PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR
"SEDE ESMERALDA"



ESCUELA DE INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

TESIS DE GRADO

GENERACIÓN DE UNA APP WEB DIRIGIDA POR MODELOS

PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO DE SISTEMAS Y COMPUTACIÓN

PROGRAMACIÓN Y DESARROLLO DE SOFTWARE

AUTORA:

PACHAY QUIÑÓNEZ MARIANA ELIZABETH

ASESOR:

MSC. JAIME PAÚL SAYAGO HEREDIA

ESMERALDAS – MARZO 2019

TRIBUNAL DE GRADUACIÓN

Trabajo de tesis aprobado luego de haber dado cumplimiento a los requisitos exigidos por el reglamento de grado de la PUCESE previo a la obtención del título de Ingeniero en Sistemas y Computación.

.....

MSc. Mtr. Jaime Paúl Sayago Heredia
DIRECTOR DE DISERTACIÓN

.....

Mgt. José Luis Carvajal Carvajal
LECTOR 1

.....

Ing. Evelin Lorena Flores García
LECTOR 2

.....

MSc. Xavier Quiñones Ku
DIRECTOR DE LA ESCUELA DE INGENIERÍA EN SISTEMA Y COMPUTACIÓN

AUTORÍA

Yo, **PACHAY QUIÑÓNEZ MARIANA ELIZABETH**, declaro que la presente investigación titulada **Generación de una APP WEB Dirigida por Modelos,** la cual, está enmarcada en una tesis de grado es absolutamente original, autentica y personal.

En virtud el contenido de esta investigación es de exclusiva responsabilidad legal y académica de la autora y de la Pontificia Universidad Católica del Ecuador "Sede Esmeraldas".

.....
Pachay Quiñónez Mariana Elizabeth
C.I: 0926626789

DEDICATORIA

Mi tesis la dedico a mis hermanos y primos diciéndoles que con esfuerzo, empeño y perseverancia se alcanzan las metas anheladas. A mis padres, aunque no están a mi lado siempre confiaron en mí. A mi amada Madre y Abuela por sus consejos y motivación para seguir luchando. A mi amiga Johanna Vélez quien con sus palabras de aliento no me dejaba decaer y me impulsaba a seguir adelante.

Infinitamente a Dios por cada día de éxitos y nuevas vicisitudes de la vida permitiéndome afrontarlas con respeto y humildad.

AGRADECIMIENTO

Agradezco a Dios, a los maestros que ayudaron a formarme como profesional. A mi Mami Marianita Klinger Ortiz que con su manera dura de decir las cosas siempre empleo un consejo para levantarme y seguir adelante.

Agradecida con mi amiga Johanna Vélez de la Cruz por ser esa mano de apoyo, por tus consejos, regaños, alegrías y por sobre todo siempre estar a mi lado brindándome tu cariño incondicional.

ÍNDICE DEL CONTENIDO

RESUMEN x
ABSTRACT xi
INTRODUCCIÓN 12
PRESENTACIÓN DEL TEMA DE INVESTIGACIÓN 12
PLANTEAMIENTO DEL PROBLEMA 13
JUSTIFICACIÓN 14
OBJETIVOS 16
GENERAL 16
ESPECÍFICOS 16
CAPÍTULO I: MARCO TEÓRICO 17
1.1. BASES TEÓRICO - CIENTÍFICAS 17
1.1.1. ARQUITECTURA WEB 17
1.1.2. LA WEB Y SU EVOLUCIÓN 18
1.1.3. TENDENCIAS ACTUALES DE PROYECTOS O PROGRAMAS 22
1.1.4. APLICACIÓN WEB 22
1.1.5. METODOLOGÍA DE DESARROLLO 24
1.1.5.1. OMG (OBJECT MANAGEMENT GROUP) 24
1.1.5.2. MDA (ARCHITECTURE DRIVEN MODELS) 26
1.1.6. ESTUDIO COMPARATIVO DEL DESARROLLO TRADICIONAL VS DESARROLLO MDA 36
1.1.7. CUADRO COMPARATIVO DE HERRAMIENTAS MDA 39
1.2. ANTECEDENTES 40
1.3. BASES LEGALES 43
CAPÍTULO II: MATERIALES Y MÉTODOS 46
2.1. TIPO DE ESTUDIO 46
2.2. DEFINICIÓN CONCEPTUAL Y OPERACIONALIZACIÓN DE LAS VARIABLES 46
2.3. MÉTODOS Y TÉCNICAS 47
2.4. INSTRUMENTO 47
CAPÍTULO III: RESULTADOS 49
3.1. StarUML 49
3.2. ENTERPRISE ARCHITECT 54
3.3. MICROSOFT VISUAL STUDIO ASP.NET ENTITY FRAMEWORK 60

3.4. NORMAS INTERNACIONALES DE BUENAS PRÁCTICAS EN LA PROGRAMACIÓN 66
CAPÍTULO IV: DISCUSIÓN 71
CAPÍTULO V: CONCLUSIONES 74
CAPÍTULO VI: RECOMENDACIONES 75
ANEXOS 76
REFERENCIAS 83

ABREVIATURAS

OMG. – (Object Management Group por sus siglas en inglés) o Grupo de Administración de Objetos.

MDD. – (Model Driven Development por sus siglas en inglés) o Desarrollo Dirigido por Modelo.

MDA. – (Model Driven Architecture por sus siglas en inglés) o Arquitectura Dirigida por Modelo.

CIM. – (Model Independent Computer por sus siglas en inglés) o Modelo Independiente de la Computación.

PIM. – (Model Independent Platform por sus siglas en inglés) o Modelo Independiente de la Plataforma.

PSM. – (Model Specific Platform por sus siglas en inglés) o Modelo Especifico de la Plataforma.

ISM. – (Model Specific Implement por sus siglas en inglés) o Implementación Modelo Especifico

MVC. – (Model – View – Controler por sus siglas en inglés) o Modelo – Vista – Controlador.

CRUD. – (Create, Read, Update, Delete por sus siglas en inglés) o Crear, Leer, Actualizar y Eliminar.

UML. – (por sus siglas en inglés, Unified Modeling Language) o Lenguaje de Modelado Unificado.

HTML. – HyperText Markup Language, es decir, Lenguaje de Marcas de Hipertexto.

CSS. – Hojas de estilo en cascada.

PHP. – Hypertext Preprocessor.

NPM. – Es el manejador de paquetes por defecto para Node.js, un entorno de ejecución para JavaScript

COMPOSER. - Es un administrador de paquetes a nivel de aplicación para el lenguaje de programación PHP.

LISTA DE ILUSTRACIONES

Ilustración 1: Proceso del Estándar MDA [1] 14
Ilustración 2: Modelos generados en MDA 15
Ilustración 3: Logo StarUML [30] 49
Ilustración 4: Interfaz Gráfica de StarUML 50
Ilustración 5: Diagrama de Caso de Uso 51
Ilustración 6: Diagrama de Clases 52
Ilustración 7: Diagrama de Componente 52
Ilustración 8: Encapsulamiento de los Modelos 53
Ilustración 9: Logo Enterprise Architect [31] 54
Ilustración 10: Interfaz de Enterprise Architect [31] 54
Ilustración 11: Diagrama de Caso de Uso EA 56
Ilustración 12: Diagrama de Componente 57
Ilustración 13: Componentes EA (Aplicativo Web) 57
Ilustración 14: Estructura Interna del Diagrama de Componente 58
Ilustración 15: Diagrama de Clases EA 58
Ilustración 16: Logo Visual Studio [32] 60
Ilustración 17: Entorno de Desarrollo Microsoft Visual Studio 61
Ilustración 18: Modelo Vista Controlador ASP.NET [33] 61
Ilustración 19: Selección de Plantilla para Modelado MVC 62
Ilustración 20: Estructura MVC 62
Ilustración 21: Modelo PIM 63
Ilustración 22: Modelo PSM 64
Ilustración 23: Modelos SQL Server - Diagrama de Clase UML 65
Ilustración 24: Modelo PSM - SQL Server 65
Ilustración 25: Agregar Controladores 66
Ilustración 26: Codificación HTML y CSS 70

LISTA DE FIGURAS

Figura 1: Modelo de Arquitectura Web 17
Figura 2: Evolución de la Web 19
Figura 3: Estado de la Web 1.0. 19
Figura 4: Diagrama P2P (Personas Conectadas a Personas) 20
Figura 5: Web Semántica 21
Figura 6: Aplicación Web 22
Figura 7: Proceso de Desarrollo MDA 26
Figura 8: Fases de la MDA 26
Figura 9: Diagrama de Clases UML 28
Figura 10: Modelos MDA 28
Figura 11: Diagrama de Clases Ejemplo PIM 29
Figura 12: Arquitectura MDA 32
Figura 13: Proceso de Transformación MDA 33
Figura 14: Proceso de Desarrollo Tradicional 36
Figura 15: Proceso de Desarrollo MDA 37

LISTA DE TABLAS

Tabla 1: Comparativa Desarrollo Tradicional VS Desarrollo MDA 38
Tabla 2: Cuadro Comparativo de Herramientas MDA 39
Tabla 4: Reglas de Codificación..... 68

RESUMEN

Para alcanzar el propósito de generar una aplicación web dirigida por modelo, se abordó como principal objetivo desarrollar una aplicación web mediante el uso de la metodología y herramienta MDA orientada al diseño y construcción de modelos dirigidos para la transformación automática de código, debido a la falta de documentación al momento de realizar actualizaciones o modificaciones al sistema.

Por tal motivo, se realizó una investigación exploratoria para analizar la metodología y herramientas de la Arquitectura Dirigida por Modelo donde propone que el proceso de desarrollo sea guiado por los modelos, permitiendo la transformación e implementación del sistema de software. Dentro de este proceso es necesario utilizar herramientas de transformación, estas conciben el desarrollo a partir del diseño y construcción de modelos dirigidos llevándolos a un nivel de abstracción más alto enfocándose en sus etapas de transformación. Estas etapas son conocidas como: Modelo Independiente de la Computación (CIM), Modelo Independiente de la Plataforma (PIM), y el código fuente que resulta a través de la combinación del modelo CIM – PIM o PIM – PSM.

En base al fundamento teórico de MDA y herramienta de transformación se ha realizado una aplicación web (Sistema de Gestión y Curriculum Docente) que satisfacen las necesidades del cliente y desarrollador optimizando recursos de tiempo y costo de producción de un sistema de software.

Por lo tanto, los procesos de desarrollo de la MDA se centran en gestionar y organizar el trabajo colaborativo de un sistema separando en modelos conceptuales una representación detallada de la aplicación web a desarrollar, donde la ventaja más destacada de la Arquitectura Dirigida por Modelo es que a partir de estos modelos ampliamente validados en entornos industriales es factible la transformación sistemática de código que implemente la aplicación.

Palabras Clave. – MDA, Modelos, Metamodelos, Transformación, Código.

ABSTRACT

In order to achieve the purpose of generating a model-driven web application, the main objective was to develop a web application through the use of the MDA methodology and tool oriented to the design and construction of models for the automatic transformation of code, due to the lack of documentation at the time of making updates or modifications to the system.

For this reason, an exploratory research was carried out to analyze the methodology and tools of Model-Directed Architecture where it proposes that the development process be guided by the models, allowing the transformation and implementation of the software system. Within this process it is necessary to use transformation tools that conceive the development from the design and construction of directed models taking them to a higher abstraction level focusing on their transformation stages. These stages are known as: Computer Independent Model (CIM), Platform Independent Model (PIM), and the resulting source code through the combination of the CIM - PIM or PIM - PSM model.

Based on the theoretical foundation of MDA and transformation tool, a web application has been created (Management System and Teaching Curriculum) that satisfies the needs of the client and developer, optimizing time resources and production cost of a software system.

Therefore, the development processes of the MDA are focused on managing and organizing the collaborative work of a system separating in conceptual models a detailed representation of the web application to be developed, where the most outstanding advantage of the Model-Driven Architecture is that from these models widely validated in industrial environments is feasible the systematic transformation of code that implements the application.

Keywords. – MDA, Models, Metamodels, Transformation, Code.

INTRODUCCIÓN

PRESENTACIÓN DEL TEMA DE INVESTIGACIÓN

La generación de una aplicación web dirigida por modelos, consta del diseño y construcción de modelos conceptuales que permiten capturar los requerimientos de software para ser analizados por el desarrollador. Por ello, el Grupo de Administración de Objetos (OMG) ofrece una metodología de Arquitectura Dirigida por Modelo que facilita la integración de los modelos para documentar el proceso de desarrollo de software. Esta metodología se basa en implementar estrategias ágiles mejorando el rendimiento y calidad de los productos permitiendo el reúso de los modelos para realizar modificaciones y transformaciones respectivas que a su vez generen código automático u otros modelos más concretos que utilicen diagramas UML.

Por lo tanto, en esta investigación se pretende mostrar la factibilidad que ofrece la Arquitectura Dirigida por Modelo en el desarrollo de aplicaciones web, donde los procesos de desarrollo son modelados de forma correcta reduciendo la codificación y brindando escalabilidad al sistema, con la finalidad de detectar errores y corregirlos más rápido sin necesidad de alterar el sistema central.

El contenido de esta investigación está estructurado en los siguientes capítulos de estudio:

El capítulo I consiste en la formulación del marco teórico, donde se detallan y describen los conceptos básicos, componentes teóricos y diferentes términos que hacen entendible la investigación.

En el capítulo II se establecen los materiales y métodos que se emplean en la investigación, definiendo el tipo de investigación, los métodos y técnicas utilizados para llegar a la solución del problema, la descripción de variables conceptuales y los instrumentos utilizados para la recolección de datos.

El capítulo III presenta todos los resultados obtenidos a partir de la investigación realizada.

El capítulo IV muestra la discusión que se establece a partir de los resultados obtenidos e investigaciones científicas anteriores.

El capítulo V y capítulo VI presenta las conclusiones y recomendaciones obtenidas de los objetivos planteados en la investigación.

PLANTEAMIENTO DEL PROBLEMA

Desde el surgimiento de sistemas informáticos basados en la programación orientada a objetos, se propone un enfoque de documentación donde el modelo guía todo el proceso de desarrollo de una aplicación, pero a medida que los programadores adquirieron más conocimiento dejaron de lado esta estructura y se enfocaron en la codificación haciendo que la retroalimentación de un sistema sea difícil de comprender, es decir, realizar cambios o actualizaciones al sistema se volvió muy complejo.

Por lo tanto, al no cumplir con los lineamientos que promovía la programación orientada a objeto las empresas desarrolladoras de software aumentaron costos y tiempo de producción, ya que, al basarse solo en la codificación no obtenían documentación relevante para una posterior retroalimentación a los demás miembros de la empresa. Es así como surgen las metodologías ágiles que transforman los procesos tradicionales de programación en procesos automatizados mediante la Arquitectura Dirigida por Modelos para especificar modelos que guíen todo el proceso de transformación de un aplicativo web con ayuda de diagramas UML.

Por esta razón, la Arquitectura Dirigida por Modelo ofrece herramientas que optimizan las funcionalidades del ciclo de vida de un software, mediante el análisis, diseño, programación, pruebas y mantenimiento, incorporando el reúso de los modelos para dar solución a problemas cotidianos de los programadores.

Según Rueda [1] “MDA (Arquitectura Dirigida por Modelos), un concepto promovido por la OMG (Object Management Group), que propone basar el desarrollo de software en modelos especificados utilizando UML, para que, a partir de esos modelos, se realicen transformaciones que generen código u otro modelo con características de una tecnología particular (o con menor nivel de abstracción)”.

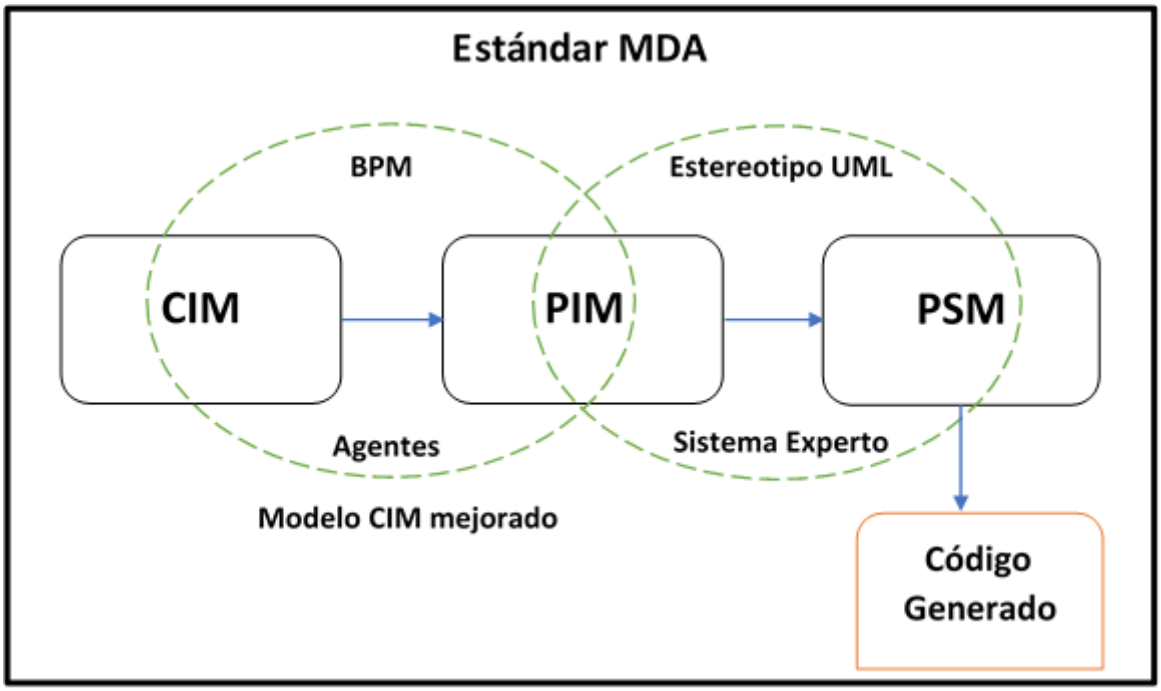


Ilustración 1: Proceso del Estándar MDA [1]

En la ilustración 1 se muestra el estándar que rige la Arquitectura Dirigida por Modelos y los procesos que se llevan a cabo para la producción de un sistema de software.

JUSTIFICACIÓN

La necesidad de implementar metodologías ágiles para el desarrollo de sistemas de información (SI) surgen a medida que las empresas desarrolladoras de software se vieron en la obligación de generar métodos y técnicas que garanticen la documentación del sistema que se está implementando y así optimizar la integración de nuevas actualizaciones sin necesidad de afectar la aplicación lanzada al mercado.

Debido a que la investigación es considerada como exploratoria el beneficio que garantiza la MDA es mantener en constante cambio los procesos de desarrollo de software adaptándose a los avances tecnológicos y permitiendo que los proyectos sean reutilizables y manejables para otros departamentos de una empresa. Al mismo tiempo esta investigación es conveniente porque facilita su comprensión en caso de dar mantenimiento al sistema, beneficiando al cliente y programador, aumentando la solución a problemas tradicionales en el desarrollo de software.

Con todo esto se podrá efectuar soporte tecnológico al sistema de software en cuanto a diseño, construcción, pruebas, mantenimiento y documentación del software se trata

donde todo el equipo de desarrollado juega un papel importante. No obstante, para alcanzar el objetivo de la MDA en el desarrollo de software se estudian las actividades que se deben realizar para generar una aplicación mediante los modelos dirigidos.

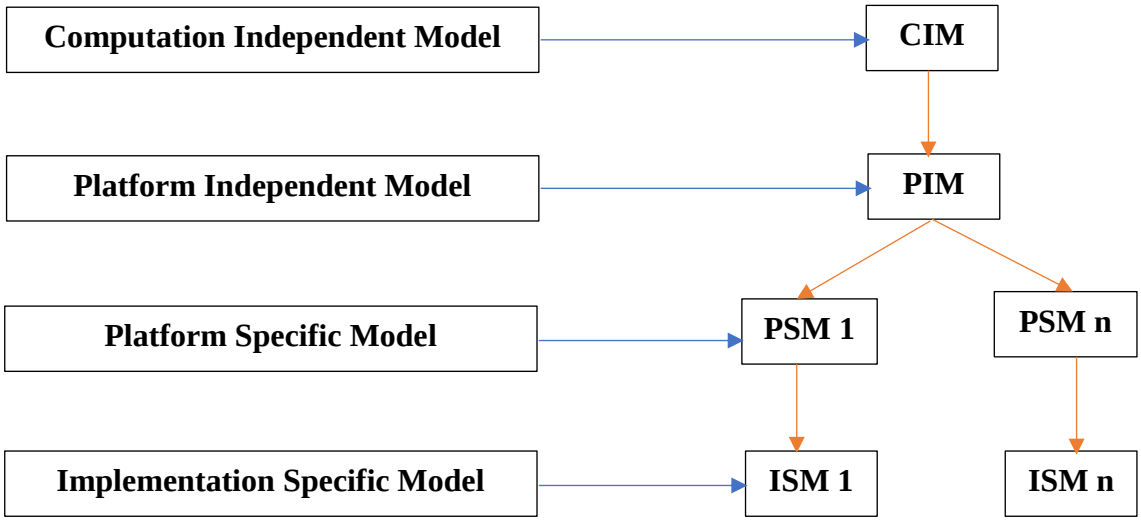


Ilustración 2: Modelos generados en MDA

A partir de la ilustración 2, Enrique et al. [2] muestran la forma en que los modelos son transformados para llegar al producto final, a continuación se describen los pasos a seguir para realizar un sistema de software mediante los modelos generados de la MDA.

Primero se debe definir el modelo conceptual de la idea que engloba todo el sistema a desarrollar, este es considerado como el Modelo Independiente de la Plataforma.

Posteriormente y partiendo de la herramienta de Arquitectura Dirigida por Modelos, se llevan a cabo las implementaciones de dichos modelos transformándolos en metamodelos específicos que demuestren un nivel de abstracción de los requerimientos del cliente.

Finalmente, la combinación o agrupación de los modelos independientes de plataforma (PIM) y los modelos específicos de plataforma (PSM) pasan a un proceso de transformación el cual, es descrito en código fuente, este código cuenta con todas las especificaciones y requerimientos que se necesitan para el desarrollo de una aplicación.

OBJETIVOS

GENERAL

- Desarrollar una aplicación web mediante el uso de la metodología y herramienta MDA orientadas al diseño y construcción de modelos dirigidos para la transformación automática de código.

ESPECÍFICOS

- Analizar la metodología de Arquitectura Dirigida por Modelo para el desarrollo de una aplicación web.
- Utilizar la herramienta MDA para el diseño y construcción de modelos dirigidos.
- Implementar los modelos dirigidos para la transformación automática.
- Demostrar la factibilidad de desarrollo mediante la transformación automática de modelos – código.

CAPÍTULO I:
MARCO TEÓRICO

1.1. BASES TEÓRICO - CIENTÍFICAS

1.1.1. ARQUITECTURA WEB

La arquitectura web es una parte fundamental al momento de desarrollar aplicaciones web, es la principal herramienta donde se va a esquematizar la estructura web de los contenidos de los patrones de búsqueda de los usuarios, experimentando una fluidez en la forma de intercomunicarse con el cliente – proveedor. La arquitectura web cuenta con recursos que hacen referencia a las aplicaciones web donde el diseñador o desarrollador plasma todas las especificaciones o requerimientos del cliente en modelos específicos para su estudio e implementación.

Leyva *et al.* [3] expresan que: El diseño y arquitectura web permiten que los usuarios puedan comparar y contrastar los productos y servicios de la competencia con un mínimo gasto de tiempo y esfuerzo.

Dentro de la tecnología de la información la arquitectura web cuenta con modelos que describen el funcionamiento de las aplicaciones web, estos modelos caracterizan la relación entre diversos elementos que componen un sistema los cuales son explicados a continuación.



Figura 1: Modelo de Arquitectura Web

La figura 1 muestra los modelos de la arquitectura web los cuales deben ser estudiado antes de implementar una aplicación web.

Modelos Punto a Punto (P2P). – Son aquellas que responden a un tipo de arquitectura de red en las que cada canal de red se utiliza para comunicar únicamente dos nodos conectados entre sí.

Modelo Cliente – Servidor. – Es un modelo informático que distribuye los recursos, servicios y aplicaciones, donde el cliente realizas peticiones y el servidor responde a dichas peticiones. La ventaja de contar con un modelo cliente – servidor es que permite

controlar y actualizar las tareas, y también que al estar distribuidas las funcionalidades entre varios ordenadores brinda la facilidad de dar mantenimiento tanto en seguridad y actualizaciones del sistema.

Modelo con Servidor de Aplicaciones. – El servidor de aplicaciones envía una petición al servidor web este procesa la petición y reenvía una respuesta a servidor de aplicaciones muchas veces estos procesos deben contar con una base de datos la cual puede estar en la misma máquina o en otro equipo que esté conectado a la red.

Modelos con servidor de aplicaciones externo. – El servidor web va a contar con menos carga de trabajo, por lo que el servidor de aplicaciones va a ejecutar los elementos necesarios. Esta es una comunicación bidireccional ya que mientras el servidor web envía una petición el servidor de aplicaciones se encarga de transferir todos los elementos para su ejecución.

Modelo con varios servidores de aplicaciones. – Es un servidor web conectado a través de un redirector de servidores de aplicaciones, es decir que los servidores de aplicaciones deben ser iguales para realizar la misma función.

Para construir una aplicación web primeramente se deben tener en cuenta todos los componentes que se relacionan a la arquitectura web puesto que facilitan la organización de los recursos tanto de hardware como de software optimizando cada esquema de la tecnología de la información responda a la evolución de los sistemas de información.

1.1.2. LA WEB Y SU EVOLUCIÓN

Al comienzo el internet representaba computadoras conectadas con otros computadores que permitían intercambiar información entre ellas, fue así como en los años 80 surgieron protocolos o reglas que hacen que la navegación explore más sitios convirtiendo así lo que ahora se llama internet. De esta manera la web fue evolucionando convirtiéndose en uno de los grandes potenciales del internet puesto que permite compartir recursos que solventan las necesidades de los usuarios. Sin embargo, poco a poco la web ofrece métodos y técnicas que hacen que el usuario vea el mundo tecnológico de otra manera, es decir, lo que antes era impredecible en la actualidad ya es manejable, Molina *et al*. [4] expresan que: “Las aplicaciones web son cada vez más populares y su uso ha acaparado el ámbito científico, cultual, académico, empresarial entre otros, y esto es debido a las

múltiples ventajas que el usuario tiene respecto a los programas de escritorio". Por lo tanto, la web está ocasionando grandes innovaciones ya que todos los procesos de negocios o empresariales se manejan a través de las aplicaciones web, de allí varios artículos expresan que la web ha venido evolucionando, hasta el momento se pueden distinguir las siguientes etapas:

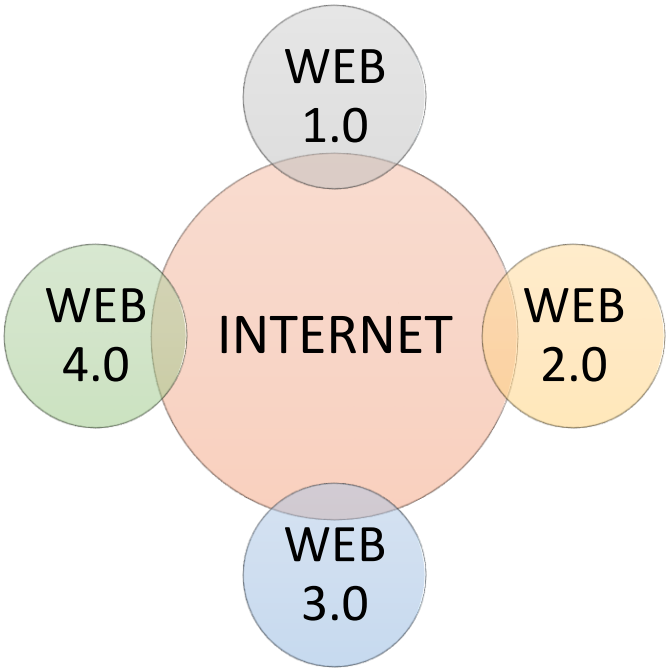


Figura 2: Evolución de la Web

La figura 2 muestra la evolución que ha tenido el internet en la web pasando de tradición en tradición gestionando los procesos que son esenciales para satisfacer las necesidades del usuario.

La web 1.0: Personas conectadas a la web, donde todo era estático, las personas eran emisor y receptor de la información de la web, no existían actualizaciones que respalden esta información. Aquino [5] dice que: "Esta generación nace como un sistema de hipertexto para compartir información en Internet, con la finalidad de publicar documentos".



Figura 3: Estado de la Web 1.0.

La figura 3 muestra los estados de comunicación entre la web y los usuarios, donde la accesibilidad a través de sitios direccionales permite un discurso lineal entre los usuarios y la web.

La web 2.0: Personas conectadas a personas – redes sociales, posibilidad de compartir información, es la web que en la actualidad se está viviendo donde todos se comunican al mismo tiempo, comparten recursos, esta es una mejora de la web 1.0 donde ya se implementan navegadores de búsqueda que hacen del usuario una persona capaz y colaborativa.

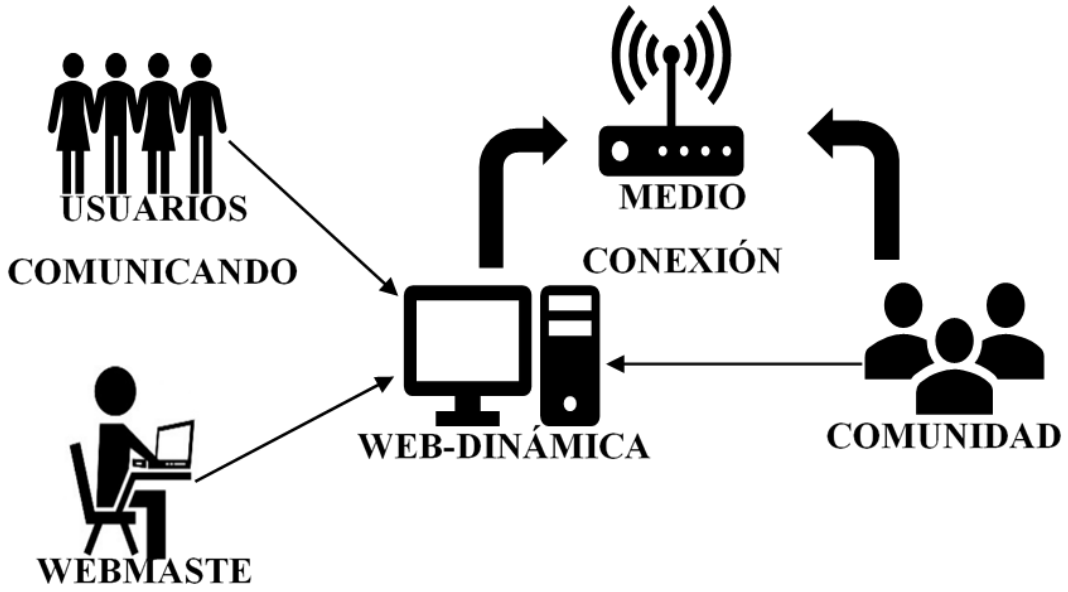


Figura 4: Diagrama P2P (Personas Conectadas a Personas)

Aquino [5] en la figura 4 dice que: "Surgen aplicaciones y páginas que utilizan la inteligencia colectiva, consecuencia de ello las páginas pueden ser personalizadas, los usuarios intercambian e introducen datos y se convierten en una herramienta dinámica que permite el intercambio de información". Por lo tanto, es algo más sofisticado ya que no solo comparte documentos, sino que interactúa con los usuarios.

La web 3.0: Aplicaciones web conectándose a aplicaciones web o también llamada web semántica es más intuitiva y fácil de manejar, permite transferir información de manera continua organizando los datos en conjuntos de grandes redes de información descifrables donde la inteligencia artificial y la web se unen. "La web 3.0 está muy asociada al concepto de personalización. Ofrece un flujo de información y de contenidos adaptados a gustos y preferencias" [6].

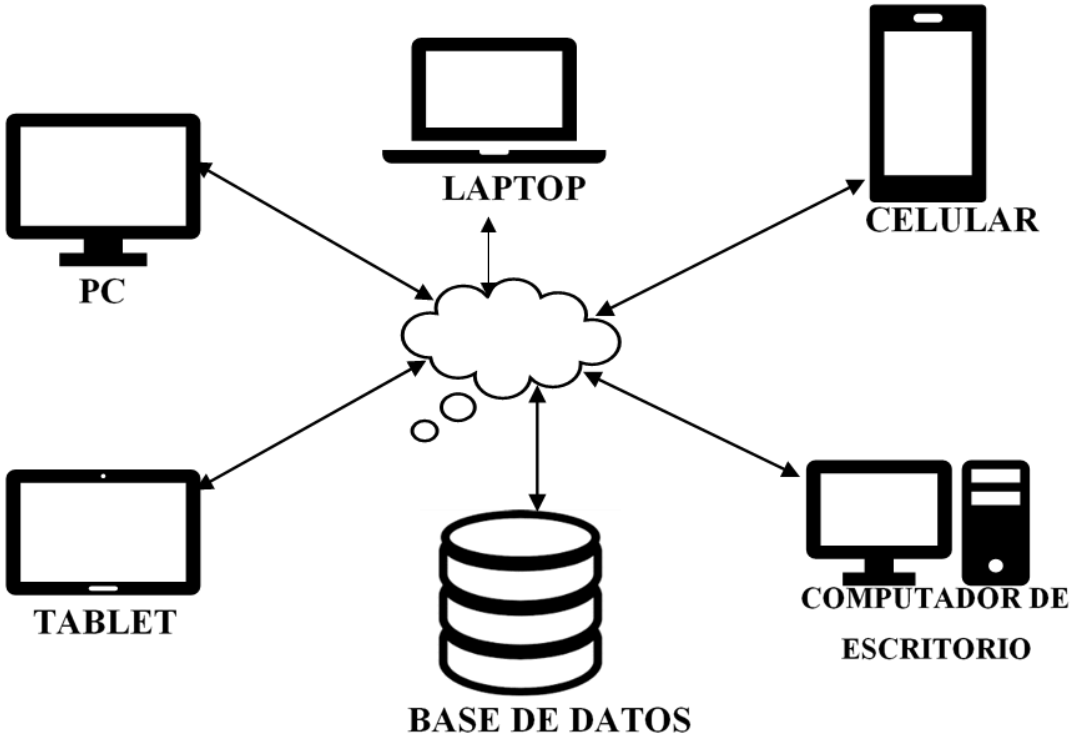


Figura 5: Web Semántica

La figura 5, muestra el internet de ahora, el internet de las cosas, todos conectados a todos, compartiendo aplicaciones y tareas útiles para la sociedad.

La web 4.0: web 3D + web 3.0 + inteligencia artificial + voz como vehículo de intercomunicación. Esta es la web del futuro donde las personas y las cosas se comunican a través de la inteligencia artificial permitiendo la toma de decisiones conjunta. A medida que la tecnología avanza se piensa que la web será capaz de comprender, aprender, asimilar y razonar como los seres humanos.

La Web semántica propone superar las limitaciones de la Web actual mediante la introducción de descripciones explícitas del significado, la estructura interna y la estructura global de los contenidos y servicios disponibles en la Web, es decir, se logrará un medio universal que permita el intercambio de datos y brindar un mayor significado a la misma para que puedan ser interpretadas por las máquinas. Los principios en los que se va a basar este nuevo modelo de Web son la descentralización, la compartición, la compatibilidad, máxima facilidad de acceso y contribución, apertura al crecimiento y uso de la ontología. Esta última, tomada del área de la Inteligencia Artificial como herramienta fundamental en la representación del conocimiento [7].

1.1.3. TENDENCIAS ACTUALES DE PROYECTOS O PROGRAMAS

Las tendencias actuales de programas o proyectos permiten a las organizaciones tomar decisiones convenientes en todo el proceso de desarrollo de software abaratando costos y tiempo de ejecución. Rojas *et al.* [8] expresan que actualmente muchas empresas se organizan en función de los proyectos que desarrollan, lo cual se dificulta cuando aumenta la cantidad de los mismo, decidiendo entonces organizarlos por portafolios o programas de proyectos, garantizando una mejor gestión y seguimiento de los mismos.

A partir de esto, las tendencias actuales permiten organizar los proyectos de forma jerárquica, dependiendo de su prioridad mediante el seguimiento y control de los objetivos globales tales como el alcance, el cronograma a través del tiempo y un costo que se medirá con el presupuesto que se genere para el programa. Sin embargo, la tecnología y el internet evolucionan a cada momento generando metodologías ágiles para el desarrollo de software, donde es posible desarrollar aplicaciones web que cuenten con documentación que luego pueda ser entendida por otros empleados de la organización o empresa y así fortalecer la retroalimentación del sistema mediante el avance tecnológico.

1.1.4. APLICACIÓN WEB

Las aplicaciones web son un tipo de software que se codifica en un lenguaje soportado por el navegador para interactuar con el usuario final, permitiendo el acceso a la información del internet, de allí que han sido catalogadas como App Web como lo muestra la figura 6.

Por lo tanto, una aplicación web (web-based application) es un tipo especial de aplicación cliente/servidor, donde el cliente (el navegador, explorador o visualizador), el servidor (el servidor web) y el protocolo por el cual se comunican (HTTP) estén estandarizados y no han de ser creados por el programador de aplicaciones [9].



Figura 6: Aplicación Web

Una aplicación web contiene diversos elementos que interactúan con el cliente permitiendo el acceso directo a los datos de forma interactiva, puesto que las App Web se encargan de responder a todas las acciones que realiza el usuario de forma continua a través de un navegador, por ejemplo, el llenado de un formulario, acceder a videojuegos, etc.

Las aplicaciones web actualmente son codificadas a través de lenguajes de alto nivel y altamente soportadas por un navegador, pero con el paso del tiempo la tecnología ha dado un giro a la forma en que se crean estas aplicaciones. Por ello, las aplicaciones de software tienen la singularidad de no poder predecir de antemano si la facilidad de uso del diseño de una interfaz cumple con los requerimientos propios de la aplicación y de la persona usuaria final; asimismo, es importante que las aplicaciones de software requieran poco esfuerzo de persona usuaria, no recargue la memoria de trabajo y sea fácil e intuitiva de usar [10].

Las aplicaciones web también son utilizadas para actualizar y mantener los resultados de negocios que se realizan mediante un navegador de internet. Estas aplicaciones son desarrolladas por medio de lenguajes de alto nivel como ASP.NET, Java, PHP, JavaScript, HTML, etc., que brindan al cliente una interfaz amigable donde se interactúa mediante un computador y la navegación en el internet.

Las Aplicaciones Web cuentan con ventajas que ofrecen al mundo desarrollador un ambiente de ahorro de tiempo, completa compatibilidad, actualización continua e inmediata, recuperación de datos, ahorro de recursos en equipos y dispositivos debido a que los recursos que consume una aplicación web no se encuentran directamente en nuestro dispositivo más bien están dentro de un servidor previamente adquirido por el desarrollador.

También cuenta con la compatibilidad de múltiples plataformas, es decir, que se puede utilizar desde cualquier dispositivo electrónico conectado al internet, debido a esto las aplicaciones web que son dirigidas a dispositivos móviles cuentan con un estilo o fichero de CSS para facilitar las animaciones o acceso a los recursos de estas App Web.

Estas se pueden clasificar acorde a las funciones que cumplen, entre estos tenemos las siguientes:

Aplicaciones web estáticas. - Este tipo de aplicación muestra muy poca información en su presentación y casi nunca cambia, suelen estar desarrolladas en HTML y CSS.

Aplicaciones web dinámicas. – Mantienen un cambio constante debido a que permiten la carga directa desde una base de datos con la cual interactúa el usuario.

Portales para aplicaciones web. – Con portales se refiere a diversas acciones que puede realizar el usuario en su App Web, tales como, secciones, catálogos, etc.

Tiendas Online para aplicaciones web. – La tienda online cuenta con una estructura mucho más complicada debido al tipo de seguridad y accesos que deben ser aplicados para las compras de artículos, electrodomésticos, etc., esto funciona como un e-commerce.

Aplicaciones Web Animadas. – Permite el posicionamiento de la web y la optimización de motores en la búsqueda, este tipo de aplicación no es muy recomendada ya que los motores de búsqueda casi no leen bien la información.

Cada aplicación web tiene ventajas y desventajas que son esenciales para su implementación puesto que en el internet hay que cumplir con ciertas normas que hacen de este una fuente de información.

1.1.5. METODOLOGÍA DE DESARROLLO

1.1.5.1. OMG (OBJECT MANAGEMENT GROUP)

Es un consorcio internacional sin ánimos de lucro de la industria de la informática, que integra múltiples organizaciones dedicadas al desarrollo de software desde 1989 fecha de su creación. Este consorcio está enfocado a diversos mercados de negocios donde se desarrollan aplicaciones de software. En el año 2001 adoptó un nuevo marco de trabajo llamado Arquitectura Dirigida por Modelos, esta arquitectura ofrece a los desarrolladores de sistemas de información o de cualquier software en general un camino viable para mantener las inversiones estables.

Este grupo de administración de objeto cuenta con diversos estándares que integran soluciones factibles a los problemas generados durante la programación tradicional.

Cabrera *et al.* [11] dicen que la Object Management Group considera que los modelos son independientemente de la plataforma (PIM) y su derivación convierte estos modelos

a modelos específicos de la plataforma (PSM). En MDA las transformaciones son el núcleo principal del enfoque. Convertir un modelo a otro modelo del mismo sistema, son transformaciones que se pueden llevar a cabo siguiendo muchas vías: usando tipos, marcas, plantillas, etc. De esta manera el desarrollo de software se convierte en un proceso iterativo de transformaciones de modelos: cada paso transforma un PIM del sistema a un nivel dentro de un PSM al siguiente nivel, hasta que se alcanza la implementación final, con la particularidad que cada PSM de una transformación puede convertirse en el PIM de la siguiente transformación (con otro nivel de abstracción). En este contexto, la implementación es justo otro modelo, el cual provee toda la información necesaria para construir el sistema y ponerlo en funcionamiento.

MDA evoluciona el mercado de la tecnología de información debido a que los desarrolladores ven a esta metodología como una manera de organizar y administrar los modelos empresariales e industriales que se desarrollan, facilitando la planificación de estrategias ágiles para el desarrollo de sistemas de software a través de lenguajes de programación optimizando la obtención de código fuente. Esta es una propuesta promovida por el Object Management Group donde se describen las ideas fundamentales de este paradigma y se define como una propuesta para el desarrollo de sistemas y que es dirigido por modelos porque provee los recursos para que los modelos dirijan el curso del entendimiento, diseño, construcción, despliegue, operación, mantenimiento y modificación de los sistemas. La Arquitectura Dirigida por Modelos pretende obtener aplicaciones con alta flexibilidad en la implementación, integración, mantenimiento y prueba. Los tres principales objetivos de esta arquitectura es: portabilidad, interoperabilidad y reusabilidad. Porque tres puntos de vistas para un sistema son: punto de vista independiente de la computación, punto de vista independiente de la plataforma y punto de vista especifico de la plataforma [12].

A partir de lo antes mencionado la figura 7 muestra el ciclo de vida tradicional e iterativo de la implementación de esta arquitectura, donde en la parte inferior se encuentran las etapas del ciclo de vida mientras que por otro lado en la parte superior se encuentra los procesos de salida.

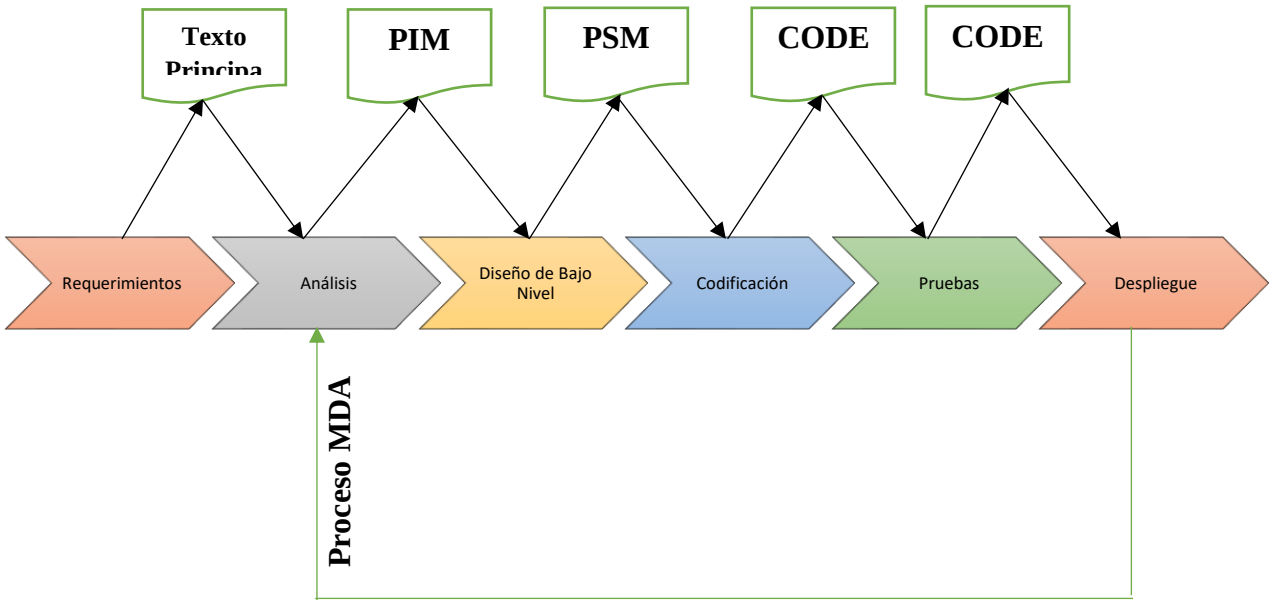


Figura 7: Proceso de Desarrollo MDA

Por tal motivo, el desarrollo de sistemas de software de la MDA se basa en tres fases principales:

Primero se procede a construir un modelo con alto nivel de abstracción, es decir, independiente de cualquier tecnología lo cual es conocido como Modelo Independiente de Plataforma.

A continuación, se transforman los PIM en uno o varios modelos específicos de la plataforma.

Para finalmente transformar los modelos específicos de la plataforma en código fuente que se sea manejable en cualquier lenguaje de programación.

Todo esto se muestra en la figura 8 es decir las fases principales que realiza la arquitectura dirigida por modelo para construir un sistema de software.



Figura 8: Fases de la MDA

1.1.5.2. MDA (ARCHITECTURE DRIVEN MODELS)

Toledo *et al.* [13] describen que la MDA es una propuesta promovida por el Object Management Group (OMG, por sus siglas). En (OMG, 2003) se describen las ideas

fundamentales de este paradigma, dentro de las que se destacan que es una propuesta para el desarrollo de sistemas y que es dirigido por modelos porque provee los recursos para que los modelos dirijan el curso del entendimiento, diseño, construcción, despliegue, operación, mantenimiento y modificación de los sistemas. MDA pretende obtener aplicaciones con alta flexibilidad en la implementación, integración, mantenimiento y prueba. De acuerdo con (OMG, 2003) los tres objetivos principales de MDA son portabilidad, interoperabilidad y reusabilidad. Propone tres puntos de vistas para un sistema los cuales son: punto de vista independiente de la computación, punto de vista independiente de la plataforma y punto de vista especifico de la plataforma.

Modelos MDA

Son una representación simplificada de todos los aspectos reales que se dan al momento de plasmar una idea o descartarla, son muy importantes para la industria del desarrollo de software ya que permiten entender y comunicar todos los aspectos que van a ser utilizados en el desarrollo de un sistema de software.

"Los modelos constituyen el núcleo dentro del proceso de desarrollo, éstos dejan de ser solo documentación para ahora formar parte de la implementación del sistema, están escritos en lenguajes bien definido y a diferentes niveles de abstracción" [14].

Lenguajes de Modelado Unificado (UML) son importantes para el desarrollo de modelos dirigidos, se dice que es importante porque la arquitectura dirigida por modelos no restringe los diagramas de lenguaje unificado, sino que se puede utilizar cualquier lenguaje bien definido para construir los modelos dirigidos en el desarrollo de un sistema.

Ejemplo: *La figura 9 muestra un diagrama de clases, que representa el modelo de un sistema de gestión y curriculum docente*.

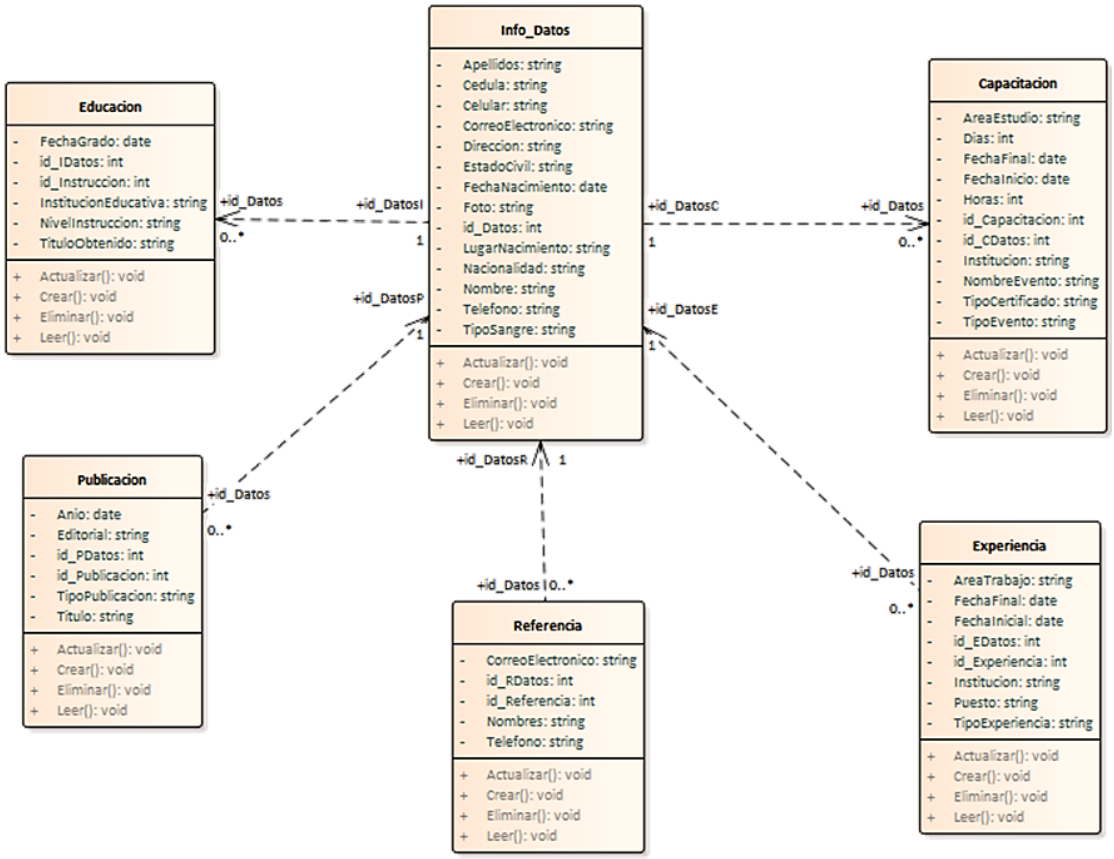


Figura 9: Diagrama de Clases UML

Los modelos o vistas que manejan esta arquitectura muestran que a partir de los modelos independiente de la plataforma se crean los metamodelos abstractos expresados en modelos específicos de la plataforma y estos son transformados en código fuente. Básicamente el PIM representa toda la documentación del sistema informático. Mientras que la trazabilidad se genera a partir del cambio que se realiza en el PIM y esto se refleja tanto en el PSM y en el código fuente.

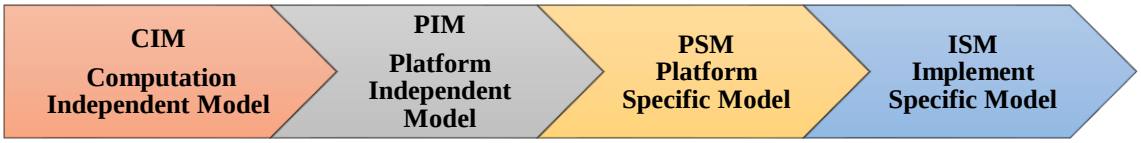


Figura 10: Modelos MDA

A continuación, se describen cada uno de los modelos expresados en la figura 10 de la Arquitectura Dirigida de Modelos.

CIM o (Modelo Independiente de la Computación). – Este tipo de modelo o vista representa la parte principal o central del proceso de negocio, donde se debe entender o comprender todo el proceso de negocio.

Los CIM se asocian a los modelos de dominio y es recomendable que sean especificados a partir de un vocabulario común para todos los elementos (computacionales o humanos) implicados en el desarrollo del sistema [15].

PIM o (Plataforma Independiente del Modelo). – Modelo de alto nivel independiente de cualquier tecnología, permite la abstracción de las características técnicas del sistema.

El PIM representa los aspectos que no se cambiarán de una plataforma a otra, de acuerdo a una tecnología o método de implantación escogido para la representación computacional [15].

Un PIM es fácil de comprender por parte del cliente del sistema, por lo que resultará más sencillo validar y corregir el sistema.

También permite la creación de diferentes implementaciones del sistema en diversas plataformas, sin distorsionar la estructura y funcionalidades básica del sistema.

La figura 11 muestra el ejemplo de un PIM con tres clases interrelacionadas.

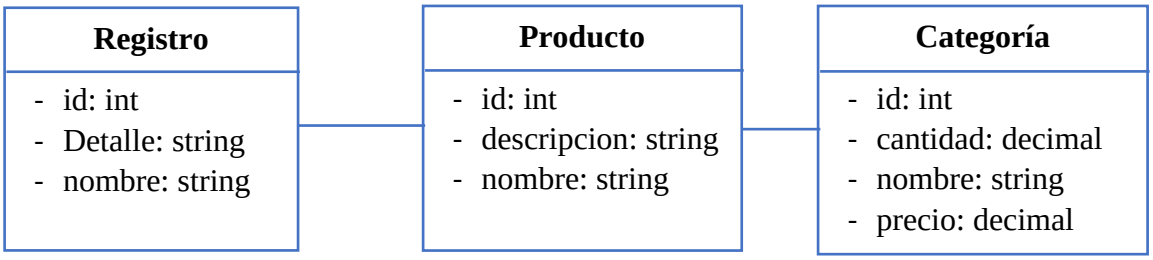


Figura 11: Diagrama de Clases Ejemplo PIM

PSM o (Plataforma Especifica de Modelo). – Esta vista o modelo combina las especificaciones del PIM con las del sistema, proporciona independencia entre las capas de negocio y la tecnología empleada.

En el PSM se puede observar la manera en la que un sistema usa las herramientas de la plataforma para el cumplimiento de los objetivos trazados en la etapa de especificación inicial [15].

ISM o (Implementación Especifica de Modelo). – Modelos o vista final de la MDA donde la generación de código fuente se da mediante la automatización de las plataformas independientes de modelos.

Por otro lado, Martínez *et al.* [16] mencionan que el CIM surgen en la fase inicial del proceso de desarrollo comprendiendo la modelación del negocio en su totalidad, mientras

que el PIM representa los modelos que describen una solución de software que no contiene detalles de la plataforma concreta donde será implementada la solución, a través de esto el PSM surge del PIM y se crea entre la fase del diseño y la codificación de la solución.

Los modelos forman parte importante en la estructuración del sistema o una parte de estos, ya que en ingeniería de software los modelos tienen una larga e importante historia, los diseñadores utilizan los modelos para planificar todo el proceso de desarrollo de un software o también para comunicar a otras personas sobre el proyecto que se está generando.

Por ello, como se mencionó anteriormente las principales ventajas de la Arquitectura Dirigida por Modelos son:

Portabilidad. – Aumenta la reutilización de la aplicación, reduciendo el coste y complejidad de desarrollo.

Interoperabilidad. – Permite utilizar métodos rigurosos mediante estándares que demuestren las funcionalidades del sistema.

Productividad. – Muestra el trabajo colaborativo entre los miembros de una empresa, logrando una buena comunicación entre ellos.

Estas ventajas ayudan a reducir el tiempo y coste de desarrollo durante todo el ciclo de vida de la aplicación mejorando la calidad.

Metamodelos MDA

Un metamodelo en ingeniería de software describe y especifica todos los elementos de diseño, análisis, construcción y reglas que se utilizan para modelar todos los componentes de un sistema.

Metamodelo es la descripción y especificación de los elementos y reglas que se utilizan para crear modelos semánticamente correctos para un dominio en particular. También suele definirse como un lenguaje de modelado [17] .

Metamodelos de Cuatro Capas

La clave para gestionar o construir un sistema informático reside en la inteligencia del uso y administración de los metadatos para una integración e interoperabilidad exitosa dependiendo de cualquier aplicación, herramienta, plataforma o base de datos. Las descripciones del cliente no cambian sin importar la plataforma, aplicaciones y herramientas mediante esto se aplican estándares que son esenciales para la modelación de un sistema de software especificado por el cliente, los estándares que maneja la MDA son (UML, MOF, XMI y CMW) los cuales sirven como base para crear, administrar y publicar los modelos o arquitectura dirigida por modelos sin importar el tipo de sistema que se esté construyendo.

En la figura 12 se muestran todos los movimientos que se generan a través de los estándares MDA.

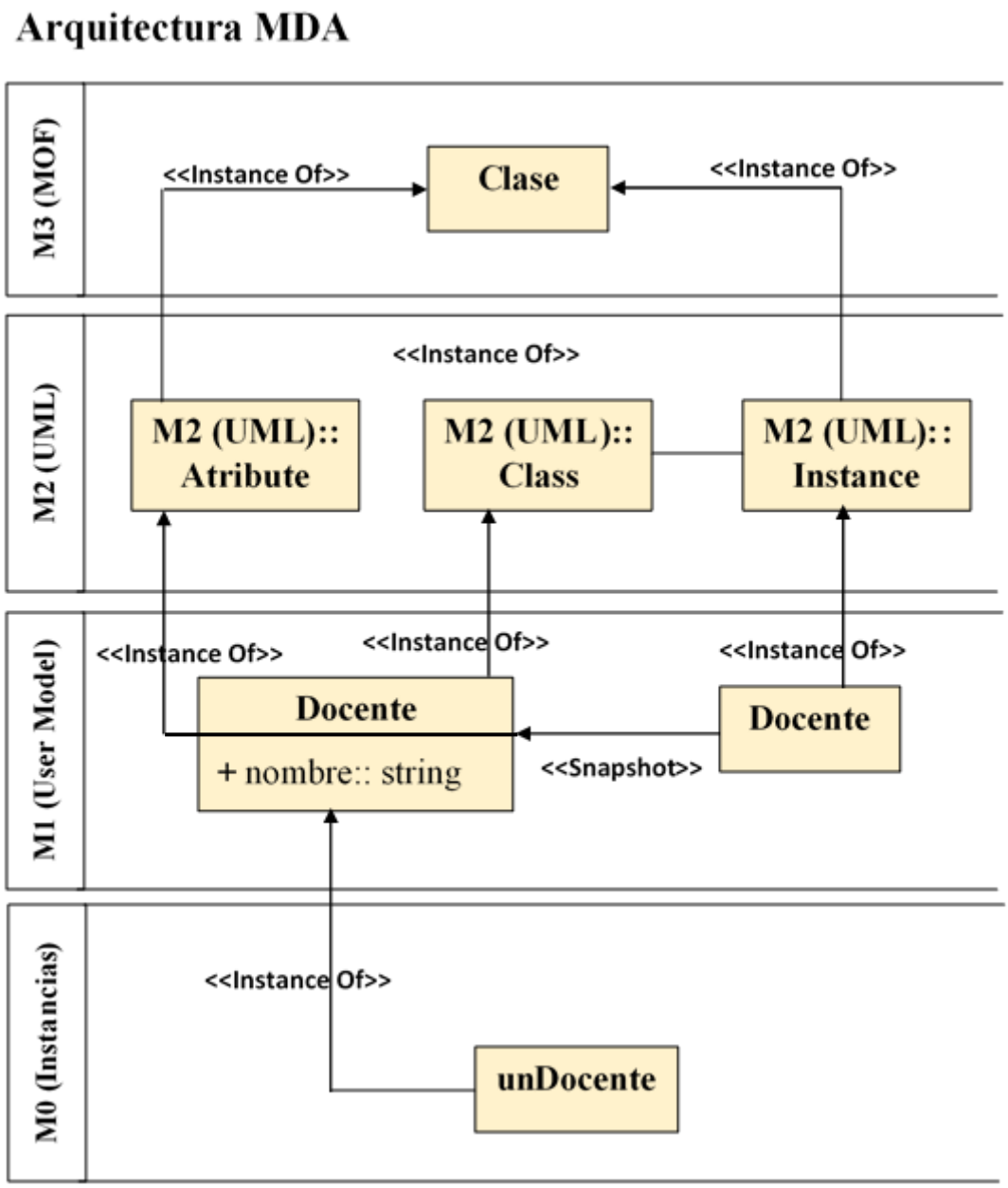


Figura 12: Arquitectura MDA

Meta Object Facility (MOF). – Lenguaje común y abstracto para la especificación de metamodelos. Representa el nivel meta más general y tiene el objetivo de permitir la incorporación de nuevos lenguajes de modelado (metamodelos) para fines específicos.

Sus componentes o capas de esta arquitectura son: Capa M3 (Metametamodelo), Capa M2 (Metamodelos), Capa M1 (Modelos), Capa M0 (Instancias) [17].

CMW o Common Metamodel Warehouse. – Permite el intercambio de metamodelos a datos almacenados.

UML o Lenguaje de Modelo Unificado. – Sirve como base para la definición de los CMW, a partir de estos modelos creados a través de UML se pueden realizar las traducciones automáticas a otros lenguajes.

XML o Metadata Interchange (XMI). – Define una taza que permite transformar modelo UML en XML para poder ser tratados automáticamente por otras aplicaciones.

El XMI es un lenguaje para permitirles a los usuarios desarrolladores de software el intercambio de modelos UML. Es una especificación del OMG definida con el objetivo de facilitar el intercambio de modelos entre herramientas de modelado [17].

Transformación de Modelos

La transformación de modelos juega un papel muy importante en la metodología MDA puesto que permite la traducción automática de los modelos dirigidos para ser comprendidas en código fuente. En la figura 13 se muestra el proceso de transformación MDA o de diferentes modelos.

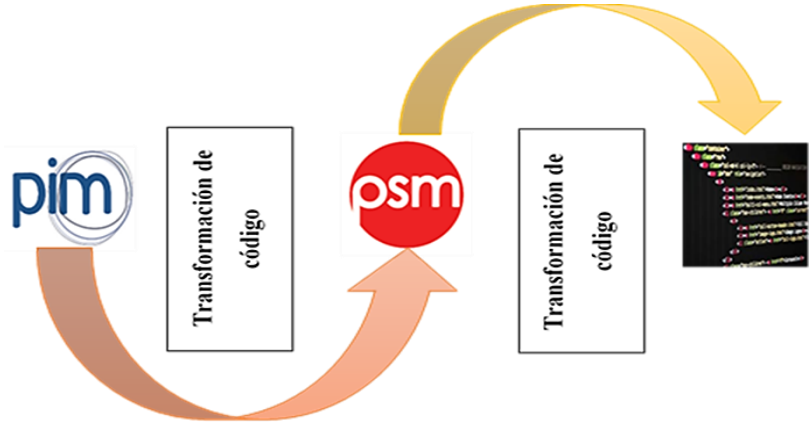


Figura 13: Proceso de Transformación MDA

Conceptos importantes en las transformaciones de modelos

Definición de transformación. – Es un conjunto de reglas de transformaciones para definir un juego de transformaciones.

Reglas de transformación. – Es un grupo de reglas de transformación para definir cómo usar un modelo y convertirlo en otro modelo.

Herramienta de transformación. – Estas juegan un papel importante en el proceso de desarrollo de la MDA, debido al enfoque de transformación de modelo independiente de la plataforma a modelo específico de plataforma, es decir, de PIM – PSM.

Transformación MDA

Se pueden transformar los modelos de diversas maneras a continuación se explicarán algunas de estas alternativas.

Transformación Manual. – Este proceso es el que se viene dando al realizar una transformación de modelo independiente de la computación a la plataforma especifica de modelo.

Usando un Archivo. – Mediante este proceso la MDA implementa los modelos UML para la transformación automática en las plataformas de especificaciones de modelo.

Usando Patrones y Marcas. – Este permite el mapeo de los patrones y marcas especificadas en el PIM para luego ser instanciadas en el PSM.

Automático. – En este contexto ya no es necesario que el desarrollador implemente los procesos de archivos o de patrones y marca, debido a que el código es generado por la plataforma que se está utilizando.

La transformación de modelos se considera el proceso central de MDA. Para especificar estas transformaciones se apoya el uso del estándar Queries/Views/ Transformations (QVT). Este estándar se expresa en MOF y queda establecido un lenguaje para la transformación de modelos, la consulta y la generación de vistas. QVT brinda la posibilidad de explorar los modelos desde disímiles puntos de vistas. Considera en el mismo la transformación, el mapeo y el marcado, elementos de interés dentro de este proceso [16].

QVT (Query Views and Transformations). - Es un estándar de desarrollo que define cómo llevar a cabo el proceso de transformación de modelos. En este estándar se incluirá el lenguaje MOF el cual constará de tres partes esenciales para la construcción.

- El lenguaje para crear vista de modelos.
- El lenguaje para generar consultas de los modelos.
- El lenguaje para escribir definiciones de transformaciones.

Estos estándares también son conocidos como metamodelos de cuatro capas, donde en la capa M0 se efectúan las instancias, es decir, el sistema real. Mientras que la Capa M1 se define el modelo en general que va a tener la aplicación. En la Capa M2 se generan los metamodelos donde se definen las clases y atributos del sistema real, por último, la Capa M3 donde se organizan los metas – metamodelos para la traducción automática de todo el sistema generado en el proceso de negocio.

1.1.6. ESTUDIO COMPARATIVO DEL DESARROLLO TRADICIONAL VS DESARROLLO MDA

El proceso de desarrollo tradicional cuenta con algunos problemas en el desarrollo de software, pero, aunque existen algunas inconsistencias los desarrolladores siguen empleando esta forma de estructuración para expresar un software, en la figura 14 se muestra el proceso o esquema tradicional del desarrollo de software.

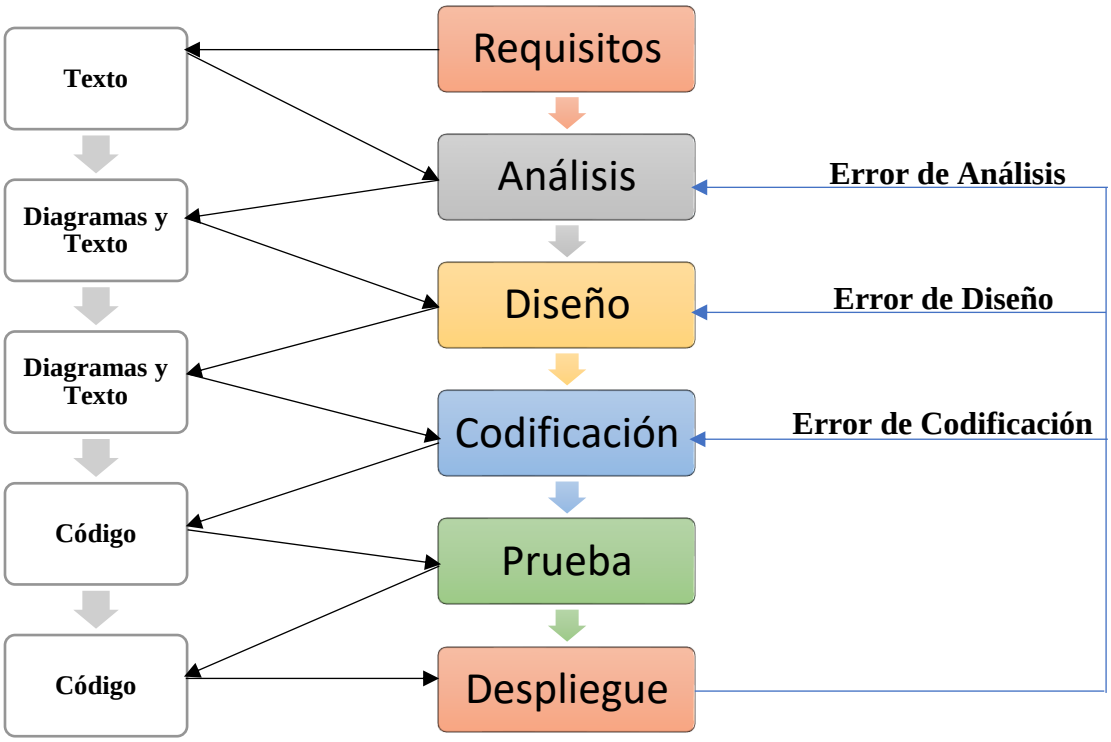


Figura 14: Proceso de Desarrollo Tradicional

Mientras que la arquitectura dirigida por modelo combina múltiples modelos que hacen de sus ventajas algo novedoso e importante para las empresas desarrolladoras de software, donde todo debe estar bien organizado, entendible y comprensible para otros miembros de una misma organización o para comunicar al cliente con los avances de la aplicación que se están realizando. A continuación, en la figura 15 se muestra el proceso de desarrollo MDA.

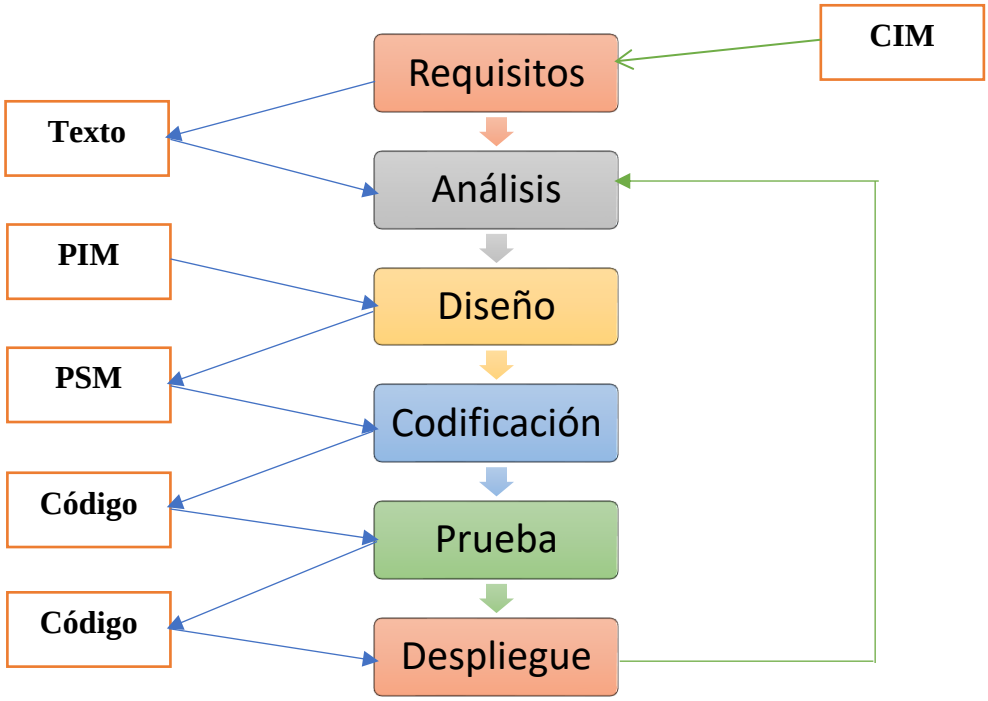


Figura 15: Proceso de Desarrollo MDA

En consecuencia, los procesos de desarrollo tradicionales se están viendo obsoletos ante el desarrollo MDA que propone un reúso de modelos para realizar modificaciones, actualizaciones de nuevas versiones del sistema de software.

Por lo tanto, se organizan las ventajas y desventajas que ofrece esta metodología ante el desarrollo tradicional lo cual se expresa en la tabla 1 a continuación descrita.

	DESARROLLO TRADICIONAL	DESARROLLO MDA
Interoperabilidad	Conectados entre sí, pero dependen de otros archivos para mantenerse operativos.	Un PIM genera varios PSM relacionados y pueden pertenecer a distintas tecnologías.
Portabilidad	Al mantenerse la tecnología en constante cambio las empresas se ven obligadas a desembolsar importantes gastos para mantener en auge el proyecto que se está llevando a cabo.	En esta parte los procesos también recaen en los PIM y a partir de estos generar automáticamente los PSM independiente de cualquier tecnología.
Productividad	Al no mantener una documentación estable, esto complica la comunicación del sistema cuando se dan cambios en la tecnología.	Una vez realizada la transformación de PIM – PSM se centra en los requerimientos del usuario final.
Mantenimiento y Documentación	Debido a que los desarrolladores en el proceso tradicional no llevan una buena documentación esto genera pérdida de tiempo y no permite un entendimiento del código fuente lo que ocasiona doble gasto tanto en tiempo como en costo ya que se vuelve a reorganizar todo el proyecto.	Debido a que el PIM contiene todas las especificaciones del usuario para organizar todas las ideas este obtiene todos los cambios que se verán reflejados en el PSM y el código fuente, permitiendo el reúso de los archivos y brindando un buen mantenimiento de producción.

Tabla 1: Comparativa Desarrollo Tradicional VS Desarrollo MDA

El desarrollo de aplicaciones web mediante el uso de herramientas MDA hace que las operaciones relacionadas con las funcionalidades del sistema deben cumplirse en cuestión de minutos, ahorrando costo y tiempo de producción.

1.1.7. CUADRO COMPARATIVO DE HERRAMIENTAS MDA

Para demostrar las características de las herramientas de Arquitectura Dirigida por Modelo se establece un cuadro comparativo que especifica las funcionalidades o procesos que se pueden realizar, tal como se muestra en la tabla 2.

CARACTERÍSTICAS	STARUML	ENTERPRISE ARCHITECT	MICROSOFT VISUAL STUDIO
Entrada	UML 2.x	UML 2.5	UML
Lenguaje Generador	Java, C#, C++	ActionScript, C, C#, C++, Delphi, Java, PHP, Python, Visual Basic, Visual Basic .NET, DDL, EJB, XML Schema, Ada, VHDL, Verilog, WSDL, BPEL, Corba IDL	Java, C#, C++, PHP
Interoperabilidad	Si	Si	Si
Soporta CIM	-	Si	-
Soporta PIM	Si	Si	Si
Soporta PSM	Si	Si	Si
Soporta XMI o XML	Si	Si	Si
Transformación CIM – PIM	-	Si	-
Transformación PIM – PSM	Si	Si	Si
PIM – Código	Si	Si	Si
PSM – Código	Si	Si	Si
Trazabilidad	Si	Si	Si
Plantilla	Si	Soporta plantilla MDA y plantilla de generación de código	Si
Ingeniería Inversa	Java, C#, C++, Generador de Código e Ingeniería Inversa	ActionScript, C, C#, C++, Delphi, Java, PHP, Python, Visual Basic, Visual Basic .NET, DDL, XML Schema, WSDL	Java, C#, C++
Se Puede Integrar con	Java Script, Node.js	Eclipse y Visual Studio	Con cualquier Lenguaje de Programación

Tabla 2: Cuadro Comparativo de Herramientas MDA

En este cuadro comparativo se muestran las características que relevan el funcionamiento de estas herramientas donde comprenden un sinnúmero de estándares, lenguajes de entrada, lenguajes de salida, ingeniería inversa, programas con los que puede ser integrado, plantillas que utilizan, transformaciones CIM – PIM o PIM – PSM, etc.

1.2. ANTECEDENTES

En términos generales la Arquitectura Dirigida por Modelos impulsa a los programadores a llevar una documentación bien estructurada partiendo de los diagramas de Lenguaje de Modelado Unificado donde estén encapsulados los diferentes niveles de abstracción del sistema. En cuanto a esto la OMG cuenta con paradigmas que son útiles para el estudio de modelos dirigidos, esta cuenta con el MDD (Desarrollo Dirigido por Modelo) mejora la consistencia y calidad de las soluciones a problemas generados en el desarrollo, al automatizar los patrones de implementación con la finalidad de eliminar los procesos repetitivos y de bajo nivel en el desarrollo de sistema.

Según Texier *et al.* [18] dicen que: "Model Driven Development (MDD), es un paradigma de construcción de software cuyas motivaciones principales son la independencia de los productores de software a través de estandarizaciones y la portabilidad de los sistemas de software". A partir de esto, el objetivo del Desarrollo de Software Dirigido por Modelos es separar la arquitectura tecnológica del diseño del sistema con la finalidad de organizar las ideas en modelos concretos y abstractos, los cuales sino alcanzan el grado de abstracción esencial se transforman en metamodelos que son analizados nuevamente para ser entendidos por un ordenador. Estos metamodelos conforman una parte de la retroalimentación de los procesos abstractos en los modelos, es decir, si un modelo es demasiado extenso necesita ser dividido en modelos más pequeños que puedan ser estudiados y generar ideas más concretas.

Por lo tanto, el desarrollo dirigido por modelo hace referencia a las definiciones formales del desarrollo de software, mientras que la OMG (Object Management Group) promueve la implementación de la arquitectura dirigida por modelos mediante sus herramientas que permiten al desarrollador llevar a cabo el proceso de implementación y mantenimiento del software. Garcia & Fonseca en su artículo científico sobre el Desarrollo Rápido de Aplicaciones apoyada en MDA expresa que; "MDA es separar la especificación de los sistemas de los detalles de su implementación en una determinada plataforma" [19].

Dichas plataformas permiten transformar los modelos independientes en modelos específicos que aprovechen las inversiones en los requisitos, es decir, todo sistema de desarrollo de software debe estar expuesto al cambio de tecnología a largo plazo de las implementaciones por tal motivo MDA permite la interoperabilidad, portabilidad y la reutilización de los procesos de desarrollo.

Gaitán [20] menciona que: la generación automatizada en una línea de producto software se convierte en una herramienta poderosa que facilita la transformación y generación de gran número de artefactos, reduciendo considerablemente el tiempo de desarrollo e incrementando la productividad a partir de desarrollos ágiles. Por lo tanto, permite la reutilización de código para potencializar la flexibilidad de diversos desarrollos de software, por lo cual, utilizan esta metodología para la implementación de Patrones de diseño como MVC (Modelo, Vista, Controlador) presente en sistemas operativos de tipo Unix (GNU) que permiten la generación de código para transformarlo en sistemas transaccionales CRUD, usando plataformas como PHP, JSP, ASP, etc.;

Sayago [21] menciona que el generador pretende automatizar el proceso de codificación de las anotaciones de persistencia e implementación de los métodos con Java Persistence API (JPA) del Data Access Object (DAO) de la Entidad para el acceso a la base de datos. Esto se pretende realizar a través del lenguaje de programación Ruby que permite la integración de librerías para el entorno gráfico que se va a realizar, optimizando la calidad y tiempo en el desarrollo de un producto de software.

Estévez *et al.* [22] expresan que pese a que todos los trabajos comentados previamente hacen uso del diseño basado en modelos, pretende ir un paso más allá: tiene como objetivo especificar unas pautas de diseño de aplicaciones robóticas fomentando la reutilización de código de una manera independiente a la plataforma de ejecución y además haciendo uso de una notación estándar. Actualmente la ingeniería dirigida por modelos (MDE) brinda grandes ventajas en el diseño y desarrollo de aplicaciones que permitan dar movimiento a un robot, para ello se va a utilizar los diagramas UML que representan todo el diseño de la aplicación transformando los datos en componentes más entendibles a la vista del usuario. Empleando así dos métodos de estudio el primero llamado OROSCOS que tiene como funcionalidad la comunicación basada en componentes (Robótica Manipuladora Industrial) y el segundo para mostrar la reutilización de código (robótica manipuladora de servicios).

Bernal *et al.* [23] relatan en su artículo sobre el manejo de la herramienta Simcam2Cloud la cual debe ser configurada para la generación de framework, mediante la implementación de perfiles UML, a través de la herramienta de modelado y meta modelado del sistema de desarrollo. SIMCAM es una plataforma de simulación orientada a analizar y estudiar aplicaciones paralelas en sistemas distribuidos. Ha sido diseñada para proporcionar flexibilidad, precisión, rendimiento y escalabilidad. Estas características la convierten en una potente plataforma de simulación para diseñar, probar y analizar tanto arquitecturas reales como inexistentes. Ha sido desarrollada utilizando los frameworks de simulación INET1 y OMNeT++2. La gama de sistemas a simular puede variar desde un único nodo de computación hasta un sistema distribuido de alto rendimiento completo. Para lograrlo, sigue un diseño modular que permite simular un amplio rango de arquitecturas con sólo unos pocos módulos implementados y una configuración bien definida.

Para mantener el equilibrio entre rendimiento, precisión, flexibilidad y escalabilidad, el núcleo principal de esta plataforma sigue un diseño flexible dirigido a la integración de cada sistema básico en un solo modelo de simulación. Los sistemas básicos son el sistema de almacenamiento, el sistema de memoria, el sistema de procesamiento y el sistema de red.

Por otro lado, Moreno *et al*. [24] mencionan que: la forma de capturar los aspectos de usabilidad que se emplean para generar una aplicación web, donde los modelos son la parte principal y esencial de todo sistema de software y mediante estos proceder a la transformación de los procesos de negocio para conseguir los modelos conceptuales del programa. El desarrollo de los sistemas Web en estos métodos se lleva a cabo mediante modelos que capturan distintas vistas del sistema: un modelo estructural (modela contenido y comportamiento), un modelo de navegación (modela acceso al contenido) y un modelo de presentación abstracto (modela cómo el contenido es mostrado). La evaluación de estos modelos puede proporcionar información, para evaluar la usabilidad en etapas tempranas del desarrollo antes de que cualquier línea de código sea generada. Se debe considerar el nivel de abstracción de los modelos para poder evaluar las características de usabilidad. Una vez obtenido un modelo conceptual (Casos de Usos, Diagramas de Interacción, etc.), partiendo de un modelo de negocios sería interesante poder elaborar modelos abstractos donde se pueda aplicar la evaluación de Usabilidad y

Accesibilidad simultáneamente. Esto es posible aplicando teoría de grafos empleando Autómatas Finitos Deterministas.

Arsaute *et al.* [25] dicen que las API REST desplazaron a SOAP y WSDL al ser un servicio web que permite consumir los recursos en cuestión de instante, mediante la implementación de algunos métodos específicos de esta arquitectura. La contribución principal de este trabajo es proponer un mecanismo de transformación de programas JAVA, haciendo uso de estándares y herramientas existentes en el contexto de desarrollo dirigido por modelos, obteniendo como resultado una aplicación API REST que permite la interacción entre sistemas heterogéneos, promoviendo así la gestión de modelos en entornos distribuidos. El mecanismo de generación de código es muy particular, dado que está basado en el tratamiento de los modelos JAVA (XMI), y no sobre el código fuente JAVA, dada la existencia de nuevas herramientas que permiten obtener dicho modelo, en caso contrario sería necesario manipular un parser del lenguaje.

Debido a los procesos cambiantes en la tecnología y diversas aplicaciones se creó una segunda versión del sistema de crianza de camarones el cual será sometida a una reingeniería bajo la metodología de desarrollo de software Iconix es una metodología de desarrollo ágil, la cual se encuentra entre la complejidad del RUP (Rational Unified Processes) y la simplicidad y pragmatismo de XP (Extreme Programming) sin eliminar las tareas de análisis y diseño que XP no contempla. Iconix es una metodología de desarrollo ágil que unifica métodos de orientación a objetivos con el fin de abarcar todo el ciclo de vida de un proyecto, además presenta claramente las actividades de cada etapa. Algunas de las características más importantes de esta tecnología están: Iterativo e incremental, Trazabilidad, Dinámica [26].

1.3. BASES LEGALES

Esta investigación es desarrollada con el fin de conocer los criterios especificados de varios artículos en donde menciona la estructura que debe tener un software, con la finalidad de expresar los requerimientos de un software al ser desarrollo; por lo que basándose en el estándar IEEE 830 de la Especificación de Requerimiento de software el cual tiene como propósito cumplir con todas las características de desarrollo que se deben implementar en un software.

Una buena especificación de requisitos software ofrece una serie de ventajas entre las que destacan el contrato entre cliente y desarrolladores (como ya se ha indicado con anterioridad), la reducción del esfuerzo en el desarrollo, una buena base para la estimación de costes y planificación, un punto de referencia para procesos de verificación y validación, y una base para la identificación de posibles mejoras en los procesos analizados. [27]

También Flores & Pino [28] destacan que; El proceso de Elicitación de Requisitos (ER) se refiere a todo sobre aprender y entender las necesidades de los usuarios y de los interesados del proyecto de software con el objetivo de comunicar estas necesidades a los desarrolladores del sistema.

Estos requerimientos de software ayudan al desarrollador a crear estrategias de información y conocimientos para la comunicación con el cliente haciéndole entender el procedimiento y propósito de todos los modelos que se están generando a través de las metodologías y tecnología MDA, analizando y sintetizando los principios por los cuales está siendo creada esta aplicación estableciendo la relación entre el futuro y el entorno del sistema.

También al ser esta una investigación de innovación y conocimiento de nuevas tecnologías para la documentación de procesos de desarrollo de software, las bases legales en las cuales se va a centrar esta investigación consisten en el Código Orgánico de Economía Social del Conocimiento e Innovación, o Código Ingenios de la República del Ecuador (2017), donde se atribuyen los siguientes artículos:

Artículo 14, se refiere a las Entidades de Investigación Científica emitidas por la entidad rectora del Sistema que dedica sus actividades a la investigación científica, al desarrollo tecnológico, o servicios relacionados.

El Artículo 42, que hace referencia a la Investigación Responsable, y establece comprender los procesos investigativos encaminados a obtener resultados orientados al incremento de la productividad, la diversificación productiva, la satisfacción de necesidades o al efectivo ejercicio de los derechos de las personas las comunidades, los pueblos, nacionalidades y de la naturaleza.

A partir de esto, el Articulo 76, se refiere al Emprendimiento innovador, donde establece que un proyecto orientado al desarrollo de un nuevo o significativamente mejorado bien

o servicio cuyo factor fundamental es el uso del conocimiento que se genera a partir de procesos de investigación, desarrollo experimental y tecnológico o procesos creativos con base científica, cuyo fin último es su introducción en el mercado.

Los artículos antes mencionados hacen referencia a las bases legales establecidas en el Código Ingenios para hacer del desarrollo de software una tecnología de innovación en donde prevalezca la comunicación entre cliente y desarrollador.

CAPÍTULO II:
MATERIALES Y MÉTODOS

2.1. TIPO DE ESTUDIO

La investigación elaborada fue experimental puesto que se basó en la recolección de datos científicos para un posterior análisis y resultados, por tal motivo, según los objetivos planteados se analiza la metodología que ofrece la Arquitectura Dirigida por Modelo para obtener conocimientos relevantes sobre la automatización de modelos dirigidos en el desarrollo de aplicaciones.

También es considerada como una investigación exploratoria ya que una vez analizada esta metodología se procede a la práctica mediante las herramientas de arquitectura dirigida por modelos para describir las características fundamentales de los distintos modelos y estándares que genera este paradigma con la finalidad de cumplir con los lineamientos establecidos de los requerimientos específicos de software de la IEEE-830.

Nicomedes [29] afirma que los estudios exploratorios sirven para familiarizarnos con fenómenos relativamente desconocidos, obtener información sobre la posibilidad de llevar a cabo una investigación más completa respecto de un contexto particular, investigar nuevos problemas, identificar conceptos o variables promisorias, establecer prioridades para investigaciones futuras, o sugerir afirmaciones y postulados.

2.2. DEFINICIÓN CONCEPTUAL Y OPERACIONALIZACIÓN DE LAS VARIABLES

Como variables conceptuales de esta investigación se analizó los modelos como parte fundamental del desarrollo de software organizando la idea principal en diagramas representativos del Lenguajes de Modelado Unificado (UML), también a través de los metamodelos se construyó modelos más abstractos con un alto nivel de comprensión para su posterior transformación en código fuente. Esta transformación se la realiza mediante las herramientas MDA orientadas al diseño y construcción de modelos dirigidos con la finalidad de ahorrar recursos de implementación.

Además, la metodología MDA permite generar arquitectura MVC (Modelo – Vista - Controlador) ya sea mediante la utilización de una Base de Datos o de una idea global

que incluya el uso de los diagramas UML automatizando la transformación de código para su implementación en una aplicación web. Por tal motivo, la operatividad de esta aplicación consiste en demostrar la factibilidad de uso del modelo, la dimensión de estudio a través de los lineamientos que muestra los resultados de esta investigación. Esto no significa que la programación va a ser fácil sino va a permitir generar documentación relevante para una posterior actualización del sistema.

2.3. MÉTODOS Y TÉCNICAS

Para esta investigación fue implementado el método inductivo porque permite obtener conclusiones generales a partir de afirmaciones o ideas presentadas para el desarrollo de una aplicación web mediante el diseño y construcción de modelos dirigidos que permitan la transformación automática de código fuente. Debido a que el método inductivo cuenta con cuatro pasos esenciales: primero se observará el modelo abstracto independiente de la plataforma para posteriormente ser diseñado en la herramienta MDA a través de los diagramas específicos de UML, a medida que avanza el proyecto se estudian todos los hechos que acontecen de estos modelos y por último se llega a la obtención del aplicativo web, permitiendo la transformación inductiva de los hechos llegando a la generalización y contracción de estos resultados.

Adicional a esto se estableció la técnica de la observación, la cual se detalla en los resultados a través de la metodología aplicada a los procesos de desarrollo de una aplicación web dirigida por modelos.

2.4. INSTRUMENTO

En el desarrollo y estudio de los modelos dirigidos se investigaron las características funcionales de herramientas que permiten el diseño, desarrollo e implementación de arquitectura dirigida por modelo como son: (StarUML, Enterprise Architect, Microsoft Visual Studio ASP.NET – Entity Framework) que muestran el proceso de transformación y creación de los modelos a través de diagramas UML, estos diagramas son paquetes de diversos requerimientos que son expresados por el cliente hacia el desarrollador. Debido a esto el trabajo se vuelve colaborativo al mantener cada paso documentado a través de paquetes de estudio los cuales son repartidos a los distintos departamentos de

colaboración aumentando el nivel de exactitud y cumplimiento en el plazo establecido durante el periodo de concepción del sistema.

CAPÍTULO III:
RESULTADOS

En el proceso de desarrollo de esta investigación se analizaron algunos aspectos que hacen de la Arquitectura Dirigida por Modelo (MDA) un paradigma dedicado a la modelación y documentación de procesos de software con el apoyo de los diagramas UML.

Posteriormente, para evaluar las herramientas antes expuestas en el cuadro comparativo se ha diseñado un modelo de base de datos, el cual identifica las características de un Curriculum Digital Docente.

Este sistema permitirá al docente ingresar sus datos personales y así plasmar mediante un PDF la información ingresada. Este procedimiento se identifica como un CRUD de los sistemas automatizados.

El sistema debe contar con las siguientes funcionalidades:

- Autenticación del docente mediante login/ password.
- Consulta de datos personales del docente.
- Realizar o generar PDF de los datos ingresados.

A continuación, se detallan los diagramas UML que se generan a través de las herramientas seleccionadas en el transcurso de esta investigación (StarUML, Enterprise Architect y Microsoft Visual Studio ASP.NET Entity Framework).

3.1.StarUML

Es una herramienta de software libre u open source cuyo logo está expresado en la ilustración 3; sirve para el modelado de procesos mediante la especificación UML, debido a esto se podrá construir, visualizar y documentar un sistema.



Ilustración 3: Logo StarUML [30]

En la ilustración 4, se identifica la interfaz gráfica de la herramienta, donde se puede seleccionar diversos elementos para la creación de modelos, diferentes funcionalidades las cuales se encuentran en la barra de herramientas y en ambos lados de la interfaz. Esta permite el diseño y desarrollo rápido gracias a las múltiples abreviaturas que soporta la herramienta (Quick Edit) cuenta con reglas asincrónicas mediante la validación de modelos UML que han sido implementados en StarUML, estas reglas se revisan cuando se guarda o se abre cualquier archivo para continuar con la edición del sistema de software sin ningún inconveniente y cero errores.

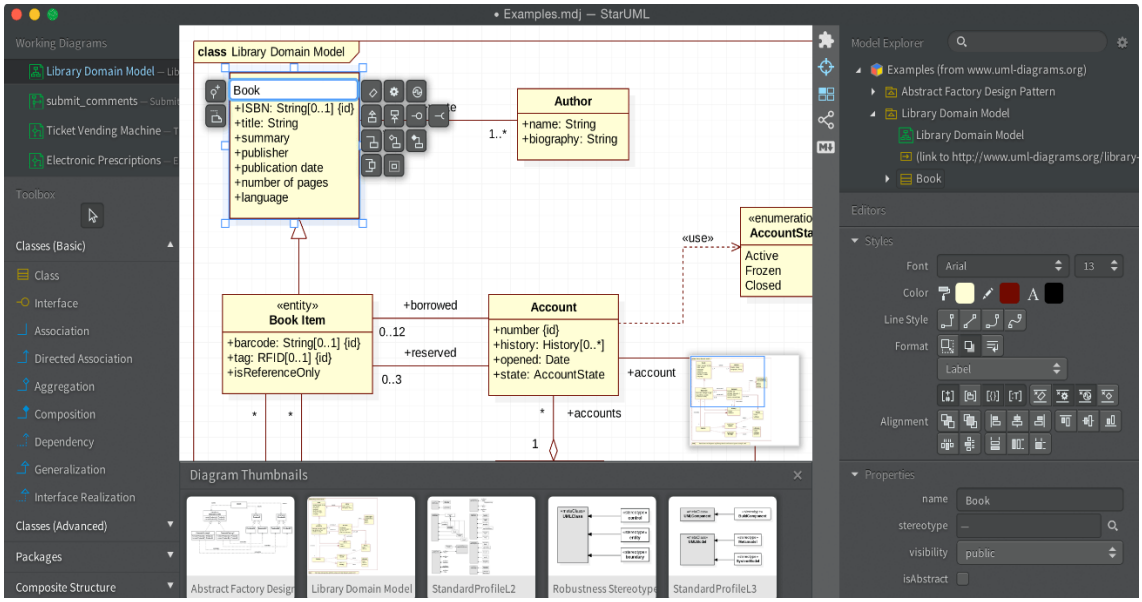


Ilustración 4: Interfaz Gráfica de StarUML

StarUML es un sofisticado modelador de software destinado a soportar un modelado ágil y conciso. Los principales objetivos de los usuarios son: Equipos de desarrollo ágil y pequeño, personas profesionales e instituciones educativas [30].

Modelado

StarUML utiliza la especificación UML 2 que es compatible con UML 2.x, también cuenta con metamodelos estándares y 11 diagramas de lenguaje de modelado unificado, aparte de permitir los diagramas UML también se podrán crear ERD (Entity-Relationship Diagrams), diagramas de flujos de datos (DFD) y diagramas de diagramas de flujos. Cuenta con el soporte de multiplataforma ya que trabaja con el mismo UX.

Es compatible con la tecnología MDA (Model Driven Architecture), la cual propone un proceso basado en modelado y transformación de modelos, es decir, que el cometido

principal de la MDA es crear modelos independientes de la plataforma para aumentar la extensión de UML.

De tal manera, StarUML soporta Retina Display puesto que permite mostrar los diagramas de forma clara o nítida en cualquier formato de alta resolución de imágenes como son (JEPG, PNG). También permite administrar extensiones de código abierto alojadas en Github. Por otro lado, el desarrollo está basado en modelos con formato JSON muy simple, por lo que se puede usar fácilmente para la generación automática de código fuente. A parte StarUML soporta lenguajes de programación de alto nivel tales como C#, Java y C++, los cuales permiten la ingeniería inversa y la generación de código fuente a partir de los modelos o viceversa.

Por tal motivo, cualquier usuario puede escribir extensiones utilizando HTML5, CSS3, JavaScript y Node.js. a través de la Open APIs se pueden crear interfaces de usuarios, menú, cuadro de diálogos, incluso lenguajes propios de modelado incluyendo metamodelos, esto permite las anotaciones de reglas y validación para aumentar las posibilidades de diseño mediante la adaptación de este.

Básicamente, StarUML permite generar tres tipos de modelos:

Creación de diagramas o modelos UML. –Esta herramienta facilita el encapsulamiento de los diversos modelos que se pueden generar cuando se realiza un sistema de software.

A continuación, se desarrolló el diagrama de caso de uso, diagrama de clases y diagrama de componente del aplicativo web a implementar mediante los modelos dirigidos.

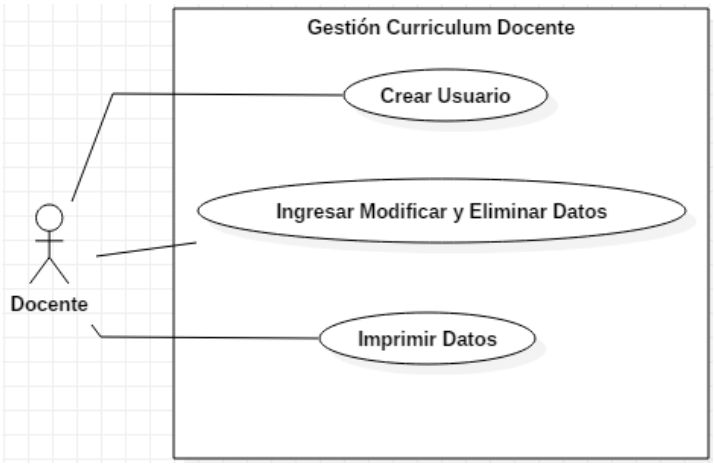


Ilustración 5: Diagrama de Caso de Uso

En la ilustración 5 se muestra el diagrama de caso de uso representando los procedimientos que se van a realizar en la aplicación web.

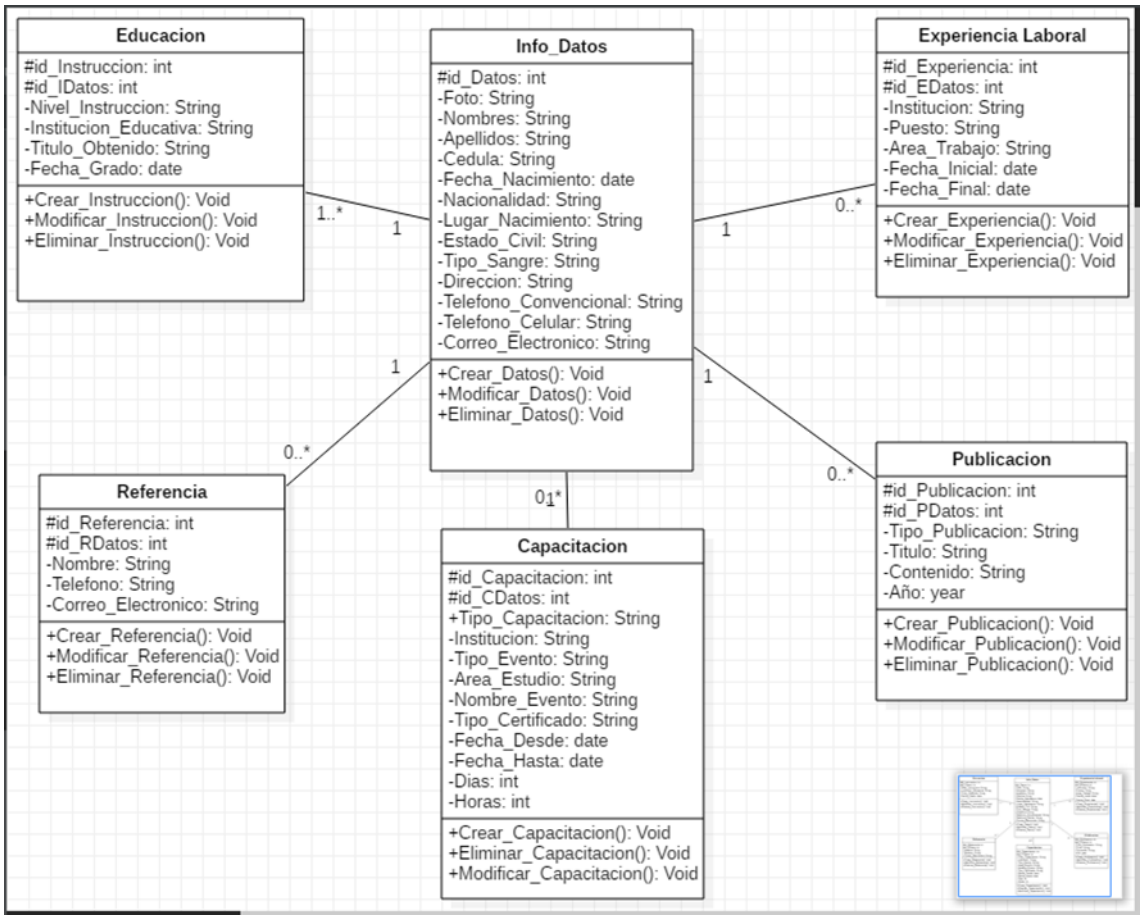


Ilustración 6: Diagrama de Clases

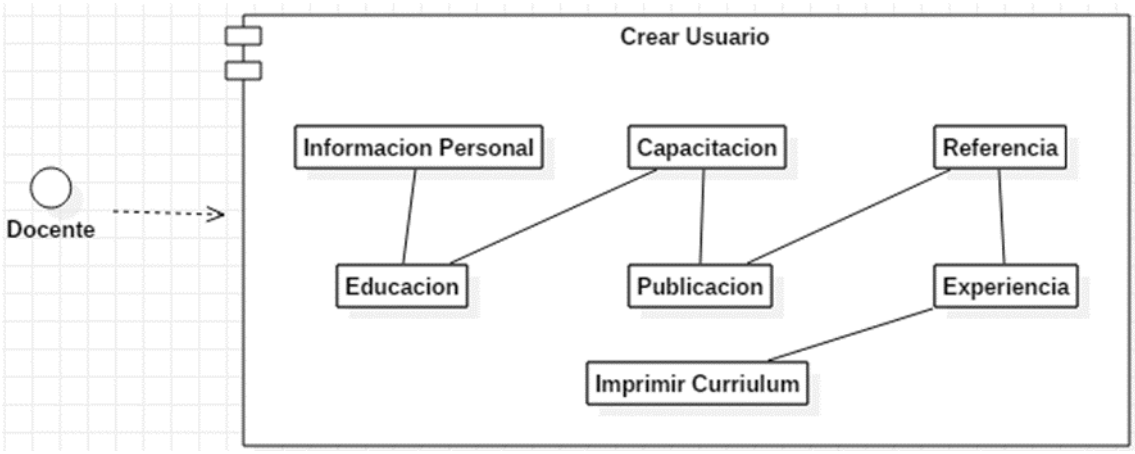


Ilustración 7: Diagrama de Componente

Por lo tanto, StarUML organiza todos los pasos a seguir dentro un sistema encapsulándolos en paquetes dinámicos y comprensibles para cualquier miembro de la empresa desarrolladora de software. Así pues, estos paquetes aumentan el trabajo colaborativo de los miembros de una empresa, disminuyendo el tiempo de producción de un producto.

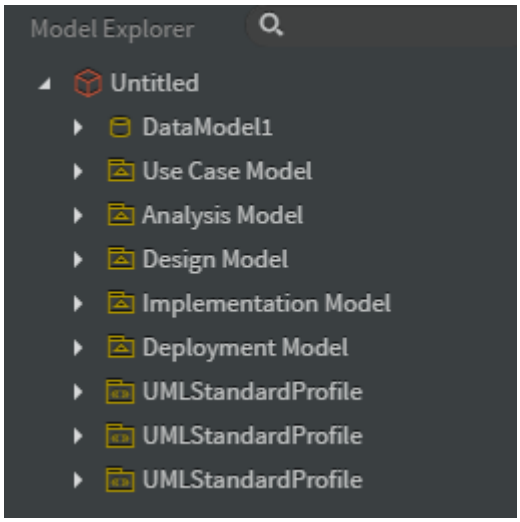


Ilustración 8: Encapsulamiento de los Modelos

Dentro de todo el marco de desarrollo StarUML ofrece a los desarrolladores la opción de generar código fuente automáticamente partiendo de un modelo específico.

Dentro de todo el marco de desarrollo StarUML ofrece a los desarrolladores la opción de generar código fuente automáticamente partiendo de un modelo específico.

Generación de código. – Una vez diseñado el modelo del sistema a desarrollar, se debe crear el perfil del modelo, es decir, describir en qué lenguaje va a estar prescrito el sistema para así generar el código fuente automático.

Partiendo del diagrama de clases antes mencionado se muestran a continuación los códigos que permiten generar con sus respectivos métodos, constructores y atributos.

El código generado está expresado en el lenguaje C#, Java y el Framework Django, el cual será demostrado en el **(ANEXO 1)**.

StarUML organiza toda la estructura del código fuente con sus métodos, atributos, relaciones y constructores con la diferencia que al programador le corresponde especificar los procesos que va a realizar cada operación.

Generación de documento. – Una vez creado el modelo y generado el código, se procede a la documentación de este, la cual se muestra mediante la generación de un archivo HTML que está plasmado en el **(ANEXO 2)**.

3.2. ENTERPRISE ARCHITECT

En la ilustración 9 se muestra el logo de la EA (Enterprise Architect) es una herramienta que cuenta con licencia de propietario como open source, el cual a través de su interfaz gráfica permite la combinación de varios elementos para modelar o diseñar las diversas especificaciones de requerimientos que se emplean en una organización para cubrir la implementación de un sistema de software.

Ilustración 9: Logo Enterprise Architect [31]

Descripción

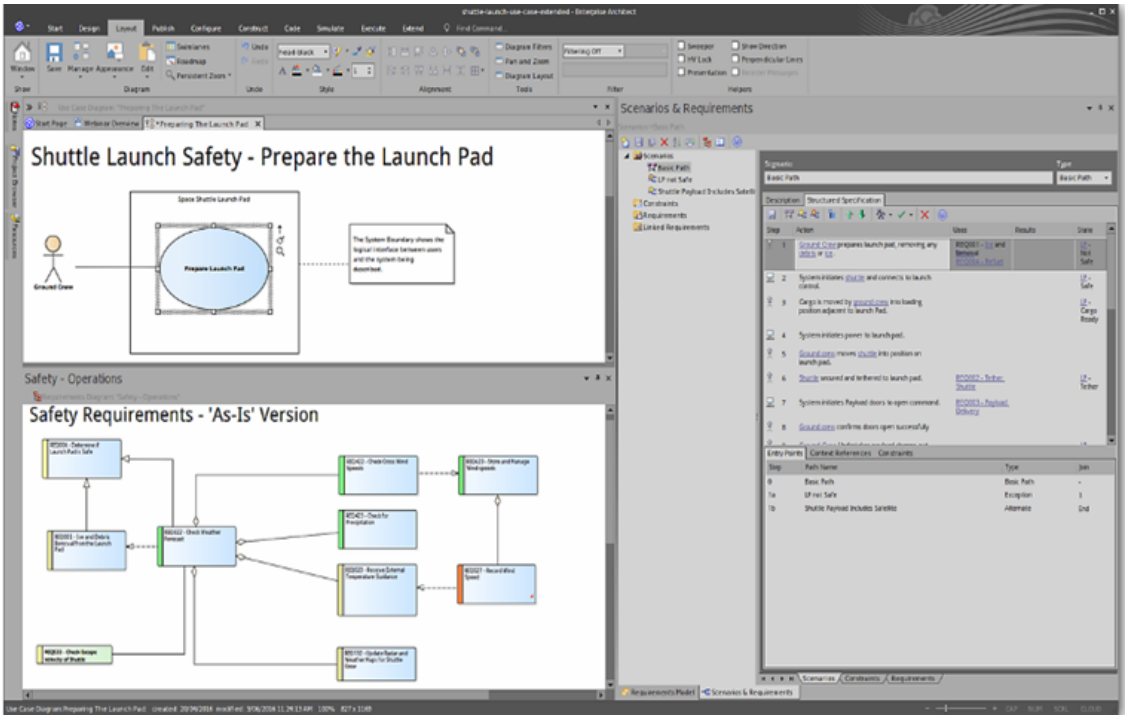


Ilustración 10: Interfaz de Enterprise Architect [31]

En la ilustración 10 se describe la interfaz gráfica, barras de herramientas, pantalla de edición de los modelos y algunas funcionalidades con las que cuenta la EA. Por lo tanto, la EA una herramienta de diseño y modelado MDA que trabaja mediante los estándares de la OMG agilizando los procesos de desarrollo de un sistema. Esta arquitectura empresarial permite el diseño y construcción de modelados de procesos de negocios, modelado de dominio basado en la industria, esta herramienta es utilizada no solo para la

estructura u organización de un sistema, sino también en la implementación de los modelos durante todo el ciclo de vida de un sistema.

Las arquitecturas empresariales ofrecen el desarrollador características específicas de elaboración, permitiendo la colaboración de todo el equipo de desarrollo dentro de una empresa o industria, al permitir generar tareas desglosadas en varios paquetes de modelos garantizando el cumplimiento de todas las actividades encomendadas a cada rol optimizando tiempo y costo de productividad.

Acompañamiento en todo el proceso de desarrollo, es decir, permite la colaboración de cliente – desarrollador.

Administración de modelos UML, cuenta con la innovación de los diagramas UML para el desarrollo de modelos empaquetados en diferentes marcos estructurales.

Generación de reportes. – A partir de los modelos creados en la EA mediante el Lenguaje de Modelado Unificado (UML) se generan automáticamente los reportes de todo el proceso de desarrollo.

Administración de proyectos. – Al ser EA una empresa que trabaja en base la colaboración de todo su equipo de trabajo permite organizar y estructurar de mejor manera todos los requerimientos del cliente.

Generación de código. – Muestra el código fuente a través de la automatización de los modelos.

Ingeniería inversa. – Permite una ingeniería de ida y vuelta mejorando desfaces en la implementación del sistema.

Modelos de datos. – Enterprise Architect permite la ingeniería directa o inversa, creando así la admisión de modelados de datos a nivel conceptual o físico.

Modelado de XMI o XML. – es un lenguaje de marcado de propósito general, es decir que XML no está definido, sino que permite definir nuestras propias etiquetas.

Transformación MDA. – como ya se explicó anteriormente estas transformaciones permiten convertir los elementos de un modelo a otro.

Pero en fin la EA a través de la MDA promueve el trabajo corporativo y colaborativo de todo el personal de desarrollo de un sistema de software permitiendo la documentación

de todo el proceso de negocio para que posteriormente en un futuro sea actualizado a través de las innovaciones que requiera el cliente.

Enterprise al igual que StarUML organiza todo el proceso de desarrollo mediante diagramas UML encapsulando los procesos y distribuyendo el trabajo según el departamento asignado.

Modelo de Domino. – es un modelo de alto nivel que muestra la arquitectura general del sistema. Este viene a ser la Plataforma Independiente de Modelo, es decir, el PIM.

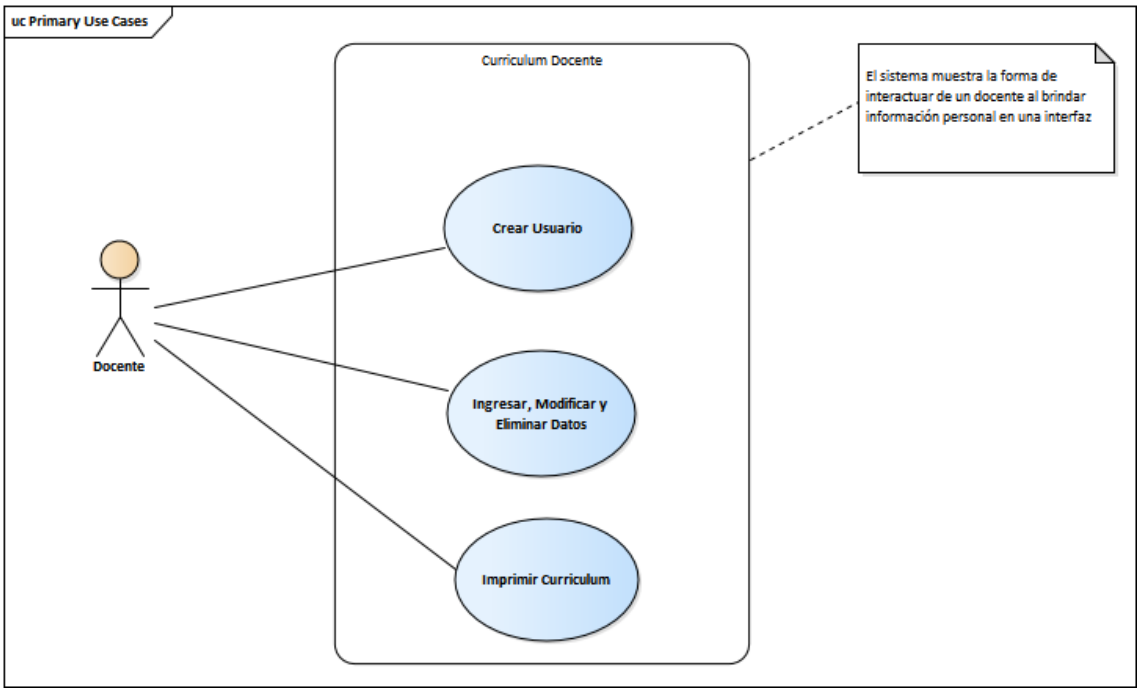


Ilustración 11: Diagrama de Caso de Uso EA

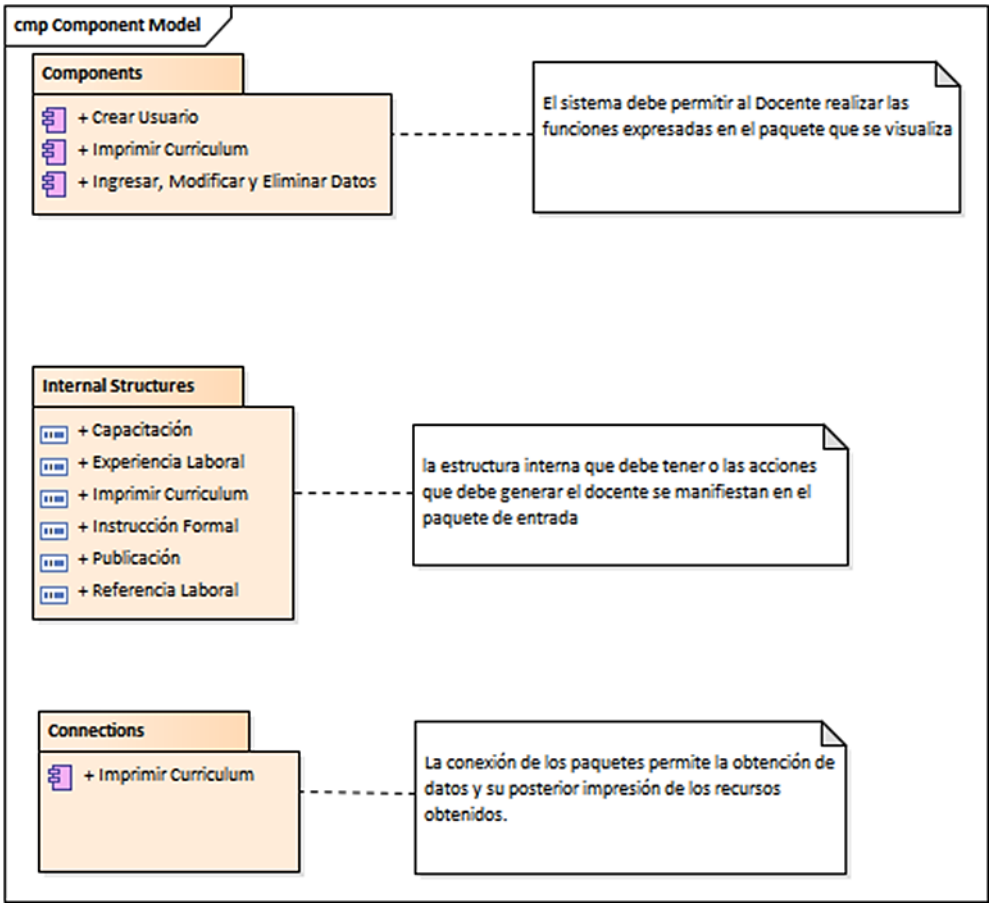


Ilustración 12: Diagrama de Componente

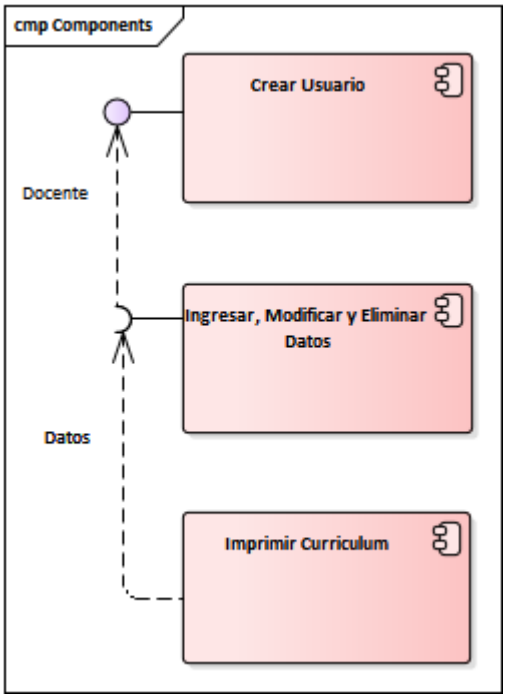


Ilustración 13: Componentes EA (Aplicativo Web)

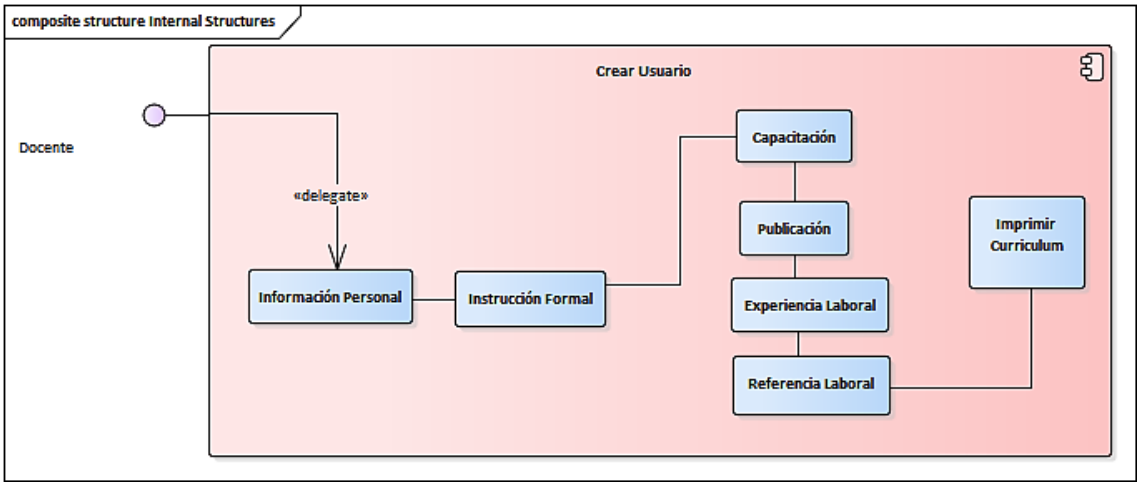


Ilustración 14: Estructura Interna del Diagrama de Componente

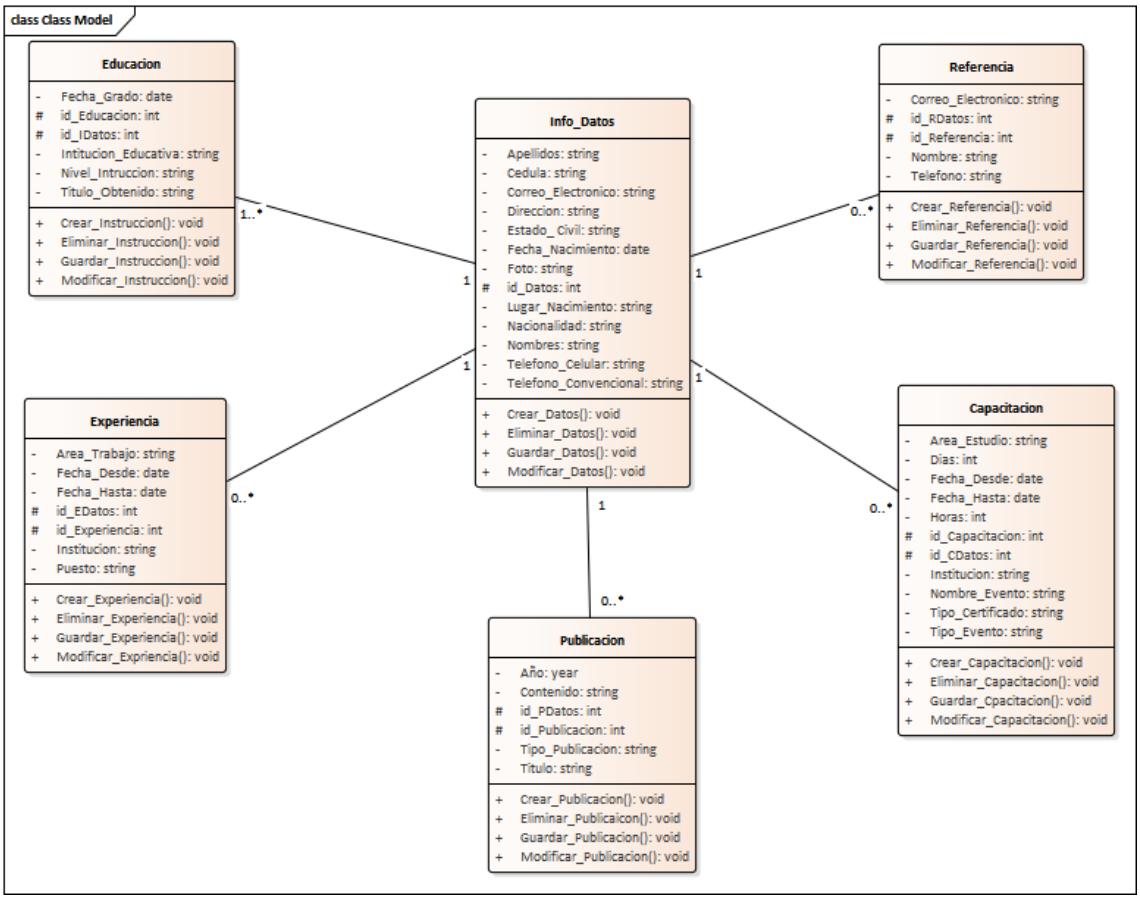


Ilustración 15: Diagrama de Clases EA

Modelo de Presentación. – Enterprise Architect orienta a sus clientes en el producto final que van a obtener mediante la muestra de la posible interfaz gráfica que puede tener el producto. Asimismo, se combinan los modelos PIM – PSM para reorganizar la estructura del sistema en desarrollo.

En las ilustraciones 11 – 15 se muestran los procesos de transformación de PIM – PSM.

En el **(ANEXO 3)** se muestra la interfaz gráfica del aplicativo web que se está desarrollando con la diferencia que EA solo ofrece el encapsulamiento de todo el proceso de desarrollo de la aplicación.

Generación de Código. – La obtención de código fuente automático, se realiza mediante la construcción de los modelos antes mencionados, cabe recalcar que la ingeniería inversa en esta herramienta es permitida. Por lo tanto, es mucho más amigable al momento de organizar todo el proceso de ejecución de un sistema de software.

A continuación, en el **(ANEXO 4)** se observa el código generado a partir de los procesos de diseño y construcción de los modelos. Estos códigos están expresados en los lenguajes C#, Java, PHP.

Partiendo de las referencias del cuadro comparativo de las herramientas de Arquitectura Dirigida por Modelos (StarUML, Enterprise Architect y Microsoft Visual Studio ASP.NET Entity Framework) se establece lo sieguiente:

StarUML es un sofisticado modelador de software que encapsula todos los metodos, tablas relaciones que se llevan a cabo en un sistema básico de CRUD en este caso la modelacion de un Sistema de Gestión y Curriculum Docente.

En consecuencia a esto se puede manifestar que StarUML facilita la comprension de un sistema con sus diferenctes roles, debido a que ofrece modelos UML que permiten diseñar todo el proceso que se relaciona al tema, con esto se establece un modelo de caso de uso, modelo de componente y modelo de clase donde se reflejen las operaciones que se van a utilizar en el sistema y posteriormente permitir la transformacion de estos modelos a código hábil para utilizar en la creación de una aplicación web.

Desde un punto de vista, Enterprise Architect permite el trabajo colaborativo de los desarrolladores al organizar en paquetes la estructura del sistema, asimismo permite la ingeniería inversa, realiza matrices de trazabilidad, contiene plugin que ofrecen la comunicación entre visual studio y eclipe.

En relación, a lo antes mecionado se establece utilizar la herramienta de Arquitectura Dirigida por Modelos (Microsoft Visual Studio) debido a que ofrece un alto nivel de absatracción en el diseño de los modelos y metamodelos del sistema, con el fin de obtener código bien estructurado.

3.3.MICROSOFT VISUAL STUDIO ASP.NET ENTITY FRAMEWORK

En esta sección se describe la estructura y software utilizado para desarrollar el aplicativo web y componentes que se utilizan para la transformación automática de los modelos, en primera instancia se realiza una breve introducción sobre el IDE a utilizar, características y componentes, luego se empieza con la estructuración de cómo va estar expresada la aplicación mediante un cuadro (Modelo – Vista – Controlador) conexiones a la base de datos y cómo se extraen los datos, modelos independiente de la plataforma, los modelos específicos de la plataforma, controladores específicos para generar toda la estructura de un CRUD, finalmente mostrar la presentación al usuario y su funcionalidad.



Ilustración 16: Logo Visual Studio [32]

Descripción

En la ilustración 16 se muestra el logo de Microsoft Visual Studio (IDE) Entorno de Desarrollo Integrado que ofrece a los desarrolladores de software una amplia gama de productos integrados a su desarrollo haciendo que los sistemas sean implementados fácilmente y con criterio de retroalimentación y reúso de sus códigos. Soporta múltiples lenguajes de programación, tales como C++, Java, Python, .NET, Visual Basic, C#, entre otros, permitiendo la colaboración entre ellos. También cuenta con un entorno de desarrollo web MVC (Modelo Vista Controlador) a través de ASP.NET, Django, etc., a lo cual se le suma las capacidades online de Azure en forma del editor Mónaco.

Visual Studio permite a los desarrolladores o programadores crear entornos de sitios web de desarrollo al mismo tiempo que aplicaciones web, la transformación MDA mediante las especificaciones independientes de la plataforma, así como también se puede crear aplicaciones que se comuniquen entre estaciones de trabajo, páginas web, dispositivos móviles, etc.

Esta herramienta cuenta con un editor de texto que permite buscar y corregir errores lo más pronto posible. Esta cuenta con una integración en nube a través de las aplicaciones Azure, .NET Core, .NET, etc., al igual que un contenedor Docker, permite la colaboración más eficiente, optimizado para el rendimiento, permite crear el IDE ideal para la carga de trabajo, a continuación, en la ilustración 17, se muestra la interfaz gráfica de Microsoft

Visual Studio 2017 última versión colaborativa con la comunidad desarrolladora de software.

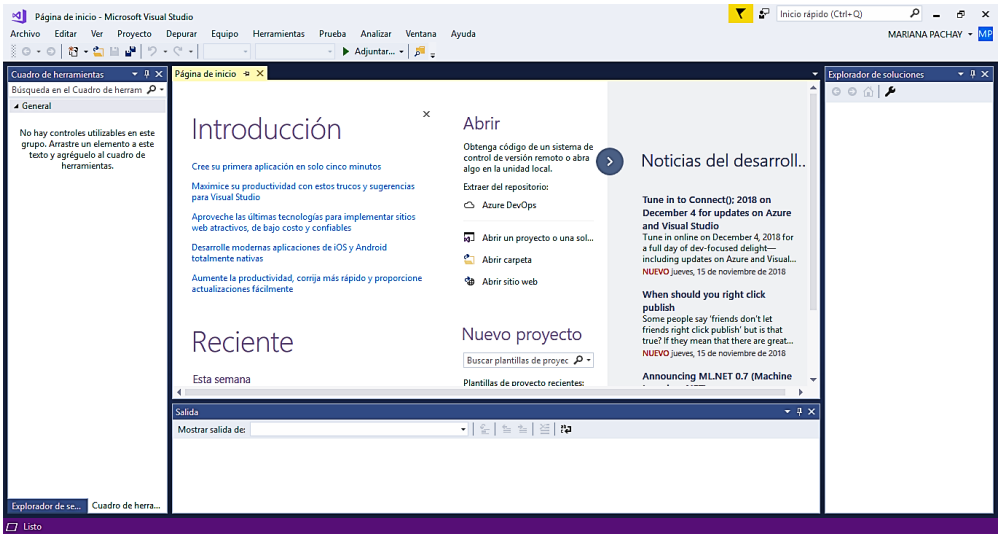


Ilustración 17: Entorno de Desarrollo Microsoft Visual Studio

A partir de lo antes mencionado se considera una de las herramientas con mayores funcionalidades para hacer el trabajo de los diseñadores de modelos, programadores y cliente un entorno amigable para comunicar y entender las estrategias que se están implementando con todas las características requeridas.

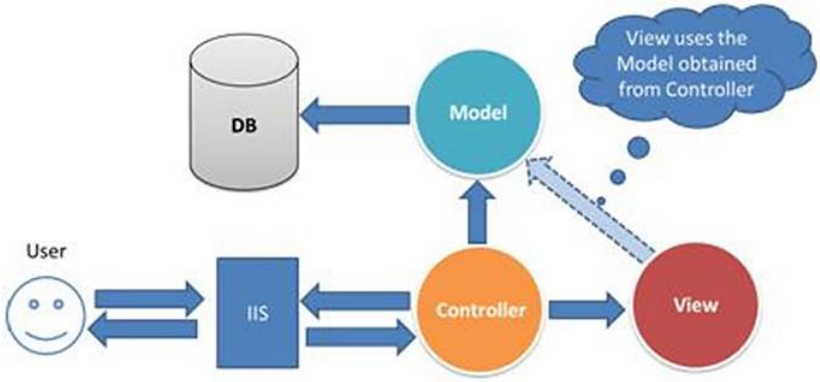


Ilustración 18: Modelo Vista Controlador ASP.NET [33]

Mediante los antes expresado y a través de la característica ASP.Net que ofrece Microsoft Visual Studio se desarrolla el MVC que consiste en la extracción de datos desde la Base de datos y cumplir con la demostración de los modelos para luego mediante este proceso generar los controladores específicos para obtener las vistas y finalmente ser mostrado al cliente, claro está que todo este proceso se realiza automáticamente, todo rige en el modelo que esté bien estructurado con todas sus dependencias y atributos, tal y como se muestra en la ilustración 18 donde los datos son vistos por el usuario a través de una IIS

(Internet Information Server) consumidos por los controladores indicados, organizados por las vistas y empleado por los modelos y base de dato respectivamente bien esquematizados.

MODELO CONCEPTUAL DE LA APP WEB

ASP.NET cuenta con diversas plantillas para hacer el modelado, en este caso para demostrar las funcionalidades de automatización que ofrece Visual Studio se utilizó el MVC con todas sus dependencias para extraer la base de datos generada y visualizar los controladores y las vistas.

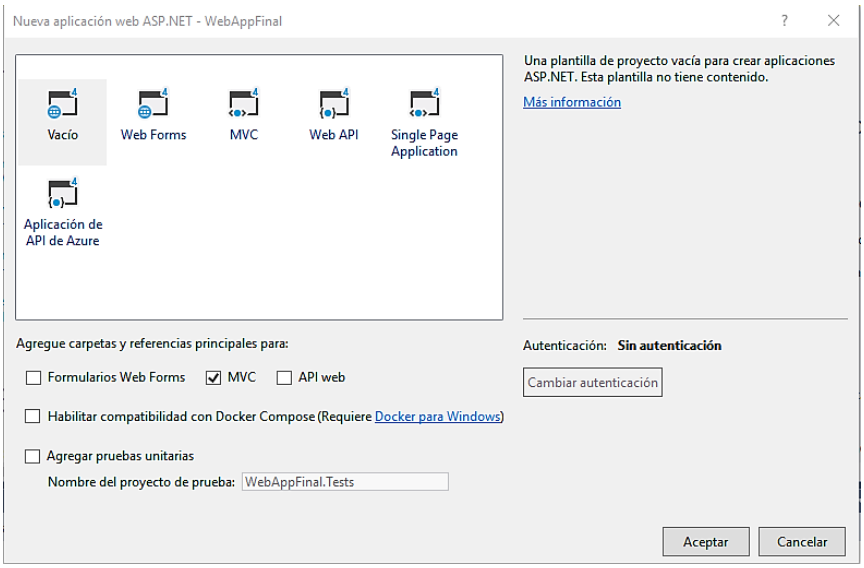


Ilustración 19: Selección de Plantilla para Modelado MVC

En el apartado de la ilustración 19 se procede a crear una plantilla de proyecto en este caso el modelo MVC agregándole pruebas unitarias para hacer validaciones del sistema.

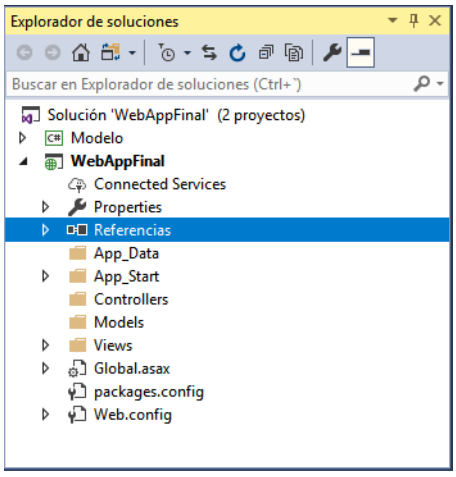


Ilustración 20: Estructura MVC

Ilustración 20 que muestra la forma MVC que genera el IDE de Microsoft Visual Studio.

En la carpeta de Models se especificará el diagrama de clases con sus respectivas relaciones, también visual studio permite obtener los modelos a partir de una base de datos bien estructurada, organizando los modelos independientes de la computación y transformándolos en modelos independientes de la plataforma.

MODELO DE DOMINIO. - Este es el modelo de alto nivel que muestra la arquitectura general del sistema, es decir, esto equivale al modelo independiente de la plataforma (PIM) sin detalle alguno para su implementación de la siguiente forma catalogado en la ilustración 21 del proceso de aplicación web.

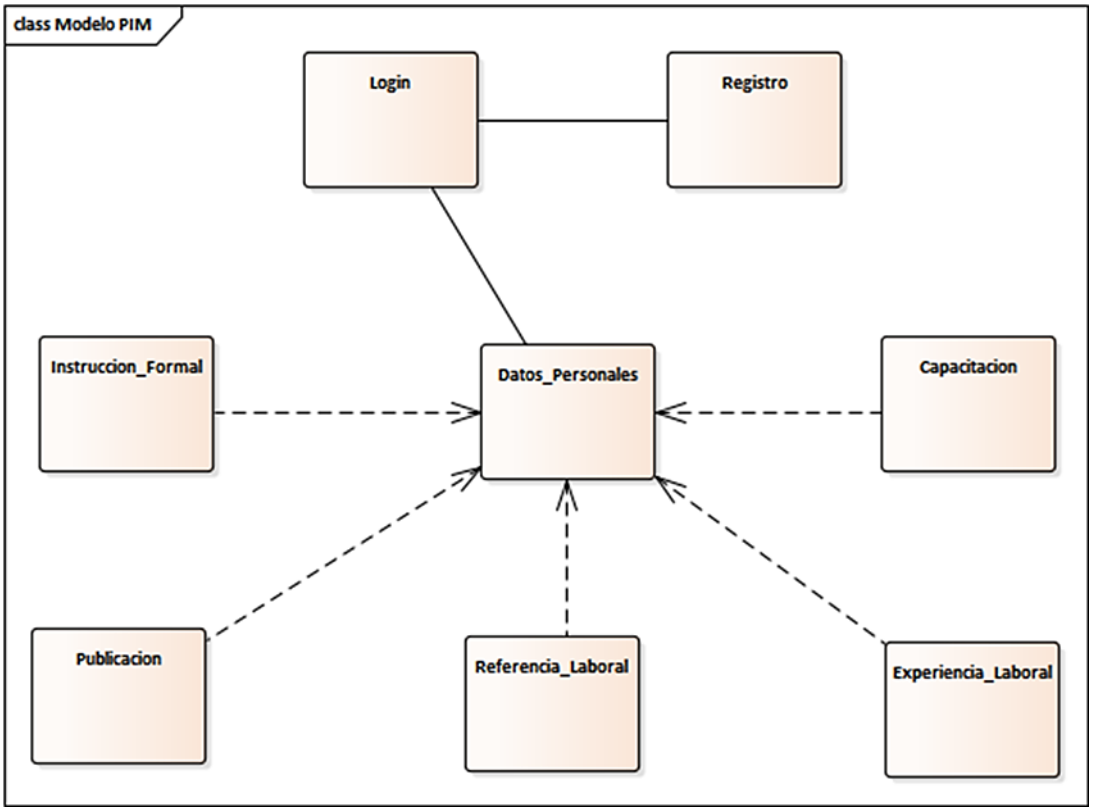


Ilustración 21: Modelo PIM

MODELO DE LA APLICACIÓN. – En esta parte se especifican los modelos específicos de la plataforma, por tal motivo a partir de los PIM se generan uno o varios PSM como ya antes fue mencionado en las características de los modelos de la arquitectura dirigida por modelo, en la ilustración 22 veremos el diagrama de clases con sus respectivas reglas, atributos, operaciones, etc.

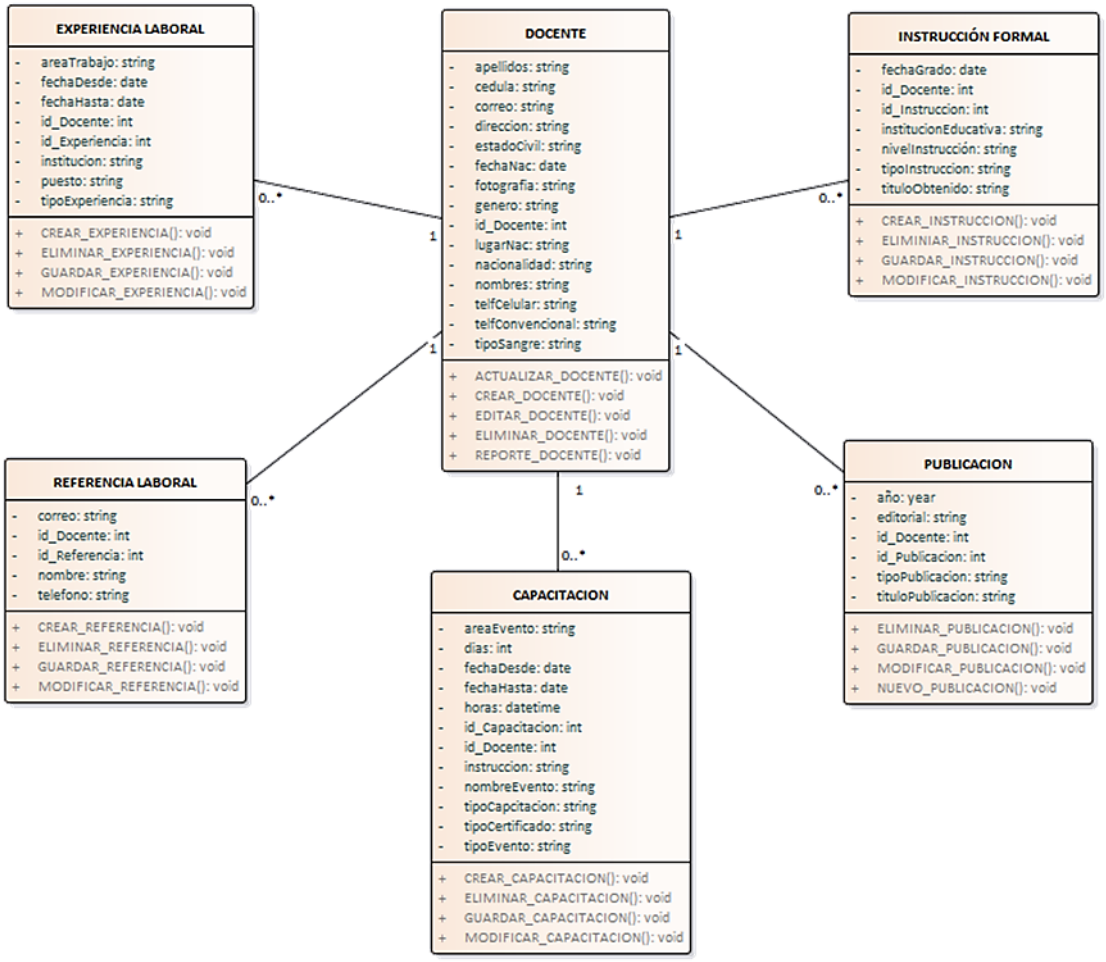


Ilustración 22: Modelo PSM

Para implementar los modelos PIM – PSM se procede a la obtención de la base de datos que está estructurada en SQL Server y así adquirir los modelos específicos UML. Tal y como se muestra en la ilustración 23 del Entorno de Desarrollo Integrado de Visual Studio.

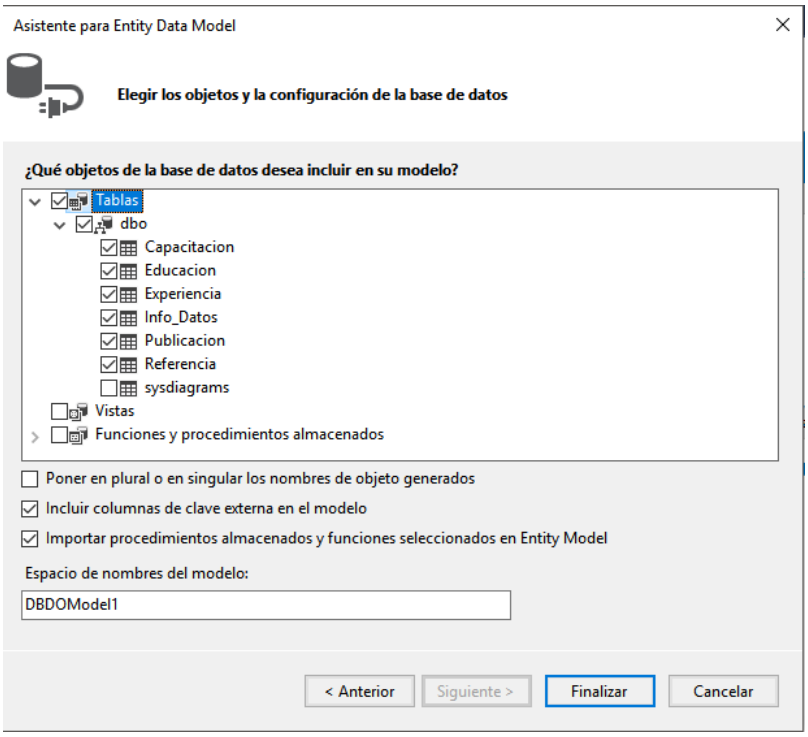


Ilustración 23: Modelos SQL Server - Diagrama de Clase UML

Posteriormente se visualizará el siguiente diagrama expuesto en la ilustración 24 del documento desarrollado.

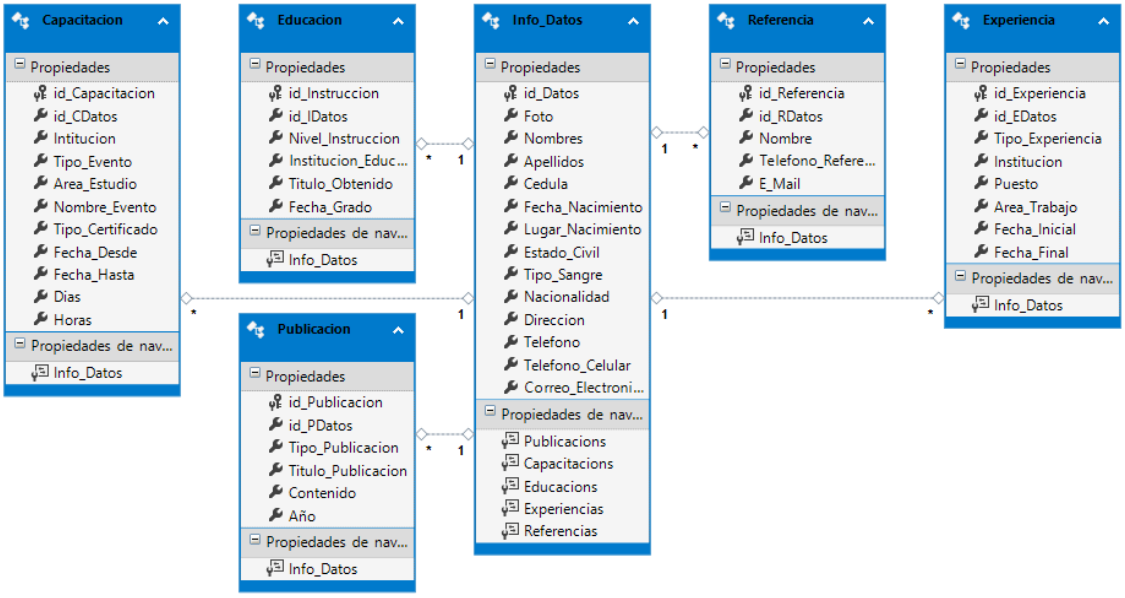


Ilustración 24: Modelo PSM - SQL Server

En consecuencia, una vez obtenidos los modelos se produce el proceso de transformación de código fuente, el cual, forma parte de los controladores del sistema que se esta desarrollando.

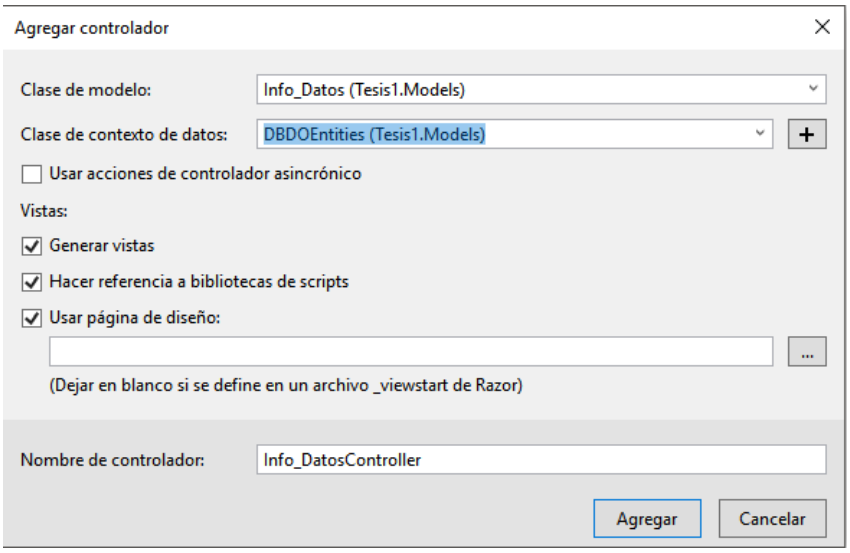


Ilustración 25: Agregar Controladores

A través de la ilustración 25 se muestra el proceso de agregación de controladores del sistema organizando las acciones y vistas de Razor para crear, leer, actualizar, eliminar y listar las entidades del contexto de datos de Entity Framework, el código fuente será mostrado en el **(ANEXO 5).**

Una vez construido el modelo de dominio, Visual Studio permite generar automáticamente los métodos en el controlador, estableciendo las validaciones necesarias para la implementación del CRUD en el sistema, la cual esta especificada en el **(ANEXO 6).**

3.4.NORMAS INTERNACIONALES DE BUENAS PRÁCTICAS EN LA PROGRAMACIÓN

La programación web también es reconocida como software de desarrollo, por ello, sus estándares se mantienen según la programación de escritorio.

Estos estándares deben estar previamente definidos, puesto que, el gran problema de los programadores es no seguir estos estándares a la hora de escribir código. Esto provoca que sea complicado entender el código a otros programadores incluso a los mismos desarrolladores de un sistema de software. Es decir, existen programadores que generan su propio estándar de desarrollo, al crear funciones de código donde ni ellos en un transcurso de tiempo podrán realizar una retroalimentación al sistema, haciendo que el trabajo se vuelva más engorroso al actualizar las versiones del software debido a que si no se entiende tocará desarrollar o codificar nuevamente todo, mientras que a partir de

los modelos dirigidos el cambio se realiza únicamente en los modelos sin afectar el código que esta generado.

Estos modelos deben estar bien estructurados, mediante la ayuda de los modelos específicos de UML se podrá empaquetar todo un sistema en diferentes bloques encapsulados que organizan todo el sistema.

Estos paquetes deben contar con normas y estándares que sean comprensibles para otros programadores, así pues, se organiza todo el trabajo y se emplea menos tiempo en la construcción del sistema.

Estas normas de programación deben ser conocidas, sin importar en que lenguaje de programación estés trabajando o desarrollando un sistema. De esta manera, en la presente investigación se desarrolló una Aplicación Web Dirigida por Modelos siguiendo los lineamientos que ofrece la metodología de Arquitectura Dirigida por Modelo.

MDA también conocido por sus siglas convierte a la programación tradicional en una programación que se basa en modelos optimizando tiempo, usabilidad y portabilidad en cada sistema.

Estos modelos en la actualidad son considerados importantes a la hora de crear una aplicación web, debido a que todo el proceso de diseño y construcción de un sistema parte de este automatizando la obtención de código factible para su funcionamiento. Es decir, los modelos deben contar con atributos, reglas, nomenclaturas, relaciones, etc., para hacer de este un sistema entendible para el cliente.

En consecuencia, para el estudio de esta aplicación se establecen las normas internacionales de buenas prácticas en la programación, en los lenguajes de (C#, .NET, Java) obteniendo la más idónea en el enrolamiento de modelos dirigidos para el desarrollo de software.

Buenas Practicas

Se establece seguir un conjunto de buenas prácticas en la programación que sean aplicables a los proyectos de desarrollo de software en el ámbito de la tecnología y la innovación.

En este contexto se describen los criterios específicos para elaborar aplicaciones en el lenguaje de programación Java, es decir, las reglas de codificación y reglas de construcción.

Reglas de codificación en java permiten estructurar un sistema de software con especificaciones, según la Junta de Extremadura [34] en el convenio de codificación en Java.

Sintaxis General	*Ejemplo*	*Descripción*
Clases e interfaces	public class Experiencia {}	Los nombres de las clases serán descritos y no muy largos, la primera letra será mayúscula y las siguiente minúsculas, usando generalmente sustantivos y adjetivos.
Métodos e identificadores	getId_RDatos(): int setId_RDatos(in value:int)	Utilizar "get" y "set" para hacer consultas o modificaciones de atributos.
Comentarios	/** * @author Mariana * @version 1.0 * @created 04-dic.-2018 12:35:48 */	Los comentarios son descritos para mejorar la compresión del código, la reutilización del código y la mejora de las aplicaciones.
Paquetes	import java.util.*;	Los paquetes estructuran el código de la aplicación.
Parámetros y variables	public static int sumar(num1, num2) {}	Las nomenclaturas de los parámetros y variables no deberán empezar con "_" o "$".

Tabla 3: Reglas de Codificación

Reglas de construcción en Java

Estas reglas consisten en características de la construcción de métodos, identificadores, clases e interfaces, estructuras de control, importación de paquetes, cadena de caracteres, LOG, etc.

Criterios específicos de C# .NET

En este apartado se describen los criterios específicos y de codificación que deben tener los programas diseñados en C# .NET, con la finalidad de promover la unificación de todos los proyectos realizados en este lenguaje, además para establecer las reglas de programación y su eficiencia en el desarrollo de aplicaciones.

A continuación se describen las normas generales que presenta la Junta de Extremadura [35] en el convenio de codificación en .NET

1. Los nombres deben ser simples, representativos e intuitivos.
2. Es preferible la legibilidad a la brevedad, por lo que se aconseja elegir nombres lo más descriptivo posible, aunque resulten largos.
3. No utilizar abreviaturas ni contracciones como parte de los nombres.
4. En nombres compuestos, no utilizaremos separación entre palabras, como el guión bajo "_".
5. Mal "Tipo_Dato".
6. Bien "TipoDato"
7. Las letras acentuadas se reemplazarán con las equivalentes no acentuadas, y se sustituirá la letra "ñ" por "ni". (Ej. Año -> Anio).
8. Los nombres irán siempre en singular.
9. Se utilizarán caracteres alfabéticos, salvo que la naturaleza del nombre se necesiten dígitos numéricos.

Criterio específico de HTML y CSS

Con respecto a este apartado las aplicaciones web utilizan etiquetados de HTML y CSS para dar forma a las presentaciones del sistema que se está efectuando o desarrollando. De esta manera se establecen todas las normas a seguir en el formato de una aplicación web.

Las reglas de estilo que deben tener las aplicaciones compuestas con código HTML y CSS la describe la Junta de Extremadura [36] del convenio de codificación en HTML y CSS donde especifica que: Se debería utilizar el protocolo HTTPS (https:) para las imágenes y otros ficheros multimedia, ficheros de hojas de estilo y ficheros de script, a menos que esos ficheros no estén disponibles sobre HTTPS. Por ejemplo, se muestra el siguiente código HTML y CSS esquematizado en la ilustración 26.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
        <link href='http://fonts.googleapis.com/
        css?family=Source+Sans+Pro:300,400,700,300italic,400italic,700italic|Source+Code+Pro:300,400,700' rel='
        stylesheet' type='text/css'>
    </head>

    <frameset cols="25%,75%">
        <frame src="contents/navigation.html" name="navigation">
        <frame src="contents/0fa3e26321c26558efa77ff144c36f21.html" name="content">
    </frameset>

    <noframes>
        <h2>Frame Alert</h2>
        This document is designed to be viewed using the frames feature. If you see this message, you are using
        a non-frame-capable web client.
        <br>
        Link to<a href="overview-summary.html">Non-frame version.</a>
    </noframes>

</html>
```

Ilustración 26: Codificación HTML y CSS

A raíz del estudio realizado se muestra un cuadro comparativo de ventajas y desventajas obtenidas de las herramientas estudiadas.

	STARUML	ENTERPRISE ARCHITECT	MICROSOFT VISUAL STUDIO
VENTAJAS	Permite el encapsulamiento de todos los modelos que se necesiten para la organización del proceso de software	Es compatible con Visual Studio y Eclipse. Fácil de comprender. La documentación del desarrollo de una aplicación es encapsulada en cada proceso para ser administrada por diferentes miembros de una organización.	Fácil de comprender. Permite la integración de github para las versiones del proyecto. Permite integrar e instalar complementos necesarios para su aplicativo web. Admite la integración de complementos de diagramas UML
DESVENTAJAS	No admite la integración de símbolos y letras con tildes.	Es difícil su comprensión	Debe estar constantemente actualizándose.

CAPÍTULO IV:
DISCUSIÓN

A lo largo de esta investigación se estudió la metodología y herramientas de la Arquitectura Dirigida por Modelos, así que, Hilario *et al.* [37] expresan que adoptaron usar la metodología Programación Extrema o Extreme Programming (XP), modelando con el Lenguaje Unificado de Modelado (UML) la aplicación, ajustando el desarrollo de los flujos de trabajo según las especificaciones de UML para la ingeniería web. Haciendo uso de la herramienta CASE (Computer Aided Software Engineering) Enterprise Architect (EA), una herramienta comprensible de diseño y análisis UML, permite cubrir el desarrollo de software desde el paso de los requerimientos a través de las etapas del análisis, modelos de diseño, pruebas y mantenimiento. El Enterprise Architect es una herramienta multi-usuario, basada en Windows y diseñada para ayudar a construir un software robusto y fácil de mantener que ofrece la salida de documentación flexible y de alta calidad.

Posterior a esto Enterprise Architect evoluciona y mantiene sus funcionalidades activas con el fin de optimizar los procesos de desarrollo mediante el uso de modelos dirigidos, implementa la metodología MDA para establecer modelos independientes de la plataforma automatizando la elaboración de aplicaciones.

Arsaute *et al.* [25] bajo la metodología MDA la funcionalidad de un sistema es definida en primer lugar como un modelo independiente de la plataforma (Platform-Independent Model, PIM). Dado un modelo de definición de la plataforma (Platform Definition Model, PDM), el modelo PIM puede traducirse entonces a uno o más modelos específicos de la plataforma (Platform-Specific Models, PSM) para la implementación correspondiente, usando diferentes lenguajes específicos del dominio, o lenguajes de propósito general como Java, C#, Python, entre otros. La traducción entre un modelo PIM y modelos PSM se realiza normalmente utilizando herramientas automatizadas, como lo son las herramientas de transformación de modelos, por ejemplo, aquellas que cumplen con el estándar OMG denominado Query/View/Transformation, (QVT). Donde Castro et al. [38] dicen que: en el caso de la Ingeniería del software se preocupa por la calidad del producto software a través de la evaluación de características tales como la funcionalidad, la Fiabilidad, la eficiencia, la mantenibilidad, la portabilidad, y de la Usabilidad. La

Interacción Hombre Máquina, en cambio, se preocupa por la calidad del proceso de interacción, verificando la accesibilidad, la experiencia de usuario y la usabilidad.

De esta manera, Microsoft Visual Studio contiene todos los elementos para generar aplicaciones web ASP.NET, servicios web XML, aplicaciones de escritorio y aplicaciones móviles, utilizando el mismo entorno de desarrollo, es decir, cuenta con todas las facilidades para desarrollar aplicaciones de calidad para el cliente.

La tecnología y herramienta empleada para el desarrollo de esta aplicación web permitió mostrar que la implementación de los modelos dirigidos facilita la conversión de PIM – PSM y de PSM a código optimizando la obtención de aplicativos más rápidos y bien estructurados.

Gallego Alzate [39] menciona que uno de los temas tratados fue la arquitectura, en la cual todos concuerdan que aunque es una arquitectura buena, rápida y funcional, se deberían mejorar aspectos como la inclusión de gestores de paquetes como npm y composer, para facilitar y optimizar mucho más el tiempo que se requiere invertir durante los ajustes manuales que se le deban realizar al código; otro aspecto a destacar aquí fue la utilización de JQuery en el lado del front-end, la cual para los expertos en web-ui, es una librería que ya va saliendo del mercado, debido a que van entrando formas más optimizadas de trabajar todo el front-end. Desde el lado del back-end, también se discutió bastante sobre el uso del framework prado, el cual para el experto en php en un futuro se podría pensar en cambiar por uno que tenga mayor versatilidad y una comunidad más activa en su desarrollo y soporte, como lo son por ejemplo laravel, symfony y yii, enfatizando que el framework prado no es que sea malo, sino que, a juicio de él, existen mejores herramientas para encarar el problema.

La Arquitectura Dirigida por Modelos facilita la manera de emplear modelos de desarrollo, la misma que permite implementar procesos que no estén orientados a ninguna plataforma general. Esta se basa en la utilización de modelos bien estructurados, involucrando a la ingeniería de software en la transformación de código fuente para su implementación.

MDA ofrece funcionalidad que hace del trabajo de desarrollo de software un método más eficiente y eficaz al momento de producir un producto debido a que emplea características específicas que permiten llevar una documentación y realizar un mantenimiento favorable a través de la reutilización de los modelos generados.

Las herramientas MDA estructuran un modelo rápido de desarrollo, que son independientes de la plataforma y pueden ser generados, utilizados en cualquier plataforma. Estas ofrecen al programador estrategias ágiles que fomentan el estudio de los modelos de dominio, proporcionando ventajas concretas en el tiempo de productividad, interoperabilidad, portabilidad, mantenimiento y documentación de los modelos mejorando el tiempo de desarrollo.

Esta metodología y sus herramientas engloban todo el proceso de desarrollo al encapsular los modelos PIM – PSM, centrándose en el problema y facilita el desarrollo de aplicaciones en menos tiempo y con mayor confiabilidad de los sistemas.

Esta tecnología favorece a los desarrolladores y a todos los miembros de una empresa porque facilita la comunicación de todo el proceso y puede ser actualizado y mejorado por todos los integrantes de desarrollo del sistema.

CAPÍTULO V:
CONCLUSIONES

- A través del análisis de la metodología MDA se logró comprender los procesos de desarrollo de aplicaciones web mediante modelos dirigidos.
- Se utilizaron las herramientas de Arquitectura Dirigida por Modelos para especificar los requerimientos o los modelos conceptuales de la aplicación a desarrollar.
- Se implementaron los modelos dirigidos en el desarrollo de la aplicación web obteniendo como resultado la automatización de código fuente y la reducción de costo y tiempo en la producción de la aplicación.
- Al implementar los procesos de transformación en las herramientas de la Arquitectura Dirigida por modelo se constató que: algunas solo permiten encapsular todo el trabajo de desarrollo mientras que Visual Studio es fácil de comprender y organiza todo el software hasta su lanzamiento.
- A partir de los modelos dirigidos implementados en la generación de una aplicación web se comprobó la factibilidad de uso, factibilidad económica al ahorrar tiempo y costo de producción y factibilidad técnica al permitir diseñar, desarrollar y brindar mantenimiento constante a la aplicación web de forma independiente de la plataforma.

CAPÍTULO VI:
RECOMENDACIONES

- Se recomienda establecer nuevas estrategias de aprendizaje de los procesos de documentación mediante los modelos dirigidos.
- También es recomendable estudiar las metodologías que en la actualidad forman parte del proceso de desarrollo de aplicaciones, analizando características y funcionalidades que brinden un desarrollo completo de aplicaciones web mediante la transformación de los modelos.
- Es recomendable que los programadores implementen metodologías ágiles en el desarrollo de aplicaciones web, debido a que la tecnología avanza e incrementa el cambio evolutivo en los procesos de documentación.

ANEXOS

Anexo 1: Código Generado a partir de Modelo de Clases UML (C#, Java y Django)

```csharp
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

public class Info_Datos {
    public Info_Datos() {}

    protected int id_Datos;
    private String Foto;
    private String Nombres;
    private String Apellidos;
    private String Cedula;
    private date Fecha_Nacimiento;
    private String Nacionalidad;
    private String Lugar_Nacimiento;
    private String Estado_Civil;
    private String Tipo_Sangre;
    private String Direccion;
    private String Telefono_Convencional;
    private String Telefono_Celular;
    private String Correo_Electronico;

    /// <summary>/// @return  /// </summary>
    public Void Crear_Datos() {
        return null;}

    /// <summary>/// @return/// </summary>
    public Void Modificar_Datos() {
        return null;}

    /// <summary>/// @return /// </summary>
    public Void Eliminar_Datos() {
        return null;}
}
```

```java
import java.util.*;

public class Referencia {

    /**
     * Default constructor
     */
    public Referencia() {}

    protected int id_Referencia;
    protected int id_RDatos;
    private String Nombre;
    private String Telefono;
    private String Correo_Electronico;

    /**
     * @return
     */
    public Void Crear_Referencia() {
        return null;}

    /**
     * @return
     */
    public Void Modificar_Referencia() {
        return null;}

    /**
     * @return
     */
    public Void Eliminar_Referencia() {
        return null; }
}
```

```python
#-*- coding: utf-8 -*-

from django.db import models

class Referencia(models.Model):
    class Meta:
        pass

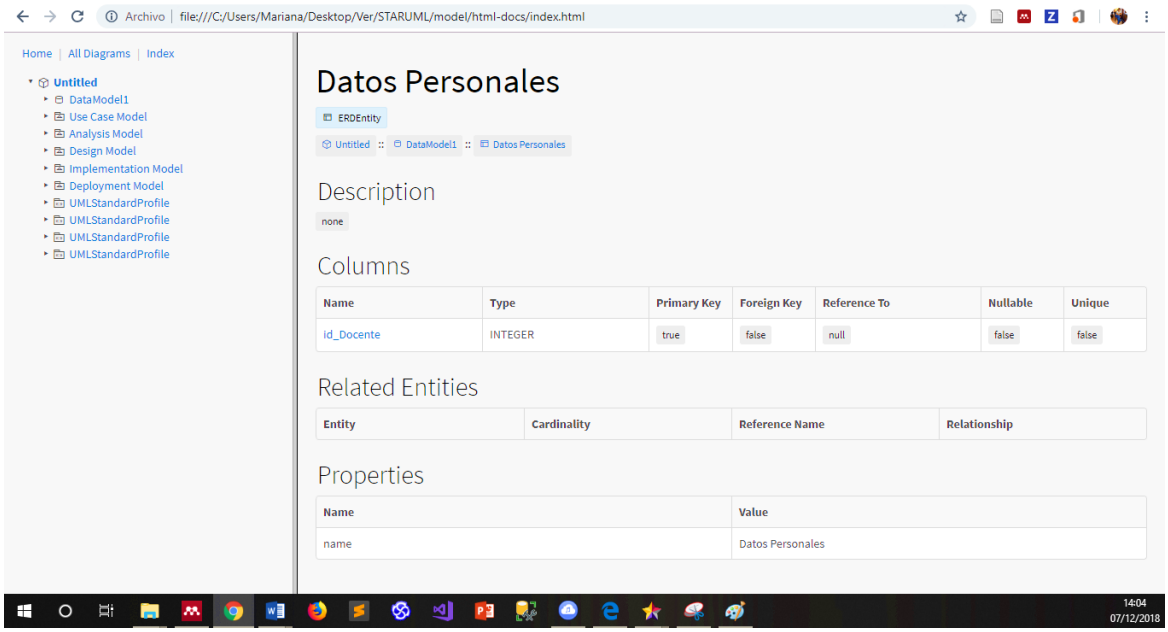
    id_Referencia = undefined()
    id_RDatos = undefined()
    Nombre = undefined()
    Telefono = undefined()
    Correo_Electronico = undefined()


    def Crear_Referencia(self, ):
        pass

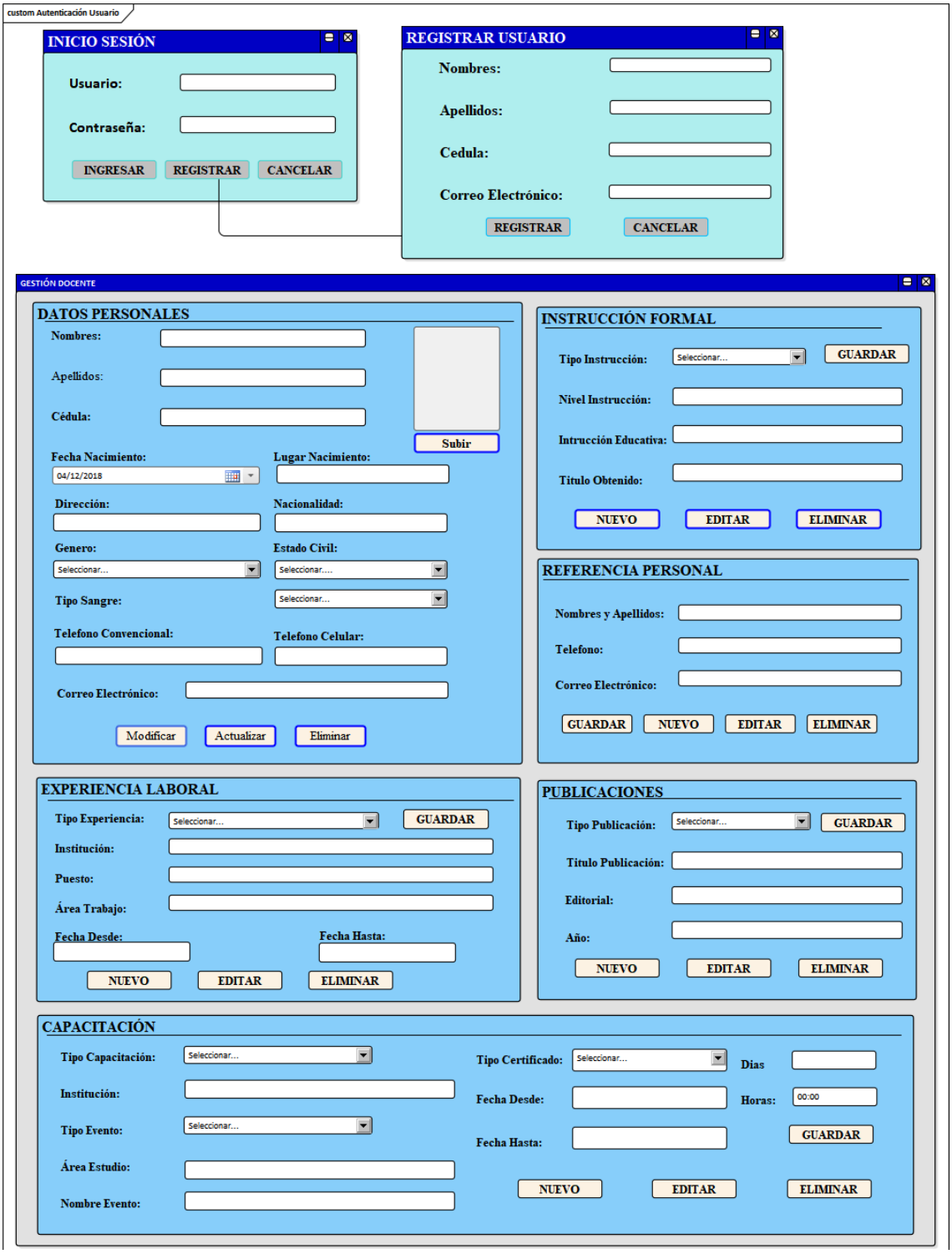
    def Modificar_Referencia(self, ):
        pass

    def Eliminar_Referencia(self, ):
        pass
```

Anexo 2: Documentación HTML del Aplicativo Web



Anexo 3: Interfaz Gráfica en EA de aplicativo



Anexo 4: Código Fuente EA (C#, Java y PHP)

```csharp
///////////////////////////////////////////////////////////
//  Educacion.cs
//  Implementation of the Class Educacion
//  Generated by Enterprise Architect
//  Created on:      04-dic.-2018 12:42:05
//  Original author: Mariana
///////////////////////////////////////////////////////////

using System;
using System.Collections.Generic;
using System.Text;
using System.IO;

public class Educacion {

    private date FechaGrado;
    private int id_IDatos;
    private int id_Instruccion;
    private string InstitucionEducativa;
    private string NivelInstruccion;
    private string TituloObtenido;

    public Educacion(){}

    ~Educacion(){}

    public void Actualizar(){}

    public void Crear(){}

    public void Eliminar(){}

    public void Leer(){}

}//end Educacion
```

```java
/**
 * @author Mariana
 * @version 1.0
 * @created 04-dic.-2018 12:35:42
 */
public class Educacion {

    private date FechaGrado;
    private int id_IDatos;
    private int id_Instruccion;
    private string InstitucionEducativa;
    private string NivelInstruccion;
    private string TituloObtenido;

    public Educacion(){}

    public void finalize() throws Throwable {}
    public void Actualizar(){}

    public void Crear(){}

    public void Eliminar(){}

    public void Leer(){}
}//end Educacion
```

```php
<?php
/**
 * @author Mariana
 * @version 1.0
 * @created 04-dic.-2018 12:49:51
 */
class Educacion
{

    private $FechaGrado;
    private $id_IDatos;
    private $id_Instruccion;
    private $InstitucionEducativa;
    private $NivelInstruccion;
    private $TituloObtenido;

    function __construct(){}

    function __destruct(){}

    public function Actualizar(){}

    public function Crear(){}

    public function Eliminar(){}

    public function Leer(){}
}
?>
```

Anexo 5: Transformación MDA (Código Fuente)

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Net;
using System.Web;
using System.Web.Mvc;
using Tesis1.Models;

namespace Tesis1.Controllers
{
    public class Info_DatosController : Controller
    {
        private DBDOEntities db = new DBDOEntities();

        // GET: Info_Datos
        public ActionResult Index()
        {
            return View(db.Info_Datos.ToList());
        }

        // GET: Info_Datos/Details/5
        public ActionResult Details(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Info_Datos info_Datos = db.Info_Datos.Find(id);
            if (info_Datos == null)
            {
                return HttpNotFound();
            }
            return View(info_Datos);
        }

        // GET: Info_Datos/Create
        public ActionResult Create()
        {
            return View();
        }

        // POST: Info_Datos/Create
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Create([Bind(Include =
"id_Datos,Foto,Nombres,Apellidos,Cedula,Fecha_Nacimiento,Lugar_Nacimiento,Estado_Ci
vil,Tipo_Sangre,Nacionalidad,Direccion,Telefono,Telefono_Celular,Correo_Electronico
")] Info_Datos info_Datos)
        {
            if (ModelState.IsValid)
            {
                db.Info_Datos.Add(info_Datos);
                db.SaveChanges();
                return RedirectToAction("Index");
            }

            return View(info_Datos);
        }
        // GET: Info_Datos/Edit/5
        public ActionResult Edit(int? id)
        {
```

```
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Info_Datos info_Datos = db.Info_Datos.Find(id);
            if (info_Datos == null)
            {
                return HttpNotFound();
            }
            return View(info_Datos);
        }

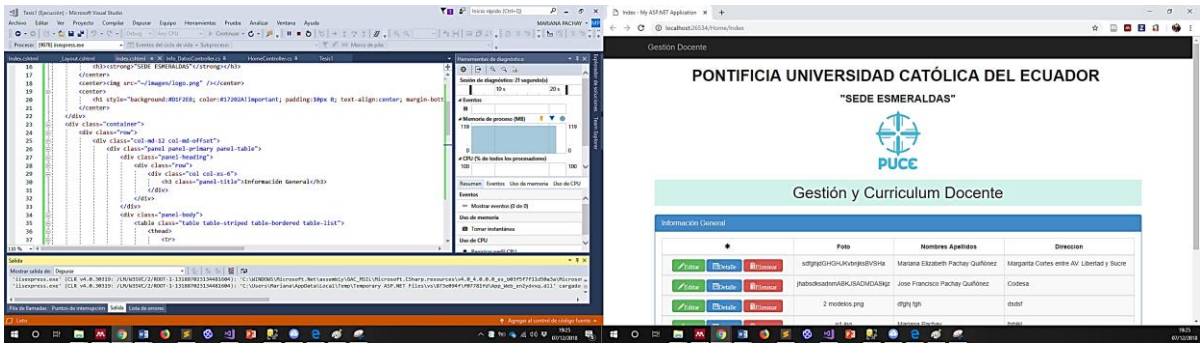
        // POST: Info_Datos/Edit/5
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Edit([Bind(Include =
"id_Datos,Foto,Nombres,Apellidos,Cedula,Fecha_Nacimiento,Lugar_Nacimiento,Estado_Ci
vil,Tipo_Sangre,Nacionalidad,Direccion,Telefono,Telefono_Celular,Correo_Electronico
")] Info_Datos info_Datos)
        {
            if (ModelState.IsValid)
            {
                db.Entry(info_Datos).State = EntityState.Modified;
                db.SaveChanges();
                return RedirectToAction("Index");
            }
            return View(info_Datos);
        }

        // GET: Info_Datos/Delete/5
        public ActionResult Delete(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Info_Datos info_Datos = db.Info_Datos.Find(id);
            if (info_Datos == null)
            {
                return HttpNotFound();
            }
            return View(info_Datos);
        }

        // POST: Info_Datos/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteConfirmed(int id)
        {
            Info_Datos info_Datos = db.Info_Datos.Find(id);
            db.Info_Datos.Remove(info_Datos);
            db.SaveChanges();
            return RedirectToAction("Index");
        }

        protected override void Dispose(bool disposing)
        {
            if (disposing)
            {
                db.Dispose();
            }
            base.Dispose(disposing);
        }
    }
}
```

Anexo 6: Aplicación Web Dirigida por Modelos



REFERENCIAS

[1] I. Rueda Cáceres, “Generación Automática De Aplicaciones Software a Partir Del Estandar Mda Basándose En La Metodologia De Sistemas Expertos E Inteligencia Artificial,” *Rev. Gti*, vol. 10, pp. 67–72, 2011.

[2] C. Enrique and et al., “Aplicación De Ingeniería Dirigida Por Modelos (Mda), Para La Construcción De Una Herramienta De Modelado De Dominio Específico (Dsm) Y La Creación De Módulos En Sistemas De Gestión De Aprendizaje (Lms) Independientes De La Plataforma Application of Model-,” vol. 169, pp. 43–52, 2011.

[3] K. Leyva Vallejo and et al., “Exploración del diseño y arquitectura web. Aplicación a páginas electrónicas del sector bancario desde la perspectiva del usuario,” *Adm. Negocios*, no. 80, pp. 41–58, 2016.

[4] J. R. Molina *et al.*, “COMPARACIÓN DE METODOLOGÍAS EN APLICACIONES WEB COMPARISON OF METHODOLOGIES IN WEB APPLICATIONS Citación sugerida,” 2018.

[5] A. Aquino Caje, “Evolución de la Web,” pp. 1–17, 2016.

[6] L. Mariano, “Historia De Las Web ,” *Univ. Marcelino Champagnat*, pp. 1–8, 2018.

[7] C. A. Pereira Marín, “Marco de Integración de Paradigmas Tecnológicos hacia una Web Semántica Distribuida basada en la Gestión de Procesos (Integration Framework of Technological Paradigms Towards a Distributed Semantic Web based on Process Management),” *GECONTEC Rev. Int. Gestión del Conoc. y la Tecnol.*, vol. 6, no. 1, pp. 82–100, May 2018.

[8] M. Rojas Rodriguez and J. Acanda Velázquez, “Tendencias actuales de programas de proyectos Current Trends in Project Programs,” vol. 10, no. 10, pp. 1–15, 2017.

[9] E. P. Rueda Tapiero and et al., “Aplicativo web de gestión de información para el semillero ‘Kerverux,’” 2018.

[10] I. Rodríguez-Ramírez, “Incorporación del tema de usabilidad en el diseño de sitios web en el curso de Multimedios,” *Rev. Educ.*, vol. 39, no. 2, p. 27, Jun. 2015.

[11] A. Cabrera Mondeja and O. Gómez Baryolo, “ESTADO DEL ARTE EN EL DESARROLLO DE APLICACIONES WEB BASADO EN INGENIERÍA

DIRIGIDA POR MODELOS. STATE OF THE ART IN THE DEVELOPMENT OF WEB APPLICATIONS BASED ON MODEL DRIVEN ENGINEERING," *Rev. Científica ECOCIENCIA ESTADO*, vol. 4, no. 1, pp. 1390–9320, 2017.

[12] C. Alonso-Torres, "Framework basado en ontología para la descripción y validación de procesos de negocio," *Ingeniería Industrial*, vol. 35, no. 3, Instituto Superior Politicnico Jose Antonio Echeverria (Cujae), pp. 276–288, 2017.

[13] D. Toledo Enriquez and et al., *Framework basado en MDA y ontologías para la representación y validación de modelos de componentes*, vol. 8, no. 2. Univ. de las Ciencias Informaticas (UCI), 2015.

[14] G. Nuñez, "Un Enfoque MDD para el desarrollo de RIA," *Univ. CATÓLICA "NUESTRA SEÑORA LA ASUNCIÓN" Fac. CIENCIAS Y Tecnol.*, 2016.

[15] L. Muñoz, G. Solarte, and D. Heredia Hernández, "Una visión a la propuesta de Arquitectura Dirigida por Modelos * A vision to the proposal of Model Driven Architecture Uma visão a proposta de Model Driven Architecture," *Unimar*, vol. 33, no. 2, pp. 270–279, 2015.

[16] S. Martínez, L. Lamonth, and N. Jacho, "Análisis de la Transformación de Modelo CIM a PIM en el Marco de Desarrollo de la Arquitectura Dirigida por Modelos (MDA)," *Rev. Politécnica -Septiembre*, vol. 36, no. 3, 2015.

[17] L. V. Venegas Loor, "Arquitectura Manejada por Modelos," *Univ. Part. San Gregor. Portoviejo*, vol. 8, p. 5, 2014.

[18] J. Texier, M. R. De Giusti, and S. E. Gordillo, "El desarrollo de software dirigido por modelos en los repositorios institucionales," *Dyna*, vol. Edition 18, no. 184, pp. 186–192, 2014.

[19] I. García Portuondo and L. Fonseca Suárez, "Desarrollo Rápido de Aplicaciones Apoyado en MDA," *Conv. Cient. Ing. y Aqrquitectura*, p. 5, 2010.

[20] C. A. Gaitán Peña, "Líneas de Productos Software: Generando Código a Partir de Modelos y Patrones," *Sci. Tech.*, vol. 22, no. 2, p. 178, Jun. 2017.

[21] J. Sayago Heredia, "Generador de Código Utilizando el Paradigma de Líneas de Producto Software," *Rev. Científica Hallazgos21*, vol. 3, no. 2, pp. 190–212, Jul. 2018.

[22] E. Estévez and et al., “Aproximación Basada en UML para el Diseño y Codificación Automática de Plataformas Robóticas Manipuladoras,” *RIAI - Rev. Iberoam. Autom. e Inform. Ind.*, vol. 14, no. 1, pp. 82–93, 2017.

[23] A. Bernal, M. E. Cambronero, V. Valero, and A. Núnez, “Modelado, Transformación y Validación de Sistemas Cloud,” 2018.

[24] J. C. Moreno and et al., *Evaluación Temprana de la Usabilidad Empleando Patrones Embebidos en la Construcción del Modelo Conceptual para Aplicaciones Web*. 2018.

[25] A. Arsaute and et al., “Generación automática de API REST a partir de API Java, basada en transformación de Modelos (MDD),” 2018.

[26] L. A. Jumbo Flores and et al., “Desarrollo de Aplicación Web para la Gestión de Producción de Camarón Development of application for the management of Shrimp Production,” *Rev. Espac.*, vol. 39, no. Nª 4, p. 28, 2018.

[27] R. Monferrer and A. Resumen, “Especificación de Requisitos Software según el estándar de IEEE 830,” 2000.

[28] B. L. Flores-rios and F. J. Pino, “Elicitación de requisitos no funcionales basada en la gestión de conocimiento de los stakeholders Non-functional requirements elicitation based on stakeholders ’ s knowledge management,” vol. 26, no. 1, pp. 142–156, 2018.

[29] E. N. Nicomedes Teodoro, “Tipos de investigación,” 2018. [Online]. Available: http://repositorio.unisdg.edu.pe/bitstream/USDG/34/1/Tipos-de-Investigacion.pdf. [Accessed: 20-Oct-2018].

[30] “StarUML,” *Lanzamiento Estable (v3.0.2 / 17 de agosto de 2018)*, 2018. [Online]. Available: http://staruml.io/. [Accessed: 20-Oct-2018].

[31] “Herramientas de modelado UML para empresas, software, sistemas y arquitectura.,” *Lanzamiento Estable (14.1 Construir 1427)*, 2017. [Online]. Available: https://sparxsystems.com/. [Accessed: 20-Oct-2018].

[32] Microsoft, “Novedades de Visual Studio 2017 | Noticias y actualizaciones de productos - Visual Studio,” *2018*, 2018. [Online]. Available: https://visualstudio.microsoft.com/es/vs/whatsnew/. [Accessed: 20-Oct-2018].

[33] M. Sukesh, “WebForms vs. MVC - CodeProject,” 2014. [Online]. Available: https://www.codeproject.com/Articles/528117/WebForms-vs-MVC. [Accessed: 17-Nov-2018].

[34] “Convenio de codificación en JAVA,” 2018.

[35] “Convenio de codificación en .NET,” 2018.

[36] “Convenio de codificación en HTML y CSS,” 2018.

[37] J. Hilario and et al., “Sistema de Información Clínico Hospitalaria. Arquitectura y Mapa de camas Hospital Clinical Information System: architecture and the map of beds,” *Rev. Ciencias Médicas Pinar del Río*, vol. 22, no. 1, pp. 92–100, 2018.

[38] M. Castro *et al.*, “Desarrollo e Implementación de un Modelo Basado en Estándares para el Logro y Evaluación de la Usabilidad en Aplicaciones Informáticas,” 2017.

[39] D. A. Gallego Alzate, “Desarrollo del módulo de orientación académica , para el software Open Curricular Management System -OCMS .,” *UNIR*, Feb. 2018.