**PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR**
**FACULTAD DE INGENIERÍA**
**INGENIERÍA DE SISTEMAS Y COMPUTACIÓN**

**DISERTACIÓN PREVIA A LA OBTENCIÓN DEL TÍTULO DE INGENIERO DE SISTEMAS Y COMPUTACIÓN**

# "EVALUACIÓN ENTRE LAS METODOLOGÍAS DE DESARROLLO DE SOFTWARE XP - RUP Y SU APLICACIÓN EN LA CONSTRUCCIÓN DE UN SISTEMA PARA LA GESTIÓN DE UNA ESTÉTICA. CASO DE ESTUDIO: CADAMA ESTÉTICA "

**AUTOR**

DAVID ALEJANDRO ARÁUZ MOYA

**DIRECTOR**

ING. FABIÁN DE LA CRUZ

**QUITO, 2020**

# DEDICATORIA

El presente trabajo de disertación está dedicado a Dios, por ser la luz y la guía de toda mi familia y brindarme sabiduría, fuerza y fe para seguir adelante todos los días.

A mis padres Nadya y Javier por darme la vida tan maravillosa que tengo, por ser los pilares fundamentales en este recorrido, por todos los consejos, enseñanzas valores infundados, sobre todo por la humildad y el amor que ellos saben infundir en mí y en todos mis hermanos.

A mis hermanos Sergio, Emilio y Micaela, por saber alegrar mis días en los momentos más difíciles, y saber que sin ellos no sería la persona que soy ahora, además saber que siempre puedo contar y confiar en ellos.

A mis abuelitos que en todo momento supieron darme apoyo, ánimo, consejos y sobre todo amor, tanto en los buenos como en los malos momentos, ellos han sido testigos del esfuerzo realizado por todo este tiempo.

Al Ing. Fabián de la Cruz por ser un excelente maestro, por siempre estar pendiente, por cada uno de los consejos durante toda la carrera que fueron primordiales para la culminación de la misma, a todos mis profesores que con su sabiduría y enseñanza durante toda la carrera lograron ayudarme directa o indirectamente en la culminación del presente trabajo de disertación.

Finalmente, a mis grandes amigos que siempre han estado tanto en los momentos felices como en los tristes; su amistad y ayuda incondicional, hicieron que se forme, más que una amistad, una hermandad.

¡Gracias a todos!

Esto es para ustedes…

# AGRADECIMIENTOS

Primeramente, quisiera agradecer a Dios, por haberme dado la capacidad y fuerzas para culminar mis estudios.

A la Pontifica Universidad Católica del Ecuador, que ha sido mi segundo hogar, y por haberme brindado una educación de calidad y excelencia, construyendo un ser orientado al mundo profesional, con valores éticos y morales para ser mejor persona en mi vida profesional.

A mi director, Ing. Fabián de la Cruz por el tiempo, esfuerzo, dedicación, quien, con todo su conocimiento, sabiduría y sobre todo motivación, ha conseguido en mí que pueda concluir mis estudios.

Finalmente, a mi familia y amigos que siempre supieron estar en los momentos más indicados, con consejos, apoyo y fuerzas.

Gracias.

# ÍNDICE DE CONTENIDOS

# ÍNDICE DE ILUSTRACIONES

# ÍNDICE DE TABLAS

# ÍNDICE DE DIAGRAMAS

# CAPITULO I: INTRODUCCIÓN

La presente disertación se basa en la evaluación entre las metodologías de desarrollo de software eXtreme Programming (Programación Extrema) - Rational Unified Process (Proceso Unificado de Rational) y su aplicación en la construcción de un sistema para la gestión de una estética. Al comenzar a realizar una aplicación de software, se ve en la necesidad de escoger una metodología de desarrollo que sea factible y adaptable al proceso previamente dicho, y al no tener claro que metodología es adecuada se generan demoras en el tiempo de ejecución.

Cadama, al ser una empresa nueva en el medio, no cuenta con un sistema especializado para la gestión de sus funcionalidades principales, por este motivo se vio la necesidad de crear una aplicación web para poder automatizar estos procesos y mejorar la atención hacia el cliente.

## 1.1. Datos de la Organización o institución

**Nombre de la Empresa**: Cadama Estética.

**Actividad**: Cadama Estética ofrece una alta variedad de servicios estéticos, se especializa en reducción de medidas, tonificación, limpieza facial, eliminación de arrugas, rejuvenecimiento y depilación definitiva.

Todos los servicios estéticos ofrecidos por Cadama se lo realizan sin cirugía alguna, con la experiencia y garantía que tiene, ofrecen la mejor atención y servicio pertinente a sus clientes.

**Ubicación**:

Actualmente se encuentra en:

Oficina Matriz: Naciones Unidas y Núñez de Vela. Edificio Metropolitan, Oficina: 1006.

## 1.2. Alcance

Para realizar la construcción de un sistema para la gestión de una estética se va a efectuar un análisis de las metodologías orientadas al desarrollo de software entre eXtreme Programming (Programación Extrema) y Rational Unified Process (Proceso Unificado de Rational) con el fin de comprender el enfoque del proceso de desarrollo de software de cada una para mejorar, optimizar y sobre todo mantener la calidad en el producto.

Sobre la base de las consideraciones anteriores es conveniente realizar una comparativa para poder seleccionar la mejor de acuerdo a las consideraciones del sistema que se planea construir.

Al no contar con una página web ni con los procesos definidos dentro de la empresa, se procederá a realizar un análisis de la situación actual para poder abordar los problemas de manera optimizada con el fin de reducir tiempo, costo y recursos.

De acuerdo con los razonamientos que se han venido realizando, con una página definida dentro de la empresa, se pueden automatizar y definir varios procesos actualmente inexistentes, para comenzar con una mejora continua que además pueda satisfacer al cliente ayudándolo a facilitar tareas.

Una vez realizado las pruebas respectivas por los usuarios finales, se realizará la publicación de la aplicación en el servidor web para corroborar el funcionamiento correcto.

## 1.3. Objetivos

### *1.3.1. Objetivo General*

- Evaluar las metodologías de desarrollo de software XP - RUP y aplicar en la construcción de un sistema para la gestión de una estética.

### *1.3.2. Objetivos Específicos*

- Identificar los planteamientos de cada una de las metodologías de desarrollo de software que van a ser evaluadas.
- Definir los criterios de comparación para la evaluación de las metodologías de desarrollo.
- Efectuar el análisis completo de requerimientos para un óptimo desarrollo en la gestión en la estética.
- Investigar las herramientas necesarias para diseñar la aplicación.
- Desarrollar la aplicación que cumpla con los requisitos de la empresa.
- Implementar una aplicación intuitiva y fácil de usar para que cumpla con las necesidades tanto del usuario final como de la empresa.

## 1.4. Mapa de Procesos

El mapa de procesos es un diagrama que muestra una visión general de cómo está funcionando la empresa con respecto a la cadena de valor. Consta de 3 procesos fundamentales:

### 1.4.1. Procesos Estratégicos

Son los procesos que especifican cómo se maneja el negocio y de qué modo se genera valor tanto para el cliente como para la organización. Los encargados de manejar esta responsabilidad son la alta dirección o gerencia general, que a través de la planificación estratégica les permite conocer los planes operativos, objetivos y metas a alcanzar a corto, mediano o largo plazo.

### 1.4.2. Procesos Generadores de valor

Son los procesos que están relacionados directamente al propósito de la empresa, es decir, a los servicios que ofrecen, está dirigido principalmente hacia el cliente.
Cadama Estética opera con 3 procesos fundamentales, corporales, faciales y masajes, y estos a la vez constan de subprocesos para complementar las actividades.

### 1.4.3. Procesos de Apoyo

Son los procesos que aportan, mejoran y apoyan a los procesos generadores de valor, contribuyen a los principales procesos de la empresa a tal punto que son los encargados de cumplir con los objetivos de cada proceso para cubrir con las necesidades de los clientes.
Dentro de Cadama Estética se apoyan en dos procesos:

- Recursos Humanos. - Encargado de realizar el reclutamiento, selección y contratación del personal dentro de la empresa, con el fin de contar con personal capaz de cumplir con los objetivos de la empresa y satisfacer al cliente.

- Recursos Financieros. - Encargado de llevar tanto la contabilidad como la gestión de impuestos de la empresa.

En la Ilustración 1 se presenta el mapa de procesos de Cadama Estética proporcionado por la empresa en el que se detalla los procesos estratégicos, generadores de valor y de apoyo.

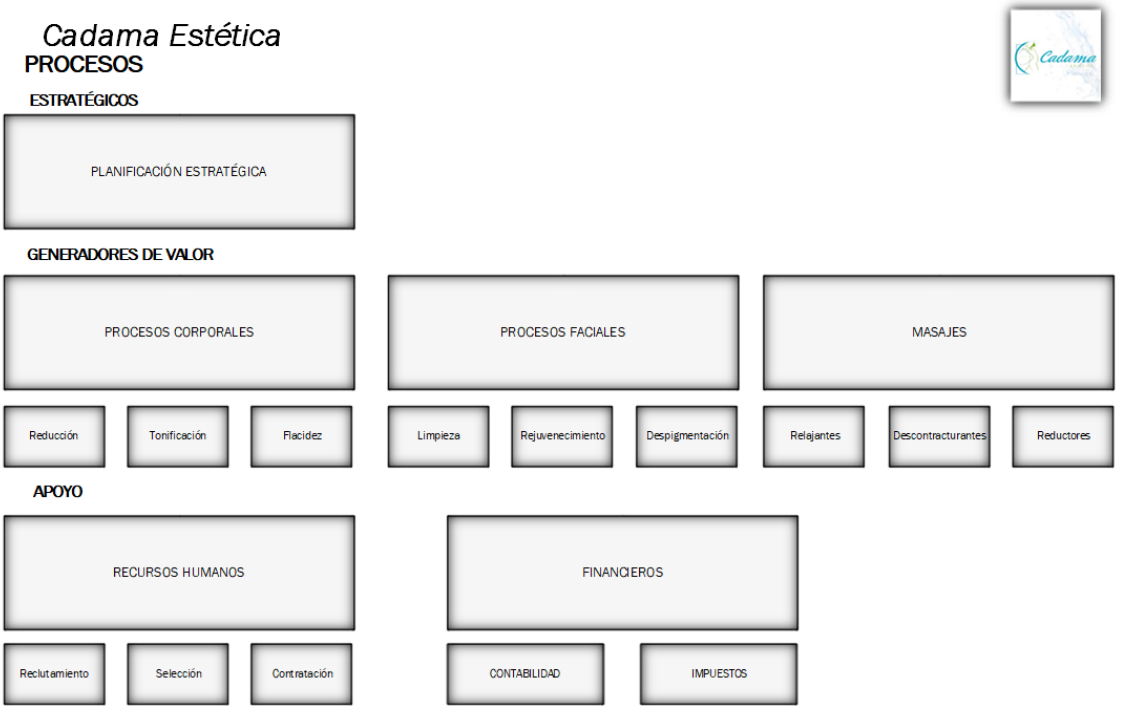



Ilustración 1 - Mapa de Procesos

Fuente: Cadama Estética

## 1.5. Diagrama Jerárquico

La empresa cuenta con una estructura organizativa tal y como se muestra en la Ilustración 2.

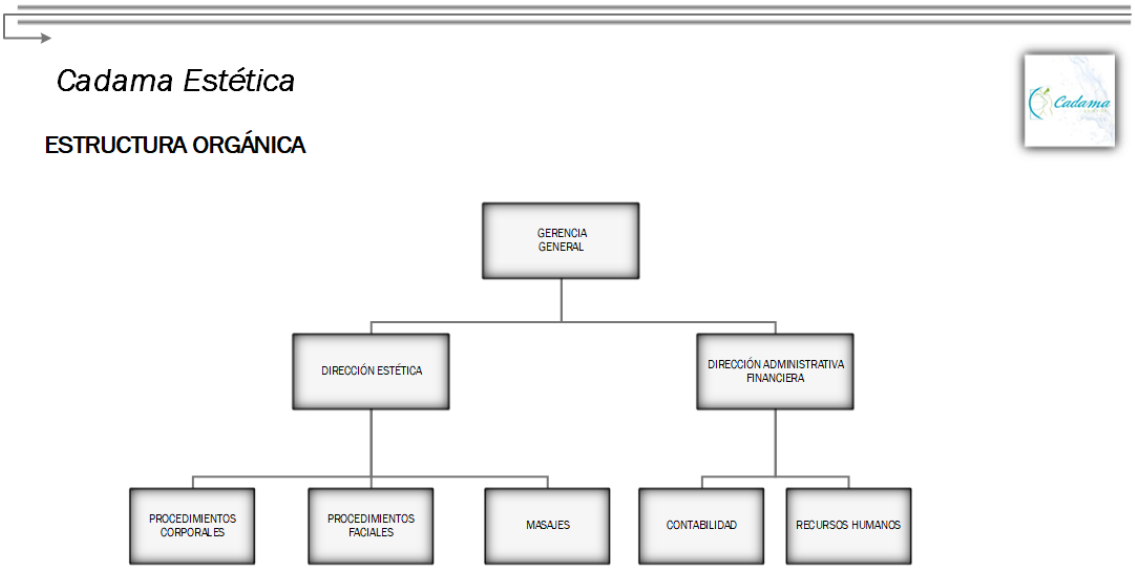



Ilustración 2 - Diagrama Jerárquico

Fuente: Cadama Estética

# CAPÍTULO 2: MARCO TEÓRICO

En este capítulo se maneja todos los aspectos técnicos de las metodologías a analizar en la disertación para el desarrollo del sistema. Se expone cada metodología a cabalidad con sus respectivas características para después realizar la selección que mejor convenga para el sistema.

Además, se definen las herramientas y el entorno de trabajo con el fin de saber en qué lenguaje se lo va a realizar, el motor de base de datos, el uso de Frameworks para agilitar y facilitar el desarrollo del sistema.

## 2.1. Concepto de Metodología de Desarrollo de Software

El concepto de metodología de desarrollo de software viene dado por un grupo de procesos o técnicas para la correcta implementación de software como producto.

Estas metodologías son guías en las que se muestran detalladamente las tareas que se van a efectuar en el proceso, para lograr el resultado esperado, además, se indican las personas responsables o encargadas de desarrollar las distintas actividades y cuáles son sus funciones.

Existen diferentes tipos de metodologías dentro del desarrollo de software, que están centradas tanto al cliente como a la documentación en particular, cada una de éstas con diferentes técnicas para el desarrollo.

Estas técnicas muestran el proceso que debe seguir las actividades técnicas propiamente dichas que se encuentran descritas en la metodología. Adopta el uso de modelos gráficos adicional con el uso de procedimientos detallados.

Hay que tener en cuenta que la técnica a utilizar puede ser usada en una o varias actividades de la metodología de desarrollo de software, asimismo, procurar tener en cuenta cuando se va a realizar un cambio de técnica por otra.

Para la presente disertación se revisará dos de las principales metodologías que se usan para el desarrollo de software XP y RUP, a continuación, se presenta los principales conceptos, características, ventajas y desventajas de las metodologías a evaluar.

## 2.2. Metodología de desarrollo Extreme Programming

Extreme Programming o XP es una metodología que fue creada en el año del 2001 por un grupo de 17 expertos que habían ya creado o impulsado metodologías anteriores.
Uno de los principales objetivos de esta metodología es fortalecer las relaciones entre el desarrollador y el usuario final para llegar al éxito en el desarrollo de software, trabajando conjuntamente entre las dos partes y proporcionando un buen ambiente laboral.

Para que funcione esta metodología, debe existir una buena retroalimentación entre los desarrolladores y el usuario final, es decir, contar con una buena comunicación entre los participantes, soluciones sencillas, prácticas y eficientes.

Es conveniente para proyectos que tienen requerimientos indeterminados y que varían con el tiempo, en el cual hay un alto riesgo técnico por los constantes cambios realizados al desarrollar el proyecto (Letelier & Penadés, 2006).

En la Ilustración 3 se detalla el proceso iterativo del ciclo de vida de la metodología XP:

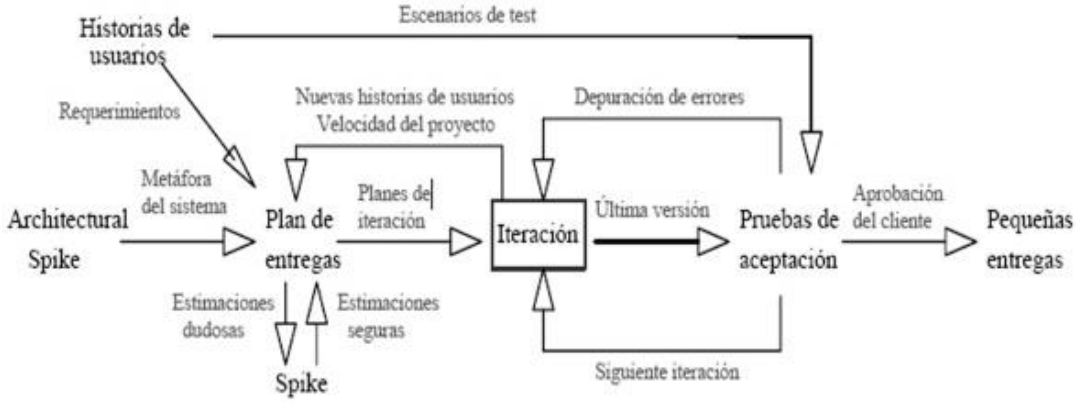



Ilustración 3 - Ciclo de Vida XP

Fuente: (Beck, 1999)

### *2.2.1. Las Historias de Usuario*

Las historias de usuario dentro de XP se utilizan para detallar los requerimientos de software.
A través de tarjetas generalmente de papel, el usuario final detalla rápidamente las características que la aplicación deberá poseer, sean requisitos funcionales o no funcionales (Letelier & Penadés, 2006).

Los manejos de las historias de usuario son básicamente manejables y a la vez adaptables, ya que, dependiendo de las historias, estas pueden remplazarse y eliminarse por otras más específicas y detalladas o a su vez crear o modificar las ya existentes.

Cada una de estas historias de usuarios son entendibles y claramente definidas para que los programadores responsables estén en la capacidad de entender lo que dice, desarrollarlo e implementarlo en el tiempo adecuado.

Al comenzar a identificar todas las historias de usuario, se debe dejar a un lado la preocupación, porque al iniciar la iteración se registrarán los cambios realizados en las historias de usuario y en base a esto se procederá a la planificación de la siguiente iteración, estas historias son divididas en tareas de programación las cuales deben ser asignadas a cada programador para ser implementada en esa iteración.

### *2.2.2. Proceso XP*

Un proyecto implementado con XP culmina con éxito cuando el cliente se basa en el equipo y su habilidad para gestionar las funcionalidades que puede entregar en un período de tiempo, el proceso de desarrollo en general se basa en los siguientes pasos:

1. El cliente define el valor de negocio a implementar.
2. El programador mide el tiempo y esfuerzo preciso para la implementación.
3. El cliente elige que se va a elaborar, con respecto a las necesidades y limitaciones de tiempo.
4. El programador construye ese valor de negocio.
5. Regresa al paso número 1.

El proceso o ciclo de vida de la metodología XP cuenta con seis fases que se van a detallar a continuación:

#### Fase I: Exploración

En esta etapa, los clientes empiezan con la descripción de las historias de usuario, estas a su vez, son de suma importancia para el primer reléase del producto. Simultáneamente, el equipo de desarrolladores se adapta a nuevas tecnologías, buenas prácticas de programación y todas las nuevas herramientas que se toman para operar el proyecto. Se realiza las pruebas respectivas sobre la nueva tecnologías y se explora cual podría ser la eventual arquitectura del sistema mediante la fabricación de un prototipo. La fase de exploración lleva de unas pocas semanas a unos pocos meses, de acuerdo al tamaño y la familiaridad que los programadores tienen con la tecnología.

### Fase II: Planificación de la Entrega

Durante esta fase, el cliente prioriza las historias de usuario y, como resultado, los programadores evalúan el esfuerzo requerido de cada uno. Las primeras entregas de los contenidos del programa son creadas en colaboración con el cliente. La entrega se realizará en un plazo no superior a tres meses. Esta fase tiene una duración de varios días.

### Fase III: Iteraciones

En esta etapa contiene algunas iteraciones en el sistema antes de ser entregado. El plan de entrega se compone de iteraciones y estas generalmente no pasan de dos o tres semanas. En la primera iteración, la arquitectura del sistema se plantea y define para implementarlo durante todo el proyecto. Para ello, se selecciona las historias para la respectiva creación. No obstante, a veces esto no es posible, ya que es el cliente quien decide qué historias se implementará en cada iteración. Al concluir la última iteración, el sistema está listo para entrar en producción.

Etapa IV: El sistema de producción de paso de la producción antes de ser enviado al entorno del cliente requiere pruebas adicionales y evaluación del desempeño. Esta disposición se propuso solución

### Fase IV: Producción

En la fase de producción, antes que el sistema sea migrado al entorno del cliente, demanda pruebas y evaluaciones de rendimiento adicionales. A su vez, se toman varias resoluciones con respecto a la inserción de nuevas características en la versión actual debido a los cambios realizados durante esta fase.
Es viable que el tiempo empleado en cada iteración se reduzca de tres a una semana. Las ideas y sugerencias propuestas se documentan y preparan para su implantación posterior.

### Fase V: Mantenimiento

Mientras que la primera versión está en producción, el proyecto que utiliza XP debe conservar el sistema en ejecución mientras se desarrollan nuevas iteraciones.

Para ello, el cliente debe realizar tareas de soporte, por lo tanto, la velocidad de desarrollo se puede reducir después de la salida a producción del sistema.

### Fase VI: Muerte del proyecto

Esto fase sucede cuando el cliente ya no tiene historias para añadir en el sistema, esto supone que las necesidades se satisfacen en otros aspectos, como el rendimiento y la fiabilidad. La documentación final del sistema se creó y no existe ningún otro cambio en la arquitectura. La culminación del proyecto también corresponde cuando las expectativas del sistema no son las adecuadas para el cliente o cuando no cuenta con en el presupuesto para poder mantenerlo.

## 2.3. Metodología de desarrollo RUP

Esta es la metodología creada por Grady Butch, Ivar Jacobson y James Rumbaugh, que apareció en 1998, pero estuvo disponible para el público en general en 1999, su negocio principal son los procesos, las personas y las herramientas.

Este es un modelo que posibilita el desarrollo de software de una manera exhaustiva a través de un proceso de feedback y pruebas constantes, asegurando que los niveles de calidad se lleven a cabo, pero generando problemas en el manejo de la complejidad en el control administrativo de estos últimos, aunque, los beneficios son recompensados por el esfuerzo invertido en este aspecto.

El Proceso Unificado de Rational se basa en componentes, es decir, que el sistema a ser construido está conformado por componentes de software que están conectados por medio de interfaces previamente definidas. Esta metodología utiliza el Lenguaje Unificado de Modelado (UML) para modelar con antelación los esquemas del sistema software, es una parte esencial para el proceso.
Los aspectos que definen como único al Proceso Unificado de Rational se resumen en tres elementos fundamentales:

- Dirigido por casos de uso
- Centrado en la arquitectura
- Iterativo e Incremental

Dirigido por casos de uso

"Un caso de uso es un fragmento de funcionalidad del sistema que proporciona al usuario un resultado importante" (Jacobson, Booch, & Rumbaugh, 2000)

Los casos de uso representan los requisitos funcionales, los cuales se unifican formando un modelo general que describe la funcionalidad total del sistema.

Centrado en la arquitectura

"El concepto de arquitectura se software incluye aspectos estáticos y dinámicos más significativos del sistema" (Jacobson, Booch, & Rumbaugh, 2000)

La arquitectura es una vista de diseño organizada con sus partes más importantes dejando de lado los detalles más mínimos, este proceso ayuda a centrarse en los objetivos apropiados, como la comprensibilidad, la capacidad para adaptarse a cambios, y la reutilización.

Iterativo e Incremental

"Las iteraciones hacen referencia a pasos en el flujo de trabajo, y los incrementos, al crecimiento del producto. Para una efectividad máxima, las iteraciones deben estar controladas, es decir, deben seleccionarse y ejecutarse de una forma planificada." (Jacobson, Booch, & Rumbaugh, 2000)

Esta fase cuenta con varias iteraciones, que a su vez se convierten en entregables, y cada vez que se cumple una iteración, su versión se mejora en diferentes etapas.

El ciclo de vida RUP se encuentra representado en la Ilustración 4:

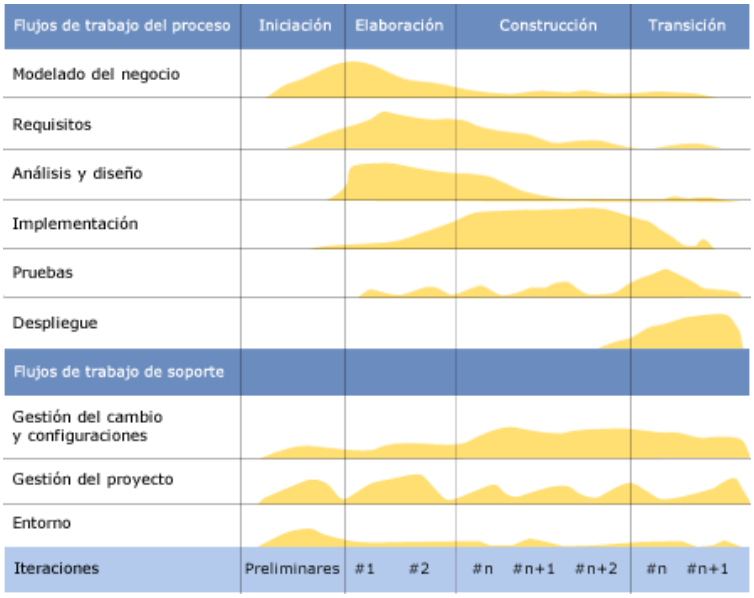



*Ilustración 4 - Ciclo de Vida RUP*

*Fuente: (Jacobson, Booch, & Rumbaugh, 2000)*

### *2.3.1. Orden del ciclo de vida de la metodología RUP*

#### Fase de iniciación

El propósito de esta fase es identificar y ver cual es el alcance del proyecto el cliente, determinar cuáles son los riesgos ligados al proyecto, tener una idea clara acerca de la arquitectura de software a utilizar y con esto elaborar el plan de proyecto con sus respectivas iteraciones.

#### Fase de elaboración

Se elabora los casos de uso y se seleccionan para determinar y desarrollar la arquitectura del sistema, se completan la especificación de los casos de uso y se realiza un primer análisis del área del problema con todo esto la solución inicial está prediseñada.

#### Fase de construcción

La intención de esta etapa es ultimar el funcionamiento del sistema y las mejoras pertinentes, para lo cual es necesario aclarar los requisitos pendientes, los cambios deben realizarse de acuerdo a las estimaciones y pruebas realizadas por parte de los usuarios

### Fase de transición

Esta fase tiene como objetivo el garantizar el software como recurso para el cliente final, arreglar los errores y defectos generados en las pruebas de integración y aceptación, una vez cumplido el paso anterior, proporcionar el soporte técnico y la capacitación necesaria para los usuarios. Finalmente se debe tener en cuenta que los requerimientos solicitados por los usuarios implicados en el proyecto, deben estar cumplidos dentro del producto final.

## 2.4. Herramientas

### *2.4.1. Lenguaje de Programación PHP*

Para la aplicación web a realizar se va a utilizar el lenguaje de programación PHP, porque es un lenguaje libre y abierto, además como es muy popular, existe una gran comunidad de desarrolladores la cual pueden dar soporte y ayuda a cualquier requerimiento que se tenga.

PHP es un lenguaje de código abierto muy popular que se centra en la creación de scripts del lado del servidor, que es particularmente apropiado para el desarrollo web y se puede integrar con HTML.

PHP es un lenguaje interpretado del lado del servidor que está compuesto de una secuencia de comandos y está diseñado para el desarrollo Web. Para que las páginas funcionen, el código PHP se incrusta en el código HTML y lo ejecuta en el servidor web, que debe interpretarse antes de mostrarse al lado del cliente.

Características de PHP

- Fácil de usar: PHP es un lenguaje fácil de aprender y sencillo de manejar, debido a que se basa en los lenguajes Cobol y C, además la mayoría de funciones vienen integradas por defecto, como la utilización de base de datos MySql.
- Multiplataforma: PHP opera sin ningún problema en varios sistemas operativos, como Linux, Windows,
- Uso de bases de datos: Primero, cuenta con una conexión propia a una base de datos (p.ej., MySql), segundo, puede tener acceso a distintas bases de datos que admitan el modelo de conexión a través de la extensión ODBC.

- Costo: Es totalmente gratis, se lo puede descargar directamente desde su página oficial: www.php.net.
- Compatibilidad: Desde la versión 5 de PHP, utiliza funciones de orientación a objetos, tales como clases y herencia entre otras.
- Licencia Open Source: PHP tiene licencia de código abierto, es decir, puede ser descargado sin ningún costo, puede ser modificado y agregar más elementos por el usuario

### *2.4.2. CSS*

CSS (Cascading Style Sheets) el lenguaje de hojas de estilo utilizado para controlar la interfaz del HTML que sirve como interfaz hacia el usuario, divide el contenido y la presentación de una aplicación y es factible para crear aplicaciones web fáciles de usar.

Las hojas de estilo se crearon detrás del lenguaje de etiqueta SGML en 1970. El World Wide Web Consortium (W3C), encargado de establecer los estándares y modelos dentro de la Web, propuso crear un lenguaje que se encargue de todo lo referente a los estilos y diseños para HTML, en 1995 el W3C decidió usar CSS en HTML.

Características de CSS

- Optimización de tiempo: Puede controlar el diseño de varias páginas a la vez con un mismo formato
- Estructura: Los estilos se pueden manejar y almacenar en un archivo con extensión .css.
- Limpieza del código fuente: Al separar la hoja de estilo con el código fuente, agilita la búsqueda entre líneas de código y mejora el aspecto
- Flexibilidad: Se puede cambiar el diseño de la página web fácilmente debido a que solo se debe modificar la hoja de estilos para cambiar totalmente la fachada de la página web.

### *2.4.3. Lenguaje JavaScript*

El lenguaje de programación JavaScript es utilizado principalmente en los navegadores web para crear páginas web más dinámicas. (Mozilla Developer Network, 2017).
JavaScript permite a los desarrolladores administrar todo el contenido de la página a través del DOM, administrar datos con AJAX e IndexedDB, instaurar gráficos con canvas

e interactuar con el dispositivo a través de un navegador con varias API y así sucesivamente. (Mozilla Developer Network, 2017)

JavaScript es uno de los esquemas más seguidos de la web ya que, al ser un lenguaje interpretado, no requiere ser compilado dentro del servidor en el que se encuentre alojado. Como es un lenguaje muy utilizado en desarrollo web, existe una gran cantidad de frameworks que facilitan el desarrollo y añaden más funcionalidades al lenguaje, como, por ejemplo: JQuery, AngularJS.

### *2.4.4. Framework*

Es un conjunto de herramientas, bibliotecas, convenciones y mejores prácticas para incorporar tareas iterativas en módulos genéricos que se pueden reutilizar para facilitar el desarrollo de una aplicación web.

La arquitectura que se utiliza en casi todos los frameworks se llama MVC (Modelo, Vista, Controlador), esto divide a todo el desarrollo en 3 partes:

- Modelo: Es la lógica del negocio, su rol principal es conseguir la información que requiere la aplicación, además es el nexo para establecer el enlace con la base de datos.
- Vista: Es el Front End del sistema a desarrollar, en esta, el usuario puede manejar la información, también se realiza la transformación del modelo en la aplicación web que sea amigable y cumpla las necesidades del usuario.
- Controlador: Es el encargado de procesar todas las peticiones del usuario desde la interface y realizar los cambios pertinentes ya sea en la vista o en el modelo, es el canal de comunicación entre el modelo y la vista.

Utilizar framework en el desarrollo de aplicaciones es importante porque ayuda a facilitar ciertas cosas:

- Proporciona código más entendible.
- Permite automatizar tareas que son comunes en varias aplicaciones como los CRUD (create, read, update, delete).
- Facilita mucho la programación porque involucra operaciones complejas en instrucciones simples.

## 2.5. Entorno de trabajo

### *2.5.1. Framework Laravel*

El framework Laravel, en su versión número 5, posee la simplicidad, el poder y la interfaz refinada y jovial que lo convierten en uno de los entornos de código abierto más fáciles para PHP.

Se creó en 2011 y tiene varias bases de conceptos en entornos como ASP.NET MVC, Ruby. (Baquero García, 2015).

Laravel tiene un propósito fundamental, convertirse en la base para el uso de sintaxis expresiva y minuciosa para la generación de código simple, previniendo el llamado "código spaghetti" y proporcionando muchas características. Posee todos los beneficios de otros entornos y usa las características de las últimas versiones de PHP.

La mayoría de sus estructuras consisten en dependencias, tal y como lo maneja Symfony, es decir, Laravel está inmerso y es dependiente del desarrollo de sus dependencias.

#### Características de Laravel

- Blade: está formado de plantillas para crear vistas en Laravel, que permite desarrollar los modelos creados y las secciones de otras vistas donde las variables también están disponibles. También se puede usar el código PHP asociado con el uso de bootstrap u otro famework HTML conocido que generará resultados mejorados para diferentes dispositivos (teléfonos móviles, tabletas, PC, etc.).
- Eloquent: Laravel incluye el Eloquent que es el ORM que utiliza el mismo, que incluye para la gestión simple y fácil de las bases de datos y sus procesos relacionados en proyectos. Convierte las consultas SQL y las implementa en el Modelo Vista Controlador (MVC) la cual se encarga de procesar directamente las consultas SQL, en ese mismo sentido, se protege de ataques como SQL Inyection.
- Routing: Laravel facilita un sistema de planificación y gestión de rutas, el cual se encarga de manejar y controlar todas las rutas que posea el sistema.
- Middlewares: Estos son tipos de controladores que se forman antes y después de enviar una solicitud al servidor, es decir, está en la capacidad de ingresar varias verificaciones, inspecciones o procesos en el flujo de la aplicación.

### *2.5.2. Gestor de base de datos MySQL*

MySQL es un sistema de gestión de bases de datos relacionales de código abierto que está basado en el lenguaje de consulta estructurado (SQL) que se ejecuta en cualquier plataforma que maneje datos como Linux, Windows, Unix.

Aunque puede usarse en muchas aplicaciones diferentes, MySQL está más estrechamente relacionado con las aplicaciones web y la publicación como tal, este es un componente de suma importancia para la pila empresarial de código libre denominada LAMP.

#### Características de MySQL

- Multiplataforma: Es decir que consta de un extenso catálogo de sistemas que son compatibles y lo soportan como Windows, Linux, Solaris.
- Extensos mecanismos de almacenamiento que se adecúan a las diferentes necesidades de cada entorno: InnoDB, MyIsam, MariaDB etc. En efecto, se puede utilizar varios mecanismos para el almacenamiento de cada tabla.
- Las transacciones que se realizan se ejecutan con gran rapidez.
- Cualquier tipo de datos es aceptado por el motor.
- La documentación fundamental es proporcionada por la amplia comunidad de desarrolladores.
- Se puede realizar transacciones y operaciones de manera integral y relacional.
- Posee la capacidad para indexar, buscar y utilizar el texto completo, que proporciona una serie de herramientas para la búsqueda compleja utilizando plantillas

## 2.6. Arquitectura

La arquitectura es una base importante para el desarrollo de un sistema porque muestra una estructura general con la que se va a trabajar, es decir, muestra cómo está diseñado el sistema tanto en la parte física como la parte lógica.

### *2.6.1. Tipos de Arquitectura*

#### *2.6.1.1. Cliente – Servidor*

Este tipo de arquitectura se basa principalmente en un cliente realiza peticiones hacia un programa (servidor) y este devuelve una respuesta. El cliente tiene acceso a la

información del lado del servidor de forma completa ya que existe una comunicación directa por parte de los dos lados.

Cliente: Es la función que desempeña el equipo que requiere servicios desde el servidor, pero que también logra ejecutar procesamientos locales, como mostrar páginas web, desplegar ventanas y crear correos electrónicos.

Servidor: Es el rol desempeñado por el equipo que proporciona una amplia gama de servicios para clientes, como administración de archivos, impresión, páginas web, dirección de correo electrónico, actualización de bases de datos y control de acceso.

La Ilustración 5 muestra claramente cómo se desempeña la arquitectura cliente servidor.

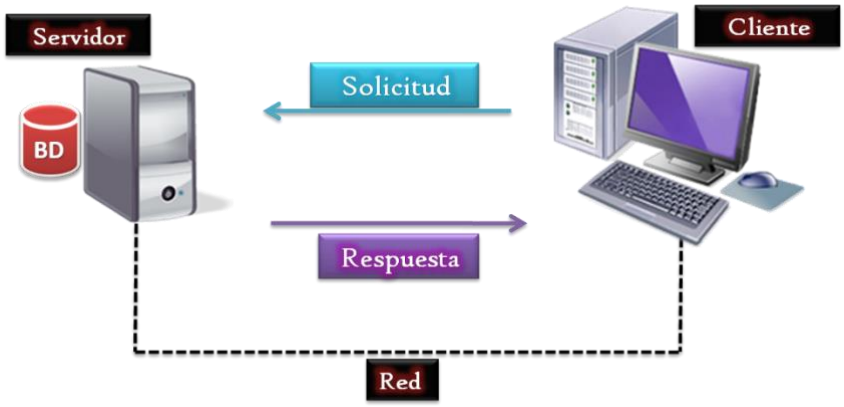



*Ilustración 5 - Arquitectura Cliente Servidor*

*Fuente: (Valdés, 2016)*

Ventajas

- Una de las principales ventajas que existe al utilizar esta arquitectura son los costos, existen cada vez plataformas de hardware a menor precio y esto abre camino a utilizarlas y no optar por las soluciones centralizadas
- Permite integrar varios sistemas diferentes y establecer una comunicación efectiva para compartir información, es decir, se puede formar un conjunto de computadoras con sistemas medianos o grandes sin la necesidad que estos posean el mismo sistema operativo.
- La infraestructura de red tiene la capacidad de reaccionar y adaptarse al momento de aumentar o disminuir el número de clientes en un período de tiempo.

- La estructura modular, por su naturaleza, también contribuye a la unificación de nuevas tecnologías y al desarrollo de la infraestructura de TI, lo que contribuye a la escalabilidad de las soluciones.

Desventajas

- El mantenimiento de los equipos y sistemas es mucho más complicado ya que se involucran diferentes partes tanto de hardware como software, esto dificulta un mantenimiento y diagnóstico de fallas.
- Las herramientas para la administración y ajuste de los sistemas, son muy escasas.
- Otra gran desventaja es que tanto el lado del cliente como el servidor deben manejar un mismo componente como sockets o RPC, esto es un gran problema ya que generalmente se manejan en diferentes plataformas.

#### *2.6.1.2. Tres Capas*

La arquitectura tres capas es una de las arquitecturas más usadas en los sistemas tradicionales, frecuentemente se utiliza cuando se implementan modelos de negocio como un Core financiero, tiendas virtuales, etc (Gómez, 2015). Es importante aclarar que no se recomienda usarlo en sistemas en tiempo real como se los utiliza en automóviles o en aviones.

Como su nombre lo indica, esta arquitectura consta de 3 capas, que son:

Capa de datos

En un modelo de base de datos previamente definido al que se emplean varias especificaciones de datos, como categorías, subcategorías, colecciones, productos, datos de productos, etc.

Este modelo se puede desarrollar mediante la exportación de datos contenidos en varias aplicaciones de base de datos, por ejemplo, Microsoft SQL Server.

El almacenamiento de datos se basa en XML para el posterior montaje adecuado de la capa de presentación y consta de uno o más administradores de bases de datos que efectúan todo el almacenamiento de datos, receptan solicitudes de almacenamiento o recuperación de la información de los datos desde el nivel de gestión.

Capa de lógica de negocio

En este nivel se establecen todas las reglas que deben ser respetadas en el comercio electrónico. Este nivel se comunica con la capa de presentación, acepta consultas y muestra resultados, así como con la capa de datos para pedir al administrador de la base de datos que guarde o recupere datos.

Capa de presentación

Finalmente, hay un tercer nivel, denominado Presentación, que representa todos los aspectos afines con la presentación y el diseño de una solución de comercio electrónico.

Este tercer nivel es completamente personalizable y definible, que se basa en una sucesión de plantillas que definen cómo se presentarán los datos y la funcionalidad del comercio electrónico y cómo interactúan con el usuario final.

Las tres capas de la arquitectura se muestran en la Ilustración 6, se nota claramente la diferenciación entre las tres

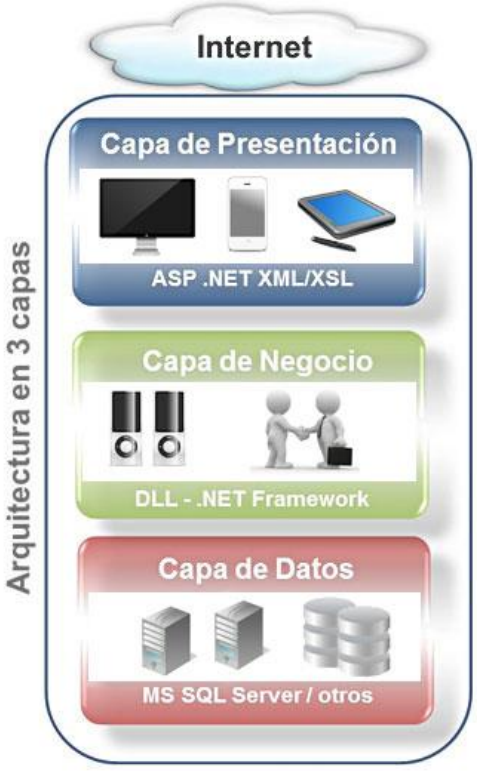



*Ilustración 6 - Arquitectura tres capas*

*Fuente: (Modelo de Tres Capas, 2013)*

Ventajas

- Una de las principales ventajas en esta arquitectura es poder separarse en varios ordenadores, es decir, si aumenta la complejidad del sistema, el tamaño de base de datos o la parte de la lógica del negocio.
- Se maneja de mejor manera la estandarización dentro de las 3 capas, este es un factor clave dentro de empresas de desarrollo.
- Se puede reutilizar una capa si fuese necesario, al tener la lógica, los datos y la presentación separadas, se puede ocupar cualquiera de estas capas en otros módulos y así se mejora la rapidez con la que se desarrolla.
- Si se necesita modificar algún elemento de cualquier capa, solo se dirige a la ubicación del elemento sin afectar a las demás capas de los demás módulos.

Desventajas

- Existe pérdida de eficiencia al realizar pruebas de carga y stress al momento de simular un envío de solicitudes por parte de clientes externos, genera tráfico en la red y el servidor deja de responder
- Existen dificultades al realizar el diseño correcto de la granularidad de las capas, es decir, no se logra un nivel de detalle correcto.
- Consumo excesivo de espacio de memoria en la aplicación debido a la extensión de las capas, esto sucede cuando no existe un balance entre el número de capas y subcapas que componen el programa

### 2.6.2. Selección de la Arquitectura

La arquitectura que se seleccionará para la realización de la aplicación será el modelo en tres capas, esta arquitectura ya ha sido utilizada en proyectos anteriores y se tiene un conocimiento previo, además en la presente disertación, no existe el riesgo de crear más capas extra y si en algún caso, se viera la necesidad de realizar algún cambio en las capas, no afectaría en lo absoluto a las demás.

## 2.7. Patrón de Diseño

### 2.7.1. Modelo Vista Controlador (MVC)

El Modelo Vista Controlador (MVC) es una propuesta de diseño de software que, mediante estos componentes, Vistas Modelos y Controladores, clasifica y separa la capa lógica de la vista de la aplicación hacia el usuario.

MVC o Model-View-Controller es un modelo de desarrollo de software que, manejando 3 componentes (vistas, controladores y modelos), en la cual se encuentran separados la lógica de la capa de presentación dentro de la aplicación hacia el usuario.

En la Ilustración 7 están definidos los tres componentes de la arquitectura, como se comunican entre ellos y el proceso que sigue.

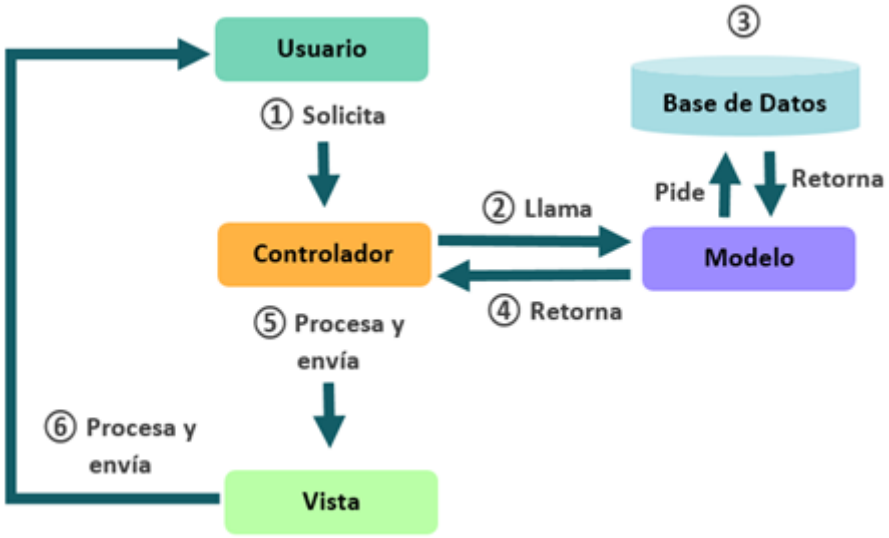



Ilustración 7 - Modelo Vista Controlador

Fuente: (Gómez R. , 2015)

Modelo: Es la lógica del negocio, se encarga de obtener la información correspondiente para que la aplicación se ejecute de manera correcta, para obtener esta información, se realiza la conexión entre el modelo y la base de datos.
Vista: Es el Front - End de la aplicación, en esta, el usuario logra manejar la información, también se realiza la transformación del modelo en la aplicación web que sea amigable y cumpla las necesidades del usuario.

Controlador: Es el encargado de procesar todas las peticiones del usuario desde la interface y realizar los cambios pertinentes ya sea en la vista o en el modelo, es el canal de comunicación entre el modelo y la vista.

# CAPITULO 3: EVALUACIÓN ENTRE LAS METODOLOGÍAS DE DESARROLLO DE SOFTWARE XP – RUP

En este capítulo se va a describir las metodologías a evaluar, además se va a indicar los criterios de usabilidad y eficiencia con sus correspondientes indicadores, los mismos que se utilizaron para la investigación y análisis para continuar con la respectiva evaluación entre las metodologías, de las cuales se determinan la valoración cuantitativa y cualitativa para la calificación correspondiente y optar por la metodología que se ajusta para el desarrollo de la presente disertación.

Del estudio realizado se pudo seleccionar a la metodología más adecuada y más adaptable para el presente trabajo, siendo XP (eXtreme Programming) la metodología que mayor puntaje obtuvo en la evaluación. Con esto se concluye que es la más adecuada para el desarrollo del proyecto en la web en cuanto a usabilidad y eficiencia.

## 3.1. Metodologías a describir

Metodologías Orientadas a objetos

La metodología orientada a objetos, al igual que otras orientadas al diseño, permite recrear el problema interpretado del mundo real y lo transforma en una solución dentro del software.

Esta metodología se construye sobre tres importantes conceptos fundamentales para el diseño de software, las cuales son:

Encapsulación

La encapsulación es la manera de agrupar bajo una misma entidad a datos, funciones o métodos

“Según este criterio, al agrupar las funciones que trabajan con una serie de datos en la misma entidad que estos, se consigue independizar la implementación interna de la misma, de manera que posibles cambios en su interior afectan mínimamente a sus usuarios” (Gómez J. , 2001).

Al trabajar de esta manera, se logra optimizar tiempos empleados en desarrollo dentro de un sistema.

Herencia

“Consiste en un mecanismo para expresar la similitud entre clases, se utiliza en la construcción de nuevas clases a partir de otras ya existentes, de manera que las nuevas

clases así creadas incorporan la estructura y el comportamiento de la clase previamente heredada." (Gómez J. , 2001)

"Cuando una clase hereda de otra u otras, se dice que es subclase o clase derivada, de la principal y que, a su vez, se considera superclase o clase base." (Gómez J. , 2001)

La herencia brinda muchas facilidades al momento de desarrollar, como, por ejemplo:

- Reusabilidad.
- Código compartido.
- Consistencia en la interfaz.
- Posibilidad de construir y reconstruir Software.
- Prototipado rápido.
- Polimorfismo.
- Ocultación de información.

A la misma vez, tiene una serie de desventajas, las cuales son:

- La velocidad de ejecución es menor.
- El tamaño de los sistemas es mayor
- Sobrecarga debida al paso de mensajes.
- Aumento de la complejidad de los programas.

(Gómez J. , 2001)

Polimorfismo

El término proviene del griego polimorphos (poly = muchas, morphos = formas).

"Es una propiedad que permite que un método tenga numerosas implementaciones, que se seleccionan en base al tipo de objeto indicado al solicitar la ejecución del método" (Carrillo Ledesma, s.f.)

En la siguiente tabla se indican cuáles son las principales metodologías orientadas a objetos:

*Tabla 1 Características de metodologías orientada a objeto*

| Metodología | Características |
|---|---|
| OMT (Object Modelling Technique) | Fue una de las primeras metodologías Orientadas a Objetos.<br>Fue presentado por Rumbaugh en 1991<br>Utiliza tres modelos diferentes que se combinan de una manera similar a las |

| | |
|---|---|
| | metodologías estructuradas más antiguas.<br>El objetivo del análisis es construir un modelo del mundo |
| Object Process Methodology (OPM) | Se introdujo por primera vez en 1995.<br>Tiene solo un diagrama de proceso de objetos (OPD) para modelar la estructura, función y comportamiento del sistema.<br>Tiene un fuerte énfasis en el modelado, pero no se enfoca mucho en el proceso. |
| Rational Unified Process (RUP) | Fue desarrollado en Rational Corporation en 1998 por las mismas personas que desarrollaron UML.<br>Aunque se dice que tiene un 'ciclo de vida', RUP tiene un enfoque evolutivo / iterativo en lugar del enfoque lineal / en cascada de metodologías anteriores. |

*Autor: David Alejandro Aráuz Moya, 2018*

**Metodologías Ágiles**

Las metodologías ágiles se centran básicamente en diseñar principios y valores que permitan realizar cambios y sobre todo responder rápidamente a todo lo que pueda surgir dentro de un sistema al momento de desarrollar, todo esto, para ir mejorando constantemente el proceso.

Tras una reunión celebrada en Utah – Estados Unidos en febrero de 2001, surge el término “ágil” dentro del desarrollo de software. Una vez concluida la reunión se fundó The Agile Alliance, que se basa en fomentar los conceptos que tengan que ver con el desarrollo ágil de software y a su vez, se encargan de ayudar a otras empresas para que obtengan estos conceptos. El inicio de todo fue el Manifiesto Ágil, un documento el cual comprende toda la filosofía “ágil”.

Se basa en fomentar los conceptos que tengan que ver con el desarrollo ágil de software y a su vez, se encargan de ayudar a otras empresas para que obtengan estos conceptos

**Manifesto Ágil**

El manifesto Ágil señala lo siguiente:

- Equipos de interacción y desarrollo humano en relación al proceso y herramientas.
- El principal componente de triunfo de un proyecto de software son las personas.
- La creación de un buen equipo de trabajo es mucho más significativa que crear un buen entorno
- A menudo, el error es crear primero el entorno y esperar a que el equipo se ajuste instintivamente.
- Es más conveniente crear un equipo y configurar su propio entorno de desarrollo para satisfacer sus necesidades.
- Implementar un software que funcione más que una buena documentación: la regla es: "no cree documentos si no son necesarios de inmediato para una decisión significativa".
- Estos documentos deben ser concisos y centrados en lo importante
- La cooperación con el cliente es más que negociaciones de contrato: se plantea una reunión continua entre el cliente y el equipo de desarrollo, esta cooperación marcará el progreso del proyecto y garantizará el éxito.
- Afrontar los cambios, no seguir estrictamente el plan: El éxito o el fracaso de un proyecto se puede medir en la capacidad de poder actuar frente a los cambios que se pueden suscitar a lo largo del proyecto como los cambios de requerimiento, cambios en la tecnología, dentro del equipo, etc.

*Tabla 2 Metodologías ágiles y sus características*

| Metodología | Características |
|---|---|
| Extreme Programming | Fue presentado en 1996 por Kent Beck. Los métodos utilizados, como la programación de pares, las pruebas unitarias y las pruebas de aceptación del cliente, ya existían anteriormente. Lo que XP hizo de manera diferente fue llevar estas mejores prácticas a "niveles extremos". Fue creado para tener en cuenta los cambios que están ocurriendo |

| | |
|---|---|
| | en ese momento en el desarrollo de software y la tecnología. |
| Dynamic Systems Development Method | Fue publicado en 1995 por DSDM Consortium.<br>El Consorcio desarrolló y promovió conjuntamente un marco de desarrollo independiente de herramientas y técnicas a partir de las experiencias de mejores prácticas de personas que trabajan en grandes empresas como British Airways, American.<br>Ahora se ha convertido en un marco de entrega de proyectos que es totalmente compatible con ISO 9000 y PRINCE2. |
| Crystal Methodologies | Crystal Methods es una familia de metodologías de desarrollo de software desarrolladas por Alistair Cockburn a partir de un estudio y entrevistas a equipos de desarrollo.<br>Los métodos están codificados por colores para indicar el riesgo para la vida humana. Por ejemplo, los proyectos que pueden implicar riesgos para la vida humana usarán Crystal Sapphire, mientras que los proyectos que no tengan tales riesgos utilizarán Crystal Clear. Crystal se enfoca en seis aspectos principales: personas, interacción, comunidad, comunicación, habilidades y talentos. |

Autor: (De Los Santos, 2017)

## 3.2. Análisis comparativo

Para el análisis comparativo de las metodologías se enumerarán los criterios con sus respectivos indicadores, en ese mismo sentido se realizará el cuadro comparativo de las metodologías RUP vs XP.

Los criterios de evaluación de cada metodología son detallados en la tabla presentada a continuación:

*Tabla 3 Criterios de evaluación*

| N° | Criterio | Concepto |
|---|---|---|
| **1** | Usabilidad | Evaluará que tan fácil es usar la metodología y realizar un seguimiento a la misma |
| **2** | Eficiencia | Evaluará que tan capaz es la metodología para cumplir una función optimizando recursos para la obtención de los resultados |

*Autor: David Alejandro Aráuz Moya, 2018*

## 3.3. Criterios de comparación

Para realizar el estudio comparativo de las metodologías de software a evaluar, se establecieron varios indicadores de acuerdo a cada criterio escogido anteriormente, donde cada uno va a ser analizado de acuerdo a su comportamiento para entregar la información específica.

Los indicadores, se han postulado como destrezas para la medida de los criterios; en muchos casos, estas destrezas resultan igual de genéricas que los mismos criterios, para tal caso se va a definir las métricas particulares a utilizar en el contexto de la evaluación y se precise de argumentos numéricos que permitan estimar una medida exacta.

Dentro de cada criterio se dispone de un conjunto de indicadores que, según resultados de lecturas, investigaciones y juicio propio, se han dispuesto como las más adecuadas para cada criterio.

En las tablas 3.3.1 y 3.3.2, se muestran los indicadores seleccionados tanto para la Usabilidad como para la Eficiencia.

### 3.3.1. Indicadores de usabilidad

*Tabla 4 Indicadores de usabilidad*

| Indicadores | Concepto |
|---|---|
| Claridad | Precisión en la información contenida dentro del documento |
| Capacidad | Habilidad y experiencia en el manejo y representación de la información |
| Simplicidad | Busca que los sistemas desarrollados bajo esta característica sean sencillos al estar bajo el proceso |
| Manejo de Usuarios | Se define cuáles son los usuarios y sus correspondientes funciones específicas, determina si los usuarios son aptos para la aplicación |
| Interacción con el Usuario | Manejan todo el aspecto relacionado con el usuario y el diseño y como se relacionan entre ellos. |
| Diseños | Mediante el uso de diagramas y gráficos describen a detalle la transparencia y el comportamiento de la aplicación |
| Navegabilidad e Interfaces | Simplifica la visualización mediante diagramas, con el fin de facilitar la navegación a través por parte del usuario |

*Autor: David Alejandro Aráuz Moya, 2018*

### 3.3.2. Indicadores de eficiencia

*Tabla 5 Indicadores de eficiencia*

| Indicadores | Concepto |
|---|---|
| Tiempo | Es el período en el que los desarrolladores realizan una tarea asignada |
| Uso de recursos | Evalúa los recursos adecuados en cada fase cuando se las ejecuta |
| Entregables | Cantidad de entregables en cada metodología |
| Validación | Evaluará el diseño correcto mediante pruebas asignadas a usuarios |

*Autor: David Alejandro Aráuz Moya, 2018*

## 3.4. Criterios de evaluación

Para la evaluación de las metodologías se basó en la calificación de cada criterio de acuerdo a la escala mostrada anteriormente, esto, permitió definir que metodología es la más adecuada para el presente trabajo de disertación.

### 3.4.1. Valoración cualitativa y cuantitativa

Como se puede apreciar en la siguiente tabla, se fijan los valores cualitativos y cuantitativos y se simbolizan en intervalos de porcentajes.

*Tabla 6 Valoración cualitativa y cuantitativa*

| Mala | Regular | Buena | Excelente |
|---|---|---|---|
| < 50% | >= 50% y < 70% | >= 70% y <90% | >= 90% |

*Autor: David Alejandro Aráuz Moya, 2018*

### 3.4.2. Escala de valoración para los criterios

Para cada indicador, se realizará la evaluación pertinente, los mismos que estarán en un rango de 1 hasta 4, tal y como se indica en la siguiente tabla.

*Tabla 7 Escala de valoración para los criterios*

| Valor Cualitativo | | Valor Representativo |
|---|---|---|
| Insuficiente | Insatisfactorio | 1 |
| Suficiente | Poco Satisfactorio | 2 |
| Bueno | Satisfactorio | 3 |
| Excelente | Muy Satisfactorio | 4 |

*Autor: David Alejandro Aráuz Moya, 2018*

## 3.5. Análisis de los criterios de comparación

### 3.5.1. Usabilidad

Mide la facilidad de estudio de las metodologías mediante los criterios anteriormente seleccionados, de esta forma se podrá evaluar al proyecto en las etapas iniciales del desarrollo.

#### *3.5.1.1. Claridad*

Mide la precisión de la información acerca de los pasos a seguir en la aplicación de cada fase de las metodologías a comparar, esto se puede ver en la siguiente tabla.

*Tabla 8 Criterios de comparación - Claridad*

| **Valoración** | |
|---|---|
| **Pasos** | **Valor Cualitativo** |
| Sin Detalles | Insatisfactorio |
| Poco Detallado | Poco Satisfactorio |
| Fase Detallada | Satisfactorio |
| Fase y Subfase Detallada | Muy Satisfactorio |

*Autor: David Alejandro Aráuz Moya, 2018*

#### *3.5.1.2. Capacidad*

Mide el manejo, procesamiento, estructuración y presentación de la información que se tiene al final de cada fase, tal como se muestra en la siguiente Tabla.

*Tabla 9 Criterios de comparación - Capacidad*

| **Valoración** | |
|---|---|
| **Presentación de la Información** | **Valor Cualitativo** |
| No presenta diagramas | Insatisfactorio |
| No se especifican | Poco Satisfactorio |
| Diagramas básicos | Satisfactorio |
| Diagramas básicos y propios | Muy Satisfactorio |

*Autor: David Alejandro Aráuz Moya, 2018*

#### *3.5.1.3. Simplicidad*

Mide si la metodología promueve el desarrollo de sistemas que sean simples y sencillos al momento de realizar una retroalimentación del proceso, tal como se muestra en la siguiente Tabla.

*Tabla 10 Criterios de comparación - Valoración*

| Valoración | |
|---|---|
| **Enfoque** | **Valor Cualitativo** |
| Sin retroalimentación | Insatisfactorio |
| Fases Secuenciales | Poco Satisfactorio |
| Fases incrementales | Satisfactorio |
| Fases iterativas | Muy Satisfactorio |

*Autor: David Alejandro Aráuz Moya, 2018*

#### *3.5.1.4. Manejo de usuarios*

Este indicador se mide de acuerdo a la cantidad de usuarios con sus respectivas tareas específicas, tal como se muestra en la siguiente Tabla.

*Tabla 11 Criterios de comparación - Manejo de usuarios*

| Valoración | |
|---|---|
| **Definición de Usuarios** | **Valor Cualitativo** |
| No define usuarios | Insatisfactorio |
| Define usuarios generales | Poco Satisfactorio |
| Define usuarios en cada proceso | Satisfactorio |
| Define usuarios y roles de cada uno | Muy Satisfactorio |

*Autor: David Alejandro Aráuz Moya, 2018*

#### *3.5.1.5. Interacción con el usuario*

Mide las actividades relacionados con el diseño enfocado en el usuario y la relación entre las fases, tal como se muestra en la siguiente Tabla.

*Tabla 12 Criterios de comparación - Interacción con el usuario*

| Valoración | |
|---|---|
| **Usuarios** | **Valor Cualitativo** |
| Una sola fase | Insatisfactorio |
| Más de una fase | Poco Satisfactorio |
| Todas las fases de la metodología | Satisfactorio |
| Todas las fases y pruebas de la metodología | Muy Satisfactorio |

#### *3.5.1.6. Diseños*

Se mide conforme al uso de diagramas para poder expresar el comportamiento y las relaciones con la certeza de que no exista ninguna adulteración, tal como se muestra en la siguiente Tabla.

*Tabla 13 Criterios de comparación - Diseños*

| Valoración | |
|---|---|
| **Número de Diagramas** | **Valor Cualitativo** |
| 2 | Insatisfactorio |
| 3 | Poco Satisfactorio |
| 4 | Satisfactorio |
| Más de 5 y propios | Muy Satisfactorio |

*Autor: David Alejandro Aráuz Moya, 2018*

#### *3.5.1.7. Navegabilidad e interfaces*

Se mide conforme a la facilidad que tienen los usuarios al momento de navegar dentro del sistema, tal como se muestra en la siguiente Tabla.

*Tabla 14 Criterios de comparación - Navegabilidad e interfaces*

| Valoración | |
|---|---|
| **Usuarios** | **Valor Cualitativo** |
| No tiene Diagramas | Insatisfactorio |
| Tiene diagramas de navegación | Poco Satisfactorio |
| Diagramas de navegación e interfaces básicas | Satisfactorio |
| Diagramas de navegación y uso de plantillas | Muy Satisfactorio |

*Autor: David Alejandro Aráuz Moya, 2018*

### 3.5.2. Calificaciones del indicador usabilidad

Las calificaciones respectivas al indicador de usabilidad se muestran en la siguiente Tabla.

*Tabla 15 Calificaciones del indicador Usabilidad*

| INDICADORES | XP | | | RUP | | |
|---|---|---|---|---|---|---|
| | Valor Cualitativo | Valor Obtenido / 4 | Valor en Porcentaje | Valor Cualitativo | Valor Obtenido / 4 | Valor en Porcentaje |
| **Claridad** | Satisfactorio | 3 | 10,71% | Muy Satisfactorio | 4 | 14,29% |
| **Capacidad** | Muy Satisfactorio | 4 | 14,29% | Satisfactorio | 3 | 10,71% |
| **Simplicidad** | Muy Satisfactorio | 4 | 14,29% | Poco Satisfactorio | 2 | 7,14% |
| **Manejo de Usuarios** | Satisfactorio | 3 | 10,71% | Muy Satisfactorio | 4 | 14,29% |
| **Interacción con el Usuario** | Muy Satisfactorio | 4 | 14,29% | Poco Satisfactorio | 2 | 7,14% |
| **Diseños** | Satisfactorio | 3 | 10,71% | Muy Satisfactorio | 4 | 14,29% |
| **Navegabilidad e Interfaces** | Muy Satisfactorio | 4 | 14,29% | Muy Satisfactorio | 4 | 14,29% |
| **Totales** | | 25 | 89,29% | | 23 | 82,14% |

*Autor: David Alejandro Aráuz Moya, 2018*

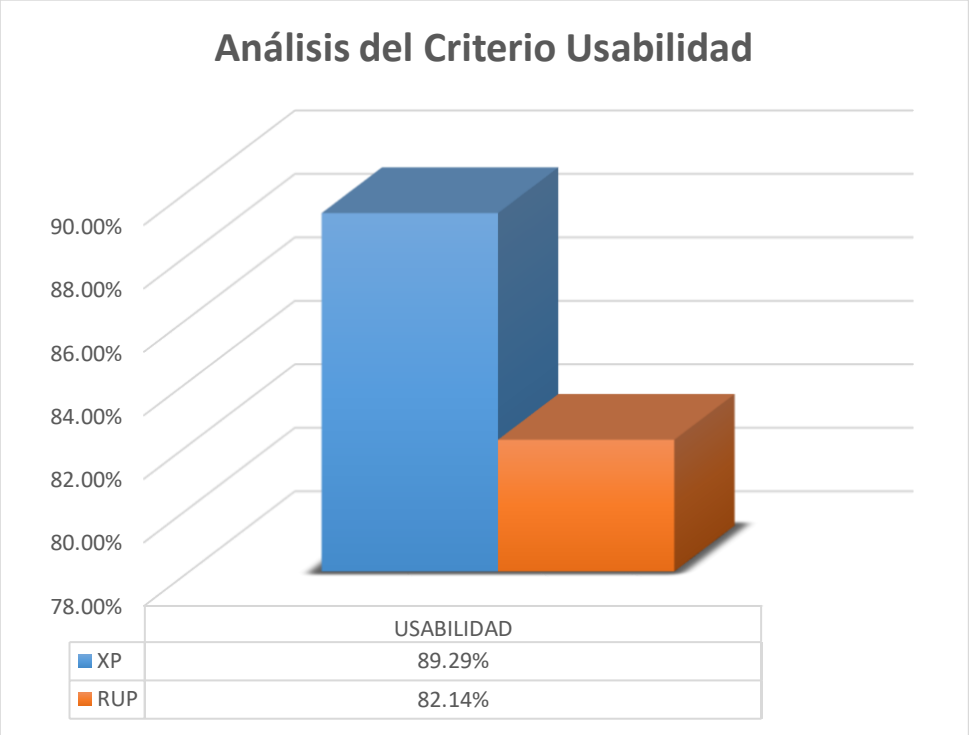



*Ilustración 8 Análisis del criterio Usabilidad*

*Fuente: David Alejandro Aráuz Moya*

## 3.5.3. Interpretación de resultados

### *3.5.3.1. Claridad*

En este indicador se evaluó que la información en cada fase y subfase de las metodologías a ser estudiadas, sea detallada, entendible y precisa, en cuanto a la

puntuación de las metodologías se tiene que RUP obtiene una calificación de 4 puntos que equivale a muy satisfactorio puesto que en cada fase se detalla el trabajo a realizar y tiene muy bien definidas las fases, mientras que XP alcanza una puntuación de 3 equivalente a satisfactorio ya que al ser una metodología ágil, indica que hacer en cada fase pero no como se las debe realizar.

#### *3.5.3.2. Capacidad*

En este indicador se evaluó el manejo y presentación de la información de los procesos que tendrá el sistema a través de la representación de diagramas, en este indicador, la metodología XP alcanza un puntaje de 4 puntos equivalente a muy satisfactorio dado que presenta las características antes mencionadas, mientras que RUP presenta como calificación 3 equivalente a satisfactorio, puesto que tiene una fase específica para la presentación de diagramas mientras que en otras no tiene.

#### *3.5.3.3. Simplicidad*

En este indicador se evaluó que las fases de la metodología promuevan el desarrollo de sistemas simples e iterativos al realizar el proceso, con el fin de agilizar el desarrollo del sistema. Aquí, la metodología XP obtiene una calificación de 4 correspondiente a muy satisfactorio, por el hecho de que combina las mejores prácticas de desarrollo y las lleva al extremo logrando simplificar el proyecto como tal. Mientras que la metodología RUP tiene fases previas al momento de realizar una iteración completa y al final de cada iteración ejecuta un producto entregable.

#### *3.5.3.4. Manejo de usuarios*

En este indicador se evaluó la cantidad de usuarios que tienen asignados tareas para cada uno de sus perfiles, XP obtuvo una calificación de 3 que significa satisfactorio ya que no cuentan con muchos roles definidos dentro de su metodología de desarrollo, al ser una metodología ágil, se enfocan en el rendimiento. La metodología RUP tiene como puntaje 4 correspondiendo a muy satisfactorio debido a que cuentan con roles especificados para cada fase del proyecto.

#### *3.5.3.5. Interacción con el usuario*

En este indicador se evaluó al usuario y la relación que tiene con el diseño y comunicación en cada una de las fases, aquí la metodología XP obtuvo un total de 4 equivalente a muy satisfactorio debido a que en cada fase cuentan con evaluaciones de

las mismas, mientras que RUP obtuvo un puntaje de 3, puesto que tienen una fase específica para realizar lo indicado.

#### *3.5.3.6. Diseños*

En este indicador se evaluó la cantidad de diagramas usados poder expresar el comportamiento y las relaciones de la información en cada fase de las metodologías, la metodología RUP alcanzó un puntaje de 4 correspondiente a muy satisfactorio, al tener toda una fase de diseño, esta metodología se centra mucho en esto, mientras que la metodología XP consiguió el puntaje de 3 puesto que no cuenta con todos los diagramas y algunos de estos están definidos por el usuario.

#### *3.5.3.7. Navegabilidad e interfaces*

En este indicador se evaluó mediante diagramas de navegabilidad que permitieron asociar los elementos a la interfaz, con el fin de facilitar la visualización que tendrán los usuarios finales, las dos metodologías obtuvieron puntajes de 4 que corresponden a muy satisfactorio, puesto que cumplen con las características previamente definidos y además las dos metodologías trabajan desde un prototipo de diseño en el cual se define la navegabilidad del sistema.

### 3.5.4. Eficiencia

Mide la utilización óptima de los recursos que dispone las metodologías, de esta forma se podrá obtener los resultados esperados.

#### *3.5.4.1. Tiempo*

Mide según el tiempo que toma a los desarrolladores efectuar las tareas, tal como se muestra en la siguiente Tabla.

*Tabla 16 Criterios de comparación - Valoración*

| **Valoración** | |
|---|---|
| **Tiempo** | **Valor Cualitativo** |
| Más de 16 semanas | Insatisfactorio |
| De 15 a 12 semanas | Poco Satisfactorio |
| De 11 a 8 semanas | Satisfactorio |
| Menos de 7 semanas | Muy Satisfactorio |

*Autor: David Alejandro Aráuz Moya, 2018*

#### *3.5.4.2. Uso de recursos*

Mide de acuerdo a la utilización de recursos apropiados en las fases que se los lleva a cabo, tal como se muestra en la siguiente Tabla.

*Tabla 17 Criterios de comparación - Uso de recursos*

| Valoración | |
|---|---|
| **Tiempo** | **Valor Cualitativo** |
| Índice de recursos altos | Insatisfactorio |
| Índice de recursos intermedios | Poco Satisfactorio |
| Índice de recursos escasos | Satisfactorio |
| Índice de recursos bajos | Muy Satisfactorio |

*Autor: David Alejandro Aráuz Moya, 2018*

#### 3.5.4.3. *Entregables*

Mide la cantidad de entregables realizados por cada metodología en las fases correspondientes, tal como se muestra en la siguiente Tabla.

*Tabla 18 Criterios de comparación - Entregables*

| Valoración | |
|---|---|
| **Entregables** | **Valor Cualitativo** |
| Menos de 2 | Insatisfactorio |
| 3 | Poco Satisfactorio |
| 4 | Satisfactorio |
| Más de 5 | Muy Satisfactorio |

*Autor: David Alejandro Aráuz Moya, 2018*

#### *3.5.4.4. Validación*

Mide las validaciones relacionadas con el diseño enfocadas en el usuario, tal como se muestra en la siguiente Tabla.

*Tabla 19 Criterios de comparación - Validación*

| Valoración | |
|---|---|
| **Tiempo** | **Valor Cualitativo** |
| Ninguna fase | Insatisfactorio |
| En una sola fase | Poco Satisfactorio |
| Más de una fase | Satisfactorio |
| Presente en todas las fases | Muy Satisfactorio |

*Autor: David Alejandro Aráuz Moya, 2018*

### 3.5.5. Calificaciones del indicador eficiencia

Las calificaciones respectivas al indicador de Eficiencia se muestran en la siguiente Tabla.

*Tabla 20 Calificaciones del indicador Eficiencia*

| INDICADORES | XP | | | RUP | | |
|---|---|---|---|---|---|---|
| | Valor Cualitativo | Valor Obtenido / 4 | Valor en Porcentaje | Valor Cualitativo | Valor Obtenido / 4 | Valor en Porcentaje |
| **Tiempo** | Muy Satisfactorio | 4 | 25,00% | Poco Satisfactorio | 2 | 12,50% |
| **Uso de recursos** | Satisfactorio | 3 | 18,75% | Satisfactorio | 3 | 18,75% |
| **Entregables** | Muy Satisfactorio | 4 | 25,00% | Poco Satisfactorio | 2 | 12,50% |
| **Validación** | Satisfactorio | 3 | 18,75% | Satisfactorio | 3 | 18,75% |
| **Totales** | | 14 | 87,50% | | 10 | 62,50% |

*Autor: David Alejandro Aráuz Moya, 2018*

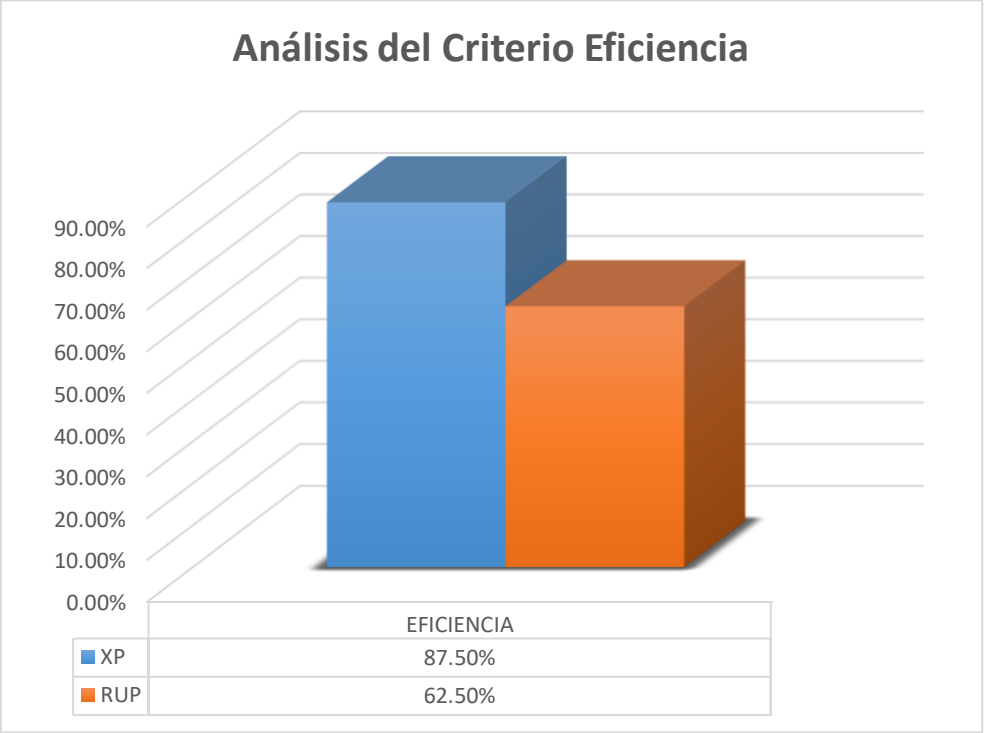



*Ilustración 9 Análisis del criterio de Eficiencia*

*Fuente: David Alejandro Aráuz Moya*

### 3.5.6. Interpretación de resultados

#### *3.5.6.1. Tiempo*

En este indicador se evaluó el tiempo que toma a los desarrolladores cumplir las tareas hasta conseguir el producto final del sistema en cada una de las metodologías, para lo cual la metodología XP obtuvo un puntaje de 4 que representa muy satisfactorio debido a que al ser una metodología ágil, se trabaja con aplicaciones pequeñas y los tiempos se reducen al extremo tratando de entregar el proyecto en el menor tiempo posible, mientras tanto la metodología RUP obtuvo una puntuación de 2 equivalente a poco satisfactorio porque esta metodología ayuda a proyectos de gran alcance que sobrepasan las 16 semanas.

#### *3.5.6.2. Uso de recursos*

En este indicador se evaluó la utilización de recursos apropiados en el desarrollo del sistema en las metodologías correspondientes, tanto en la metodología XP como RUP obtuvieron un puntaje similar, de 3 siendo equivalente a satisfactorio, esto quiere decir que no se necesitaron de recursos adicionales al entrar a fase de desarrollo, estos recursos son: humanos, económicos y materiales.

#### *3.5.6.3. Entregables*

En este indicador se evaluó los documentos entregables realizados en cada fase, como las dos metodologías cuentan con fases específicamente para la realización de documentos, se tomarán los entregables en cada fase. La metodología XP obtuvo un puntaje de 4, equivalente a muy satisfactorio, cumpliendo con las características antes mencionadas, ya que al final de cada fase se genera un entregable con todas las pruebas debidas del caso, sin embargo, la metodología RUP obtuvo una puntuación de 2 ya que al final de cada fase no se genera un entregable.

#### *3.5.6.4. Validación*

En este indicador se evaluó a los usuarios finales en cada fase del sistema con herramientas de diseño enfocadas en el usuario, en este punto, las dos metodologías XP y RUP obtuvieron el mismo resultado, siendo 3 la puntuación equivalente a satisfactorio, porque las dos metodologías presentan validaciones en la mayoría de sus fases, estas se las realiza de forma implícita, pero no siempre se las realiza, está tomada como buena práctica del programador

### 3.5.7. Cuadro comparativo

En este cuadro se unifica de manera general los resultados obtenidos anteriormente por parte de las metodologías: eXtreme Programming (Programación Extrema) y Rational Unified Process (Proceso Unificado de Rational). Los resultados que se exponen en la Tabla 21 son los indicadores de Usabilidad y Eficiencia.

*Tabla 21 Cuadro Comparativo*

| RESULTADOS | | | | |
|---|---|---|---|---|
| | XP | | RUP | |
| | Valor Total Obtenido | Valor en Porcentaje | Valor Total Obtenido | Valor en Porcentaje |
| **USABILIDAD** | 25 | 89,29% | 23 | 82,14% |
| **EFICIENCIA** | 14 | 87,50% | 10 | 62,50% |

*Autor: David Alejandro Aráuz Moya, 2018*

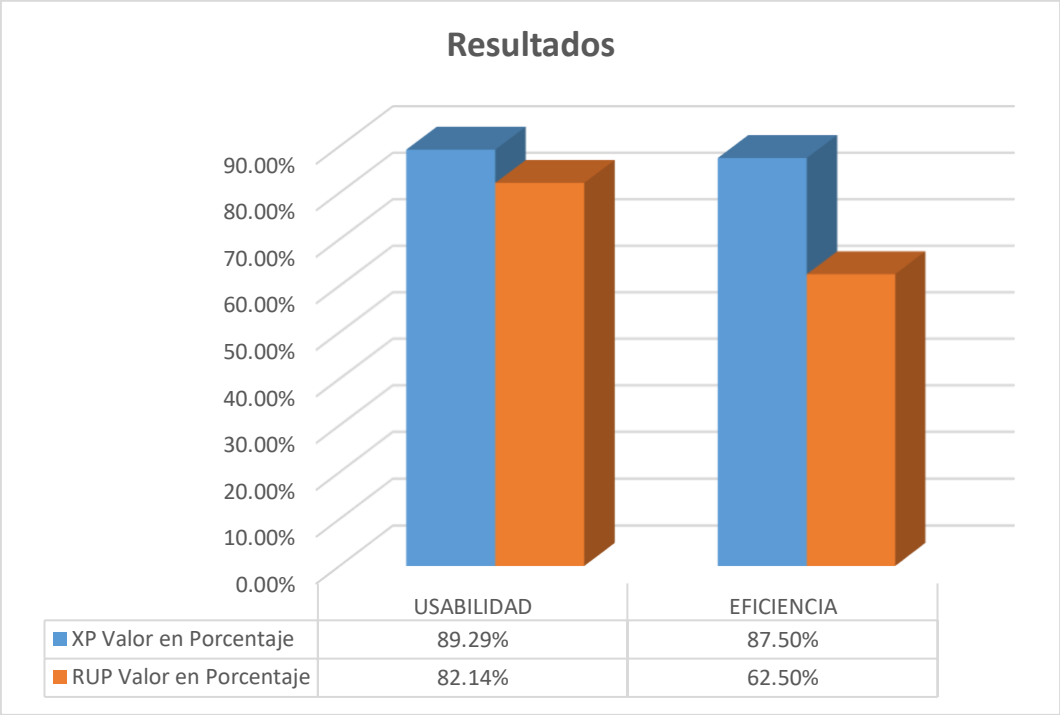



*Ilustración 10 Resultados*

*Fuente: David Alejandro Aráuz Moya*

### 3.5.8. Interpretación de resultados finales

Los criterios de Usabilidad y Eficiencia fueron medidos en las metodologías de desarrollo eXtreme Programming (Programación Extrema) y Rational Unified Process (Proceso Unificado de Rational) y están descritas en la Tabla 3.

Como se puede observar para el criterio de Usabilidad, la metodología XP obtuvo un 89,29 % calificándola como Buena según la Tabla 15 definida en los criterios de evaluación, mientras que la metodología RUP consiguió una puntuación de 82,14 % obteniendo la valoración Buena de la Tabla 15, es evidente entonces que la metodología XP destaca en 7,14 % de RUP, esto se debe a que supera en ciertos indicadores como la Capacidad, la simplicidad y la Interacción que tiene con el usuario, que son fundamentales a la hora de desarrollar software como se puedo constatar en el análisis comparativo.

Con respecto a las calificaciones obtenidas por el criterio de Eficiencia, se observa claramente que la metodología XP obtuvo 87,50 % considerándose Buena según la Tabla 20, por otra parte, la metodología RUP alcanzó el 62,50 % calificándola como Regular basada en la Tabla 20, dejando como resultado una diferencia de 25 % a favor de eXtreme Programming (XP), de los planteamientos anteriores se deduce que XP está por delante que RUP en cuanto a Eficiencia dado que sobresale más en ciertos aspectos como el tiempo de desarrollo y la entregables por cada fase.

De acuerdo con los razonamientos que se han venido realizando, la metodología XP al ser una metodología ágil permite una programación mucho más organizada, es más eficiente en los procesos de planificación y pruebas, su tasa de errores es muy pequeña, facilita los cambios, es escalable y flexible, además es una de las metodologías más utilizadas en la implementación de nuevas tecnologías, y de acuerdo a los resultados obtenidos en la comparativa para los criterios de usabilidad y eficiencia, la metodología más adecuada para el desarrollo e implementación de la presente disertación es eXtreme Programming (XP) en cuanto a usabilidad y eficiencia.

# CAPÍTULO 4: ANÁLISIS DE REQUERIMIENTOS

En este capítulo se puntualiza los requerimientos no funcionales y funcionales que tiene la aplicación, esta última representada de manera de Historias de Usuario, además se diagraman los principales procesos que tiene Cadama Estética

## 4.1. Requerimientos funcionales

La metodología escogida para el desarrollo de la presente disertación, eXtreme Programming, propone una ficha técnica denominada Historia de Usuario para el análisis de requerimientos funcionales. En la Tabla 22, se muestra las historias de usuario generados a partir de los requerimientos funcionales.

Historia de Usuario N°1 – Gestionar Usuarios

*Tabla 22 Historia de Usuario 1 - Gestionar Usuarios*

| | |
|---|---|
| | **Historia de Usuario** |
| **Número:** 1 | **Usuario: Administrador del Sistema** |
| **Nombre historia: Gestionar Usuarios** | |
| **Prioridad en negocio:** Alta | **Riesgo en Desarrollo:** Media |
| **Puntos estimados:** 1 | **Iteración asignada:** 1 |
| **Programador responsable:** David Aráuz Moya | |
| **Descripción:**<br><br>**Como Administrador del Sistema, quiero poder ingresar, modificar, eliminar y consultar los usuarios creados para el manejo del sistema, para poder tener un mejor control de lo que se realiza.** | |

*Autor: David Alejandro Aráuz Moya, 2018*

Historia de Usuario N°2 – Gestionar Clientes

*Tabla 23 Historia de Usuario 2 - Gestionar Clientes*

| | |
|---|---|
| | **Historia de Usuario** |
| **Número:** 2 | **Usuario: Administrador del Sistema** |
| **Nombre historia: Gestionar Pacientes** | |
| **Prioridad en negocio:** Alta | **Riesgo en Desarrollo:** Media |
| **Puntos estimados:** 1 | **Iteración asignada:** 2 |
| **Programador responsable:** David Aráuz Moya | |
| **Descripción:**<br>**Como Administrador del Sistema, quiero poder ingresar, modificar, eliminar y consultar los clientes que van a realizar la reserva de la cita en la estética.**<br>**Con esto podré llevar un registro de los clientes que posee la empresa** | |

*Autor: David Alejandro Aráuz Moya, 2018*

Historia de Usuario N°3 – Gestionar Empleados

*Tabla 24 Historia de Usuario 3 - Gestionar Empleados*

| | |
|---|---|
| | **Historia de Usuario** |
| **Número:** 3 | **Usuario: Administrador del Sistema** |
| **Nombre historia: Gestionar Empleados** | |
| **Prioridad en negocio:** Alta | **Riesgo en Desarrollo:** Media |
| **Puntos estimados:** 1 | **Iteración asignada:** 3 |
| **Programador responsable:** David Aráuz Moya | |
| **Descripción:**<br>**Como Administrador del Sistema, quiero poder ingresar, modificar, eliminar y consultar los empleados que posee la empresa y gestionarlos dependiendo del horario de trabajo que posean** | |

*Autor: David Alejandro Aráuz Moya, 2018*

## Historia de Usuario N°4 – Gestionar Horas de Trabajo

*Tabla 25 Historia de Usuario 4 - Gestionar Horas de Trabajo*

| | |
|---|---|
| | **Historia de Usuario** |
| **Número:** 4 | **Usuario: Administrador del Sistema** |
| **Nombre historia: Gestionar Horas de Trabajo** | |
| **Prioridad en negocio:** Alta | **Riesgo en Desarrollo:** Media |
| **Puntos estimados:** 1 | **Iteración asignada:** 4 |
| **Programador responsable:** David Aráuz Moya | |
| **Descripción:**<br><br>**Como Administrador del Sistema, quiero poder ingresar, modificar, eliminar y consultar las reservas de citas creadas para el sistema.**<br><br>**Para poder seleccionar el empleado encargado a realizar el tratamiento, asignar un horario específico dependiendo de la disponibilidad del mismo y llevar una agenda de las reservas existentes.** | |

*Autor: David Alejandro Aráuz Moya, 2018*

## Historia de Usuario N°5 – Gestionar Servicios

*Tabla 26 Historia de Usuario 5 - Gestionar Servicios*

| | |
|---|---|
| | **Historia de Usuario** |
| **Número:** 5 | **Usuario: Administrador del Sistema** |
| **Nombre historia: Gestionar Servicios** | |
| **Prioridad en negocio:** Alta | **Riesgo en Desarrollo:** Media |
| **Puntos estimados:** 1 | **Iteración asignada:** 5 |
| **Programador responsable:** David Aráuz Moya | |
| **Descripción:**<br><br>**Como Administrador del Sistema, quiero poder ingresar, modificar, eliminar y consultar los servicios que ofrece la empresa.**<br><br>**Para poder asignar un servicio a la respectiva reserva del cliente.** | |

*Autor: David Alejandro Aráuz Moya, 2018*

Historia de Usuario N°6 – Gestionar Reservas

*Tabla 27 Historia de Usuario 6 - Gestionar Reservas*

| | |
|---|---|
| | **Historia de Usuario** |
| **Número:** 6 | **Usuario: Administrador del Sistema** |
| **Nombre historia: Gestionar Reservas** | |
| **Prioridad en negocio:** Alta | **Riesgo en Desarrollo:** Media |
| **Puntos estimados:** 1 | **Iteración asignada:** 6 |
| **Programador responsable:** David Aráuz Moya | |
| **Descripción:**<br><br>**Como Administrador del Sistema, quiero poder ingresar, modificar, eliminar y consultar las reservas de citas creadas para el sistema.**<br><br>**Para poder seleccionar el empleado encargado a realizar el tratamiento, asignar un horario específico dependiendo de la disponibilidad del mismo y llevar una agenda de las reservas existentes.** | |

*Autor: David Alejandro Aráuz Moya, 2018*

A manera de resumen, las principales funcionalidades del sistema recogidas por las historias de usuarios son las siguientes:

- F1. Gestionar Usuarios
- F2. Gestionar Clientes
- F3. Gestionar Empleados
- F4. Gestionar Horas de Trabajo
- F5. Gestionar Reservas
- F6. Gestionar Servicios

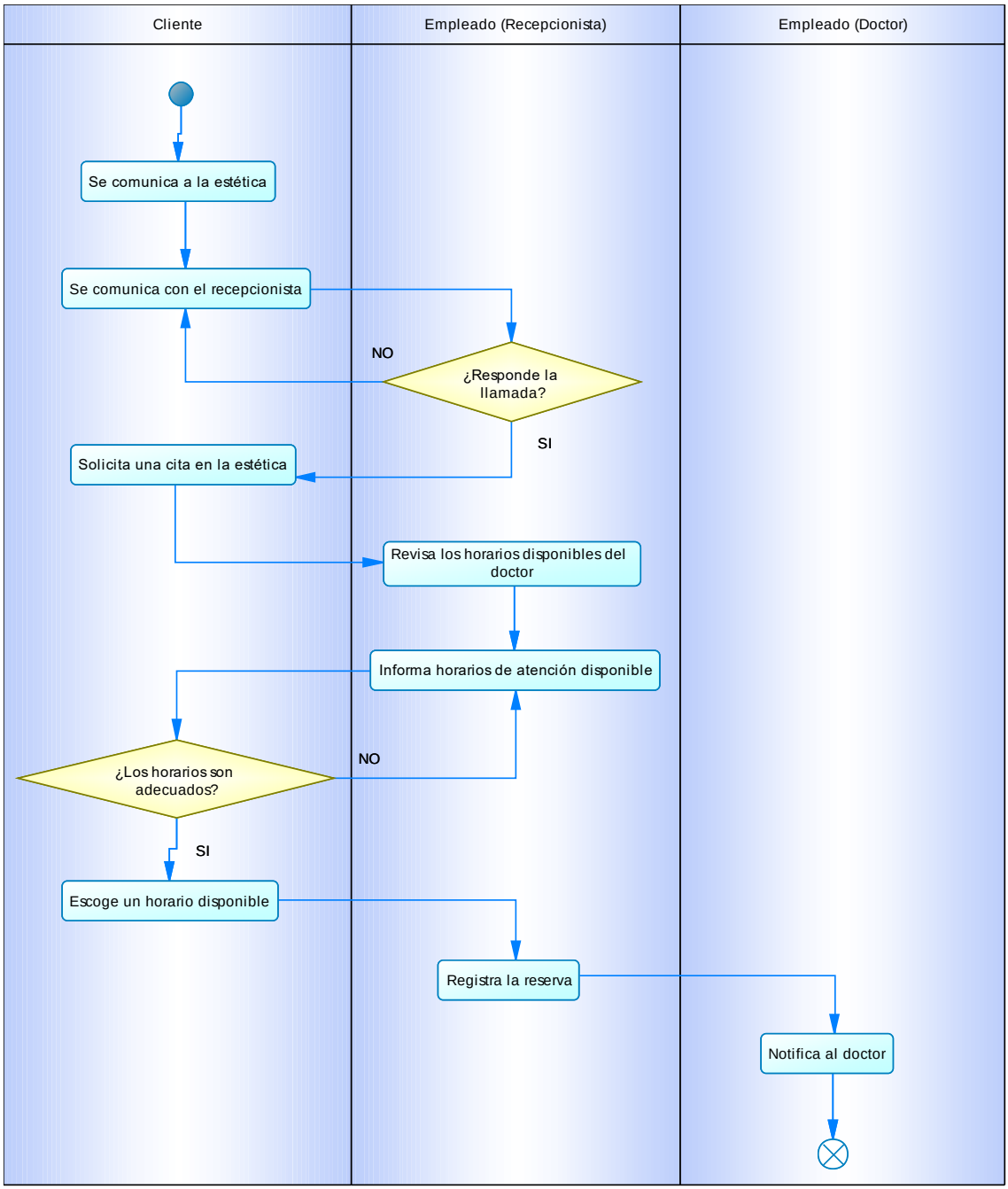
En el siguiente diagrama, se realiza una representación de las actividades que van a seguir cada uno de los procesos a automatizar dentro de la empresa.
Con esto se tendrá una visión más general y clara de las actividades que va a realizar la empresa, como se lo va a ejecutar, quienes son los encargados de llevar a cabo dicho proceso y cuál va a ser la relación entre ellas.

## 4.2. Requerimientos no funcionales

- El sistema será implementado en entorno Web.
- El sistema será multiplataforma (PHP).
- El sistema funcionará con una base de datos MySql para el almacenamiento de la información generada.
- El sistema deberá ser desarrollado con herramientas libres y son costo, para evitar la compra de licencias que supongan extra a la estética.
- El sistema será accesible para el uso del usuario.
- El sistema tendrá control de usuarios y manejo de excepciones de todo tipo.

## 4.3. Identificación de procesos

*Diagrama 1 Identificación de procesos - Reserva*




*Autor: David Alejandro Aráuz Moya, 2018*

# CAPÍTULO 5: DISEÑO DEL SISTEMA

Este capítulo está enfocado en diseñar la estructura general del sistema, siendo este la base para la construcción del mismo, definiendo las interacciones de las diferentes clases que actuarán en el sistema.

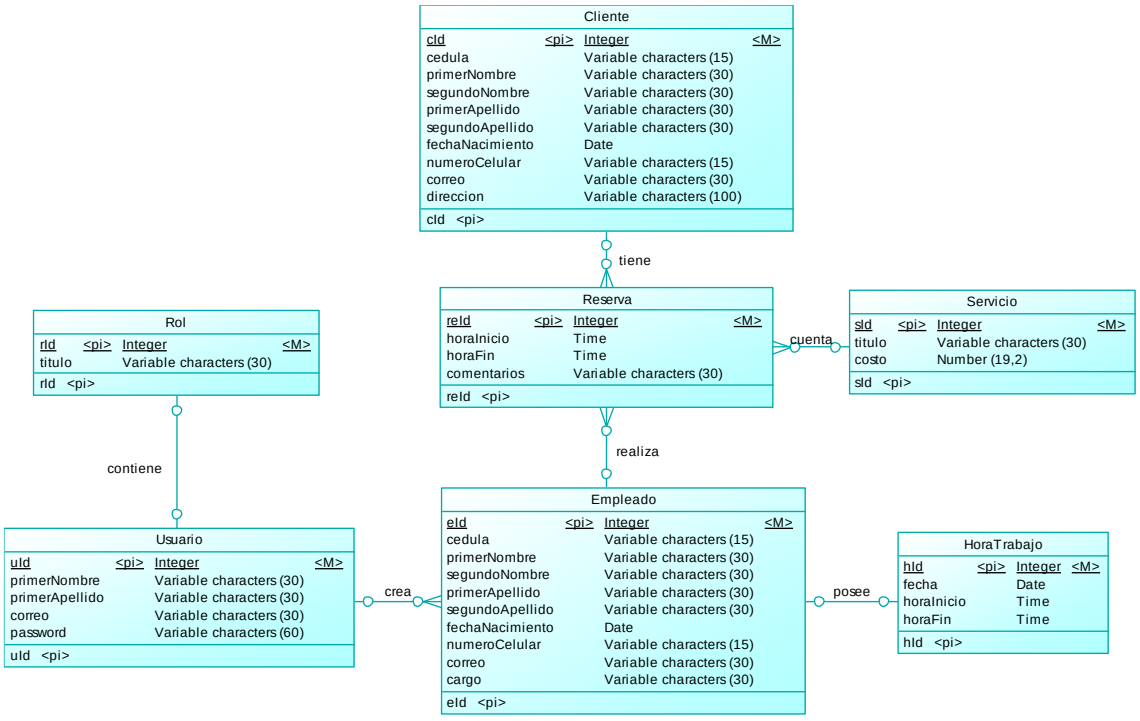
Con los diagramas realizados en este capítulo, se genera un punto de partida para la implementación del sistema, estos diagramas deben estar correctamente diseñados para que no haya ningún problema en la siguiente fase y no entrar en constante cambio al diseño ya que genera cambios en la funcionalidad y aparte retrasa los tiempos estimados al desarrollo.

## 5.1. Diseño de la base de datos

### 5.1.1. Diseño del modelo conceptual

El modelo conceptual es un diagrama que ayuda a entender al desarrollador todos los requerimientos que el cliente pide representados de forma gráfica y general para que se maneje un mismo lenguaje entre el usuario final y el desarrollador

*Diagrama 2 - Modelo Conceptual*

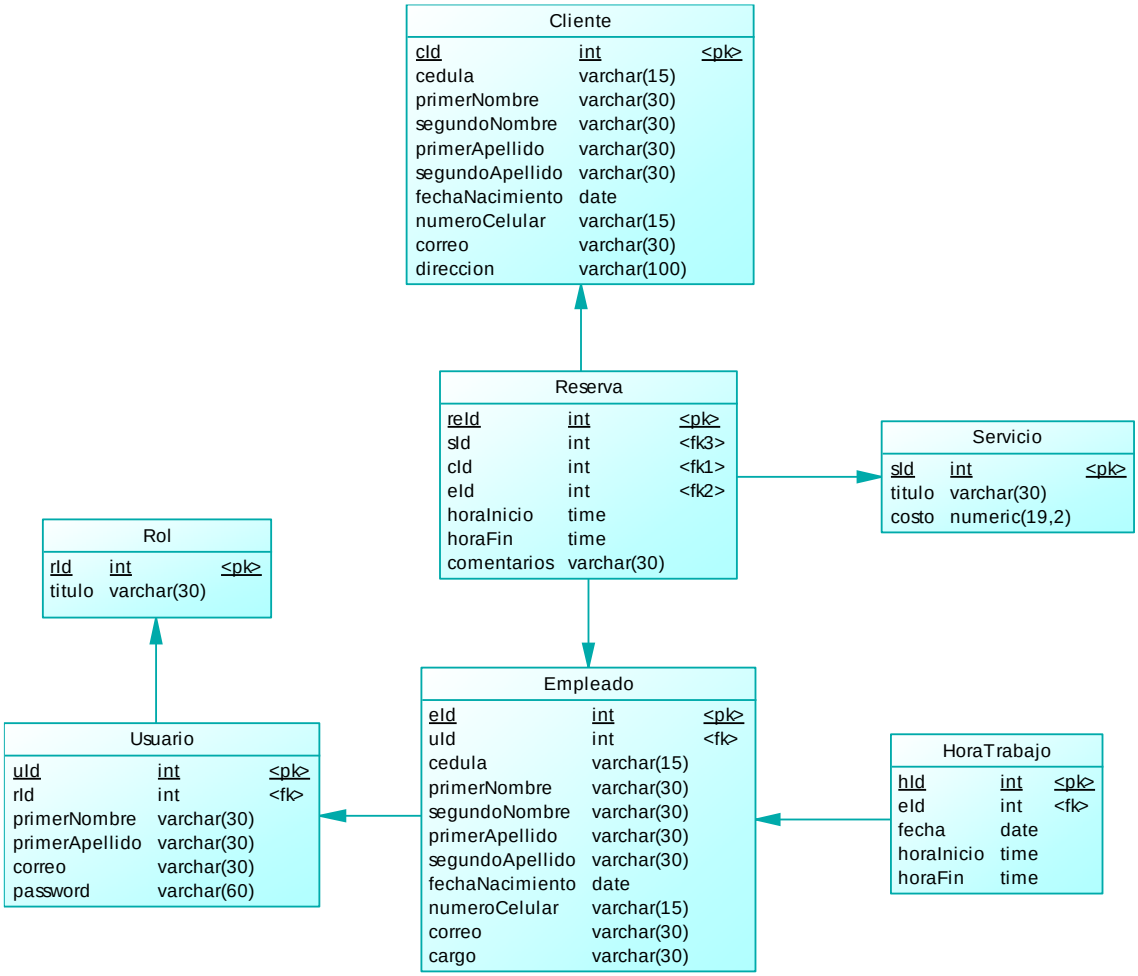



*Autor: David Alejandro Aráuz Moya, 2018*

### 5.1.2. Diseño del modelo físico

El modelo físico es un diagrama que presenta las clases o también llamadas entidades y las relaciones que tienen entre sí, también muestra los atributos con sus respectivos tipos de datos que se usan para crear la base de datos a partir del modelo propiamente dicho.

*Diagrama 3 - Modelo Físico*




*Autor: David Alejandro Aráuz Moya, 2018*

## 5.2. Diseños de la aplicación

En este apartado, se muestran los diseños de la aplicación seleccionados para la presente disertación, la metodología Extreme Programming, al ser una metodología ágil centrada en desarrollar aplicaciones de manera rápida, tiene sus limitaciones en cuanto tiene que ver con la documentación, es decir, no debe ser muy extensa, sino que debe ser clara y mostrar cómo funciona la aplicación.

Al utilizar ésta metodología, cabe recalcar que se utilizó el lenguaje de modelado unificado (UML) como estándar dentro de Extreme Programming para mejorar los aportes que ésta pueda brindar a través de sus herramientas, a cambio de las falencias propuestas que tiene la metodología XP.

Por motivos de presentación, se muestra el desarrollo de los diagramas para las clases Cliente y Empleado, y para el proceso de Reserva, los diagramas faltantes se adjuntan en el material virtual de la presente disertación porque presentan similitudes y su comportamiento es el mismo.

### 5.2.1. Diagrama de casos de uso

Caso de uso se entiende como la representación ordenada a manera de pasos, a las acciones que debe realizar el actor del sistema para cumplir un proceso de manera satisfactoria.

En el siguiente diagrama se muestra de manera general los casos de uso que tendrá el sistema a implementar.

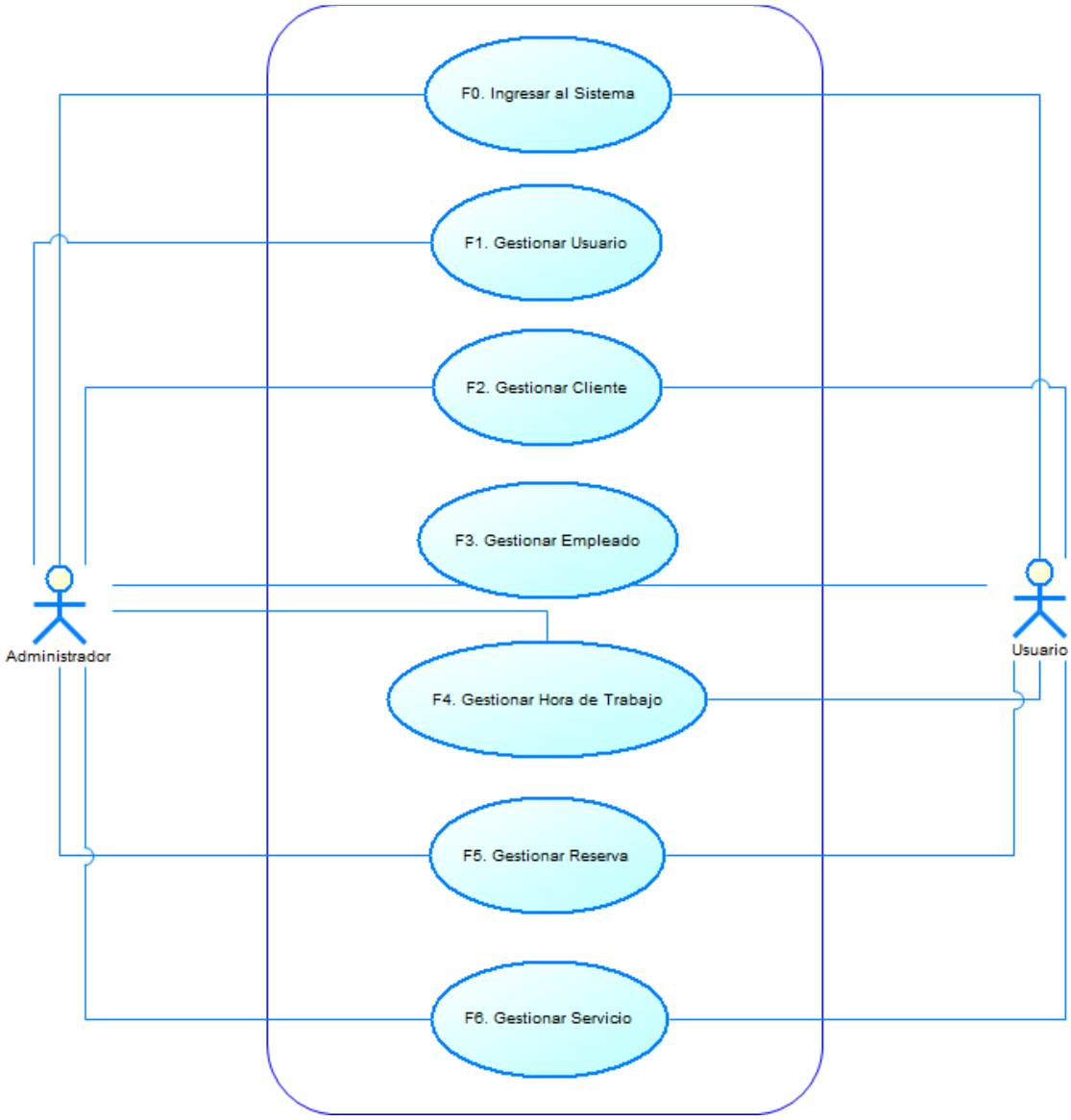
#### Diagrama general de casos de uso

En este diagrama se muestra de forma general las funcionalidades que posee el sistema desde el punto de vista del usuario, fundamental para la recopilación de requerimientos con el usuario final.

El caso de uso se representa mediante una figura ovalada indicando la función del sistema.

El actor viene a representar al usuario que va a ocupar la funcionalidad, entendiéndose como un rol que va a ocupar en el sistema.

*Diagrama 4 - Diagrama General Casos de Uso*

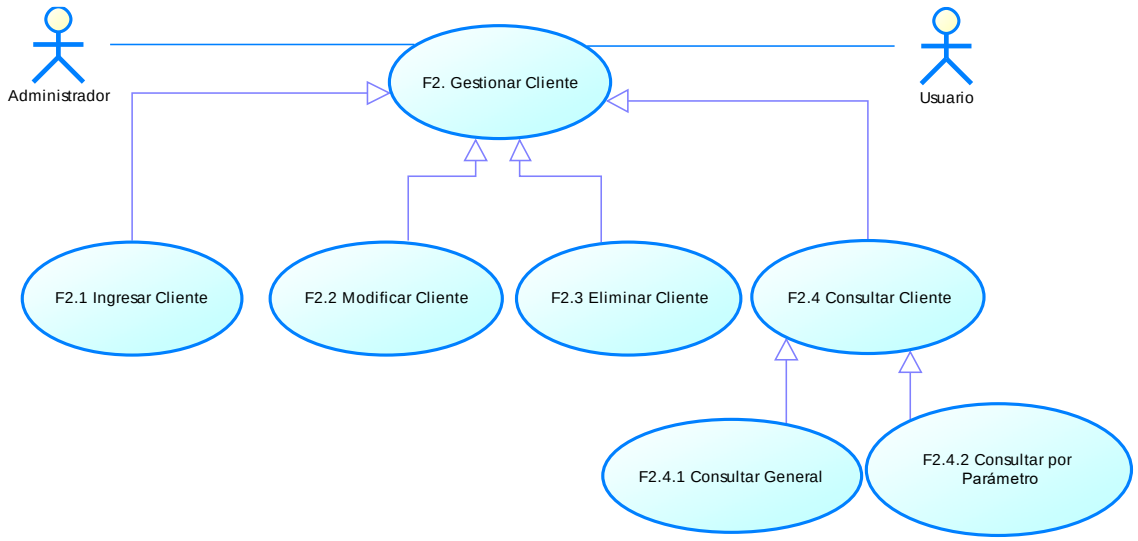



*Autor: David Alejandro Aráuz Moya, 2018*

## Diagrama específico de casos de uso

## F2. Gestionar Cliente

*Diagrama 5 - F2. Gestionar Cliente - Caso de Uso*




*Autor: David Alejandro Aráuz Moya, 2018*

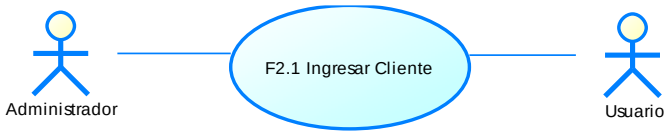
### Caso de uso a detalle: Cliente

**ID**: F2.1 Ingresar Cliente

**DESCRIPCIÓN**: El sistema deberá comportarse como se describe en el siguiente caso de uso cuando el actor añade un Cliente al sistema.

**ACTORES**: Administrador, Usuario

*Diagrama 6 - F2.1 Ingresar Cliente - Caso de Uso*




*Autor: David Alejandro Aráuz Moya, 2018*

**FLUJO PRINCIPAL**:

1. El actor selecciona Cliente del menú principal.
2. El sistema despliega una ventana para Gestionar al Cliente.
3. El sistema carga los datos del Cliente (E1).
4. El actor presiona el botón Ingresar, para añadir un nuevo Cliente.
5. El sistema presenta la ventana de Ingreso del Cliente.
6. El actor ingresa cédula del Cliente.

7. El actor ingresa datos del paciente.
8. El actor presiona el botón guardar.
9. El sistema guarda los datos. (E1) (E2).

**FLUJO ALTERNO**:

3. El sistema no tiene Clientes registrados, la tabla se muestra vacía.

6. El número de cédula ya existe

Ver caso de uso F2.2: Modificar Cliente.

Ver caso de uso F2.3: Eliminar Cliente.

**EXCEPCIONES**:

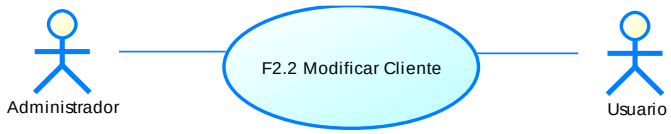
| Motivo: | Solución: |
|---|---|
| E1: No hay conexión a la Base de Datos | Contactarse con el encargado de la base de datos y su respectiva gestión con la conexión. |
| E2: Ya existe el Cliente | La cédula ya existe, vuelva a ingresar. |

**ID**: F2.2 Modificar Cliente

**DESCRIPCIÓN**: El sistema deberá comportarse como se describe en el siguiente caso de uso cuando el actor modifica un Cliente que ya ha sido ingresado al sistema.

**ACTORES**: Administrador, Usuario

*Diagrama 7 - F2.2 Modificar Cliente - Caso de Uso*




*Autor: David Alejandro Aráuz Moya, 2018*

**FLUJO PRINCIPAL**:

1. El actor selecciona Cliente del menú principal.
2. El sistema despliega una ventana para Gestionar al Cliente.
3. El sistema carga los datos del Cliente (E1).
4. El actor presiona el botón Editar, dentro de la tabla en la que se encuentra el Cliente.
5. El sistema presenta la ventana de Edición del Cliente.

6. El sistema carga los datos del Cliente seleccionado. (E1)
7. El actor modifica los datos deseados del Cliente.
8. El actor presiona actualizar
9. El sistema actualiza los datos de Cliente (E1).

**FLUJO ALTERNO**:

3. El sistema no tiene Clientes registrados, la tabla se muestra vacía.

8. Faltan datos obligatorios por llenar, completar los campos

**EXCEPCIONES**:

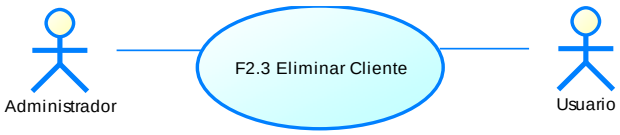
| Motivo: | Solución: |
|---|---|
| E1: No hay conexión a la Base de Datos | Contactarse con el encargado de la base de datos y su respectiva gestión con la conexión. |

**ID**: F2.3 Eliminar Cliente

**DESCRIPCIÓN**: El sistema deberá comportarse como se describe en el siguiente caso de uso cuando el actor elimina un Cliente que ya ha sido ingresado al sistema.

**ACTORES**: Administrador, Usuario

*Diagrama 8 - F2.3 Eliminar Cliente - Caso de Uso*




*Autor: David Alejandro Aráuz Moya, 2018*

**FLUJO PRINCIPAL**:

1. El actor selecciona Cliente del menú principal.
2. El sistema despliega una ventana para Gestionar al Cliente.
3. El sistema carga los datos del Cliente (E1).
4. El actor presiona el botón Eliminar, dentro de la tabla en la que se encuentra el Cliente.
5. El sistema presenta una ventana emergente para notificar que el Cliente va a ser eliminado.
6. El actor presiona la opción SI en la ventana emergente.

7. El sistema elimina el Cliente de la Base de Datos (E1).

**FLUJO ALTERNO**:

3. El sistema no tiene Clientes registrados, la tabla se muestra vacía.

6. El actor presiona la opción NO en la ventana emergente, no se elimina el Cliente.

**EXCEPCIONES**:

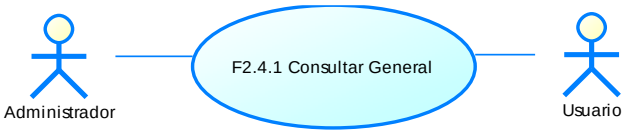
| Motivo: | Solución: |
|---|---|
| E1: No hay conexión a la Base de Datos | Contactarse con el encargado de la base de datos y su respectiva gestión con la conexión. |

**ID**: F2.4.1 Consultar Cliente (General)

**DESCRIPCIÓN**: El sistema deberá comportarse como se describe en el siguiente caso de uso cuando el actor consulta un Cliente que ya ha sido ingresado al sistema.

**ACTORES**: Administrador, Usuario

*Diagrama 9 - F2.4.1 Consulta General Cliente - Caso de Uso*




*Autor: David Alejandro Aráuz Moya, 2018*

**FLUJO PRINCIPAL**:

1. El actor selecciona Cliente del menú principal.
2. El sistema despliega una ventana para Gestionar al Cliente.
3. El sistema carga los datos del Cliente (E1).
4. El sistema presenta una tabla con los Clientes ingresados.
5. El sistema muestra los primeros 100 registros.

**FLUJO ALTERNO**:

4. El sistema no tiene Clientes registrados, la tabla se muestra vacía.

**EXCEPCIONES**:

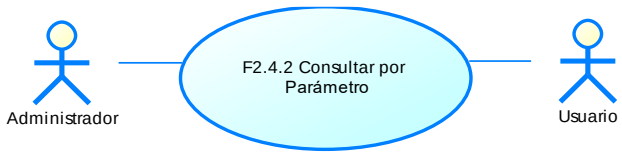
| Motivo: | Solución: |
|---|---|
| E1: No hay conexión a la Base de Datos | Contactarse con el encargado de la base de datos y su respectiva gestión con la conexión. |

**ID**: F2.4.2 Consultar Cliente (Parámetro)

**DESCRIPCIÓN**: El sistema deberá comportarse como se describe en el siguiente caso de uso cuando el actor consulta por medio de un parámetro un Cliente que ya ha sido ingresado al sistema.

**ACTORES**: Administrador, Usuario

*Diagrama 10 - F2.4.2 Consultar Parámetro Cliente - Caso de Uso*




*Autor: David Alejandro Aráuz Moya, 2018*

**FLUJO PRINCIPAL**:

1. El actor selecciona Cliente del menú principal.
2. El sistema despliega una ventana para Gestionar al Cliente.
3. El sistema carga los datos del Cliente (E1).
4. El sistema presenta una tabla con los Clientes ingresados.
5. El sistema muestra los primeros 100 registros.
6. El actor ingresa el parámetro deseado a buscar en la caja de texto ubicada en la parte superior derecha de la tabla de Clientes.
7. El sistema busca automáticamente la información por el parámetro ingresado.
8. El sistema presenta en la tabla la información encontrada.

**FLUJO ALTERNO**:

3. El sistema no tiene Clientes registrados, la tabla se muestra vacía.

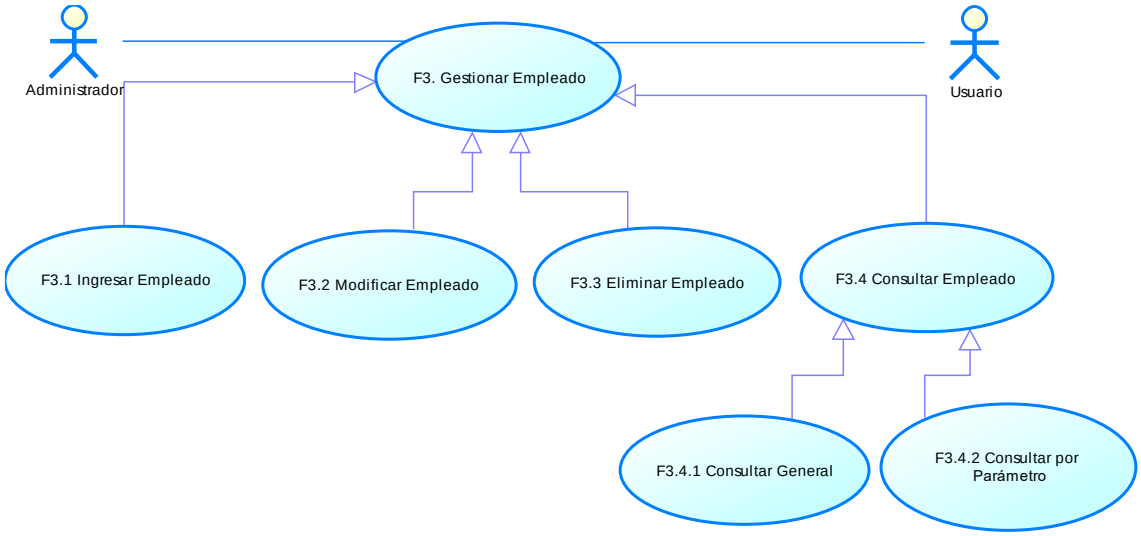
6. El sistema no encuentra ningún registro, la tabla se muestra vacía.

**EXCEPCIONES**:

| Motivo: | Solución: |
|---|---|
| E1: No hay conexión a la Base de Datos | Contactarse con el encargado de la base de datos y su respectiva gestión con la conexión. |

## F3. Gestionar Empleado

*Diagrama 11 - F3. Gestionar Empleado - Caso de Uso*




*Autor: David Alejandro Aráuz Moya, 2018*

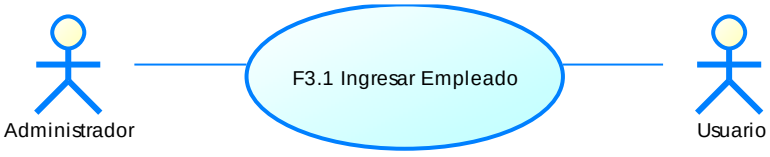
### Caso de uso a detalle: Empleado

**ID**: F3.1 Ingresar Empleado

**DESCRIPCIÓN**: El sistema deberá comportarse como se describe en el siguiente caso de uso cuando el actor añade un Empleado al sistema.

**ACTORES**: Administrador, Usuario

*Diagrama 12 - F3.1 Ingresar Empleado - Caso de Uso*




*Autor: David Alejandro Aráuz Moya, 2018*

**FLUJO PRINCIPAL**:

1. El actor selecciona Empleado del menú principal.

2. El sistema despliega una ventana para Gestionar al Empleado.
3. El sistema carga los datos del Empleado (E1).
4. El actor presiona el botón Ingresar, para añadir un nuevo Empleado.
5. El sistema presenta la ventana de Ingreso del Empleado.
6. El actor ingresa cédula del Empleado.
7. El actor ingresa datos del paciente.
8. El actor presiona el botón guardar.
9. El sistema guarda los datos. (E1) (E2).

**FLUJO ALTERNO**:

3. El sistema no tiene Empleados registrados, la tabla se muestra vacía.

6. El número de cédula ya existe

Ver caso de uso F3.2: Modificar Empleado.

Ver caso de uso F3.3: Eliminar Empleado.

**EXCEPCIONES**:

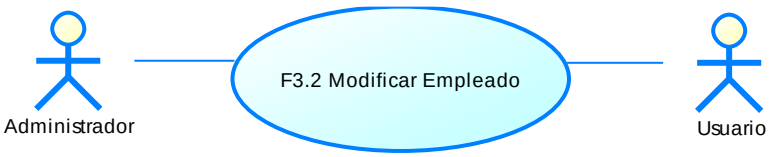
| Motivo: | Solución: |
|---|---|
| E1: No hay conexión a la Base de Datos | Contactarse con el encargado de la base de datos y su respectiva gestión con la conexión. |
| E2: Ya existe el Cliente | La cédula ya existe, vuelva a ingresar. |

**ID**: F3.2 Modificar Empleado

**DESCRIPCIÓN**: El sistema deberá comportarse como se describe en el siguiente caso de uso cuando el actor modifica un Empleado que ya ha sido ingresado al sistema.

**ACTORES**: Administrador, Usuario

*Diagrama 13 - F3.2 Modificar Empleado - Caso de Uso*




*Autor: David Alejandro Aráuz Moya, 2018*

**FLUJO PRINCIPAL**:

1. El actor selecciona Empleado del menú principal.

2. El sistema despliega una ventana para Gestionar al Empleado.
3. El sistema carga los datos del Empleado (E1).
4. El actor presiona el botón Editar, dentro de la tabla en la que se encuentra el Empleado.
5. El sistema presenta la ventana de Edición del Empleado.
6. El sistema carga los datos del Empleado seleccionado. (E1)
7. El actor modifica los datos deseados del Empleado.
8. El actor presiona actualizar
9. El sistema actualiza los datos de Empleado (E1).

**FLUJO ALTERNO**:

8. Faltan datos obligatorios por llenar, completar los campos

**EXCEPCIONES**:

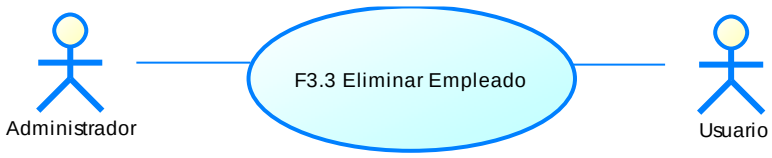
| Motivo: | Solución: |
|---|---|
| E1: No hay conexión a la Base de Datos | Contactarse con el encargado de la base de datos y su respectiva gestión con la conexión. |

**ID**: F3.3 Eliminar Empleado

**DESCRIPCIÓN**: El sistema deberá comportarse como se describe en el siguiente caso de uso cuando el actor elimina un Empleado que ya ha sido ingresado al sistema.

**ACTORES**: Administrador, Usuario

*Diagrama 14 - F3.3 Eliminar Empleado - Caso de Uso*




*Autor: David Alejandro Aráuz Moya, 2018*

**FLUJO PRINCIPAL**:

1. El actor selecciona Empleado del menú principal.
2. El sistema despliega una ventana para Gestionar al Empleado.
3. El sistema carga los datos del Empleado (E1).

4. El actor presiona el botón Eliminar, dentro de la tabla en la que se encuentra el Empleado.
5. El sistema presenta una ventana emergente para notificar que el Empleado va a ser eliminado.
6. El actor presiona la opción SI en la ventana emergente.
7. El sistema elimina el Empleado de la Base de Datos (E1).

**FLUJO ALTERNO**:

6. El actor presiona la opción NO en la ventana emergente, no se elimina el Empleado.

**EXCEPCIONES**:

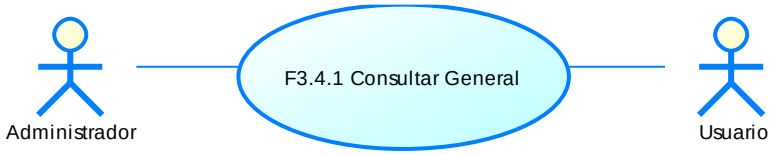
| Motivo: | Solución: |
|---|---|
| E1: No hay conexión a la Base de Datos | Contactarse con el encargado de la base de datos y su respectiva gestión con la conexión. |

**ID**: F3.4.1 Consultar Empleado (General)

**DESCRIPCIÓN**: El sistema deberá comportarse como se describe en el siguiente caso de uso cuando el actor consulta un Empleado que ya ha sido ingresado al sistema.

**ACTORES**: Administrador, Usuario

*Diagrama 15 - F3.4.1 Consulta General Empleado - Caso de Uso*




*Autor: David Alejandro Aráuz Moya, 2018*

**FLUJO PRINCIPAL**:

1. El actor selecciona Empleado del menú principal.
2. El sistema despliega una ventana para Gestionar al Empleado.
3. El sistema carga los datos del Empleado (E1).
4. El sistema presenta una tabla con los Empleados ingresados.
5. El sistema muestra los primeros 100 registros.

**FLUJO ALTERNO**:

4. El sistema no tiene Empleado registrados, la tabla se muestra vacía.

**EXCEPCIONES**:

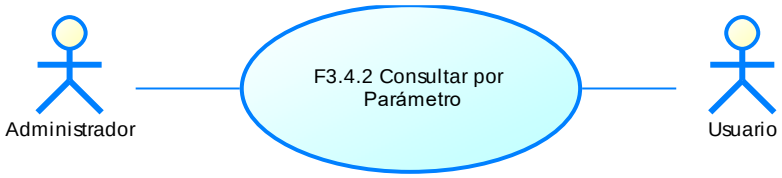
| Motivo: | Solución: |
|---|---|
| E1: No hay conexión a la Base de Datos | Contactarse con el encargado de la base de datos y su respectiva gestión con la conexión. |

**ID**: F3.4.2 Consultar Empleado (Parámetro)

**DESCRIPCIÓN**: El sistema deberá comportarse como se describe en el siguiente caso de uso cuando el actor consulta por medio de un parámetro un Empleado que ya ha sido ingresado al sistema.

**ACTORES**: Administrador, Usuario

*Diagrama 16 - F3.4.2 Consultar Parámetro Empleado - Caso de Uso*




*Autor: David Alejandro Aráuz Moya, 2018*

**FLUJO PRINCIPAL**:

1. El actor selecciona Empleado del menú principal.
2. El sistema despliega una ventana para Gestionar al Empleado.
3. El sistema carga los datos del Empleado (E1).
4. El sistema presenta una tabla con los Empleados ingresados.
5. El sistema muestra los primeros 100 registros.
6. El actor ingresa el parámetro deseado a buscar en la caja de texto ubicada en la parte superior derecha de la tabla de Empleado.
7. El sistema busca automáticamente la información por el parámetro ingresado.
8. El sistema presenta en la tabla la información encontrada.

**FLUJO ALTERNO**:

4. El sistema no tiene Empleados registrados, la tabla se muestra vacía.

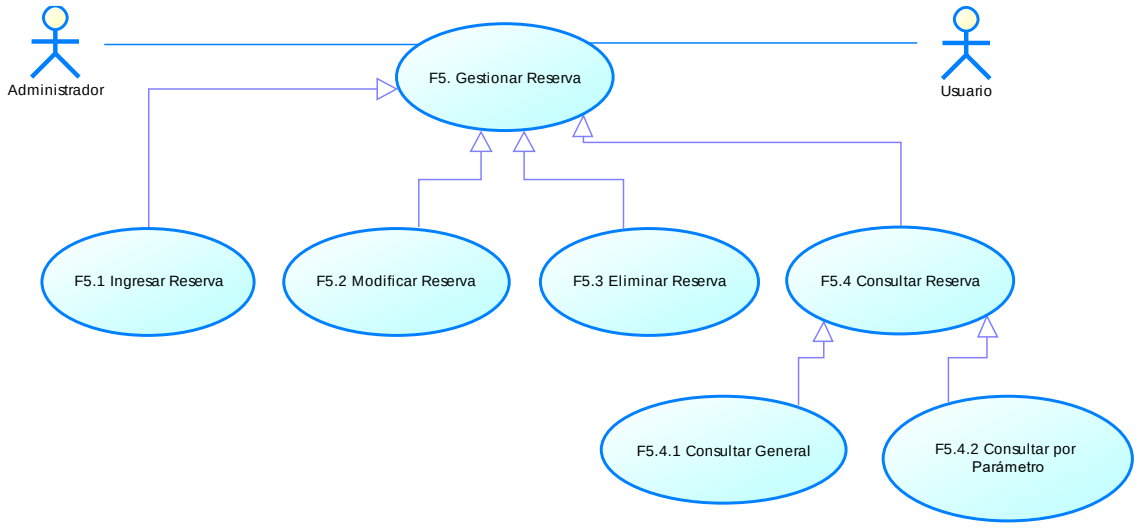
6. El sistema no encuentra ningún registro, la tabla se muestra vacía.

**EXCEPCIONES**:

| Motivo: | Solución: |
|---|---|
| E1: No hay conexión a la Base de Datos | Contactarse con el encargado de la base de datos y su respectiva gestión con la conexión. |

## F5. Gestionar Reserva

*Diagrama 17 - F5. Gestionar Reserva - Caso de Uso*




*Autor: David Alejandro Aráuz Moya, 2018*

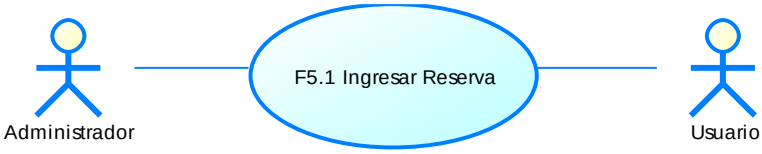
### Caso de uso a detalle del proceso: Reserva

**ID**: F5.1 Ingresar Reserva

**DESCRIPCIÓN**: El sistema deberá comportarse como se describe en el siguiente caso de uso cuando el actor añade una Reserva al sistema, teniendo previamente cargados los datos adicionales para la gestión.

**ACTORES**: Administrador, Usuario

*Diagrama 18 - F5.1 Ingresar Reserva - Caso de Uso*



**FLUJO PRINCIPAL**:

1. El actor selecciona Reserva del menú principal.
2. El sistema despliega una ventana para Gestionar al Reserva.
3. El sistema carga los datos de la Reserva en el calendario y en la tabla (E1).
4. El actor presiona el botón Ingresar, para añadir un nuevo Reserva.
5. El sistema presenta la ventana de Ingreso de la Reserva.
6. El sistema carga la cédula del Cliente (E1) (E2)
7. El sistema carga la cédula del Empleado (E1) (E2)
8. El sistema carga el Servicio ofrecido por la empresa (E1) (E2)
9. El actor selecciona la fecha de la Reserva.
10. El actor selecciona la hora de inicio de la Reserva.
11. El actor selecciona la hora de fin de la Reserva.
12. El actor ingresa algún comentario en la Reserva
13. El actor presiona el botón guardar.
14. El sistema guarda los datos. (E1) (E2).

**FLUJO ALTERNO**:

3. El sistema no tiene Empleados registrados, la tabla se muestra vacía.
6. Si no existe el Cliente, dirigirse al Caso de Uso Gestionar Cliente.
7. Si no existe el Empleado, dirigirse al Caso de Uso Gestionar Empleado.
8. Si no existe el Servicio, dirigirse al Caso de Uso Gestionar Servicio.

**EXCEPCIONES**:

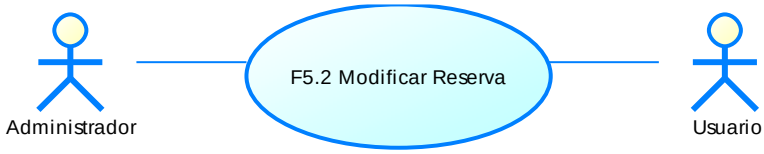
| Motivo: | Solución: |
|---|---|
| E1: No hay conexión a la Base de Datos | Contactarse con el encargado de la base de datos y su respectiva gestión con la conexión. |
| E2: No existen datos | Crear al Cliente, Empleado o Servicio |

**ID**: F5.2 Modificar Reserva

**DESCRIPCIÓN**: El sistema deberá comportarse como se describe en el siguiente caso de uso cuando el actor modifica una Reserva que ya ha sido ingresado al sistema.

**ACTORES**: Administrador, Usuario

*Diagrama 19 - F5.2 Modificar Reserva - Caso de Uso*




*Autor: David Alejandro Aráuz Moya, 2018*

**FLUJO PRINCIPAL**:

1. El actor selecciona Reserva del menú principal.
2. El sistema despliega una ventana para Gestionar la Reserva.
3. El sistema carga los datos de la Reserva seleccionada (E1).
4. El actor presiona el evento creado dentro del calendario para poder realizar la modificación de la Reserva.
5. El sistema presenta la ventana de Edición de la Reserva.
6. El sistema carga los datos de la Reserva. (E1)
7. El actor modifica los datos deseados de la Reserva.
8. El actor presiona actualizar
9. El sistema actualiza los datos de la Reserva (E1).

**FLUJO ALTERNO**:

4. El actor presiona el botón Editar, dentro de la tabla en la que se encuentra la Reserva.

8. Faltan datos obligatorios por llenar, completar los campos

**EXCEPCIONES**:

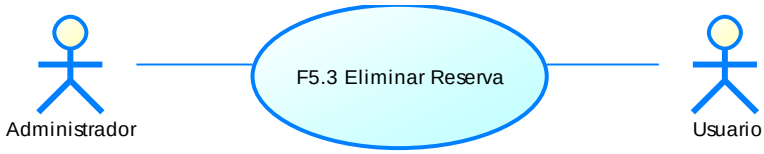
| Motivo: | Solución: |
|---|---|
| E1: No hay conexión a la Base de Datos | Contactarse con el encargado de la base de datos y su respectiva gestión con la conexión. |

**ID**: F5.3 Eliminar Reserva

**DESCRIPCIÓN**: El sistema deberá comportarse como se describe en el siguiente caso de uso cuando el actor elimina una Reserva que ya ha sido ingresado al sistema.

**ACTORES**: Administrador, Usuario

*Diagrama 20 - F5.3 Eliminar Reserva - Caso de Uso*




*Autor: David Alejandro Aráuz Moya, 2018*

**FLUJO PRINCIPAL**:

1. El actor selecciona Reserva del menú principal.
2. El sistema despliega una ventana para Gestionar la Reserva.
3. El sistema carga los datos de la Reserva (E1).
4. El actor presiona el botón Eliminar, dentro de la tabla en la que se encuentra la Reserva.
5. El sistema presenta una ventana emergente para notificar que la Reserva va a ser eliminada.
6. El actor presiona la opción SI en la ventana emergente.
7. El sistema elimina la Reserva de la Base de Datos (E1).

**FLUJO ALTERNO**:

6. El actor presiona la opción NO en la ventana emergente, no se elimina la Reserva.

**EXCEPCIONES**:

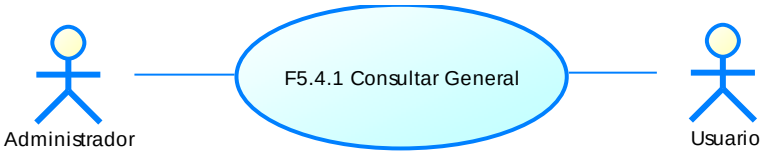
| Motivo: | Solución: |
|---|---|
| E1: No hay conexión a la Base de Datos | Contactarse con el encargado de la base de datos y su respectiva gestión con la conexión. |

**ID**: F5.4.1 Consultar Reserva (General)

**DESCRIPCIÓN**: El sistema deberá comportarse como se describe en el siguiente caso de uso cuando el actor consulta una Reserva que ya ha sido ingresado al sistema.

**ACTORES**: Administrador, Usuario

*Diagrama 21 - F5.4.1 Consulta General Reserva - Caso de Uso*




*Autor: David Alejandro Aráuz Moya, 2018*

**FLUJO PRINCIPAL**:

1. El actor selecciona Reserva del menú principal.
2. El sistema despliega una ventana para Gestionar la Reserva.
3. El sistema carga los datos de la Reserva (E1).
4. El sistema muestra las reservas creadas dentro del calendario dependiendo de la fecha ingresada y las horas elegidas.
5. El sistema muestra los eventos por semana.

**FLUJO ALTERNO**:

4. El sistema no tiene Empleado registrados, la tabla se muestra vacía.

4. El sistema presenta una tabla con las Reserva ingresadas.

**EXCEPCIONES**:

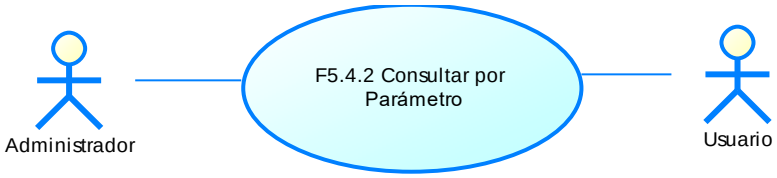
| Motivo: | Solución: |
|---|---|
| E1: No hay conexión a la Base de Datos | Contactarse con el encargado de la base de datos y su respectiva gestión con la conexión. |

**ID**: F5.4.2 Consultar Empleado (Parámetro)

**DESCRIPCIÓN**: El sistema deberá comportarse como se describe en el siguiente caso de uso cuando el actor consulta por medio de un parámetro una Reserva que ya ha sido ingresado al sistema.

**ACTORES**: Administrador, Usuario

*Diagrama 22 - F5.4.2 Consultar Parámetro Reserva - Caso de Uso*




*Autor: David Alejandro Aráuz Moya, 2018*

**FLUJO PRINCIPAL**:

1. El actor selecciona Reserva del menú principal.
2. El sistema despliega una ventana para Gestionar la Reserva.
3. El sistema carga los datos de la Reserva (E1).
4. El sistema presenta una tabla con las Reserva ingresadas.
5. El sistema muestra los primeros 100 registros.
6. El actor ingresa el parámetro deseado a buscar en la caja de texto ubicada en la parte superior derecha de la tabla de la Reserva.
7. El sistema busca automáticamente la información por el parámetro ingresado.
8. El sistema presenta en la tabla la información encontrada.

**FLUJO ALTERNO**:

4. El sistema no tiene Reservas registrados, la tabla se muestra vacía.

6. El sistema no encuentra ningún registro, la tabla se muestra vacía.

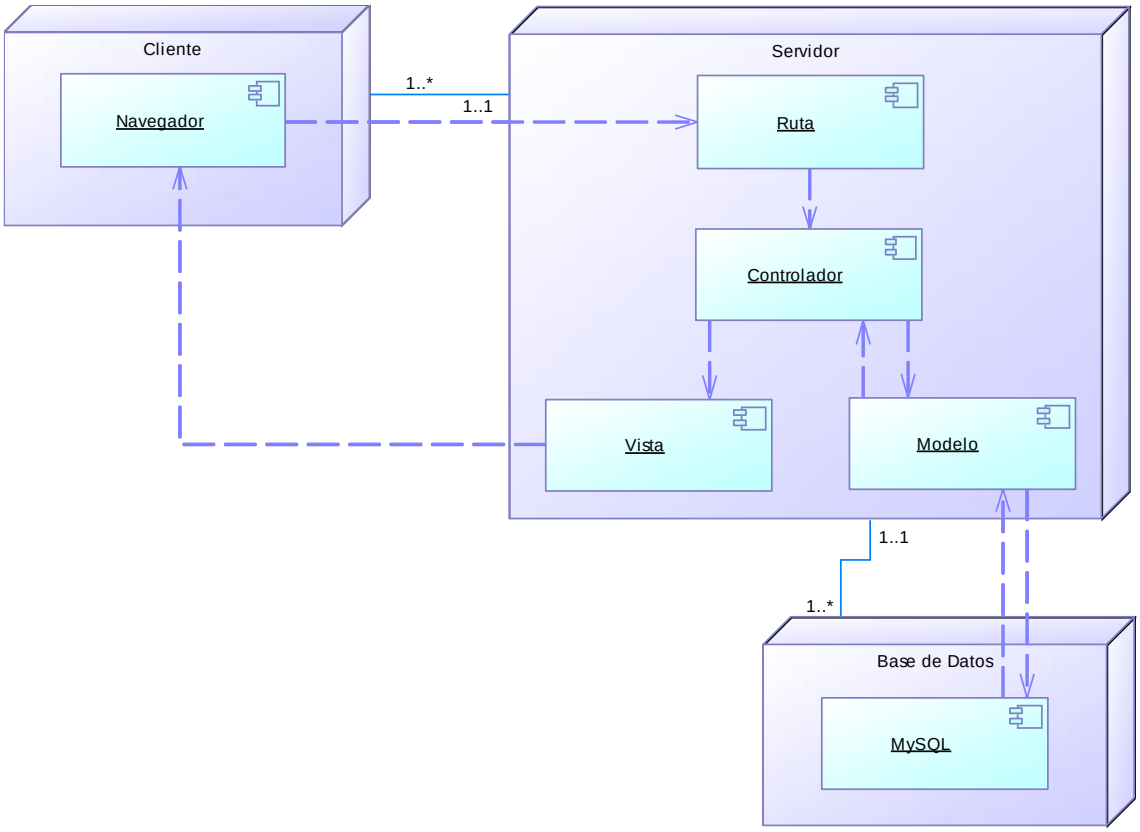
**EXCEPCIONES**:

| Motivo: | Solución: |
|---|---|
| E1: No hay conexión a la Base de Datos | Contactarse con el encargado de la base de datos y su respectiva gestión con la conexión. |

## 5.3. Arquitectura del sistema

El sistema desarrollado para la presente disertación utiliza el framework Laravel, este a su vez ocupa una variante de la arquitectura Modelo – Vista – Controlador, que es presentado a continuación.

*Diagrama 23 - Diagrama de Despliegue*




*Autor: David Alejandro Aráuz Moya, 2018*

## 5.4. Diagramas de secuencia

Una vez descrito los casos de uso del sistema y teniendo clara cuál va a ser la arquitectura del sistema, los diagramas de secuencia van a ayudar a representar cómo y en qué orden funcionan los objetos en conjunto, es decir, guían a una mayor comprensión de los requisitos del sistema.

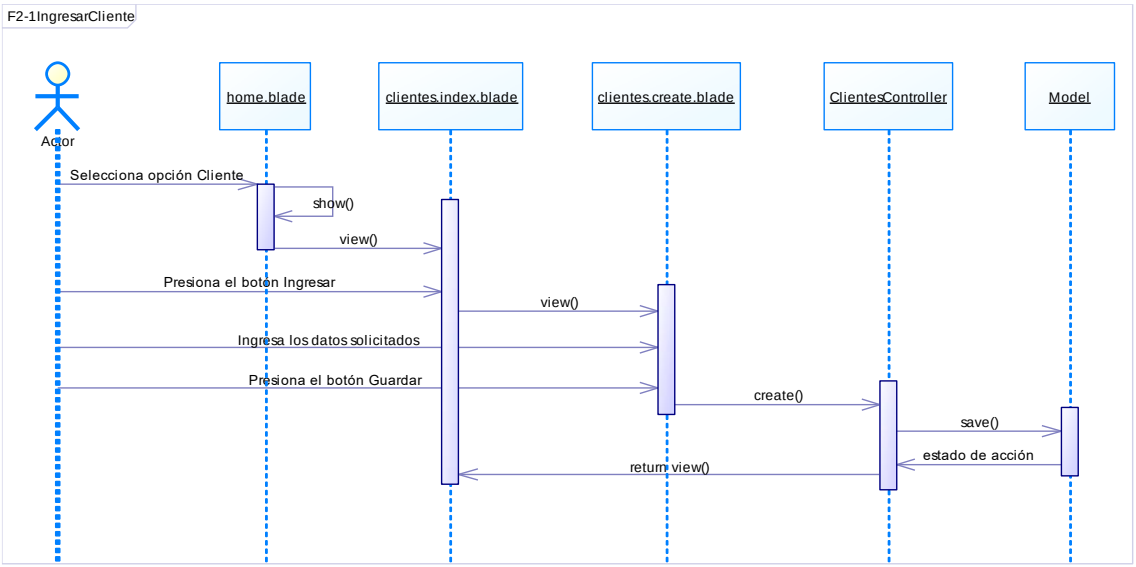
A continuación, se describirán los diagramas de secuencia aplicados al cliente, empleado y la reserva, con sus respectivas acciones de insertar, modificar, eliminar, consultas (general y parámetro).

Por motivos de presentación, se muestra el desarrollo de los diagramas para las clases Cliente y Empleado, y para el proceso de Reserva, los diagramas faltantes se adjuntan en el material virtual de la presente disertación porque presentan similitudes y su comportamiento es el mismo.

## F2. Gestionar Cliente

### F2.1 Ingresar Cliente

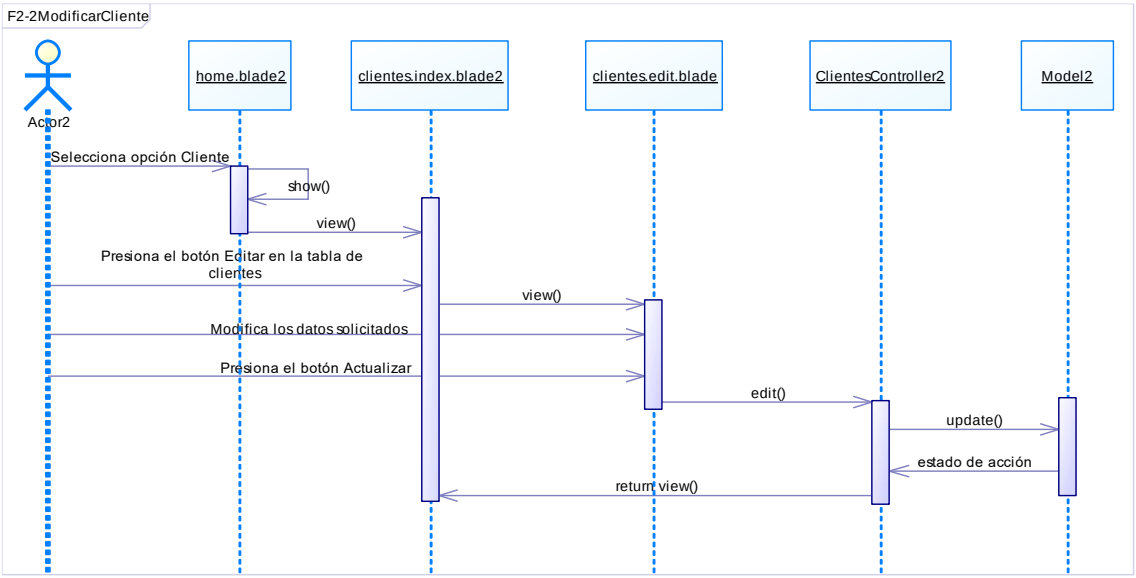
*Diagrama 24 F2.1 Ingresar Cliente - Secuencia*




*Autor: David Alejandro Aráuz Moya, 2018*

### F2.2 Modificar Cliente

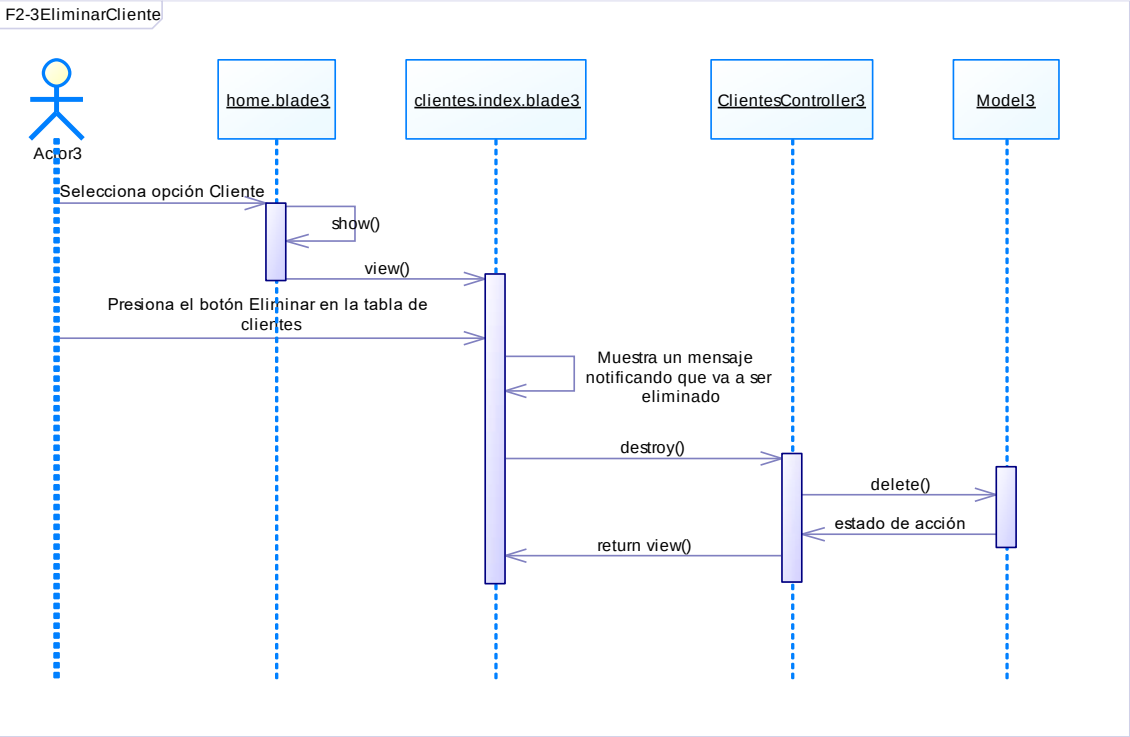
*Diagrama 25 F2.2 Modificar Cliente - Secuencia*




*Autor: David Alejandro Aráuz Moya, 2018*

## F2.3 Eliminar Cliente

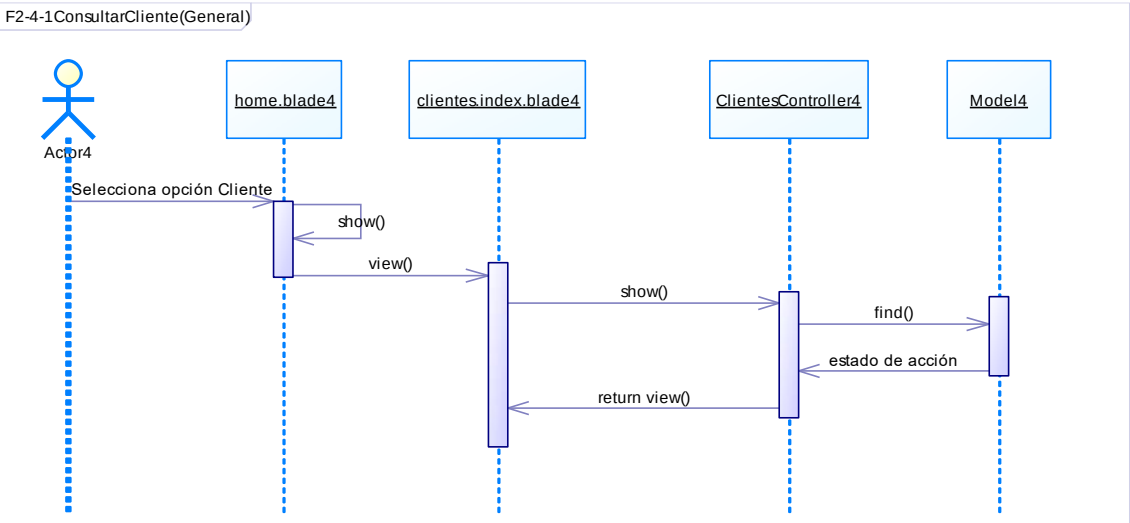
*Diagrama 26 F2.3 Eliminar Cliente - Secuencia*




*Autor: David Alejandro Aráuz Moya, 2018*

## F2.4.1 Consultar Cliente (General)

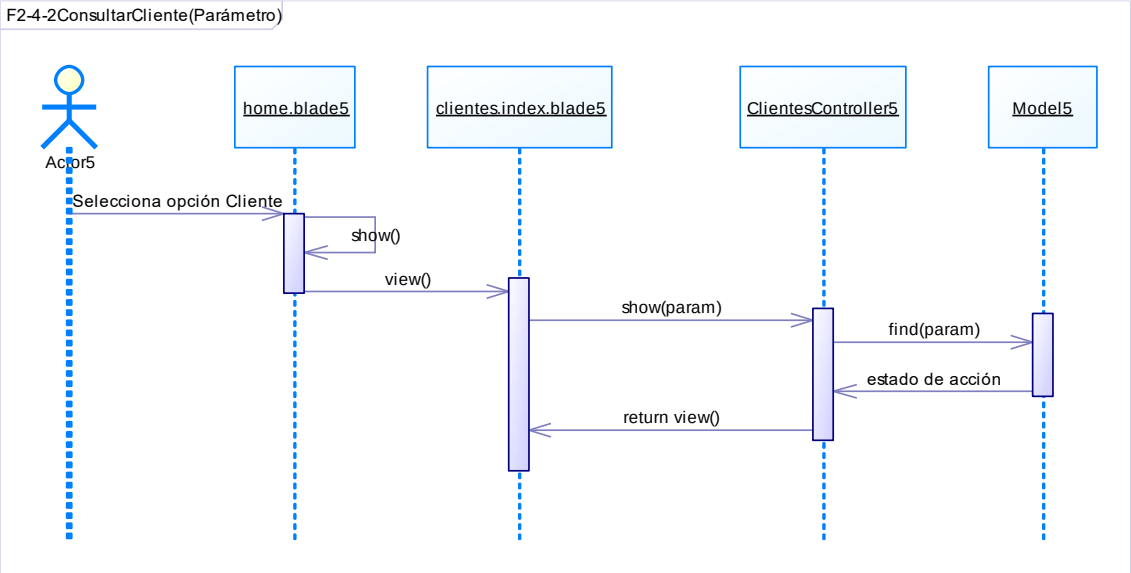
*Diagrama 27 F2.4.1 Consultar Cliente (General) - Secuencia*




*Autor: David Alejandro Aráuz Moya, 2018*

### F2.4.2 Consultar Cliente (Parámetro)

*Diagrama 28 F2.4.2 Consultar Cliente (Parámetro) - Secuencia*

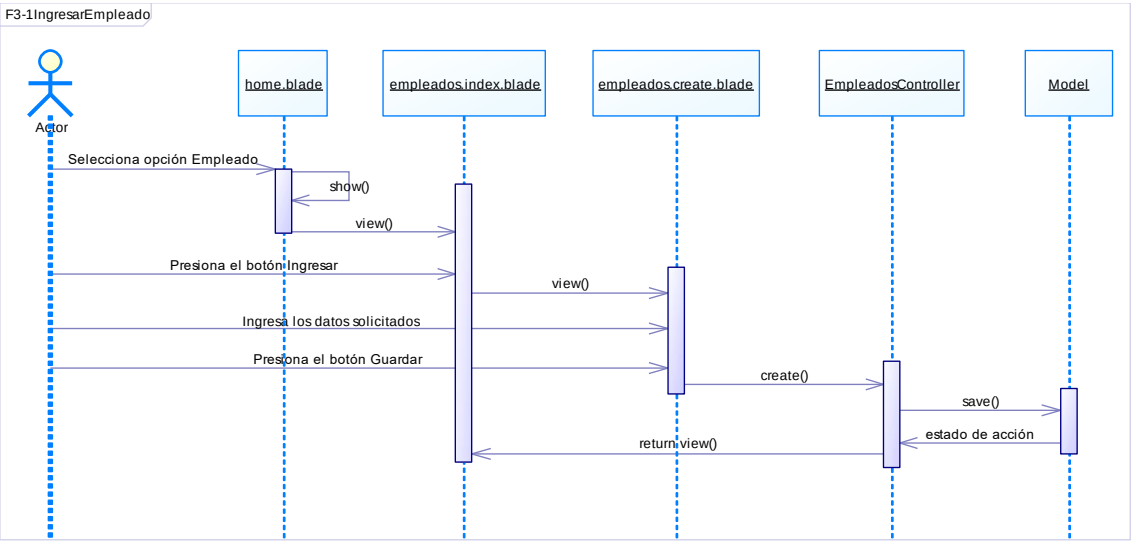



*Autor: David Alejandro Aráuz Moya, 2018*

## F3. Gestionar Empleado

### F3.1 Ingresar Empleado

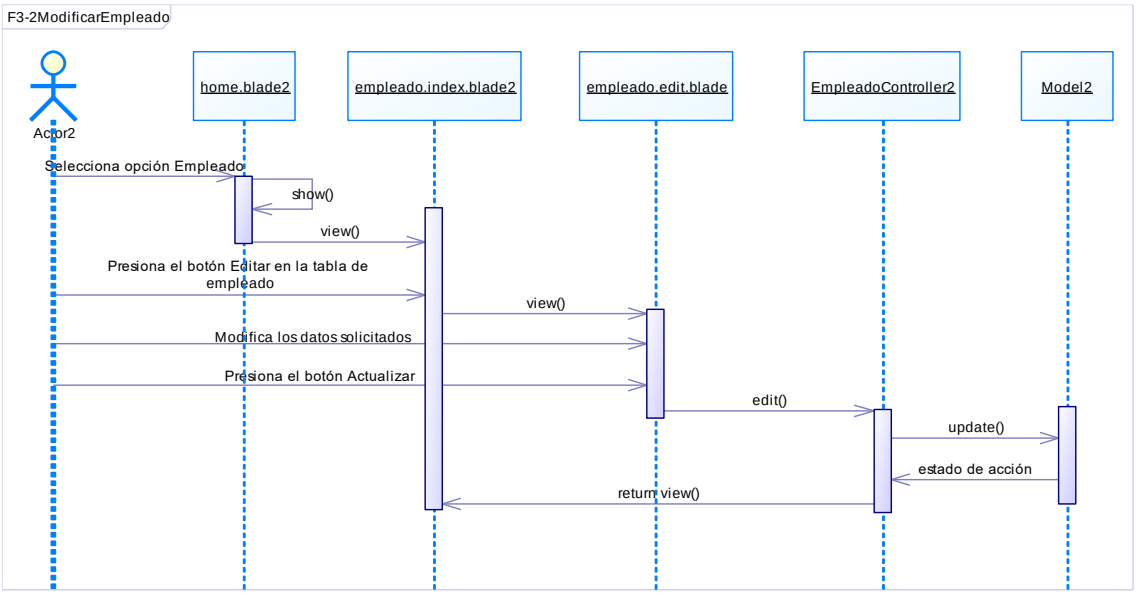
*Diagrama 29 F3.1 Ingresar Empleado - Secuencia*




*Autor: David Alejandro Aráuz Moya, 2018*

## F3.2 Modificar Empleado

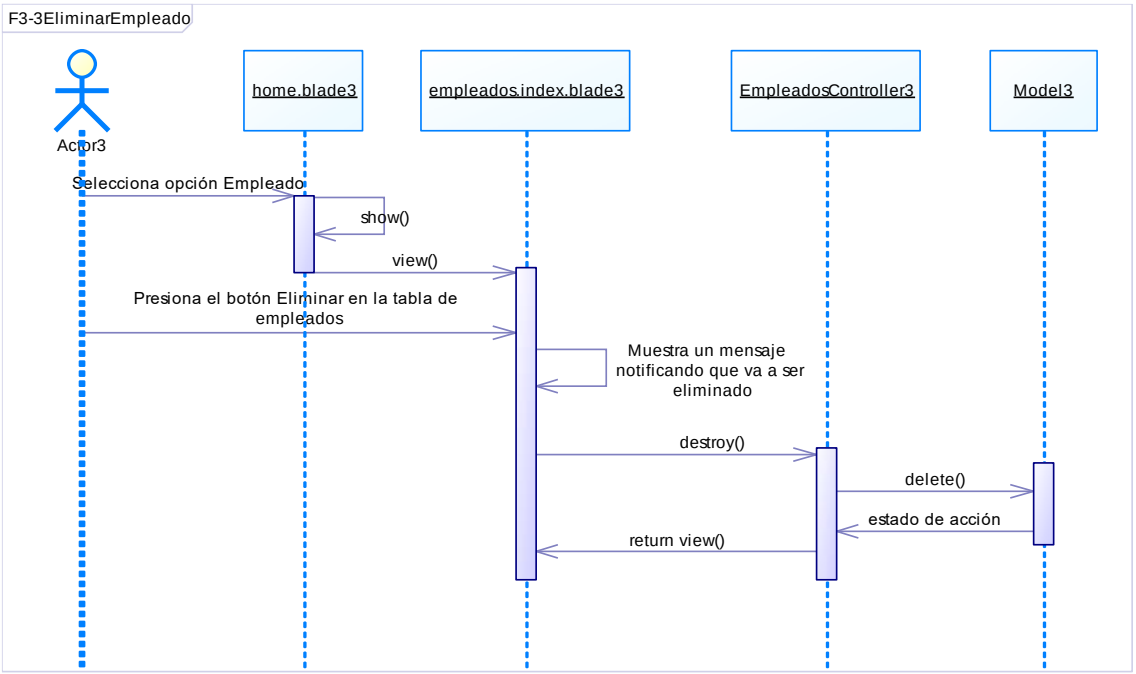
*Diagrama 30 F3.2 Modificar Empleado - Secuencia*




*Autor: David Alejandro Aráuz Moya, 2018*

## F3.3 Eliminar Empleado

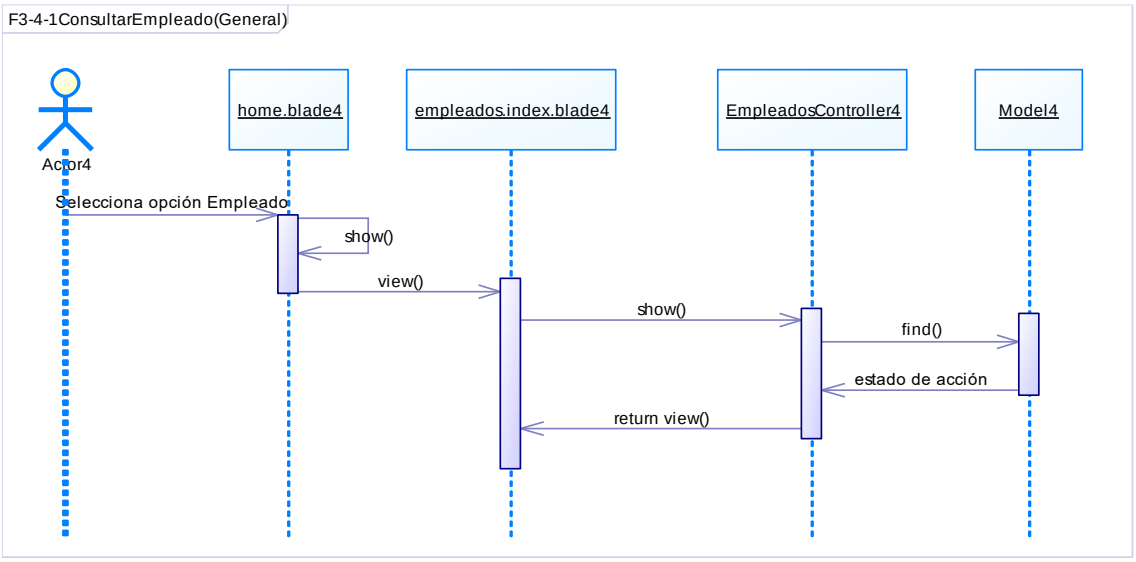
*Diagrama 31 F3.3 Eliminar Empleado - Secuencia*




*Autor: David Alejandro Aráuz Moya, 2018*

### F3.4.1 Consultar Empleado (General)

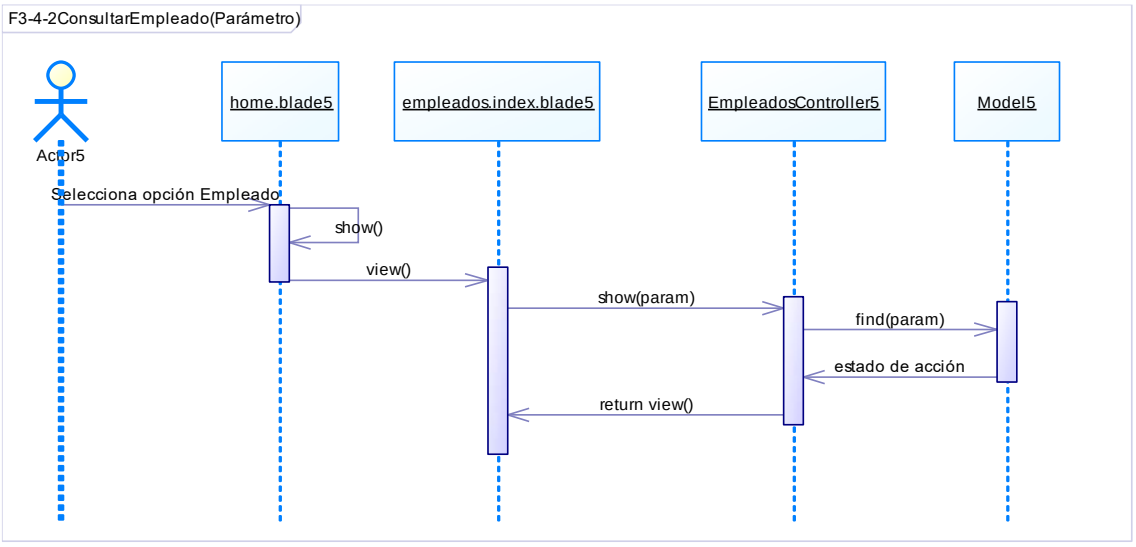
*Diagrama 32 F3.4.1 Consultar Empleado (General) - Secuencia*




*Autor: David Alejandro Aráuz Moya, 2018*

### F3.4.2 Consultar Empleado (Parámetro)

*Diagrama 33 F3.4.2 Consultar Empleado (Parámetro)- Secuencia*

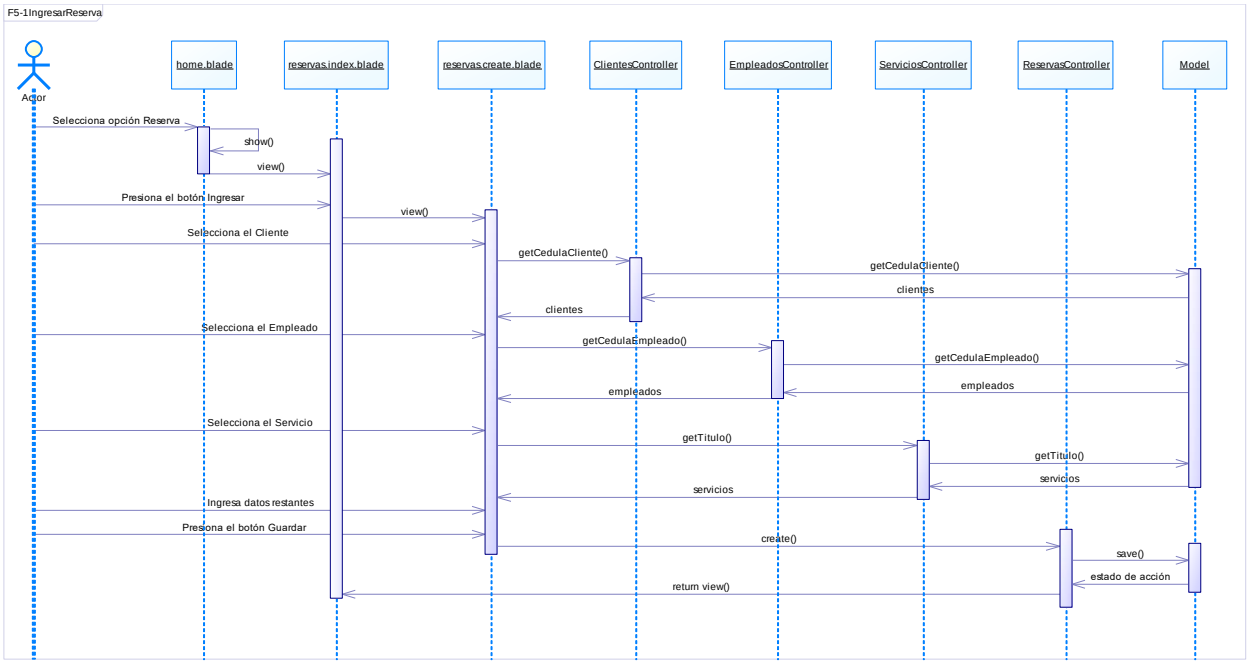



*Autor: David Alejandro Aráuz Moya, 2018*

## F5. Gestionar Reserva

### F5.1 Ingresar Reserva

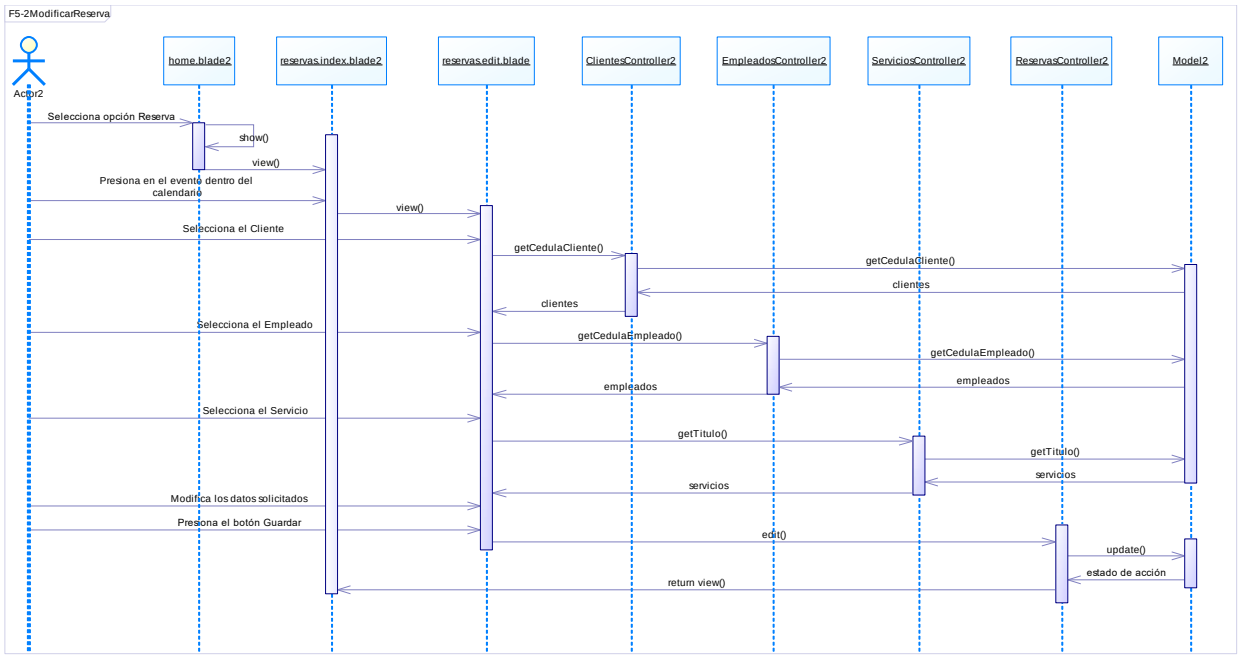
*Diagrama 34 F5.1 Ingresar Reserva - Secuencia*




*Autor: David Alejandro Aráuz Moya, 2018*

### F5.2 Modificar Reserva

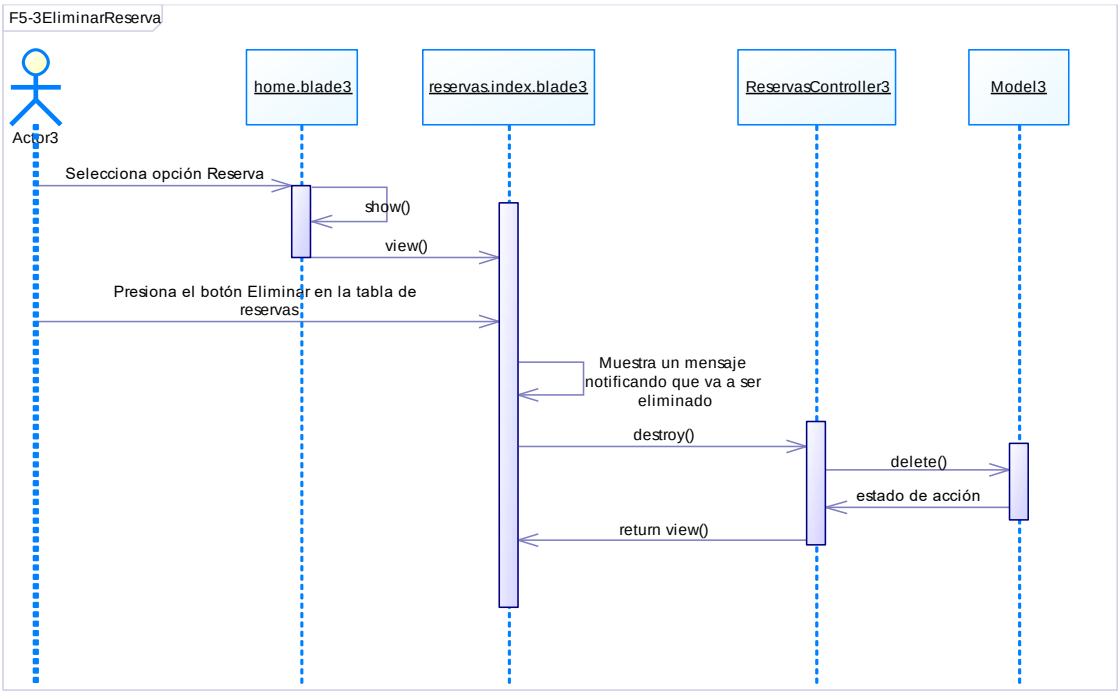
*Diagrama 35 F5.2 Modificar Reserva - Secuencia*




*Autor: David Alejandro Aráuz Moya, 2018*

## F5.3 Eliminar Reserva

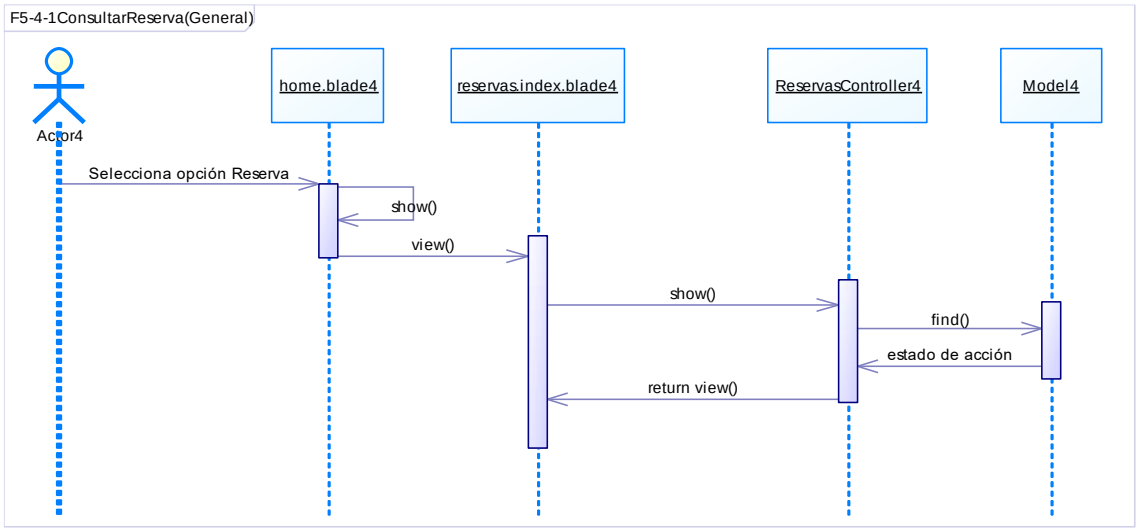
*Diagrama 36 F5.3 Eliminar Reserva - Secuencia*




*Autor: David Alejandro Aráuz Moya, 2018*

## F5.4.1 Consultar Reserva (General)

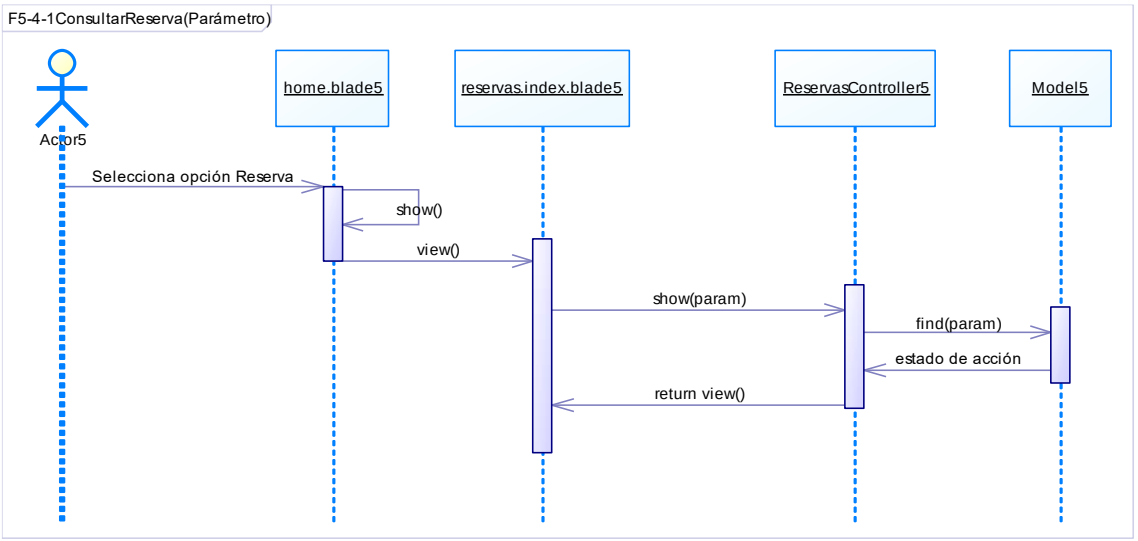
*Diagrama 37 F5.4.1 Consultar Reserva (General) - Secuencia*




*Autor: David Alejandro Aráuz Moya, 2018*

F5.4.2 Consultar Reserva (Parámetro)

*Diagrama 38 F5.4.2 Consultar Reserva (Parámetro) - Secuencia*




*Autor: David Alejandro Aráuz Moya, 2018*

## 5.2.2. Diccionario de datos

"El diccionario de datos es una lista organizada de todos los datos pertenecientes al sistema, con una serie de definiciones precisas y rigurosas para que tanto el analista como el usuario comprendan entradas, salidas, elementos de los almacenamientos y cálculos intermedios" (Gomez, 1997)

En las siguientes tablas, se indican cada uno de los campos pertenecientes a las tablas con las que cuenta el sistema, con su respectivo nombre y tipo de dato.

### 5.2.2.1. Tabla Cliente

*Tabla 28 Diccionario de datos - Tabla Cliente*

| Nombre | Tipo de Dato | Longitud | Precisión | Descripción |
|---|---|---|---|---|
| cedula | Variable characters | 15 | | Cédula del Cliente |
| cId | Integer | | | Identificador de la tabla Cliente |
| correo | Variable characters | 30 | | Correo del Cliente |
| direccion | Variable characters | 100 | | Dirección del Cliente |
| fechaNacimiento | Date | | | Fecha de nacimiento del Cliente |
| numeroCelular | Variable characters | 15 | | Número celular del Cliente |
| primerApellido | Variable characters | 30 | | Primer apellido del Cliente |
| primerNombre | Variable characters | 30 | | Primer nombre del Cliente |
| segundoApellido | Variable characters | 30 | | Segundo apellido del Cliente |
| segundoNombre | Variable characters | 30 | | Segundo nombre del Cliente |

*Autor: David Alejandro Aráuz Moya, 2018*

#### 5.2.2.2. Tabla Reserva

*Tabla 29 Diccionario de datos - Tabla Reserva*

| Nombre | Tipo de Dato | Longitud | Precisión | Descripción |
|---|---|---|---|---|
| cId | Integer | | | Identificador de la tabla Cliente |
| comentarios | Variable characters | 30 | | Comentarios de la Reserva |
| eId | Integer | | | Identificador de la tabla Empleado |
| horaFin | Time | | | Hora de finalización de la Reserva |
| horaInicio | Time | | | Hora de inicio de la Reserva |
| reId | Integer | | | Identificador de la tabla Reserva |
| sId | Integer | | | Identificador de la tabla Servicio |

*Autor: David Alejandro Aráuz Moya, 2018*

#### 5.2.2.3. Tabla Empleado

*Tabla 30 Diccionario de datos - Tabla Empleado*

| Nombre | Tipo de Dato | Longitud | Precisión | Descripción |
|---|---|---|---|---|
| cedula | Variable characters | 15 | | Cédula del Empleado |
| eId | Integer | | | Identificador de la tabla Empleado |
| correo | Variable characters | 30 | | Correo del Empleado |
| cargo | Variable characters | 30 | | Cargo del Empleado |
| fechaNacimiento | Date | | | Fecha de nacimiento del Empleado |
| numeroCelular | Variable characters | 15 | | Número celular del Empleado |
| primerApellido | Variable characters | 30 | | Primer apellido del Empleado |
| primerNombre | Variable characters | 30 | | Primer nombre del Empleado |
| segundoApellido | Variable characters | 30 | | Segundo apellido del Empleado |
| segundoNombre | Variable characters | 30 | | Segundo nombre del Empleado |
| uId | Integer | | | Identificador de la tabla Usuario |

*Autor: David Alejandro Aráuz Moya, 2018*

#### 5.2.2.4. Tabla Rol

*Tabla 31 Diccionario de datos - Tabla Rol*

| Nombre | Tipo de Dato | Longitud | Precisión | Descripción |
|---|---|---|---|---|
| rId | Integer | | | Identificador de la tabla Rol |
| titulo | Variable characters | 30 | | Título del Rol |

*Autor: David Alejandro Aráuz Moya, 2018*

#### 5.2.2.5. Tabla Usuario

*Tabla 32 Diccionario de datos - Tabla Usuario*

| Nombre | Tipo de Dato | Longitud | Precisión | Descripción |
|---|---|---|---|---|
| correo | Variable characters | 30 | | Correo del Usuario |
| passw ord | Variable characters | 60 | | Contraseña del Usuario |
| primerApellido | Variable characters | 30 | | Primer apellido del Usuario |
| primerNombre | Variable characters | 30 | | Primer nombre del Usuario |
| rId | Integer | | | Identificador de la tabla Rol |
| uId | Integer | | | Identificador de la tabla Usuario |

*Autor: David Alejandro Aráuz Moya, 2018*

#### 5.2.2.6. Tabla Servicio

*Tabla 33 Diccionario de datos - Tabla Servicio*

| Nombre | Tipo de Dato | Longitud | Precisión | Descripción |
|---|---|---|---|---|
| costo | Number | 19 | 2 | Costo del Servicio |
| sId | Integer | | | Identificador de la tabla Servicio |
| titulo | Variable characters | 30 | | Título del Servicio |

*Autor: David Alejandro Aráuz Moya, 2018*

#### 5.2.2.7. Tabla Hora Trabajo

*Tabla 34 Diccionario de datos - Tabla Hora Trabajo*

| Nombre | Tipo de Dato | Longitud | Precisión | Descripción |
|---|---|---|---|---|
| eId | Integer | | | Identificador de la tabla Empleado |
| fecha | Date | | | Fecha de la tabla Hora Trabajo |
| hId | Integer | | | Identificador de la tabla Hora Trabajo |
| horaFin | Time | | | Hora de finalización de Hora Trabajo |
| horaInicio | Time | | | Hora de inicio de Hora Trabajo |

*Autor: David Alejandro Aráuz Moya, 2018*

# CAPÍTULO 6: IMPLEMENTACIÓN DEL SISTEMA

## 6.1. Plan de implementación

Para la presente disertación se elaboró una planificación de acuerdo a las guías que brinda la metodología Extreme Programming, que está basada en ciclos conocidos como iteraciones, en las cuales intervienen tanto las funcionalidades, historias de usuarios no abordadas, pruebas integrales y del sistema y tareas no finalizadas de la iteración anterior. A continuación, se presenta la planificación de las funcionalidades por iteración y el responsable asignado para el desarrollo con tiempos de duración.

*Tabla 35 Plan de implementación*

| | | | Semana | | | | |
|---|---|---|---|---|---|---|---|
| **Iteración** | **Responsable** | **Funcionalidad** | 1 | 2 | 3 | 4 | 5 |
| 1 | David Aráuz Moya | **F1. Gestionar Usuarios** | | | | | |
| 2 | | **F2. Gestionar Clientes** | | | | | |
| 3 | | **F3. Gestionar Empleados** | | | | | |
| 4 | | **F4. Gestionar Horas de Trabajo** | | | | | |
| 5 | | **F6. Gestionar Servicios** | | | | | |
| 6 | | **F5. Gestionar Reservas** | | | | | |

*Autor: David Alejandro Aráuz Moya, 2018*

## 6.2. Pruebas del sistema

El plan de pruebas del sistema dentro de la metodología Extreme Programming, se van dando en cada una de las iteraciones para un mejor control tanto para el programador como para el usuario final, estas se dividen en 2 grupos:

Las pruebas unitarias y pruebas del sistema, en las primeras, el desarrollador es el encargado de realizar las pruebas pertinentes tanto para el código como para la funcionalidad, las segundas, son destinadas para el fin de ciclo de la iteración y comprobar si se cumplió con la funcionalidad solicitada por el usuario final.

Los casos de uso obtenidos en el capítulo anterior, generaron casos de prueba que son útiles para un mayor control y validación del sistema al momento de realizar la etapa de pruebas.

Estos casos de prueba son una guía que se complementan necesariamente con los casos de uso al momento de realizar las pruebas pertinentes.

A continuación, se presentan las pruebas funcionales o del sistema por cada funcionalidad.

**CASO DE PRUEBA**: Gestionar Usuarios (F1.0)

**PRECONDICIONES**: Ninguna

**CUADRO**

*Tabla 36 Gestionar Usuarios - Pruebas del sistema*

| Entradas | Resultados Esperados | Casos de Uso | Estado |
|---|---|---|---|
| 1. Seleccionar opción gestionar Empleado del menú principal. | Presentar pantalla de gestión de Usuario | F1 | |
| 2. Ingresar datos del Usuario<br>• Datos correctos<br>• Datos incorrectos | Datos almacenados<br>Mensaje de error apropiado | F1.1 | |
| 3. Modificar Usuario<br>• Datos correctos<br>• Datos incorrectos | Datos almacenados<br>Mensaje de error apropiado | F1.2 | |
| 4. Eliminar Usuario<br>• Ingresar cédula | Datos almacenados<br>Mensaje de error apropiado | F1.3 | |
| 5. Consulta General de Usuario | Presentar datos de los clientes | F1.4.1 | |
| 6. Consulta por parámetros de Usuario | Datos almacenados<br>Mensaje de error apropiado | F1.4.2 | |
| 7. Validación del parámetro ingresado | Error en la consulta a base de datos<br>Datos correctos en el campo | | |

*Autor: David Alejandro Aráuz Moya, 2018*

**POSTCONDICIONES**: Ninguna

**CASO DE PRUEBA**: Gestionar Clientes (F2.0)

**PRECONDICIONES**: Ejecutar 5 veces F1 como requisito para poder ejecutar F2

**CUADRO**

*Tabla 37 Gestionar Clientes - Pruebas del sistema*

| Entradas | Resultados Esperados | Casos de Uso | Estado |
|---|---|---|---|
| 1. Seleccionar opción gestionar Cliente del menú principal. | Presentar pantalla de gestión de Cliente | F2 | |
| 2. Ingresar datos del Cliente<br>• Datos correctos<br>• Datos incorrectos | Datos almacenados<br>Mensaje de error apropiado | F2.1 | |
| 3. Validación de Cédula<br>• Verificar datos ingresados | Error en la consulta a base de datos<br>Datos correctos en el campo | | |
| 4. Modificar Cliente<br>• Datos correctos<br>• Datos incorrectos | Datos almacenados<br>Mensaje de error apropiado | F2.2 | |
| 5. Eliminar Cliente<br>• Ingresar cédula | Datos almacenados<br>Mensaje de error apropiado | F2.3 | |
| 6. Consulta General de Cliente | Presentar datos de los clientes | F2.4.1 | |
| 7. Consulta por parámetros de Cliente | Datos almacenados<br>Mensaje de error apropiado | F2.4.2 | |
| 8. Validación del parámetro ingresado | Error en la consulta a base de datos<br>Datos correctos en el campo | | |

*Autor: David Alejandro Aráuz Moya, 2018*

**POSTCONDICIONES**: Ejecutar 5 veces F2 como requisito para poder ejecutar F3 (Gestionar Empleados)

**CASO DE PRUEBA**: Gestionar Empleados (F3.0)

**PRECONDICIONES**: Ejecutar 5 veces F1 como requisito para poder ejecutar F3

**CUADRO**

*Tabla 38 Gestionar Empleados - Pruebas del sistema*

| Entradas | Resultados Esperados | Casos de Uso | Estado |
|---|---|---|---|
| 1. Seleccionar opción gestionar Empleado del menú principal. | Presentar pantalla de gestión de Empleado | F3 | |
| 2. Ingresar datos del Empleado<br>• Datos correctos<br>• Datos incorrectos | Datos almacenados<br>Mensaje de error apropiado | F3.1 | |
| 3. Validación de Cédula<br>• Verificar datos ingresados | Error en la consulta a base de datos<br>Datos correctos en el campo | | |
| 4. Modificar Empleado<br>• Datos correctos<br>• Datos incorrectos | Datos almacenados<br>Mensaje de error apropiado | F3.2 | |
| 5. Eliminar Empleado<br>• Ingresar cédula | Datos almacenados<br>Mensaje de error apropiado | F3.3 | |
| 6. Consulta General de Empleado | Presentar datos de los clientes | F3.4.1 | |
| 7. Consulta por parámetros de Empleado | Datos almacenados<br>Mensaje de error apropiado | F3.4.2 | |
| 8. Validación del parámetro ingresado | Error en la consulta a base de datos<br>Datos correctos en el campo | | |

*Autor: David Alejandro Aráuz Moya, 2018*

**POSTCONDICIONES**: Ejecutar 5 veces F3 como requisito para poder ejecutar F4

**CASO DE PRUEBA**: Gestionar Horas de Trabajo (F4.0)

**PRECONDICIONES**: Ejecutar 5 veces F3 como requisito para poder ejecutar F4

**CUADRO**

*Tabla 39 Gestionar Horas de trabajo - Pruebas del sistema*

| Entradas | Resultados Esperados | Casos de Uso | Estado |
|---|---|---|---|
| 1. Seleccionar opción gestionar Horas de trabajo del menú principal. | Presentar pantalla de gestión de Horas de trabajo | F4 | |
| 2. Ingresar datos de las horas de trabajo<br>• Datos correctos<br>• Datos incorrectos | Datos almacenados<br>Mensaje de error apropiado | F4.1 | |
| 3. Modificar horas de trabajo<br>• Datos correctos<br>• Datos incorrectos | Datos almacenados<br>Mensaje de error apropiado | F4.2 | |
| 4. Eliminar horas de trabajo<br>• Ingresar cédula | Datos almacenados<br>Mensaje de error apropiado | F4.3 | |
| 5. Consulta General de horas de trabajo | Presentar datos de los clientes | F4.4.1 | |
| 6. Consulta por parámetros de horas de trabajo | Datos almacenados<br>Mensaje de error apropiado | F4.4.2 | |
| 7. Validación del parámetro ingresado | Error en la consulta a base de datos<br>Datos correctos en el campo | | |

*Autor: David Alejandro Aráuz Moya, 2018*

**POSTCONDICIONES**: Ejecutar 5 veces F3 como requisito para poder ejecutar F5

**CASO DE PRUEBA**: Gestionar Reservas (F5.0)

**PRECONDICIONES**: Existencia de Servicios, Empleados, Servicios

**CUADRO**

*Tabla 40 Gestionar Reservas - Pruebas del sistema*

| Entradas | Resultados Esperados | Casos de Uso | Estado |
|---|---|---|---|
| 1. Seleccionar opción gestionar Reserva del menú principal. | Presentar pantalla de gestión de Reserva | F5 | |
| 2. Ingresar datos de la Reserva<br>• Datos correctos<br>• Datos incorrectos | Datos almacenados<br>Mensaje de error apropiado | F5.1 | |
| 3. Validación de Cédula<br>• Verificar datos | Error en la consulta a base de datos<br>Datos correctos en el campo | | |
| 4. Modificar Reserva<br>• Datos correctos<br>• Datos incorrectos | Datos almacenados<br>Mensaje de error apropiado | F5.2 | |
| 5. Eliminar Reserva<br>• Ingresar cédula | Datos almacenados<br>Mensaje de error apropiado | F5.3 | |
| 6. Consulta General de Reserva | Presentar datos de los clientes | F5.4.1 | |
| 7. Consulta por parámetros de Reserva | Datos almacenados<br>Mensaje de error apropiado | F5.4.2 | |
| 8. Validación del parámetro ingresado | Error en la consulta a base de datos<br>Datos correctos en el campo | | |

*Autor: David Alejandro Aráuz Moya, 2018*

**POSTCONDICIONES**: Ninguna

**CASO DE PRUEBA**: Gestionar Servicios (F6.0)

**PRECONDICIONES**: Ninguna

**CUADRO**

*Tabla 41 Gestionar Servicios - Pruebas del sistema*

| Entradas | Resultados Esperados | Casos de Uso | Estado |
|---|---|---|---|
| 1. Seleccionar opción gestionar Servicios del menú principal. | Presentar pantalla de gestión de Servicios | F6 | |
| 2. Ingresar datos del Servicio<br>• Datos correctos<br>• Datos incorrectos | Datos almacenados<br>Mensaje de error apropiado | F6.1 | |
| 3. Modificar Servicio<br>• Datos correctos<br>• Datos incorrectos | Datos almacenados<br>Mensaje de error apropiado | F6.2 | |
| 4. Eliminar Servicio<br>• Ingresar cédula | Datos almacenados<br>Mensaje de error apropiado | F6.3 | |
| 5. Consulta General de Servicio | Presentar datos de los clientes | F6.4.1 | |
| 6. Consulta por parámetros de Servicio | Datos almacenados<br>Mensaje de error apropiado | F6.4.2 | |
| 7. Validación del parámetro ingresado | Error en la consulta a base de datos<br>Datos correctos en el campo | | |

**POSTCONDICIONES**: Ejecutar 5 veces F6 como requisito para poder ejecutar F5

## 6.3. Implantación del sistema

Para seguir con el modelo de evidencia presentado para las clases básicas en la presente disertación y por la similitud que presentan entre sí, se muestra fragmentos de código que contienen las partes más fundamentales de las funcionalidades para el desarrollo del proyecto, el código faltante se lo adjuntará en formato digital dentro del cd correspondiente.

En la siguiente ilustración se muestra el código perteneciente a la vista de clientes

```
@inject('request', 'Illuminate\Http\Request')
@extends('layouts.app')

<!--
    Titulo y botón de Ingresar nuevo cliente
-->
@section('content')
    <h3 class="page-title">('cliente.title')</h3>
    @can('cliente_create')
    <p>
        <a href="{{ route('admin.clientes.create') }}" class="btn btn-success">(qa_add_new')</a>

    </p>
    @endcan

    <div class="panel panel-default">
        <div class="panel-heading">
            ('qa_list')
        </div>
<!--
    Muestra las cabeceras de las columnas
-->
        <div class="panel-body table-responsive">
            <table class="table table-bordered table-striped {{ count($clientes) > 0 ? 'datatable' : '' }}
                    @can('cliente_delete') @if ( request('show_deleted') != 1 ) dt-select @endif @endcan">
                <thead>
                    <tr>
                        @can('cliente_delete')
                            @if ( request('show_deleted') != 1 )<th style="text-align:center;"><input type="checkbox" id="select-all" /></th>@endif
                        @endcan

                        <th>('cliente.fields.cedula')</th>
                        <th>('cliente.fields.primernombre')</th>
                        <th>('cliente.fields.segundonombre')</th>
                        <th>('cliente.fields.primerapellido')</th>
                        <th>('cliente.fields.segundoapellido')</th>
                        <th>('cliente.fields.fechanacimiento')</th>
                        <th>('cliente.fields.numerocelular')</th>
                        <th>('cliente.fields.correo')</th>
                        <th>('cliente.fields.direccion')</th>
```

*Ilustración 11 Vista Cliente 1*

*Fuente: David Alejandro Aráuz Moya, 2018*

```
                        @if( request('show_deleted') == 1 )
                        <th> </th>
                        @else
                        <th> </th>
                        @endif
                    </tr>
                </thead>
<!--
    Muestra el detalle de la tabla si es que existen registro, caso contrario muestra la tabla vacia
-->
                <tbody>
                    @if (count($clientes) > 0)
                        @foreach ($clientes as $cliente)
                            <tr data-entry-id="{{ $cliente->id }}">
                                @can('cliente_delete')
                                    @if ( request('show_deleted') != 1 )<td></td>@endif
                                @endcan

                                <td field-key='cedula'>{{ $cliente->cedula }}</td>
                                <td field-key='primernombre'>{{ $cliente->primernombre }}</td>
                                <td field-key='segundonombre'>{{ $cliente->segundonombre }}</td>
                                <td field-key='primerapellido'>{{ $cliente->primerapellido }}</td>
                                <td field-key='segundoapellido'>{{ $cliente->segundoapellido }}</td>
                                <td field-key='fechanacimiento'>{{ $cliente->fechanacimiento }}</td>
                                <td field-key='numerocelular'>{{ $cliente->numerocelular }}</td>
                                <td field-key='correo'>{{ $cliente->correo }}</td>
                                <td field-key='direccion'>{{ $cliente->direccion }}</td>
                                @if( request('show_deleted') == 1 )
                                <td>
                                    @can('cliente_delete')
                                                                    {!! Form::open(array(
                                        'style' => 'display: inline-block;',
                                        'method' => 'POST',
                                        'onsubmit' => "return confirm('".trans("qa_are_you_sure")."');",
                                        'route' => ['admin.clientes.restore', $cliente->id])) !!}
                                    {!! Form::submit(trans('qa_restore'), array('class' => 'btn btn-xs btn-success')) !!}
                                    {!! Form::close() !!}
                                    @endcan
```

*Ilustración 12 Vista Cliente 2*

*Fuente: David Alejandro Aráuz Moya, 2018*

```
                                    @can('cliente_delete')
                                                                    {!! Form::open(array(
                                        'style' => 'display: inline-block;',
                                        'method' => 'DELETE',
                                        'onsubmit' => "return confirm('".trans("qa_are_you_sure")."');",
                                        'route' => ['admin.clientes.perma_del', $cliente->id])) !!}
                                    {!! Form::submit(trans('qa_permadel'), array('class' => 'btn btn-xs btn-danger')) !!}
                                    {!! Form::close() !!}
                                @endcan
                                </td>
                                @else
                                <td>
                                    @can('cliente_view')
                                    <a href="{{ route('admin.clientes.show',[$cliente->id]) }}" class="btn btn-xs btn-primary">('qa_view')</a>
                                    @endcan
                                    @can('cliente_edit')
                                    <a href="{{ route('admin.clientes.edit',[$cliente->id]) }}" class="btn btn-xs btn-info">('qa_edit')</a>
                                    @endcan
                                    @can('cliente_delete')
{!! Form::open(array(
                                        'style' => 'display: inline-block;',
                                        'method' => 'DELETE',
                                        'onsubmit' => "return confirm('".trans("qa_are_you_sure")."');",
                                        'route' => ['admin.clientes.destroy', $cliente->id])) !!}
                                    {!! Form::submit(trans('quickadmin.qa_delete'), array('class' => 'btn btn-xs btn-danger')) !!}
                                    {!! Form::close() !!}
                                    @endcan
                                </td>
                                @endif
                            </tr>
                        @endforeach
                    @else
                        <tr>
                            <td colspan="14">('qa_no_entries_in_table')</td>
                        </tr>
                    @endif
                </tbody>
            </table>
        </div>
```

*Ilustración 13 Vista Cliente 3*

*Fuente: David Alejandro Aráuz Moya, 2018*

En la ilustración presentada a continuación, se presenta la vista para crear a los clientes, aquí se realiza las validaciones correspondientes tanto para cédulas ya ingresadas, como para los campos obligatorios que no están llenos

```
@extends('layouts.app')
<!--
    Título de la pantalla de creación del cliente
-->
@section('content')
    <h3 class="page-title">('cliente.title')</h3>
    {!! Form::open(['method' => 'POST', 'route' => ['admin.clientes.store']]) !!}

    <div class="panel panel-default">
        <div class="panel-heading">
            ('qa_create')
        </div>
<!--
    Labels y textbox del ingreso del cliente, contiene validaciones si es que ya existe una cédula o no está ingresado un campo
-->
        <div class="panel-body">
            <div class="row">
                <div class="col-xs-12 form-group">
                    {!! Form::label('cedula', trans('cliente.fields.cedula').'*', ['class' => 'control-label']) !!}
                    {!! Form::text('cedula', old('cedula'), ['class' => 'form-control', 'placeholder' => '', 'required' => '']) !!}
                    <p class="help-block"></p>
                    @if($errors->has('cedula'))
                        <p class="help-block">
                            {{ $errors->first('cedula') }}
                        </p>
                    @endif
                </div>
            </div>
            <div class="row">
                <div class="col-xs-12 form-group">
                    {!! Form::label('primernombre', trans('cliente.fields.primernombre').'*', ['class' => 'control-label']) !!}
                    {!! Form::text('primernombre', old('primernombre'), ['class' => 'form-control', 'placeholder' => '', 'required' => '']) !!}
                    <p class="help-block"></p>
                    @if($errors->has('primernombre'))
                        <p class="help-block">
                            {{ $errors->first('primernombre') }}
                        </p>
                    @endif
                </div>
            </div>
```

*Ilustración 14 Vista crear Cliente 1*

*Fuente: David Alejandro Aráuz Moya, 2018*

```
            <div class="row">
                <div class="col-xs-12 form-group">
                    {!! Form::label('segundonombre', trans('cliente.fields.segundonombre').'*', ['class' => 'control-label']) !!}
                    {!! Form::text('segundonombre', old('segundonombre'), ['class' => 'form-control', 'placeholder' => '', 'required' => '']) !!}
                    <p class="help-block"></p>
                    @if($errors->has('segundonombre'))
                        <p class="help-block">
                            {{ $errors->first('segundonombre') }}
                        </p>
                    @endif
                </div>
            </div>
            <div class="row">
                <div class="col-xs-12 form-group">
                    {!! Form::label('primerapellido', trans('cliente.fields.primerapellido').'*', ['class' => 'control-label']) !!}
                    {!! Form::text('primerapellido', old('primerapellido'), ['class' => 'form-control', 'placeholder' => '', 'required' => '']) !!}
                    <p class="help-block"></p>
                    @if($errors->has('primerapellido'))
                        <p class="help-block">
                            {{ $errors->first('primerapellido') }}
                        </p>
                    @endif
                </div>
            </div>
            <div class="row">
                <div class="col-xs-12 form-group">
                    {!! Form::label('segundoapellido', trans('cliente.fields.segundoapellido').'*', ['class' => 'control-label']) !!}
                    {!! Form::text('segundoapellido', old('segundoapellido'), ['class' => 'form-control', 'placeholder' => '', 'required' => '']) !!}
                    <p class="help-block"></p>
                    @if($errors->has('segundoapellido'))
                        <p class="help-block">
                            {{ $errors->first('segundoapellido') }}
                        </p>
                    @endif
                </div>
            </div>
```

*Ilustración 15 Vista crear Cliente 2*

*Fuente: David Alejandro Aráuz Moya, 2018*

```
113             <div class="row">
114                 <div class="col-xs-12 form-group">
115                     {!! Form::label('direccion', trans('cliente.fields.direccion').'', ['class' => 'control-label']) !!}
116                     {!! Form::text('direccion', old('direccion'), ['class' => 'form-control', 'placeholder' => '']) !!}
117                     <p class="help-block"></p>
118                     @if($errors->has('direccion'))
119                         <p class="help-block">
120                             {{ $errors->first('direccion') }}
121                         </p>
122                     @endif
123                 </div>
124             </div>
125 
126         </div>
127     </div>
128 
129     {!! Form::submit(trans('qa_save'), ['class' => 'btn btn-danger']) !!}
130     {!! Form::close() !!}
131 @stop
132 
133 @section('javascript')
134     @parent
135 
136     <script src="{{ url('adminlte/plugins/datetimepicker/moment-with-locales.min.js') }}"></script>
137     <script src="{{ url('adminlte/plugins/datetimepicker/bootstrap-datetimepicker.min.js') }}"></script>
138     <script>
139         $(function(){
140             moment.updateLocale('{{ App::getLocale() }}', {
141                 week: { dow: 1 } // Lunes es el primer día del mes
142             });
143 
144             $('.date').datetimepicker({
145                 format: "{{ config('app.date_format_moment') }}",
146                 locale: "{{ App::getLocale() }}",
147             });
148 
149         });
150     </script>
151 
152 @stop
```

*Ilustración 16 Vista crear Cliente 3*

*Fuente: David Alejandro Aráuz Moya, 2018*

La siguiente ilustración presenta el código del controlador del cliente, que establece la comunicación entre la vista y el modelo.

```php
<?php

namespace App\Http\Controllers\Admin;

use App\Cliente;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Gate;
use App\Http\Controllers\Controller;
use App\Http\Requests\Admin\StoreClientesRequest;
use App\Http\Requests\Admin\UpdateClientesRequest;

class ClientesController extends Controller
{
    /**
     * Muetsra la lista de Clientes.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        if (! Gate::allows('cliente_access')) {
            return abort(401);
        }


        if (request('show_deleted') == 1) {
            if (! Gate::allows('cliente_delete')) {
                return abort(401);
            }
            $clientes = Cliente::onlyTrashed()->get();
        } else {
            $clientes = Cliente::all();
        }

        return view('admin.clientes.index', compact('clientes'));
    }
```

*Ilustración 17 Controlador Cliente 1*

*Fuente: David Alejandro Aráuz Moya, 2018*

```php
    /**
     * Muestra el formulario para crear un nuevo Cliente.
     *
     * @return \Illuminate\Http\Response
     */
    public function create()
    {
        if (! Gate::allows('cliente_create')) {
            return abort(401);
        }
        return view('admin.clientes.create');
    }

    /**
     * Muestra el formulario para editar un Cliente.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function edit($id)
    {
        if (! Gate::allows('cliente_edit')) {
            return abort(401);
        }
        $cliente = Cliente::findOrFail($id);

        return view('admin.clientes.edit', compact('cliente'));
    }
```

*Ilustración 18 Controlador Cliente 2*

*Fuente: David Alejandro Aráuz Moya, 2018*

```php
    /**
     * Muestra el Cliente.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function show($id)
    {
        if (! Gate::allows('cliente_view')) {
            return abort(401);
        }
        $reservas = \App\Reserva::where('cliente_id', $id)->get();

        $cliente = Cliente::findOrFail($id);

        return view('admin.clientes.show', compact('cliente', 'reservas'));
    }


    /**
     *Elimina el Cliente.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function destroy($id)
    {
        if (! Gate::allows('cliente_delete')) {
            return abort(401);
        }
        $cliente = Cliente::findOrFail($id);
        $cliente->delete();

        return redirect()->route('admin.clientes.index');
    }
```

*Ilustración 19 Controlador Cliente 3*

*Fuente: David Alejandro Aráuz Moya, 2018*

```php
    /**
     * Elimina todos los clientes seleccionados
     *
     * @param Request $request
     */
    public function massDestroy(Request $request)
    {
        if (! Gate::allows('cliente_delete')) {
            return abort(401);
        }
        if ($request->input('ids')) {
            $entries = Cliente::whereIn('id', $request->input('ids'))->get();

            foreach ($entries as $entry) {
                $entry->delete();
            }
        }
    }
}

```

*Ilustración 20 Controlador Cliente 4*

*Fuente: David Alejandro Aráuz Moya, 2018*

El código faltante del proyecto de disertación se encuentra ubicados en los anexos dentro del CD.

Una vez concluido con la codificación y la etapa de pruebas, se procedió a realizar la negociación con la empresa para empezar con la implantación y la entrada a producción del sistema, sin embargo, por temas netamente de administración se realizó un cambio

de dueños en la empresa, y los nuevos encargados decidieron no poner el sistema en producción por el momento, hasta que se realicen los trámites respectivos.

Según lo conversado la estética cerraría por 2 meses ya que van a cambiar de oficinas, después de esto, para el cierre del primer trimestre del año 2019, se podría entrar en negociación para la implantación del sistema.

A continuación, se presenta un esquema de implementación después de las pruebas unitarias y funcionales ya realizadas en Cadama Estética, esto se lo va a ejecutar para terminar el ciclo del proyecto propuesto por la metodología Extreme Programming.

### 6.3.1. Capacitación

Para realizar este proceso, la persona encargada de capacitar a las personas debe tener un conocimiento total acerca de la aplicación web. Por este motivo se realizó un acta de capacitación para tener constancia de lo realizado y contar con un sustento para que no surja ningún inconveniente.

**ACTA DE CAPACITACIÓN #**

| | |
|---|---|
| **FECHA:** | |
| **ASUNTO:** | |
| **LUGAR:** | |
| **HORA:** | |

| | |
|---|---|
| **CONVOCANTE:** | |
| **MOTIVACIÓN:** | |

**Participantes:**

| # | NOMBRE | CARGO / ROL |
|---|---|---|
| 1 | | |
| 2 | | |
| 3 | | |
| 4 | | |
| 5 | | |

**Temas tratados en la capacitación:**

| DESCRIPCIÓN DEL TEMA TRATADO | |
|---|---|
| | **TIEMPO CAPACITACIÓN:**<br>**# PERSONAL CAPACITADO:**<br>**TEMA:** |

**Control de Asistencia:**

Para formalizar suscriben por la presente acta los siguientes participantes:

| # | NOMBRE | CARGO / ROL | FIRMAS |
|---|---|---|---|
| 1 | | | |
| 2 | | | |
| 3 | | | |
| 4 | | | |

### 6.3.2. Acompañamiento

Para cumplir con la planificación establecida para el proyecto, se realizará el acompañamiento del personal al finalizar la capacitación y al momento de poner en producción el proyecto.

Al no ser un proyecto muy amplio, según lo estimado, este acompañamiento se lo realizará de 3 a 4 días hasta que el personal no tenga dudas y sepa cómo manejar el sistema de una manera fluida para cumplir con los requerimientos pedidos.

# CAPITULO 7: CONCLUSIONES Y RECOMENDACIONES

## 7.1. Conclusiones

Cadama Estética, al ser una empresa que ha sido constituida en el año 2017, ha visto la necesidad de implementar una herramienta que le permita gestionar de mejor manera las citas fortaleciendo sus procesos internos, y de esta manera mantener la fidelidad de sus clientes. Es por esto que es necesario utilizar metodologías orientadas al desarrollo de software que permita generar un aplicativo que contemple todos los requerimientos de Cadama con eficiencia, eficacia y calidad, como lo son Extreme Programming (Programación Extrema) y Rational Unified Process (Proceso Unificado de Rational).

El implementar un sistema de gestión de citas no es un tema que se deba tomar a la ligera. Sin un control de los procesos de implementación, acarrearía muchos problemas al entregar el producto al cliente final como lo es la falta de calidad, es por esto que es necesario la utilización de metodologías de desarrollo ágil basados en desarrollo iterativo e incremental, ya que no se está exento de los cambios recurrentes y evolutivos de los requisitos y las soluciones, con el objetivo fundamental de mejorar, optimizar procesos y ofrecer mejor calidad del aplicativo.

Las metodologías de desarrollo Extreme Programming (Programación Extrema) y Rational Unified Process (Proceso Unificado de Rational), son grandes referentes en el mercado, brindando control de los procesos de desarrollo como se requiere para la implementación de un software de gestión de citas para Cadama, es por esto fue necesario realizar una comparativa entre ambas metodologías donde se concluye que la más apropiada para ser implementada en el desarrollo de la aplicación es XP(Extreme Programming) debido a que es una metodología ágil orientada a obtener rápidos resultados y a la satisfacción del cliente final, tendiendo a la simplicidad y mejora continua del aplicativo, permitiendo una mejora de la gestión del proyecto con entregas tempranas y con valor.

Extreme Programming propone realizar el levantamiento de requerimientos funcionales mediante Historias de Usuarios, entendiendo que la base fundamental del desarrollo de la solución es conocer las necesidades del usuario de una manera más personalizada, por lo que se concluye que esta herramienta facilitó, y ayudó en el proceso de levantamiento de requerimientos de Cadama Estética, debido a que permitió detallar las

actividades de cada funcionalidad que intervinieron en la solución propuesta, para mejor entendimiento y control del desarrollador.

Luego de la etapa de levantamiento de información es necesario diseñar los planos de construcción del proyecto, es por esto que es concluyente que para iniciar con el desarrollo de la solución propuesta es necesario utilizar herramientas ampliamente conocidas que permitan diseñar la arquitectura del sistema como lo son los diagramas UML (lenguaje unificado de modelado), logrando forjar un lenguaje visual común entre el cliente y el desarrollador.

Se concluye que la etapa de implementación es una fase crítica en el plan de proyecto, por lo que es necesario detallar las iteraciones que intervienen a lo largo del desarrollo, especificando el tiempo y los responsables de cada funcionalidad, para llevar un control y poder dar seguimiento a cada una de ellas, y de esta manera tomar decisiones estratégicas en caso de encontrarse algún inconveniente presentado.

## 7.2. Recomendaciones

Debido que Cadama actualmente realiza procesos manuales para el manejo y gestión de citas, ocasionado la pérdida de información por la deficiencia de sus procesos y la falta de una herramienta que le permita el mejor manejo de la misma, se recomienda la implementación de un software robusto que permita el ingreso, modificación, y eliminación de citas en tiempo real.

Es recomendable la implementación de metodologías ágiles o tradicionales para el desarrollo de software ya que aumenta la productividad, mejora la gestión del riesgo, simplifica el manejo de la sobrecarga de procesos entre otras cosas, como lo son Extreme Programming (Programación Extrema) siendo idónea para proyectos con requerimientos escuetos y muy variantes, en el que puede existir un alto riesgo por las variaciones constantes realizadas al momento de desarrollar la aplicación, y Rational Unified Process (Proceso Unificado de Rational) que es recomendable para el desarrollo de software en proyectos de alta magnitud, a través de una proceso continuo de pruebas y retroalimentación, certificando que los estándares de calidad se cumplan.

Extreme Programming al ser una metodología ágil permite una programación mucho más organizada, eficiencia en los procesos de planificación y pruebas, su tasa de

errores es muy pequeña, facilita los cambios, es escalable y flexible, además es una de las metodologías más utilizadas en la implementación de nuevas tecnologías, es por esto que es recomendable utilizar dicha metodología para la implementación de este proyecto de disertación.

Es recomendable utilizar herramientas que permitan realizar un levantamiento de requerimientos funcionales que detalle de manera clara los procesos que intervienen en lo solicitado por el cliente para que de esta manera el desarrollador tenga una visión clara no solo del tiempo que conlleva el desarrollo de la solución sino también de lo que el cliente espera de cada funcionalidad, es por esto que se recomienda el uso de Historias de Usuario de Extreme Programming, ya que es una herramienta que detalla claramente las especificaciones de la solución permitiendo que el desarrollador tenga un entendimiento claro antes de la etapa de desarrollo.

Es recomendable la utilización de normas y estándares gráficos que permitan diseñar la arquitectura base de la solución propuesta de manera clara y concisa, por lo que UML al ser un lenguaje gráfico permite establecer los requerimientos del cliente y las estructuras necesarias para plasmar la solución propuesta previo a la etapa de desarrollo de software ya que facilita la comunicación entre los analistas de la aplicación y los conocedores de las reglas de negocio.

Es recomendable realizar la evaluación de otras metodologías de desarrollo de software que puedan basarse en las variables sustentadas en el presente trabajo para mejorar y simplificar la elección de una metodología al momento de iniciar un proyecto.

## BIBLIOGRAFÍA


- ✓ *Arquitectura Cliente Servidor*. (s.f.). Obtenido de EcuRed: https://www.ecured.cu/Arquitectura_Cliente_Servidor
- ✓ Baquero García, J. M. (11 de Diciembre de 2015). *Blog de Arsys*. Obtenido de Arsys: https://www.arsys.es/blog/programacion/que-es-laravel/
- ✓ Beck, K. (1999). *Iteration*. Obtenido de Extreme Programming: http://www.extremeprogramming.org/map/iteration.html

- ✓ Carrillo Ledesma, A. (s.f.). *Polimorfismo*. Obtenido de Grupo de Geofísica Computacional: http://mmc.geofisica.unam.mx/acl/MaterialCursos/POO-Java/POO_2.pdf
- ✓ Cubillos, C. (s.f.). *Arquitectura Cliente/Servidor*. Obtenido de Escuela Ingeniería Industrial Pontificia Universidad Católica de Valparaiso: http://ocw.pucv.cl/cursos-1/arquitectura-de-sistemas-de-software/materiales-de-clases/web-cliente-servidor
- ✓ Gómez, J. (10 de Mayo de 2001). *Metodología orientada al objeto*. Obtenido de Universidad de Aliante: ftp://www.dlsi.ua.es/people/jaime/apuntes/isi_tema3.1.pdf
- ✓ Gómez, R. (11 de Noviembre de 2015). *¿Qué es MVC?* Obtenido de Modelo Vista Controlador: http://rodrigogr.com/blog/modelo-vista-controlador/
- ✓ Gómez, V. (23 de Junio de 2015). *Arquitectura en Tres Capas*. Obtenido de Instinto Binario: https://instintobinario.com/arquitectura-en-tres-capas/
- ✓ González, A. (s.f.). *¿Qué es MySQL*. Obtenido de Tu Programación: http://www.tuprogramacion.com/glosario/que-es-mysql/
- ✓ Hernandez, U. (s.f.). *MVC (Model, View, Controller) explicado*. Obtenido de Código Facilito: https://codigofacilito.com/articulos/mvc-model-view-controller-explicado
- ✓ Jacobson, I., Booch, G., & Rumbaugh, J. (2000). El Proceso Unificado de Desarrollo de Software. Madrid: Pearson.
- ✓ Letelier Torres, P., & Sánchez López, E. (12 de Noviembre de 2003). *Metodologías Ágiles en el Desarrollo de Software*. Obtenido de Universidad Politécnica de Valencia: http://issi.dsic.upv.es/archives/f-1069167248521/actas.pdf
- ✓ *Metodologías de desarrollo de software*. (30 de Diciembre de 2006). Obtenido de Universidad de Murcia: http://www.um.es/docencia/barzana/IAGP/Iagp2.html
- ✓ *Modelo de Tres Capas*. (2013). Obtenido de NewComLab: http://www.newcomlab.com/default.aspx?id_seccion=936
- ✓ *Modelo RUP*. (27 de Noviembre de 2012). Obtenido de Competencia de Ingeniería de Software: https://compisw.wordpress.com/2012/11/27/rup/
- ✓ Rouse, M. (Enero de 2015). *MySQL*. Obtenido de TechTarget: https://searchdatacenter.techtarget.com/es/definicion/MySQL
- ✓ Valdés, J. (01 de Marzo de 2016). *Tarea 5: Arquitectura cliente servidor*. Obtenido de Tareas de Programación y Servicios Web:

http://tareaspswjessicavaldes.blogspot.com/2016/03/tarea-5-cliente-servidor.html

- ✓ Welling, L., & Thomson, L. (2005). *Desarrollo Web con PHP y MySQL.* Madrid: Ediciones Anaya.