autos representaría un objeto de la clase Automóvil.

A decir de Durán, Gutiérrez y Pimentel [16], *"Los conceptos de clase y objeto son los más importantes de la programación orientada a objetos. Un objeto es cualquier cosa tangible o intangible que se pueda imaginar, definida frente al exterior mediante unos atributos y las operaciones que permiten modificar dichos atributos"*, y además manifiestan que *"Cada objeto particular se obtiene como una especificación de una entidad más general denominada clase"*. Esta acción de crear un objeto específico basándose en la definición contenida en una clase se conoce dentro de la POO como instanciación.

Es muy importante tener en cuenta que cada campo de una clase corresponde a un tipo de dato específico que puede ser cadena de texto o alfanumérica, valor numérico entero o fraccionario, valor booleano (verdadero o falso), fecha y/u hora y otros. Además, al conjunto de valores que se encuentran asignados en un determinado momento a los campos de un objeto se le denomina estado del objeto. Si cualquiera de esos valores cambia, se dice que el estado en general del objeto ha cambiado.

La siguiente figura representa, mediante otro ejemplo práctico, la conceptualización de clases y objetos y la relación entre ambos conceptos.

Figura 4: Relación entre clases y objetos



Fuente: Lo esencial de la gestión documental: sistemas integrales de gestión de los documentos electrónicos ([14])

Toda vez que se tienen claros y se dominan los conceptos de clase y objeto, conviene adentrarse en otros conceptos más avanzados pero no menos importantes dentro de la POO, puesto que es muy necesario dominarlos para poder desarrollar exitosamente una aplicación basada en lenguajes orientados a objetos.

3.1.2.1.2. Constructores

Como se menciona en párrafos anteriores, las clases no definen únicamente los campos o valores que van a tener sus correspondientes objetos, sino que además implementan métodos que permiten manipular dichos valores, es decir, cambiar el estado de un objeto. Estos métodos evitan que los campos de un objeto sean manipulados directamente, siendo necesario invocar a dichos métodos con

un envío opcional de parámetros, y son estos métodos los que se encargan de establecer los nuevos valores en los campos de un objeto, siguiendo determinadas reglas.

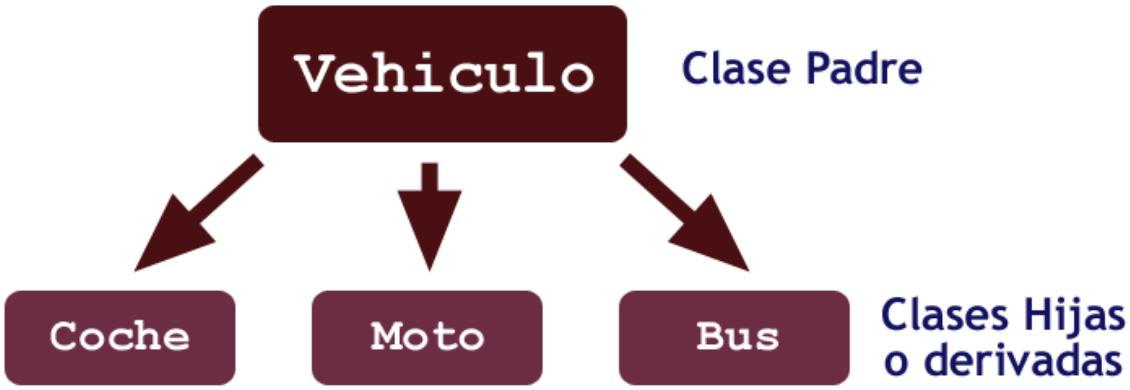
Uno o varios de estos métodos son especiales y reciben el nombre de constructores, y su finalidad es la de inicializar los valores de los campos de un objeto cuando este es creado. Un constructor puede opcionalmente programarse para recibir o no parámetros, que generalmente serán los valores con los que se desea inicializar el estado del objeto. En los casos de constructores que no reciben parámetros, se deberá disponer de una serie de valores iniciales por defecto con los que se creará cada nuevo objeto correspondiente a una clase particular.

A este respecto, se concuerda íntegramente con lo que Barnes [17] comenta en su libro: *"Los constructores de una clase tienen un rol especial que cumplir: su responsabilidades poner cada objeto de esa clase en un estado adecuado para que pueda ser usado una vez que haya sido creado. Esta operación se denomina inicialización. El constructor inicializa el objeto en un estado razonable"*. La inicialización es, por tanto, un concepto muy importante que se debe dominar y tomar en cuenta a la hora de desarrollar aplicaciones usando lenguajes o plataformas orientadas a objetos.

3.1.2.1.3. Herencia

La herencia es otro concepto de suma importancia dentro de la POO, ya que básicamente fomenta la reutilización de código y por ende el ahorro de tiempo y esfuerzo de programación. Básicamente, se refiere a la capacidad de una clase de "derivarse" o "descender" de otra, adquiriendo todos sus campos y métodos, y pudiendo añadir los suyos propios. En otras palabras, se puede definir una clase base general y de ella heredar a otras clases más especializadas, que se conocen como clases hijas, y que incrementarían la funcionalidad de la clase padre original. Dentro de la POO, en muchos casos resulta útil crear una clase cuya única función es la de servir como base para la definición de otras clases, sin que sea posible instanciar la clase original. Este tipo de clases se conocen como "abstractas". Esto permite definir una sola vez todos los campos y métodos que son comunes a todas las clases hijas que se requiere crear, y posteriormente ir añadiendo o modificando los elementos necesarios que son específicos de cada una de ellas.

Figura 5: Representación de la herencia dentro de las clases



Fuente: Polimorfismo en Programación Orientada a Objetos ([18])

Así, en la ilustración se puede observar un ejemplo sencillo que representa la herencia entre clases, retomando una vez más el ejemplo basado en vehículos. Dentro de la clase padre (Vehículo) estarían definidos los campos que se menciona en los apartados anteriores (color, marca, año, velocidad, etc.) excepto el tipo de vehículo, ya que el mismo se utilizará como parámetro para la definición de las clases hijas (Coche, Moto, Bus). Cada una de ellas puede, a su vez, incrementar los campos y métodos definidos en la clase padre, así, la clase bus podría incluir nuevos campos tales como número de pasajeros o cooperativa, los mismos que no tendrían sentido en las otros dos clases hijas.

3.1.2.1.4. Polimorfismo

El polimorfismo es otro de los conceptos de fundamental importancia dentro del desarrollo de aplicaciones usando lenguajes orientados a objetos, y es un concepto íntimamente ligado al de herencia, como lo expone el autor Barnes [17]. El polimorfismo es un concepto en realidad bastante clásico dentro de la programación en general y no es completamente exclusivo de los lenguajes orientados a objetos, aunque básicamente es aplicable a lenguajes fuertemente tipados como Java o C#.

Es por tanto importante tener en claro la diferencia entre lenguajes fuertemente tipados o de tipado estático y débilmente tipados o de tipado dinámico. Se refiere básicamente a la flexibilidad que presentan los lenguajes de programación a la hora de declarar y usar variables. Mientras los lenguajes fuertemente tipados obligan a declarar el tipo de dato que contendrá una variable dentro de un determinado contexto, y únicamente permiten que ese tipo de dato sea asignado a dicha variable dentro del contexto donde ésta existe, los lenguajes débilmente tipados como PHP o Javascript son mucho más flexibles en ese sentido y permiten declarar variables "generales" que pueden admitir diferentes tipos de datos dentro del mismo contexto. En este sentido, Barnes [17] considera que *"Denominamos tipo estático al tipo declarado de una variable porque la variable se declara en el código fuente, la representación estática del programa"* mientras que *"Denominamos tipo dinámico al tipo del objeto almacenado en una variable porque depende de su asignación en tiempo de ejecución, el comportamiento dinámico del programa"*.

Así, en el primer caso a una variable que es declarada como entera se le podrá únicamente asignar valores numéricos enteros durante su existencia dentro de un contexto, puesto que si le trata de asignar otro tipo de valor, como una cadena, el compilador arrojará un error y no permitirá la ejecución del programa. En el segundo caso, por otro lado, una variable se declara de manera general y se le pueden asignar valores de diferentes tipos dependiendo de las necesidades, sin que esto represente un error para el compilador o que no se pueda ejecutar el programa.

En los lenguajes fuertemente tipados, esta política se aplica no solo a los tipos de datos simples sino también a los objetos. Es decir, si se crea una clase y posteriormente se define una variable de dicha clase para instanciarla, los objetos que se asignen a dicha variable deben ser forzosamente de esa clase, puesto que si se trata de usar la variable para instanciar otra clase se tendrá un error como el que se describe antes. Aquí es cuando juega un papel muy importante el polimorfismo, ya que esta propiedad de las clases permite declarar una variable de una clase para instanciar no solamente esa clase sino también sus clases derivadas (hijas, nietas y demás). A este respecto, el mismo autor Barnes [17] manifiesta que *"El término «polimórfico» (literalmente: muchas formas) se refiere al hecho de que una misma variable puede contener objetos de diferentes tipos (del tipo declarado o de cualquier subtipo del tipo declarado). El polimorfismo aparece en los lenguajes orientados a objetos en numerosos contextos, las variables polimórficas constituyen justamente un primer ejemplo"*.

Por tanto, se puede definir el polimorfismo como una propiedad de las clases en los lenguajes orientados a objetos que permite instanciar en una misma variable tanto la clase original para la que

es declarada así como sus clases derivadas. Si bien en un principio puede parecer una propiedad de poca importancia, en realidad es una gran ayuda en muchos casos dentro de la POO, sobre todo en el sentido de que puede ahorrar mucho tiempo y esfuerzo de programación al evitar tener que repetir código con ligeras modificaciones para una clase y cada una de sus clases derivadas, sobre todo en el caso de clases avanzadas que representar objetos complejos del mundo real. Por otro lado, de la definición anterior se observa claramente la relación entre polimorfismo y herencia, ya que la segunda permite la existencia de clases derivadas (clases padre, hija, nieta y demás), que serían la base para la existencia y aplicación del polimorfismo.

3.1.2.1.5. Encapsulamiento

Otra de las propiedades características de las clases dentro de la POO es el encapsulamiento, que desde luego es también necesario comprender y dominar para poder programar exitosamente usando lenguajes orientados a objetos. Básicamente, el encapsulamiento se refiere a los niveles de acceso que van a tener los atributos (campos) y los métodos (funciones) de una clase determinada. El nivel de acceso, a su vez, se refiere básicamente a quién va a tener acceso a los elementos de la clase y desde dónde.

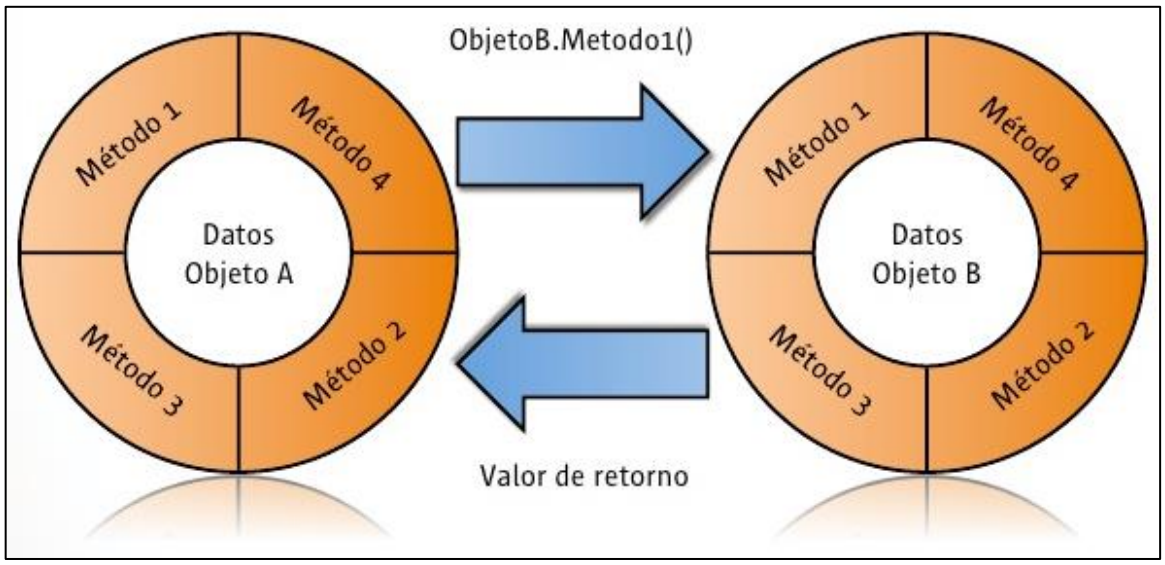
Dentro de los lenguajes orientados a objetos de mayor uso en la actualidad como son C# o Java, se suelen manejar tres niveles de acceso: público, protegido y privado. El nivel público significa que desde cualquier parte del código se podrá acceder a los atributos o métodos de un objeto en particular declarados como públicos, inclusive desde fuera de la clase. El nivel privado, por otro lado, determina que únicamente se podrá acceder a un atributo o método declarado como tal desde el código perteneciente a la misma clase, es decir, desde los métodos de la misma. Por tanto, desde el código fuera de la clase, dichos atributos y métodos serán inaccesibles. Por último, le nivel protegido viene a ser un híbrido entre los dos anteriores, ya que permite el acceso a atributos y métodos de una clase únicamente desde su propio código y desde el código de sus clases derivadas. De esto se puede también deducir que el encapsulamiento es otra propiedad que guarda una íntima relación con la herencia.

Cabe mencionar que la funcionalidad principal del encapsulamiento es justamente proteger el contenido de una clase en particular y, por ende, de sus objetos declarados. Es decir, es una propiedad netamente relacionada con la seguridad y la privacidad. Como principio fundamental dentro de la POO, la recomendación es, al programar una clase, hacer visible y accesible desde el exterior únicamente lo que es estrictamente necesario, y lo demás dejarlo como privado, protegido e invisible. Esto permite

forzar al usuario de una clase particular a usar únicamente los métodos creados como públicos para poder interactuar con los atributos de dicha clase, evitando así la manipulación directa por parte del usuario, lo que podría llevar a un estado inconsistente.

En tal sentido, el siguiente gráfico ilustra el funcionamiento del encapsulamiento haciendo públicos únicamente los métodos de una clase pero protegiendo sus atributos.

Figura 6: Representación del encapsulamiento en POO



Fuente: Programación Orientada a Objetos ([19])

3.1.2.2. Modularidad

Más que una propiedad de las clases en sí, la modularidad puede considerarse una característica de los lenguajes orientados a objetos. Básicamente consiste en la capacidad que ofrecen estos para desarrollar una aplicación por módulos, es decir en partes más pequeñas que son independientes pero sin embargo se relacionan estrechamente entre sí. Esta característica es muy útil sobre todo cuando se trata de aplicaciones excesivamente grandes o complejas, ya que permite dividirlas en partes que resultan más manejables, comprensibles y fáciles de documentar y mantener.

La modularidad permite concentrarse en un problema o aspecto a la vez, lo que a su vez permite resolverlo de manera más rápida, óptima y sin dejar pasar detalles importantes que pueden ser

fácilmente olvidados cuando se trata de programar con toda una gran aplicación en mente de una sola vez. Una vez que los módulos están listos, se los integra y se forma la aplicación que resuelve de manera integral el gran problema para el que fue originalmente creada, de la mejor manera posible. Otra gran ventaja que brinda la modularidad es la posibilidad de reutilizar el código, ya que un módulo que fue programado para un fin determinado en una aplicación puede ser fácilmente adaptado a otra, para lo cual se suelen requerir apenas pequeños cambios y adaptaciones a la nueva realidad que se considera. Sin embargo, para lograr esto es necesario programar los módulos con una visión de generalidad y considerando la posibilidad de reutilización posterior de los mismos.

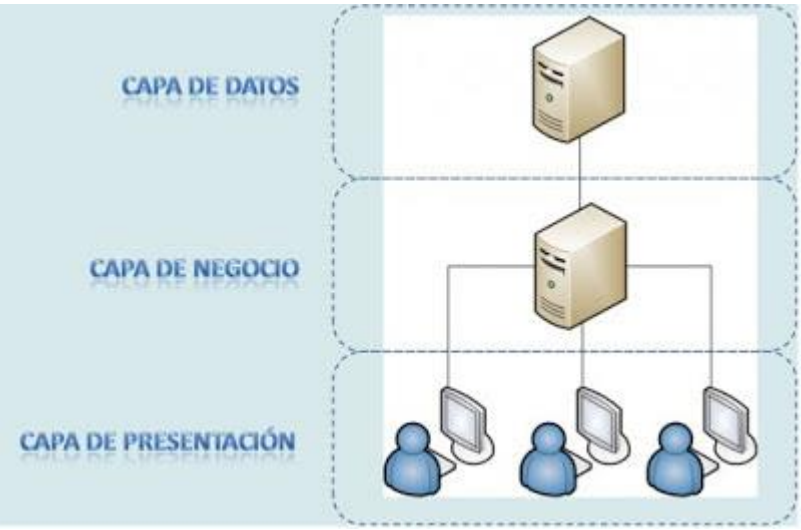
3.1.2.3. Programación en capas

Una de las mayores ventajas que ofrecen las plataformas de desarrollo de software orientadas a objetos más utilizadas en la actualidad en relación a la modularidad es lo que se conoce como programación en capas. Vargas y Maltés [20] la definen como *"una técnica de ingeniería de software propia de la programación por objetos, éstos se organizan principalmente en 3 capas: la capa de presentación o frontera, la capa de lógica de negocio o control, y la capa de datos"*. En esta definición se observa claramente que la programación por capas es un estándar bien definido que ayuda a dividir el código de una aplicación en tres módulos. Estos módulos son completamente independientes entre sí, sin embargo, para que la aplicación pueda funcionar como tal requiere que dichos módulos se interrelacionen de modo que uno pueda usar las funcionalidades implementadas por otro sin ningún inconveniente.

Cada lenguaje o plataforma de desarrollo ofrece sus propias facilidades para aplicar la programación por capas. Por ejemplo, la plataforma Microsoft Visual Studio, cuyos lenguajes de programación orientados a objetos por excelencia son C# y Visual Basic, permite crear varias aplicaciones dentro de una sola solución, pudiendo cada una de éstas dedicarse exclusivamente a cada una de las tareas que se menciona en la definición anterior. Posteriormente, cada proyecto puede acceder a la funcionalidad implementada en los otros cuando lo requiera, así, si el proyecto de presentación necesita desplegar un dato que está guardado en una base de datos, tendrá que recurrir al proyecto de lógica de negocio, el cual a su vez tendrá que recurrir al proyecto de datos para recuperar el dato requerido desde la base de datos. Cada proyecto dentro de la solución representa una capa dentro del modelo de desarrollo en capas.

La siguiente figura ilustra claramente la filosofía de la programación por capas, permitiendo ver la independencia y a su vez la relación entre las mismas.

Figura 7: Programación por capas



Fuente: Programación por capas ([21])

3.1.2.4. Recolección de basura

En el mundo de los lenguajes orientados a objetos, cuando se define una clase y posteriormente se declara una variable de esa clase para instanciarla, es decir, para crear un objeto de la misma, lo que en realidad contiene esta variable es una referencia al objeto en cuestión. Éste a su vez, no es otra cosa sino un espacio en la memoria del computador donde se guardan todos los atributos del mismo, así como sus métodos y/o funciones. Tantos objetos se creen de una clase determinada, tantos espacios en memoria de idénticas características se reservarán para contenerlos. La variable que se utiliza para acceder a la información de un objeto es en realidad un puntero, es decir que contiene la dirección de memoria donde se guarda toda la información correspondiente al objeto y, de hecho, es la única manera de acceder a dicha información.

Por muchas razones técnicas puede darse el caso de que un objeto se quede sin la variable referencia al mismo, con lo cual se volvería completamente inaccesible y por lo tanto, de ninguna utilidad dentro de la aplicación en desarrollo. Sin embargo, el espacio en memoria se mantiene

ocupado con la información correspondiente al objeto independientemente de que exista una referencia al mismo o no, lo cual constituye un desperdicio de recursos que, si por ejemplo son miles de objetos conteniendo muchos atributos y métodos cada uno, puede resultar bastante considerable e incluso producir un impacto negativo notable en el desempeño de la aplicación en general.

Para resolver este inconveniente, las plataformas de desarrollo de software orientadas a objetos implementan un mecanismo llamado "recolección de basura", el mismo que consiste en detectar estos objetos "perdidos" en memoria, objetos que se encuentran ocupando espacio innecesariamente ya que no existe referencia alguna a ninguno de ellos, y eliminarlos, recuperando así la memoria mal utilizada lo cual a su vez devuelve agilidad y ligereza al sistema en general.

El concepto de recolección de basura nace en el año de 1958 y es concebido por John McCarthy [22] dentro del lenguaje de programación Lisp, para evitar la gestión manual de la memoria que, hasta ese momento, debía hacerse por parte del programador quien se encargaba de liberar manualmente los espacios de memoria reservados en su programa. Por su puesto, esto abría la posibilidad de cometer muchos errores humanos que a su vez conducían a problemas tales como la sobresaturación y fragmentación de la memoria de trabajo del programa. Como se observa, la recolección de basura no es propiedad exclusiva de los lenguajes orientados a objetos, sino que el concepto ya existía claramente antes del surgimiento de dichos lenguajes.

La técnica de la recolección automática de basura introducida por McCarthy [22] resulta, por tanto, sumamente beneficiosa tanto para el programador, ya que no debe preocuparse por implementar esta funcionalidad en sus programas, así como para el sistema y sobre todo su memoria, que se mantiene libre, ordenada y sin almacenar objetos que no tienen razón de ser.

3.1.3. Ingeniería de software.

Hoy en día, casi todas las actividades del ser humano se encuentran automatizadas e informatizadas en mayor o menor medida. Gracias a los avances tecnológicos, toda institución ya sea pública, privada, comercial, industrial, bancaria, gubernamental o de cualquier otro índole, ha logrado automatizar sus procesos clave, logrando así agilidad y seguridad en los mismos. Esta automatización, a su vez, requiere de la instalación y puesta en marcha de una infraestructura informática cuyo tamaño y complejidad estará acorde a las necesidades específicas de cada institución, incluyendo infraestructuras de red, telecomunicaciones, servidores, impresoras y demás.

Todo este conjunto de equipos informáticos y de telecomunicaciones, sin embargo, requiere de un componente fundamental para su funcionamiento: el software. Sin software, cualquier equipo informático, por muy sofisticado que sea, resulta de muy poca o ninguna utilidad, ya que en realidad no podría hacer prácticamente nada. Se puede decir, por consiguiente, que el software es como el alma del hardware; es el componente que le da vida.

Por otro lado, las empresas y todo tipo de instituciones poseen requerimientos cambiantes, que van evolucionando a través del tiempo, lo cual ha obligado que el hardware vaya también evolucionando para responder a tales necesidades. Por este motivo, no es raro observar equipos cada vez más potentes que, sin embargo, en poco tiempo se vuelven obsoletos para dar lugar a una nueva generación de equipos más avanzados todavía. De lo expuesto en los párrafos anteriores se puede deducir fácilmente que todo este proceso de evolución del hardware lleva consigo forzosamente una evolución paralela del software, ya que de lo contrario de muy poco servirían los avances tecnológicos. De allí que, al igual que sucede con el hardware, cada vez aparecen sistemas operativos más potentes, paquetes de programas más acordes a la realidad y al surgimiento de nuevas necesidades, sin descuidar aspectos tan críticos como la seguridad.

Es muy común, además, que la gran mayoría de instituciones requieran de paquetes de software especializados y que respondan oportunamente a sus necesidades particulares, por lo que dichas instituciones deben buscar la forma de elaborar el software que requieren, y de almacenar y manejar de forma ágil y segura sus datos críticos. A todo este proceso de diseño y elaboración de un software se le conoce como ingeniería de software, el mismo que abarca desde las etapas iniciales de análisis de requerimientos hasta las etapas de mantenimiento del software, pasando por la codificación y las pruebas que garantizan que el software que se está construyendo verdaderamente responde de manera efectiva a las necesidades que se pretende satisfacer.

A decir de Sommerville [23], *"La ingeniería del software es una disciplina de la ingeniería cuya meta es el desarrollo costeable de sistemas de software"*. Al decir que es una disciplina de la ingeniería, se puede comparar el proceso de construcción de un software con el de construcción de un automóvil, de un edificio o de un puente, con la gran diferencia de que el software no está sujeto a muchas de las limitaciones, sobre todo de carácter físico, a las que están sujetos los otros ítems mencionados, como el peso o el tamaño, por obvias razones. Esto a su vez conlleva a que el software puede resultar muy intrincado y por tanto difícil de mantener y modificar. Por otro lado, el concepto de Sommerville menciona el término "costeable" que es muy importante dentro del desarrollo de software, ya que el

costo es un factor clave que influye en todo este proceso, tanto desde el punto de vista del cliente como del equipo que lo desarrolla, y determina la duración y alcance total de un proyecto.

El origen del concepto de ingeniería de software se remonta al año de 1968 cuando se dio el surgimiento de los primeros computadores basados en circuitos integrados (microchips), lo cual abrió la posibilidad de crear programas y aplicaciones hasta entonces inimaginables. Esto a su vez dio origen a la llamada "crisis del software", ya que los programas existentes hasta tal fecha, diseñados para la tecnología anterior, no aprovechaban en nada las bondades ni potencia que vino con la nueva tecnología. Hasta ese momento, las aplicaciones se diseñaban y desarrollaban de manera empírica e informal, sin seguir una metodología específica para crearlas, sin embargo resultó claro que esta modalidad ya no iba a funcionar con la nueva tecnología disponible, ya que la misma iba a demandar la construcción de aplicaciones mucho más complejas y diseñadas para resolver problemas de mucha mayor envergadura. Precisamente este fenómeno fue lo que se denominó "crisis del software", y la ingeniería de software surgió como una respuesta para tratar de sistematizar la creación de nuevas aplicaciones, pudiendo así responder a los nuevos requerimientos de manera efectiva.

Desde entonces, la ingeniería de software ha ido evolucionando y desarrollándose sostenidamente hasta la actualidad, guiada por este constante proceso de avance tecnológico y, por otro lado, de surgimiento de nuevas necesidades para las instituciones de toda índole. En la actualidad, la ingeniería de software resulta más útil y necesaria que nunca, ya que tanto el número de instituciones existentes en la actualidad a nivel mundial así como sus necesidades individuales se han incrementado geométricamente.

3.1.3.1. El proceso del software

Dentro del mundo de la ingeniería de software, algunos de los más reconocidos autores como son Sommerville [23] y Pressman [25] hacen referencia al proceso del software, un concepto que resulta fundamental para comprender el fin último que persigue la ingeniería de software. Para Sommerville, *"Un proceso del software es un conjunto de actividades que conducen a la creación de un producto software. Estas actividades pueden consistir en el desarrollo de software desde cero en un lenguaje de programación estándar como Java o C. Sin embargo, cada vez más, se desarrolla nuevo software ampliando y modificando los sistemas existentes y configurando o integrando software comercial o componentes del sistema"*. El mismo autor menciona que, si bien los procesos de software varían de

acuerdo a la naturaleza y características de cada proyecto, existen algunas actividades que son comunes para todos, como son:

- Especificación del software
- Diseño e implementación del software
- Validación del software
- Evolución del software

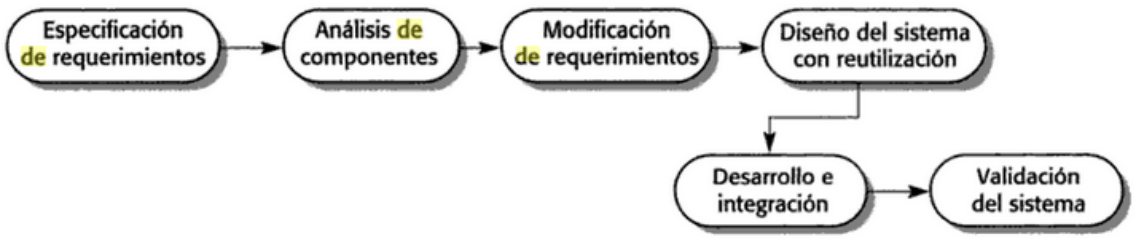
Igualmente, este autor habla de tres modelos básicos dentro del proceso del software, los mismos que se encuentran íntimamente ligados a las actividades listadas. Dichos modelos son:

Modelo en cascada: considera que cada una de las actividades debe desarrollarse secuencialmente una sola vez, por lo que hay que realizarlas a profundidad, ya que no se repetirán.

Modelo evolutivo: entrelaza las mencionadas actividades de modo que se comienza desarrollando un software con visión general, cumpliendo con todas ellas, y luego se va refinando el mismo hasta lograr todas las características y funcionalidad requeridas, para lo cual habrá que repetir la secuencia de actividades las veces que sea necesario.

Modelo basado en componentes: su filosofía se basa en la preexistencia de componentes de software reutilizables, que ya han pasado debidamente por todas las actividades mencionadas, y que simplemente deben ser integrados y configurados correctamente, evitándose así la necesidad de crearlo todo desde cero para cada proyecto. Si bien este modelo tiene un gran potencial de reducción de costos y tiempos de entrega, al ser componentes de software preexistentes pueden dar lugar a la creación de un producto software que no cumpla cabalmente con los requerimientos planteados en un inicio, por lo cual los ingenieros de software y desarrolladores deben ser especialmente cuidadosos al basarse en este modelo. A continuación, una representación gráfica de este modelo:

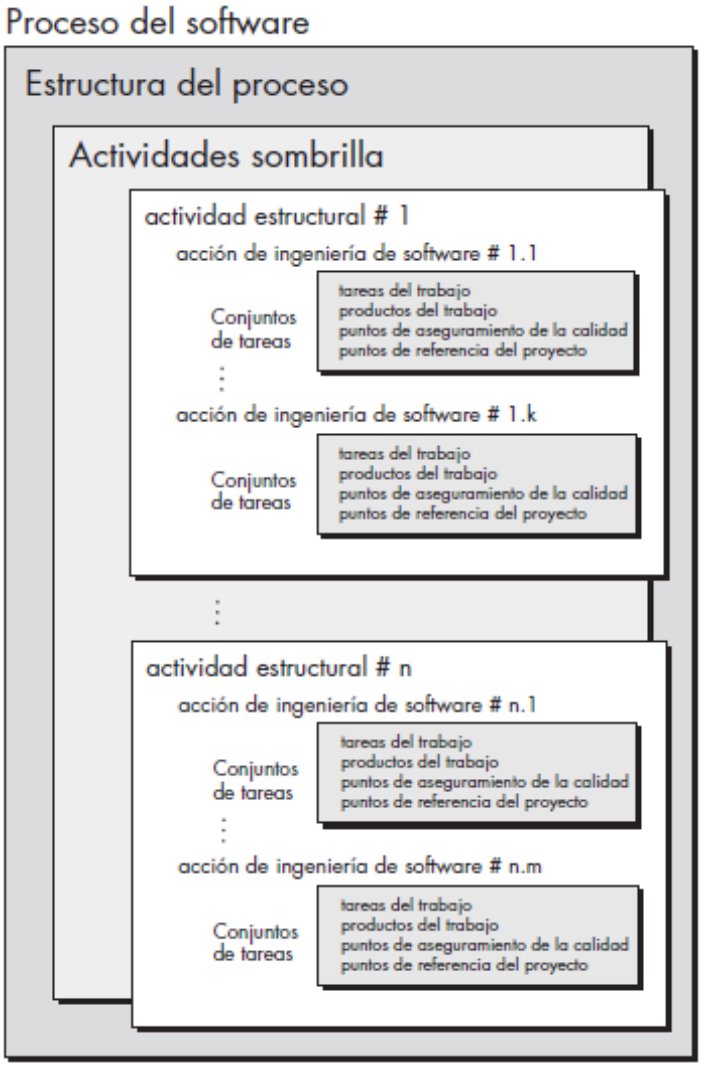
Figura 8: Ingeniería del software basada en componentes



Fuente: Bases de datos ([24])

El autor Pressman [25], por su lado, manifiesta que "*Cuando se trabaja en la construcción de un producto o sistema, es importante ejecutar una serie de pasos predecibles —el mapa de carreteras que lo ayuda a obtener a tiempo un resultado de alta calidad—. El mapa que se sigue se llama "proceso del software""*. Y, al igual que el autor anterior, define una serie de actividades estructurales generales para la gran mayoría de procesos de software como son: comunicación, planeación, modelado, construcción y despliegue. Además, todas estas actividades principales se encuentran entrelazadas entre sí mediante actividades más pequeñas, pero no menos importantes, conocida actividades de sombrilla que incluyen seguimiento y control del proyecto, aseguramiento de la calidad, revisiones técnicas y otras. La siguiente ilustración lo explica de mejor manera:

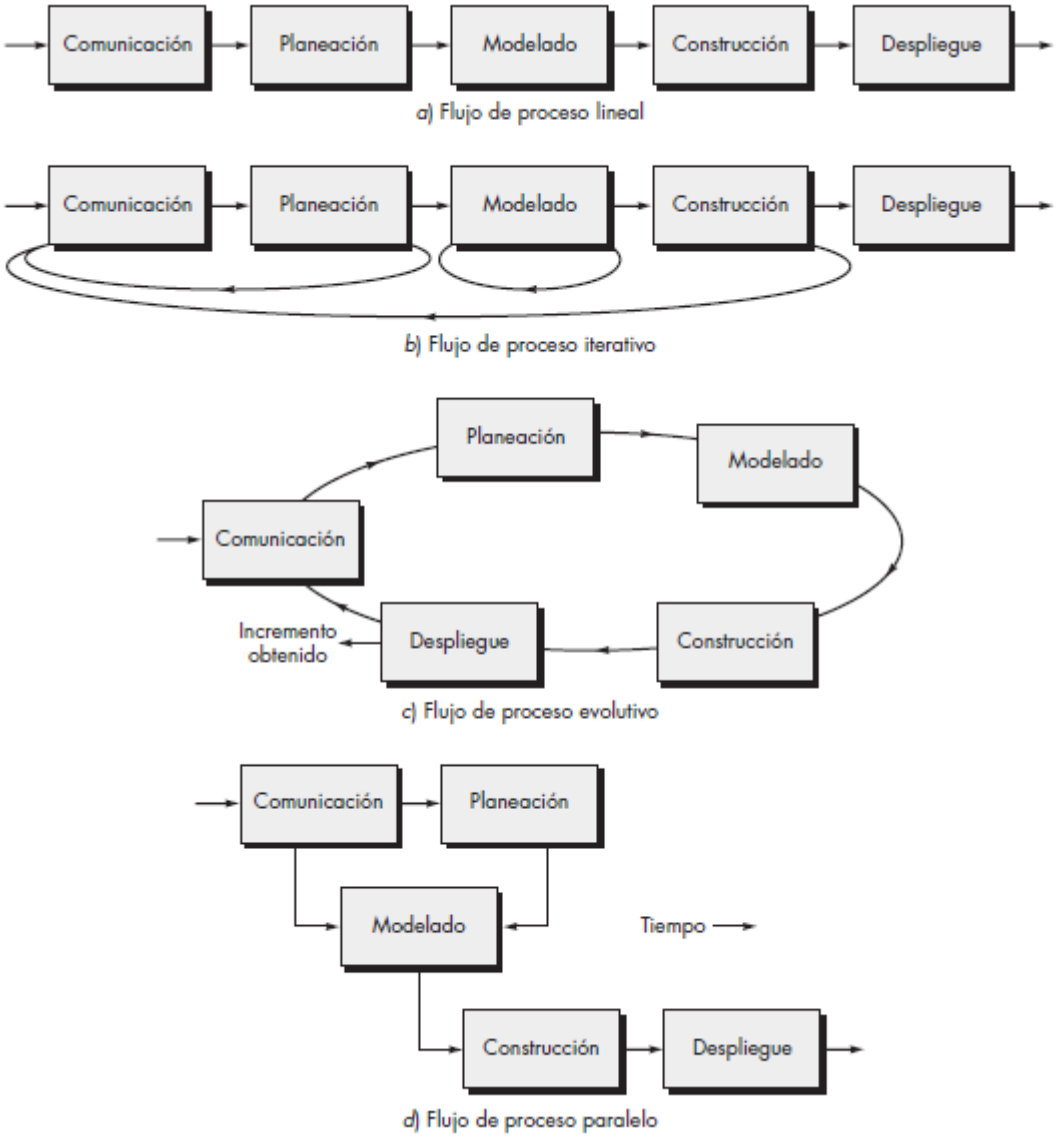
Figura 9: Proceso del software



Fuente: Ingeniería del software ([25])

Para Pressman, existen cuatro modalidades de flujo de proceso, basados en las actividades generales listadas, como se observa en el siguiente gráfico:

Figura 10: Flujo del proceso del software



Fuente: Ingeniería del software ([25])

Como se puede observar, existe una considerable correspondencia entre los modelos expuestos por ambos autores analizados, si bien algunos términos y premisas varían. Sin embargo, al ser Sommerville un autor con una visión mucho más moderna de la ingeniería de software, ya que su publicación tiene aproximadamente 20 años de diferencia con la del autor Pressman, es lógico que el primero haga referencia al modelo basado en componentes, concepto que aún no se encuentra en auge cuando Pressman publica su obra. Hoy en día se puede observar que la programación basada en componentes va tomando cada vez mayor impulso dentro del mundo del desarrollo de software, por lo que se coincide mayormente con la visión del autor Sommerville, sin dejar de lado las importantes bases establecidas por Pressman.

3.1.3.2. Metodologías de desarrollo de software

Desde el surgimiento de la ingeniería de software hasta la actualidad, han ido apareciendo dentro de ésta diferentes metodologías de desarrollo de software, cada una diseñada para responder a las necesidades generales de diseño e implementación de productos software existentes en su respectivo momento histórico. Conforme tales necesidades han ido cambiando, y en consecuencia ha ido evolucionando la ingeniería de software, ofreciendo cada vez nuevas o renovadas modalidades de flujos de actividades dentro del proceso del software, dichas innovaciones han ido traduciéndose en diferentes metodologías de desarrollo de software, que no son otra cosa que intentos por sistematizar las etapas a seguir para desarrollar un software de calidad, optimizando costos y tiempos de desarrollo, a la vez que cumpliendo cabalmente con todos los requerimientos del cliente, en la mayor medida en la que esto sea posible.

Si bien resulta complicado dar una definición exacta de lo que es una metodología de desarrollo de software, Avison y Fitzgerald [26] en el año 2003 manifiestan que *"Una metodología es una colección de procedimientos, técnicas, herramientas y documentos auxiliares que ayudan a los desarrolladores de software en sus esfuerzos por implementar nuevos sistemas de información. Una metodología está formada por fases, cada una de las cuales se puede dividir en sub-fases, que guiarán a los desarrolladores de sistemas a elegir las técnicas más apropiadas en cada momento del proyecto y también a planificarlo, gestionarlo, controlarlo y evaluarlo"*. Resulta obvio, sin embargo, que estas fases y sub – fases, y su secuencia de aplicación dependerán en gran medida de la naturaleza y los requerimientos específicos de cada proyecto.

Al decir que la historia de las metodologías de desarrollo de software va de la mano con la de la ingeniería de software, se puede deducir que las primeras metodologías en este sentido aparecieron en la década de 1960, cuando los mayores requerimientos de software se concentraban básicamente en programas de carácter comercial y contable para empresas de gran tamaño en Estados Unidos. Entonces, aparecieron las primeras metodologías de desarrollo de software como SOL y Jackson, que fundamentalmente estaban relacionadas con la programación estructurada que, a su vez, fue la modalidad de programación que se encontraba en auge en ese momento.

Conforme avanzaron los años y los requerimientos de software fueron cambiando y extendiéndose a empresas de mediano y pequeño tamaño de diverso índole, e inclusive a individuos, las metodologías de desarrollo de software se vieron también forzadas a evolucionar, siempre con la finalidad de definir un camino apropiado para desarrollar software de calidad que cumpla de la mejor manera con los siempre cambiantes requerimientos. Así, por ejemplo, en la década de 1980 apareció la metodología *Rapid Application Development* (RAD), desarrollada por James Martin, la que a su vez se caracterizó por el uso de herramientas CASE, herramientas de ingeniería de software asistida por computadora.

Sin embargo, en la década de 1990 el desarrollo de software sufrió una revolución sin precedentes, guiado por un lado por la popularización de sistemas operativos de carácter visual, como es el caso de Microsoft Windows 95, lo cual a su vez dio origen a un nuevo paradigma de programación conocido como Programación Orientada a Objetos, del que se habla en secciones anteriores, lo cual introdujo forzosamente y para siempre conceptos como clase y objeto dentro de los lenguajes de programación y plataformas de desarrollo. Esto, a su vez, disparó y diversificó geométricamente los requerimientos de software a todo nivel y, sobre todo, provocó la necesidad de desarrollar software en cada vez menos tiempo, pero sin sacrificar su calidad. Así se comenzó a desplazar poco a poco las metodologías tradicionales de desarrollo, que resultaban ya demasiado lentas, en pro de metodologías que favorezcan la rapidez en el desarrollo.

Finalmente, en la década del 2000 surgen las llamadas metodologías ágiles de desarrollo de software, a raíz de la elaboración del documento conocido como el "Manifiesto Ágil", cuyo principal promotor fue Kent Beck, quien formó un equipo con algunos de los más reconocidos profesionales dentro del campo de la ingeniería y el desarrollo de software de Estados Unidos. Este documento establece los principios fundamentales para un desarrollo de software mucho más ágil y menos burocrático, y de él surgieron varias metodologías de desarrollo que, justamente, pasaron a conocerse como las metodologías ágiles de desarrollo de software. Las metodologías que se venían utilizando

hasta entonces pasaron a conocerse como “metodologías tradicionales” y fueron tachadas de demasiado pesadas y rígidas, por lo que prácticamente ya no se adaptaban a las nuevas necesidades y requerimientos actuales de productos de software.

3.1.3.2.1. Metodologías ágiles vs. metodologías tradicionales

Como se menciona en párrafos anteriores, como resultado del surgimiento de diferentes metodologías de desarrollo de software a lo largo de décadas de evolución de la ingeniería de software se obtiene dos grandes grupos de metodologías: las “tradicionales”, que resultaban apropiadas cuando las exigencias del mercado de software no eran aún tan agresivas, y las “ágiles”, que surgen a partir del año 2001 como respuesta a un crecimiento nunca antes visto en la cantidad y naturaleza de los nuevos requerimientos del software y, sobre todo, a la exigencia de producir software de gran calidad en tiempos mucho más reducidos que antes.

Si bien es cierto las metodologías ágiles son las que por regla general mejor responden a los requerimientos actuales de desarrollo de software, esto no necesariamente significa que las metodologías tradicionales ya no se puedan utilizar o que hayan quedado en el olvido, ya que para ciertos tipos de proyectos de desarrollo de software, sobre todo de gran envergadura, puede resultar mejor recurrir a una metodología tradicional

Para entender de mejor manera las diferencias más marcadas existentes entre estos dos grupos de metodologías, es necesario primeramente comprender y analizar los diferentes modelos o enfoques en los que pueden estar basadas las metodologías de desarrollo de software. Si bien existen muy variados y diversos, a continuación se procede a describir los más conocidos y utilizados, de acuerdo a la definición de Tinoco [27].

Desarrollo secuencial. Es uno de los más antiguos y normalmente está relacionado con las metodologías tradicionales, de las cuales la más clásica es probablemente la de cascada. Las etapas del proceso del software, como son análisis, diseño, implementación y pruebas, se ejecutan secuencialmente, una sola vez cada una. Es por tanto necesario que una etapa esté completamente finalizada y documentada para poder pasar a la siguiente. Este tipo de desarrollo funcionaba muy bien cuando los proyectos de desarrollo de software eran bastante predecibles y era improbable que vayan a surgir cambios de consideración durante el desarrollo, peor aún durante las etapas finales del mismo, lo cual podría resultar catastrófico para el proyecto.

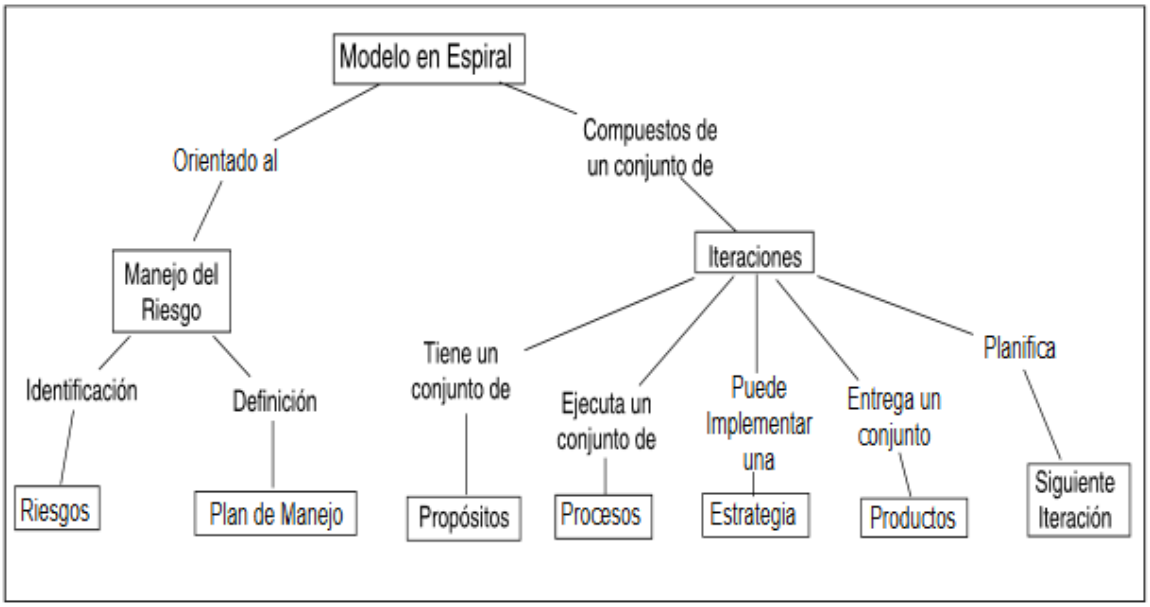
Desarrollo incremental. Ésta es ya una evolución del modelo anterior, cuya visión principal es la de dividir un proyecto complejo de desarrollo de software en partes más pequeñas y manejables, acelerándose así el tiempo de desarrollo total del proyecto y, a la vez, volviéndolo mucho más tolerable a cambios en los requerimientos. Si bien se realiza un análisis inicial de requerimientos, al irse desarrollando el proyecto en incrementos, que vienen a ser las partes en las que se divide el proyecto, se puede introducir cambios en el rumbo del desarrollo y la codificación con relativa facilidad, y las pruebas van dejando en evidencia la funcionalidad individual de cada iteración desarrollada.

Desarrollo iterativo. A decir de Castro [28], *"El desarrollo iterativo es un método de construcción de productos cuyo ciclo de vida está compuesto por un conjunto de iteraciones, las cuales tienen como objetivo entregar versiones del software. Cada iteración se considera un subproyecto que genera productos de software y no sólo documentación, permitiendo al usuario tener puntos de verificación y control más rápidos e induciendo un proceso continuo de pruebas y de integración desde las primeras iteraciones"*. Como se puede ver, existe una analogía bastante marcada con el enfoque incremental, y de hecho, se suele hablar comúnmente de un desarrollo incremental e iterativo, las iteraciones son repeticiones cada una de las cuales se encarga de desarrollar una de las partes en las que se divide el proyecto total, lo cual permite mantener un control mucho mejor y más detallado de las versiones del software que se van desarrollando y de los requerimientos que se van cumpliendo.

Desarrollo en espiral. Su principal objetivo es maximizar la tolerancia al riesgo y es el modelo que mejor soporta y maneja los cambios en los requerimientos que pudieran surgir durante el desarrollo del proyecto. Éste modelo es a su vez una evolución de los modelos anteriores e igualmente se basa en iteraciones, que en este caso también consisten en subdivisiones del proyecto total en proyectos más pequeños y fáciles de manejar y desarrollar. Sin embargo, esta modalidad introduce algunas innovaciones, como por ejemplo, en las primeras iteraciones se debe abordar las partes más riesgosas y críticas del proyecto para evaluar su viabilidad y así reducir drásticamente el riesgo de fracaso en etapas más avanzadas. Se introduce también el concepto de la participación de uno o más miembros designados por el cliente como parte del equipo de desarrollo del proyecto, quienes deberán participar activamente, sobre todo en la priorización de requisitos y en las pruebas.

En el siguiente gráfico, Boehm [29] describe a manera de mapa conceptual los principales preceptos y principios en los que se basa el modelo de desarrollo en espiral.

Figura 11: Mapa conceptual del modelo en espiral

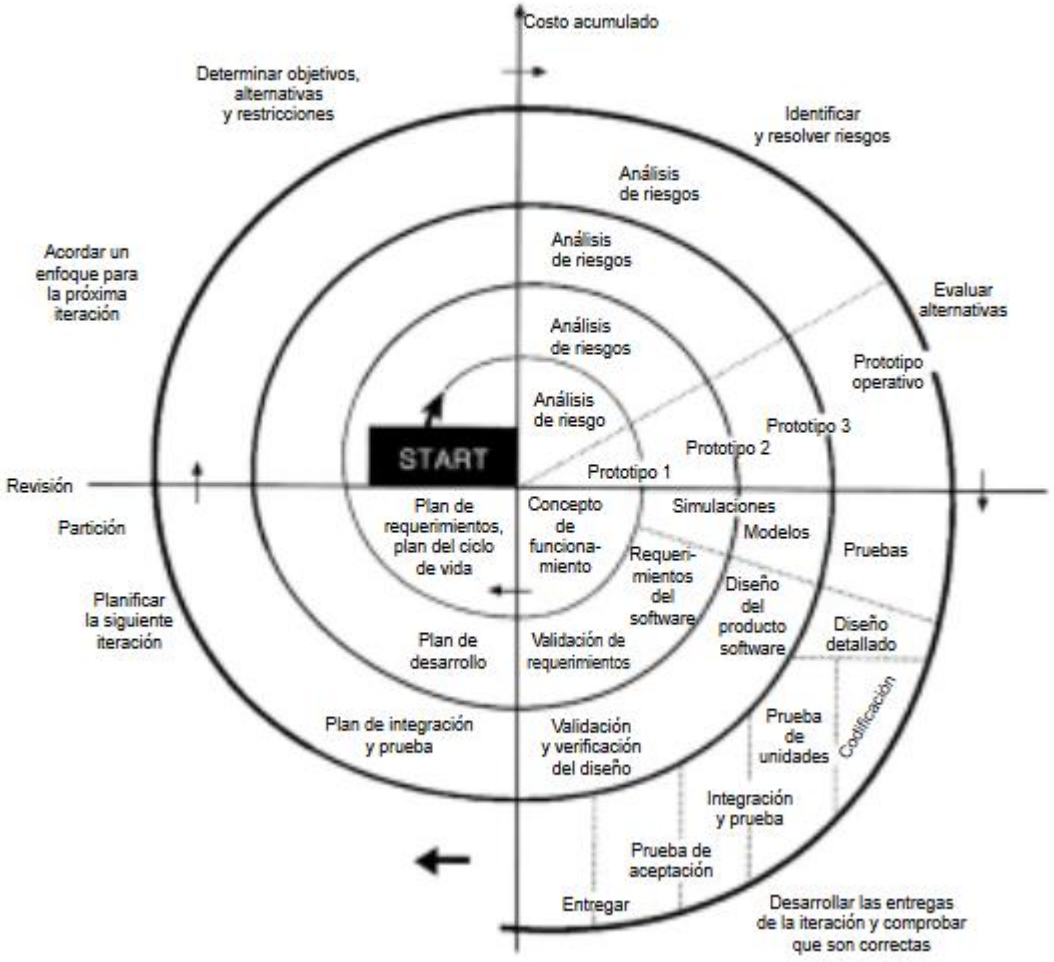


Fuente: A spiral model of software development and enhancement ([29])

El modelo en espiral es el que más y mejor se adapta a las metodologías ágiles de desarrollo de software, y resulta un modelo especialmente útil cuando se trata de crear productos de software completamente nuevos o de resolver situaciones cuyos requerimientos iniciales son demasiado vagos y podrían fácilmente variar durante el desarrollo. Al ser un modelo iterativo, permite ir realizando una nueva evaluación de requerimientos al inicio de cada iteración, lo cual facilita enormemente la terea de ir identificando las variaciones que se dan en los mismos.

Por último, cabe mencionar que uno de los puntos más fuertes del modelo en espiral es que permite ir verificando y asegurando la calidad en cada entrega (iteración), lo cual es la clave para eliminar el riesgo siempre presente en las metodologías tradicionales de desarrollar un sistema completo hasta sus fases finales para apenas ahí descubrirse que existen requerimientos iniciales mal planteados, lo que bien puede echar a perder todo el proyecto. La siguiente gráfica ilustra de manera bastante clara el desarrollo de las diferentes etapas dentro del modelo en espiral.

Figura 12: Representación gráfica del modelo en espiral



Fuente: Diseño e implementación de una herramienta de representación del conocimiento para apoyar la gestión de requisitos en un proceso de desarrollo de software ([30])

Una vez que se tienen en claro la filosofía de cada uno de los modelos descritos, se puede pasar a detallar las diferencias existentes entre las metodologías tradicionales y las ágiles. Como se menciona con anterioridad, las metodologías ágiles tienen su base en el manifiesto ágil, documento elaborado por un grupo de ingenieros de software liderados por Kent Beck en el año 2001 en Utah, donde se detallan los preceptos que deben seguir las nuevas metodologías de desarrollo de software para cumplir con la premisa de producir software de calidad en mucho menor tiempo.

El manifiesto ágil establece cuatro valores fundamentales en los que deben basarse las nuevas metodologías de desarrollo; dichos valores son traducidos por Canós [31] como:

- *Se valora al individuo y las interacciones del equipo de desarrollo sobre el proceso y las herramientas.*
- *Desarrollar software que funciona más que conseguir una buena documentación.*
- *La colaboración con el cliente más que la negociación de un contrato.*
- *Responder a los cambios más que seguir estrictamente un plan.*

Basados en dichos valores, proceden a elaborar los principios del manifiesto, que el mismo autor traduce así:

I. *La prioridad es satisfacer al cliente mediante tempranas y continuas entregas de software que le aporte un valor.*
II. *Dar la bienvenida a los cambios. Se capturan los cambios para que el cliente tenga una ventaja competitiva.*
III. *Entregar frecuentemente software que funcione desde un par de semanas a un par de meses, con el menor intervalo de tiempo posible entre entregas.*
IV. *La gente del negocio y los desarrolladores deben trabajar juntos a lo largo del proyecto.*
V. *Construir el proyecto en torno a individuos motivados. Darles el entorno y el apoyo que necesitan y confiar en ellos para conseguir finalizar el trabajo.*
VI. *El diálogo cara a cara es el método más eficiente y efectivo para comunicar información dentro de un equipo de desarrollo.*
VII. *El software que funciona es la medida principal de progreso.*
VIII. *Los procesos ágiles promueven un desarrollo sostenible. Los promotores, desarrolladores y usuarios deberían ser capaces de mantener una paz constante.*
IX. *La atención continua a la calidad técnica y al buen diseño mejora la agilidad.*
X. *La simplicidad es esencial.*
XI. *Las mejores arquitecturas, requisitos y diseños surgen de los equipos organizados por sí mismos.*
XII. *En intervalos regulares, el equipo reflexiona respecto a cómo llegar a ser más efectivo, y según esto ajusta su comportamiento.*

Considerando tales principios, se establece la siguiente tabla que denota las diferencias más básicas entre las metodologías tradicionales y ágiles de desarrollo de software:

Tabla 1: Diferencias entre las metodologías tradicionales y ágiles de desarrollo de software

Metodología Ágil	Metodología Tradicional
Pocos Artefactos. El modelado es prescindible, modelos desechables.	Más Artefactos. El modelado es esencial, mantenimiento de modelos
Pocos Roles, más genéricos y flexibles	Más Roles, más específicos
No existe un contrato tradicional, debe ser bastante flexible	Existe un contrato prefijado
Cliente es parte del equipo de desarrollo (además in-situ)	El cliente interactúa con el equipo de desarrollo mediante reuniones
Orientada a proyectos pequeños. Corta duración (o entregas frecuentes), equipos pequeños (< 10 integrantes) y trabajando en el mismo sitio	Aplicables a proyectos de cualquier tamaño, pero suelen ser especialmente efectivas/usadas en proyectos grandes y con equipos posiblemente dispersos
La arquitectura se va definiendo y mejorando a lo largo del proyecto	Se promueve que la arquitectura se defina tempranamente en el proyecto
Énfasis en los aspectos humanos: el individuo y el trabajo en equipo	Énfasis en la definición del proceso: roles, actividades y artefactos
Basadas en heurísticas provenientes de prácticas de producción de código	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo
Se esperan cambios durante el proyecto	Se espera que no ocurran cambios de gran impacto durante el proyecto

Fuente: Metodologías ágiles en el desarrollo de software ([31])

3.1.3.2.2. Metodología *Extreme Programming*

La metodología de desarrollo de software *Extreme Programming* (XP) es, probablemente, la más utilizada hoy en día dentro de las metodologías ágiles, por su gran versatilidad, efectividad y porque se adapta con relativa facilidad a la gran mayoría de proyectos de desarrollo de software. Si bien existen otras metodologías ágiles como SCRUM, *Crystal Methodologies, Dynamic Systems Development Method* (DSDM), *Adaptative Software Development* (ASD), *Feature-Driven Development* (FDD) y *Lean Development* (LD), entre otras, que también suelen ser bastante utilizadas para diferentes tipos de

proyectos, XP ha logrado consolidarse como una de las más reconocidas y generalmente utilizadas dentro de la ingeniería de software a nivel mundial. Y esto tiene bastante lógica ya que el padre y creador de XP es justamente Kent Beck, quien formó y lideró el grupo de ingenieros de software que crearon el manifiesto ágil.

La siguiente tabla muestra, en una escala del 1 al 5, una estimación de la agilidad correspondiente a las metodologías de mayor uso en la actualidad, en relación a diferentes aspectos dentro del desarrollo de software.

Tabla 2: Ranking de agilidad de las metodologías ágiles

	CMM	ASD	Crystal	DSDM	FDD	LD	Scrum	XP
Sistema como algo cambiante	1	5	4	3	3	4	5	5
Colaboración	2	5	5	4	4	4	5	5
Características Metodología (CM)								
-Resultados	2	5	5	4	4	4	5	5
-Simplicidad	1	4	4	3	5	3	5	5
-Adaptabilidad	2	5	5	3	3	4	4	3
-Excelencia técnica	4	3	3	4	4	4	3	4
-Prácticas de colaboración	2	5	5	4	3	3	4	5
Media CM	2.2	4.4	4.4	3.6	3.8	3.6	4.2	4.4
Media Total	1.7	4.8	4.5	3.6	3.6	3.9	4.7	4.8

Fuente: Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP) ([32])

Como se observa, una metodología puede resultar más apropiada que otra para un determinado proyecto de desarrollo de software, dependiendo de las características que más se requieran o sean útiles según la naturaleza del proyecto. Sin embargo, la metodología XP tiende a tener, en términos

generales, un ranking superior al de la mayoría de las otras metodologías analizadas, tanto en el análisis individual de características como en un análisis general. Se puede por tanto concluir que la metodología XP resultará bastante conveniente para prácticamente cualquier proyecto de desarrollo de software, dada su gran simplicidad y fomento de la colaboración entre los miembros del equipo de desarrollo.

En este sentido, Letelier [32] manifiesta que *"XP es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico"*. Se concuerda ampliamente con el mencionado autor, ya que esta definición deja una idea bastante concisa de lo que persigue la metodología XP y sus bases fundamentales.

Además de lo mencionado, es importante mencionar que esta metodología se basa en el enfoque de desarrollo en espiral el cual, a su vez, es un enfoque incremental e iterativo, como se analiza en párrafos anteriores. Esto le brinda especial facilidad a esta metodología para adaptarse a los cambios en los requisitos, sobre todo cuando estos no estuvieron especialmente bien planteados al inicio del proyecto, y para subdividir proyectos muy grandes en subproyectos más pequeños y fáciles de manejar.

3.1.3.2.3. Historias de usuario

Las historias de usuario son documentos con un determinado formato cuya intención es recopilar los requerimientos establecidos por el cliente o por la o las personas designadas por este para colaborar con el equipo de desarrollo. El mismo autor Letelier [32] explica acertadamente que *"Las historias de usuario son la técnica utilizada en XP para especificar los requisitos del software. Se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. El tratamiento de las historias de usuario es muy dinámico y flexible, en cualquier momento historias de usuario pueden romperse, reemplazarse por otras más específicas o generales, añadirse nuevas o ser modificadas. Cada historia de usuario es lo*

suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas".

Como se observa, la idea de las historias de usuario considera fuertemente el principio de las metodologías ágiles de que los requisitos de usuario pueden cambiar en cualquier punto del desarrollo del proyecto, y justamente la intención es poder adaptarse con relativa facilidad a tales cambios. Además, hay que tener en cuenta que en muchas ocasiones, el cliente o sus representantes no tienen del todo claro el panorama y/o los pormenores del problema que realmente debe resolver el software que se está diseñando. Por esta razón se menciona que las historias de usuario son proclives a ser modificadas, replanteadas o incluso reemplazadas conforme avanza el desarrollo del proyecto.

En relación al formato de las historias de usuario, no existe una regla específica de cómo debe ser, sin embargo, existen varios modelos que pueden ser empleados según las necesidades. En todo caso, lo recomendable es que contengan toda la información que pueda esclarecer el punto a resolverse con la historia. El autor de XP, Kent Beck, en su libro [33] ofrece un modelo de historia de usuario en el que recomienda que debería contener al menos información como la fecha, el número de historia (es muy importante mantener la secuencia correcta de historias), la prioridad técnica y para el cliente, una estimación de riesgo y esfuerzo (tiempo) requerido para desarrollarla y la descripción, que debe ser lo más detallada posible. Sin embargo, si el caso lo amerita, puede también incluirse información adicional como notas, una lista de seguimiento, puntos pendientes y otros. La siguiente gráfica muestra un ejemplo de historia de usuario, de uso común.

Figura 13: Ejemplo de historia de usuario

Historia de Usuario	
Número: **1**	Nombre: **Ingreso o supresión de Roles**
Usuario: **Administrador**	
Modificación de Historia Número:	Iteración Asignada:
Prioridad en Negocio: **Baja** (Alta / Media / Baja)	Puntos Estimados:
Riesgo en Desarrollo: (Alto / Medio / Bajo)	Puntos Reales:
Descripción: **En la administración del sistema tendrá la opción de administrar usuarios, al ingresar a esta opción se desplegará un listado de los usuarios, los usuarios van a tener la opción de asignar roles, el administrador hace clic sobre esta opción relacionada con el usuario y el sistema le despliega el listado de roles disponibles para que el administrador seleccione los adecuados para ese usuario. Una vez el usuario administrador del sistema de la opción de guardar, el sistema pide confirmación y luego procederá a almacenar los cambios.**	
Observaciones:	

Fuente: XP – Extreme Programming ([34])

Un aspecto importante a tener en cuenta para la elaboración de las historias de usuario es la granularidad adecuada para las mismas, la cual dependerá del tamaño y la naturaleza del proyecto, pero lo recomendable es que exista al menos una historia de usuario por cada punto clave del problema a resolver. Sin embargo, es conveniente ir evaluando las historias que se obtiene para determinar si abarcan demasiada funcionalidad, en cuyo caso será necesario dividirlas en dos o más historias o si, por el contrario, se encuentran demasiado detalladas en donde será necesario agruparlas y/o reformularlas.

3.1.3.2.4. Roles en XP

De acuerdo a la visión original de Kent Beck [33], los roles dentro de un equipo de desarrollo de XP deberían ser los siguientes:

Programador. El o los programadores son las personas que desarrollan propiamente el código de la aplicación o sistema a elaborarse y, de ser el caso, las pruebas unitarias.

Cliente. Como se menciona, debe formar parte dentro del equipo de desarrollo. Colabora activamente en la elaboración de las historias de usuario y en la ejecución de las pruebas funcionales.

***Tester*.** Su función es la de realizar pruebas periódicamente y mantener informado al equipo de desarrollo sobre los resultados obtenidos. También le ayuda al cliente a desarrollar las pruebas funcionales.

***Tracker*.** Se encarga de realizar un seguimiento constante al equipo para verificar el correcto cumplimiento de las funciones y, sobre todo, cuán acertadas están las estimaciones originales de tiempo, para mejorar las futuras estimaciones.

Entrenador. Tiene conocimiento certero acerca de las funciones de cada miembro del equipo, y es un observador que se encarga de verificar que cada uno cumpla cabalmente sus tareas.

Consultor. Generalmente es una persona externa al equipo de desarrollo con conocimientos especializados sobre algún aspecto en particular del proyecto a desarrollarse, sobre el cual apoya a los miembros del equipo. De ser el caso, se encarga también de investigar por su cuenta.

Gestor. Su función es la de coordinar la relación entre el cliente y el equipo de desarrollo, y asegurarse de que todas las actividades se cumplan como se debe. Es el gerente del proyecto.

3.1.3.2.5. Prácticas de XP

Kent Beck define también algunas prácticas fundamentales sobre las que debe basarse el desarrollo de software mediante la metodología XP, las mismas que se enuncia brevemente a continuación.

Planificación. La comunicación entre el cliente y los desarrolladores debe ser continua para planificar correctamente el alcance y los tiempos de entrega de cada iteración.

Entregas pequeñas. Se trata de producir en el menor tiempo posible software entregable que, aunque no cumpla con todos los requerimientos del sistema, entregue un valor real útil para el negocio. En las siguientes iteraciones se podrá ir perfeccionando y aumentando su funcionalidad.

Metáfora. Consiste en describir la funcionalidad del sistema mediante un lenguaje no técnico, comprensible por el cliente, que al mismo tiempo ayude a los programadores a definir los requerimientos que deben satisfacer en cada iteración.

Diseño simple. Uno de los pilares fundamentales del desarrollo de software usando metodologías ágiles en general, y XP en particular, es la simpleza del código, pues mientras más simple es, es también más fácil de mantener y optimizar.

Pruebas. Es muy importante ir realizando pruebas para cada iteración del desarrollo, a fin de asegurar que el entregable correspondiente cumple efectivamente con los requerimientos establecidos por el cliente. Se pueden realizar diferentes tipos de pruebas, pero en XP las más utilizadas son las pruebas unitarias y las de usuario.

Refactorización del código. Consiste en la práctica constante de ir revisando y optimizando el código durante el desarrollo del proyecto, a fin de obtener código lo más limpio y legible posible. Se debe, entre otras tareas, comentar correctamente el código, eliminar código duplicado y simplificarlo.

Programación en parejas. Se trata de formar parejas de programadores para desarrollar el código, lo cual ofrece múltiples ventajas como menos errores, código más limpio, aprendizaje implícito de los programadores y entregas más rápidas.

Propiedad colectiva del código. Este principio dicta que todo el código producido por todos los programadores del equipo es accesible a todos, y todos lo pueden modificar u optimizar, siempre que documente los cambios realizados. Cualquier programador puede usar un método o función elaborado por otros, y optimizarlo si tiene la posibilidad.

Integración continua. Cada unidad de código que es elaborada por cualquiera de los programadores del equipo, debidamente probada y verificada, es inmediatamente integrada a la totalidad de la aplicación, evidenciándose así el enfoque incremental que tiene XP.

Tiempo de trabajo. El tiempo de trabajo de los programadores y el resto del equipo de desarrollo debe determinarse según el tamaño del proyecto, sin embargo, es recomendable que no excedan en ningún caso las 40 horas semanales de trabajo, a fin de evitar su fatiga y baja de rendimiento.

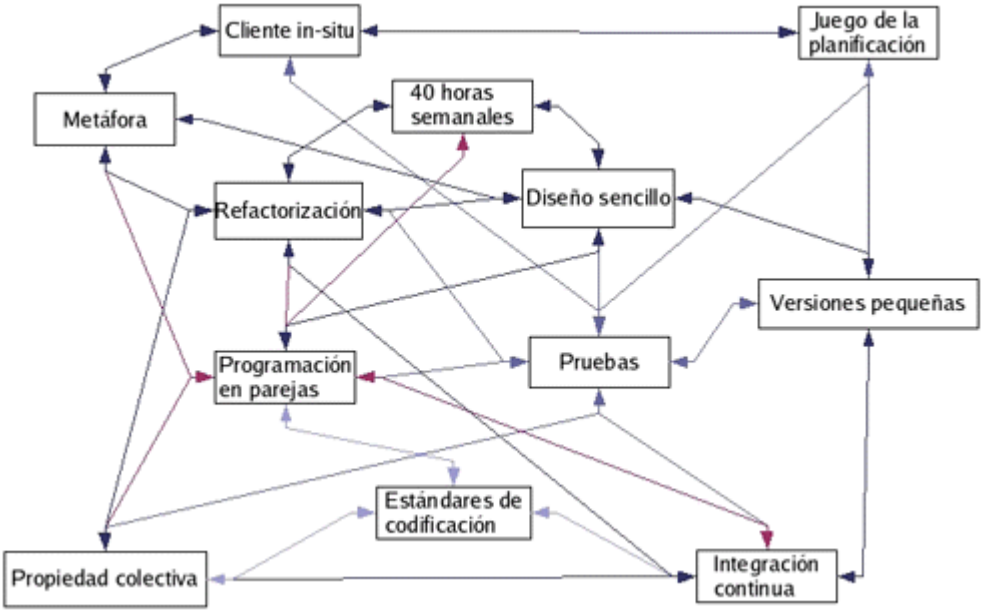
Participación activa del cliente. Como se menciona en párrafos anteriores, el cliente o sus delegados deben estar presentes e interactuar activamente como parte del equipo de desarrollo del proyecto. Esto es un factor de fundamental importancia para el éxito en los proyectos que se desarrollan mediante XP.

Estándares de programación. La elaboración de código en sí es una importante forma que tienen los programadores para comunicarse entre sí. Pero para que la misma resulte efectiva, todos deben

regirse a los mismos estándares de programación, así, el código que desarrolle un programador debe ser fácilmente comprensible, utilizable y optimizable por cualquier otro.

Es muy importante también notar que todas estas prácticas se deben aplicar juntas y se refuerzan entre sí, además de que todo el equipo de desarrollo debe conocerlas y practicarlas igualmente. El siguiente gráfico ilustra la relación existente entre estas prácticas.

Figura 14: Relación entre las prácticas de XP



Fuente: Metodologías ágiles en el desarrollo de software ([31])

3.1.3.2.6. Fases de XP

Al ser XP una metodología que se basa íntegramente en el enfoque iterativo e incremental, es importante definir el ciclo de desarrollo que se sigue dentro de esta metodología para cada una de las iteraciones a desarrollarse. Para el autor Jeffries [35], este ciclo de desarrollo está definido por los siguientes pasos:

1. *El cliente define el valor de negocio a implementar.*
2. *El programador estima el esfuerzo necesario para su implementación.*
3. *El cliente selecciona qué construir, de acuerdo con sus prioridades y las restricciones de tiempo.*
4. *El programador construye ese valor de negocio.*
5. *Vuelve al paso 1.*

Estos pasos definen, en términos generales, el procedimiento al que se rigen los programadores para desarrollar las diferentes iteraciones y es vital que lo realicen en coordinación con el cliente, ya que es éste el que define la prioridad de las historias de usuario a implementarse, en base a los valores de negocio que requiere con mayor urgencia. Es muy importante también que los programadores aprendan a estimar correctamente el esfuerzo que requerirán para desarrollar una determinada iteración, a fin de hacer cálculos correctos de los tiempos de entrega, que fácilmente se pueden subestimar.

Tomando en cuenta este ciclo de desarrollo, se procede a describir brevemente las seis fases generales que abarcan los proyectos que se desarrollan bajo la metodología XP.

Exploración. Esta fase contempla reuniones periódicas entre el o los programadores y el cliente, a fin de definir una primera versión de las historias de usuario, las que se espera que sean incluidas en la primera o primeras entregas del sistema. Esta fase sirve también para que los programadores se familiaricen con las tecnologías y herramientas a utilizarse durante el desarrollo del proyecto, y para repasar los estándares de programación con los que se regirán todos.

Planificación de la entrega. Esta fase permite al cliente determinar cuáles son las historias de usuario de mayor prioridad para su negocio y, a la vez, a los programadores a estimar el esfuerzo requerido para desarrollar cada una de ellas. Como resultado de este proceso se elabora un cronograma sobre la planificación de las entregas.

En esta fase también se establece la velocidad del proyecto, que es una medida aproximada que permite tener una estimación de cuántas de las historias de usuario se pueden implementar en un período determinado de tiempo, o cuánto tiempo tomaría implementar un conjunto determinado de historias de usuario.

Iteraciones. Obviamente, ésta es la fase en la que los programadores desarrollan cada una de las iteraciones requeridas para cumplir con toda la funcionalidad establecida en las historias de usuario.

El tiempo que toma cada iteración variará de acuerdo a la naturaleza del proyecto pero, en términos generales, se espera que no sobrepase las 3 semanas, a decir de Letelier [32].

Una excelente práctica es tratar de desarrollar en la primera iteración un conjunto de historias de usuario que permitan establecer un bosquejo general del sistema con funciones que serán útiles a lo largo de toda el desarrollo, sin embargo, esto estará sujeto a la prioridad de historias de usuario establecida por el cliente. Fuera de esto, las iteraciones se irán desarrollando secuencialmente hasta cumplir con todos los requerimientos establecidos por el cliente, obteniéndose como resultado el sistema completo listo para entrar en producción.

Además, al finalizar cada iteración se debe ejecutar las pruebas unitarias y de usuario establecidas para la misma, con la finalidad de asegurar la calidad de y la completa satisfacción del cliente con la entrega. De detectarse errores de cualquier tipo, se deberá proceder a corregirlos inmediatamente, no pudiéndose realizar la entrega sin antes finalizar esta etapa.

Producción. Determina la primera fase de puesta en marcha del sistema finalizado en las instalaciones del cliente. Sin embargo, antes de esto se requiere de una evaluación final del mismo y, de ser el caso, la ejecución de pruebas adicionales, como las pruebas de integración. Si ha habido cambios en los requerimientos, este es el momento de aplicarlos y documentarlos, para lo cual serán necesarias nuevas iteraciones aunque de tiempo mucho más reducido que las originales.

Mantenimiento. En la fase de producción se pone en marcha una primera versión del sistema desarrollado en las instalaciones del cliente, sin embargo, aparte de mantenerla funcionando será necesario ejecutar iteraciones adicionales para ir puliendo el sistema y, de ser el caso, introducir correcciones o funcionalidad extra que se haya pasado por alto en la primera versión. Para esta fase, sin embargo, la velocidad de desarrollo de las nuevas iteraciones se vuelve notablemente más lenta y, de ser el caso, se puede cambiar miembros del equipo de desarrollo.

Muerte del proyecto. Se da cuando el cliente ya no tiene más requerimientos o funcionalidad a ser añadida al sistema desarrollado y por lo tanto ya no realizará más tareas de mantenimiento. En esta fase se puede elaborar la versión final y completa de la documentación del sistema y, si bien se pueden ejecutar una o más iteraciones adicionales para corregir errores menores, la estructura y el funcionamiento generales del sistema ya no se modifican. La muerte del proyecto también puede darse por causas de fuerza mayor, como por ejemplo por falta de presupuesto del cliente para continuar con el mismo.

3.1.4. Bases de datos

El término base de datos nació casi al mismo tiempo que la informática en sí, y ha ido evolucionando constantemente, de forma paralela a ésta. Hoy en día existe una amplia gama de tecnologías y motores de bases de datos disponibles, siendo cada cual más o menos apropiado para diferentes escenarios. El desarrollo de aplicaciones de todo tipo está íntimamente relacionado con lo que son las bases de datos y su manejo, por lo que todo programador o desarrollador de software debe entender perfectamente la función y utilidad de las mismas.

Un primer concepto de base de datos se lo obtiene de la autora Marqués [24], quien manifiesta que *"Una base de datos es un conjunto de datos almacenados en memoria externa que están organizados mediante una estructura de datos. Cada base de datos ha sido diseñada para satisfacer los requisitos de información de una empresa u otro tipo de organización, como por ejemplo, una universidad o un hospital"*. Por otro lado, el autor Silberschatz [36] ofrece una definición mucho más concisa al decir que una base de datos consiste en *"una colección de datos interrelacionados y un conjunto de programas para acceder a dichos datos"*. Por último, para la autora Gómez [37], *"una base de datos no es más que un conjunto de información (un conjunto de datos) relacionada que se encuentra agrupada o estructurada"*.

Contrastando todos los conceptos comparados y atendiendo a las principales similitudes entre ellos se puede deducir que una base de datos consiste en un almacén que permite guardar conjuntos de datos con una estructura común y relacionados entre sí, es decir, que pertenecen a un mismo tipo de entidad, fenómeno o institución. Una base de datos, sin embargo, necesariamente requiere de un sistema de gestión de bases de datos (SGBD) para funcionar y gestionar el almacenamiento y posterior acceso a su información adecuadamente. El SGBD es un programa o servicio que corre en un computador con la finalidad de guardar, modificar, eliminar o recuperar los datos desde un disco duro, disco en estado sólido o disco SCSI, que pueden ser los soportes físicos donde se almacenan dichos datos.

3.1.4.1. Sistemas de bases de datos y modelos de SGBD

Antes del surgimiento y popularización de las bases de datos, los programadores trabajaban directamente con ficheros de disco donde se almacenaba la información correspondiente a una determinada aplicación de forma manual. Según la complejidad de ésta, se podía requerir del manejo

de muchos ficheros para gestionar exitosamente la información que requería tal aplicación. Sin embargo, este sistema de manejo de la información traía muchas limitaciones como la falta de estandarización (ya que cada programador podía elegir su propia metodología para la creación y manejo de los archivos), la falta de seguridad, la dificultad para compartir la información entre los diferentes departamentos o divisiones de una empresa u organización, y la posibilidad de terminar con información inconsistente y, por tanto, no confiable en los ficheros.

Para sortear tales inconvenientes se crearon las bases de datos. Una base de datos puede ser vista como un almacén de datos centralizado donde todos los departamentos de una organización guardan la información y de donde todos la reciben, con lo cual se obtiene una fuente de información mucho más segura, fiable, rápida y actualizada. El primer sistema de base de datos surgió en la década de 1960 en Estados Unidos, como parte del proyecto de su gobierno de enviar al hombre a la luna, y se denominó *General Update Access Method* (GUAM), en cuyo diseño y desarrollo participó IBM, según lo manifiesta la autora Marqués [24]. Este sistema pertenece al grupo de los denominados sistemas jerárquicos, ya que la información se organiza en distintos niveles, formando la estructura de un árbol.

Posteriormente, en la misma década, el sistema *Integrated Data Store* (IDS) fue creado por General Electric, como parte de un nuevo modelo de base de datos conocido como sistema de red. La filosofía de este modelo fue permitir manejar las relaciones entre los datos de las bases de datos de manera más precisa y fiable, formándose así una evolución del modelo jerárquico. Los dos modelos juntos, a su vez, y sus respectivas bases de datos se constituirían en los primeros SGBD que surgieron dentro del mundo de las bases de datos, sin embargo, todavía presentaban múltiples desventajas como la dependencia de los datos y la necesidad de escribir aplicaciones complejas para poder manejarlos y hacer uso efectivo de los datos almacenados.

Posteriormente, en la década de 1970, de la mano de los ingenieros de IBM, en especial de Edgar Codd, se concibió por primera vez el modelo de bases de datos relacional, que ha tenido mucho auge durante los últimos tiempos y tiene amplio uso hasta la actualidad. Uno de los primeros sistemas desarrollados atendiendo a esta nueva corriente fue System R, que posteriormente, en la década de 1980, sirvió de base para el desarrollo de otros sistemas como DB2 (también de IBM), Oracle y Microsoft SQL Server. De System R también surgió la creación del lenguaje *Structured Query Language* (SQL), un lenguaje de consultas a bases de datos que permite la adición, consulta, eliminación y modificación de datos en las bases de datos relacionales, tanto dentro del desarrollo de software como desde otros ámbitos, siendo SQL un lenguaje muy utilizado hasta la actualidad.

Sin embargo, el modelo relacional aún presenta algunos inconvenientes, sobre todo, la poca flexibilidad para representar fielmente los fenómenos del mundo real mediante bases de datos. Esto provocó que algunos ingenieros, incluyendo Codd, trabajaran en optimizar el modelo relacional de múltiples formas, sobre todo para volverlo más flexible. Así surgió el modelo entidad – relación en 1976, como un método para diseño de bases de datos, y las versiones extendidas conocidas como RM/T en 1979 y RM/V2 en 1990, respectivamente.

Pese a tales avances, surge una tercera generación de los SGBD conocida como el modelo orientado a objetos, el cual, obviamente, está inspirado y va de la mano de alguna manera con el surgimiento de los lenguajes de programación orientados a objetos, en la década de 1990. En este sentido, Silberschatz [36] explica que *"el enfoque orientado a objetos para la programación fue introducida por primera vez con el lenguaje Simula 67, que se diseñó para la programación de simulaciones. Smalltalk fue uno de los primeros lenguajes de programación orientada a objetos para aplicaciones generales. Actualmente, los lenguajes C++ y Java son los lenguajes de programación orientada a objetos más usados"*. Los sistemas orientados a objetos siguen utilizando SQL como lenguaje de consultas, si bien lo han extendido y optimizado, por lo cual se habla también de sistemas objeto – relacionales.

Si bien el modelo relacional sigue siendo ampliamente utilizado hasta hoy en día y, según el escenario, las bases de datos orientadas a objetos también tienen un gran margen de utilización, en las últimas décadas han surgido otros modelos de SGBD tales como las bases de datos distribuidas, activas, deductivas, homogéneas, heterogéneas, temporales y multimedia, muchos de los cuales aún se encuentran en etapa de desarrollo y en plena evolución. En muchos casos se aplican sistemas que combinan dos o incluso más de los modelos analizados.

Dado el crecimiento geométrico en los volúmenes de información que las grandes corporaciones y organizaciones de diferentes índoles requieren manejar, han surgido además otros paradigmas dentro de la administración de datos, como lo son los *Data Warehouses*, que no son otra cosa que grandes almacenes de información cuyo propósito principal es el de servir como herramientas de apoyo a la toma de decisiones a nivel gerencial y el procesamiento automático en tiempo real de dicha información, sin importar la cantidad de datos involucrados en el proceso. Subyacentemente surgen técnicas como el *Data Mining* o minería de datos, cuya función es la de generar automáticamente información previamente desconocida a partir del procesamiento de grandes volúmenes de información, Dicha información resulta de vital importancia hoy en día para sus respectivas empresas

y organizaciones, ya que les permite conocer su estado exacto y tomar las decisiones más acertadas en el momento preciso, fortaleciendo su competitividad y mejorando su posicionamiento en el mercado.

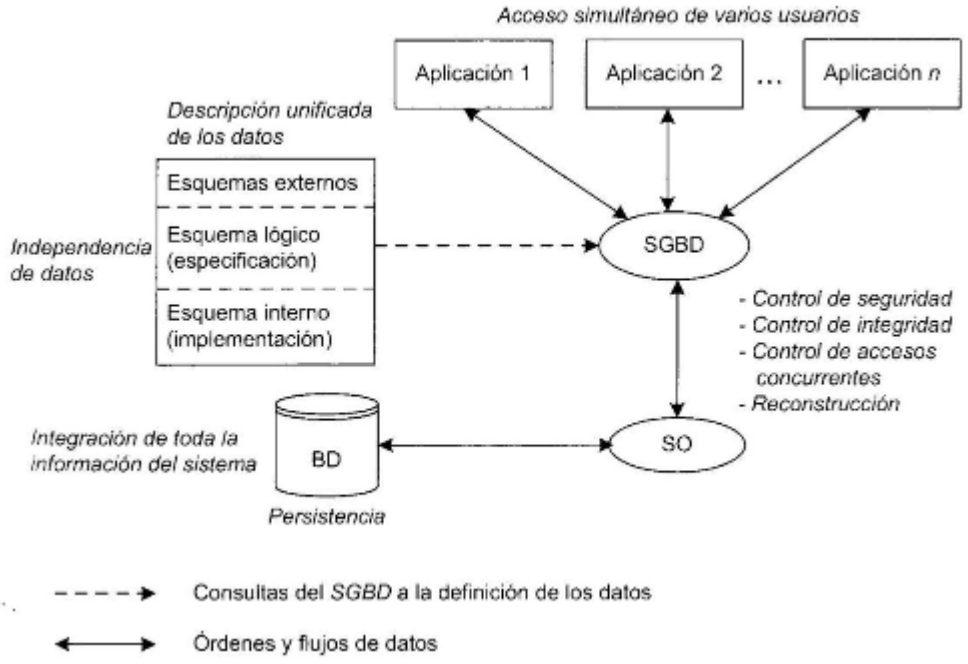
Finalmente, hay que tener muy en cuenta que la generalización en el uso de la web a todo nivel y en todo el mundo ha influido decisivamente dentro del diseño e implementación de bases de datos. Las aplicaciones web se han vuelto cada vez más populares y éstas a su vez requieren del acceso a servidores de bases de datos y SGBD que deben manejar en muchos casos el acceso de hasta varios millones de usuarios alrededor del mundo, simultáneamente. Esto a su vez ha obligado a perfeccionar los SGBD aumentando su seguridad, fiabilidad, flexibilidad y robustez, a la vez que haciéndolas más ágiles.

Según manifiesta la autora Celma [5], para que un SGBD pueda ser considerado como tal, debe cumplir como mínimo las siguientes características:

- *Integración de toda la información de la organización*
- *Persistencia de los datos*
- *Accesibilidad simultánea para distintos usuarios*
- *Independencia de los programas respecto a la representación física de los datos*
- *Definición de vistas parciales de los datos para distintos usuarios*
- *Mecanismos para controlar la integridad y seguridad de los datos.*

Como se ve, si bien son pocas características, son las mínimas que debe cumplir cualquier SGBD para poder llamarse así, ya que representan la razón misma por la que estos sistemas fueron creados. El siguiente gráfico muestra claramente la forma centralizada en la que trabajan los SGBD y la relación existente con el resto de los elementos de su entorno.

Figura 15: Relación de un SGBD con los elementos de su entorno



Fuente: Bases De Datos Relacionales ([5])

Otro factor clave a tomar en cuenta son los diferentes tipos de usuarios que se involucrarán en la operación de un SGBD y en la implementación, operación y mantenimiento de una base de datos, de cualquier modelo que esta sea. Para la autora Marqués [24], los usuarios que tendrán alguna interacción con una base de datos son 4: el administrador, el o los diseñadores, los programadores y los usuarios finales. Sin embargo, Gómez [37] tiene una visión algo diferente en este sentido y define los usuarios sofisticados, en reemplazo de el o los diseñadores, ya que en la práctica éstos últimos suelen coincidir con los administradores. Se concuerda más con la segunda autora y, en ese sentido, una breve descripción de estos tipos de usuarios sería la siguiente:

Usuario común o final. Son, por regla general, los representantes o empleados del cliente, quienes van a interactuar a diario con la aplicación y, por ende, con su base de datos. Estos usuarios desconocen los pormenores del funcionamiento o estructura de la base de datos o del SGBD, únicamente se limitan a utilizar las interfaces implementadas por el o los programadores, las que a su vez les permiten visualizar, guardar, modificar o eliminar los datos contenidos en la base de datos.

Programador. Es el tipo de usuario que crea la aplicación, o parte de ella, para el usuario final, diseñando las interfaces y estableciendo la funcionalidad de las mismas a través de las líneas de código. Desde ellas, el programador requiere conectarse y acceder a las bases de datos, mediante las funciones provistas por su respectivo SGBD, para a su vez programar las operaciones requeridas por el usuario final.

Usuario sofisticado. Son usuarios como los administradores de sistemas, que por cualquier motivo requieren conectarse a las bases de datos y realizar consultas a las mismas, pero lo hacen directamente, sin necesidad de usar las interfaces creadas por los programadores. Estos usuarios tienen conocimientos técnicos que les permiten usar directamente el SQL u otro lenguaje de consulta a bases de datos, de ser el caso, para realizar las operaciones requeridas.

Administrador. Como se menciona en párrafos anteriores, el administrador es, por regla general, el diseñador o uno de los diseñadores de la base de datos. Obviamente, este usuario tiene también un buen nivel de conocimientos técnicos y puede conectarse directamente a la base de datos y hacerle consultas usando SQL o cualquier lenguaje requerido. Entre las funciones más importantes del administrador se pueden citar las siguientes:

- Definir el esquema de la base de datos
- Definir la estructura y el método de acceso a la base de datos
- Modelar el esquema y la organización física de la base de datos
- Conceder autorizaciones para el acceso a los datos a los usuarios requeridos
- Realizar tareas de mantenimiento rutinario incluyendo copias de seguridad periódicas de la base de datos, velar por que siempre haya el espacio de disco requerido, revisar que no exista sobrecarga en la base de datos y monitorear el nivel de rendimiento del SGBD, entre otras.

3.1.4.1.1. Microsoft SQL Server

Microsoft SQL Server es un SGBD de muy amplio uso a nivel mundial, ya que ofrece ediciones tanto para equipos personales como para servidores de cualquier tipo y capacidad, incluyendo Enterprise, Developer, Standard, Express y SQL Azure, esta última especialmente diseñada para funcionar en el entorno de *Cloud Computing* Microsoft Azure. El sistema incluye gran cantidad de utilidades y herramientas visuales como, *Microsoft SQL Server Management Studio*, que facilitan enormemente las tareas de creación, administración y mantenimiento de las bases de datos. Es uno de los SGBD más

robustos disponible en la actualidad, y periódicamente se publican nuevas versiones que mejoran la seguridad de sus bases de datos y, al mismo tiempo, facilitan la administración de sus bases de datos mediante la inclusión de nuevas herramientas y la incorporación de nuevas características.

Microsoft SQL Server vio la luz por primera vez en la década de 1980 y fue originalmente diseñado para sistema UNIX, donde fue conocido simplemente como SQL Server. A partir del año 1993, Microsoft comenzó a participar activamente en el desarrollo de este SGBD y lo incluyó como parte de su sistema operativo Windows NT, adoptando desde entonces el nombre de Microsoft SQL Server e independizándose definitivamente del desarrollo de SQL Server original. Hoy en día, es el SGBD que se utiliza por excelencia dentro del desarrollo de software mediante plataformas Microsoft, como Visual Studio, ya que las mismas incluyen soporte nativo para Microsoft SQL Server, y se garantiza la compatibilidad total entre ellos, siendo la última versión publicada Microsoft SQL Server 2016. La siguiente tabla muestra la evolución de las versiones de Microsoft SQL Server desde sus inicios, con sus respectivos nombres clave de proyecto.

Tabla 3: Evolución de las versiones de Microsoft SQL Server

Versión	Año	Nombre de la versión	Nombre clave
1.0 (OS/2)	1989	SQL Server 1-0	SQL
4.21 (WinNT)	1993	SQL Server 4.21	SEQUEL
6.0	1995	SQL Server 6.0	SQL95
6.5	1996	SQL Server 6.5	Hydra
7.0	1998	SQL Server 7.0	Sphinx
-	1999	SQL Server 7.0 OLAP Tools	Plato
8.0	2000	SQL Server 2000	
8.0	2003	SQL Server 2000 64-bit Edition	Liberty
9.0	2005	SQL Server 2005	Yukon
10.0	2008	SQL Server 2008	Katmai
10.25	2010	SQL Azure DB	CloudDatabase
10.50	2010	SQL Server 2008 R2	Kilimanjaro
11.0	2012	SQL Server 2012	Denali
12.0	2014	SQL Server 2014	SQL14 (antes Hekaton)
13.0	2016	SQL Server 2016	

Fuente: Fundamentos de bases de datos ([36])

Microsoft SQL Server implementa como lenguaje de consultas el *Transact-SQL* (T-SQL), que es la principal forma de comunicarse con el servidor y que constituye una extensión del SQL estándar que incluye funcionalidad extra como el uso de variables, la programación de procedimientos y tiene múltiples funciones integradas para facilitar operaciones matemáticas, conversiones de tipos de datos, procesamiento de fechas, etc. Lógicamente, incluye también toda la funcionalidad estándar del lenguaje SQL que, a su vez, soporta comandos de *Data Definition Language* (DDL), que permite crear y modificar bases de datos con sus respectivas tablas, relaciones, índices y demás, y de *Data*

Manipulation Language (DML), que trabaja directamente con los datos almacenados en las tablas permitiendo consultarlos, añadirlos, eliminarlos o modificarlos.

Microsoft SQL Server ofrece varias características interesantes que resultan muy útiles para responder efectivamente a las actuales necesidades de almacenamiento y administración de datos de las organizaciones, entre ellas se puede mencionar:

- Ofrece un entorno visual muy completo para la administración
- Trabaja en modo cliente – servidor
- Soporta la ejecución de bloques de comandos mediante transacciones
- Permite crear y ejecutar procedimientos almacenados
- Es capaz de integrarse e intercambiar información con otros SGBD

Para elegir un SGBD a ser implementado en una organización o empresa es muy importante tener en cuenta los tipos de datos que soporta el mismo, ya que se debe asegurar que se puede representar fielmente el flujo de datos que se da entre todos los departamentos y divisiones de dicha organización, para implementarlos en la base de datos.

Microsoft SQL Server soporta muchos tipos de datos diferentes que se pueden clasificar en varias categorías, siendo las más importantes los siguientes:

Numéricos. Permiten guardar valores numéricos tanto enteros como fraccionarios en múltiples formatos. Dentro de esta categoría, los tipos más utilizados son *int, bigint, money, numeric, real, decimal, float, smallint, smallmoney* y *tinyint*.

Textuales. Sirven para almacenar valores alfanuméricos, cadenas de texto e inclusive textos completos con caracteres especiales como saltos de línea y otros. Los tipos más utilizados dentro de esta categoría son: *string, char, varchar, nvarchar* y *text*.

Fecha/hora. Permiten almacenar valores relativos al tiempo. Incluye tipos de datos como *date, datetime, datetime2, time, smalldatetime* y *timestamp*.

Booleanos. Dentro de esta categoría, el tipo de dato ofrecido por Microsoft SQL Server es *bit*, el mismo que permite guardar valores de 0 o 1, equivalentes a verdadero o falso.

Multimedia. Se puede también usar campos dentro de una base de datos de Microsoft SQL Server para almacenar información multimedia, específicamente imágenes. El tipo de dato a utilizarse en este caso es *image*.

Por último, existen otros tipos de datos de uso menos frecuente, sin embargo, no por eso menos importantes. Algunos de ellos son de tipo geográficos como *geography* y *geometry*, mientras que otros son, básicamente, tipos que almacenan textos aunque en formatos específicos. Ejemplos de estos tipos son *sql_variant* y *xml*. Existen aún tipos de datos adicionales como *hierarchyid, uniqueidentifier* y *datetimeoffset* que tienen usos muy específicos y que son requeridos dentro del diseño de bases de datos para aplicaciones de diversas índoles.

3.1.5. Aplicaciones Web

Por experiencia propia, se concuerda íntegramente con el autor Lujan [38], quien manifiesta que *"Internet y la Web han influido enormemente tanto en el mundo de la informática como en la sociedad en general. Si nos centramos en la Web, en poco menos de 10 años ha transformado los sistemas informáticos: ha roto las barreras físicas (debido a la distancia), económicas y lógicas (debido al empleo de distintos sistemas operativos, protocolos, etc.) y ha abierto todo un abanico de nuevas posibilidades. Una de las áreas que más expansión está teniendo en la Web en los últimos años son las aplicaciones web"*. De esta afirmación se deduce que, indiscutiblemente, unos de los usos más extendidos que se da en la actualidad a las *World Wide Web* consiste precisamente en las aplicaciones web, debido a que estas superan fácilmente muchas de las barreras existentes en las aplicaciones convencionales de escritorio, de amplio y casi exclusivo uso en la década de los 90.

En sus inicios, la *World Wide Web* tenía como objetivo casi único el permitir a sus usuarios acceder a diversos documentos que contenían información textual y opcionalmente gráfica, sobre temas específicos de interés para dichos usuarios. Sin embargo, lo verdaderamente novedoso de este sistema era que la información estaba organizada en forma de "malla", es decir, cada documento estaba enlazado con uno o más documentos que contenían información relacionada o complementaria, permitiéndose a los usuarios "saltar" de un documento a otro para revisar la información que le resulte realmente interesante de forma inmediata. Esto tiene obvias ventajas en relación a la investigación bibliográfica tradicional mediante libros impresos, siendo la principal el drástico incremento en la velocidad a la que un usuario podía acceder a un determinado número de documentos y llegar a una porción específica de información que le resulte útil. Así, el autor Theodor Nelson [39] manifiesta ya en los años 60 que el hipertexto es *"un cuerpo de material escrito o pictórico interconectado en una forma compleja que no puede ser representado en forma conveniente haciendo uso del papel"*.

Sin embargo, todo este conglomerado de información electrónica interconectada entre sí estaba disponible únicamente para ciertas personas afortunadas que podían tener acceso al mismo, especialmente investigadores y personas fuertemente relacionadas con los ámbitos universitario y científico. Pero con el pasar de los años y de las décadas, el sistema se fue haciendo cada vez más accesible a mayor número de personas de todos los países y continentes lo que, a su vez, determinó que esta red se vaya haciendo cada vez más amplia y rica, diversificándose enormemente la cantidad y variedad de información contenida en la misma, así como el número y tipo de personas que interactuaban con la misma. Y junto con la cantidad de información y la cantidad de usuarios, fueron también creciendo los usos y tipos de servicios ofrecidos por esta gran red mundial, apareciendo así servicios que hasta hoy en día son de vital importancia dentro de todo ámbito del quehacer humano, tales como el correo electrónico, los servicios de noticias, los servicios de mensajería, los sistemas de intercambio de archivos y los sistemas de acceso remoto, para mencionar algunos de los principales.

Toda esta variedad de servicios a su vez dieron origen al aparecimiento de las aplicaciones web, que son el objeto de estudio principal dentro del presente capítulo, las mismas que han ido evolucionando constantemente a lo largo del tiempo, hasta la actualidad. Pero para comprender de mejor manera el concepto y funcionamiento de las aplicaciones web, es necesario primero entender y analizar lo que son las arquitecturas cliente / servidor.

3.1.5.1. Arquitecturas Cliente / Servidor

La gran mayoría de aplicaciones tradicionales de escritorio funcionan en un solo computador donde se encuentran instalados los archivos necesarios para tal funcionamiento, como archivos ejecutables, *Dynamic Linked Libraries* (DLLs), archivos de datos y diferentes tipos de archivos auxiliares. Todo este conglomerado de archivos generan a su vez procesos que permiten la ejecución de una aplicación de escritorio de manera independiente y mediante el uso de los recursos locales del computador en cuestión, como son su memoria, disco duro y procesador, si bien desde el inicio de las aplicaciones informáticas ya existían algunas de ellas que funcionaban de manera distribuida, requiriendo acceder a recursos tales como bases de datos instaladas en servidores remotos.

Tales aplicaciones distribuidas, que inicialmente tuvieron mayor incidencia en los ámbitos comercial y corporativo, fueron ganando cada vez mayor popularidad con el pasar de los años dadas las ventajas que ofrecían, como la posibilidad de que múltiples usuarios compartan información accediendo a una base de datos en común, mientras que cada uno ejecutaba su propia instancia de la

aplicación, instalada en su propio computador. Sin embargo, resulta lógico mencionar que, para que dicha mecánica de funcionamiento de este tipo de aplicaciones sea posible, se requería de un medio de conexión entre el computador del usuario (cliente) y el computador que contenía y gestionaba los datos compartidos (servidor). Por lo tanto, se puede deducir fácilmente que el aparecimiento y evolución de estas aplicaciones distribuidas (cliente / servidor) estuvieron íntimamente ligadas al aparecimiento y evolución de las redes informáticas.

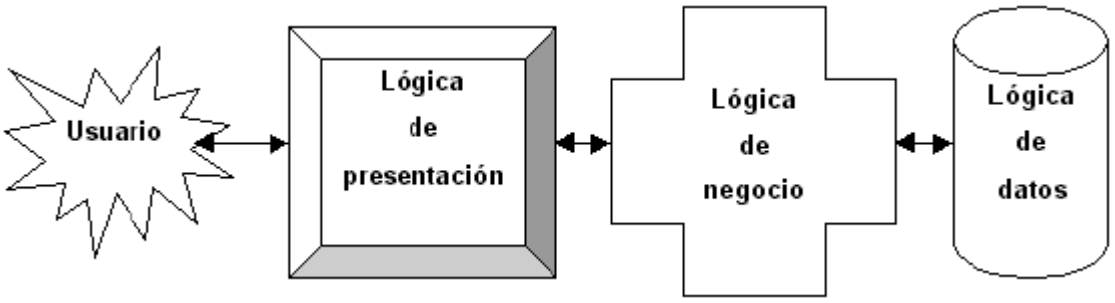
A este respecto, el autor Luján [38] explica que *"La arquitectura cliente/servidor permite la creación de aplicaciones distribuidas. La principal ventaja de esta arquitectura es que facilita la separación de las funciones según su servicio, permitiendo situar cada función en la plataforma más adecuada para su ejecución"*. Y además, él mismo menciona como principales ventajas de esta arquitectura las siguientes:

- *Las redes de ordenadores permiten que múltiples procesadores puedan ejecutar partes distribuidas de una misma aplicación, logrando concurrencia de procesos.*
- *Existe la posibilidad de migrar aplicaciones de un procesador a otro con modificaciones mínimas en los programas.*
- *Se obtiene una escalabilidad de la aplicación. Permite la ampliación horizontal o vertical de las aplicaciones. La escalabilidad horizontal se refiere a la capacidad de añadir o suprimir estaciones de trabajo que hagan uso de la aplicación (clientes), sin que afecte sustancialmente al rendimiento general. La escalabilidad vertical se refiere a la capacidad de migrar hacia servidores de mayor capacidad o velocidad, o de un tipo distinto de arquitectura sin que afecte a los clientes.*
- *Posibilita el acceso a los datos independientemente de donde se encuentre el usuario.*

Un concepto de vital importancia que se menciona por el citado autor es la separación de funciones, ya que en este tipo de arquitecturas es fundamental definir qué procesos de la aplicación se ejecutarán en el servidor y qué procesos en el o los clientes. Sin embargo, ésta separación de funciones se suele reflejar también a nivel interno del servidor, lo cual da origen al desarrollo en capas, que no es otra cosa que una sub – separación de funciones aplicada a los procesos del servidor dentro de esta arquitectura distribuida.

El siguiente gráfico ilustra de manera sencilla y comprensible el concepto de separación de funciones aplicada a los procesos del servidor:

Figura 16: Separación de funciones en el servidor

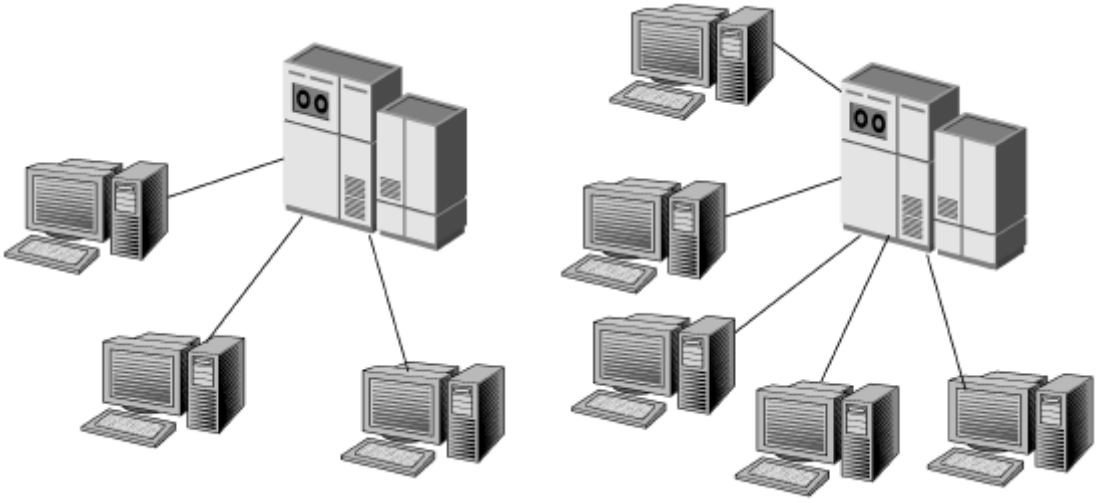


Fuente: Programación de aplicaciones web: historia, principios básicos y clientes web ([8])

En el presente apartado, sin embargo, no se profundiza sobre la separación de funciones del servidor y el modelo de desarrollo en capas, puesto que esto ya se trató con mayor detalle en el apartado 3.1.2.3. (Programación en capas).

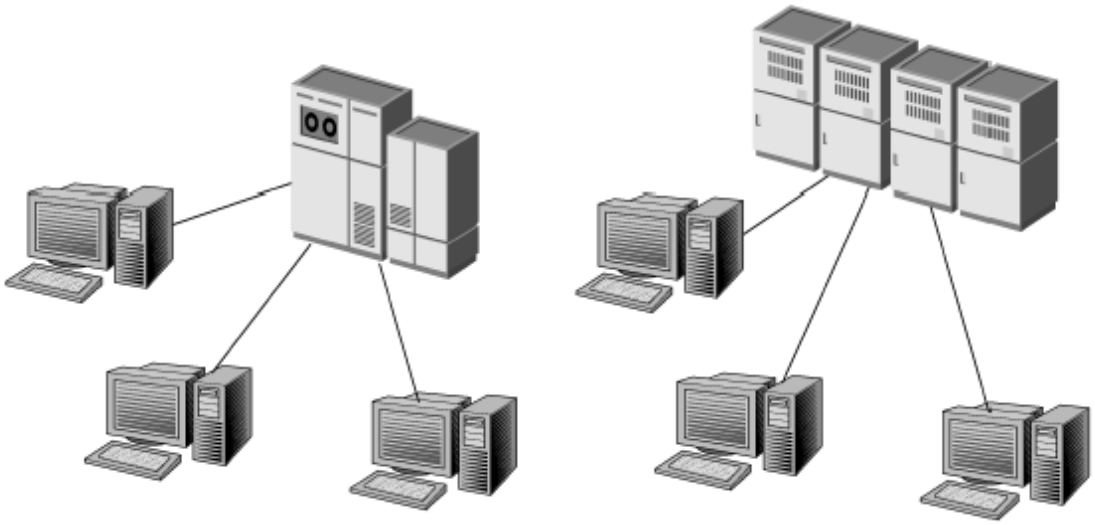
En cuanto a las ventajas de la arquitectura cliente / servidor, el autor menciona la escalabilidad horizontal y vertical de las aplicaciones desarrolladas atendiendo a la arquitectura cliente servidor, lo cual también es una ventaja clave que determinó la popularización de las aplicaciones cliente / servidor. Los dos siguientes gráficos permiten comprender estas ventajas de mejor manera.

Figura 17: Escalabilidad horizontal de las aplicaciones cliente / servidor



Fuente: Programación de aplicaciones web: historia, principios básicos y clientes web ([8])

Figura 18: Escalabilidad vertical de las aplicaciones cliente / servidor



Fuente: Programación de aplicaciones web: historia, principios básicos y clientes web ([8])

Cabe por último mencionar que, la popularización y globalización del uso de la *Web* y el Internet, las mismas que por obvias razones estuvieron también sustentadas en el desarrollo de las redes informáticas, conjuntamente con el aparecimiento y evolución de las aplicaciones cliente / servidor, dieron lugar al aparecimiento de las primeras aplicaciones web, las mismas que hoy en día son muy difundidas a nivel mundial, siendo la opción preferida por organizaciones de todo tipo dadas las múltiples ventajas y flexibilidad que ofrecen. De hecho, el autor Luján [38] considera que las aplicaciones web son un tipo de aplicaciones cliente / servidor, y menciona la importancia de entender este tipo de arquitectura para quien se interese por el desarrollo de aplicaciones web. Citando textualmente lo mencionado por el autor, *"Las aplicaciones web son un tipo especial de aplicaciones cliente/servidor. Antes de aprender a programar aplicaciones web conviene conocer las características básicas de las arquitecturas cliente/servidor"*.

3.1.5.2. Programación y lenguajes del lado del servidor

Al ser las aplicaciones web aplicaciones que funcionan bajo la arquitectura cliente / servidor, existe código que se ejecuta en ambos lados. En este caso, el usuario de la aplicación se vale en la gran mayoría de los casos un navegador web, que es el software que actúa como cliente y hace peticiones al servidor

que, nuevamente en la gran mayoría de los casos, se trata de páginas web que éste envía al cliente como respuesta a través de un programa conocido como servidor web. De hecho, la gran mayoría de aplicaciones web funcionan bajo el protocolo HTTP y se basan en la transferencia de páginas web, las mismas que emplean el lenguaje HTML.

Como ya se ha analizado con más detalle en apartados anteriores, en los inicios de la web y del hipertexto todos los documentos disponibles en la web eran estáticos, es decir, contenían siempre la misma información que era enviada a los usuarios, y esto fue así por muchos años. Sin embargo, en un determinado momento se llegó a identificar la necesidad de crear páginas web dinámicas, es decir, páginas cuyo contenido pudiera variar dependiendo del usuario que las solicitaba y de sus preferencias y necesidades específicas. Así nació el *Common Gateway Interface* (CGI) que, según Cobo [40] *"lo que define es un estándar para establecer la comunicación entre un servidor web y un programa. Esta interfaz define una forma cómoda y simple de ejecutar programas que se encuentran en la máquina en la que se aloja el servidor, a través de la definición de una serie de reglas que deben cumplir tanto las aplicaciones como los servidores para hacer posible la ejecución de los programas"*.

Al tratarse CGI de una interfaz, se podía programar sin una dependencia rígida de algún lenguaje de programación en especial, incluyéndose entre los más utilizados por los programadores C, C++, Perl y Visual Basic. Todo esto supuso un gran salto dentro del avance de la Web, ya que la volvió mucho más útil, dinámica e interactiva, y por primera vez los usuarios podían ver información personalizada, introducir datos para que sean procesados remotamente, o ver los resultados de cálculos matemáticos realizados en el servidor. La evolución de este tipo de páginas web fue desde entonces avanzando a pasos agigantados y como parte de este proceso surgieron lenguajes de programación específicamente diseñados para los servidores de aplicaciones web, siendo los más conocidos y utilizados a nivel global PHP, ASP, JSP y ColdFusion. Estos lenguajes se conocen comúnmente como lenguajes del lado del servidor.

Posteriormente, con el aparecimiento de otras tecnologías y plataformas de desarrollo de software, como lo es .NET, han aparecido nuevos lenguajes del lado del servidor como lo es ASP.NET el mismo que, si bien se puede considerar una evolución del tradicional ASP, representa en realidad una nueva filosofía de desarrollo de aplicaciones web que permite crear aplicaciones potentes, seguras y versátiles, teniendo a su disposición toda la funcionalidad característica de un *framework* de desarrollo de software como lo es .NET y sus librerías.

Se procede a continuación a definir a breves rasgos lo que es ASP.NET y sus principales características.

3.1.5.2.1. ASP.NET

Considerando las múltiples ventajas que ofrecen las aplicaciones web sobre las aplicaciones tradicionales de escritorio, es comprensible que su creación y uso se haya incrementado y popularizado notablemente a lo largo de los años. A este respecto, los autores Berzal, Cubero y Cortijo [41] manifiestan que *"Hoy en día, resulta bastante común implementar la interfaz de una aplicación utilizando páginas web en vez de las ventanas y los controles específicos de un sistema operativo concreto. En lugar de escribir una aplicación para un sistema operativo concreto, como puede ser Windows, en muchas situaciones es preferible crear aplicaciones web a las que se accede a través de Internet"*.

Así, los lenguajes del lado del servidor también se han vuelto cada vez más versátiles y populares entre los programadores, ya que constituyen la base para la creación de aplicaciones web. Uno de estos lenguajes es ASP.NET, que viene siendo el sucesor del tradicional *Active Server Pages* (ASP), cuya primera versión apareció en el año 1996 y fue un lenguaje del lado del servidor orientado a *Internet Information Services* (IIS), el servidor web por excelencia de Microsoft. ASP.NET, sin embargo, está enlazado a la plataforma .NET, lo que ofrece múltiples ventajas como la disponibilidad de usar una extensa librería de potentes clases predefinidas que simplifican enormemente las tareas de programación, la versatilidad de usar un lenguaje orientado a objetos y la posibilidad de separar la lógica de programación del diseño de interfaces.

ASP.NET ha ido evolucionando a lo largo de los años conjuntamente con toda la plataforma .NET, lo que lo vuelve un lenguaje cada vez más potente y con mayores prestaciones para desarrollar aplicaciones web de la más alta calidad. ASP.NET viene integrado en el entorno de desarrollo Microsoft Visual Studio por lo que desarrollar una aplicación web en ASP.NET es casi tan cómodo y sencillo como desarrollar una aplicación tradicional de escritorio. Sin embargo, para un desarrollador que no tenga ninguna experiencia con ASP o con el desarrollo bajo .NET, la curva de aprendizaje puede ser bastante considerable y requerir tiempo. Se recomienda la lectura completa de ciertas obras que explican detalladamente todo el funcionamiento y las características de ASP.NET, como la mencionada en [42].

Una de las características claves de ASP.NET es que funciona a través de *postbacks*, que es el comportamiento predeterminado de las páginas creadas con esta tecnología. Básicamente consiste en

que una página web se recarga a sí misma una o más veces, pudiendo variar en su contenido y funcionalidad en cada carga. Si en una página web creada con ASP.NET se coloca un botón y se ejecuta la página, al hacer click en este botón se produce un *postback*, es decir una recarga de la misma página, sin necesidad de crear código alguno ni de cambiar ninguna configuración. Lo mismo puede suceder con múltiples tipos de controles y eventos, y no exclusivamente con botones.

Otra característica clave de ASP.NET se conoce como *ViewState*, la que funciona en estrecha relación con los *postbacks* y proporciona una gran versatilidad a las páginas ASP.NET. *ViewState* es una característica que permite mantener la información de una página y de sus controles entre diferentes cargas de dicha página, es decir, entre *postbacks*. De este modo, los datos ingresados en un cuadro de texto, por ejemplo, se mantienen cuando se realiza una recarga de la página, lo que permite utilizar dicho valor para realizar cualquier operación requerida por el usuario.

Finalmente, cabe mencionar que, al ser parte de la plataforma .NET, ASP.NET tiene a su disposición una amplia gama de librerías y clases para conectarse y manejar bases de datos de diferentes motores, sin embargo, lo más recomendable es trabajar con Microsoft SQL Server ya que al ser ambas tecnologías propietarias de Microsoft existe total compatibilidad entre ellas, estando ASP.NET especialmente diseñado para trabajar con Microsoft SQL Server. .NET provee la biblioteca *.NET Entity Framework*, la misma que ofrece métodos muy versátiles para comunicarse y manejar bases de datos con todas las operaciones necesarias, mediante una sintaxis más sencilla y a la vez potente que lenguajes como SQL. En la siguiente figura se observa un ejemplo del lenguaje ASP.NET.

Figura 19: Ejemplo de una página ASP.NET sencilla

```
<%@ Page language="c#" %>
<html>
 <head>
   <title>Hora.aspx</title>
 </head>
 <script runat="server">
  public void Button_Click (object sender, System.EventArgs e)
  {
    LabelHora.Text = "La hora actual es " + DateTime.Now;
  }
 </script>
 <body>
  <form method="post" runat="server">
    <asp:Button onclick="Button_Click" runat="server"
                Text="Pulse el botón para consultar la hora"/>
    <p>
    <asp:Label id=LabelHora runat="server" />
  </form>
 </body>
</html>
```

Fuente: Desarrollo profesional de aplicaciones web con Asp.net ([41])

3.1.6. Procesos de atención al usuario

Toda organización tiene como objeto crear productos y servicios destinados para su consumo por parte de uno o más segmentos de usuarios. Paradójicamente, muchas organizaciones tienden a centrar sus procesos en las tareas meramente productivas, dejando de lado la calidad en la atención a sus usuarios como si esto fuera algo ajeno a su razón de ser. Este fenómeno provoca una atención deficiente a tales usuarios lo que a su vez crea insatisfacción en los mismos, independientemente de la buena o mala calidad que puedan tener los productos / servicios que la organización expende.

En los últimos años, sin embargo, la mayoría de organizaciones se han concientizado de este problema y han comenzado a poner mayor énfasis en mejorar la calidad de atención a sus usuarios, considerando que las tareas relacionadas a este fin se pueden también considerar procesos, de ahí que se puede hablar de procesos de atención al usuario. Sin embargo, para lograr esta mejorar en la calidad de la atención a los usuarios de las organizaciones, éstas deben comenzar por concientizar a su personal sobre la importancia de tal tarea. A este respecto, se coincide con el autor Rojas [43] quien manifiesta que *"El adoptar un enfoque centrado en el usuario requiere un cambio de actitud. Al*

orientarse hacia el usuario, se reconoce que las inquietudes y preferencias del usuario también son válidas e importantes".

Para entender mejor lo que son los procesos de atención al usuario y la forma de optimizarlos, es necesario tener claro el concepto de lo que son los procesos y los tipos de usuarios a los que éstos pueden estar dirigidos.

3.1.6.1. Procesos

Según Agudelo [44], un proceso es *"Conjunto de actividades secuenciales que realizan una transformación de una serie de inputs en los outputs deseados añadiendo valor"*. En tal virtud, podemos hablar de que un proceso en sí consiste en una tarea de transformación de algo mediante una serie de pasos ordenados, con un fin determinado. Un ejemplo clásico de este concepto consiste en entender los procesos productivos de una fábrica: la misma adquiere insumos y materias primas que los somete a diversos tipos de transformaciones simples y/o complejas, y como resultado obtiene un producto orientado a los intereses y necesidades de sus clientes. Todas estas actividades de transformación, que tienen que seguir pasos exactos y muy bien definidos, consistirían en los procesos productivos de tal fábrica.

Centrándose ya más propiamente en los procesos relacionados a la atención al usuario, el mismo autor [43] manifiesta que un proceso es un *"Conjunto de actuaciones, decisiones, actividades y tareas que se encadenan de forma secuencial y ordenada para conseguir un resultado que satisfaga plenamente los requerimientos del cliente al que va dirigido"*. Esto se aplica de manera particular a las empresas e instituciones dedicadas a servicios, en las que su producto final debe estar especialmente centrado en la satisfacción y el contento de sus clientes, ya que esa será la principal medida de su efectividad.

En este tipo de procesos intervienen como entradas los requisitos necesarios para dar inicio al mismo, los recursos humanos y materiales que tomarán parte en el proceso y los métodos y procedimientos a usarse para lograr el servicio final, el mismo que se busca que sea satisfactorio para el cliente. El mismo autor Agudelo [44] lo expresa gráficamente de la siguiente manera:

Figura 20: Descripción gráfica de un proceso de relacionado a atención a usuarios

Fuente: Gestión por procesos. Bogotá: Instituto Colombiano de Normas Técnicas y Certificación ([44])

3.1.6.2. Atención al usuario y su satisfacción

El usuario es la razón de ser de toda empresa, organización o institución, de manera particular las que se dedican a la prestación de servicios, ya que sin usuarios, los productos o servicios que genera dicha organización o empresa simplemente carecen de sentido, por muy alta que sea su calidad. Por otro lado, en un mundo tan competitivo a nivel corporativo, la satisfacción de sus usuarios debe ser un motivo de especial preocupación para las organizaciones, ya que si un usuario no se siente considerablemente satisfecho con el servicio que recibe, indiscutiblemente tratará de buscar otra organización que le provea productos o servicios similares pero de mejor calidad, y de lo posible a menor costo que la organización inicial.

En las organizaciones se puede distinguir dos tipos de usuarios, los internos y los externos. Los usuarios internos son aquellos que operan dentro de la organización, es decir, trabajan para ella. Los usuarios externos, en cambio, son los clientes de la organización, es decir, quienes acuden a ella en busca de contratar sus servicios o adquirir sus productos. Ambos tipos de usuarios son igualmente importantes para el correcto y saludable funcionamiento de una organización, por lo que los directivos de la misma deben poner especial énfasis en su satisfacción. En este sentido, el autor Rojas [43] expresa que *"Los miembros del personal de una organización pueden considerarse usuarios internos porque reciben productos y servicios esenciales para su trabajo por parte de sus compañeros de trabajo. Los*

gerentes deben escuchar y responder a las necesidades de los proveedores iniciales, supervisores y otros usuarios internos en la misma forma en que escuchan y responden a las necesidades de los usuarios externos…", con lo cual se coincide plenamente.

Además, el mismo autor expresa que *"La satisfacción del usuario es uno de los resultados más importantes de prestar servicios de buena calidad. Dado que la satisfacción del usuario influye de tal manera en su comportamiento, es una meta muy valiosa"*. Esto refuerza la afirmación de que la adecuada satisfacción de sus usuarios es de vital importancia para la supervivencia de cualquier organización, lo cual se hace más evidente conforme sigue pasando el tiempo, ya que el entorno se vuelve cada vez más competitivo a todo nivel y para todo tipo de organizaciones.

3.2. Estado del arte

Como se expone a lo largo de múltiples páginas y secciones de este documento, la información es hoy en día un activo que resulta clave para el correcto y normal funcionamiento y desarrollo de las empresas y organizaciones de toda índole. Las instituciones educativas y, específicamente de educación superior, no escapan a esta realidad, resultando para ellas particularmente importante y necesario el contar con sistemas de administración y procesamiento de su información, los mismos que deben garantizar la seguridad en el acceso y manipulación de dicha información, asegurándose de que únicamente las personas adecuadas deben poder hacerlo.

Las instituciones de educación superior cuentan con múltiples departamentos y áreas que generan y requieren manejar información de diferentes índoles tales como de estudiantes, maestros, carreras, asignaturas, ciclos, facultades, financiera, administrativa y demás. Esto convierte en una necesidad vital el contar con una base de datos centralizada y correctamente diseñada e implementada, a fin de que dicha información se encuentre disponible para quien sea necesario, en el momento que sea necesario, evitando problemas tales como la duplicidad, inconsistencia, desactualización y falta de la misma.

En este sentido, Cano [45], quien realizó un estudio sobre la gestión de la información en varias instituciones universitarias de Latinoamérica, manifiesta que *"La información, la documentación y el conocimiento son recursos que los centros de Educación Superior deben trata de manera prioritaria dentro de los procesos de Formación Profesional, Ciencia y Técnica, y Extensión Universitaria para gestionarlos y explotarlos de manera adecuada. Controlar estos recursos es de suma importancia porque*

inciden en el estatus del alumno y podrían provocar un retroceso en la calidad de la institución". En su estudio, este autor propone la implementación de redes académicas como medios de gestión de la información universitaria y, a la vez, como medios de comunicación que agilicen el intercambio de información entre las distintas áreas de una institución universitaria, la cual obviamente se basa en la red y demás infraestructura informática existente en dicha institución.

Hoy en día, casi toda institución que pueda considerarse una universidad cuenta con un área de posgrados, a fin de ofrecer a sus estudiantes la oportunidad de continuar con sus respectivas carreras y no conformarse con un título de tercer nivel. A nivel mundial, se ha convertido en una exigencia cada vez más imperativa para los profesionales en todas las áreas del conocimiento humano el obtener títulos de cuarto nivel, a fin de ser más competitivos y desempeñarse mejor en las empresas y organizaciones. Por supuesto, las áreas de posgrados de las universidades cuentan con sus oficinas administrativas que generan un flujo de información que requiere un almacenamiento y tratamiento de igual o, probablemente, mayor criticidad en relación a las demás áreas universitarias. Por tanto, para estas áreas es particularmente importante el contar con una base de datos bien diseñada y con aplicaciones que les permitan administrar adecuadamente toda esa información, procesarla y obtener resultados que constituyan un verdadero apoyo para toma de decisiones a nivel administrativo.

A pesar de lo mencionado, existe muy poca evidencia de que las instituciones de educación superior hayan realizado en general esfuerzos para desarrollar herramientas de gestión de procesos de postgrados y su información, tanto a nivel local como regional, lo que ocasiona diferentes tipos de problemas y retrasos en el proceso en general, lo que a su vez ocasiona retrasos a nivel individual, tanto a estudiantes como a docentes y personal administrativo de los mencionados departamentos. Muchas veces, a las universidades les falta ser autocríticas y analizarse suficientemente a sí mismas, a fin de descubrir este tipo de problemas y resolverlos de manera oportuna.

Mencionado esto, se coincide plenamente con el autor Pardo [46], quien en su tesis doctoral manifiesta que *"Las universidades son instituciones con una profunda vocación hacia la investigación, pero pocas veces se animan a estudiarse a sí mismas y a su propio funcionamiento, en pos de adaptarse y evolucionar bajo normas de calidad. Pero dicha calidad es la consecuencia de un proceso. La detección de una estrategia y/o herramienta deficiente obliga a buscar sus causas al interior de la organización. Nadie desarrolla con premeditación un mal producto. Hay razones específicas que determinan los fallos y sus posibles soluciones; la falta de un modelo a seguir es una de ellas. Las buenas prácticas deben reconocerse globalmente, para poder luego imitarlas en productos adaptados a las realidades particulares"*. Como se

ve, el autor pone en evidencia la carencia de un modelo a seguir por las instituciones universitarias para elaborar aplicaciones y bases de datos de calidad, que cumplan eficientemente con los requisitos para los cuales son creadas.

Cabe mencionar que este autor trabajó en la Universidad Autónoma de Barcelona en el año 2005, lo que pone en evidencia que incluso a nivel de otras realidades, como la del continente europeo, la falta de automatización y de un proceso adecuado de gestión de la información universitaria en ese año creaba problemas institucionales de consideración. Por otro lado, el mismo autor manifiesta haber trabajado en Argentina, en la Universidad de Buenos Aires, creando y publicando el sitio web de la Federación Universitaria lo cual, en un inicio pareció tomar el rumbo de un proyecto exitoso pero, a medio proyecto, se vino abajo debido justamente a la falta de una correcta planificación y, como factor decisivo, a la falta de un modelo a seguir para la elaboración de ese tipo de proyecto.

Haciendo un análisis a nivel de Ecuador, se encuentra que existe muy poca información sobre sistemas de automatización universitarios, lo que demuestra que es un problema que se mantiene latente en nuestro medio. Sin embargo, existen ciertos casos como el de los autores Castillo, Fernández, Rea y Tapia [3], quienes elaboran un sistema de gestión académica para el departamento de posgrados de la Universidad de Cuenca denominado SGAP, intentando palear, al menos parcialmente, el problema que se expone. En dicho trabajo, los autores realizan un análisis de los flujos de información del departamento de posgrados de la universidad de Cuenca y desarrollan una aplicación web en Java, integrando su base de datos con la información previamente existente en la institución. Sin embargo, al no existir una estandarización ni una metodología para el desarrollo de aplicaciones, al menos a nivel institucional, tuvieron que migrar varias plataformas de .NET a Java, lo cual es una tarea que consume mucho tiempo y esfuerzo de programación.

Otro caso que es pertinente mencionar dentro del presente análisis es el trabajo del autor Reyes [47], quien también realiza un estudio y elabora una aplicación web para gestión de procesos de posgrado de la Universidad Técnica de Ambato (UTA), apenas en el año 2014, lo que evidencia la inexistencia de tal aplicación hasta la fecha. Tal autor manifiesta en su estudio que *"Actualmente los procesos del manejo de información son realizados de forma manual y en hojas electrónicas, los cuales no abastecen a la gran cantidad de información que se maneja, generando pérdida de la misma, de tiempo, datos duplicados, demora de trámites, desperdicio de recursos humanos y materiales. Por esta razón, cada vez se hace más imprescindible la necesidad de tomar la decisión de desarrollar una aplicación para manejar y almacenar la información que se genera en la Dirección de Posgrado de la UTA, con la cual*

podremos tener un mejor control, mejorar la gestión, optimizando el tiempo y los recursos económicos". Esto deja entrever una vez más la falta de automatización y de aplicaciones de gestión de la información en los departamentos de posgrados de las instituciones de educación superior ecuatorianas a nivel general, y del centro del país en particular.

En torno al análisis realizado en la PUCESA, y haciendo una comparación con los casos citados a nivel mundial, regional, nacional y local, se concluye que lamentablemente esta institución no escapa al citado problema, por lo que resulta necesaria y urgente la elaboración de una aplicación para el manejo adecuado de la información generada por su departamento de posgrados, así como la creación de una base de datos que almacene de forma centralizada y segura dicha información, la cual, como se ha visto en los casos de otras instituciones, deberá integrarse y funcionar en concordancia con los datos ya existentes en la institución, correspondientes a los flujos de datos previamente automatizados.

Capítulo 4

Metodología

La inexistencia de una herramienta informática adecuada para el manejo de la información clave de la OIP de la PUCESA, proceso que hasta el momento se realiza de manera manual, apoyándose en hojas de cálculo de Microsoft Excel, por lo que resulta lento y tedioso y crea malestar tanto para el personal de la OIP como para los maestrantes y docentes de la institución, determina la necesidad urgente de elaborar dicha herramienta para lo cual se realiza el respectivo análisis de flujos de información y de bases de datos ya existentes en los correspondientes departamentos de la institución.

Por otro lado, cabe mencionar que se decide elaborar una aplicación web para los fines mencionados, ya que este tipo de aplicación presenta múltiples ventajas para los potenciales usuarios. En primer lugar, una aplicación web puede ser utilizada con total independencia de entornos de hardware y software desde el punto de vista del usuario, quien únicamente requiere de un navegador web estándar para utilizarla, pudiendo hacerlo desde su computador de escritorio, portátil, tableta e incluso teléfono móvil, siempre que éste soporte la navegación web. En segundo lugar, existe también independencia del lugar desde donde se utilice la aplicación, ya que el usuario lo puede hacer tanto desde su hogar como desde su trabajo, desde la misma institución, desde cualquier biblioteca o cibercafé o, como se menciona, virtualmente desde cualquier lugar a través de dispositivos móviles, siendo los únicos requisitos una conexión a Internet y un navegador de Internet funcional. Desde luego, de ser necesario las aplicaciones web pueden ser configuradas para restringir el acceso a sus usuarios desde determinados sitios o tipos de dispositivos.

Además, es necesario también citar que la institución cuenta con toda la infraestructura técnica y tecnológica requerida para el desarrollo e implementación de aplicaciones web y bases de datos, debiéndose mencionar que existen ya varias de ellas funcionando en la institución como parte de los procesos de automatización parciales de otras de sus áreas. Desde el punto de vista del equipo de diseño y desarrollo de la aplicación, será necesario adaptarse a las herramientas tecnológicas existentes en la institución, puesto que es de vital importancia lograr obtener una aplicación que se

integre satisfactoriamente y desde todos los puntos de vista con aquellas que ya se encuentran en marcha.

Hechas todas estas consideraciones, se decide utilizar la metodología de desarrollo de software XP, ya que al ser una metodología ágil ofrece múltiples ventajas como velocidad de desarrollo, entregas periódicas y adaptabilidad a cambios en los requerimientos, como se puede apreciar con mayor detalle en la sección 3.1.3.2.2. del capítulo III.

Por último, es necesario mencionar que dentro de la metodología seleccionada para el desarrollo del presente proyecto, se requiere de la colaboración activa del personal de la institución, en particular del personal administrativo de la OIP, quien deberá facilitar toda la información requerida y sus flujos, y por otra parte del área de informática de la institución, quienes deberán facilitar la información técnica requerida como bases de datos ya existentes, así como también brindar el soporte que necesariamente se requerirá para el desarrollo de la presente aplicación.

4.1. Diagnóstico

Una vez que se observan los problemas presentes en la OIP debido a la ausencia de una aplicación que maneje y centralice adecuadamente su información, se procede a tener una primera reunión con el personal administrativo del departamento, en su momento con el Dr. Juan Mayorga Zambrano, como director del mismo, y la Lcda. Miriam Mayorga, en su calidad de secretaria. Esta reunión sirve para que ellos manifiesten su disposición a resolver estos problemas utilizando los recursos tecnológicos de los que dispone la PUCESA, habiendo observado que otros departamentos y áreas de la institución ya lo han hecho exitosamente en el pasado. Para esto, se comprometen a brindar toda la información y el tiempo que sean necesarios para poder establecer los requerimientos técnicos y demás bases necesarias para iniciar con la ejecución del proyecto de desarrollo de software.

Con este fin, se programan reuniones periódicas en las mismas oficinas de la OIP para ir identificando detalladamente los procesos que se llevan a cabo y de este modo ir definiendo tales requerimientos, que se proceden a registrar técnicamente. La mayor parte de los problemas que se identifican en ese momento están relacionados a la lentitud en las tareas de atención a los usuarios externos, debido justamente a que todos aquellos procesos se realizan manualmente y, pese a que el departamento cuenta con un computador que se utiliza sobre todo para la elaboración de ciertos documentos y para el registro de ciertos datos utilizando los programas Microsoft Word y Microsoft

Excel respectivamente, la información crítica de los procesos de posgrado en relación a la atención a los usuarios externos no se encuentra disponible de manera adecuada, por lo que hay que localizarla y revisarla manualmente.

En relación a los usuarios internos, que en este caso son las mismas personas ya mencionadas, quienes tienen a su cargo el manejo y administración de la OIP, también existen tareas que representan cuellos de botella y pérdidas de tiempo para ellos, que requieren ser automatizadas y su información requiere ser correctamente almacenadas en bases de datos específicamente diseñadas para tal fin. Por ejemplo, en ese momento el director del departamento elaboraba sus distributivos de docentes y asignaturas de posgrado de manera manual, apenas apoyándose en una hoja de Excel que, sin embargo, le obligaba a revisar manualmente la información de horarios de disponibilidad de los docentes para asignarlos a las materias evitando los cruces de horarios, una tarea bastante tediosa y proclive a errores.

Posteriormente, sale de la institución el mencionado director del departamento y llega en su reemplazo la Dra. Varna Hernández Junco, con quien también se realiza varias entrevistas personales para definir nuevos requerimientos. Entre estos, la nueva directora menciona que otro de los procesos que aún se realiza manualmente es el de determinar el estado en el que se encuentra un determinado maestrante, ya que aún no existe la aplicación que pueda desplegar esa información automáticamente. Este proceso implica tener que revisar manualmente la carpeta del maestrante, ver los documentos y solicitudes que ha presentado, buscar cuáles ya le han sido aprobados, determinar si el maestrante requiere ya de tomar actualizaciones y demás revisiones, para al fin poder determinar en qué situación se encuentra el mismo y cuál es el siguiente paso que debe dar en su proceso de graduación.

Un caso similar sucede con los docentes, quienes para saber en cuántos proyectos se encuentran trabajando al momento ya sea en calidad de tutor, director del proyecto, decente informante o docente revisor, había que contarlo manualmente apoyándose también en la información de una hoja de Microsoft Excel, la cual además podía estar desactualizada lo que a su vez podía llevar a información inconsistente y, en el mejor de los casos, apenas aproximada.

Finalmente, la Dra. Hernández deja también su cargo como directora del departamento y en su lugar ingresa el Mtr. Diego Jiménez, quien ingresa en calidad de coordinador del departamento que ahora pasa a llamarse Oficina de Investigación y Posgrados (OIP). Con él también se procede a tener varias reuniones donde se define aún más requisitos, como la necesidad de contar con la generación automática de cuadros estadísticos que desplieguen información clave que sirva como herramienta

para la toma de decisiones desde un punto de vista gerencial, basándose en información actual y consistente, y que se mantenga actualizada en tiempo real.

A lo largo de todas estas entrevistas con las diferentes personas que fueron formando parte del personal administrativo del OIP, se definieron los requerimientos definitivos que era necesario plasmar en la aplicación a desarrollarse, y posteriormente, se procedió a definir la urgencia tanto técnica como operativa con la que debía implementarse cada uno de ellos, procediéndose entonces a definirse el plan de trabajo a seguir durante el desarrollo de la presente aplicación web.

Finalmente, es necesario indicar que la aplicación a desarrollarse en el presente proyecto guarda total relación y es complementaria con otra [1] que se desarrolla paralelamente para el OIP, cuya misión es la de automatizar y gestionar la información correspondiente a los procesos de graduación de posgrados. Es muy importante tomar en cuenta que ambas aplicaciones utilizan las mismas bases de datos y comparten muchos de los webservices creados por parte del personal del departamento de informática de la PUCESA, como parte de su compromiso de apoyo técnico para la exitosa ejecución de ambos proyectos.

4.2. Métodos aplicados

Desde el punto de vista científico, los métodos que se utilizan para recolectar la información necesaria para delimitar el problema y definir los requerimientos que van a ser resueltos mediante la aplicación producto del presente trabajo son la observación directa y las entrevistas con el personal administrativo del OIP. Al no requerirse el desarrollo de módulos destinados a maestrantes o docentes, no se encuentra necesaria la utilización de otros métodos de recolección de información como podrían ser las encuestas de satisfacción, si bien al final del proceso se aplica una breve encuesta al personal del OIP con la finalidad de estimar su grado de satisfacción con la aplicación elaborada.

Desde el punto de vista de una metodología específica de desarrollo de software, se opta por la metodología XP, cuyas bases teóricas se exponen en el capítulo anterior. El motivo para haber elegido tal metodología es debido a que es una metodología ágil que es la que mejor se ajusta a las características y naturaleza del proyecto a desarrollarse, sobre todo considerando que se trata de una aplicación web que requiere completarse y ponerse en producción de manera rápida, con un alto grado de satisfacción por parte del cliente (representado en este caso por el personal administrativo del OIP), y a que se requiere de la participación activa del cliente como parte del equipo de desarrollo de

software. Otro factor determinante es la flexibilidad que ofrece esta metodología en relación a los cambios de requerimientos durante el proyecto, factor que, como se expone en la sección anterior, se dio varias veces durante la fase de definición de los mismos.

4.2.1. Desarrollo de la aplicación web mediante la metodología XP

Como se expone en el capítulo 3, la metodología XP requiere de la formación de un equipo de trabajo donde uno o dos representantes del cliente forman parte activa del mismo, y fomenta las buenas relaciones entre todos los miembros del mismo, como estrategia para crear un ambiente de confianza donde la información requerida por todos fluya libremente, el código creado sea de propiedad común entre los programadores, se realicen constantemente pruebas sobre el correcto funcionamiento de los módulos que se va creando, asegurándose de la satisfacción del cliente en todo momento, y donde se pueda poner en práctica los principios del modelo de desarrollo iterativo e incremental, que buscan ir respondiendo de manera inmediata al siguiente requerimiento más urgente que, a su vez, es definido por el cliente.

Tomando en cuenta lo anterior, se procede a definir el equipo de trabajo para el desarrollo del presente proyecto, considerando que algunos de los integrantes para distintos roles han ido cambiando a lo largo del tiempo. Específicamente, el Dr. Juan Mayorga fue el cliente inicial conjuntamente con la Lcda. Miriam Mayorga en su calidad de director y secretaria de la OIP respectivamente, quienes en lo posterior fueron reemplazados por la Dra. Varna Hernández y la Lcda. Doris Rosales. La Dra. Varna Hernández, a su vez, fue reemplazada posteriormente por el Prof. Diego Jiménez. De igual forma, al inicio del proyecto el Ing. César Guevara es el director del departamento de informática de la PUCESA, por lo que las conversaciones y definición de políticas iniciales se realiza con él, sin embargo, en lo posterior es reemplazado por el Ing. Gabriel Altamirano con quien también se requiere realizar múltiples conversaciones y de quien se requiere mucha ayuda. En la siguiente tabla se encuentran mencionados todos los miembros que participan o han participado como parte del equipo de desarrollo para el presente proyecto. Se sombrean los integrantes con los que se finaliza e implementa el proyecto.

Tabla 4: Equipo de trabajo para el desarrollo del proyecto

Integrante	***Roles***
Dr. Juan Mayorga	• Cliente
Dra. Varna Hernández	• Cliente
Prof. Diego Jiménez	• Cliente
Lcda. Miriam Mayorga	• Cliente • Usuario final
Lcda. Doris Rosales	• Cliente • Usuario final
Ing. Ricardo Soria	• Gerente del Proyecto • Programador
Ing. Gabriela De la Torre	• Programador
Ing. Mg. Marcelo Balseca	• Co-gerente del Proyecto • Tester
Ing. Mg. José Enríquez	• Consultor • Personal de apoyo del Departamento de Informática
Ing. César Guevara	• Consultor • Personal de apoyo del Departamento de Informática
Ing. Gabriel Altamirano	• Consultor • Personal de apoyo del Departamento de Informática

Fuente: Elaboración propia

Una vez conformado el equipo de trabajo, se procede a desarrollar cada una de las fases de la metodología XP, elegida como la más óptima considerando las características del presente proyecto, por los motivos que se expone anteriormente.

4.2.1.1. Fase de planeación

En este punto ya se ha ejecutado parte de la fase de planeación dentro de los lineamientos establecidos para el desarrollo del presente proyecto, puesto que ya se ha realizado una primera reunión de carácter introductorio con las personas involucradas de algún modo dentro del mismo. Como resultado de esa primera reunión se ha establecido el equipo de trabajo que se detalla más arriba, habiéndose socializado con todo el grupo el objetivo general y los objetivos específicos que se persigue con este proyecto, así como las funciones encomendadas a todos y cada uno de los integrantes. Todos ellos, a su vez, se han comprometido a facilitar la información requerida, colaborar con las funciones a ellos encomendadas y apoyar dentro de lo que esté a su alcance con el desarrollo del proyecto, que es de amplio interés social dentro de la institución.

Dentro de esta primera reunión se procede también a planificar reuniones periódicas a razón de dos veces por semana en la etapa inicial, donde se elabora la primera versión de las historias de usuario y el cliente define aquellas más críticas y urgentes, es decir los requerimientos que necesita que sean implementados con prioridad. Como se menciona en párrafos anteriores, el personal administrativo de la OIP cambió durante dos ocasiones y, en consecuencia, los requerimientos del proyecto también cambiaron, algunos de ellos drásticamente, lo cual implica también cambios en las historias de usuario, que se elaboran en esta fase de la metodología. Como se explicó en el capítulo teórico, las historias de usuario están sujetas a cambios en cualquier momento, e incluso a la eliminación completa de una o más de ellas y su consecuente reemplazo por otra u otras. Por este motivo resultó especialmente útil una metodología de desarrollo de software como XP, ya que está diseñada para adaptarse con relativa facilidad a los cambios en los requerimientos en cualquier punto de la evolución del proyecto.

4.2.1.1.1. Historias de usuario

A continuación, se procede a listar la versión final de las historias de usuario, las mismas que corresponden a la entrega final de la aplicación resultante del presente proyecto, tal y como éste se describe en el capítulo de resultados.

Tabla 5: Historia de usuario No. 1

Historia de Usuario			
Número:	1	**Usuario:**	Secretaria/Director DIP
Nombre de la Historia:	Registro de maestrantes activos		
Prioridad en Negocio:	Alta	**Riesgo en desarrollo:**	Alto
Puntos estimados:	2	**Iteración asignada:**	1
Descripción:			
Se requiere de una interfaz que permita dar de alta un maestrante activo en la aplicación, registrándose con exactitud el paso en el que se encuentra dentro de su proceso de graduación y que permita ingresar toda la información requerida de manera ágil y precisa			
Observaciones:			

Fuente: Elaboración propia

Tabla 6: Historia de usuario No. 2

Historia de Usuario			
Número:	2	**Usuario:**	Secretaria/Director DIP
Nombre de la Historia:	Lista de maestrantes activos		
Prioridad en Negocio:	Alta	**Riesgo en desarrollo:**	Medio
Puntos estimados:	1	**Iteración asignada:**	1
Descripción:			
Se debe generar una lista de maestrantes activos que despliegue información detallada sobre los mismos y sus procesos de graduación incluyendo el número de matrícula, fecha de egresamiento, tema, nombres de docentes director, informante y revisores, estado actual del proyecto y porcentaje de avance del mismo, a nivel general y por carreras			
Observaciones:			

Fuente: Elaboración propia

Tabla 7: Historia de usuario No. 3

Historia de Usuario			
Número:	3	**Usuario:**	Secretaria/Director DIP
Nombre de la Historia:	Lista de maestrantes pasivos		
Prioridad en Negocio:	Alta	**Riesgo en desarrollo:**	Bajo
Puntos estimados:	1	**Iteración asignada:**	2
Descripción:			
Se debe generar una lista de todos los maestrantes pasivos en el mismo formato de la lista de maestrantes activos, tanto a nivel general como por carreras			
Observaciones:			

Fuente: Elaboración propia

Tabla 8: Historia de usuario No. 4

Historia de Usuario			
Número:	4	**Usuario:**	Secretaria/Director DIP
Nombre de la Historia:	Lista de maestrantes graduados		
Prioridad en Negocio:	Media	**Riesgo en desarrollo:**	Bajo
Puntos estimados:	1	**Iteración asignada:**	5
Descripción:			
Se requiere generar una lista con los maestrantes graduados, desplegando la misma información que en los casos anteriores, a nivel general y por carreras			
Observaciones:			

Fuente: Elaboración propia

Tabla 9: Historia de usuario No. 5

Historia de Usuario			
Número:	5	**Usuario:**	Secretaria/Director DIP
Nombre de la Historia:	Lista de maestrantes por estado		
Prioridad en Negocio:	Alta	**Riesgo en desarrollo:**	Bajo
Puntos estimados:	1	**Iteración asignada:**	2
Descripción:			
Al igual que en los casos anteriores, se requiere una lista de los maestrantes que se encuentran en determinado estado dentro de su proceso de graduación, para lo cual se puede elegir el estado. Se despliega también la información completa del maestrante y su proceso de graduación, y debe listar los maestrantes en general y por carreras			
Observaciones:			

Fuente: Elaboración propia

Tabla 10: Historia de usuario No. 6

Historia de Usuario			
Número:	6	**Usuario:**	Secretaria/Director DIP
Nombre de la Historia:	Lista de maestrantes que requieren tomar actualizaciones		
Prioridad en Negocio:	Alta	**Riesgo en desarrollo:**	Alto
Puntos estimados:	2	**Iteración asignada:**	3
Descripción:			
Se requiere también una lista de los maestrantes que requieren tomar actualizaciones, tanto a nivel general como por carreras, indicando toda la información pertinente, sobre todo la fecha de egresamiento y/o de última actualización. Deben aparecer los maestrantes para los que haya pasado más de un año desde dicha fecha			
Observaciones:			

Fuente: Elaboración propia

Tabla 11: Historia de usuario No. 7

Historia de Usuario			
Número:	7	**Usuario:**	Secretaria/Director DIP
Nombre de la Historia:	Desplegar datos de un maestrante		
Prioridad en Negocio:	Alta	**Riesgo en desarrollo:**	Medio
Puntos estimados:	1	**Iteración asignada:**	5
Descripción:			
Se debe desplegar toda la información referente un maestrante específico, que deberá ser posible elegirlo de una lista. Despliega toda la información detallada del mismo incluyendo email, fecha de egresamiento y/o última actualización, si debe tomar actualizaciones o no, estado actual y siguiente paso en el proceso de graduación			
Observaciones:			

Fuente: Elaboración propia

Tabla 12: Historia de usuario No. 8

Historia de Usuario			
Número:	8	**Usuario:**	Secretaria/Director DIP
Nombre de la Historia:	Desplegar los datos de un docente		
Prioridad en Negocio:	Media	**Riesgo en desarrollo:**	Bajo
Puntos estimados:	1	**Iteración asignada:**	6
Descripción:			
Se debe desplegar la información detallada de un docente que se podrá elegir de una lista, incluyendo su email y el número de proyectos en los que está trabajando actualmente como tutor, director, informante o revisor, y el total de proyectos en los que trabaja al momento			
Observaciones:			

Fuente: Elaboración propia

Tabla 13: Historia de usuario No. 9

Historia de Usuario			
Número:	9	**Usuario:**	Secretaria/Director DIP
Nombre de la Historia:	Registrar porcentajes de avance para los proyectos		
Prioridad en Negocio:	Alta	**Riesgo en desarrollo:**	Bajo
Puntos estimados:	1	**Iteración asignada:**	5
Descripción:			
Se debe registrar el porcentaje de avance del proyecto de graduación de un maestrante, siempre y cuando se encuentre en fase de desarrollo, para lo cual el mismo debe acercarse al OIP portando un certificado del porcentaje emitido por su director de proyecto			
Observaciones:			

Fuente: Elaboración propia

Tabla 14: Historia de usuario No. 10

Historia de Usuario			
Número:	10	**Usuario:**	Secretaria/Director DIP
Nombre de la Historia:	Registro de emails de maestrantes y docentes		
Prioridad en Negocio:	Baja	**Riesgo en desarrollo:**	Bajo
Puntos estimados:	1	**Iteración asignada:**	6
Descripción:			
Se debe guardar un registro de las direcciones actualizadas de correo electrónico tanto de maestrantes como de docentes, ya que frecuentemente se requiere enviarles correos con información de su interés			
Observaciones:			

Fuente: Elaboración propia

Tabla 15: Historia de usuario No. 11

Historia de Usuario			
Número:	11	**Usuario:**	Secretaria/Director DIP
Nombre de la Historia:	Actualización de parámetros de la aplicación		
Prioridad en Negocio:	Baja	**Riesgo en desarrollo:**	Bajo
Puntos estimados:	1	**Iteración asignada:**	6
Descripción:			
Se debe poder actualizar ciertos parámetros de índole general dentro de la aplicación tales como título, nombre y cargo del director del OIP, y nombre de la secretaria del OIP. También se debe permitir cambiar la contraseña de acceso de la secretaria del OIP			
Observaciones:			

Fuente: Elaboración propia

Tabla 16: Historia de usuario No. 12

Historia de Usuario			
Número:	12	**Usuario:**	Secretaria/Director DIP
Nombre de la Historia:	Gráfico de maestrantes graduados por carrera		
Prioridad en Negocio:	Alta	**Riesgo en desarrollo:**	Medio
Puntos estimados:	1	**Iteración asignada:**	3
Descripción:			
Se debe generar un gráfico estadístico que indique el número de maestrantes graduados por carreras, tanto a nivel general como por un rango de fechas			
Observaciones:			

Fuente: Elaboración propia

Tabla 17: Historia de usuario No. 13

Historia de Usuario			
Número:	13	**Usuario:**	Secretaria/Director DIP
Nombre de la Historia:	Gráfico de situación de maestrantes por proceso		
Prioridad en Negocio:	Alta	**Riesgo en desarrollo:**	Medio
Puntos estimados:	2	**Iteración asignada:**	4
Descripción:			
Se debe generar un gráfico que indique el número de maestrantes que se encuentras dentro de cada subproceso general de su proceso de graduación, incluyendo en tema, plan, desarrollo del proyecto, en proceso de docentes revisores y en proceso de requisitos para graduación			
Observaciones:			

Fuente: Elaboración propia

Tabla 18: Historia de usuario No. 14

Historia de Usuario			
Número:	14	**Usuario:**	Secretaria/Director DIP
Nombre de la Historia:	Gráfico de situación de maestranes por condición		
Prioridad en Negocio:	Alta	**Riesgo en desarrollo:**	Medio
Puntos estimados:	1	**Iteración asignada:**	4
Descripción:			
Se generará también un gráfico que muestre el número de maestrantes que se encuentren en cada condición general, es decir, activos, pasivos, que requieren tomar actualizaciones y graduados, tanto a nivel general como por carreras			
Observaciones:			

Fuente: Elaboración propia

4.2.1.1.2. Plan de entregas

Siguiendo con la estructura definida en la metodología XP, otro punto clave dentro de la fase de planeación es la elaboración del plan de entregas, el mismo que trata de definir las historias de usuarios que se van a implementar en cada iteración durante la fase de codificación y, por tanto, lo que se va a desarrollar para cada versión de la aplicación. En este punto es también muy importante la presencia y la colaboración activa del cliente, ya que es éste quien por un lado negociará los tiempos de entrega de cada iteración y, por otro lado, definirá sus prioridades estableciendo qué historias de usuario se deben implementar primero.

Cabe mencionar también que en esta etapa, se trata de realizar un estimado lo más aproximado posible de los tiempos de entrega, sin embargo, durante en desarrollo del proyecto, sobre todo en la fase de codificación, pueden surgir múltiples inconvenientes que a su vez pueden ocasionar retrasos en las entregas de ciertas versiones de la aplicación, lo cual es un aspecto normal en muchos tipos de proyectos, y que se debe tomar en cuenta. Por lo tanto, es necesario también alertar al cliente sobre

este particular ya que por regla general, éste espera que los tiempos de entrega sean estrictamente ajustados a la planificación inicial y, de posible, incluso menos.

Hay que decir también que, así como las historias de usuario pueden ir cambiando conforme cambian los requerimientos establecidos inicialmente por el cliente, así también puede ir cambiando el plan de entregas, ya que el mismo está basado en las historias de usuario. Por consiguiente, un cambio de consideración en las historias de usuario se reflejará automáticamente en un cambio en el plan de entregas.

Por último, para la elaboración del plan de entregas es muy importante tomar en cuenta la recomendación de que cada iteración no debería sobrepasar las tres semanas de duración, lo cual es uno de los principios de la metodología XP, ya que de lo contrario se puede perder efectividad en la comunicación con el cliente y entre los demás miembros del equipo.

En la siguiente tabla se observa el plan de entregas final elaborado para la presente aplicación, el mismo que guarda total relación con las historias de usuario detalladas anteriormente.

Tabla 19: Versión final del plan de entregas

No.	Nombre de la historia	Tiempo estimado			Iteraciones					
		Semanas	Días	Horas	1	2	3	4	5	6
1	Registro de maestrantes activos	2	10	30						
2	Lista de maestrantes activos	1	5	15						
3	Lista de maestrantes pasivos	1	5	15						
5	Lista de maestrantes por estado	1	5	15						
6	Lista de maestrantes que requieren tomar actualizaciones	2	10	30						
12	Gráfico de maestrantes graduados por carrera	1	5	15						
13	Gráfico de situación de maestrantes por proceso	2	10	30						
14	Gráfico de situación de maestrantes por condición	1	5	15						
7	Desplegar datos de un maestrante	1	5	15						
9	Registrar porcentajes de avance para los proyectos	1	5	15						
4	Lista de maestrantes graduados	1	5	15						
8	Desplegar los datos de un docente	1	5	15						
10	Registro de emails de maestrantes y docentes	1	10	30						
11	Actualización de parámetros de la aplicación	1	5	15						

Fuente: Elaboración propia

4.2.1.1.3. Velocidad del proyecto

Otro de los preceptos de la metodología XP es la definición de la velocidad del proyecto, el cual está obviamente relacionado con la estimación de los tiempos de entrega en el plan de entregas. La medida de la velocidad del proyecto pretende establecer una estimación del número de historias de usuario que se puede implementar en una unidad determinada de tiempo, típicamente semanas o meses, dependiendo del tamaño del proyecto.

Partiendo del plan de entregas expuesto más arriba, se observa que lo planificado es implementar un total de catorce historias de usuario en un total de diecisiete semanas. Realizando una simple división se puede calcular la velocidad del proyecto, de acuerdo a la siguiente fórmula.

Figura 21: Fórmula de cálculo de la velocidad del proyecto

$$v = \frac{Número\ de\ historia\ de\ usuario}{Unidad\ de\ tiempo}$$

Fuente: Elaboración propia

Por consiguiente, para el siguiente proyecto se obtiene una velocidad estimada de **0.82** historias de usuario por semana, para un total estimado de 17 semanas de desarrollo del proyecto.

Sin embargo, es importante recalcar que esta medida es simplemente una estimación que puede variar durante el desarrollo del proyecto, sobre todo si existe una alta tendencia a las variaciones en los requerimientos inicialmente definidos. Por esta razón, la recomendación es ir recalculando la velocidad del proyecto periódicamente, lo ideal sería cada 3 o 4 historias de usuario implementadas, y cada vez que surja una modificación en un requerimiento o nuevos requerimientos lo cual, obviamente, obliga a realizar nuevas estimaciones de tiempo.

4.2.1.2. Fase de diseño

Una vez establecidas correctamente las historias de usuario y elaborado el plan de entregas se puede proceder con la fase de diseño, donde se debe considerar diversos aspectos dentro del desarrollo de la aplicación, incluyendo el diseño de interfaces, clases, bases de datos y protocolos de programación que deberán seguir los programadores, en particular cuando hay varios de ellos formando parte del equipo de desarrollo.

Un aspecto muy importante a considerarse dentro de esta fase de diseño para el presente proyecto son las entrevistas que se realizaron con el personal del departamento de informática de la institución, en particular con el Ing. Mg. José Enríquez quien, en su calidad de administrador de la base de datos institucional, fue la persona designada para exponer con detalles la información ya existente en dicha base de datos. Esta información incluía datos referentes a estudiantes, docentes, carreras de posgrados, ciclos, asignaturas de posgrados y matrículas de estudiantes. En la primera reunión con el Ing. Mg. Enríquez se determinó que el mecanismo de acceso a la información ya existente que pudiera resultar de interés para el proyecto sería a través de webservices que el mencionado profesional crearía cuando fuere necesario.

Cabe especificar que los webservices son métodos programados por el administrador de la base de datos institucional para devolver ciertos grupos de datos que a su vez pueden ser utilizados por la aplicación a desarrollarse en el presente proyecto, ya sea mediante el envío de parámetros o no. Tanto las solicitudes hacia dichos métodos como la información que estos devuelven viajan a través de Internet, por lo que son accesibles desde cualquier computador con acceso a Internet.

Por ejemplo, los webservices pueden devolver información como la lista de maestrantes, la lista de docentes tanto internos como externos, la lista de carreras de posgrados, y la lista de asignaturas correspondientes a un determinado nivel y carrera de posgrados. Los webservices pueden también servir para validar información, por ejemplo, un usuario y contraseña enviados por el usuario para determinar si se permite el acceso a dicho usuario hacia la aplicación o no. En el apéndice C se procede a describir los webservices habilitados por el Ing. Mg. Enríquez para facilitar el desarrollo de la presente aplicación.

En las siguientes reuniones con el Ing. Mg. Enríquez se procedió también a analizar el funcionamiento y estructura de la plataforma institucional conocida como “Academics” [4], lo que resulta de vital importancia para el desarrollo del presente proyecto, ya que el mismo se considerará

una extensión de Academics y, por lo tanto, deberá integrarse enteramente con la misma, siendo íntegramente compatible con ella. Se procede a analizar y estudiar las herramientas de diseño y desarrollo de software y bases de datos utilizadas para la elaboración de Academics ya que, para asegurar la compatibilidad con la misma, lo más aconsejable es usar las mismas herramientas, en sus mismas versiones y ediciones.

En base a este análisis se determina que las herramientas y lenguajes a utilizarse para el desarrollo de la presente aplicación son ASP.NET 4.0 mediante Microsoft Visual Studio 2010, C# como lenguaje de programación de *code behind*, y Microsoft SQL Server 2008 R2 como SGBD. Si bien las mencionadas no son las últimas versiones de las plataformas a utilizarse para el desarrollo, son las que aseguran compatibilidad con Academics y las bases de datos ya existentes en la institución. Se utilizan también ciertos paquetes adicionales como los controles *Obout*, que son un complemento para Microsoft Visual Studio que permitirá guardar una compatibilidad desde el punto de vista visual y de diseño con Academics.

Por otro lado, en este punto cabe recalcar que el presente proyecto guarda total relación con otro titulado "Desarrollo de una aplicación web para la administración de procesos de graduación de una unidad de posgrados" [1], el mismo que se desarrolla paralelamente y es de similares características a este y también obtiene como producto final una aplicación compatible con Academics desarrollada utilizando exactamente las mismas herramientas y versiones ya descritas. Ambas aplicaciones comparten muchos de los datos ya existentes en la institución y por ende los mismos webservices y, por otro lado, si bien cada aplicación diseñará e implementará su propia base de datos, ambas bases de datos terminarán conteniendo información utilizable por ambas aplicaciones.

De lo expuesto se deduce que ambas aplicaciones también deben ser compatibles entre sí y deben poder integrarse sin problemas, ya que ambas se implementarán en su momento como extensiones de Academics. Por tanto, el desarrollo de ambas aplicaciones se deberá realizar de forma sincronizada y, para lograr esto, muchas de las reuniones que se realizan tanto con el personal administrativo del OIP como con el personal técnico del departamento de informática se llevan a cabo conjuntamente con la Ing. Gabriela De la Torre, autora del otro proyecto en cuestión.

Si bien para el desarrollo del presente proyecto el personal del departamento de TI de la PUCESA no ha autorizado el libre acceso a las bases de datos del Academics, ha facilitado el acceso a su información requerida mediante webservices, cuyo detalle se puede ver en el apéndice C.

En cuanto al diseño de clases, para obtenerlo se apoya en la elaboración de las tarjetas CRC, las mismas que se detallan en la siguiente sección. Como resultado, se procede a elaborar el modelo entidad – relación para la presente aplicación, el mismo que se puede observar en la figura 32 del capítulo V. Sin embargo, hay que considerar que también se requerirá tener conocimiento sobre el modelo entidad – relación para la aplicación paralela en desarrollo, por lo que el mismo se incluye en la figura 31 del mismo capítulo V.

Por último, dentro de lo que es el diseño de interfaces, se procede a definir un bosquejo de lo que será la parte visual de las páginas web a incluirse en la presente aplicación, las mismas que deberán ceñirse estrictamente a dicho bosquejo. Éste ha sido elaborado considerando un diseño simple, agradable e intuitivo para el usuario final de la aplicación, así como una gama de colores compatibles con los institucionales y con Academics. El bosquejo en cuestión queda definido de la siguiente manera, tomándose en cuenta que los textos incluidos entre corchetes serán reemplazados por el contenido real de las páginas web:

Figura 22: Bosquejo diseñado para las páginas web

Fuente: Elaboración propia

4.2.1.2.1. Tarjetas Clase – Responsabilidad – Colaboración (CRC)

Una vez que concluye el análisis de los datos ya existentes en la institución, de los webservices que van a ser habilitados por el departamento de informática y de los requerimientos establecidos por el personal del OIP, de donde se obtuvieron las historias de usuario expuestas previamente, se procede a elaborar las tarjetas CRC, las mismas que son pilar fundamental para el diseño de las clases a implementarse dentro de la aplicación, considerando que se va a utilizar un lenguaje y una plataforma de desarrollo de software netamente orientados a objetos. Estas tarjetas son la base que define las clases a utilizarse durante todo el desarrollo de la aplicación, en especial para el diseño de las bases de datos de la misma.

A continuación, se expone las tarjetas CRC resultantes del análisis previo.

Figura 23: Tarjeta CRC Estudiante

Estudiante	
Guardar información detallada	TemadeTesis
sobre los maestrantes y su es-	PlandeTesis
tado general y dentro del pro-	AvancesTesis
ceso de graduación	TesisPosgrados
Registrar dirección de e-mail	

Fuente: Elaboración propia

Figura 24: Tarjeta CRC Docente

Docente	
Desplegar información sobre	TemadeTesis
los docentes y número de pro-	PlandeTesis
yectos en los que trabaja	AvancesTesis
Registrar dirección de e-mail	TesisPosgrados
Emitir informes sobre los avan-	
ces en los proyectos	

Fuente: Elaboración propia

Figura 25: Tarjeta CRC TemadeTesis

TemadeTesis	
Guardar información sobre	Docente
temas de proyectos incluyendo	Estudiante
resumen, tutor, estudiante,	
estado y fecha de aprobación	

Fuente: Elaboración propia

Figura 26: Tarjeta CRC PlandeTesis

PlandeTesis	
Guardar información sobre	Docente
planes de proyectos incluyendo	Estudiante
fecha de aprobación, estado,	TemadeTesis
docente informante y tema con	
el que se relaciona	

Fuente: Elaboración propia

Figura 27: Tarjeta CRC AvancesTesis

AvancesTesis	
Guardar avances del IFP inclu-	TesisPosgrados
yendo porcentajes, fechas de	
aprobación y estados	

Fuente: Elaboración propia

Figura 28: Tarjeta CRC TesisPosgrados

TesisPosgrados	
Guardar información sobre proyectos de titulación incluyendo fechas de los diferentes eventos, docentes revisores, plan con el que se relaciona y estado	PlandeTesis AvancesTesis Estudiante Docente

Fuente: Elaboración propia

Figura 29: Tarjeta CRC Parametros

Parametros	
Guarda información sobre parámetros generales de la aplicación, incluyendo nombre y cargos del personal administrativo	

Fuente: Elaboración propia

4.2.1.3. Fase de codificación

Siendo la siguiente fase dentro de la metodología XP la que corresponde a la codificación, donde se procede ya directamente con la creación del código para cada iteración definida, procurando cumplir con los tiempos programados de entrega y, sobre todo, asegurándose de que se cumplen con todos los requisitos establecidos por el cliente y de su satisfacción, ya que cada entrega debe pasar por las pruebas de usuario, donde el cliente puede verificar la funcionalidad del producto creado y dar su visto bueno para poder solo entonces continuar con la siguiente iteración.

Se entra directamente a desarrollar y programar cada una de las historias de usuario establecidas para el proyecto, respetando al máximo la planificación que quedó plasmada en el plan de entregas, sobre todo la prioridad de las historias a ser implementadas prioritariamente, lo cual fue establecido por el mismo cliente. Cuando se completa la codificación de todos los puntos establecidos para una determinada iteración se procede a presentársela al cliente y, de ser el caso, se anota sus observaciones y se procede a realizar correcciones las veces que sea necesario hasta lograr la funcionalidad y satisfacción demandadas por el cliente. Una vez que cada módulo es completamente satisfactorio, se realiza una breve revisión de su código para optimizarlo en la medida posible con la finalidad de obtener un código limpio y legible. Este paso es necesario porque en muchas ocasiones el código resultante en esta fase contiene parches y modificaciones que responden a correcciones o cambios de última hora en los requisitos solicitados por el cliente, sin embargo, estos parches pueden crear código inconsistente, redundante o que de una u otra manera se puede mejorar.

Cabe mencionar que para el presente proyecto se comienza diseñando y creando la base de datos propia de la aplicación, sin embargo, hay que tener en cuenta que se requiere también del uso paralelo de otra base de datos perteneciente a otro proyecto de similares características, por lo que en muchos puntos es necesario analizar el diseño de la misma, la cual incluso influye considerablemente en el diseño de la base de datos de la presente aplicación. Además de ambas bases de datos, hay que considerar también la información ya existente en las bases de datos previas de la institución, cuya información es accesible para ambas aplicaciones a través de webservices, como se explica previamente, algunos de los cuales son comunes a ambos proyectos. El diseño final de ambas bases de datos mencionadas se puede visualizar en la figura 31 y en la figura 32 respectivamente, en el capítulo de resultados (Capítulo 5). De igual forma, el detalle de los webservices se puede ver en el apéndice C al final del documento.

Por otro lado, al tratarse de la elaboración de una aplicación web se procede a crear páginas web dinámicas bajo la plataforma ASP.NET 4.0, utilizándose para su desarrollo el entorno de Microsoft Visual Studio 2010 [42]. Esto obliga a crear ciertos elementos como requisitos previos que servirán de base para el resto del proyecto. Por ejemplo, antes de comenzar con el desarrollo en sí de las páginas, es necesario definir previamente aspectos como el estilo visual de las páginas, que debe ser común y consistente en todas ellas, una página maestra que servirá de base para todas las demás páginas, configurar la conexión a ambas bases de datos y a los webservices habilitados por el departamento de informática de la institución, y crear las hojas de estilos que proporcionarán uniformidad en todas las

páginas creadas. Además, dentro de Microsoft Visual Studio, se opta por un modelo de desarrollo en capas en vista de las grandes ventajas que tal modelo ofrece. Esto implica la creación de tres proyectos dentro de la misma solución, uno para la conexión a bases de datos, uno para la lógica del negocio, y un último para las interfaces de usuario.

Por último, al desarrollarse un proyecto visualmente compatible con la plataforma Academics su análisis detallado es muy importante ya que permite establecer las reglas que se seguirá dentro del presente desarrollo, sobre todo desde el punto de vista visual. El uso de componentes *Obout*, por ejemplo, permite guardar amplia compatibilidad con Academics, sin embargo, además de esto se ve la necesidad de recurrir a otro conjunto de controles denominado *AjaxControlToolkit*, cuya función es, de forma similar a *Obout*, proporcionar a Microsoft Visual Studio controles adicionales no estándar que introducen funcionalidades que resultan sumamente útiles al presente proyecto.

En el siguiente capítulo, donde se detalla los resultados del proyecto, se procede a proporcionar capturas de pantalla de las páginas elaboradas, el modelo entidad – relación para el diseño de ambas bases de datos, y ejemplos de la codificación, los mismos que resultaron de la aplicación de la metodología XP descrita en el presente capítulo. De igual forma, se incluyen tres apéndices donde se describe el diccionario de datos de las bases de datos utilizadas por la aplicación (apéndices A y B respectivamente), así como los webservices habilitados por el personal del departamento de informática de la PUCESA para el desarrollo de la presente aplicación (apéndice C). En el apéndice D, por otro lado, se puede ver porciones relevantes del código obtenido como resultado de la fase de codificación del presente proyecto, a modo de ejemplos que representan convenientemente el extenso código total de la aplicación.

4.2.1.4. Fase de pruebas

La fase final dentro de XP consiste en las pruebas, que necesariamente se deben realizar y documentar para evidenciar la satisfacción y conformidad del cliente con el trabajo realizado por el equipo de desarrollo. Si bien es cierto se van realizando pruebas parciales una vez que se concluye el desarrollo programado para cada iteración, y esos productos son presentados al cliente para su evaluación y, de ser el caso, corrección y posterior refactorización del código, es necesario realizar pruebas de usuario en relación al proyecto una vez que está concluido en su totalidad, pues solo así se puede dar una última palabra sobre el nivel de satisfacción del cliente.

Estas pruebas permiten identificar errores o vacíos de última hora que se puedan haber producido en la fase de codificación, y serán visibles únicamente cuando se realiza una evaluación global de la aplicación, mientras que en las evaluaciones parciales pueden no ser detectados. Desde el punto de vista del usuario, se hace una presentación final de la aplicación terminada para que el mismo pueda probar cada una de sus secciones y módulos, dándole así la oportunidad de detectar cualquier error o desacuerdo en relación a sus requisitos. De ser detectados tales errores, la metodología XP prevé la posibilidad de realizar una o más iteraciones adicionales de corta duración con la finalidad de pulir tales errores, hasta conseguir la satisfacción total del cliente.

Sin embargo, desde un punto de vista técnico también se debe realizar ciertos tipos de pruebas, ya que es necesario asegurarse de que los flujos de información están funcionando correctamente y de que las interfaces diseñadas son claras y funcionales, evitando así cualquier posibilidad de inconsistencia o mal entendido.

En la siguiente tabla se procede a detallar las pruebas funcionales realizadas sobre la versión final del código generado, las mismas que permitieron asegurar que el mismo cumple con toda la funcionalidad requerida y maneja adecuadamente la información pertinente en cada una de las páginas y secciones. Cabe aclarar que dichas pruebas funcionales han sido ejecutadas principalmente por el cliente, futuro usuario de la aplicación, quien debe dar su visto bueno y comprobar la funcionalidad desarrollada en la misma. La información que se utiliza para elaborar la siguiente tabla se extrae de varias de las preguntas de las encuestas de satisfacción del cliente aplicadas al personal de la OIP, las mismas que se pueden observar en el apéndice E y cuyos resultados se discuten con más detalle en la sección 5.2. (Evaluación preliminar).

Tabla 20: Pruebas funcionales realizadas sobre la versión final de la aplicación

Prueba	*Resultado*
La aplicación permite registrar el estado actual de un maestrante dentro de su proceso de graduación con toda la información correspondiente	✓
La aplicación genera una lista con todos los maestrantes activos y toda su información detallada	✓
La aplicación genera una lista con todos los maestrantes pasivos	✓
La aplicación genera una lista con todos los maestrantes graduados y toda su información detallada	✓
La aplicación genera una lista con todos los maestrantes que se encuentran en determinada etapa dentro de su proceso de graduación, con toda su información detallada	✓
La aplicación genera una lista con todos los maestrantes que requieren tomar actualizaciones con toda su información detallada	✓
La aplicación permite filtrar todas las lista de maestrantes generadas por una carrera específica	✓
La aplicación es capaz de reconocer automáticamente la fecha en la que ha egresado un maestrante, de ser el caso	✓
La aplicación es capaz de reconocer automáticamente la fecha en la que un maestrante ha culminado su última actualización, de ser el caso	✓
La aplicación es capaz de reconocer automáticamente las prórrogas a las que se someten o se han sometido los maestrantes, de ser el caso	✓
La aplicación recupera y despliega automáticamente información detallada sobre los procesos de graduación de los maestrantes listados	✓
La aplicación es capaz de reconocer automáticamente a los maestrantes que ya han pagado por el desarrollo del proyecto	✓
La aplicación recupera y despliega información detallada acerca de un maestrante en particular, incluyendo la información sobre su proceso de graduación	✓
La aplicación recupera y despliega información detallada acerca de un docente de posgrados en particular, incluyendo el número de proyectos en los que se encuentra trabajando actualmente	✓

La aplicación almacena y permite actualizar la información referente a direcciones de correo electrónico tanto de maestrantes como de docentes	✓
La aplicación permite registrar el porcentaje de avance de los maestrantes en sus respectivos proyectos cuando se encuentran en etapa de desarrollo	✓
La aplicación almacena y permite actualizar la información referente a parámetros generales del OIP que pueden ser utilizado posteriormente por ambas aplicaciones	✓
La aplicación genera un cuadro estadístico con el número de maestrantes graduados por cada carrera	✓
La aplicación genera un cuadro estadístico con el número de maestrantes que se encuentran en cada etapa general dentro de su proceso de graduación	✓
La aplicación genera un cuadro estadístico con el número de maestrantes que se encuentran en cada condición dentro de los lineamientos establecidos por el OIP	✓

Fuente: Elaboración propia

De este modo se verifica que todas y cada una de las funciones para las que fue originalmente creada la aplicación, y que se fueron desarrollando a través de las diferentes iteraciones programadas, cumplen su objetivo efectivamente desde un punto de vista técnico y funcional.

4.3. Materiales y herramientas

Al tratarse el presente proyecto de la elaboración de un producto de software para el OIP de la PUCESA, con la finalidad de resolver de la manera más urgente posible los problemas que dicho departamento enfrenta por falta de automatización y de herramientas que manejen adecuadamente su información, la mayor parte de herramientas utilizadas está obviamente relacionada con hardware y software informático, en especial las herramientas de diseño y desarrollo de software así como de diseño e implementación de bases de datos utilizadas.

Dentro de lo que es el desarrollo propiamente de software, se utiliza Microsoft Visual Studio 2010, la cual permite crear diferentes tipos de aplicaciones, tanto de escritorio como web y para la nube. Dentro de esta plataforma se opta por crear una aplicación web utilizando como lenguaje de servidor ASP.NET en su versión 4.0, la máxima soportada por esta versión de Visual Studio, la que a su vez

soporta los lenguajes de programación *Visual* Basic y C#. Se elige como lenguaje de programación C#. Por otro lado, se opta por usar Microsoft SQL Server 2008 R2 como SGBD, ya que es un motor muy potente y ampliamente utilizado tanto a nivel personal como empresarial en instituciones de todo el mundo. Pese a su potencia, es un SGBD muy amigable, intuitivo y fácil de usar, ya que ofrece interfaces gráficas para prácticamente todas las tareas que permite realizar y además soporta T-SQL, un lenguaje de consultas a bases de datos muy versátil.

Si bien hoy en día las versiones de los programas mencionados para el desarrollo del presente proyecto no son las últimas disponibles, se opta por utilizar específicamente éstas para guardar máxima compatibilidad con las herramientas utilizadas para crear y mantener la plataforma Academics ya existente en la PUCESA, con la cual debe integrarse completamente por las razones ya mencionadas en apartados anteriores. Como también se menciona, la presente aplicación debe guardar compatibilidad y concordancia con otra que se está desarrollando en paralelo [1], la cual a su vez también debe integrarse con Academics, por lo cual para el desarrollo de dicha aplicación también se está empleando las mismas herramientas. Se utilizan además herramientas complementarias a Microsoft Visual Studio como lo son los controles *Obout* y *AjaxControlToolkit*, las mismas que son conjuntos de controles adicionales que ofrecen su propia funcionalidad y amplían las características estándar de dicha plataforma de desarrollo. Se utiliza también una tercera herramienta complementaria conocida como *iTextSharp*, una biblioteca de clases que también amplía la funcionalidad de Visual Studio, y su objetivo es el de permitir general documentos en formato *Portable Document Format* (PDF) de manera nativa, que es un formato muy ampliamente utilizado a nivel mundial. Esto resulta muy útil a la hora de elaborar informes y reportes en cualquier tipo de aplicación, incluyendo el caso de las aplicaciones web.

Adicionalmente, se utiliza el programa *Oracle VirtualBox* como plataforma de máquina virtual, ya que por motivos de seguridad y rendimiento es aconsejable trabajar sobre un sistema operativo limpio y correctamente instalado y configurado, sin que exista la intervención de programas externos. Para ello, en este caso la mejor opción es trabajar sobre una máquina virtual.

En cuanto al hardware, para el desarrollo del presente proyecto se utiliza un computador portátil marca Toshiba de características procesador Intel Core I5, 8 GB de memoria RAM y 640 GB de disco duro, con una pantalla de 15,6 pulgadas y una resolución de 1366 x 768 pixels. En este computador se instala el software de máquina virtual mencionado más arriba con un límite de memoria RAM de 4 GB,

lo que permite instalar el sistema operativo Microsoft Windows 7 edición Ultimate de 64 bits, sobre el que se instalan la plataforma y SGBD antes descritos y se desarrolla la codificación del proyecto.

Finalmente, en relación a materiales, se recurre a muy pocos de ellos para el desarrollo del presente proyecto, pudiendo mencionarse principalmente lo relacionado a papelería, impresoras, tinta de impresora, pastas plásticas, espirales plásticos, carpetas, esferográficos y lápices. Esto resulta más que comprensible considerando que se trata de un proyecto de desarrollo de software.

Capítulo 5

Resultados

5.1. Producto final del proyecto de titulación

El producto final del presente proyecto de desarrollo consiste en una aplicación web cuya finalidad es la de solventar los diferentes problemas que aquejan al OIP de la PUCESA justamente por la falta de un adecuado medio de almacenamiento y administración de su información clave. En este capítulo se procede a describir dicha aplicación y cada una de sus secciones y apartados.

Sin embargo, previamente al desarrollo en sí de la aplicación es necesario diseñar, elaborar e implementar la base de datos de la misma, así como también conectar con la base de datos de la otra aplicación mencionada cuyo desarrollo se ejecuta en paralelo a la presente, así como también con los webservices habilitados por el departamento de informática de la institución. En este capítulo se incluye los diagramas de modelo entidad – relación de ambas bases de datos, mientras que en los apéndices A y B se procede a describir las bases de datos y sus tablas y en el apéndice C se procede a describir los mencionados webservices.

Si bien la presente aplicación se ha desarrollado y probado en un entorno local funcionando sobre una máquina virtual, para poder probarla externamente se procede a publicar el sitio web resultante usando un servicio de alojamiento en Internet que soporta las tecnologías y versiones de software utilizadas para la elaboración del proyecto, como son ASP.NET 4.0 y Microsoft SQL Server 2008 R2 para las bases de datos. De entre múltiples opciones disponibles en el mercado, se elige como servicio de hosting el proporcionado por MyASP.NET (https://www.myasp.net/) ya que ofrece múltiples ventajas como una cuenta gratuita de alojamiento por 60 días, que resulta muy útil para realizar pruebas, y soporte de múltiples versiones tanto del framework de ASP.NET como del SGBD Microsoft SQL Server.

En el siguiente gráfico se muestra la página principal del mencionado servicio de hosting.

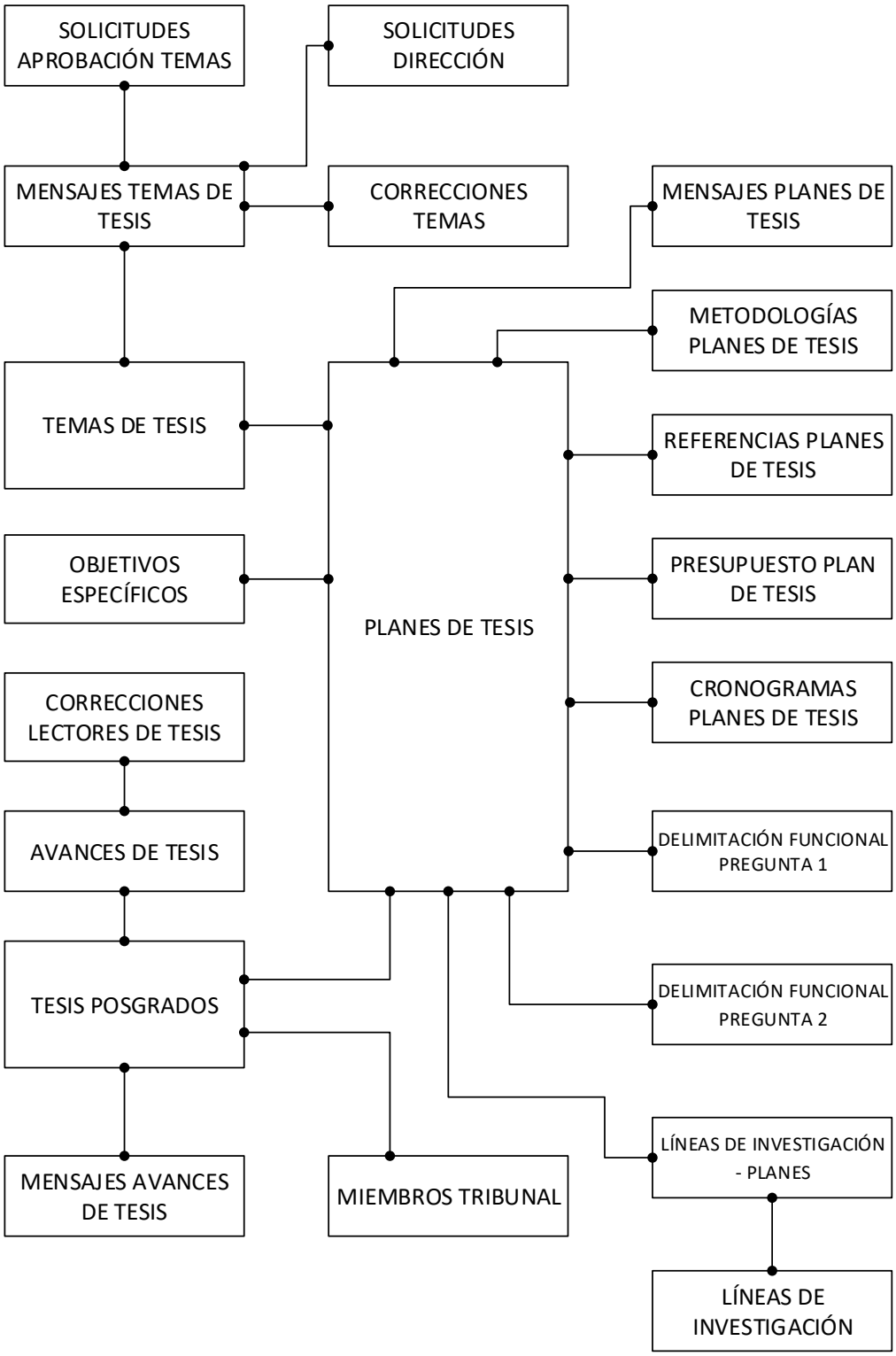
Figura 30: Página principal del servicio de alojamiento de MyASP.NET

Fuente: http://www.myasp.net/

Se comienza diseñando e implementando la base de datos de la aplicación usando el entorno gráfico de Microsoft SQL Server 2008 R2, así como también conectando a la base de datos de la aplicación a desarrollarse en [1], que para el momento ya está implementada.

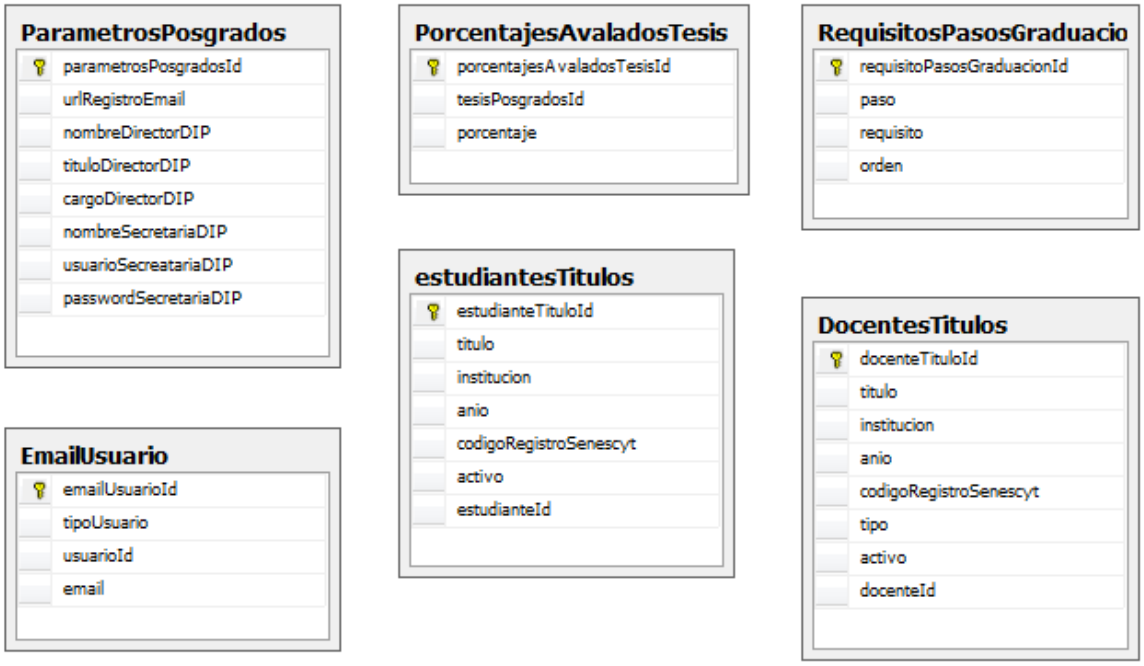
En los siguientes gráficos se pueden observar los modelos entidad – relación de ambas bases de datos, generados automáticamente por el SGBD que, al manejar bases de datos estrictamente relacionales, se apoya en este modelo de diseño de bases de datos.

Figura 31: Modelo entidad – relación para la base de datos de la aplicación paralela



Fuente: Elaboración propia

Figura 32: Modelo entidad – relación para la base de datos de la aplicación

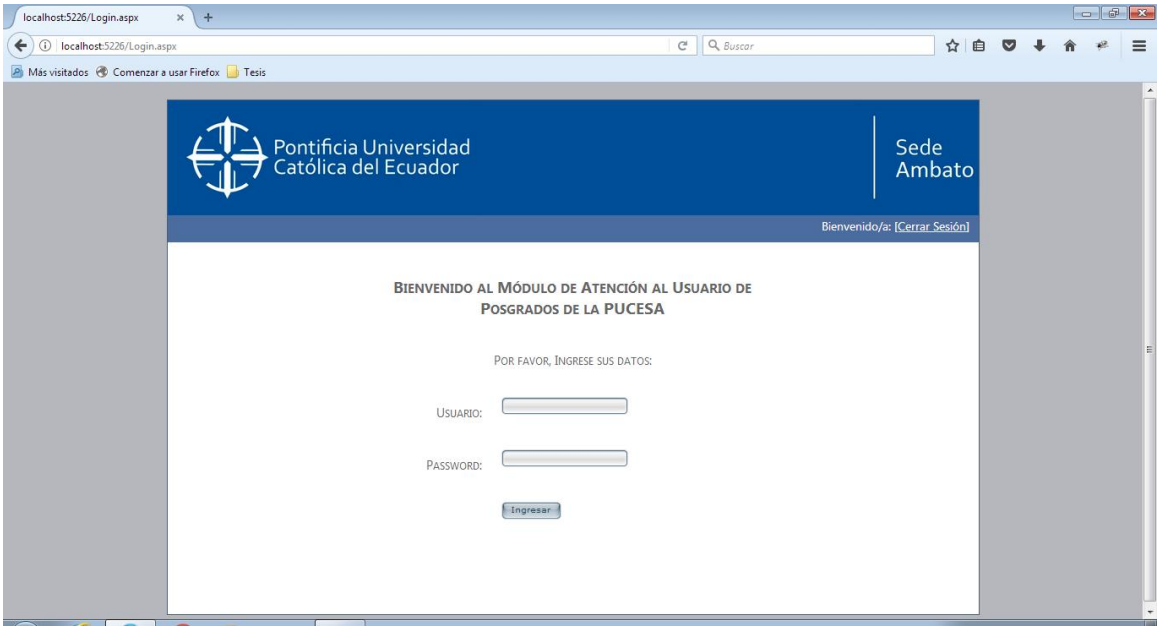


Fuente: Microsoft SQL Server 2008 R2

Como se observa, la base de datos propia de la aplicación consta de seis tablas que no están relacionadas entre sí, sin embargo, la mayoría de ellas guardan identificadores de entidades como estudiantes, docentes, tesis y usuarios, tal como se lo describe en el apéndice B, los mismos que sirven para acceder a sus respectivos bancos de información, ya sea que estos estén localizados en la base de datos del otro proyecto [1] o en la de la institución, que será por tanto accesible a través de los webservices habilitados (ver descripción en el apéndice C).

Por otro lado, la aplicación desarrollada comienza por una página de login, que se observa en la figura 33, donde el usuario de la misma, en este caso la secretaria del OIP, puede proceder a digitar su nombre de usuario y contraseña para acceder así al menú principal de la aplicación. Cabe mencionar que en este caso, la secretaria del OIP o en su lugar el director del OIP o la persona por él designada será el/la único(a) usuario(a) de la aplicación, por lo que la misma no soporta la creación de otros usuarios. Por otro lado, el usuario cuenta también con una página donde puede cambiar de contraseña de manera fácil.

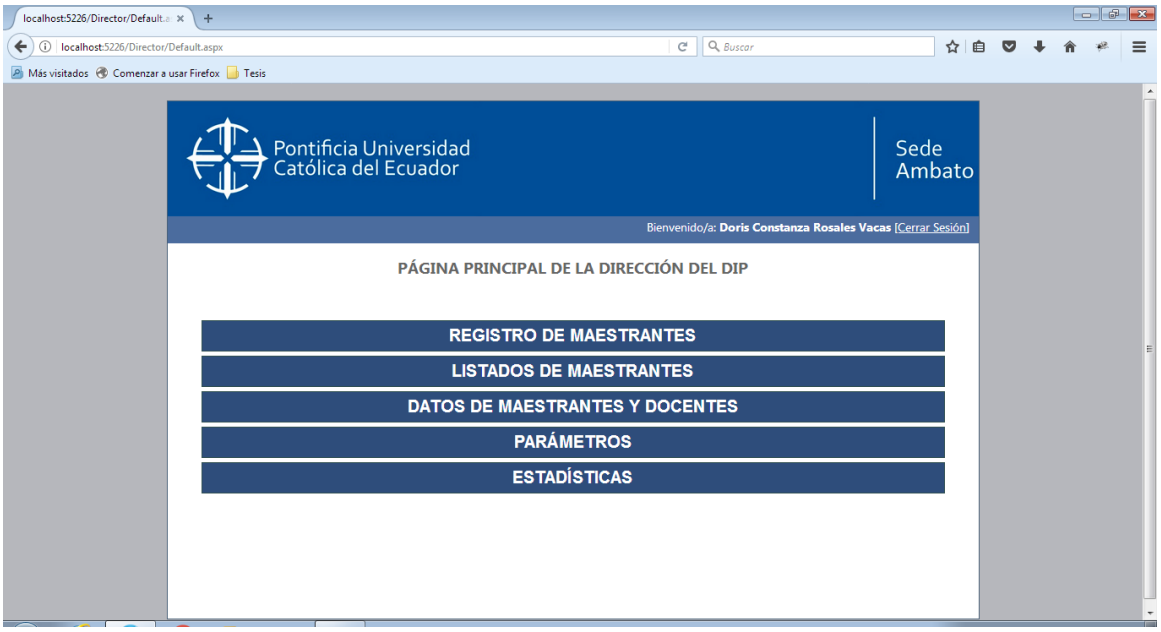
Figura 33: Página de login de usuario



Fuente: Elaboración propia

El menú principal (figura 34), a su vez, es sumamente intuitivo y se encuentra claramente demarcado por textos muy fáciles de entender y seguir. Es un menú de tipo acordeón en el que, cuando el usuario selecciona un ítem, éste se expande para mostrar las opciones que contiene.

Figura 34: Menú principal de la aplicación



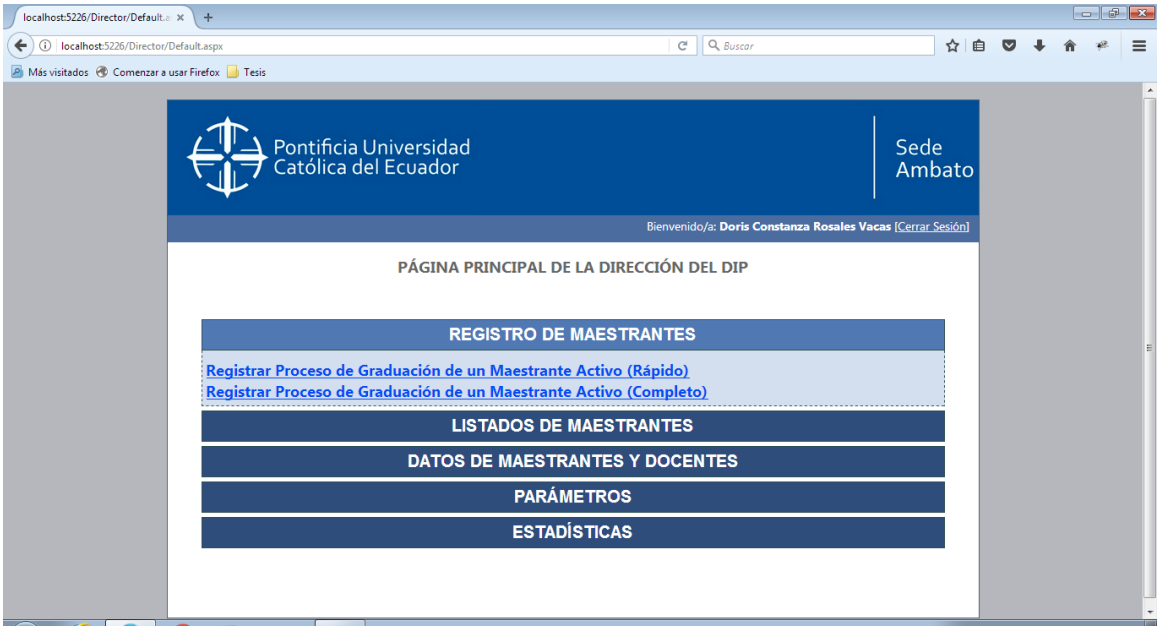
Fuente: Elaboración propia

Cada sección del menú principal contiene a su vez un submenú que proporciona las opciones que puede elegir el usuario final. A continuación, se procede a describir cada menú:

Menú "Registro de Maestrantes"

Las opciones del menú "Registro de maestrantes", que se observan en la figura 35, ofrecen dos tipos de interfaces para registrar los datos de un maestrante que se encuentra en determinado punto del proceso de graduación, lo que implica un importante ahorro de tiempo en vista de que la aplicación a desarrollarse en [1] está diseñada para ir registrando los pasos de dicho proceso de uno por vez, por lo que el registrar a un maestrante que se encuentre en las últimas etapas de su proceso de graduación utilizando tal aplicación toma un tiempo muy considerable.

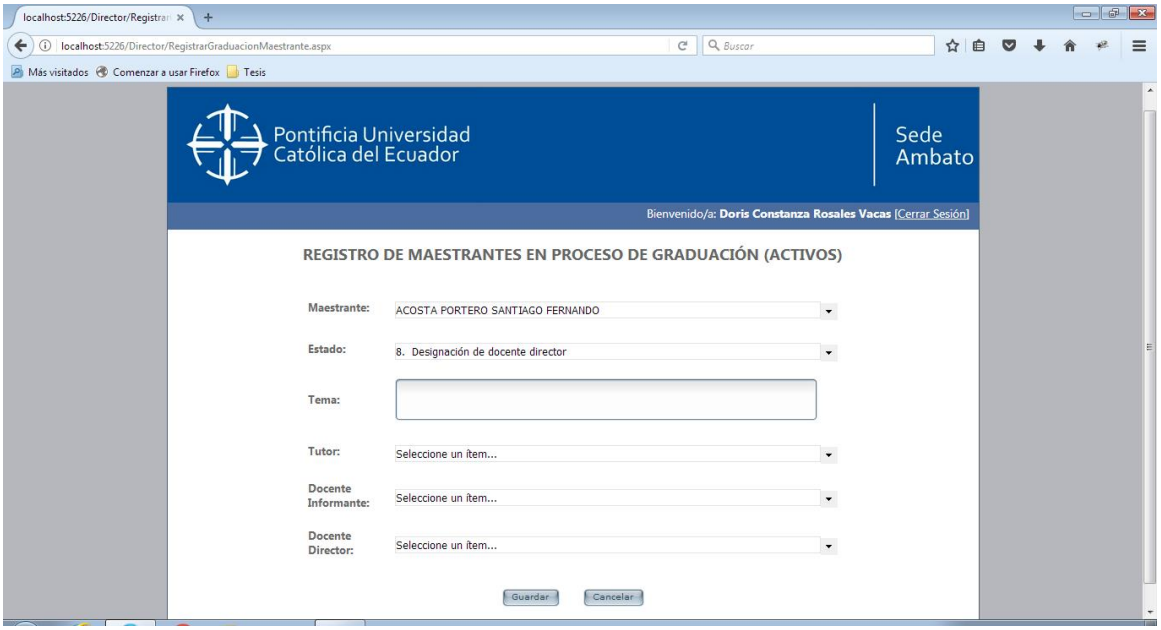
Figura 35: Menú "Registro de maestrantes"



Fuente: Elaboración propia

La presente aplicación en cambio, a través de la primera de las interfaces mencionadas, la misma que se observa en la figura 36, permite elegir directamente el punto del proceso en el que se encuentra el maestrante en cuestión y proporcionar la información requerida, como lo es el tema del proyecto, el docente tutor, el docente informante del plan, los docentes revisores, fecha de defensa de ser el caso, etc.

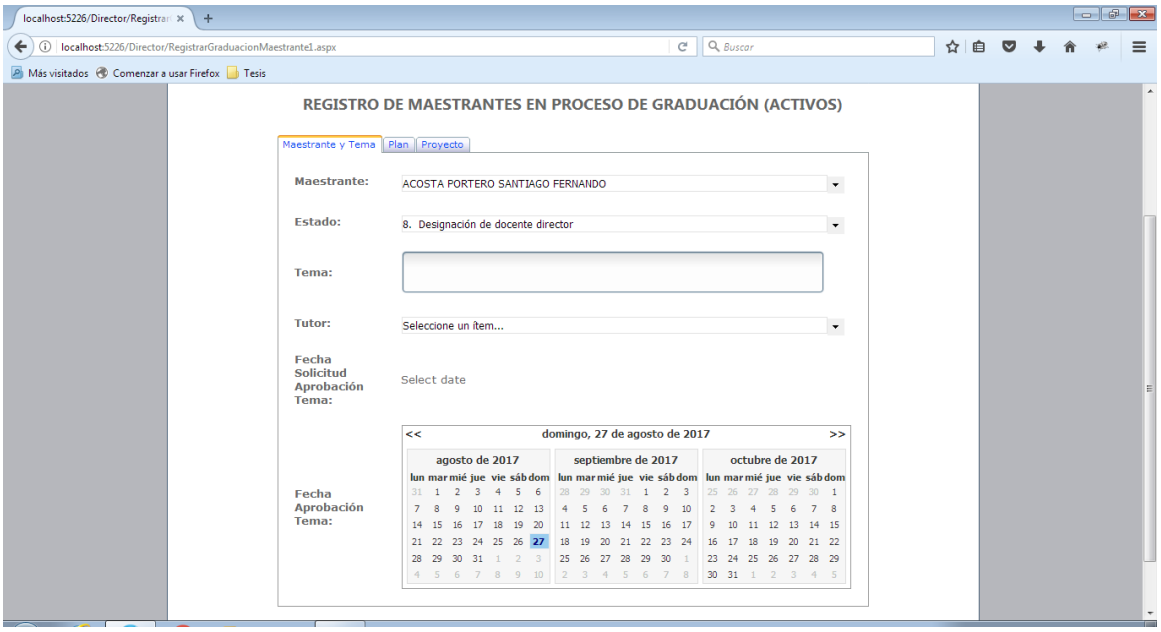
Figura 36: Primera interfaz para registro de maestrantes en proceso de graduación



Fuente: Elaboración propia

La segunda interfaz disponible, como se observa en la figura 37, ofrece una funcionalidad similar a la primera, con la diferencia de que es algo más compleja en el sentido de que permite registrar las fechas para los diferentes eventos que tienen lugar dentro del proceso de graduación de un maestrante, lo cual, en la aplicación que se desarrolla paralelamente [1], se realiza de forma automática. La utilidad de tener tales fechas registradas consiste en que se puede calcular los días transcurridos entre un evento y otro relacionado dentro del proceso de graduación, lo cual a su vez permite la posterior formulación de estadísticas, pudiéndose así conocer promedios del tiempo que tarda cada paso del proceso de graduación en un determinado rango de tiempo, por ejemplo, o simplemente conocer el tiempo que un maestrante en particular ha tardado en cada paso del proceso.

Figura 37: Segunda interfaz para registro de maestrantes en proceso de graduación



Fuente: Elaboración propia

La primera interfaz expuesta no presenta tal característica y, por consiguiente, todos los pasos que se registran automáticamente para un maestrante que se encuentra en determinado punto de su proceso de graduación se guardan con una misma fecha, por lo que resultan inútiles para elaborar estadísticas como las mencionadas en el párrafo anterior. La política que se implementa en la aplicación para estos casos es simplemente ignorarlos, excluyéndolos de los datos estadísticos. La segunda interfaz en cambio, tiene un diseño en pestañas que permite al usuario seleccionar las fecha en las que realmente tuvieron lugar los diferentes eventos a registrase. Cada pestaña corresponde a una etapa dentro del proceso de graduación, es decir tema, plan, desarrollo del proyecto y requisitos para la graduación. En el ejemplo se observa un calendario que permite registrar la fecha de la aprobación del tema para un maestrante que se encuentra en un punto posterior del proceso de graduación.

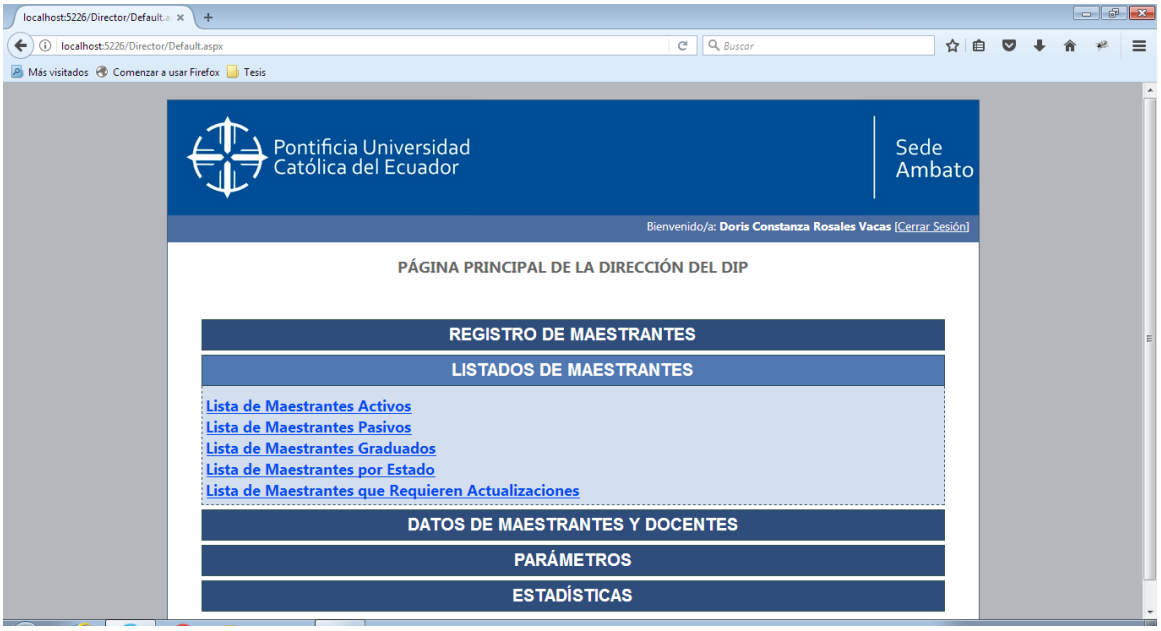
Cabe además mencionar que ambas interfaces descritas están diseñadas para registrar los procesos de graduación únicamente de los maestrantes que no se encuentran aún registrados en el sistema. Una vez que se registra un maestrante en su estado actual dentro del proceso de graduación, cuando este cumple con los requisitos para avanzar al siguiente paso ya no se lo puede hacer utilizando la presente

aplicación, siendo necesario recurrir a la otra aplicación que se ha venido mencionando a lo largo de este capítulo [1], para lo cual la misma ha sido diseñada.

Menú "Listados de Maestrantes"

El siguiente menú de la aplicación (figura 38) corresponde a los listados de maestrantes que la misma desplegará, los mismos que han sido solicitados por el personal de la OIP, es decir, por el cliente.

Figura 38: Menú "Listados de maestrantes"

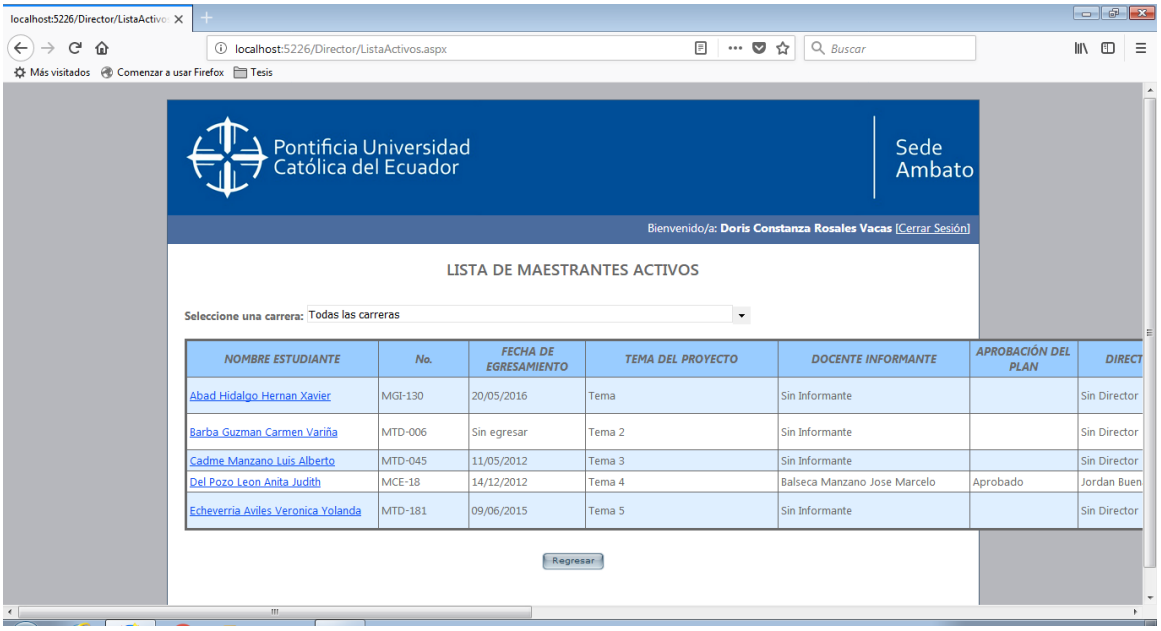


Fuente: elaboración propia

El primero de estos listados corresponde a los maestrantes activos, que son aquellos cuyo proceso de graduación ha sido registrado, ya sea utilizando una de las dos interfaces provista en la presente aplicación, o bien utilizando la aplicación alternativa que se encuentra en desarrollo [1]. Cabe mencionar, sin embargo, que el contenido de esta lista será fiable únicamente cuando se hayan ya registrado absolutamente todos los casos de maestrantes que mantengan un proceso de graduación en marcha.

Aquel es un proceso de digitación manual que estará a cargo del personal del OIP, y es de vital importancia para el correcto funcionamiento de la aplicación por el motivo expuesto. Mientras tanto, todo maestrante cuyo proceso de graduación no haya sido debidamente registrado y almacenado en las bases de datos, se considerará como pasivo, excepto en el caso de los maestrantes graduados, quienes también deben ser debidamente registrados en las bases de datos de la aplicación utilizando los mismos métodos disponible para el registro de los maestrantes activos. Cabe mencionar además que, al igual que en todas las páginas de listados provistas por la aplicación, la de maestrantes activos permite elegir una de las carreras de posgrados para filtrar el listado, o bien eliminar el filtro para volver a ver la lista completa de maestrantes que, en todos los casos, se despliega ordenada alfabéticamente por los apellidos de los maestrantes, como se ve en la figura 39.

Figura 39: Listado de maestrantes activos



Fuente: Elaboración propia

El siguiente listado disponible en la aplicación es el de maestrantes pasivos, que son los que no se han registrado en las bases de datos de la aplicación y, por consiguiente, se considera que no mantienen un proceso de graduación en marcha. Al igual que en el caso anterior, el listado de maestrantes pasivos

también es desplegado en orden alfabético y también se permite elegir una carrera para filtrar los resultados, o regresar a la lista completa.

Sigue la lista de maestrantes graduados, la misma que obviamente despliega la información de los maestrantes que se encuentran registrados como graduados en la base de datos de la aplicación, ofreciendo las mismas características del listado anterior en cuanto a orden de los registros y posibilidad de filtrado por carreras.

A continuación se brinda acceso al listado de maestrantes que se encuentran en un determinado paso del proceso de graduación para lo cual, es necesario elegir dicho paso de un listado. Es decir, esta página se encarga de desplegar un subconjunto de los maestrantes activos, de los cuales se listará únicamente los que se encuentren en el paso especificado. Esta lista brinda también las mismas características de los listados antes descritos en cuanto a orden de los registros y posibilidad de filtrarlos por carreras.

Finalmente se proporciona acceso al listado de los maestrantes que requieren tomar actualizaciones, lo cual es una función de suma importancia y de necesidad urgente para el personal de la OIP. Este listado ofrece las mismas características descritas para los listados anteriores incluyendo el filtrado por carrera, y muestra todos los maestrantes que ya deberían tomar actualizaciones los cuales a su vez se consideran todos aquellos para los que ya ha transcurrido más de un año calendario (365 días) desde su fecha de egresamiento, o desde la fecha en la que han concluido su última actualización. Esta información, a su vez, es provista por la misma institución a través de los webservices habilitados por el departamento de informática (ver descripción en el apéndice C).

Cabe mencionar que el diseño visual de los listados de maestrantes pasivos, graduados, por estado y que requieren tomar actualizaciones es idéntico al de maestrantes activos, por lo que no se incluye capturas de pantalla de cada uno de ellos. Además, como se puede observar, dicho diseño visual incluye una tabla que se desborda de la pantalla debido a la gran cantidad de información que se despliega para los maestrantes, lo cual se realiza así por pedido explícito del cliente. Sin embargo, todos los listados ofrecen la posibilidad de hacer click sobre el nombre de cualquier maestrante para abrir una nueva página donde se despliega información aún más detallada sobre el mismo en formato vertical. Dicha página se describe en la siguiente sección.

Menú "Datos de Maestrantes y Docentes"

Este es el siguiente menú disponible, como se observa en la figura 40, y ofrece un conjunto de opciones que permiten visualizar y administrar la información relevante relacionada a maestrantes y docentes, por parte de la dirección de la OIP.

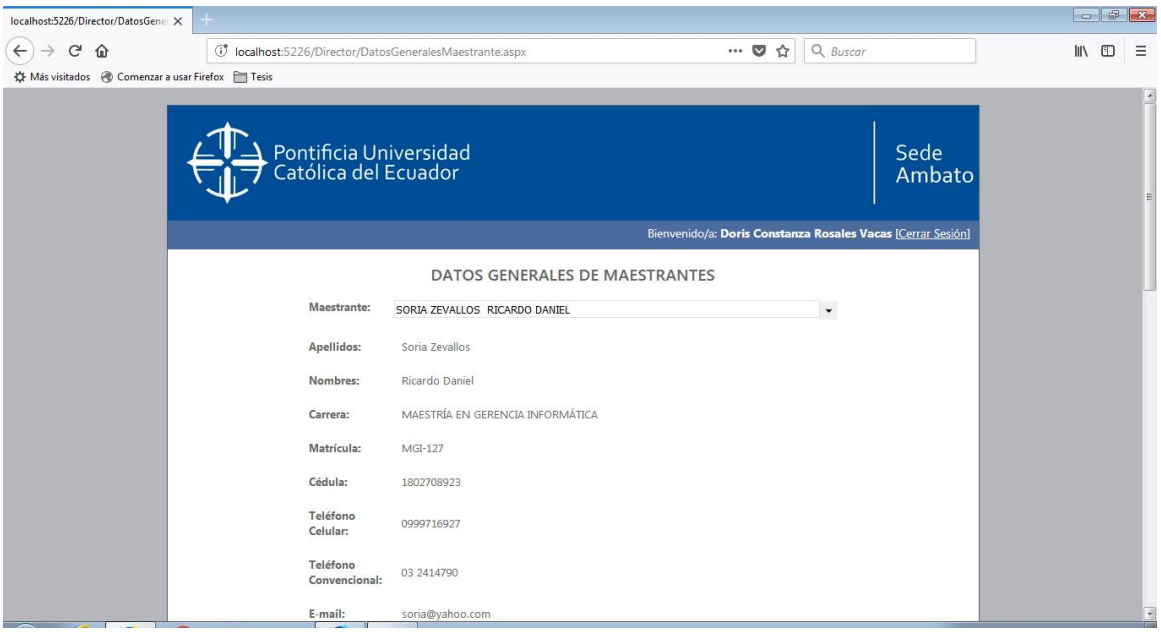
Figura 40: Menú "Datos de maestrantes y docentes"

Fuente: Elaboración propia

La primera opción abre una página donde se pueden ver los datos detallados de un maestrante específico, que puede ser seleccionado de una lista. En ella constan todos los maestrantes registrados en la base de datos institucional preexistente, sin distinción de carreras o estados, y en orden alfabético. La información desplegada incluye nombres, apellidos, carrera, número de matrícula, número de cédula, teléfonos, dirección de e-mail registrada, fecha de nacimiento y, de ser el caso, fecha de egresamiento, fecha de culminación de la última actualización, tema del proyecto de graduación, docentes relacionados a su proyecto (informante, director y revisores), si requiere tomar actualizaciones, si ha realizado el pago por derechos de desarrollo del proyecto y si ha tomado prórrogas (primera y segunda). Esta es la misma página que se despliega automáticamente cuando se

hace click sobre el nombre de un maestrante en cualquiera de los listados descritos en la sección anterior, en cuyo caso ya no es necesario seleccionar el maestrante. Su diseño visual se puede observar a continuación en la figura 41.

Figura 41: Datos generales de un maestrante



Fuente: Elaboración propia

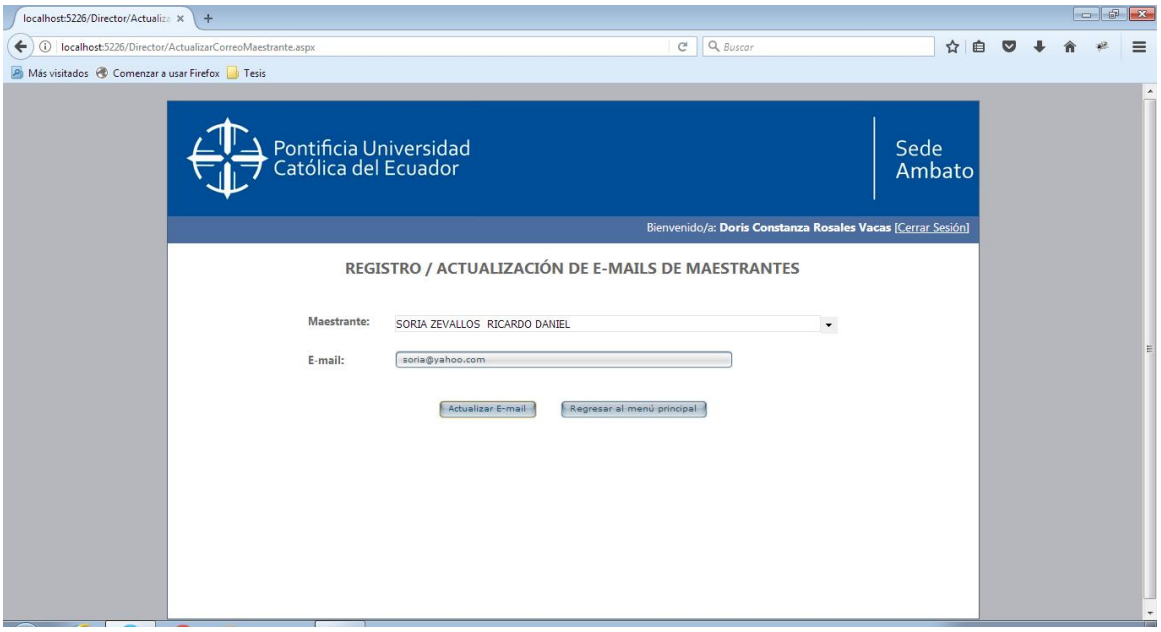
De igual forma existe una opción que permite desplegar datos generales de un docente que también puede ser seleccionado de una lista, al igual que en el caso anterior, como se muestra en la figura 42. En ella se listan igualmente todos los docentes de posgrados registrados en la base de datos de Academics, sean estos docentes propios de la institución o docentes externos, en orden alfabético. Los datos que se despliegan son los apellidos, nombres, número de cédula, dirección de e-mail registrada y el número de proyectos en los que se encuentra trabajando actualmente en calidad de tutor, informante, director y revisor, así como el total de proyectos. Esta información resulta de vital importancia para el OIP, según los requerimientos establecidos por su personal.

Figura 42: Datos generales de un docente

Fuente: Elaboración propia

Dentro de este menú existen también opciones que permiten registrar y/o actualizar las direcciones de correo electrónico tanto de docentes como de maestrantes, como se ve en la figura 43, ya que en el caso de los docentes no existe tal información en la base de datos de Academics y, en el caso de los maestrantes, en algunos casos existe la información pero puede ser inconsistente o desactualizada y en otros casos, ni siquiera existe tal información. Por tal motivo se decide almacenar y administrar esa información en la base de datos propia de la aplicación.

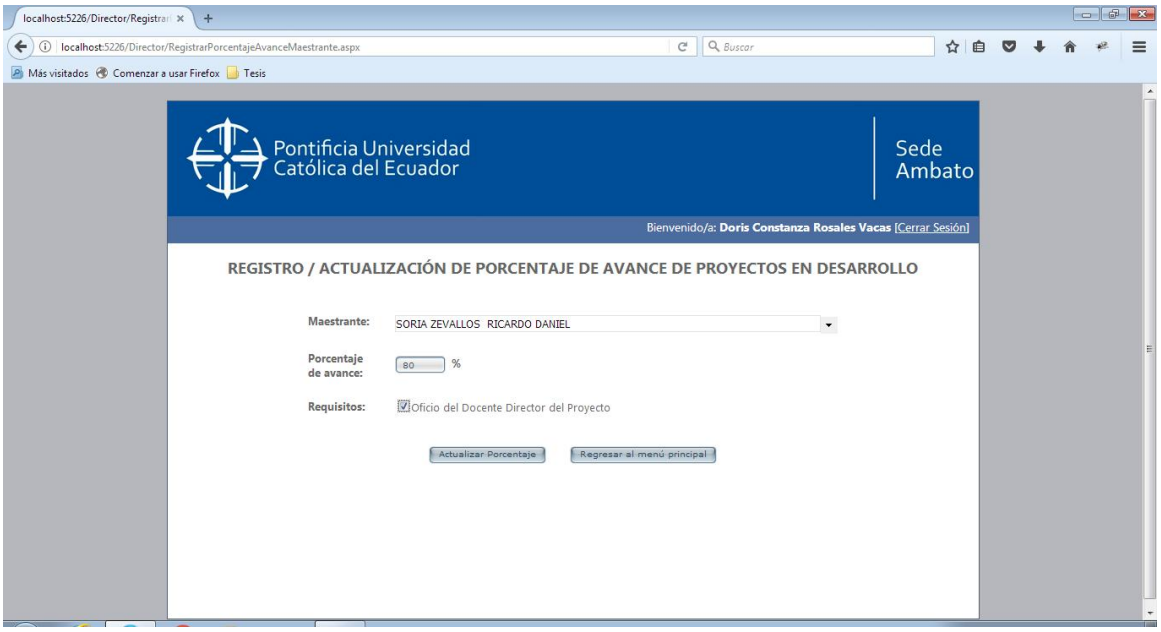
Figura 43: Registro/Actualización de dirección de correo electrónico para maestrante



Fuente: Elaboración propia

Por último, un caso similar se da para el registro de porcentajes de avance que presentan los maestrantes dentro del desarrollo de su proyecto de graduación. Esta información también se la maneja independientemente dentro de la base de datos de la aplicación, y se proporciona una interfaz muy sencilla para su registro (figura 44), teniéndose como condiciones para poder hacerlo el que el maestrante se encuentre en proceso de desarrollo del proyecto y que se incluya un oficio por parte del docente director certificando el porcentaje de avance que presente el maestrante.

Figura 44: Registro/Actualización de porcentaje de avance en proyecto para maestrante

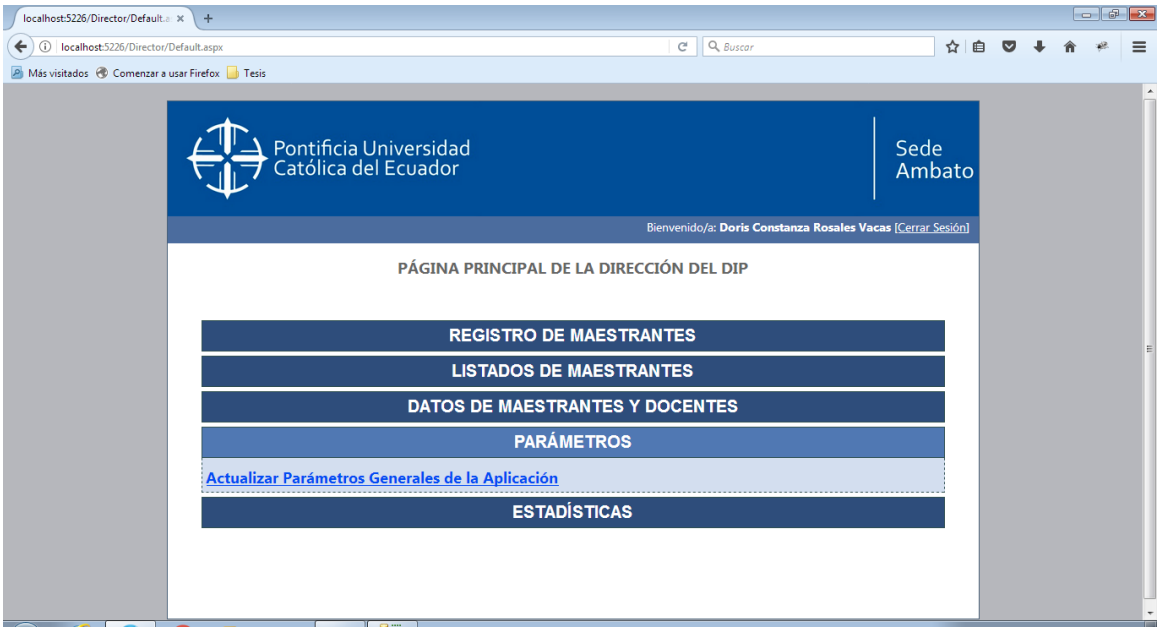


Fuente: Elaboración propia

Menú "Parámetros"

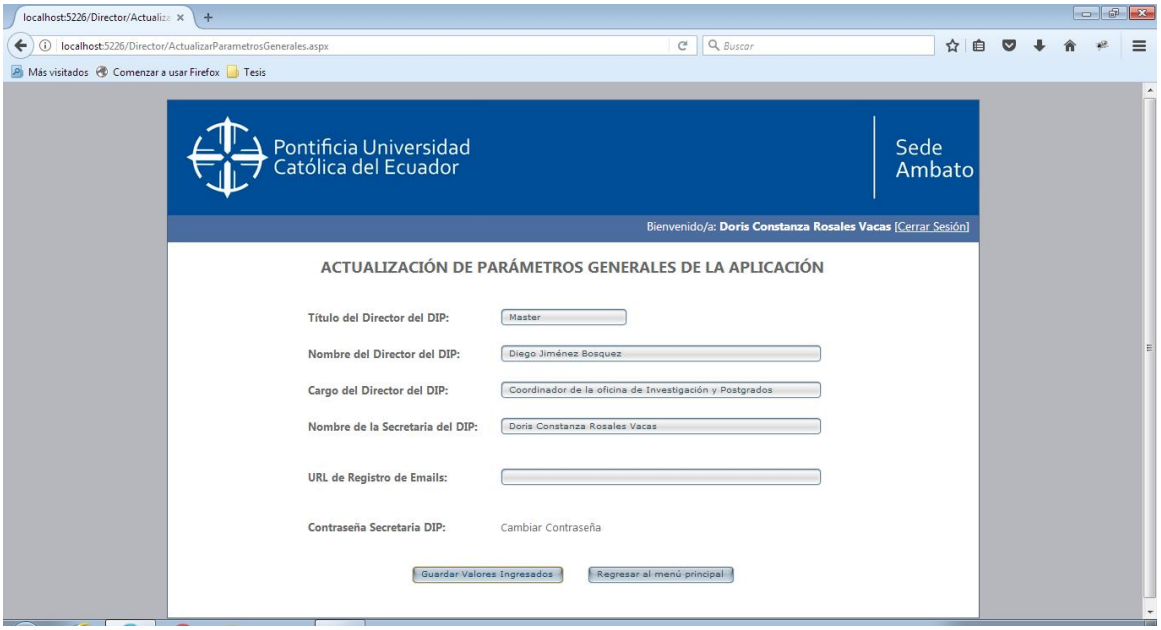
La siguiente entrada en el menú principal se titula "Parámetros", como se observa en la figura 45, y su única opción de submenú permite al usuario de la aplicación la actualización de ciertos parámetros de índole general que resultan útiles en muchos casos a lo largo no solo de la presente aplicación sino también de la aplicación paralela que se desarrolla en [1], como por ejemplo, para la generación automática de oficios dirigidos a la dirección del OIP por parte de los maestrantes. El diseño web de esta página puede observarse en la figura 46.

Figura 45: Menú "Parámetros"



Fuente: Elaboración propia

Figura 46: Actualización de parámetros generales de la aplicación

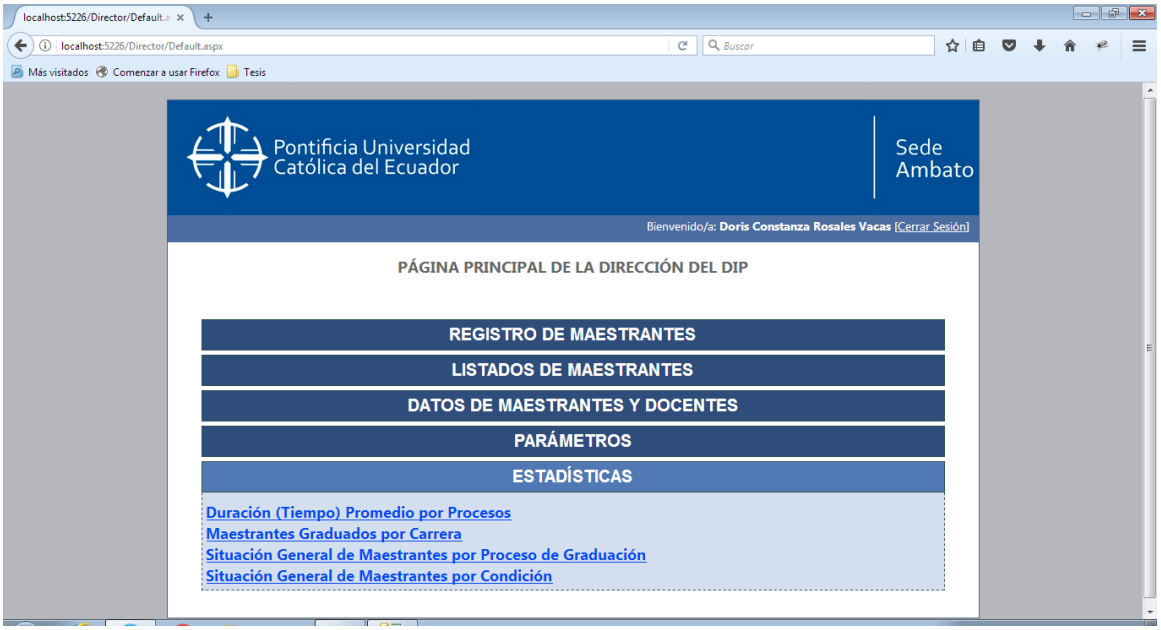


Fuente: Elaboración propia

Menú "Estadísticas"

La sección final del menú principal de la aplicación (figura 47) ofrece al usuario acceso a los diferentes cuadros estadísticos que se han generado como herramientas de apoyo en la toma de decisiones para el personal administrativo del OIP. Cada gráfico funciona independientemente del resto y, donde se requiere, se incluye controles que permiten filtrar los datos que se resumen en un cuadro estadístico, por ejemplo, para seleccionar una carrera específica o un rango de fechas para el cual tomar los datos.

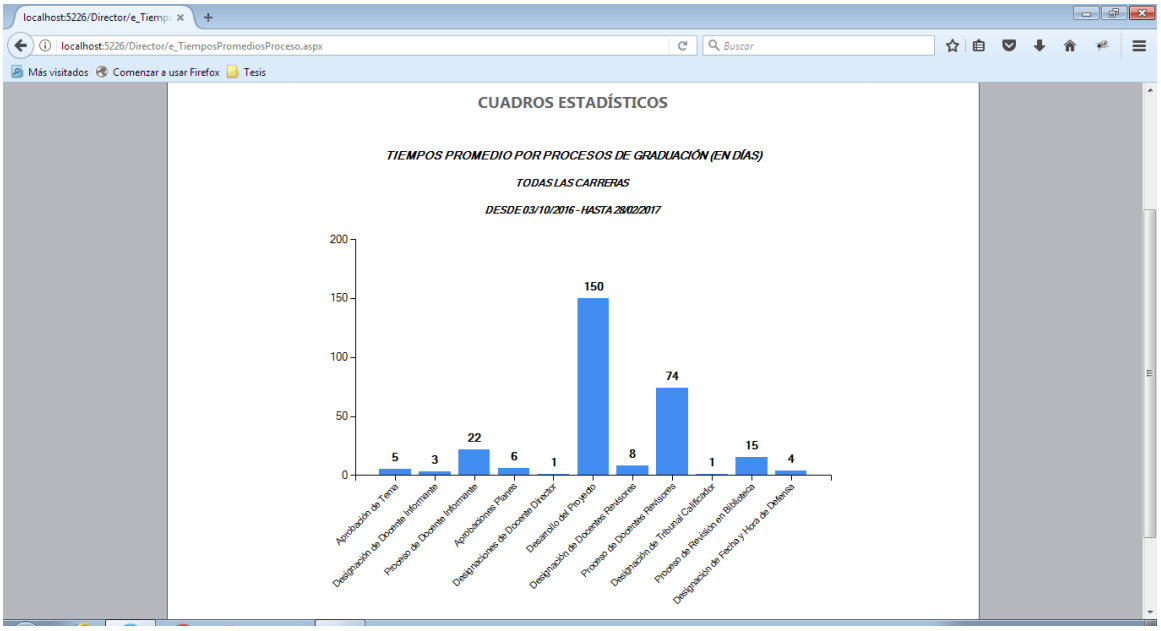
Figura 47: Menú "Estadísticas"



Fuente: Elaboración propia

El primer gráfico estadístico (figura 48) se encarga de mostrar los tiempos promedios en días que les toma a los maestrantes completar cada etapa clave de su proceso de graduación, ya sea por factores internos o externos. Esta es una herramienta muy importante para la dirección del OIP, ya que permite identificar visualmente cuellos de botella que se pueden presentar en el proceso de graduación y así tomar decisiones oportunamente para corregir dichas falencias. Para este tipo de gráfico se permite seleccionar un rango de fechas del cual tomar los datos y contabilizarlos, lo cual convierte el gráfico en una herramienta mucho más poderosa ya que permite, por ejemplo, comparar los tiempo promedio del proceso en un semestre determinado contra los de otro semestre y así verificar si alguna política aplicada por el OIP ha dado los resultados esperados y en qué medida.

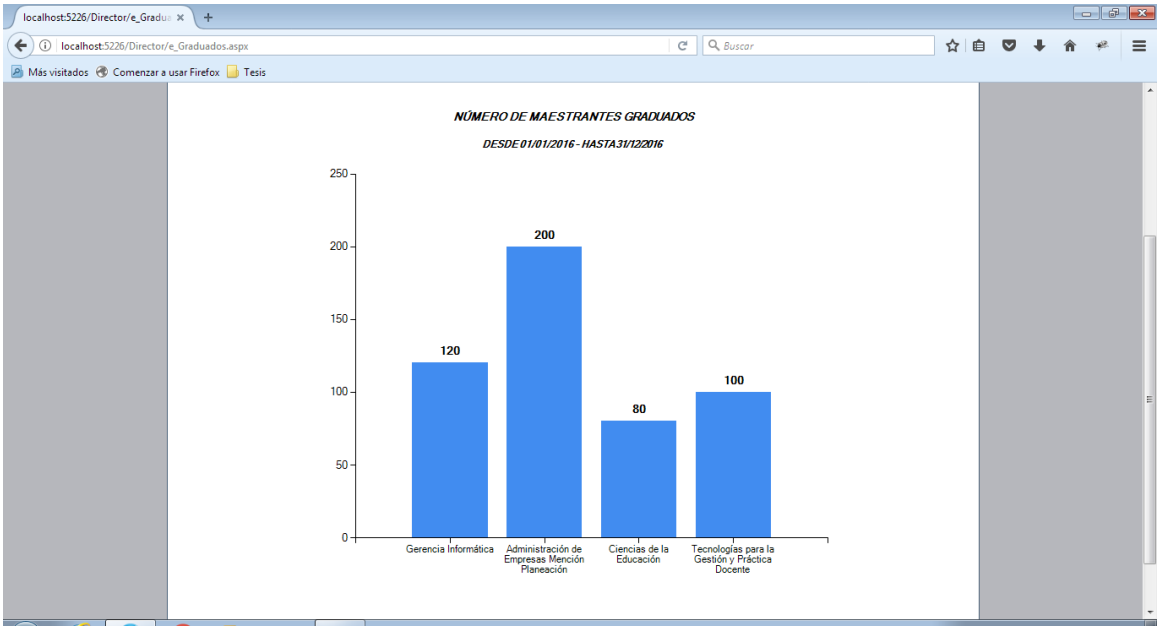
Figura 48: Duración (tiempo) promedio por procesos



Fuente: Elaboración propia

El segundo gráfico estadístico (figura 49) muestra un conteo de la cantidad de maestrantes que se han graduado por cada una de las carreras de posgrado existentes. La interfaz correspondiente también permite elegir un rango de fechas determinado para poder visualizar la cantidad de maestrantes graduados por carrera exclusivamente dentro de dicho rango.

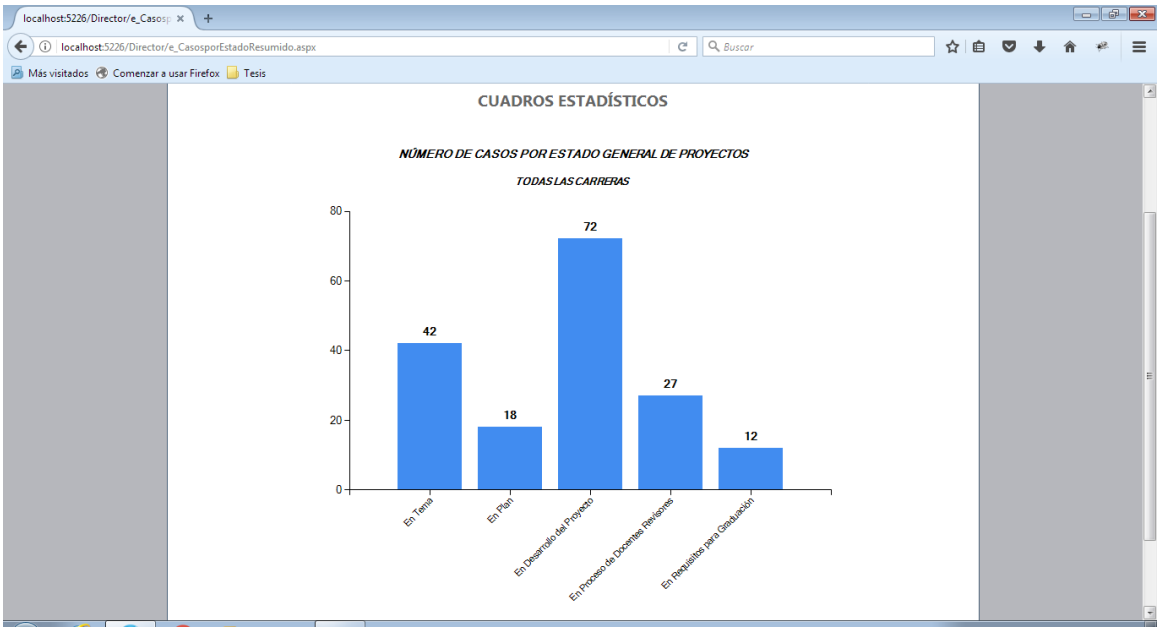
Figura 49: Maestrantes graduados por carrera



Fuente: Elaboración propia

El tercer gráfico estadístico (figura 50) sirve para mostrar la cantidad de maestrantes que se encuentra en cada etapa del proceso de graduación, sin embargo, no se consideran las etapas detalladas del proceso sino únicamente las etapas generales, esto es tema, plan, desarrollo del proyecto, proceso de revisores y requisitos para graduación. El gráfico equivalente que considera las etapas detalladas del proceso de graduación de los maestrantes no se incluye en la presente aplicación ya que ha sido implementada en la aplicación paralela que se encuentra en desarrollo [1], con la cual la presente aplicación tendrá que integrarse completamente en el corto plazo. Además, en el caso de este gráfico se permite filtrar la información por carrera, ya que puede también resultar bastante útil para el departamento el saber la cantidad de casos por etapa general de graduación de una carrera determinada, y compararla con la de otras carreras.

Figura 50: Situación general de maestrantes por proceso de graduación



Fuente: Elaboración propia

Por último, el cuarto gráfico estadístico (figura 51) que se incluye al momento en la aplicación despliega un conteo del número de casos de los maestrantes que se encuentran en las diferentes condiciones o estados que pueden considerarse a nivel general, esto es, activos, pasivos, graduados, y los que requieren tomar actualizaciones. En este último caso también se permite filtrar la información por carreras, ya que así puede ser requerido en cualquier momento por la dirección del OIP.

Figura 51: Situación general de maestrantes por condición

Fuente: Elaboración propia

5.1.1. Codificación

A continuación, en la figura 52, se muestra un ejemplo simple del código elaborado durante el desarrollo del presente proyecto, lo cual forma también parte de los resultados del proyecto, en lo que se refiere a una porción de una clase de la capa de lógica de negocio. Sin embargo, se puede observar porciones relevantes más amplias y detalladas del código de la aplicación desarrollada en el apéndice D.

Figura 52: Ejemplo de codificación – archivo *ParametrosPosgradosManager.cs*

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using AtencionUsuario.Model;

namespace AtencionUsuario.Logic
{
    public class ParametrosPosgradosManager
    {
        public static ParametrosPosgrados GetParametrosPosgrados()...

        public static bool Save(ParametrosPosgrados parametros)
        {
            try
            {
                using (var db = new AtencionUsuarioEntities())
                {
                    ParametrosPosgrados _parametros =
                        db.ParametrosPosgrados.FirstOrDefault();
                    parametros.parametrosPosgradosId =
                        _parametros.parametrosPosgradosId;
                    db.AttachTo("ParametrosPosgrados", _parametros);
                    db.ParametrosPosgrados.ApplyCurrentValues(parametros);
                    db.SaveChanges();
                    return true;
                }
            }
            catch
            {
                return false;
            }
        }

    }
}
```

Fuente: Elaboración propia

5.2. Evaluación preliminar

Como herramienta de evaluación del grado de satisfacción de los usuarios internos en cuanto a la optimización de los procesos de atención a los mismos dentro de la OIP, se procede a aplicar encuestas

de satisfacción con preguntas de opción múltiple. El modelo de dichas encuestas se puede observar en el apéndice E, el mismo que contiene copias de las encuestas originales aplicadas al personal administrativo de la OIP, el mismo que consta de dos personas, debidamente llenos y con firmas de responsabilidad.

Cada cuestionario consta de trece preguntas orientadas a evaluar la percepción del futuro usuario de la aplicación web desarrollada en cuanto a su funcionalidad, usabilidad, fiabilidad y rendimiento. Para las respuestas se deben marcar con una X en una de las opciones incluidas: Excelente, Muy bueno, Bueno, Regular o Malo, para cada pregunta. Para poder realizar un análisis objetivo de las respuestas provistas por los usuarios encuestados, y establecer numéricamente un valor promedio que represente el grado de satisfacción de los usuarios para cada aspecto evaluado, se procede a asignar valores correspondientes a porcentajes de satisfacción para cada posible respuesta, de la siguiente manera:

Tabla 21: Equivalencia entre posibles respuestas de la encuesta aplicada y porcentajes de satisfacción de los usuarios

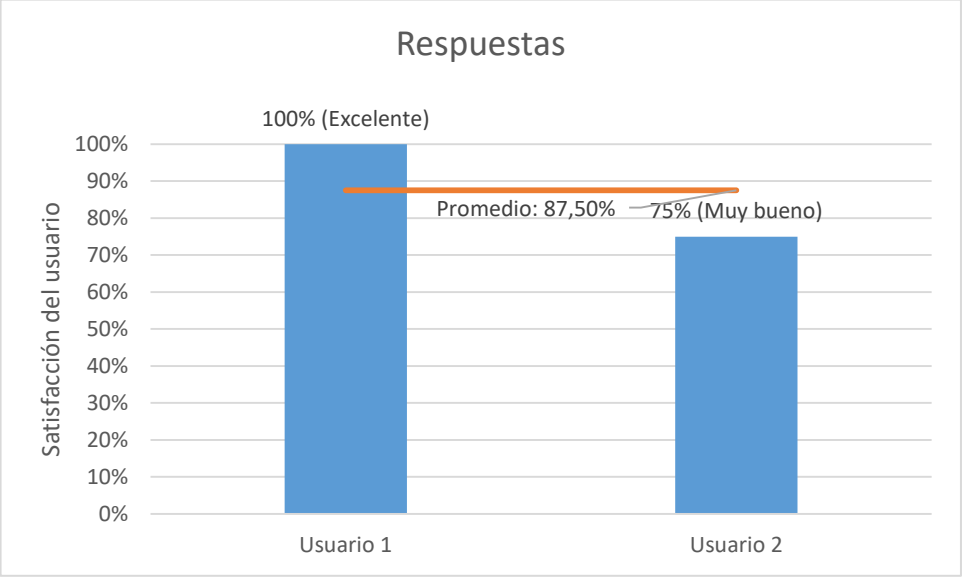
Respuesta	Porcentaje de satisfacción
Excelente	100%
Muy bueno	75%
Bueno	50%
Regular	25%
Malo	0%

Fuente: Elaboración propia

En base a tales valores, se procede a analizar las respuestas de ambos usuarios y a calcular su promedio, también expresado en porcentaje de satisfacción, mediante los siguientes gráficos:

Pregunta 1: ¿En qué grado permite la aplicación registrar los procesos de graduación de los maestrantes de la PUCESA de manera ágil y directamente en el estado en el que se encuentra actualmente cada uno?

Figura 53: Respuestas a la pregunta 1 de la encuesta de satisfacción

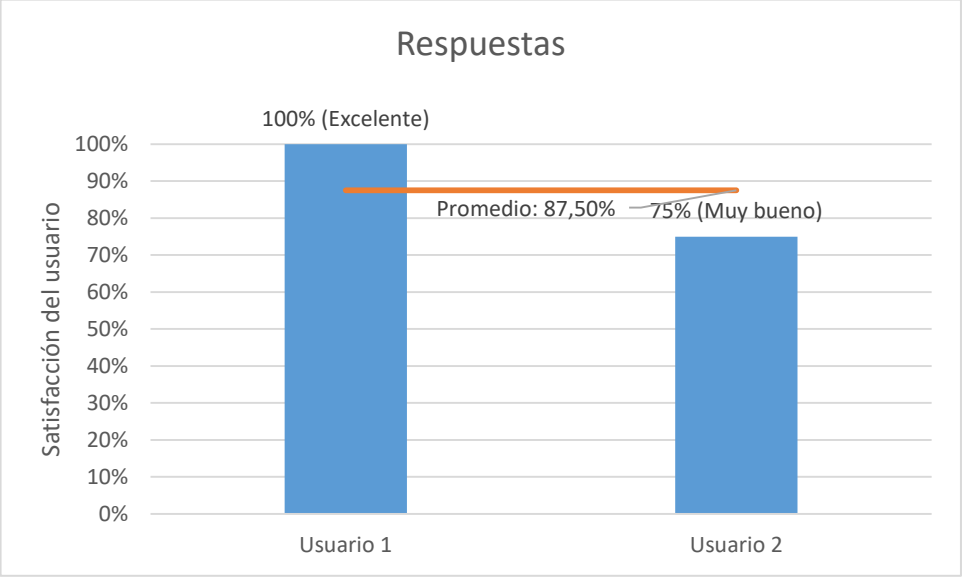


Fuente; Elaboración propia

Como se observa, existe un 87.5% de satisfacción general de los usuarios en cuanto a la agilidad en el registro de procesos de graduación de los maestrantes, que es una función clave dentro de la aplicación ya que permite alimentar su base de datos de manera rápida y directa, sin necesidad de pasar individualmente por todos los puntos del proceso de graduación para cada maestrante, hasta llegar a su estado actual.

Pregunta 2: ¿En qué grado permite la aplicación registrar las fechas correspondientes a cada paso del proceso de graduación de los maestrantes de la PUCESA con la finalidad de guardar información veraz y precisa de cada caso?

Figura 54: Respuestas a la pregunta 2 de la encuesta de satisfacción

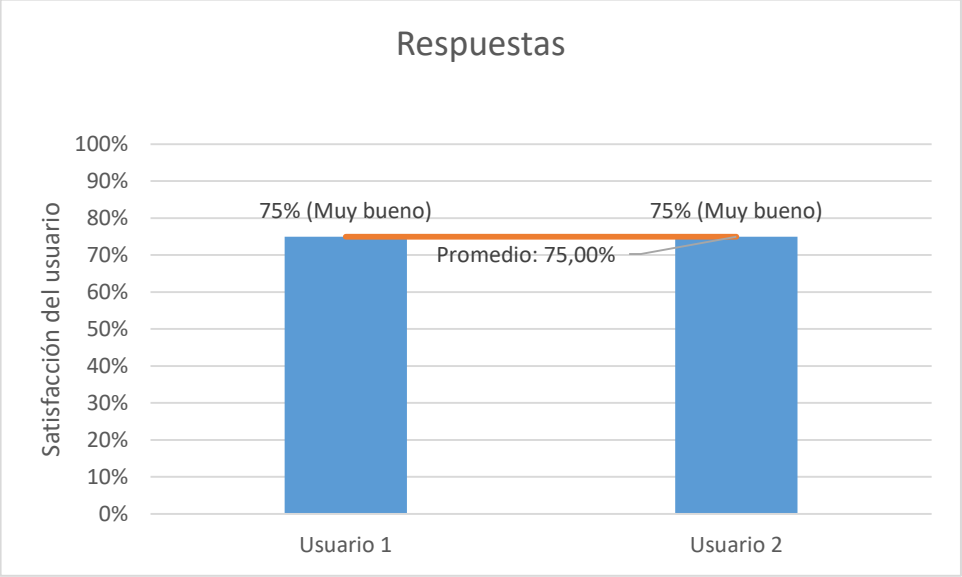


Fuente: Elaboración propia

Para la segunda pregunta también existe un 87.5% de satisfacción promedio de los usuarios encuestados, en este caso en cuanto a la posibilidad de registrar fechas correspondientes a cada evento dentro del proceso de graduación de los maestrantes. Esto es muy importante ya que permite almacenar información veraz que a su vez permitirá obtener estadísticas reales relativas a tiempos de cada punto del proceso de graduación.

Pregunta 3: ¿En qué grado detecta la aplicación cuando el usuario ingresa datos erróneos o realiza operaciones inconsistentes, desplegando mensajes de error detallados y comprensibles para el usuario?

Figura 55: Respuestas a la pregunta 3 de la encuesta de satisfacción

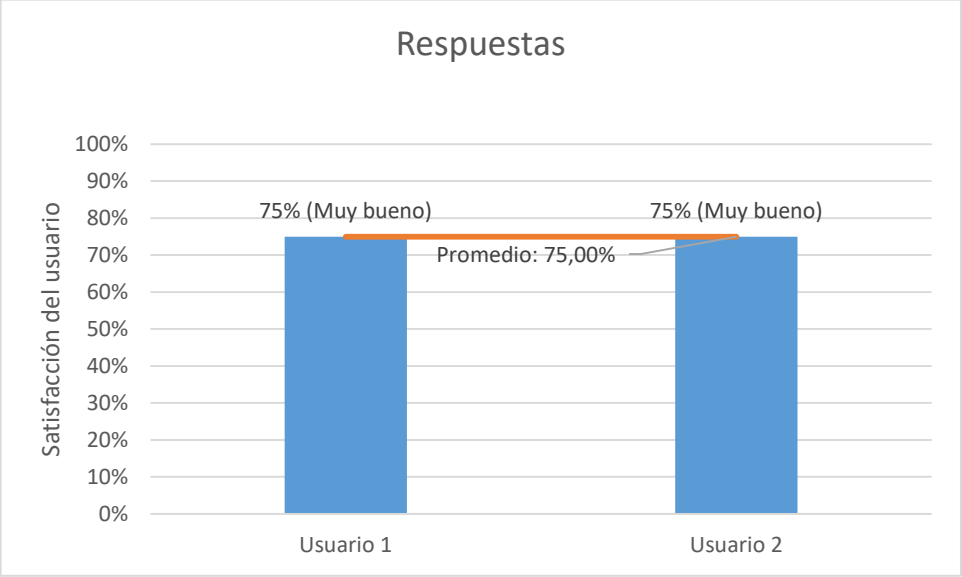


Fuente: Elaboración propia

En el gráfico se observa que existe un 75% de satisfacción promedio de los futuros usuarios de la aplicación en cuanto a la detección de errores y despliegue de mensajes adecuados y comprensibles por parte del usuario. Cabe mencionar que en algunas ocasiones, los mensajes de error desplegados pueden resultar demasiado técnicos para los usuarios, lo cual puede dificultar su comprensión.

Pregunta 4: ¿Lista la aplicación correctamente todos los casos de maestrantes de la PUCESA en estado activo, pasivo, graduados y que requieren tomar actualizaciones, con toda su información detallada pertinente, pudiéndose filtrar dicha información por carreras?

Figura 56: Respuestas a la pregunta 4 de la encuesta de satisfacción

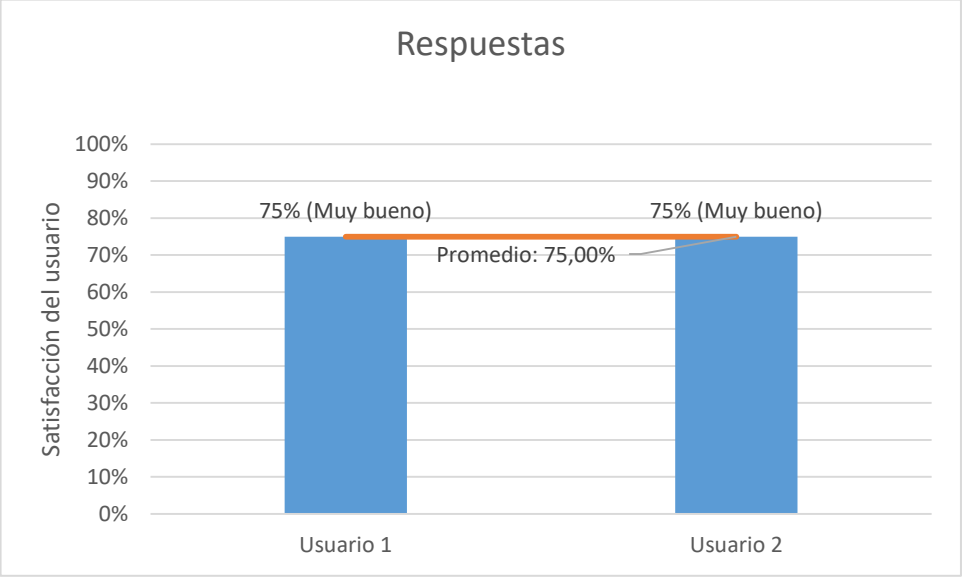


Fuente: Elaboración propia

En este caso se observa también un 75% de satisfacción general de los usuarios de la aplicación web desarrollada en relación al despliegue de listados de maestrantes activos, pasivos, graduados y que requieren tomar actualizaciones, otra función clave de la aplicación que permite al personal de la OIP obtener información veraz y actualizada sobre los casos de maestrantes que se encuentran en los diferentes estados posibles.

Pregunta 5: ¿Lista la aplicación correctamente todos los casos de maestrantes de la PUCESA que s encuentran en un determinado paso del proceso de graduación, con toda su información detallada pertinente, pudiéndose filtrar dicha información por carreras?

Figura 57: Respuestas a la pregunta 5 de la encuesta de satisfacción

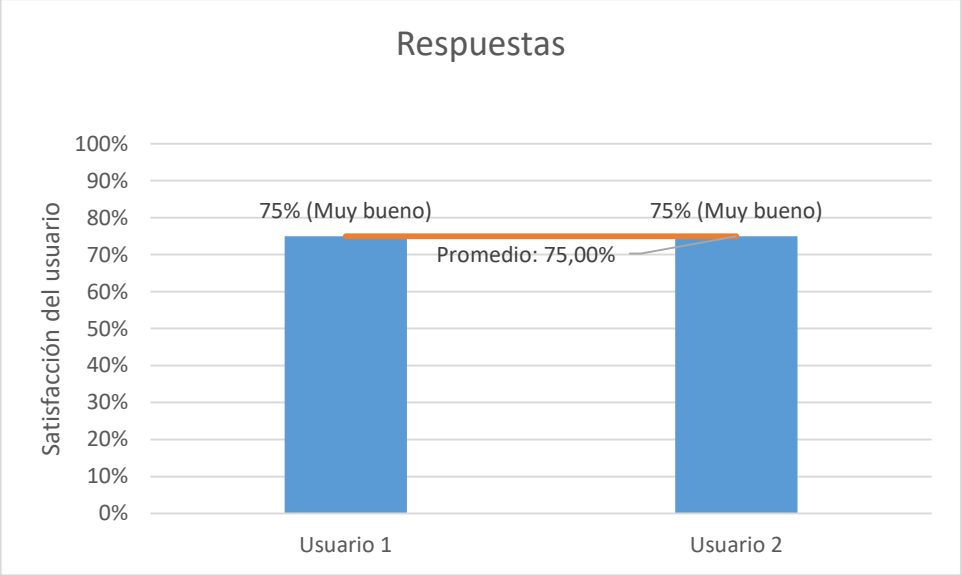


Fuente: Elaboración propia

En la pregunta 5 se obtiene también un 75% (muy bueno) de satisfacción promedio de los usuarios en lo que se refiere a listado de maestrantes en un determinado punto dentro del proceso general de graduación. Es también una función clave de la aplicación que tiene relación con el aspecto evaluado en la pregunta anterior, aunque su funcionalidad es incluso más detallada.

Pregunta 6: ¿En qué grado despliega y maneja la aplicación la información correspondiente a maestrantes, docentes y carreras de posgrados preexistentes en las bases de datos de la PUCESA?

Figura 58: Respuestas a la pregunta 6 de la encuesta de satisfacción

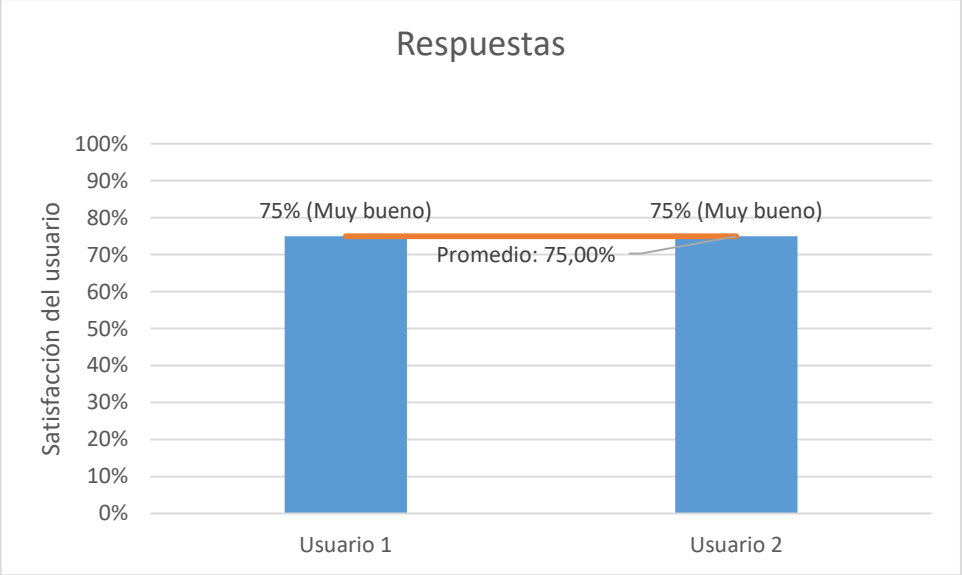


Fuente: Elaboración propia

Se obtiene también un 75% promedio de grado de satisfacción de los futuros usuarios de la aplicación en relación al uso de la información preexistente en las bases de datos de la PUCESA, la misma que se comparte con la presente aplicación mediante el uso de webservices.

Pregunta 7: ¿Permite la aplicación visualizar la información detallada de un maestrante específico incluyendo aquella relativa a su estado actual dentro del proceso de graduación?

Figura 59: Respuestas a la pregunta 7 de la encuesta de satisfacción

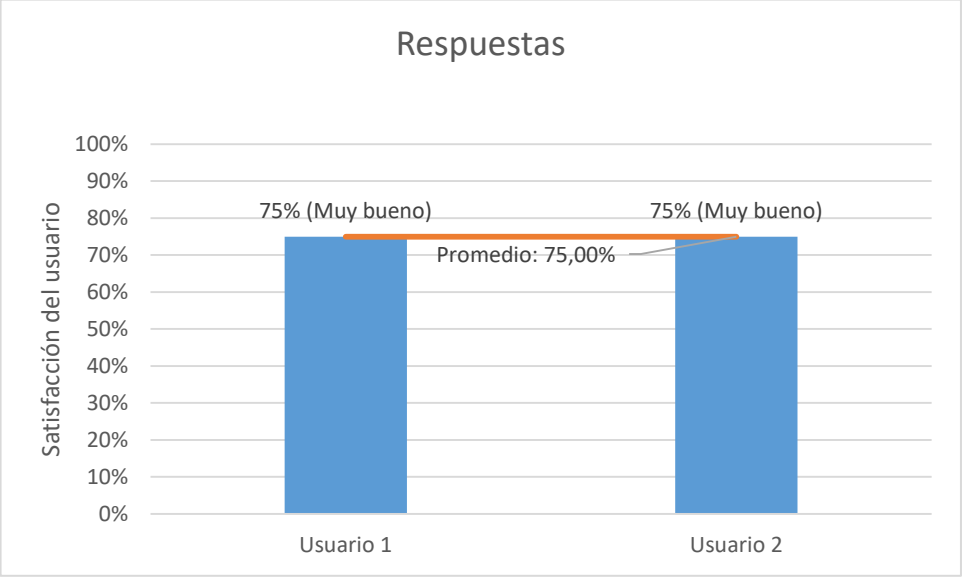


Fuente: Elaboración propia

El personal de la OIP se siente un 75% satisfecho con el despliegue de información referente a un maestrante específico. Cabe mencionar que es información bastante detallada e incluye el estado del maestrante en tiempo real en relación a su proceso de graduación. También refleja información veraz sobre si un maestrante requiere tomar actualizaciones para poder continuar con su proceso de graduación.

Pregunta 8: ¿Permite la aplicación visualizar la información de tallada de un docente específico incluyendo el número de proyectos en los que se encuentra trabajando actualmente ya sea en calidad de tutor, informante, director o revisor?

Figura 60: Respuestas a la pregunta 8 de la encuesta de satisfacción

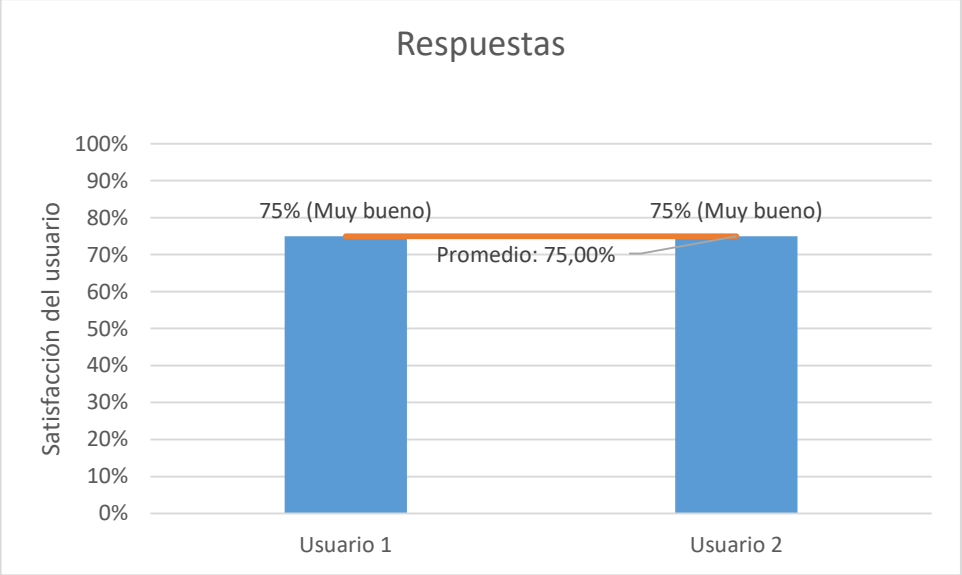


Fuente: Elaboración propia

Esta pregunta guarda bastante relación con la anterior, pero en este caso se refiere al despliegue de información detallada de un docente específico. Se incluye el número de proyectos total y detallado en los que se encuentra trabajando un docente, lo cual es un dato clave para el personal de la OIP. Se obtiene también un 75% de satisfacción general de los usuarios de la aplicación en relación a este aspecto.

Pregunta 9: ¿Permite la aplicación registrar y actualizar adecuadamente las direcciones de correo electrónico de maestrantes y docentes?

Figura 61: Respuestas a la pregunta 9 de la encuesta de satisfacción

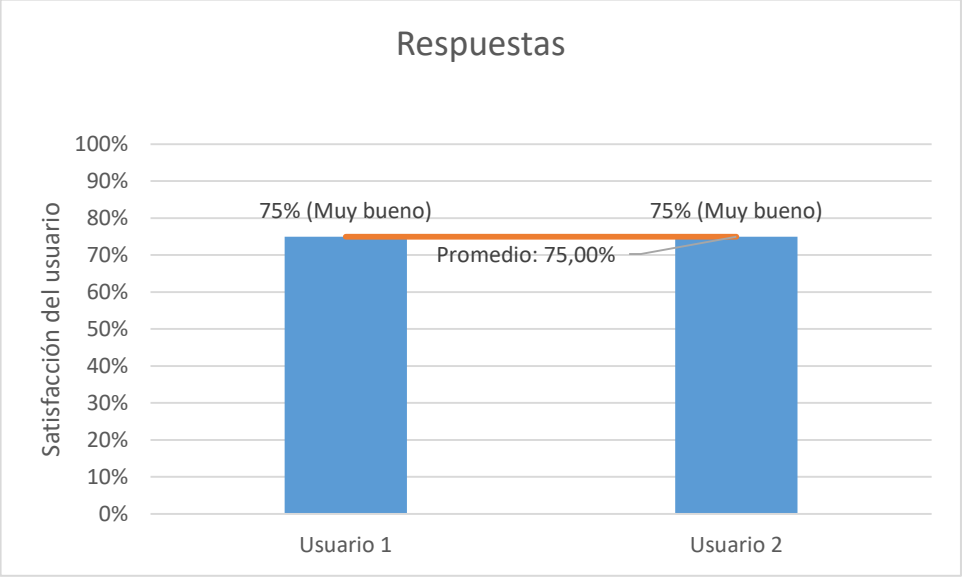


Fuente: Elaboración propia

Se mantiene la tendencia del 75% de satisfacción general de los usuarios de la aplicación para esta pregunta. En este caso se refiere a la posibilidad de mantener un registro de los correos electrónicos, tanto de docentes como de maestrantes. Esto es muy importante tanto para el envío de correos electrónicos generados automáticamente por el sistema, como para el envío manual de correos electrónicos por parte del personal de la OIP.

Pregunta 10: ¿En qué grado genera la aplicación los cuadros estadísticos requeridos por la OIP, reflejando verazmente los datos registrados sobre los procesos de graduación y estado de los maestrantes?

Figura 62: Respuestas a la pregunta 10 de la encuesta de satisfacción

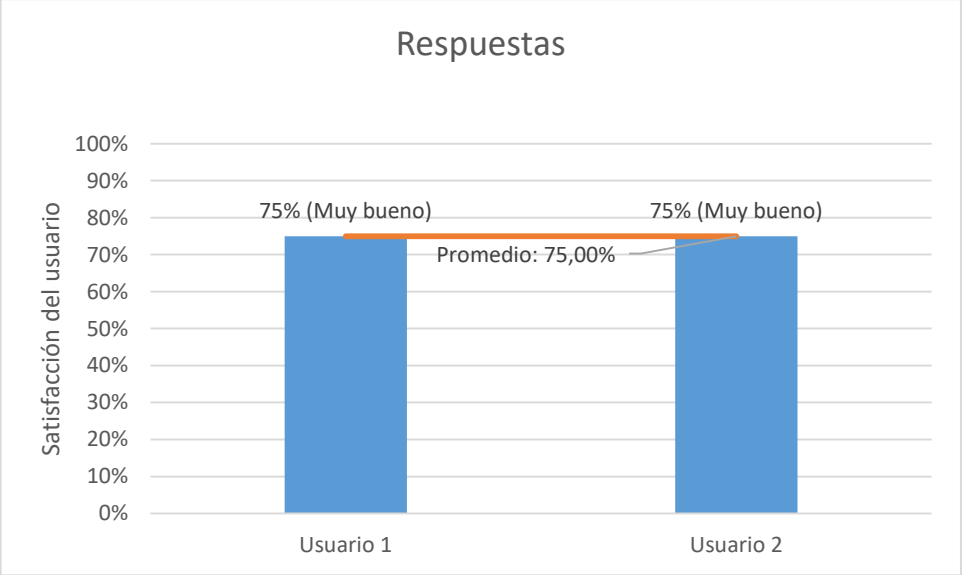


Fuente: Elaboración propia

Se observa también un 75% de satisfacción de los usuarios de la aplicación en cuanto a los diferentes cuadros estadísticos generados por la misma. Dichos cuadros reflejan información de índole general, como número de maestrantes que se encuentran en determinadas etapas del proceso de graduación. Tal información puede también ser filtrada por carreras y/o por rangos de fechas según sea pertinente.

Pregunta 11: ¿En qué grado considera usted que la aplicación es ágil y que el tiempo de carga de sus diferentes páginas web es reducido?

Figura 63: Respuestas a la pregunta 11 de la encuesta de satisfacción

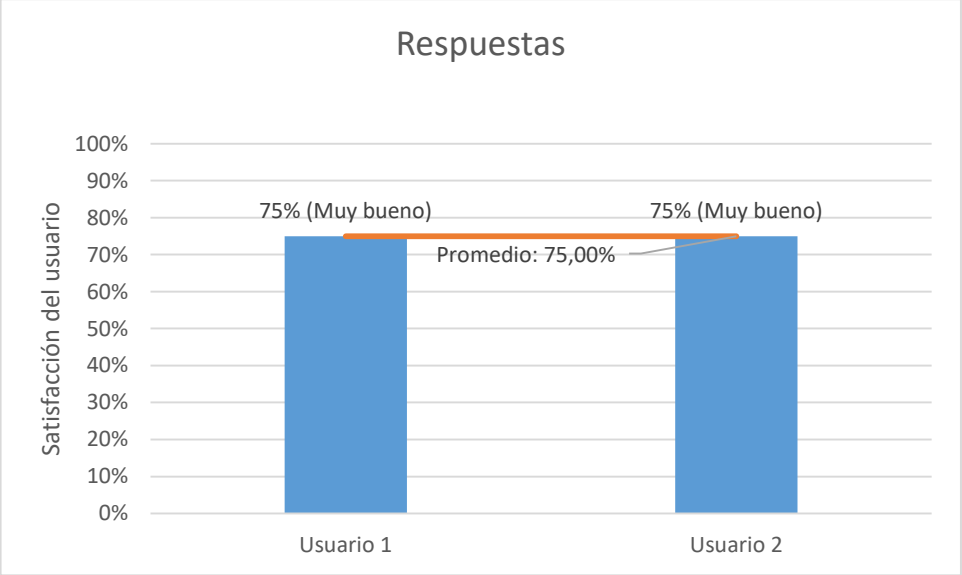


Fuente: Elaboración propia

Existe nuevamente un 75% de satisfacción de los usuarios de la aplicación web con respecto a su velocidad y tiempo de carga de sus páginas. Se debe mencionar que en muchas ocasiones, los webservices provistos por el personal del departamento de TI de la institución tardan demasiado tiempo en enviar la información requerida por la aplicación, lo que produce pérdidas de tiempo e incremento en el tiempo de carga de algunas páginas web de la aplicación de forma notoria.

Pregunta 12: ¿En qué grado considera usted que la aplicación es clara, intuitiva, amigable y fácil de usar, funcionando sin que requiera de conocimientos técnicos por parte del usuario?

Figura 64: Respuestas a la pregunta 12 de la encuesta de satisfacción

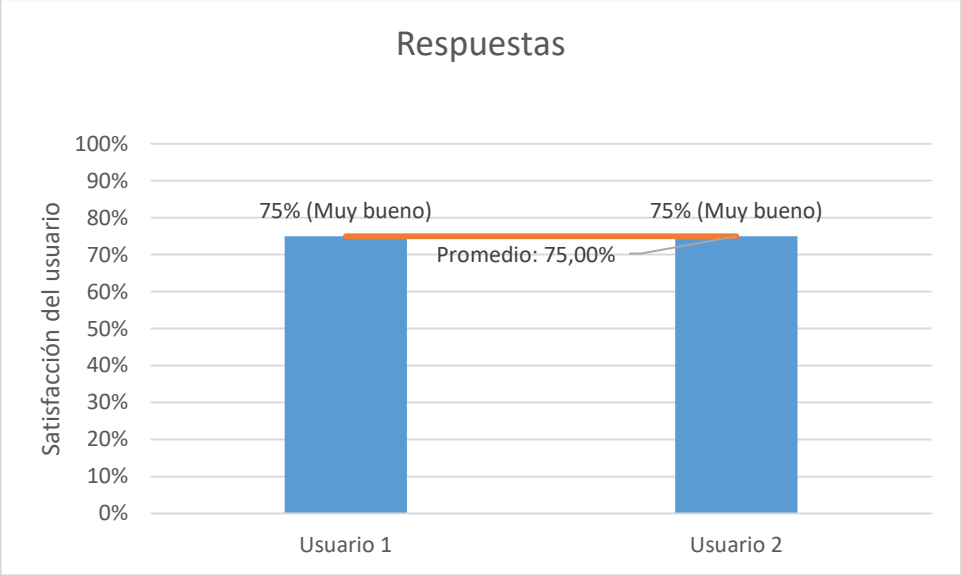


Fuente: Elaboración propia

Se puede deducir que existe también un 75% de satisfacción de los usuarios en cuanto a la usabilidad y claridad de la aplicación. Cabe mencionar que la misma ha sido diseñada considerando siempre la facilidad de uso y comodidad por parte del usuario, de modo que no se requieran conocimientos avanzados de ningún tipo para usar correctamente la aplicación.

Pregunta 13: En términos generales, ¿En qué grado considera usted que la aplicación cumple los objetivos para los que fue creada y será de la utilidad prevista para la OIP?

Figura 65: Respuestas a la pregunta 13 de la encuesta de satisfacción



Fuente: Elaboración propia

La última pregunta del cuestionario es una pregunta de evaluación general de la aplicación web desarrollada, por parte del usuario final de la misma. Al igual que en la gran mayoría de preguntas, se obtiene un 75% de satisfacción de los usuarios para con la aplicación, lo que equivale a decir que en términos generales, los usuarios de la misma la consideran una aplicación muy buena.

5.3. Análisis de resultados

De lo que se observa en los gráficos anteriores, se puede decir que en términos generales la aplicación satisface efectivamente las necesidades del usuario final de la misma, habiéndose cumplido con el objetivo principal de automatizar en la medida posible los procesos internos de atención a usuarios que tienen lugar en el OIP, y de informatizar el manejo de su información clave, cubriéndose así el vacío que existía previamente por la falta de una aplicación oportunamente desarrollada.

Cabe mencionar también que, luego de haber realizado una exhaustiva investigación acerca de las herramientas existentes en la PUCESA, las mismas que fueron previamente implementadas y utilizadas para la elaboración del sistema Academics [4] y a las cuales, por fuerza era necesario adaptarse, como lo son ASP.NET 4.0, Visual Studio 2010, controles *Obout* y Microsoft SQL Server 2008 R2, entre otras de menor alcance, se logró instalar, configurar, interconectar y aplicar todas ellas de manera efectiva, lográndose así una aplicación web compatible con Academics que no tendrá ningún problema en integrarse con la misma, tanto desde el punto de vista visual como funcional y operativo.

De igual forma, se logró acceder a los datos requeridos de la base de datos de Academics a través de los webservices habilitados por el departamento de informática de la institución, por lo que se agradece de manera especial al Ing. Mg. José Enríquez dada su oportuna colaboración cuando fue requerido, habiéndose logrado así utilizar eficazmente la información ya existente en la institución, evitándose así problemas como la duplicidad y la inconsistencia en la información.

Por último, se debe mencionar que, si bien la aplicación resultante de este proyecto es compatible con los programas y servicios instalados en la PUCESA, el trámite requerido para implementarla en sus servidores es bastante exigente y puede tomar mucho tiempo cumplir con todos los requerimientos técnicos y reglamentarios establecidos por el director de su departamento de informática, por lo que es necesario ejecutar la aplicación utilizando un servicio de hosting externo como lo es MyASP.NET, del que se comenta en párrafos anteriores. Esto permite que el personal del OIP pueda comenzar a utilizar la aplicación inmediatamente ésta comienza a funcionar online, lo cual requiere también bases de datos online instaladas preferentemente en los mismos servidores de MyASP.NET. Posteriormente, cuando la implementación de la aplicación en los servidores de la PUCESA sea factible, se requerirá simplemente de descargar la información de las bases de datos online para guardarla en las bases de datos locales implementadas en los servidores de la PUCESA, una vez que la aplicación también haya sido instalada, probada y esté lista para entrar en producción desde los mismos.

Capítulo 6

Conclusiones y Recomendaciones

6.1. Conclusiones

- Los procesos que tienen lugar actualmente en el OIP de la PUCESA en cuanto a atención a usuarios internos y externos, así como sus correspondientes flujos de información, se realizan en su gran mayoría de manera manual, apoyándose a lo sumo en herramientas aisladas de ofimática como son Microsoft Word y Microsoft Excel, lo cual crea inconsistencias en la información y retrasos en la respuesta a las demandas realizadas por ambos tipos de usuarios.
- Existen en la PUCESA las herramientas necesarias de hardware y software que se requieren para la implementación de una aplicación web que resuelva los problemas existentes en la actualidad con los flujos de información del OIP de la institución. Existe así mismo una base de datos institucional ya implementada como parte de la plataforma Academics, que contiene mucha de la información que se requeriría para automatizar dichos flujos de información.
- Se crea una aplicación web utilizando las herramientas y tecnologías existentes en la PUCESA, diseñada para integrarse perfectamente en sus servidores y funcionar conjuntamente con la aplicación Academics, recuperando la información requerida de sus bases de datos, con la finalidad de sistematizar el manejo de la información generada por la OIP de la institución, procesarla y ofrecerle a su personal administrativo información estadística que le sirva efectivamente como herramientas de apoyo en la toma de decisiones.
- Se implementa la aplicación desarrollada utilizando un servicio de hosting externo para permitir que el personal de la OIP de la PUCESA puede comenzar a usarla cuanto antes, quedando la misma lista para ser implementada en los servidores de la PUCESA una vez que se cumplan las revisiones técnicas y reglamentarias impuestas por su departamento de informática.

- No existe un modelo estándar a seguir para el desarrollo de aplicaciones institucionales para la PUCESA, lo cual ocasiona que la persona interesada en desarrollar una aplicación para dicha institución tenga que reunirse periódicamente con el personal del departamento de TI para discutir los pormenores propios de tal tarea como son la plataforma de desarrollo, el lenguaje de programación, el SGBD, las técnicas de programación y de arquitectura de software y demás. Esto obviamente ocasiona pérdidas considerables de tiempo tanto al desarrollador externo como al personal de TI de la institución, y aún así existe una alta probabilidad de que se termine desarrollando un producto con problemas de compatibilidad para integrarse con el software ya existente en la PUCESA.

6.2. Recomendaciones

- Actualizar las herramientas tecnológicas tanto de hardware como de software existentes en la PUCESA con la finalidad de poder desarrollar e implementar para sus diferentes departamentos aplicaciones más potentes, ágiles y que aprovechen las nuevas características de seguridad, fiabilidad y escalabilidad que ofrecen las nuevas versiones de las herramientas de desarrollo de software, así como para mantener una base de datos centralizada que maneje efectivamente la información de toda la institución.
- Crear y publicar un manual para el desarrollo de aplicaciones web institucionales para la PUCESA, el mismo que defina los lineamientos que deberá seguir cualquier persona interesada en desarrollar software para la institución. Dichos lineamientos deberían incluir metodología de desarrollo de software, plataforma de desarrollo, lenguajes de servidor y de programación a emplearse, modelo de desarrollo de software, SGBD a emplearse, políticas de nombres para variables, métodos, clases, miembros, campos y demás, técnicas de conexión y manejo de bases de datos, y políticas de legibilidad y mantenibilidad del código.
- Implementar un sistema de gestión documental en la PUCESA que automatice y facilite las tareas de generación, envío y procesamiento de documentos institucionales tales como solicitudes, oficios y memorándums, el mismo que maneje de manera integral la documentación correspondiente a todos los departamentos de la institución.

Apéndice A

Descripción de la base de datos “Graduacion” y sus tablas

Como se menciona en el cuerpo principal del documento, la presente aplicación utiliza dos bases de datos que se comparten con otra aplicación que se desarrolla paralelamente en [1]. El proyecto para dicha aplicación se titula “Desarrollo de una aplicación web para la administración de procesos de graduación de una unidad de posgrados”, y su base de datos principal se denomina “Graduacion”, y, en vista de que debe integrarse con la presente aplicación y con Academics [4], también se desarrolló usando el SGBD Microsoft SQL Server 2008 R2.

Cabe mencionar que no todas las tablas de dicha base de datos resultan útiles para la presente aplicación y, por otro lado, a algunas de ellas ha sido necesario ampliarles su funcionalidad mediante la adición de campos extra, lo cual no afecta la funcionalidad para la que originalmente fue creada tal base de datos con su respectiva aplicación.

A continuación se procede a describir únicamente aquellas tablas que resultan de utilidad para la presente aplicación. Los campos que se encuentran sombreados son aquellos que se ha añadido al diseño original de la base de datos para dar soporte a la presente aplicación.

A.1. Tabla TemasdeTesis

Se encarga de guardar la información correspondiente a los temas de proyectos de investigación y desarrollo de los maestrantes en proceso de graduación. En la siguiente tabla se puede ver la descripción de sus campos.

Tabla 22: Campos de la tabla "TemadeTesis"

Campo	*Tipo*	*Descripción*
TemaId	Entero	Clave principal
tema	Varchar(MAX)	Es el tema propiamente dicho. Puede ser un texto largo
directorId	Entero	Identificador del maestro director de tesis
resumen	Varchar(MAX)	Es el resumen correspondiente al tema de proyecto de titulación. Puede ser un texto largo
directorAprobado	Bit	Especifica si el director de tesis está aprobado para el presente tema
activo	Bit	Especifica si el tema de proyecto de titulación se encuentra activo
estado	Entero	Guarda un valor que indica el estado en el que se encuentra el presente tema de proyecto de titulación
estudianteId	Entero	Identificador del Estudiante al que corresponde el tema
fechaAprobacion	Fecha/Hora	Fecha y hora en las que ha sido aprobado el tema en cuestión por parte del director de la OIP
fechaUltimoEnvio	Fecha/Hora	Fecha y hora en la que el tema ha sido enviado para revisión / aprobación por última vez
usaPlataforma	Bit	Indica si el maestrante ha usado la otra aplicación a desarrollarse para enviar su tema del proyecto de investigación y desarrollo y solicitar su aprobación
solicitudAprobacionIngresada	Bit	Indica si el maestrante ha presentado ya su solicitud física (impresa) de aprobación del tema
fechaIngresoSolicitudAprobacion	Fecha/Hora	Si la solicitud física (impresa) ya ha sido presentada por el maestrante, este campo indica la fecha de dicha solicitud
carreraId	Entero	Identificador de la carrera de posgrado a la que pertenece el maestrante
estudianteReprobadoId	Entero	Identificador del maestrante cuando su proyecto de grado ha sido reprobado por cualquier motivo, en cuyo caso el campo estudianteId queda vacío

Fuente: Elaboración propia

A.2. Tabla PlanesdeTesis

Esta tabla almacena información sobre los planes de proyectos de investigación y desarrollo elaborados y presentados por los maestrantes como parte de su proceso de graduación. Se procede a describir sus campos.

Tabla 23: Campos de la tabla "PlanesdeTesis"

Campo	*Tipo*	*Descripción*
plandeTesisId	Entero	Identificador del plan de titulación, que sirve como clave principal
TemaId	Entero	Clave foránea que indica con qué registro se relaciona en la tabla "TemasdeTesis"
CarreraId	Entero	Contiene un identificador que permite saber a qué carrera pertenece el plan en cuestión. La información sobre dichas carreras se encuentra almacenada en las bases de datos preexistentes de Academics
TipoProyecto	Entero	Permite identificar mediante un número el tipo de proyecto al que se refiere el plan en cuestión
ClaseProyecto	Entero	Permite identificar mediante un número la clase de proyecto a la que se refiere el plan en cuestión
Costo	Decimal	Permite almacenar el valor monetario estimado que costará el desarrollo del proyecto en cuestión
Duración	Varchar(50)	Almacena la duración aproximada que se estima tomará el desarrollo del proyecto en cuestión
Lugar	Varchar(200)	Almacena el lugar donde principalmente se desarrollará el proyecto en cuestión
DocenteId	Entero	Almacena un valor numérico que identifica al docente designado como director del proyecto. La información sobre los docentes se encuentra almacenada en las bases de datos de Academics
DocenteTituloPregradoId	Entero	Almacena un valor numérico que identifica el título de pregrado del docente director de proyecto que más se ajusta al proyecto en cuestión
DocenteTituloPosgradoId	Entero	Almacena un valor numérico que identifica el título de posgrado del docente director de proyecto que más se ajusta al proyecto en cuestión
DocenteTituloPosdoctoralId	Entero	Permite identificar el título de formación posdoctoral del docente director del proyecto que más se ajusta a la realidad del mismo
EstudianteTituloPregradoId	Entero	Permite identificar el título de pregrado del maestrante que más se ajusta a la realidad de la carrera y proyecto
VinculoPuce	Bit	Indica si el docente director del proyecto tiene vínculo con la Universidad Católica
FechaOriginal	Fecha/Hora	Almacena la fecha en la que fue enviado el plan por primera vez por parte del maestrante
OtraUniversidad	Varchar(100)	Almacena la información de otra Universidad de vínculo con el director del proyecto, en caso de no ser la PUCE
Resumen	Varchar(MAX)	Almacena el resumen propio del plan de titulación, que puede ser igual al del tema o puede haber variado

DescripcionProblema	Varchar(MAX)	Guarda información sobre la descripción detallada del problema que se pretende solucionar mediante el desarrollo del presente proyecto
Meta	Varchar(MAX)	Permite guardar la información correspondiente a la meta que se persigue
VariableDependiente	Varchar(MAX)	Guarda información correspondiente a la variable independiente, útil sobre todo en el caso de Investigaciones
ObjetivoGeneral	Varchar(MAX)	Almacena el objetivo general del presente proyecto
EstadodelArte	Varchar(MAX)	Almacena el estado del arte para el presente plan de titulación
Metodología	Varchar(MAX)	Permite guardar información sobre la metodología general que se prevé aplicar para el desarrollo del presente proyecto
PoblacionyMuestra	Varchar(MAX)	Guarda información sobre poblaciones y muestras, útil sobre todo en el caso de investigaciones
MetodoGeneral	Varchar(MAX)	Almacena información sobre el método general a aplicarse para el desarrollo del proyecto en cuestión
EvaluacionPreliminar	Varchar(MAX)	Almacena información sobre las técnicas de evaluación preliminares que se aplicará para el presente proyecto
HipotesisdeTrabajo	Varchar(MAX)	Almacena información útil sobre todo en el caso de investigaciones.
VariableIndependiente	Varchar(MAX)	Permite guardar información muy útil en el caso de las investigaciones
PreguntadeEstudio	Varchar(MAX)	Guarda la pregunta principal que da origen al desarrollo del proyecto en cuestión
MetodoEspecifico	Varchar(MAX)	Almacena detalles sobre la metodología a aplicarse en el proyecto en cuestión
DocenteLectorId	Entero	Permite identificar al docente investigador designado por la OIP para la revisión de un determinado plan de titulación
Estado	Entero	Es un campo clave dentro del proceso de graduación del estudiante de posgrados, ya que permite conocer en qué estado se encuentra el plan de titulación elaborado y/o enviado por el estudiante
Activo	Bit	Permite identificar si el plan en cuestión se considera activo o no
FechaUltimoEnvio	Fecha/Hora	Almacena la fecha en la que por última vez el maestrante envió su plan de titulación
FechaAprobacion	Fecha/Hora	Almacena la fecha en la que un plan de titulación es definitivamente aprobado por la dirección de la OIP
ComoAparece	Varchar(MAX)	Almacena las respuestas a dichas preguntas en relación del tema de proyecto de titulación a desarrollarse
PorqueseOrigina	Varchar(MAX)	
QueloOrigina	Varchar(MAX)	
CuandoseOrigina	Varchar(MAX)	
DondeseOrigina	Varchar(MAX)	

DondeseDetecta	Varchar(MAX)	
fechaSolicitudInformante	Fecha/Hora	Fecha en la que el maestrante presenta su solicitud impresa de docente informante
fechaDesignacionInformante	Fecha/Hora	Fecha en la que la OIP designa docente informante para el plan de un maestrante que lo ha solicitado
fechaSolicitudAprobacion	Fecha/Hora	Fecha en la que el maestrante presenta su solicitud impresa de aprobación del plan

Fuente: Elaboración propia

A.3. Tabla AvancesTesis

Esta tabla guarda información referente a los avances enviados por los maestrantes durante la fase de desarrollo de su proyecto de investigación y desarrollo. Para el presente proyecto, esta tabla es importante debido a que interviene en el proceso de designación de docentes revisores para el mismo.

Tabla 24: Campos de la tabla "AvancesTesis"

Campo	***Tipo***	***Descripción***
AvanceTesisId	Entero	Clave principal
TesisId	Entero	Permite identificar el registro de la tabla TesisPosgrados con el que se relaciona el presente registro
Estado	Entero	Permite identificar el estado actual en el que se encuentra un avance de proyecto de titulación. Puede contener 3 valores: 0, que significa que el maestrante apenas ha enviado el trabajo, 1 que significa que está en revisión por parte de su docente director, y 2 que significa que el avance ha sido aprobado por el mismo
Porcentaje	Entero	Almacena el porcentaje al cual corresponde el presente avance de proyecto
Iteración	Entero	Permite identificar el número de veces que el maestrante ha enviado correcciones a un determinado nivel de avances de su proyecto
FechaEnvio	Fecha/Hora	Almacena la fecha en la que el maestrante envió el respectivo avance
FechaAprobacion	Fecha/Hora	Almacena la fecha en la que el avance fue definitivamente aprobado

Fuente: Desarrollo de una Aplicación Web para la Administración de Procesos de Graduación de una Unidad de Postgrados ([1])

A.4. Tabla CorreccionesLectoresTesis

Al igual que en el caso de la tabla anterior, esta tabla interviene en el proceso de designación y registro de docentes revisores para los proyectos de los maestrantes. Se procede a describir sus campos.

Tabla 25: Campos de la tabla "CorreccionesLectoresTesis"

Campo	*Tipo*	*Descripción*
CorreccionesLectoresTesisId	Entero	Clave principal
DocenteId	Entero	Permite identificar el docente lector al que corresponde el registro en cuestión de esta tabla
AvanceId	Entero	Permite identificar el registro de la tabla "AvancesTesis" con el que se relaciona el registro de esta tabla
Estado	Entero	Almacena un valor que permite identificar el estado en el que se encuentra este registro. Los posibles valores son 0, que indica que la corrección aún no ha sido enviada por el maestrante, 1 que significa que ya está a cargo de su docente director de proyecto, y 2 significa que la corrección ha sido definitivamente aprobado por un docente lector
FechaHoraEnvio	Date/Time	Almacena la fecha y hora exactas en la que se creó el registro en cuestión
FechaAprobacion	Date/Time	Fecha en la que la corrección fue definitivamente aprobada por un docente lector
Nota	Decimal	Registra la calificación con la que el docente lector aprueba una proyecto de titulación

Fuente: Desarrollo de una Aplicación Web para la Administración de Procesos de Graduación de una Unidad de Postgrados ([1])

A.5. Tabla TesisPosgrados

Esta tabla se encarga de guardar la información general referente a los proyectos de investigación y desarrollo elaborados por los maestrantes en proceso de graduación, y resulta de vital importancia para el presente desarrollo de software. Cabe mencionar que a la presente tabla se le ha realizado un número considerable de modificaciones que se ha coordinado debidamente con la autora De la Torre a fin de no afectar a la funcionalidad para la que originalmente fue creada y diseñada la tabla dentro de su proyecto de desarrollo de software. Tales cambios consisten en la adición de campos que en su

mayoría son de tipo Fecha/Hora, y que sirven para registrar las fechas en los que tienen lugar los diferentes eventos correspondientes al proceso de graduación de un maestrante, lo cual servirá principalmente para la elaboración de cuadros estadísticos, como se explicó en un apartado anterior.

A continuación, se procede a describir los campos de la tabla.

Tabla 26: Campos de la tabla "TesisPosgrados"

Campo	*Tipo*	*Descripción*
TesisPosgradosId	Entero	Clave principal
PlanId	Entero	Permite identificar el plan de titulación al que se encuentra vinculado el presente registro
FechaHoraDefensa	Fecha/Hora	Almacena la fecha y hora para la defensa del proyecto en cuestión, definidas por el DPI
Estado	Entero	Indica el estado actual en el que se encuentra el trámite de aprobación de un proyecto de titulación dado
FechaAprobacionDip	Fecha/Hora	Registra la fecha en la que el proyecto fue definitivamente aprobado por la dirección de la OIP
directorId	Entero	Identificador del docente director del proyecto
docenteRevisorId1	Entero	identificador del primer docente revisor del proyecto
docenteRevisorId2	Entero	identificador del segundo docente revisor del proyecto
fechaSolicitudDirector	Fecha/Hora	Fecha en la que el maestrante solicita por escrito director para su proyecto
fechaDesignacionDirector	Fecha/Hora	Fecha en la que el personal de la OIP designa director para un proyecto determinado
fechaSolicitudRevisores	Fecha/Hora	Fecha en la que el maestrante solicita por escrito docentes revisores para su proyecto
fechaDesignacionRevisores	Fecha/Hora	Fecha en la que el personal de la OIP designa los dos docentes revisores para un proyecto determinado
fechaAprobacionRevisores	Fecha/Hora	Última fecha en la que los docentes revisores envían su informe favorable para un proyecto determinado
fechaSolicitudTribunal	Fecha/Hora	Fecha en la que el maestrante solicita por escrito la designación del tribunal para su defensa
fechaDesignacionTribunal	Fecha/Hora	Fecha en la que el personal de la OIP designa tribunal para la defensa de un proyecto determinado
fechaRevisionBiblioteca	Fecha/Hora	Fecha en la que el maestrante acude a la OIP para registrar la revisión de sus requisitos en biblioteca, presentando el certificado emitido por su personal, entre otros requisitos

fechaSolicitudDefensa	Fecha/Hora	Fecha en la que el maestrante solicita por escrito la designación de fecha para la defensa de su proyecto
fechaDesignacionDefensa	Fecha/Hora	Fecha en la que el personal de la OIP designa fecha para la defensa de un proyecto determinado
fechaRegistroGraduacion	Fecha/Hora	Fecha en la que se registra la graduación de un maestrante, lo que corresponde a una defensa exitosa
segundaFechaHoraDefensa	Fecha/Hora	Fecha y hora designadas por el personal de la OIP para segunda oportunidad de defensa de un proyecto
miembro1Id	Entero	Identificador del primer miembro del tribunal designado por la OIP para la defensa de un proyecto
miembro2Id	Entero	Identificador del segundo miembro del tribunal designado por la OIP para la defensa de un proyecto
miembro3Id	Entero	Identificador del tercer miembro del tribunal designado por la OIP para la defensa de un proyecto
causaReprobacion	Varchar(MAX)	Descripción de la causa por la que la defensa de un proyecto es reprobada en primera instancia
segundaCausaReprobacion	Varchar(MAX)	Descripción de la causa por la que la defensa de un proyecto es reprobada en segunda instancia

Fuente: Elaboración propia

A.5.1. Posibles estados de un proyecto de investigación y desarrollo

El campo “estado” de la tabla TesisPosgrados, descrito en la tabla anterior, tiene la función de almacenar un valor que indica el estado en el que se encuentra un determinado proyecto de investigación y desarrollo, correspondiente a un maestrante. Para este campo, sin embargo, fue necesario redefinir los posibles valores que podía contener y su significado, lo cual también se realiza en coordinación con la Ing. Gabriela De la Torre a fin de no afectar en nada a su proyecto.

A continuación se procede a describir los nuevos valores que puede contener el campo estado de la tabla TesisPosgrados.

Tabla 27: Posibles valores para el campo "estado" de la tabla "TesisPosgrados"

Valor	*Significado*
10	El registro de la tabla TesisPosgrados apenas ha sido creado automáticamente por cualquiera de las dos aplicaciones que tienen acceso a dicha tabla, y significa que el maestrante ha solicitado por escrito docente director para su proyecto
20	El docente director ha sido designado por el personal de la OIP para el proyecto
30	El maestrante se ha matriculado para iniciar con la etapa de desarrollo del proyecto, lo cual le faculta para hacerlo oficialmente. En este estado, el maestrante puede solicitar actualización de su porcentaje de avance, la misma que debe venir acompañada de un certificado emitido por su docente director
40	El maestrante ha solicitado por escrito designación de docentes revisores, para lo cual debe haber concluido al 100% la etapa de desarrollo de su proyecto
50	Los dos docentes revisores han sido designados por el personal de la OIP para el proyecto, con lo cual arranca oficialmente el proceso de docentes revisores
60	Ambos docentes revisores han emitido su informe favorable para el proyecto, con lo cual finaliza la etapa de proceso de docentes revisores
70	El maestrante ha solicitado por escrito la designación del tribunal calificador para la defensa de su proyecto
80	El tribunal calificador ha sido designado por el personal de la OIP
90	El maestrante ha solicitado que se registre la revisión de sus requisitos en biblioteca previa la autorización para elaborar los empastados de su proyecto
100	El maestrante ha solicitado por escrito fecha y hora para la defensa de su proyecto
110	La fecha y hora para la defensa del proyecto han sido designadas por el personal de la OIP
120	El maestrante ha ejecutado exitosamente la defensa de su proyecto y por tanto se registra como graduado
130	Por algún motivo el maestrante ha fallado su defensa y no se ha graduado. Si es la primera vez que sucede, tendrá derecho a una segunda oportunidad para defender su proyecto de acuerdo a los reglamentos internos de la OIP

Fuente: Elaboración propia

Apéndice B

Descripción de la base de datos “AtencionUsusario” y sus tablas

La base de datos creada especialmente para la presente aplicación se denomina “AtencionUsuario”, si bien se usa también desde la aplicación paralela que se desarrolla en [1], por los motivos que se ha expuesto en párrafos anteriores. Cabe recalcar también que mucha de la información utilizada por la presente aplicación se encuentra almacenada en la base de datos creada específicamente para la aplicación que se desarrolla en [1], sin embargo, que es utilizada también por la presente aplicación y, por otro lado, otra parte de la información proviene de las bases de datos de Academics [4], a través de los webservices habilitados por el personal del departamento de informática de la PUCESA. Por tal razón, la base de datos “AtencionUsuario” propia de esta aplicación pareciera contener pocas tablas en relación a la funcionalidad ofrecida por la misma.

A continuación, se procede a describir dichas tabla y sus campos.

B.1. Tabla DocentesTitulos

Esta tabla tiene como objetivo guardar los títulos pertenecientes a los docentes de posgrado registrados en la base de datos de Academics. Esta información resulta importante en algunos puntos de ambas aplicaciones que acceden a esta base de datos, por ejemplo, para la elaboración del plan del proyecto de investigación y desarrollo, ya que se evita que los maestrantes puedan cometer errores al digitar los títulos de los docentes puesto que pueden elegirlos de una lista, esto en la aplicación que se desarrolla paralelamente en [1].

Tabla 28: Campos de la tabla "DocentesTitulos"

Campo	*Tipo*	*Descripción*
docenteTituloId	Entero	Identificador del registro
titulo	Varchar(200)	Descripción del título del docente
institucion	Varchar(100)	Descripción de la institución en la que el docente obtuvo el título
anio	Varchar(4)	Año en que el docente obtuvo el título
codigoRegistroSenescyt	Varchar(20)	Código con el que el título se encuentra debidamente registrado en el Senescyt o la institución gubernamental de educación superior equivalente
tipo	Entero	Contiene un valor numérico que indica el tipo de título al que hace referencia: 1 significa título de pregrado, 2 significa título de maestría o equivalente, 3 significa título de PhD o equivalente
docenteId	Entero	Identificador del docente al que pertenece el título registrado

Fuente: Elaboración propia

B.2. Tabla EstudiantesTitulos

Como su nombre lo indica, esta tabla se encarga de almacenar información acerca de los títulos de los maestrantes registrados en las bases de datos de la institución (a través de la plataforma Academics). Esta información puede también resultar muy importante en muchos puntos de la aplicación. A continuación se procede a describir sus campos.

Tabla 29: Campos de la tabla "EstudiantesTitulos"

Campo	*Tipo*	*Descripción*
estudianteTituloId	Entero	Identificador del registro
titulo	Varchar(200)	Descripción del título del maestrante
institucion	Varchar(100)	Descripción de la institución en la que el maestrante obtuvo el título
anio	Varchar(4)	Año en que el maestrante obtuvo el título
codigoRegistroSenescyt	Varchar(20)	Código con el que el título se encuentra debidamente registrado en el Senescyt o la institución gubernamental de educación superior equivalente
estudianteId	Entero	Identificador del maestrante al que pertenece el título registrado

Fuente: Elaboración propia

B.3. Tabla EmailUsuario

Esta tabla tiene la función de almacenar las direcciones de correo electrónico tanto para docentes como para maestrantes registrados en las bases de datos de la institución. Las mismas pueden ser actualizadas por solicitud expresa de cualquiera de los dos tipos de usuarios. A continuación se describen los campos de esta tabla.

Tabla 30: Campos de la tabla "EmailUsuario"

Campo	*Tipo*	*Descripción*
emailUsuarioId	Entero	Identificador del registro
tipoUsuario	Entero	Contiene un valor entero de acuerdo a la siguiente especificación: 0 significa maestrante, 1 docente
usuarioId	Entero	Identificador bien del maestrante o del docente para el que se registra la dirección de e-mail
email	Varchar(100)	Dirección de correo electrónico del usuario

Fuente: Elaboración propia

B.4. Tabla ParametrosPosgrados

Esta tabla contiene un solo registro que se encarga de almacenar la información correspondiente a parámetros generales de la aplicación, los mismos que pueden ser modificados desde ésta en el momento en el que sea requerido. Sus campos son los que se describe a continuación.

Tabla 31: Campos de la tabla "ParametrosPosgrados"

Campo	*Tipo*	*Descripción*
parametorsPosgradosId	Entero	Identificador del registro
urlRegistroEmail	Varchar(MAX)	Dirección URL de la página de la aplicación que permite actualizar las direcciones de correo electrónico tanto para maestrantes como para docentes
nombreDirectorDIP	Varchar(MAX)	Nombre del director o coordinador de la OIP
tituloDirectorDIP	Varchar(20)	Título académico del director o coordinador de la OIP
cargoDirectorDIP	Varchar(MAX)	Cargo que desempeña el director o coordinador de la OIP
nombreSecretariaDIP	Varchar(MAX)	Nombre de la secretaria de la OIP
usuarioSecretariaDIP	Varchar(20)	Usuario que utilizará la secretaria de la OIP para acceder a la aplicación
passwordSecretariaDIP	Varchar(MAX)	Contraseña encriptada que utilizará la secretaria de la OIP para acceder a la aplicación

Fuente: Elaboración propia

B.5. Tabla PorcentajesAvaladosTesis

Esta tabla tiene la función de guardar los valores correspondientes a porcentajes de avances de los maestrantes que se encuentran en etapa de desarrollo del proyecto. La aplicación brinda una interfaz adecuada para registrar este valor para cualquier maestrante específico, siempre que se encuentre en la mencionada etapa y que su solicitud venga acompañada por un oficio de su docente director certificando que el desarrollo del proyecto efectivamente se encuentra en el porcentaje de avance en cuestión.

Los campos de esta tabla son los siguientes.

Tabla 32: Campos de la tabla "PorcentajesAvaladosTesis"

Campo	*Tipo*	*Descripción*
porcentajesAvaladosTesisId	Entero	Identificador del registro
tesisPosgradosId	Entero	Identificador del proyecto para el cual se registra el porcentaje de avance
porcentaje	Entero	Porcentaje de avance registrado

Fuente: Elaboración propia

B.6. Tabla RequisitosPasosGraduacion

Esta tabla tiene por objeto guardar la información sobre los requisitos que corresponden a cada paso del proceso de graduación, de los cuales es necesario tener conocimiento para poder verificar su cumplimiento antes de dar paso a que un maestrante cambie de un estado a otro dentro de su proceso de graduación. Esta tabla es utilizada también por la aplicación paralela que se desarrolla en [1]. Sus campos son los que se detalla a continuación.

Tabla 33: Campos de la tabla "RequisitosPasosGraduacion"

Campo	*Tipo*	*Descripción*
requisitosPasosGraduacionId	Entero	Identificador del registro
paso	Entero	Indica el número de paso al que pertenece el requisito
requisito	Varchar(MAX)	Descripción del requisito
orden	Entero	Establece el orden en el que aparece el requisito dentro de un listado de requisitos

Fuente: Elaboración propia

En este caso cabe mencionar que los pasos establecidos para el proceso de graduación no se guardan en tabla alguna en ninguna de las dos bases de datos que utilizan ambas aplicaciones en cuestión. Esto debido a que son pasos ya definidos que guardan un orden establecido que no se prevé que pueda cambiar en el futuro. Estos pasos fueron definidos conjuntamente con la Ing. Gabriela De la

Torre ya que afectan directamente a ambas aplicaciones y bases de datos. A continuación, se procede a describir dichos pasos.

Tabla 34: Pasos establecidos para el proceso de graduación

Orden	*Descripción*
1	Ingreso de solicitud de aprobación de tema
2	Registro de aprobación de tema
3	Ingreso de solicitud de designación de docente informante
4	Registro de designación de docente informante
5	Ingreso de solicitud de aprobación del plan
6	Registro de aprobación del plan
7	Ingreso de solicitud de designación de docente director
8	Registro de designación de docente director
9	Registro de matrícula para desarrollo del proyecto
10	Ingreso de solicitud de designación de docentes revisores
11	Registro de designación de docentes revisores
12	Registro de aprobación de docentes revisores
13	Ingreso de solicitud de designación de tribunal calificador
14	Registro de designación de tribunal calificador
15	Registro de revisión en biblioteca para empastado
16	Ingreso de solicitud de designación de fecha y hora de defensa
17	Registro de designación de fecha y hora de defensa
18	Registro de graduación del maestrante
19	Registro de defensa fallida
20	Ingreso de solicitud de nueva fecha y hora de defensa
21	Baja de tema de proyecto de investigación y desarrollo

Fuente: Desarrollo de una Aplicación Web para la Administración de Procesos de Graduación de una Unidad de Postgrados ([1])

Apéndice C

Descripción de webservices

En el presente apéndice se procede a describir los webservices habilitados por el personal del departamento de informática de la PUCESA, en especial por el Ing. Mg. José Enríquez, como método para compartir la información ya existente en las bases de datos institucionales, facilitando y apoyando así el desarrollo de la presente aplicación.

Se debe mencionar que dicho personal ha habilitado una serie de webservices que pueden ser de uso común para la presente aplicación así como para la que se está desarrollando en paralelo en [1], o bien pueden servir únicamente para alguna de las dos. En todo caso, en este apéndice se procede a describir aquellos webservices que resultan de mayor o menor utilidad para la presente aplicación.

C.1. Webservice GetCarrerasPosgrados

Este webservice devuelve un arreglo con objetos de tipo “Carreras” (definido por el programador del webservice), que lista todas las carreras de posgrados con las que cuenta actualmente la PUCESA. Contiene la información que se detalla a continuación.

Tabla 35: Campos del webservice “GetCarrerasPosgrados”

Campo	*Tipo*	*Descripción*
carreraId	Entero	Identificador de la carrera
descripcion	Cadena	Descripción de la carrera
siglas	Cadena	Siglas con las que se identifica a la carrera
escuelaId	Entero	Identificador de la escuela a la que pertenece la carrera
tipoCarrera	Cadena	En todos los casos devuelve la cadena "PO", que denota que se trata de una carrera de posgrado
activa	Booleano	Indica si la carrera se considera activa dentro de la base de datos institucional

Fuente: Elaboración propia

C.2. Webservice GetDatosDocentes

Como su nombre lo indica, este webservice devuelve información sobre los docentes de posgrados registrados en la base de datos institucional. Devuelve un arreglo de objetos de tipo "Docentes", los cuales a su vez incluyen la información que se detalla a continuación.

Tabla 36: Campos del webservice "GetDatosDocentes"

Campo	*Tipo*	*Descripción*
docenteId	Entero	Identificador del docente
identificacion	Cadena	Número de cédula o pasaporte del docente
nombres	Cadena	Nombres del docente
apellidos	Cadena	Apellidos del docente
activo	Booleano	Especifica si el docente está registrado como activo o no

Fuente: Elaboración propia

C.3. Webservice GetDatosEstudiantes

Este webservice solicita un parámetro que es el identificador (Id) del estudiante del que se desea consultar los datos y devuelve un objeto de tipo Estudiantes que contiene la siguiente información acerca del estudiante en cuestión.

Tabla 37: Campos del webservice "GetDatosEstudiantes"

Campo	*Tipo*	*Descripción*
estudianteId	Entero	Identificador del estudiante
identificacion	Cadena	Número de cédula o pasaporte del estudiante
tipoIdentificacion	Cadena	Devuelve un carácter que indica si el número de identificación del estudiante corresponde a una cédula, pasaporte u otro tipo de documento
nombres	Cadena	Nombres del estudiante
apellidos	Cadena	Apellidos del estudiante
genero	Cadena	Devuelve M si el estudiante es hombre o F si es mujer
estadoCivil	Cadena	Devuelve un carácter que identifica el estado civil del estudiante

fechaNacimiento	Fecha/Hora	Fecha de nacimiento registrada para el estudiante
direccion	Cadena	Dirección del maestrante
telefono	Cadena	Teléfono convencional del maestrante
celular	Cadena	Teléfono celular del maestrante
fechaCreacion	Fecha/Hora	Fecha en la que el estudiante fue dado de alta en el sistema Academics
beneficiarioSeguro	Cadena	Devuelve un carácter que indica si el estudiante es beneficiario del seguro institucional o no
nacionalidadId	Entero	Identificador de la nacionalidad del estudiante
colegioId	Entero	Identificador del colegio del estudiante
tituloId	Entero	Identificador del título del estudiante
parroquiaId	Entero	Identificador de la parroquia del estudiante
etniaId	Entero	Identificador de la etnia del estudiante
creditoIECE	Booleano	Indica si el estudiante es beneficiario de crédito otorgado por el IECE
emailInstitucional	Cadena	Devuelve la dirección de e-mail institucional para el estudiante, en caso de existir
paraleloId	Entero	Identificador del paralelo del estudiante

Fuente: Elaboración propia

C.4. Webservice GetEstudiantesdePosgrado

Este webservice no requiere el envío de ningún parámetro y devuelve una tabla de tipo *DataTable* que contiene un listado de todos los estudiantes de posgrado registrados en las bases de datos institucionales. Devuelve los campos descritos a continuación.

Tabla 38: Campos del webservice "GetEstudiantesdePosgrado"

Campo	***Tipo***	***Descripción***
carreraId	Entero	Identificador de la carrera a la que pertenece el estudiante
estudianteId	Entero	Identificador del estudiante
cedula	Cadena	Número de cédula o pasaporte del estudiante
estudiante	Cadena	Apellidos y nombres del estudiante
matricula	Cadena	Número de matrícula del estudiante

Fuente: Elaboración propia

Apéndice D

Ejemplos de código fuente

En el presente apéndice se incluyen porciones relevantes del código fuente de la aplicación web desarrollada como herramienta de apoyo para el personal de la OIP, en sus diferentes módulos. Dichas porciones permiten apreciar aspectos como la legibilidad y mantenibilidad del código. Se incluyen tanto archivos ASPX, donde se puede ver el código en este lenguaje de lado del servidor, como archivos en C# donde se puede apreciar la lógica en sí de la programación, y un archivos en formato XML denominado Web.config, el miso que determina a través de muchos parámetros el comportamiento general del sitio web que se construye.

D.1. Archivo "Login.aspx"

Este archivo define la página web de inicio de la aplicación, donde sus posibles usuarios pueden proveer de sus credenciales para acceder a la misma.

Código:

```
<%@ Page Title="" Language="C#" MasterPageFile="~/Site.Master"
AutoEventWireup="true" CodeBehind="Login.aspx.cs"
Inherits="AtencionUsuario.Ui.Web.Login" %>
<%@ Register assembly="obout_Interface" namespace="Obout.Interface"
tagprefix="cc1" %>
<asp:Content ID="Content1" ContentPlaceHolderID="HeadContent" runat="server">
    <style type="text/css">
        .style2
        {
            width: 50%;
        }
        .style4
        {
            width: 111px;
            height: 40px;
        }
        .style5
        {
            height: 40px;
        }
        .style6
```

```
        {
            text-align: center;
        }
    </style>
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="MainContent" runat="server">
    <table align="center" class="style2" cellpadding="10px">
        <tr>
            <td colspan="2" class="style5">
                <h2 class="style6">Bienvenido al Módulo de Atención al Usuario
                de Posgrados de la PUCESA</h2>
            </td>
        </tr>
        <tr>
            <td colspan="2" class="style5">
                <h3 class="style6"><strong>Por favor, Ingrese sus
         datos:</strong></h3>
            </td>
        </tr>
        <tr>
            <td class="style4">
                <h3>Usuario:</h3>
            </td>
            <td>
                <cc1:OboutTextBox ID="tbUsuario"
          runat="server"></cc1:OboutTextBox>
            </td>
        </tr>
        <tr>
            <td class="style4">
                <h3>Password:</h3></td>
            <td>
                <cc1:OboutTextBox ID="tbPassword" runat="server"
          TextMode="Password"></cc1:OboutTextBox>
            </td>
        </tr>
        <tr>
            <td class="style4">
                 </td>
            <td>
                <cc1:OboutButton ID="OboutButton1" runat="server"
          Text="Ingresar" onclick="OboutButton1_Click">
                </cc1:OboutButton>
            </td>
        </tr>
        <tr>
            <td align="center" colspan="2">
                <asp:Label ID="lblError" runat="server"
                    Text="Datos Incorrectos, por favor intente nuevamente..."
              Font-Bold="True" Font-Overline="False" ForeColor="Red"
              Visible="False"></asp:Label>
            </td>
        </tr>
    </table>
</asp:Content>
```

D.2. Archivo “Login.aspx.cs”

Este archivo corresponde al *CodeBehind* de la página Login.aspx. Está escrito en lenguaje C# y se encarga de manejar los eventos generados por los diferentes componentes y controles de la página de acceso de los usuarios de la aplicación.

Código:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using AtencionUsuario.Ui.Web.wspucesa; //Para usar webservices
using AtencionUsuario.Logic;
using System.Web.Security;
using System.Data;
using AtencionUsuario.Model;

namespace AtencionUsuario.Ui.Web
{
    public partial class Login : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {

        }

        protected void OboutButton1_Click(object sender, EventArgs e)
        {
            String usuario = "";
            String password = "";
            String resultado = "";

            //Recuperar el único registro de la tabla de parámetros para
             //verificar si el usuario que se
            //está logueando es la secretaria de posgrados
            ParametrosPosgrados parametros;
            using (var db = new AtencionUsuarioEntities())
                parametros = db.ParametrosPosgrados.FirstOrDefault();

            //Si se está logueando la secretaria de posgrados
            if (tbUsuario.Text.Trim() == parametros.usuarioSecreatariaDIP)
            {
                //Si el campo "passwordSecretariaDIP" se encuentra vacío,
                //redirigir a la página para crear uno
                if (String.IsNullOrEmpty(parametros.passwordSecretariaDIP))
                    Response.Redirect("NewPassword.aspx");
                //Si la contraseña ingresada no coincide con la almacenada
                //(encriptada), desplegar error
                else if (General.Encriptar(tbPassword.Text, "PUceSA201-7") !=
```

```
    parametros.passwordSecretariaDIP)
    {
        lblError.Visible = true;
        return;
    }
    //Finalmente, si ambos datos están correctos, proceder
    else
    {
        usuario = "0";
        resultado = "DIRECTOR";
    }
}

//Conexión a webservices
wsSeguimientoTesisSoapClient ws = new
 wsSeguimientoTesisSoapClient();

//Si no fue ninguno la secretaria de posgrados, continuar con el
 //proceso normal usando webservices
if (String.IsNullOrEmpty(usuario) ||
 String.IsNullOrEmpty(resultado))
{
    usuario = tbUsuario.Text.Trim();
    password = tbPassword.Text;
    resultado = ws.EspecificarPerfilUsuario(usuario, password);
}

//Proceso para autenticar usuario mediante cookie
if (resultado != "USUARIO NO ENCONTRADO")
{
    //Creación del ticket
    FormsAuthenticationTicket ticket = new
    FormsAuthenticationTicket(
        1,
        usuario,
        DateTime.Now,
        DateTime.Now.AddMinutes(2880),
        true, //Persistencia de cookie
        resultado,
        FormsAuthentication.FormsCookiePath);

    //Encriptación del ticket
    String hash = FormsAuthentication.Encrypt(ticket);

    //Creación de la cookie
    HttpCookie cookie = new
       HttpCookie(FormsAuthentication.FormsCookieName, hash);

    //Definir la duración de la cookie
    if (ticket.IsPersistent) cookie.Expires = ticket.Expiration;

    //Adición de la cookie
    Response.Cookies.Add(cookie);

    //Creación variable de sesión para contener datos de usuario
```

```
                //usando clase UsuarioGraduacionAtencion
                Session.Timeout = 2880; //Duración de las variables de sesión:
                 //1 hora
                //Variables auxiliares
                int id = 0;
                String apellidos = "";
                String nombres = "";
                String cedula = "";
                String email = "";
                UInt16 tipo = 9;
                String _direccion = ""; //Para saber a dónde redirigir por
                //defecto
                switch (resultado)
                {
                    case "ESTUDIANTE": //Para estudiantes
                        //Recuperar registro del estudiante desde Webservice
                        DataRow[] dr =
ws.GetEstudiantesdePosgrado().Select("Cedula = '" + usuario + "'");
                        //Recuperar el Id de Estudiante, ya que no necesita la
                        //cédula sino el Id
                        int estudianteId = (int)dr[0]["EstudianteId"];
                        Estudiantes estudiante =
                          ws.GetDatosEstudiantes(estudianteId);
                        //Recolectar datos del estudiante para crear objeto
                        //tipo UsuarioGraduacionAtencion
                        id = estudiante.EstudianteId;
                        apellidos = estudiante.Apellidos.Trim();
                        nombres = estudiante.Nombres.Trim();
                        cedula = estudiante.Identificacion.Trim();
                        email = estudiante.EmailInstitucional != null ?
                          estudiante.EmailInstitucional.Trim() : "";
                        tipo = 0; //Estudiante
                        _direccion = "~/Estudiante/Default.aspx";
                        break;
                    case "PROFESOR": //Para profesores. Verificar si la cadena
                    //es "PROFESOR"
                        Docentes docente = ws.GetDatosDocentes().Where(p =>
                          p.Identificacion == usuario).FirstOrDefault();
                        id = docente.DocenteId;
                        apellidos = docente.Apellidos;
                        nombres = docente.Nombres;
                        cedula = docente.Identificacion;
                        email = "";
                        tipo = 1; //Profesor
                        _direccion = "~/Maestro/Default.aspx";
                        break;
                    case "DIRECTOR": //Para el director de la OIP. Verificar
                    //si la cadena es "DIRECTOR"
                        id = 0;
                        apellidos = parametros.nombreSecretariaDIP;
                        nombres = "";
                        cedula = "";
                        email = "";
                        tipo = 2;
                        _direccion = "~/Director/Default.aspx";
```

```
                    break;
                }
                //Creación de objeto tipo UsuarioGraduacionAtencion con los
                //datos recolectados
                UsuarioGraduacionAtencion _usuario = new
      UsuarioGraduacionAtencion(id, apellidos, nombres, cedula, email, tipo);
                //Guardado del objeto en variable de sesión
                Session["Usuario"] = _usuario;

                //Redirección luego de la autenticación exitosa
                String direccion = Request.QueryString["ReturnUrl"];
                if (direccion == null) direccion = _direccion;
                Response.Redirect(direccion);
            }
            else //Si el login falló, desplegar el mensaje de error
            {
                lblError.Visible = true;
            }
            //Cierre Webservices
            ws.Close();

        }
    }
}
```

D.3. Archivo ListaActivos.aspx

El siguiente archivo contiene el código de la página que muestra la lista de maestrantes activos registrados por el personal del a OIP, el mismo que se encuentra en ASP.NET.

Código:

```
<%@ Page Title="" Language="C#" MasterPageFile="~/Site.Master"
AutoEventWireup="true" CodeBehind="ListaActivos.aspx.cs"
Inherits="AtencionUsuario.Ui.Web.Director.ListaActivos" %>

<%@ Register Assembly="AjaxControlToolkit" Namespace="AjaxControlToolkit"
TagPrefix="ajaxToolkit" %>

<%@ Register assembly="obout_Interface" namespace="Obout.Interface"
tagprefix="cc1" %>
<asp:Content ID="Content1" ContentPlaceHolderID="HeadContent" runat="server">
    <style type="text/css">
        .cabeceraHeader
        {
             color: #000000;
             font-family: Tahoma;
             font-size: 20px;
        }
```

```
    </style>
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="MainContent" runat="server">
    <div id="Pagina">
        <div id="Cabecera">
            <h2>LISTA DE MAESTRANTES ACTIVOS</h2>
            <asp:Label ID="lblError" runat="server"
                Font-Bold="True" Font-Size="Large" ForeColor="Red"
                   style="text-align: left"
                Visible="False"></asp:Label>
                <br /><br />
        </div>
        <div id="ContenidoSolicitudAprobacion">
            <asp:ScriptManager ID="ScriptManager1" runat="server">
            </asp:ScriptManager>
            <strong>Seleccione una carrera:</strong>
            <ajaxToolkit:ComboBox ID="cbListaCarreras" runat="server"
                AutoCompleteMode="SuggestAppend" AutoPostBack="True"
                   BorderStyle="Solid"
                BorderWidth="1px" Width="500px"
                DropDownStyle="DropDownList"
                onselectedindexchanged="cbListaCarreras_SelectedIndexChanged">
            </ajaxToolkit:ComboBox><br /><br />
            <asp:GridView ID="GridView1" runat="server"
                   AutoGenerateColumns="False"
                BackColor="#DFEFFF" BorderStyle="Solid" BorderWidth="3px"
                   CellPadding="3"
                Width="2380px">
                <AlternatingRowStyle BackColor="White" />
                <Columns>
                    <asp:HyperLinkField DataNavigateUrlFields="estudianteId"
      DataNavigateUrlFormatString="DatosGeneralesMaestrante.aspx?id={0}"
                        DataTextField="nombre" HeaderText="NOMBRE ESTUDIANTE">
                    <HeaderStyle BackColor="#99CCFF" Width="220px" Font-
                          Bold="True"
                        Font-Italic="True" />
                    <ItemStyle Font-Bold="False" Font-Italic="False" />
                    </asp:HyperLinkField>
                    <asp:BoundField DataField="matricula" HeaderText="No.">
                    <HeaderStyle Width="100px" BackColor="#99CCFF" Font-
                          Bold="True"
                        Font-Italic="True" />
                    </asp:BoundField>
                    <asp:BoundField DataField="fechaEgresamiento"
                        HeaderText="FECHA DE EGRESAMIENTO">
                    <HeaderStyle Width="130px" BackColor="#99CCFF" Font-
                          Bold="True"
                        Font-Italic="True" />
                    </asp:BoundField>
                    <asp:BoundField DataField="tema" HeaderText="TEMA DEL
                          PROYECTO">
                    <HeaderStyle Width="220px" BackColor="#99CCFF" Font-
                          Bold="True"
                        Font-Italic="True" />
                    </asp:BoundField>
```

```
<asp:BoundField DataField="informante" HeaderText="DOCENTE
      INFORMANTE">
<HeaderStyle Width="220px" BackColor="#99CCFF" Font-
      Bold="True"
    Font-Italic="True" />
</asp:BoundField>
<asp:BoundField DataField="aprobacionPlan"
      HeaderText="APROBACIÓN DEL PLAN">
<HeaderStyle Width="120px" BackColor="#99CCFF" Font-
      Bold="True"
    Font-Italic="True" />
</asp:BoundField>
<asp:BoundField DataField="directorProyecto"
      HeaderText="DIRECTOR DEL PROYECTO">
<HeaderStyle Width="220px" BackColor="#99CCFF" Font-
      Bold="True"
    Font-Italic="True" />
</asp:BoundField>
<asp:BoundField DataField="revisor1" HeaderText="REVISOR
      1">
<HeaderStyle Width="220px" BackColor="#99CCFF" Font-
      Bold="True"
    Font-Italic="True" />
</asp:BoundField>
<asp:BoundField DataField="revisor2" HeaderText="REVISOR
      2">
<HeaderStyle Width="220px" BackColor="#99CCFF" Font-
      Bold="True"
    Font-Italic="True" />
</asp:BoundField>
<asp:BoundField DataField="estadoProyecto"
      HeaderText="ESTADO DEL PROYECTO">
<HeaderStyle Width="175px" BackColor="#99CCFF" Font-
      Bold="True"
    Font-Italic="True" />
</asp:BoundField>
<asp:BoundField DataField="porcentajeAvance" HeaderText="%
      DE AVANCE">
<HeaderStyle Width="100px" BackColor="#99CCFF" Font-
      Bold="True"
    Font-Italic="True" />
</asp:BoundField>
<asp:BoundField DataField="pagoProyecto" HeaderText="PAGO
      DEL PROYECTO">
<HeaderStyle Width="105px" BackColor="#99CCFF" Font-
      Bold="True"
    Font-Italic="True" />
</asp:BoundField>
<asp:BoundField DataField="prorroga1" HeaderText="PRORROGA
      I">
<HeaderStyle Width="110px" BackColor="#99CCFF" Font-
      Bold="True"
    Font-Italic="True" />
</asp:BoundField>
<asp:BoundField DataField="prorroga2" HeaderText="PRORROGA
```

```
                    II">
                <HeaderStyle Width="110px" BackColor="#99CCFF" Font-
                    Bold="True"
                Font-Italic="True" />
                </asp:BoundField>
                <asp:BoundField DataField="id" HeaderText="CARRERAID"
                    Visible="False">
                <HeaderStyle Width="110px" BackColor="#99CCFF" Font-
                    Bold="True"
                Font-Italic="True" />
                </asp:BoundField>
                <asp:BoundField DataField="estudianteId"
                    HeaderText="ESTUDIANTEID"
                Visible="False" />
            </Columns>
        </asp:GridView>
    </div>
    <div id="BarraInferiorBotones">
        <cc1:OboutButton ID="btnRegresar" runat="server"
         onclick="btnRegresar_Click"
            Text="Regresar">
        </cc1:OboutButton>
    </div>
</div>
</asp:Content>
```

D.4. Archivo ListaActivos.aspx.cs

Este archive contiene el código de *CodeBehind* del archivo ListaActivos.aspx, que se detalla en la sección anterior. La funcionalidad del siguiente código es responder a los eventos generados por los diferentes controles y elementos de la página. Código en lenguaje C#.

Código:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using AtencionUsuario.Logic;
using AtencionUsuario.Ui.Web.wspucesa; //Webservices
using System.Data; //Para usar tipo DataTable
using AtencionUsuario.Model; //Bases de datos

namespace AtencionUsuario.Ui.Web.Director
{
    public partial class ListaActivos : System.Web.UI.Page
```

```
{
    //Definición de la variable privada "usuario" para recuperar la
     //información de sesión
    private UsuarioGraduacionAtencion usuario;

    protected void Page_Load(object sender, EventArgs e)
    {
        //Si se dio click en "Cerrar Sesión", hacerlo antes de nada!
        General.VerificarCerrarSesion(this.Page);

        //Verificar existencia de variable de sesión y recuperar la misma
        if (Session["usuario"] != null)
            usuario = (UsuarioGraduacionAtencion)Session["usuario"];
        else
            //Response.Redirect("~/Login.aspx", true);
            General.CerrarSesion(this.Page);

        //Click en el botón "Regresar"
        if (!String.IsNullOrEmpty(Request.Form["__EVENTTARGET"]))
        {
            String controlFuente = Request.Form["__EVENTTARGET"];
            if (controlFuente.Substring(controlFuente.Length - 16) ==
               "btnRegresar")
                Response.Redirect("Default.aspx");
        }

        //Creación del DataTable para contener la información a listar
        DataTable dt = new DataTable();
        dt.Columns.AddRange(new DataColumn[9] {
                new DataColumn("nombre", typeof(String)),
                new DataColumn("matricula", typeof(String)),
                new DataColumn("fechaEgresamiento", typeof(String)),
                new DataColumn("tema", typeof(String)),
                new DataColumn("informante", typeof(String)),
                new DataColumn("aprobacionPlan", typeof(String)),
                new DataColumn("directorProyecto", typeof(String)),
                new DataColumn("revisor1", typeof(String)),
                new DataColumn("revisor2", typeof(String)),
                new DataColumn("estadoProyecto", typeof(String)),
                new DataColumn("porcentajeAvance", typeof(String)),
                new DataColumn("pagoProyecto", typeof(String)),
                new DataColumn("prorroga1", typeof(String)),
                new DataColumn("prorroga2", typeof(String)),
                new DataColumn("id", typeof(String)),
                new DataColumn("estudianteId", typeof(int))});

        if (!Page.IsPostBack) //Primera carga de la página
        {
            //Conexión a Webservices
            wsSeguimientoTesisSoapClient ws = new
               wsSeguimientoTesisSoapClient();

            //Recuperar lista de todos los maestrantes desde variable de
            //sesión (si existe) o desde webservices
            DataTable estudiantes;
```

```
if (Session["estudiantes"] != null)
    estudiantes = (DataTable)Session["estudiantes"];
else
{
    estudiantes = ws.GetEstudiantesdePosgrado();
    Session["estudiantes"] = estudiantes;
}

//Recuperar datos de los docentes
List<Docentes> docentes;
if (Session["docentes"] != null)
    docentes = (List<Docentes>)Session["docentes"];
else
{
    docentes = ws.GetDatosDocentes().ToList();
    Session["docentes"] = docentes;
}

//Recuperar lista de maestrantes que están o han estado en
//cuarto nivel (egresados o por egresar)
//y convertir los campos de fecha de String a DateTime
DataTable estudiantesCuarto;
if (Session["estudiantesCuarto"] != null)
   estudiantesCuarto =
         (DataTable)Session["estudiantesCuarto"];
else
{
    estudiantesCuarto = ws.GetEstudiantesCuartosNiveles();
    Session["estudiantesCuarto"] = estudiantesCuarto;
}

//Poblar combobox de carreras de posgrado
List<Carreras> carreras = ws.GetCarrerasPosgrados()
   .OrderBy(p => p.Descripcion).ToList();
cbListaCarreras.Items.Add(
   new System.Web.UI.WebControls.ListItem(
   "Todas las carreras", "-1"));
foreach (Carreras carrera in carreras)
    cbListaCarreras.Items.Add(
         new System.Web.UI.WebControls.ListItem(
         carrera.Descripcion, carrera.CarreraId.ToString()));
cbListaCarreras.SelectedValue = "-1";

//Recorrer todos los temas de la base de datos buscando los
//activos
using (var db = new GraduacionEntities())
{
    foreach (TemasdeTesis tema in db.TemasdeTesis.Where(p =>
         p.estudianteId != null))
    {
        Tuple<int, TemasdeTesis, PlanesdeTesis,
          TesisPosgrados> estadoGeneral =
          EstudiantesManager.GetEstadoGeneralbyEstudianteId((i
          nt)tema.estudianteId);
        if (estadoGeneral.Item1 != 19) //Si no es maestrante
```

```
  //GRADUADO (los graduados no son activos)
{
    DataRow[] estudiante = estudiantes.Select(
  "EstudianteId = " + tema.estudianteId.ToString());
    String nombre = General.ConvertToTitleCase(
        (String)estudiante[0]["Estudiante"]);
    String matricula =
        (String)estudiante[0]["Matricula"];
    //Recuperar la fecha de egresamiento, si existe,
    //de lo contrario mensaje "sin egresar"
    //y recuperar información de prórrogas
    DataRow[] cuartosNiveles = estudiantesCuarto
        .Select("EstudianteId = " + tema.estudianteId
        .ToString(), "FechaFin asc");
    String fechaEgresamiento = "Sin egresar";
    String prorroga1 = "";
    String prorroga2 = "";
    if (cuartosNiveles.Count() > 0)
    {
        if ((DateTime)cuartosNiveles[0]["FechaFin"] <=
               DateTime.Now)
        {
            DateTime fechaFin = (DateTime)
               cuartosNiveles[0]["FechaFin"];
            fechaEgresamiento = String.Format(
               "{0:dd/MM/yyyy}", fechaFin);
        }
        if (cuartosNiveles.Count() >= 2)
               prorroga1 = "Sí";
        if (cuartosNiveles.Count() >= 3)
               prorroga2 = "Sí";
    }

    //En la “tuple” de estadoGeneral ya vienen el plan
    //y el proyecto, si existen
    String docenteInformante = "Sin Informante";
    String aprobacionPlan = "";
    String director = "Sin Director";
    String revisor1 = "";
    String revisor2 = "";
    String pagoProyecto = "No";
    int porcentajeAvance = 0;
    String estadoProyecto = General.
        GetDescripcionEstadoGeneral(estadoGeneral.Item
        1, 0);
    if (estadoGeneral.Item3 != null)
    {
        //Docente informante designado
        if (estadoGeneral.Item3.estado >= 3)
            //Docente informante EXISTE
            if (estadoGeneral.Item3.docenteLectorId !=
               null)
            {
                Docentes informante = docentes.Where(q
               => q.DocenteId == estadoGeneral.Item3.
```

```
            docenteLectorId).FirstOrDefault();
              docenteInformante = General.
                   ConvertToTitleCase(informante
                   .Apellidos.Trim() + " " +
                   informante.Nombres.Trim());
          }
      //Plan aprobado
      if (estadoGeneral.Item3.estado == 5)
          aprobacionPlan = "Aprobado";

      if (estadoGeneral.Item4 != null)
      {
          //Director designado
          if (estadoGeneral.Item4.estado >= 20)
          {
              Docentes _director = docentes.Where(q
                   => q.DocenteId == estadoGeneral
                   .Item4.directorId)
                   .FirstOrDefault();
              director = General.ConvertToTitleCase
                   (_director.Apellidos.Trim() + " "
                   + _director.Nombres.Trim());
          }
          //Maestrante matriculado
          if (estadoGeneral.Item4.estado >= 30)
          {
              pagoProyecto = "Sí";
          }
          //Revisores designados
          if (estadoGeneral.Item4.estado >= 50)
          {
              if (estadoGeneral.Item4.
                   docenteRevisorId1 != null &&
                   estadoGeneral.Item4
                   .docenteRevisorId2 != null)
              {
                  Docentes _revisor1 = docentes.
                   Where(q => q.DocenteId ==
                   estadoGeneral.Item4.
                   docenteRevisorId1).
                   FirstOrDefault();
                  Docentes _revisor2 = docentes.
                   Where(q => q.DocenteId ==
                   estadoGeneral.Item4.
                   docenteRevisorId2).
                   FirstOrDefault();
                  revisor1 = General.
                   ConvertToTitleCase(_revisor1.
                   Apellidos.Trim() + " " +
                   _revisor1.Nombres.Trim());
                  revisor2 = General.
                   ConvertToTitleCase(_revisor2.
                   Apellidos.Trim() + " " +
                   _revisor2.Nombres.Trim());
              }
```

```
                        else
                        {
                            revisor1 = "-";
                            revisor2 = "-";
                        }
                    }

                    //Solo si el proyecto está en estado
                    //general 9 (matriculado para desarrollo)
                    //Ver si existe información sobre
                    //porcentaje de avance avalado por
                    //director
                    if (estadoGeneral.Item1 == 9)
                    {
                        PorcentajesAvaladosTesis pat =
                             PorcentajesAvaladosTesisManager.
                             GetPorcentajeAvaladobyTesisId
                             (estadoGeneral.Item4.
                             tesisPosgradosId);
                        if (pat != null) porcentajeAvance =
                             (int)pat.porcentaje;
                    }
                    else if (estadoGeneral.Item1 > 9)
                    //solicitud revisores, es porque ya
                    //finalizó el desarrollo
                        porcentajeAvance = 100;
                }
            }

            //Finalmente, agregar toda la información
            //recolectada al DataTable
            dt.Rows.Add(nombre, matricula, fechaEgresamiento,
                tema.tema.Trim(), docenteInformante,
                aprobacionPlan, director, revisor1, revisor2,
                estadoProyecto, porcentajeAvance.ToString() +
                "%", pagoProyecto, prorroga1, prorroga2,
                tema.carreraId.ToString(), tema.estudianteId);
        }
    }

    //Ordenar el DataTable por apellidos de los maestrantes
    DataView dv = dt.DefaultView;
    dv.Sort = "nombre asc"; //en vez de "asc" puede ser "desc"
    dt = dv.ToTable();
    //Guardar el DataTable como variable de sesión
    Session["dt"] = dt;

    //Finalmente, poblar el Grid con los datos del DataTable,
    //si existen, sino desplegar error
    if (dt.Rows.Count > 0)
    {
        lblError.Visible = false;
        GridView1.DataSource = dt;
        GridView1.DataBind();
        GridView1.Visible = true;
```

```
                    }
                    else
                    {
                        lblError.Text = "No existen maestrantes activos para
                          la carrera seleccionada";
                        lblError.Visible = true;
                        GridView1.Visible = false;
                    }
                }

                //Cerrar los Webservices
                ws.Close();
            }
             //Siguientes cargas de la página, presumiblemente desencadenado
             //al seleccionar otra carrera
            else
            {
                dt = (DataTable)Session["dt"];
                DataView dv = dt.DefaultView;
                //Si se seleccionó una carrera específica
                if (cbListaCarreras.SelectedValue != "-1")
                {
                    dv.RowFilter = "id = '" + cbListaCarreras.SelectedValue +
                          "'";
                }
                else
                {
                    dv.RowFilter = "";
                }

                if (dv.Count > 0)
                {
                    lblError.Visible = false;
                    dt = dv.ToTable();
                    GridView1.DataSource = dt;
                    GridView1.DataBind();
                    GridView1.Visible = true;
                }
                else
                {
                    lblError.Text = "No existen maestrantes activos para la
                          carrera seleccionada";
                    lblError.Visible = true;
                    GridView1.Visible = false;
                }
            }
        }

        protected void btnRegresar_Click(object sender, EventArgs e)
        {
            Response.Redirect("Default.aspx");
        }

    }
}
```

D.5. Archivo ActualizarCorreoDocente.aspx

Este archivo representa una página web que le permite al personal de la OIP la administración de direcciones de correo electrónico de los docentes de postgrados de la PUCESA. El código se encuentra en lenguaje ASP.NET.

Código:

```
<%@ Page Title="" Language="C#" MasterPageFile="~/Site.Master"
AutoEventWireup="true" CodeBehind="ActualizarCorreoDocente.aspx.cs"
Inherits="AtencionUsuario.Ui.Web.Director.ActualizarCorreoDocente" %>

<%@ Register Assembly="obout_Calendar2_Net" Namespace="OboutInc.Calendar2"
TagPrefix="obout" %>

<%@ Register Assembly="obout_Grid_NET" Namespace="Obout.Grid" TagPrefix="cc2"
%>
<%@ Register assembly="obout_Interface" namespace="Obout.Interface"
tagprefix="cc1" %>

<%@ Register assembly="AjaxControlToolkit" namespace="AjaxControlToolkit"
tagprefix="ajaxToolkit" %>

<asp:Content ID="Content1" ContentPlaceHolderID="HeadContent" runat="server">
    <style type="text/css">
        .style1
        {
            width: 23%;
        }
        .style2
        {
            width: 100%;
        }
        </style>
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="MainContent" runat="server">
    <asp:ScriptManager ID="ScriptManager1" runat="server">
    </asp:ScriptManager>
    <div id="Pagina">
        <div id="Cabecera">
            <h2>
                REGISTRO / ACTUALIZACIÓN DE E-MAILS DE DOCENTES</h2>
            <asp:Label ID="lblError" runat="server"
                Font-Bold="True" Font-Size="Large" ForeColor="Red"
                   style="text-align: left"
                Visible="False"></asp:Label>
        </div>
        <div id="ContenidoSolicitudAprobacion">
            <div style="text-align:center;">
                <div style="width:650px; margin: 0 auto; text-align:left;">
                    <table cellpadding="10" width="650px">
                        <tr id="trMaestrante" runat="server">
```

```
                        <td class="style1">
                            <b>Docente:</b></td>
                        <td class="style2">
                            <ajaxToolkit:ComboBox ID="cbListaDocentes"
                                   runat="server"
                                AutoCompleteMode="SuggestAppend"
                                BorderStyle="Solid" BorderWidth="1px"
                                Width="500px" AutoPostBack="True"
                                onselectedindexchanged=
                                "cbListaDocentes_SelectedIndexChanged">
                            </ajaxToolkit:ComboBox>
                        </td>
                    </tr>
                    <tr id="trEmail" runat="server" visible="false">
                        <td class="style1">
                            <b>E-mail:</b></td>
                        <td class="style2">
                            <cc1:OboutTextBox ID="tbEmail" runat="server"
                            Width="400px"></cc1:OboutTextBox>
                        </td>
                    </tr>
                </table>
            </div>
        </div>
    </div>
    <div id="BarraInferiorBotones">
        <cc1:OboutButton ID="btnActualizar" runat="server"
         Text="Actualizar E-mail" Style="top: 0px; left: 0px"
            OnClientClick="return confirm('¿Seguro que desea actualizar el
               e-mail para el maestrante seleccionado?');"
            Enabled="False" onclick="btnActualizar_Click" >
        </cc1:OboutButton>
             
        <cc1:OboutButton ID="btnCancelar" runat="server" Text="Regresar al
         menú principal"
            Style="top: 0px; left: 0px"
            OnClientClick="return confirm('¿Seguro que desea abandonar
               esta página?');"
            onclick="btnRegresar_Click" >
        </cc1:OboutButton>
    </div>
</div>
</asp:Content>
```

D.6. Archivo ActualizarCorreoDocente.aspx.cs

Corresponde al archivo de *CodeBehind* de ActualizarCorreoDocente.aspx. Contiene el código en C# que permite manejar los eventos generados por los controles y demás componentes de dicho archivo.

Código:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using AtencionUsuario.Logic;
using AtencionUsuario.Model;
using AtencionUsuario.Ui.Web.wspucesa; //Webservices
using System.Data; //Para usar tipo DataTable

namespace AtencionUsuario.Ui.Web.Director
{
    public partial class ActualizarCorreoDocente : System.Web.UI.Page
    {
        //Definición de la variable privada "usuario" para recuperar la
         //información de sesión
        private UsuarioGraduacionAtencion usuario;

        protected void Page_Load(object sender, EventArgs e)
        {
            //Si se dio click en "Cerrar Sesión", hacerlo antes de nada!
            General.VerificarCerrarSesion(this.Page);

            //Verificar existencia de variable de sesión y recuperar la misma
            if (Session["usuario"] != null)
                usuario = (UsuarioGraduacionAtencion)Session["usuario"];
            else
                //Response.Redirect("~/Login.aspx", true);
                General.CerrarSesion(this.Page);

            if (!Page.IsPostBack) //Primera carga de la página
            {
                //Conexión a Webservices y recuperación de lista de docentes
                wsSeguimientoTesisSoapClient ws = new
                   wsSeguimientoTesisSoapClient();
                List<Docentes> docentes = ws.GetDatosDocentes().OrderBy(p =>
                   p.Apellidos).ToList();

                //Poblar combobox de docentes
                cbListaDocentes.Items.Add(new System.Web.UI.WebControls
                   .ListItem("Seleccione un ítem...", "-1"));
                foreach (Docentes docente in docentes)
                    cbListaDocentes.Items.Add(new System.Web.UI.WebControls
                   .ListItem(docente.Apellidos.Trim() + " " +
                   docente.Nombres.Trim(), docente.DocenteId.ToString()));
                cbListaDocentes.SelectedValue = "-1";

                //Cerrar webservices
                ws.Close();
            }
            else //Siguientes cargas de la página
```

```
    {
    }
}

protected void cbListaDocentes_SelectedIndexChanged(object sender,
     EventArgs e)
{
    //Si es del caso, ocultar el mensaje de error
    if (lblError.Visible) lblError.Visible = false;

    if (cbListaDocentes.SelectedValue != "-1")
    {
        //Si se seleccionó un docente, tratar de recuperar su e-mail
        //registrado y desplegarlo
        EmailUsuario emailUsuario = EmailUsuarioManager.
           GetEmailUsuariobyUsuarioIdandTipo(int.Parse(cbListaDocentes
           .SelectedValue), 1);
        tbEmail.Text = emailUsuario == null ? "" :
           emailUsuario.email.Trim();
        trEmail.Visible = true;
        btnActualizar.Enabled = true;
    }
    else //Se seleccionó la opción "Escoja un ítem..." (o similar)
    {
        trEmail.Visible = false;
        btnActualizar.Enabled = false;
    }
}

protected void btnActualizar_Click(object sender, EventArgs e)
{
    if (String.IsNullOrEmpty(tbEmail.Text.Trim())) //Verificar ingreso
           de e-mail
    {
        lblError.Text = "Por favor, ingrese el e-mail del docente";
        lblError.Visible = true;
    }
    else if (!General.ComprobarFormatoEmail(tbEmail.Text.Trim()))
           //Verificar validez de e-mail
    {
        lblError.Text = "La dirección de e-mail ingresada no es
           válida";
        lblError.Visible = true;
    }
    else //Todo correcto, armar objeto y enviar a guardar el nuevo
           registro (managers)
    {
        EmailUsuario email = new EmailUsuario();
        email.usuarioId = int.Parse(cbListaDocentes.SelectedValue);
        email.tipoUsuario = 1; //Docente
        email.email = tbEmail.Text.Trim();
        if (EmailUsuarioManager.
           SaveorUpdateEmailUsuariobyObjetc(email) == -1)
        {
            lblError.Text = "Error al registrar el e-mail del
```

```
                        docente";
                    lblError.Visible = true;
                }
                else
                {
                    Response.Redirect("c_EmailActualizado.aspx");
                }
            }
        }

        protected void btnRegresar_Click(object sender, EventArgs e)
        {
            Response.Redirect("Default.aspx");
        }

    }
}
```

D.7. Archivo r_GraficoEstadistico.ashx

Este archivo corresponde a un controlador genérico, el mismo que permite enviar al proceso cliente (navegador web) contenido diferente a HTML estándar, como sucede con las página ASPX. Es un archivo en lenguaje C# que, en este caso, recibe un gráfico estadístico mediante variables de sesión y lo ubica en un documento PDF que es presentado al cliente. El archivo implementa una clase que a su vez hereda de las clases *IHttpHandler* y *IRequiresSessionState*, lo cual es de vital importancia para la función que se programa.

Código:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using iTextSharp.text;
using iTextSharp.text.pdf;
using System.Globalization;
using System.Web.SessionState;

namespace AtencionUsuario.Ui.Web.Director
{
    /// <summary>
    /// Exporta los gráficos generados estadísticamente a PDF
    /// </summary>
    public class r_GraficoEstadistico : IHttpHandler, IRequiresSessionState
    {

        public void ProcessRequest(HttpContext context)
```

```
        {
            //Inicialización del documento
            Document Doc = new Document(PageSize.A4.Rotate(), 10f, 10f, 10f,
             0f);
            PdfWriter.GetInstance(Doc, context.Response.OutputStream);
            Doc.Open();
            //Adición de la imagen de encabezado
            iTextSharp.text.Image logo = Image.GetInstance(context.Server
             .MapPath("~/Images/logonuevo1.jpg"));
            logo.Alignment = iTextSharp.text.Image.ALIGN_MIDDLE;
            logo.ScalePercent(40f);
            Doc.Add(logo);
            //Adición del párrafo con fecha y hora del resporte
            Paragraph p1 = new Paragraph();
            p1.Add(new Chunk("Fecha del reporte: ",
             FontFactory.GetFont("verdana", 12, Font.BOLD)));
            p1.Add(new Chunk(DateTime.Now.ToString("g", new CultureInfo("es-
             ES"))));
            p1.Alignment = Element.ALIGN_CENTER;
            Doc.Add(p1);
            //Recuperación de la imagen desde la variable de sesión y
             //destrucción de la misma
            iTextSharp.text.Image img = (iTextSharp.text.Image)context
             .Session["imagen"];
            context.Session.Remove("imagen");
            img.Alignment = iTextSharp.text.Image.ALIGN_MIDDLE;
            img.ScalePercent(75f);
            Doc.Add(img);
            //Fin del documento
            Doc.Close();

            //Respuesta
            context.Response.ContentType = "application/pdf";
            //La(s) siguiente(s) línea(s) (?) son para bajarlo como archivo
            //Response.AddHeader("content-disposition",
            //"attachment;filename=Chart.pdf");
            //Response.Cache.SetCacheability(HttpCacheability.NoCache);
            //context.Response.Write(Doc);
            context.Response.End();
        }

        public bool IsReusable
        {
            get
            {
                return false;
            }
        }
    }
}
```

D.8. Archivo Web.Config

Este es el archivo de configuración en todos los proyectos de desarrollo de aplicaciones web en ASP.NET. Proporciona parámetros de carácter general que determinan todos los aspectos del funcionamiento de una aplicación ASP.NET, siendo un archivo en formato XML.

Código:

```
<?xml version="1.0"?>
<!--
  Para obtener más información sobre cómo configurar la aplicación de ASP.NET,
           visite
  http://go.microsoft.com/fwlink/?LinkId=169433
  -->
<configuration>
  <appSettings>
    <add key="CrystalImageCleaner-AutoStart" value="true" />
    <add key="CrystalImageCleaner-Sleep" value="60000" />
    <add key="CrystalImageCleaner-Age" value="120000" />
    <add key="ChartImageHandler" value="storage=file;timeout=20;" />
  </appSettings>
  <connectionStrings>
    <add name="ApplicationServices" connectionString="data
           source=.\SQLEXPRESS;Integrated
           Security=SSPI;AttachDBFilename=|DataDirectory|\aspnetdb.mdf;User
           Instance=true" providerName="System.Data.SqlClient"/>
    <add name="GraduacionEntities"
           connectionString="metadata=res://*/Graduacion.csdl|res://*/Graduac
           ion.ssdl|res://*/Graduacion.msl;provider=System.Data.SqlClient;pro
           vider connection string=&quot;Data
           Source=VIRTUAL1\SQLEXPRESS;Initial Catalog=Graduacion;Integrated
           Security=True;MultipleActiveResultSets=True&quot;"
           providerName="System.Data.EntityClient"/>
    <add name="AtencionUsuarioEntities"
           connectionString="metadata=res://*/AtencionUsuario.csdl|res://*/At
           encionUsuario.ssdl|res://*/AtencionUsuario.msl;provider=System.Dat
           a.SqlClient;provider connection string=&quot;Data
           Source=virtual1\SQLEXPRESS;Initial
           Catalog=AtencionUsuario;Integrated
           Security=True;MultipleActiveResultSets=True&quot;"
           providerName="System.Data.EntityClient"/>
  </connectionStrings>
  <location path="Estudiante">
    <system.web>
      <authorization>
        <allow roles="ESTUDIANTE"/>
        <deny users="*"/>
      </authorization>
    </system.web>
  </location>
  <location path="Maestro">
    <system.web>
```

```
    <authorization>
      <allow roles="PROFESOR"/>
      <deny users="*"/>
    </authorization>
  </system.web>
</location>
<location path="Director">
  <system.web>
    <authorization>
      <allow roles="DIRECTOR"/>
      <deny users="*"/>
    </authorization>
  </system.web>
</location>
<!--El siguiente bloque es para autorizar subir archivos usando Ajax Upload
          File-->
<!--<location path="AjaxFileUploadHandler.axd">
  <system.web>
    <authorization>
      <allow users="*"/>
    </authorization>
  </system.web>
</location>-->
<system.web>
  <customErrors mode="Off"/>
  <pages>
    <controls>
      <add tagPrefix="asp"
          namespace="System.Web.UI.DataVisualization.Charting"
       assembly="System.Web.DataVisualization, Version=4.0.0.0,
          Culture=neutral, PublicKeyToken=31bf3856ad364e35" />
    </controls>
  </pages>
  <httpRuntime maxRequestLength="1048576"/>
  <compilation debug="true" targetFramework="4.0">
    <assemblies>
      <add assembly="System.Design, Version=4.0.0.0, Culture=neutral,
          PublicKeyToken=B03F5F7F11D50A3A"/>
      <add assembly="System.Windows.Forms, Version=4.0.0.0, Culture=neutral,
          PublicKeyToken=B77A5C561934E089"/>
      <!--<add assembly="CrystalDecisions.Web, Version=13.0.2000.0,
          Culture=neutral, PublicKeyToken=692FBEA5521E1304"/>
                      <add assembly="CrystalDecisions.Shared,
          Version=13.0.2000.0, Culture=neutral,
          PublicKeyToken=692FBEA5521E1304"/>
                      <add assembly="log4net, Version=1.2.10.0,
          Culture=neutral, PublicKeyToken=692FBEA5521E1304"/>
                      <add assembly="CrystalDecisions.ReportSource,
          Version=13.0.2000.0, Culture=neutral,
          PublicKeyToken=692FBEA5521E1304"/>
                      <add
          assembly="CrystalDecisions.ReportAppServer.Controllers,
          Version=13.0.2000.0, Culture=neutral,
          PublicKeyToken=692FBEA5521E1304"/>
```

```
                    <add
        assembly="CrystalDecisions.ReportAppServer.DataDefModel,
        Version=13.0.2000.0, Culture=neutral,
        PublicKeyToken=692FBEA5521E1304"/>
                    <add
        assembly="CrystalDecisions.CrystalReports.Engine,
        Version=13.0.2000.0, Culture=neutral,
        PublicKeyToken=692FBEA5521E1304"/><add
        assembly="CrystalDecisions.ReportAppServer.ClientDoc,
        Version=13.0.2000.0, Culture=neutral,
        PublicKeyToken=692fbea5521e1304"/>-->
    <add assembly="System.Web.Extensions.Design, Version=4.0.0.0,
        Culture=neutral, PublicKeyToken=31BF3856AD364E35"/>
    <add assembly="System.Web.DataVisualization, Version=4.0.0.0,
        Culture=neutral, PublicKeyToken=31BF3856AD364E35"/>
  </assemblies>
</compilation>
<authentication mode="Forms">
  <!--<forms loginUrl="~/Account/Login.aspx" timeout="2880"/>-->
  <forms loginUrl="Login.aspx" name="GRAD.ASPXAUTH" protection="All"
        path="/" />
</authentication>
<authorization>
  <allow users="*"/>
</authorization>
<machineKey

        validationKey="C50B3C89CB21F4F1422FF158A5B42D0E8DB8CB5CDA1742572A4
        87D9401E3400267682B202B746511891C1BAF47F8D25C07F6C39A104696DB51F17
        C529AD3CABE"
      decryptionKey="8A9BE8FD67AF6979E7D20198CFEA50DD3D3799C77AF2B72F"
      validation="SHA1" />
<membership>
  <providers>
    <clear/>
    <add name="AspNetSqlMembershipProvider"
        type="System.Web.Security.SqlMembershipProvider"
        connectionStringName="ApplicationServices"
        enablePasswordRetrieval="false" enablePasswordReset="true"
        requiresQuestionAndAnswer="false" requiresUniqueEmail="false"
        maxInvalidPasswordAttempts="5" minRequiredPasswordLength="6"
        minRequiredNonalphanumericCharacters="0"
        passwordAttemptWindow="10" applicationName="/"/>
  </providers>
</membership>
<profile>
  <providers>
    <clear/>
    <add name="AspNetSqlProfileProvider"
        type="System.Web.Profile.SqlProfileProvider"
        connectionStringName="ApplicationServices" applicationName="/"/>
  </providers>
</profile>
<roleManager enabled="false">
  <providers>
```

```
      <clear/>
      <add name="AspNetSqlRoleProvider"
          type="System.Web.Security.SqlRoleProvider"
          connectionStringName="ApplicationServices" applicationName="/"/>
      <add name="AspNetWindowsTokenRoleProvider"
          type="System.Web.Security.WindowsTokenRoleProvider"
          applicationName="/"/>
    </providers>
  </roleManager>
  <httpHandlers>
    <add path="CrystalImageHandler.aspx" verb="GET"
         type="CrystalDecisions.Web.CrystalImageHandler,
         CrystalDecisions.Web, Version=13.0.2000.0, Culture=neutral,
         PublicKeyToken=692fbea5521e1304" />
    <add path="AjaxFileUploadHandler.axd" verb="*"
         type="AjaxControlToolkit.AjaxFileUploadHandler,
         AjaxControlToolkit" />
    <add path="ChartImg.axd" verb="GET,HEAD,POST"
         type="System.Web.UI.DataVisualization.Charting.ChartHttpHandler,
         System.Web.DataVisualization, Version=4.0.0.0, Culture=neutral,
         PublicKeyToken=31bf3856ad364e35"
     validate="false" />
  </httpHandlers>
</system.web>
<system.webServer>
  <defaultDocument>
    <files>
      <add value="login.aspx" />
    </files>
  </defaultDocument>
  <modules runAllManagedModulesForAllRequests="true" />
  <handlers>
    <remove name="ChartImageHandler" />
    <add name="CrystalImageHandler.aspx_GET" verb="GET"
         path="CrystalImageHandler.aspx"
     type="CrystalDecisions.Web.CrystalImageHandler, CrystalDecisions.Web,
         Version=13.0.2000.0, Culture=neutral,
         PublicKeyToken=692fbea5521e1304"
     preCondition="integratedMode" />
    <!--El siguiente es para que funcione el control AjaxFileUpload-->
    <add name="AjaxFileUploadHandler" verb="*"
         path="AjaxFileUploadHandler.axd"
     type="AjaxControlToolkit.AjaxFileUploadHandler, AjaxControlToolkit" />
    <add name="ChartImageHandler" preCondition="integratedMode"
         verb="GET,HEAD,POST"
     path="ChartImg.axd"
         type="System.Web.UI.DataVisualization.Charting.ChartHttpHandler,
         System.Web.DataVisualization, Version=4.0.0.0, Culture=neutral,
         PublicKeyToken=31bf3856ad364e35" />
  </handlers>
  <validation validateIntegratedModeConfiguration="false" />
</system.webServer>
<system.serviceModel>
  <bindings>
    <basicHttpBinding>
```

```
      <binding name="wsSeguimientoTesisSoap" closeTimeout="00:01:00"
       openTimeout="00:01:00" receiveTimeout="00:10:00"
          sendTimeout="00:01:00"
       allowCookies="false" bypassProxyOnLocal="false"
          maxBufferPoolSize="524288"
       maxReceivedMessageSize="2147483647" useDefaultWebProxy="true">
        <security>
          <transport realm="" />
        </security>
      </binding>
    </basicHttpBinding>
    <customBinding>
      <binding name="wsSeguimientoTesisSoap12">
        <textMessageEncoding maxReadPoolSize="64" maxWritePoolSize="16"
         messageVersion="Soap12" writeEncoding="utf-8">
          <readerQuotas maxDepth="32" maxStringContentLength="8192"
         maxArrayLength="16384"
           maxBytesPerRead="4096" maxNameTableCharCount="16384" />
        </textMessageEncoding>
        <httpTransport manualAddressing="false" maxBufferPoolSize="524288"
         maxReceivedMessageSize="65536" allowCookies="false"
          authenticationScheme="Anonymous"
         bypassProxyOnLocal="false" decompressionEnabled="true"
          hostNameComparisonMode="StrongWildcard"
         keepAliveEnabled="true" maxBufferSize="65536"
          proxyAuthenticationScheme="Anonymous"
         realm="" transferMode="Buffered"
          unsafeConnectionNtlmAuthentication="false"
         useDefaultWebProxy="true" />
      </binding>
    </customBinding>
  </bindings>
  <client>
    <endpoint
          address="http://app.pucesa.edu.ec:9000/serviciosacademics/wsseguim
          ientotesis.asmx"
     binding="basicHttpBinding"
          bindingConfiguration="wsSeguimientoTesisSoap"
     contract="wspucesa.wsSeguimientoTesisSoap"
          name="wsSeguimientoTesisSoap" />
    <!--<endpoint
          address="http://app.pucesa.edu.ec:9000/serviciosacademics/wsseguim
          ientotesis.asmx"
  binding="customBinding" bindingConfiguration="wsSeguimientoTesisSoap12"
  contract="wspucesa.wsSeguimientoTesisSoap" name="wsSeguimientoTesisSoap12"
          />-->
  </client>
</system.serviceModel>
<runtime>
  <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
    <dependentAssembly>
      <assemblyIdentity name="WebGrease" publicKeyToken="31bf3856ad364e35"
          culture="neutral"/>
      <bindingRedirect oldVersion="0.0.0.0-1.5.2.14234"
          newVersion="1.5.2.14234"/>
```

```
      </dependentAssembly>
      <dependentAssembly>
        <assemblyIdentity name="AjaxControlToolkit"
            publicKeyToken="28f01b0e84b6d53e" culture="neutral"/>
        <bindingRedirect oldVersion="0.0.0.0-1.0.20229.20821"
            newVersion="1.0.20229.20821"/>
      </dependentAssembly>
    </assemblyBinding>
  </runtime>
</configuration>
```

Apéndice E

Encuestas de satisfacción al personal de la OIP con firmas de responsabilidad

En las siguientes páginas se muestran copias de las encuestas de satisfacción aplicadas al personal de la OIP llenadas a mano por ellos y con sus respectivas firmas de responsabilidad.

CUESTIONARIO DE SATISFACCIÓN PARA LA EVALUACIÓN DE LA APLICACIÓN WEB "ATENCIONUSUARIO", CORRESPONDIENTE AL TEMA DE PROYECTO DE INVESTIGACIÓN Y DESARROLLO DE GRADO TITULADO "DESARROLLO DE UNA APLICACIÓN WEB PARA LA OPTIMIZACIÓN DE PROCESOS DE ATENCIÓN AL USUARIO DE UNA UNIDAD DE POSTGRADOS"

Aplicado a: Lcda. Doris Constanza Rosales Vacas (usuaria de la aplicación)

(Marque con una "X" la opción que le parezca más adecuada para cada pregunta)

Preguntas:

1. ¿En qué grado permite la aplicación registrar los procesos de graduación de los maestrantes de la PUCESA de manera ágil y directamente en el estado en el que se encuentra actualmente cada uno?

   Excelente ☑ Muy bueno ☐ Bueno ☐ Regular ☐ Malo ☐

2. ¿En qué grado permite la aplicación registrar las fechas correspondientes a cada paso del proceso de graduación de los maestrantes de la PUCESA con la finalidad de guardar información veraz y precisa de cada caso?

   Excelente ☑ Muy bueno ☐ Bueno ☐ Regular ☐ Malo ☐

3. ¿En qué grado detecta la aplicación cuando el usuario ingresa datos erróneos o realiza operaciones inconsistentes, desplegando mensajes de error detallados y comprensibles para el usuario?

   Excelente ☐ Muy bueno ☑ Bueno ☐ Regular ☐ Malo ☐

4. ¿Lista la aplicación correctamente todos los casos de maestrantes de la PUCESA en estado activo, pasivo, graduados y que requieren tomar actualizaciones, con toda su información detallada pertinente, pudiéndose filtrar dicha información por carreras?

   Excelente ☐ Muy bueno ☑ Bueno ☐ Regular ☐ Malo ☐

5. ¿Lista la aplicación correctamente todos los casos de maestrantes de la PUCESA que se encuentran en un determinado paso del proceso de graduación, con toda su información detallada pertinente, pudiéndose filtrar dicha información por carreras?

Excelente ☐ Muy bueno ☑ Bueno ☐ Regular ☐ Malo ☐

6. ¿En qué grado despliega y maneja la aplicación la información correspondiente a maestrantes, docentes y carreras de posgrados preexistente en las bases de datos de la PUCESA?

Excelente ☐ Muy bueno ☑ Bueno ☐ Regular ☐ Malo ☐

7. ¿Permite la aplicación visualizar la información detallada de un maestrante específico incluyendo aquella relativa a su estado actual dentro del proceso de graduación?

Excelente ☐ Muy bueno ☑ Bueno ☐ Regular ☐ Malo ☐

8. ¿Permite la aplicación visualizar la información detallada de un docente específico incluyendo el número de proyectos en los que se encuentra trabajando actualmente ya sea en calidad de tutor, informante, director o revisor?

Excelente ☐ Muy bueno ☑ Bueno ☐ Regular ☐ Malo ☐

9. ¿Permite la aplicación registrar y actualizar adecuadamente las direcciones de correo electrónico de maestrantes y docentes?

Excelente ☐ Muy bueno ☑ Bueno ☐ Regular ☐ Malo ☐

10. ¿En qué grado genera la aplicación los cuadros estadísticos requeridos por la OIP, reflejando verazmente los datos registrados sobre los procesos de graduación y estado de los maestrantes?

Excelente ☐ Muy bueno ☑ Bueno ☐ Regular ☐ Malo ☐

11. ¿En qué grado considera usted que la aplicación es ágil y que el tiempo de carga de sus diferentes páginas web es reducido?

Excelente ☐ Muy bueno ☑ Bueno ☐ Regular ☐ Malo ☐

12. ¿En qué grado considera usted que la aplicación es clara, intuitiva, amigable y fácil de usar, funcionando sin que requiera de conocimientos técnicos por parte del usuario?

Excelente ☐ Muy bueno ☑ Bueno ☐ Regular ☐ Malo ☐

13. En términos generales, ¿En qué grado considera usted que la aplicación cumple los objetivos para los que fue creada y será de la utilidad prevista para la OIP?

Excelente ☐ Muy bueno ☑ Bueno ☐ Regular ☐ Malo ☐

Lcda. Doris Constanza Rosales Vacas

Secretaria de la OIP / Usuaria de la Aplicación

CUESTIONARIO DE SATISFACCIÓN PARA LA EVALUACIÓN DE LA APLICACIÓN WEB "ATENCIONUSUARIO", CORRESPONDIENTE AL TEMA DE PROYECTO DE INVESTIGACIÓN Y DESARROLLO DE GRADO TITULADO "DESARROLLO DE UNA APLICACIÓN WEB PARA LA OPTIMIZACIÓN DE PROCESOS DE ATENCIÓN AL USUARIO DE UNA UNIDAD DE POSTGRADOS"

Aplicado a: Mtr. Diego Armando Jiménez Bosquez (usuario de la aplicación)

(Marque con una "X" la opción que le parezca más adecuada para cada pregunta)

Preguntas:

1. ¿En qué grado permite la aplicación registrar los procesos de graduación de los maestrantes de la PUCESA de manera ágil y directamente en el estado en el que se encuentra actualmente cada uno?

   Excelente ☐ Muy bueno ☒ Bueno ☐ Regular ☐ Malo ☐

2. ¿En qué grado permite la aplicación registrar las fechas correspondientes a cada paso del proceso de graduación de los maestrantes de la PUCESA con la finalidad de guardar información veraz y precisa de cada caso?

   Excelente ☐ Muy bueno ☒ Bueno ☐ Regular ☐ Malo ☐

3. ¿En qué grado detecta la aplicación cuando el usuario ingresa datos erróneos o realiza operaciones inconsistentes, desplegando mensajes de error detallados y comprensibles para el usuario?

   Excelente ☐ Muy bueno ☒ Bueno ☐ Regular ☐ Malo ☐

4. ¿Lista la aplicación correctamente todos los casos de maestrantes de la PUCESA en estado activo, pasivo, graduados y que requieren tomar actualizaciones, con toda su información detallada pertinente, pudiéndose filtrar dicha información por carreras?

   Excelente ☐ Muy bueno ☒ Bueno ☐ Regular ☐ Malo ☐

5. ¿Lista la aplicación correctamente todos los casos de maestrantes de la PUCESA que se encuentran en un determinado paso del proceso de graduación, con toda su información detallada pertinente, pudiéndose filtrar dicha información por carreras?

Excelente ☐ Muy bueno ☒ Bueno ☐ Regular ☐ Malo ☐

6. ¿En qué grado despliega y maneja la aplicación la información correspondiente a maestrantes, docentes y carreras de posgrados preexistente en las bases de datos de la PUCESA?

Excelente ☐ Muy bueno ☒ Bueno ☐ Regular ☐ Malo ☐

7. ¿Permite la aplicación visualizar la información detallada de un maestrante específico incluyendo aquella relativa a su estado actual dentro del proceso de graduación?

Excelente ☐ Muy bueno ☒ Bueno ☐ Regular ☐ Malo ☐

8. ¿Permite la aplicación visualizar la información detallada de un docente específico incluyendo el número de proyectos en los que se encuentra trabajando actualmente ya sea en calidad de tutor, informante, director o revisor?

Excelente ☐ Muy bueno ☒ Bueno ☐ Regular ☐ Malo ☐

9. ¿Permite la aplicación registrar y actualizar adecuadamente las direcciones de correo electrónico de maestrantes y docentes?

Excelente ☐ Muy bueno ☒ Bueno ☐ Regular ☐ Malo ☐

10. ¿En qué grado genera la aplicación los cuadros estadísticos requeridos por la OIP, reflejando verazmente los datos registrados sobre los procesos de graduación y estado de los maestrantes?

Excelente ☐ Muy bueno ☒ Bueno ☐ Regular ☐ Malo ☐

11. ¿En qué grado considera usted que la aplicación es ágil y que el tiempo de carga de sus diferentes páginas web es reducido?

Excelente ☐ Muy bueno ☒ Bueno ☐ Regular ☐ Malo ☐

12. ¿En qué grado considera usted que la aplicación es clara, intuitiva, amigable y fácil de usar, funcionando sin que requiera de conocimientos técnicos por parte del usuario?

Excelente ☐ Muy bueno ☒ Bueno ☐ Regular ☐ Malo ☐

13. En términos generales, ¿En qué grado considera usted que la aplicación cumple los objetivos para los que fue creada y será de la utilidad prevista para la OIP?

Excelente ☐ Muy bueno ☒ Bueno ☐ Regular ☐ Malo ☐

Mtr. Diego Armando Jiménez Bosquez

Coordinador de la OIP / Usuario de la Aplicación

Referencias

[1] G. De la Torre, *Desarrollo de una Aplicación Web para la Administración de Procesos de Graduación de una Unidad de Postgrados*. PUCESA, 2016

[2] Escuela Superior Politécnica del Litoral, *Sistema de Administración Académica Postgrado – SAACP* [online]. 2008, disponible en: http://www.csi.espol.edu.ec/ui/es/content/sistema/sistema.aspx?op=toshow&id=106

[3] X. Castillo, C. Fernández, R. Rea y J. Tapia, *Desarrollo del sistema de gestión académica de postgrados de la Universidad de Cuenca (SGAP)*, Universidad de Cuenca, 2012.

[4] PUCESA. *Manual de usuario del Sistema Academics* [online]. 2012, disponible en: http://pucesa.edu.ec/sites/default/files/ManualdeUsuariodel_SistemaAcademics.pdf

[5] M. Celma, J. Casamayor y L. Mota, *Bases De Datos Relacionales*. Pearson Educación, 2003.

[6] G. Ponjuán, *Gestión documental, de información y del conocimiento... puntos de contacto y diferencias*. Ciencias de la Información, 34(3), 2003.

[7] J. Hessen, J. Gaos y F. Romero, *Teoría del conocimiento.* Espasa-Calpe, 1970.

[8] L. Aja, *Gestión de información, gestión del conocimiento y gestión de la calidad en las organizaciones.* ACIMED, 2002.

[9] L. Codinas, *Web semántica y sistemas de información documental*. Ed. Trea, 2009.

[10] A. Grau, *Herramientas de gestión del conocimiento*. Fundación Iberoamericana del Conocimiento, 2001.

[11] International Organization for Standarization, *ISO 15489-1: Information and Documentation. Records Management General.* Geneva: International Organization for Standarization, 2001.

[12] Doknos. *Quipux para su Gestión Documental* [online]. 2010, disponible en: http://www.doknos.com/en/node/70

[13] Correlibre.org, *Herramienta de gestión documental y de procesos OrfeoGPL* [online]. 2013, disponible en: http://orfeogpl.org/ata/

[14] M. Roberge, *Lo esencial de la gestión documental: sistemas integrales de gestión de los documentos electrónicos.* Quebec: Gestar, 2006.

[15] D. Gallardo, y C. Pomares, *Programación orientada a objetos.* Lenguajes y Paradigmas de Programación, 2008.

[16] F. Durán, F. Gutiérrez y E. Pimentel, *Programación orientada a objetos con Java*. Editorial Paraninfo, 2007.

[17] D. Barnes y M. Kölling, *Programación orientada a objetos con Java. Una introducción práctica usando BlueJ.* Pearson Education, 2007.

[18] M. Álvarez, *Polimorfismo en Programación Orientada a Objetos* [online]. 2014, disponible en: https://desarrolloweb.com/articulos/polimorfismo-programacion-orientada-objetos-concepto.html

[19] P. Pomol, *Programación Orientada a Objetos* [online]. 2016, disponible en: http://eportafoliopoopatriciapomol.blogspot.com/2016/

[20] R. Vargas, y J. Maltés, *Programación en Capas.* Di Mare, Costa Rica, 2007.

[21] J. Lopera, *Programación por capas* [online]. 2009, disponible en: http://tecnicojuansistemas.blogspot.com/2009/04/programacion-por-capas.html

[22] L. Lucarella, *Recolección de basura en D.* Universidad de Buenos Aires, 2010.

[23] I. Sommerville, *Ingeniería del software.* Pearson Education, 2005.

[24] M. Marqués, *Bases de datos.* Castelló de la Plana: Publicacions de la Universitat Jaume I. Servei de Comunicació i Publicacions, 2011.

[25] R. Pressman y J. Troya, *Ingeniería del software.* McGraw-Hill, 1988.

[26] D. Avison y G. Fitzgerald, *Information systems development: methodologies, techniques and tools.* McGraw Hill, 2003.

[27] O. Tinoco, P. Rosales y J. Salas, *Criterios de selección de metodologías de desarrollo de software.* Industrial Data, 2010.

[28] R. Castro, *Estructura básica del proceso unificado de desarrollo de software*. Universidad ICESI, 2006.

[29] B. Boehm, *A spiral model of software development and enhancement*. IEEE, 1988.

[30] C. Acosta, *Diseño e implementación de una herramienta de representación del conocimiento para apoyar la gestión de requisitos en un proceso de desarrollo de software*. Universidad de Chile, 2010.

[31] J. Canós, P. Letelier y M. Penadés, *Metodologías ágiles en el desarrollo de software.* Universidad Politécnica de Valencia, 2003.

[32] P. Letelier, *Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP)*. Universidad Politécnica de Valencia, 2006.

[33] K. Beck, *Extreme programming explained: embrace change.* Addison-Wesley Professional, 2000.

[34] L. Pérez, *XP – Extreme Programming* [online]. 2011, disponible en: http://analystperez.blogspot.com/2011/04/xp-extreme-programming.html

[35] R. Jeffries, A. Anderson y C. Hendrickson, *Extreme programming installed.* Addison-Wesley Professional, 2001.

[36] A. Silberschatz, H. Korth, y S. Sudarshan, *Fundamentos de bases de datos.* McGraw-Hill, 2002.

[37] M. Gómez, *Notas del curso bases de datos.* Universidad Autónoma Metropolitana, 2013.

[38] S. Luján, *Programación de aplicaciones web: historia, principios básicos y clientes web.* Editorial Club Universitario, 2002.

[39] T. Nelson, "Complex information processing: a file structure for the complex, the changing and the indeterminate" In Proceedings of the 1965 20th national conference (pp. 84-100). ACM, 1965.

[40] A. Cobo, *PHP y MySQL: Tecnología para el desarrollo de aplicaciones web.* Ediciones Díaz de Santos, 2005.

[41] F. Berzal, J. Cubero y F. Cortijo, *Desarrollo profesional de aplicaciones web con Asp.net.* iKor Consulting, 2005.

[42] I. Spaanjaars, *Beginning ASP.NET 4 in C# and VB*, Wiley Publishing, Inc, 2010.

[43] J. Rojas, *Gestión por procesos y atención del usuario en los establecimientos del sistema nacional de salud.* Juan Carlos Martínez Coll, 2000.

[44] L. Agudelo y J. Escobar, *Gestión por procesos. Bogotá: Instituto Colombiano de Normas Técnicas y Certificación.* ICONTEC, 2007.

[45] A. Cano, *Sistema de gestión de información en la educación superior, Universidad de Camagüey*. Universidad de Granada, 2016.

[46] H. Pardo, *Un modelo de aplicación web institucional universitaria*. Tesis doctoral, Universidad Autónoma de Barcelona, 2005.

[47] J. Reyes, *Sistema web para la gestión de los procesos críticos de la Dirección de Posgrado de la UTA*. Universidad Técnica de Ambato, 2014.