Implementación de una aplicación web utilizando herramientas MDD y MDA, basada en principios de ingeniería web, desarrollando UX/UI en el lenguaje IFML

Alquinga Gualli, Wilmer Stalin y Pumisacho Ushiña, Lizbeth Abigail

Departamento de Ciencias de la Computación

Carrera de Tecnologías de la Información

Trabajo de titulación, previo a la obtención del título de Ingeniero/a en Tecnologías de la Información

Ing. Galarraga Hurtado, Juan Fernando MSc.

15 de septiembre del 2023

Copyleaks

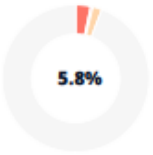
Plagiarism report

Tesis_Alquinga_Pumisacho.pdf

Scan details

Scan time:	Total Pages:	Total Words:
August 23th, 2023 at 23:51 UTC	80	19866

Plagiarism Detection



Types of plagiarism		Words
Identical	3.3%	654
Minor Changes	0.5%	94
Paraphrased	2%	406
Omitted Words	0%	0

AI Content Detection

Text coverage
- AI text
- Human text

Plagiarism Results: (93)

tcad21.pdf — 0.5%

http://www.wingtecher.com/themes/wingtecherresearch/ass...

1 MDD: A Unified Model-driven Design Framework for Embedded Control Software Zhuo Su, Dongyan Wang, Yixiao Yang , Zehong Yu, Wanli Chang...

Un enfoque MDD para el desarrollo de aplicaciones ... — 0.5%

http://www.dei.uc.edu.py/proyectos/mddplus/wp-content/up...

Lic. Manuel Núñez

Un enfoque MDD para el desarrollo de aplicaciones móviles nativas enfocadas en la capa de datos Proyecto Final de Carrera Lic. Manuel Nú...

Modelo de trabalhos — 0.4%

https://comum.rcaap.pt/bitstream/10400.26/43436/1/m%c3...

jhenriques@iscac.pt

Medicamentos Anticancerígenos (SiReMA) modelos de Machine Learning no Sistema de Recomendação de Desenvolvimento de componentes de exec...

JUAN FERNANDO GALARRAGA HURTADO

Firmado digitalmente por JUAN FERNANDO GALARRAGA HURTADO
Fecha: 2023.08.23 20:31:35 -05'00'

Departamento de Ciencias de la Computación

Carrera de Tecnologías de la Información

Certificación

Certifico que el trabajo de titulación: **"Implementación de una aplicación web utilizando herramientas MDD y MDA, basada en principios de ingeniería web, desarrollando UX/UI en el lenguaje IFML"** fue realizado por los señores **Alquinga Gualli, Wilmer Stalin y Pumisacho Ushiña, Lizbeth Abigail,** el mismo que cumple con los requisitos legales, teóricos, científicos, técnicos y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, además fue revisado y analizado en su totalidad por la herramienta de prevención y/o verificación de similitud de contenidos; razón por la cual me permito acreditar y autorizar para que se lo sustente públicamente.

Sangolquí, 21 de septiembre del 2023

Firmado electrónicamente por:
JUAN FERNANDO GALARRAGA HURTADO

Ing. Galarraga Hurtado, Juan Fernando MSc.

C.C: 1711464816

Departamento de Ciencias de la Computación

Carrera de Tecnologías de la Información

Responsabilidad de Autoría

Nosotros, **Alquinga Gualli, Wilmer Stalin** y **Pumisacho Ushiña, Lizbeth Abigail,** con cédulas de ciudadanía N° 1726705088 y N° 1751133552, declaramos que el contenido, ideas y criterios del trabajo de titulación: **"Implementación de una aplicación web utilizando herramientas MDD y MDA, basada en principios de ingeniería web, desarrollando UX/UI en el lenguaje IFML"** es de nuestra autoría y responsabilidad, cumpliendo con los requisitos legales, teóricos, científicos, técnicos, y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, respetando los derechos intelectuales de terceros y referenciando las citas bibliográficas.

Sangolquí, 21 de septiembre del 2023

Alquinga Gualli, Wilmer Stalin
C.C.: 1726705088

Pumisacho Ushiña, Lizbeth Abigail
C.C.: 1751133552

Departamento de Ciencias de la Computación

Carrera de Tecnologías de la Información

Autorización de Publicación

Nosotros **Alquinga Gualli, Wilmer Stalin y Pumisacho Ushiña, Lizbeth Abigail,** con cédulas de ciudadanía N° 1726705088 y N° 1751133552, autorizamos a la Universidad de las Fuerzas Armadas ESPE publicar el trabajo de titulación: **"Implementación de una aplicación web utilizando herramientas MDD y MDA, basada en principios de ingeniería web, desarrollando UX/UI en el lenguaje IFML"** en el Repositorio Institucional, cuyo contenido, ideas y criterios son de nuestra responsabilidad.

Sangolquí, 21 de septiembre del 2023

.....

Alquinga Gualli, Wilmer Stalin

C.C.: 1726705088

.....

Pumisacho Ushiña, Lizbeth Abigail

C.C.: 1751133552

Dedicatoria

Dedicamos este trabajo de titulación a nuestros padres quienes han sido un apoyo incondicional en el transcurso de estos años, por habernos enseñado la importancia de la perseverancia y la dedicación, por estar con nosotros en cada paso de este proceso ayudándonos a mantenernos enfocados y avanzar hacia la meta.

A nuestro director de trabajo de titulación por su apoyo, dedicación y orientación, al igual que a nuestros docentes que nos brindaron su apoyo, sabiduría y experiencia durante nuestra formación académica.

A nuestros amigos y seres queridos por su paciencia, motivación y comprensión durante esta etapa, dándonos consejos y deseándonos siempre lo mejor.

Alquinga Gualli, Wilmer Stalin & Pumisacho Ushiña, Lizbeth Abigail

Agradecimientos

Queremos expresar nuestro más profundo agradecimiento a todas las personas que hicieron este momento posible.

A nuestros padres por ser nuestra mayor motivación y por sus palabras de aliento en todo momento, por ser los pilares tanto en nuestra formación académica como de la personal, por estar presentes en cada paso de este proceso y mantenernos enfocados en nuestras metas y objetivos.

A nuestro director de trabajo de titulación por ser una guía en la realización de este proyecto, por su comprensión, paciencia y dedicación.

Finalmente, agradecemos a la Universidad de las Fuerzas Armadas ESPE, a nuestros docentes quienes nos brindaron las herramientas necesarias para poder realizar este trabajo.

Alquinga Gualli, Wilmer Stalin & Pumisacho Ushiña, Lizbeth Abigail

Índice de Contenido

Índice de Tablas

Índice de Figuras

Resumen

Durante los últimos años la creación de aplicaciones web se ha incrementado debido a su aplicabilidad en diferentes áreas donde puede mejorar la organización, productividad y costos. Pero una de las principales limitantes que tiene la implementación de una aplicación web es el tiempo de desarrollo que requiere y por ende el costo de esa aplicación, además de los costos de mantenimiento que se pueden requerir en el futuro. Debido a este incremento, se han tratado de crear diferentes formas de optimizar el proceso de desarrollo de estas aplicaciones mediante distintas soluciones basadas en la ingeniería web, una de las organizaciones que ha registrado varios estándares con este objetivo es la Object Managment Group OMG y dos de sus principales enfoques son el Desarrollo Basado en Modelos MDD y la Arquitectura Basada en Modelos MDA, a más de estas opciones también existen lenguajes como IFML que basan su desarrollo en el modelamiento de interfaces y el flujo final de la aplicación. En el presente trabajo se ha generado una aplicación web basada en estas tres propuestas, esto implica que no se ha codificado la funcionalidad de ninguna de las tres aplicaciones generadas, únicamente se han creado los modelos necesarios para que las herramientas correspondientes generen el código de la aplicación web. Además, al igual que una aplicación web creada de manera tradicional cada una de las aplicaciones puede ser desplegada y puesta en producción con relativa facilidad dependiendo del lenguaje de programación y la base de datos utilizada. Los resultados demuestran que las aplicaciones generadas tienen un buen rendimiento y cumplen con las funcionalidades para cada una de las entidades dentro del modelo utilizado.

Palabras clave: Aplicación web, ingeniería web, desarrollo basado en modelos, arquitectura basada en modelos, lenguaje de modelado de flujo de interacción.

Abstract

During the last few years, the creation of web applications has increased due to their applicability in different areas where they can improve organization, productivity and costs. But one of the main limitations in the implementation of a web application is the development time required and therefore the cost of that application, in addition to the maintenance costs that may be required in the future. One of the organizations that has registered several standards with this objective is the Object Management Group OMG and two of its main approaches are Model Driven Development MDD and Model Driven Architecture MDA, in addition to these options there are also languages such as IFML that base their development on the modeling of interfaces and the final flow of the application. In the present work a web application has been generated based on these three proposals, this implies that the functionality of any of the three generated applications has not been coded, only the necessary models have been created for the corresponding tools to generate the code of the web application. Furthermore, just like a traditionally created web application, each of the applications can be deployed and put into production with relative ease depending on the programming language and database used. The results show that the generated applications have a good performance and comply with the functionalities for each of the entities within the model used.

Keywords: Web application, web engineering, model driven development., model driven architecture., interaction flow modeling language.

Capítulo 1: Introducción

Antecedentes

En los últimos años la demanda de aplicaciones web ha incrementado debido a su necesidad de aplicación en diferentes áreas, sus beneficios en el incremento de la productividad, mejora en la organización, optimización de procesos, etc. A la par de este incremento en la demanda también ha incrementado la dificultad del desarrollo de estas aplicaciones, dicho inconveniente se produce principalmente por la implementación de sistemas que puedan cubrir necesidades cada vez más complejas y además por la necesidad de adaptar nuevas funcionalidades a un sistema ya existente (Da Silva et al., 2020).

Para disminuir los inconvenientes que ocasionan las complejidades al momento de crear una aplicación web, los equipos de desarrollo han tratado de implementar diferentes actividades, metodologías y procesos provenientes de la ingeniería de software o de la ingeniería web, dependiendo del tipo de aplicación que se requiera (Pinzón, 2017). Uno de los primeros pasos en la mejora del desarrollo de aplicaciones con ayuda de estos procesos fue el uso del Lenguaje Unificado de Modelado (UML), específicamente para el desarrollo de aplicaciones web existe la Ingeniería Web basada en UML (UWE), lo cual da un primer acercamiento al uso de modelos para la interpretación de funcionalidades y flujo que debe tener una aplicación, todo esto previo a su elaboración (Nieves-Guerrero et al., 2014).

Pero a pesar de que la aplicación de modelos previo al desarrollo de las aplicaciones ayuda a cumplir con los requerimientos y a disminuir la cantidad de correcciones o cambios que se deben hacer, sigue siendo muy complicado tomar nuevos requerimientos e incluirlos dentro de una aplicación que ya se ha finalizado, incluso solo una modificación en el flujo de un proceso puede ocasionar varios inconvenientes y por ende representar una gran inversión de tiempo, esfuerzo y recursos (Da Silva et al., 2020). Es por esto que en los últimos años las

empresas han empezado a apostar por herramientas con un contenido bajo en código. Este tipo de herramientas usan un nivel de abstracción alto, gracias al uso de lenguajes de programación declarativos, basados en modelos o basados en meta-data (Tatnall, 2020).

Para poder sobrellevar la creación de sistemas que cuentan con una gran complejidad se han propuesto modelos que permiten la visualización del sistema con el fin de identificar los problemas existentes y posibles soluciones, los modelos a lo largo de la historia han sido fundamentales en las diferentes ingenierías, ayudando a obtener resultados más predecibles, seguros y de mayor calidad. En cada ingeniería existen diversos modelos, en distintos lenguajes y niveles de abstracción. La ingeniería de software no es indiferente a esto ya que utiliza diferentes lenguajes y notaciones que permiten la abstracción de las plataformas y tecnologías subyacentes con el fin de contar con una interpretación más conceptual del tema. Cada modelo cuenta con un diferente objetivo y pueden ser usados en diferentes etapas de desarrollo (Amado, 2021), esto ayuda en gran medida al desarrollo de las aplicaciones, ya que, los modelos no sirven únicamente como guías o interpretación del flujo que debe tener la aplicación, sino que con las herramientas que utilicen estos lenguajes se puede crear directamente a partir de los modelos, las funcionalidades, reglas de negocio e incluso interfaces de usuario (Tatnall, 2020). Entre los modelos que más se han abierto campo para su uso en diversas tecnologías se encuentran el Desarrollo Basado en Modelos (MDD) y la Arquitectura Basada en Modelos (MDA).

Específicamente, MDD se centra en permitir que el desarrollo de aplicaciones pueda llevarse a cabo mediante la creación de modelos, los cuales deben ir desde los más generales hasta los más específicos y detallados, con esto MDD permite llegar desde los modelos creados hasta el código fuente de una aplicación que cumple con los requerimientos y flujo que se ha seguido para la generación de dichos modelos. Por ende, para poder cumplir con el objetivo de MDD se debe hacer uso de herramientas que permitan tanto la creación de todos

los modelos necesarios así como su transformación al código fuente que debe tener la aplicación (Wijekoon & Merunka, 2022).

Por otra parte, MDA es una iniciativa de la Object Management Group (OMG) que se enfoca en el diseño, desarrollo e implementación de software estableciendo pautas para crear especificaciones que son expresadas como modelos, que separan la lógica empresarial y el comportamiento de la aplicación independientemente de la plataforma subyacente haciendo que de esta manera cada parte pueda evolucionar a su propio ritmo, además de que permite aislar el núcleo de la aplicación y su ciclo incesante de rotación, facilitando la interoperabilidad dentro y fuera de los límites de la plataforma, ya sea abierta o propietaria (OMG, 2023).

Lenguaje de modelado de flujo de interacción (IFML) es un estándar de la OMG, que se utiliza para modelar contenido, interacciones por parte del usuario y controles front-end de las aplicaciones, con el fin de poder integrarlo con el modelado del resto de la aplicación, tratando de establecer la interacción del usuario con la aplicación cuando se ha vuelto compleja por la cantidad de elementos existentes, lo que le ofrece una ventaja contra otros lenguajes de modelado como UML, además contando con una de las características más importantes de no depender de una plataforma especifica (*About the Interaction Flow Modeling Language Specification Version 1.0*, s. f.).

Problemática

Según (Mena Roa, 2021), hasta el 2021 se han publicado más de 1800 millones de sitios web, este número incrementa o disminuye dependiendo del año, pero en general, tiene una tendencia a seguir en aumento en los siguientes años, y aunque la mayor parte de ellos se encuentran inactivos, la cantidad de sitios web activos sigue en aumento. La gran cantidad de páginas web existentes se debe a la mejora en cuanto a las funcionalidades y capacidades que pueden cubrir este tipo de aplicaciones, ya que han pasado de ser simples páginas estáticas a

presentar contenido dinámico, lo cual ha causado que cada vez más empresas, instituciones y en general nuevas áreas se interesen en tener presencia en internet (Llamuca-Quinaloa et al., 2021). Esto se debe a que un sitio web puede brindar opciones desde hacer visible un simple blog con información personal hasta el uso en el comercio de artículos a nivel internacional, prestación de servicios, entre muchas otras alternativas de uso que se puede obtener al desarrollarlo.

Sin embargo, así como el incremento de funcionalidades y capacidades incide en una mayor acogida de este tipo de aplicaciones, también incide en el aumento de la dificultad y complejidad que conlleva desarrollar una aplicación web. Esto ocasiona que para cumplir con todos los requerimientos de un sistema se necesite de todo un equipo de desarrollo, lo cual a su vez hace que el proceso sea más propenso a errores o inconsistencias. A la par de estos problemas también se encuentra los costos que conlleva realizar una aplicación web, esto incrementará proporcionalmente a la cantidad de funcionalidades, almacenamiento, velocidad, diseño, tiempo y desarrolladores que se requieran para cumplir con el objetivo de la aplicación (P. Sharma, 2021).

Para tratar de reducir estos problemas se han propuesto diferentes soluciones como el uso de metodologías, implementación de patrones de diseño, entre otras. En el caso de las metodologías se trata de aplicar las que están basadas en el manifiesto ágil, las cuales ayudan a mejorar la organización y controlar el avance durante todo el proceso de desarrollo de una aplicación web (Wijekoon & Merunka, 2022). En el caso del uso de patrones de diseño, las opciones son varias y no se basan en una opción genérica, sino que esto depende del tipo de aplicación y los requerimientos que tenga la misma, por ejemplo, en (Ahmad et al., 2022) se enfatiza que el uso del Modelo Vista Controlador o MVC para identificar las mejoras en la estructura misma de la aplicación. Adicionalmente se pueden tomar otros tipos de soluciones e

incluso combinar estos conceptos, pero ya sea individual o grupalmente estas soluciones siguen necesitando que los desarrolladores escriban la mayor cantidad del código.

Además, los inconvenientes mencionados hasta ahora se centran únicamente en el proceso de desarrollo, dejando de lado los problemas que pueden presentarse al momento de agregar nuevas funcionalidades o al cambiar una funcionalidad ya existente, esto con un desarrollo tradicional conlleva un trabajo extenso y que podría incluso requerir el cambio sobre otras funcionalidades lo cual también incrementa el riesgo de que aparezcan contratiempos debido a este tipo de afectaciones (Da Silva et al., 2020).

Justificación

Actualmente, el uso de herramientas Low-Code o No-Code, son alternativas que las empresas que no pueden tener a su disposición un equipo de desarrollo pueden utilizar, ya que, si bien estas herramientas también requieren de conocimiento para utilizarlas, su correcto uso puede lograr la generación completa de una aplicación web o también se puede utilizar más de una de estas herramientas para obtener un mejor resultado (Parra Arévalo, 2020).

La diferencia que existe entre Low-Code o No-Code con MDD y MDA es que, estas últimas tienen como elemento principal los modelos, dado que, es en base a estos modelos que se generan las aplicaciones web, mientras que en el caso de Low-Code se suele optar por una funcionalidad basada en las interfaces que el usuario puede diseñar y, además, a partir de esto se plantean las reglas de negocio con las que debe cumplir el sistema. En el caso de MDD y MDA se tiene una generación tanto de las reglas del negocio como de las interfaces de usuario, pero esto se logra a partir de un diseño detallado de los modelos que representan el flujo que debe seguir el sistema (Roig Hervás, 2021).

El uso de herramientas MDD y MDA también ayuda a tener una mejor respuesta ante posibles cambios en las funcionalidades o adicionar una nueva funcionalidad de manera más

eficiente y eficaz, ya que, esto involucraría únicamente un cambio en el modelo, lo cual generaría automáticamente la representación en el sistema de este cambio o nueva funcionalidad. Además, el uso de IFML para la representación de las interacciones que tiene el usuario con la interfaz del sistema ayuda a mantener una mejor relación entre el desarrollo e implementación de reglas de negocio y la interfaz gráfica que mostrará la aplicación web.

Teniendo en cuenta lo anteriormente mencionado, el desarrollo de este trabajo permitirá identificar las diferencias y principales características que se pueden presentar al momento de hacer uso del lenguaje IFML para el desarrollo de interfaces, además de la implementación de funcionalidades a través de MDD y MDA en el desarrollo de una aplicación web. Para lograr este propósito, se llevará a cabo un análisis exhaustivo de estos conceptos y su aplicación práctica en el desarrollo de software.

Objetivos

Objetivo General

Implementar una aplicación web utilizando herramientas MDD y MDA, basada en principios de ingeniería web, desarrollando UX/UI en el lenguaje IFML.

Objetivo Específico

- Identificar las diferencias entre el desarrollo de software tradicional versus el desarrollo de software basado en ingeniería web.
- Definir los requerimientos funcionales y no funcionales basado en el desarrollo de prototipos, utilizando el lenguaje IFML.
- Diseñar los modelos basados en principios de ingeniería web, utilizando las tecnologías MDD y MDA.
- Ejecutar la fase de pruebas.

Capítulo 2: Marco Teórico

Desarrollo de software

Tradicional

El desarrollo de software tradicional sigue un proceso lineal o secuencial, además de estar acompañado por una documentación exhaustiva, esto significa que el desarrollo de software tradicional se centra en una planificación y documentación muy extensa y únicamente se centra en la entrega final del producto, esto sumado a la restricción que representa seguir una estructura secuencial ha provocado que desde el inicio del desarrollo de software de esta manera, haya sido cuestionado por temas como la estimación de tiempos, costos que tendrá el desarrollo y la falta de cumplimiento de los requerimientos que se plantea inicialmente para un sistema. Dados estos inconvenientes un término que surgió para tratar de establecer una manera económica, fiable y eficiente para el desarrollo de software fue la Ingeniería de Software, dentro de la cual, uno de los desarrollos más representativos fueron las metodologías y modelos que se pueden seguir para mejorar el orden y organización durante el proceso de desarrollo de software, entre los modelos de desarrollo de software más importantes están: prototipo, desarrollo basado en componentes, desarrollo en espiral, modelo de desarrollo de aplicaciones rápido (RAD) y modelo en cascada (Delgado Olivera & Díaz Alonso, 2021).

Basado en Web

A diferencia de un desarrollo tradicional una aplicación web está hecha específicamente para funcionar en internet, y aunque inicialmente solo podían ser páginas estáticas compuestas por archivos de hipertexto, actualmente pueden realizar una gran cantidad de funciones y trabajar de forma dinámica gracias a la conexión a servidores y bases de datos (Palomo & Gil, 2020). Debido a elementos diferenciales en una aplicación web como la seguridad, control de concurrencia u optimización, el desarrollo de este tipo de aplicaciones es diferente al proceso que se debe seguir en una aplicación de escritorio. Además, para las aplicaciones web también

se han creado diferentes métodos y metodologías que ayuden a mejorar el proceso de desarrollo basándose en los requisitos, contenido, interacciones, interfaz de usuario y los procesos que debe cubrir la aplicación. Así como se utiliza el término Ingeniería de Software para referirse al desarrollo de aplicaciones en general, dado las diferencias mencionadas anteriormente es necesario pasar al término Ingeniería Web para referirse al desarrollo de aplicaciones web y tratar particularmente a los requerimientos específicos que debe cumplir una aplicación de este tipo (Nieves-Guerrero et al., 2014).

Ingeniería Web

El uso de aplicaciones basadas en la Web incrementó debido a la incorporación de requerimientos cada vez más complejos, esto gracias a los procesos del lado del servidor, por lo cual, se debieron tomar varios conceptos de la Ingeniería de Software para el desarrollo de estos sistemas, pero dado que varias características no podían ser cubiertas por los métodos de desarrollo tradicionales, surgió la Ingeniería Web como una disciplina orientada a cubrir las particularidades que tiene una aplicación web en cuanto a desarrollo, diseño y mantenimiento, además de adaptar los métodos y modelos tradicionales para un desarrollo más específico. Otro de los motivos para el surgimiento de la Ingeniería Web es la diferencia en las áreas que abarca la creación de estas aplicaciones, ya que, adicionalmente a lo necesario para el desarrollo de software, se requiere de escritores, administradores de bases de datos, diseñadores gráficos, entre otros. Para seguir con un proceso basado en la Ingeniería Web se deben seguir un conjunto de actividades, que pueden variar de acuerdo al enfoque o autor que las proponga, una de las más conocidas son el conjunto de actividades propuestas en un Framework genérico, estas son (Pinzon, 2017):

- Comunicación: interacción para conocer los requerimientos del cliente.
- Planeamiento: plan de avances para la obtención de resultados.

- Modelado: generación de modelos para que el cliente y desarrolladores comprendan el flujo y requerimientos de la aplicación web.
- Construcción: abarca las tecnologías para las pruebas e identificación de errores.
- Implementación: entrega de la aplicación web para que el cliente evalúe y de su retroalimentación.

UX/UI

La Interfaz de Usuario (UI) son todos los componentes dentro de una aplicación web que están diseñados para que el usuario pueda interactuar con ellos, ya sea con el teclado, ratón, pantalla táctil, etc. Esto ha hecho que se vuelva cada vez más relevante para las empresas y negocios tener un mejor resultado en cuanto a la interfaz de usuario de sus aplicaciones. Por lo tanto, UI hace referencia en sí a la interacción que se genera entre el usuario y los comandos del sistema lo cual sirve para cambiar de interfaz, ingresar datos, reproducir contenidos, entre muchas otras acciones que se pueden configurar dentro de una aplicación web (Kristiadi et al., 2017).

Por otra parte, la Experiencia de Usuario (UX) es el proceso por el cual se establecen las interacciones que debe tener un usuario con el producto terminado, en este contexto se considera también la reacción y comportamiento que el usuario tenga con respecto a las diferentes interacciones que le permite tener el sistema (Joo, 2017). Esto implica evaluar la facilidad de uso, la eficiencia, la satisfacción y otros aspectos emocionales y prácticos relacionados con la experiencia del usuario mientras navega, interactúa y utiliza la aplicación.

IFML

El Lenguaje de Modelado de Flujo de Interacción está hecho para permitir obtener una representación de la estructura, interacción del usuario con la aplicación y el flujo que debe tener el front-end de una aplicación web. Una de sus principales características es que brinda

el mismo resultado independientemente del tipo de dispositivo que se vaya a utilizar para acceder a la aplicación web, estos dispositivos pueden ser: computadoras de escritorio, tablets celulares, etc. En cualquiera de estos casos IFML se asegura de mostrar las interacciones y flujo del sistema manteniendo una representación uniforme. En el caso de su uso en aplicaciones de mayor tamaño y por ende de mayor complejidad lo que se recomienda es realizar el proceso de desarrollo y pruebas de las interfaces desde una etapa temprana del proyecto, con el fin de lograr identificar los posibles errores que se podrían presentar debido a la complejidad de una interacción o proceso en específico (Yousaf et al., 2019).

En cuanto al uso de este lenguaje según la especificación de IFML, un diagrama debe contener uno o varios contenedores de vista, los cuales a su vez contienen interfaces de las ventanas o páginas web. Dentro de un contenedor de vista es donde se puede ubicar elementos como un formulario o elementos que deban ser publicados, por ende, este formulario debe tener parámetros de entrada y salida, así mismo cada uno de estos parámetros o formulario en general puede tener uno o varios eventos que representarán la interacción con el usuario (*IFML Primer* | *IFML*, s. f.).

MDD

MDD es una metodología de desarrollo de software que utiliza los modelos para generar una aplicación web, esto significa que con MDD únicamente se manipulan los modelos, en diferentes niveles de especificación para generar el código que debe llevar una aplicación, por ende estos modelos se desarrollan antes de escribir el código de la aplicación y generalmente se realizan a la par de los requerimientos, ya que, los modelos deben estar hechos de acuerdo a los requerimientos que se necesita cumplir con la aplicación a generarse (Wijekoon & Merunka, 2022). Esta metodología ofrece una perspectiva diferente del desarrollo de software, donde la creación y evolución de modelos desempeñan un papel fundamental en la generación de código funcional y de calidad.

Este proceso centralizado en los modelos se realiza para disminuir los tiempos de desarrollo, dificultad y costes al momento de desarrollar una aplicación web, esto se logra a través de una implementación de modelos generales, así como de modelos específicos que representen una funcionalidad en concreto. Tener un modelo a un nivel más detallado ayudará a mejorar los resultados de la aplicación que se genere, pero para que se lleve a cabo esa transformación automática, es necesario el uso de herramientas que permitan el paso desde los modelos que se han creado hacia el código que cumpla con los requerimientos en los que se ha basado para la creación de los modelos. Estos modelos deben ser creados con el objetivo de cubrir todos los requerimientos necesarios, para lo cual, deben basarse en algunas características que ayuden en su posterior análisis e implementación, entre las más importantes están (Guamán Coronel, 2021):

- Reutilización: permite la reutilización incluso haciendo uso de otras tecnologías.
- Optimización: la calidad del producto y cumplimiento de los requerimientos mejora a medida que los modelos se actualizan periódicamente.
- Reduce los costos: ya que el proceso de desarrollo se realiza como un proceso automático se genera un ahorro de tiempo y costos.
- Sintaxis y semántica: los modelos generados con MDD deben ser tanto sintáctica como semánticamente precisos y consistentes.
- Abstracción: trabaja con un nivel de abstracción más alto gracias al uso de los modelos que posteriormente automatizarán la generación del código.

MDA

MDA al igual que MDD se basa esencialmente en la generación de aplicaciones web a partir de los modelos creados, pero la principal diferencia es que MDA sigue los estándares de la OMG y por ende se apoya varios de sus estándares definidos, siendo el más relevante UML.

Además de esto, uno de los aspectos que más resalta la OMG acerca del uso de MDA es que se puede separar la lógica de negocio y de las tecnologías utilizadas de las plataformas que suelen ser necesarias para el levantamiento de una aplicación web (OMG, 2023).

Por otra parte, la generación de código a partir de modelos es algo en lo que varios ingenieros en el área de tecnologías de la información aún hacen mucho énfasis, esto debido a los problemas que se generan al crear totalmente una aplicación robusta. Otro punto importante son las alternativas que han surgido para tratar de minimizar estos problemas, siendo el más importante en los últimos años, el desarrollo ágil, debido a la gran acogida que ha tenido, tanto al crear nuevos modelos y metodologías que sirven en diferentes escenarios, así como su implementación paulatina en grandes proyectos. Otro inconveniente que reduce el uso de MDA u otras soluciones basadas en la generación de código a partir de modelos es la dificultad que puede existir al aplicarlo en proyectos grandes o entornos de producción. Por lo cual, es esencial utilizar diferentes herramientas basadas en estos enfoques y tratar de combinarlas hasta obtener un resultado satisfactorio (Sebastián et al., 2020).

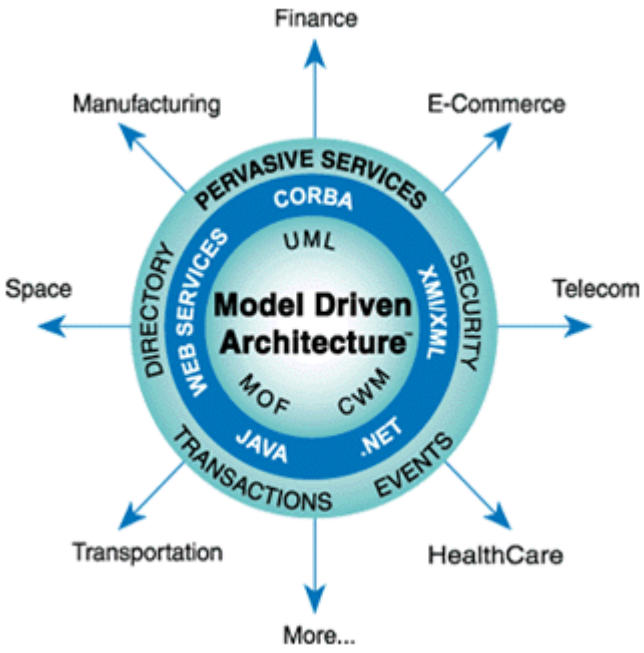
OMG define tres principales estándares para modelos MDA, MOF o por sus siglas en ingles Meta Object Facility representa el nivel meta más general el meta-metamodelo y cuenta con el propósito de permitir la incorporación de nuevos lenguajes de modelado o metamodelos entre los cuales se encuentran UML, CWM y MOF en sí mismo.(Quintero & Anaya, 2007). UML es un estándar utilizado para describir modelo de objetos y se encuentra basado en MOF lo que ayuda a que estos puedan soportar repositorios integrados, CWM o por sus siglas en ingles Common Warehouse Metamodel usa UML, MOF y XMI para modificar, manipular e intercambiar metadatas al hacer uso de herramientas warehousing y business intelligence.

Esencialmente MDA tiene características enfocadas en mejorar el ciclo de vida de una aplicación, reducir los problemas de mantenimiento e incrementar las posibilidades de

escalabilidad e interoperabilidad. Como se puede ver en la Figura 1. Principalmente la OMG resalta el apoyo de MDA en otros estándares, lo cual mejora su aplicación, ya que, hace uso de sus ventajas y soluciones que ya proveen para implementarlas o mejorarlas, la siguiente capa hace énfasis en la independencia de la plataforma y por ende de los diferentes lenguajes de programación, luego se resalta las funcionalidades que se pueden obtener y que se deben considerar al momento de utilizar una arquitectura, para finalmente resaltar varios de los tipos de aplicaciones web que se pueden crear haciendo uso de MDA.

Figura 1

Estructura y áreas de apoyo de MDA



Nota. En esta figura se pueden observar las diferentes tecnologías, estándares y procesos en los que se basa MDA empezando por los estándares hasta llegar a los tipos de aplicaciones que se pueden crear (OMG, 2023).

Modelos de transformación

Para el proceso de generación de las aplicaciones a partir de los modelos tanto en MDD como en MDA se puede implementar varios tipos de modelos y de transformaciones entre los mismos, esto dependiendo de los niveles de detalle, requerimientos y capacidades de las herramientas que se utilicen. Debido a que MDD y MDA son metodologías propuestas por la OMG, varios de los tipos de modelos que se utilizan durante el proceso de generación de aplicaciones son los mismos, entre los más comunes están (Parada et al., 2019):

- Modelo independiente de la computación CIM: es el paso inicial dentro de los procesos MDD o MDA y comprende la modelación del negocio en su totalidad y de forma general, para lograr esto se suele utilizar diagramas de Modelo y Notación de Procesos de Negocio o BPMN por sus siglas en inglés. Al ser el modelo más conceptual y abstracto suele no ser considerado en los procesos de transformación de las herramientas que utilizan MDD o MDA (Pachay Quiñónez, 2019).

- Modelo independiente de la plataforma PIM: estos modelos describen con mayor detalle el funcionamiento del sistema independientemente de la plataforma que se vaya a utilizar para su implementación, generalmente dentro del diseño de estos modelos se encuentran modelos de UML como diagramas de casos de uso, diagramas de secuencias y diagramas de clases, siendo el diagrama de clases uno de los más importantes debido a su uso dentro de las herramientas MDD y MDA, ya que, mayormente estas herramientas parten de modelos PIM y no desde los modelos CIM hacia el resto de modelos (Pons et al., 2010).

- Modelo de plataforma específico PSM: estos modelos se encargan de ir de las especificaciones creadas dentro del modelo PIM hacia las funcionalidades mismas que tendrá el sistema, para esto ya se toma en cuenta la arquitectura de la aplicación y la

plataforma en la que finalmente se obtendrá el código, esto debido a que este modelo ya es capaz de mostrar como el sistema cumplirá con los requerimientos planteados en la etapa CIM o PIM dependiendo del soporte que tenga la herramienta que se utilizó (Pachay Quiñónez, 2019).

Revisión Sistemática de la Literatura

También conocida como SLR por sus siglas en inglés tienen varias características, entre las cuales se cuenta con que al ser sistemática impide la subjetividad por parte del autor, hacen uso de toda la información disponible y es detallada, ya que se publican las fuentes y criterios requeridos en el proceso, lo que permite que la información pueda ser replicada o incluso comprobada por otros autores (Codina, 2018). Se utilizó la SLR para poder tener conocimiento de las aplicaciones, estudios y demás material relacionado con el uso de MDA, MDD e IFML con el fin de poder alcanzar los objetivos planteados en este proyecto.

Preguntas de Investigación

Se plantea 4 preguntas de investigación para la realización de la SLR, las cuales se presentan a continuación

- RQ1. ¿Cuáles son los desafíos y soluciones más relevantes en el desarrollo de aplicaciones web utilizando MDD y MDA considerando las mejores prácticas de ingeniería web?
- RQ2. ¿Cómo se logra la automatización de la transformación y generación de interfaces de usuario en aplicaciones web mediante IFML y MDD, y cuál es su impacto en la eficiencia y calidad del proceso de desarrollo?
- RQ3. ¿Cómo se comparan las herramientas y metodologías de desarrollo web basado en MDD y MDA y cuáles son las implicaciones para la implementación de aplicaciones web?

- RQ4. ¿Cómo mejoran las aplicaciones basadas en modelos que usan IFML de las aplicaciones web desarrolladas con MDD/MDA, y cómo se comparan con las aplicaciones tradicionales?

Criterios de inclusión y exclusión

Para la búsqueda se seleccionaron y revisaron artículos que se encuentran publicados desde el año 2020 hasta el 2023 para garantizar la relevancia y actualidad de la información, además se tomó en cuenta los artículos cuyo idioma de estudio sea en inglés o español, y excluir artículos que no estén disponibles de manera libre y abierta, teniendo esto en cuenta se aplicaron los siguientes criterios de inclusión:

- Artículos cuyo contenido se enfoque en MDD, MDA, IFML, herramientas de desarrollo web, aplicaciones web y calidad de software.
- Artículos que comparen herramientas, metodologías, enfoques y estrategias de MDD y MDA en el desarrollo web, así como los que evalúen la efectividad de las pruebas basadas en modelos.
- Artículos que describan enfoques y metodologías específicas de desarrollo basado en MDD, MDA e IFML en el contexto de aplicaciones web.

Proceso de búsqueda

Se realizó una búsqueda en la base de datos Google Scholar haciendo uso de la siguiente cadena de búsqueda, “MDD AND MDA AND IFML AND Desarrollo de software” esto con el fin de poder encontrar términos similares para poder ampliar el tamaño de la muestra y obtener mejores resultados, logrando como resultado la siguiente cadena de búsqueda general, “(MDD OR Desarrollo Dirigido por Modelos OR Ingeniería Dirigida por Modelos) AND (MDA OR Arquitectura Dirigida por Modelos OR Arquitectura de Software Dirigida por Modelos) AND

(IFML OR Lenguaje de modelado de flujo de interacción OR Modelado de Interfaz de Usuario) AND (Desarrollo Web OR Ingeniería de Software Web)" y , "(MDD OR Model Driven Development OR Model Driven Engineering) AND (MDA OR Model Driven Architecture OR Model Driven Software Architecture) AND (IFML OR Interaction Flow Modeling Language OR User Interface Modeling) AND (Web Development OR Web Software Engineering)", en español e inglés respectivamente, esta cadena fue modificada en cuanto a sintaxis para poder realizar la búsqueda en las bases de datos IEEE (acceso abierto), Springer (acceso abierto), y Scopus, ver en Tabla 1.

Tabla 1

Sintaxis de cadena de búsqueda

Base de datos	**Inglés**	**Español**
Springer	(MDD OR Model Driven Development OR Model Driven Engineering) AND (MDA OR Model Driven Architecture OR Model Driven Software Architecture) AND (IFML OR Interaction Flow Modeling Language OR User Interface Modeling) AND (Web Development OR Web Software Engineering)	
IEEE	("All Metadata":MDD) OR ("All Metadata":Model Driven Development) OR ("All Metadata":Model Driven	("All Metadata":MDD) OR ("All Metadata":Desarrollo Dirigido por Modelos) OR ("All Metadata":Ingeniería

Base de datos	Inglés	Español
	Engineering) AND ("All Metadata":MDA) OR ("All Metadata":Model Driven Architecture) OR ("All Metadata":Model Driven Software Architecture) AND ("All Metadata":IFML) OR ("All Metadata":Interaction Flow Modeling Language) OR ("All Metadata":User Interface Modeling) AND ("All Metadata":Web Development) OR ("All Metadata":Web Software Engineering)	Dirigida por Modelos) AND ("All Metadata":MDA) OR ("All Metadata":Arquitectura Dirigida por Modelos) OR ("All Metadata":Arquitectura de Software Dirigida por Modelos) AND ("All Metadata":IFML) OR ("All Metadata":Lenguaje de modelado de flujo de interacción) OR ("All Metadata":Modelado de Interfaz de Usuario) AND ("All Metadata":Desarrollo Web) OR ("All Metadata":Ingeniería de Software Web
Scopus	(mdd OR model AND driven AND development OR model AND driven AND engineering) AND (mda OR model AND driven AND architecture OR model AND driven AND software AND architecture) AND (ifml OR interaction AND flow AND modeling AND language OR user AND interface AND modeling) AND (web	

Base de datos	Inglés	Español
	AND development OR web AND software AND engineering) AND (LIMIT-TO (PUBYEAR , 2023) OR LIMIT-TO (PUBYEAR , 2022) OR LIMIT-TO (PUBYEAR , 2021) OR LIMIT-TO (PUBYEAR , 2020)) AND (LIMIT-TO (DOCTYPE , "ar")) AND (LIMIT-TO (OA , "all"))	

Selección de estudios primarios

Después de realizar la búsqueda en las bases de datos usando la cadena de búsqueda general se procedió a hacer la filtración de los resultados con los artículos que se presentaron en la búsqueda, se realizaron un total de 3 filtros, el primer filtro se realizó leyendo únicamente el título de los artículos, el segundo filtro de acuerdo a la lectura del resumen, el tercer y último filtro se realizó leyendo completamente cada sección del artículo, se descartó cualquier artículo que no tuviera relación con los criterios de inclusión y exclusión presentados anteriormente para finalmente tener los artículos más adecuados, los resultados del cribado se pueden observar en la Tabla 2.

Tabla 2

Resultados de la cadena de búsqueda

Base de datos	Idioma	Búsqueda	1º Filtro	2º Filtro	3º Filtro
IEEE	Español	89	8	7	4
	Inglés	1.261	2	2	1
Springer	Inglés	624	2	2	1
Scopus	Inglés	164	12	7	5

Una vez realizado el cribado se tuvo un total de 11 artículos en total, con el fin de comprender mejor el propósito de cada trabajo y su alineación con los objetivos de este trabajo, se identificaron algunas características significativas de cada artículo, las cuales se observan en la Tabla 3.

Tabla 3

Resumen de características de los estudios seleccionados

Código	Título	Objetivo	Herramientas	Cita
E1	A Family of Experiments to Compare Two Model-Driven Development Tools vs a Traditional Development Method	Comparación entre MDD y un desarrollo tradicional	WebRatio e INTEGRANOVA	(Panach et al., 2022)

Código	Título	Objetivo	Herramientas	Cita
E2	MDD: A Unified Model-driven Design Framework for Embedded Control Software	Propuesta de un marco de trabajo de diseños basado en modelos unificados	Simulink, SCADE, Tsmart y Polychrony	(Su et al., 2021)
E3	Dealing with Non-Functional Requirements in Model-Driven Development: A Survey	Mostrar las barreras y beneficios al gestionar los requisitos no funcionales como parte del MDD	Encuestas	(Ameller et al., 2021)
E4	Evaluating Model-Driven Development Claims with respect to Quality: A Family of Experiments	Comparar el efecto que genera en la calidad de software al hacer uso de MDD o un desarrollo tradicional	Integranova	(Panach et al., 2021)
E5	Effective testing of Android apps using extended IFML models	Pruebas sobre aplicaciones Android utilizando	ADAMANT, Monkey,	(Pan et al., 2020)

Código	Título	Objetivo	Herramientas	Cita
		una herramienta que extiende las características de IFML específicamente para aplicaciones Android	AndroidRipper y Gator	
E6	A methodology for transforming BPMN to IFML into MDA	Uso de las características y ventajas de MDA, IFML y business process model and notation BPMN para crear una metodología que permita generar una aplicación de forma semiautomática.		(Sajji et al., 2022)
E7	A Model Driven Approach for Unifying User Interfaces Development	Unificar los resultados de las herramientas basadas en MDD	Herramienta propia	(Soude & Koussonda, 2022)

Código	Título	Objetivo	Herramientas	Cita
		para crear interfaces de Usuario con la ayuda de diferentes lenguajes que son parte de estándares existentes para la generación de interfaces.		
E8	Graphical User Interfaces Generation from BPMN (Business Process Model and Notation) via IFML (Interaction Flow Modeling Language) up to PSM (Platform Specific Model) Level	Generación de interfaces de Usuario de BPMN con IFML, mediante la transformación de los modelos equivalentes entre los dos lenguajes.	WebRatio y Eclipse	(Sajji et al., 2023)
E9	Integrated model-driven development of	Presenta un enfoque integrado de		(Yigitbas et al., 2020)

Código	Título	Objetivo	Herramientas	Cita
	self-adaptive user interfaces	desarrollo impulsado por modelos para desarrollar interfaces de usuario que pueden adaptarse automáticamente a los cambios en el contexto.		
E10	Applying MDD in the content management system domain	Creación de una extensión de Joomla para la generación de código mediante el uso de MDD y comparación de sus resultados, facilidad de uso y curva de aprendizaje.	Joomla	(Priefer et al., 2021)

Código	Título	Objetivo	Herramientas	Cita
E11	Transformation From CIM to PIM: A Systematic Mapping	Presentar un estudio sobre la transformación de modelos de la arquitectura de información de un sistema (CIM) a la arquitectura de diseño de un sistema (PIM) en el contexto de la Arquitectura Dirigida por Modelos (MDA).		(Silega et al., 2022)

Análisis de literatura

Una vez realizado el cribado se tuvo un total de 11 documentos de los cuales se pudo obtener la siguiente información que se separan en 4 grandes categorías: (1) transformación y perspectivas, (2) exploración de fundamentos y desafíos, (3) diseño y adaptabilidad de interfaces, (4) aplicaciones, pruebas y resultados.

En la primera categoría *"Transformation From CIM to PIM:A Systematic Mapping"* (Silega et al., 2022) presenta un estudio sobre la transformación de CIM a PIM en el contexto de MDA por medio de una revisión sistemática de la literatura para identificar los enfoques, herramientas y técnicas utilizadas en la transformación de modelos, llegando a que el 84% de

las transformaciones incorporan modelos basados en UML o BPMN (Notación de Modelado de Procesos de Negocio), reflejando la extensa adhesión a las directrices de OMG dentro del contexto MDA. Los lenguajes de transformación más prevalentes son ATL (Lenguaje de Transformación Atlas) y QVT (Consulta/Vista/Transformación), empleados en el 86% de los casos, indicando su amplia adopción. Desde 2016, la mayoría de los proyectos implementan herramientas para respaldar las transformaciones, siendo Eclipse la más utilizada en el proceso de transformación de modelos.

Con respecto a la segunda categoría se abordan los fundamentos y desafíos de MDD y MDA, en *"Dealing with Non-Functional Requirements in Model-Driven Development: A Survey"* (Ameller et al., 2021) se busca proporcionar información y estrategias para el manejo de requisitos no funcionales (NFR) en el desarrollo de software impulsado por modelos (MDD), en este estudio se visualiza que los desafíos comunes en cuanto al levantamiento de NFR en el desarrollo impulsado por modelos son: la falta de una definición clara de lo que es un NFR, dificultad para medir y evaluar los NFR, complejidad en la integración y la falta de herramientas y técnicas adecuadas en el manejo de NFR en el desarrollo impulsado por modelos, en vista de lo anterior se han propuesto diversas soluciones tales como: agregar los NFR a los modelos haciendo uso de perfiles UML o agregándolos directamente a los UML (por ejemplo, como un comentario), proporcionar un metamodelo completo diseñado para el uso de NFR, entre otros. Así mismo *"A methodology for transforming BPMN to IFML into MDA"* (Sajji et al., 2022) expone una solución referente a la creciente complejidad de los sistemas y el alto costo de la migración tecnológica haciendo uso de MDA, se busca describir los requisitos funcionales y de rendimiento de una aplicación, esto independientemente de la plataforma, haciendo uso de reglas de transformación semiautomáticas para la transformación de BPMN a IFML en MDA, obteniendo como resultados una metodología cuyos beneficios se observan en la eficiencia y precisión de transformación partiendo desde un modelo empresarial a un diseño de software,

no obstante también se explica que todavía se requiere de cierta intervención humana para el ajuste y refinamiento del modelo resultante.

Dentro de la tercera categoría *"Graphical User Interfaces Generation from BPMN (Business Process Model and Notation) via IFML (Interaction Flow Modeling Language) up to PSM (Platform Specific Model) Level"* (Sajji et al., 2023) presenta información sobre la generación de interfaces graficas de usuario por medio del uso de MDA en combinación con los modelos de BPMN e IFML, para llevar a cabo este proceso se desarrollaron los metamodelos de BPMN e IFML utilizando el entorno Eclipse, se usaron reglas de transformación en el lenguaje ATL (Atlas Transformation Language) para la obtención de los modelos IFML para después ser importado a la herramienta de WebRatio, Mediante la utilización conjunta de Eclipse y el lenguaje ATL, se logra una conversión efectiva y precisa desde un nivel PIM, con el propósito de validar la metodología propuesta, se ejecutó un caso de estudio focalizado en las operaciones CRUD correspondientes al sistema de gestión de reclamos de servicios en una entidad empresarial, posteriormente se concluyó que la metodología MDA facilita el desarrollo de software al generar aplicaciones automáticamente a partir de modelos, sin requerir código fuente. En la misma línea *"Integrated model-driven development of self-adaptive user interfaces"* (Yigitbas et al., 2020) busca presentar un enfoque integrado de desarrollo impulsado por modelos para interfaces de usuario auto adaptativas a través del uso de los lenguajes de modelado específicos del dominio: ContextML para contextualización y AdaptML para modelar la interfaz. Este método permite la adaptación automática de la interfaz a cambios contextuales, para validar este enfoque, se crearon dos aplicaciones (correo electrónico y gestión de bibliotecas), evaluadas con pruebas de usabilidad con 23 y 1 participantes respectivamente donde se recolectaron datos sobre la aceptación de las adaptaciones de la interfaz de usuario y se realizaron entrevistas para obtener comentarios adicionales. Los resultados destacaron la utilidad y facilidad de uso de las adaptaciones automáticas, respaldadas por el desarrollo

basado en modelos que agiliza la creación y mantenimiento de la interfaz. Sin embargo, se observó que el enfoque carece de soporte para analizar y resolver conflictos en reglas de adaptación de interfaz de usuario, lo que puede ser complejo y propenso a errores al especificar manualmente adaptaciones sólidas, además de requerir un cierto nivel de experiencia en desarrollo dirigido por modelos y lenguajes específicos de dominio. Además, IFML se empleó para especificar vistas y flujos de navegación en interfaces auto adaptativas, separando preocupaciones, facilitando el modelado y mantenimiento. Por otra parte *"A Model Driven Approach for Unifying User Interfaces Development"* (Soude & Koussonda, 2022) busca establecer un enfoque impulsado por modelos con el fin de unificar el desarrollo de interfaces de usuarios haciendo que el desarrollo de aplicaciones web sea más eficiente y accesible independientemente de la plataforma o framework elegido, para esto se definió una metodología de diseño de interfaz basada en un portal con el fin de que los usuarios puedan detallar las acciones relacionadas con el menú y la aplicación, posteriormente se implementó una arquitectura basada por modelos haciendo uso del Lenguaje de Modelado de Interfaz de Usuario (UIML), consecuentemente se generó el código proponiendo un lenguaje de modelado basado en texto y directivas de sustitución simples, y finalmente se implementó y refino la interfaz a través del uso de un panel de propiedades para cada elemento, para probar el enfoque se realizó una prueba donde participaron 15 estudiantes donde se hizo uso de plantillas en diferentes campos usando lenguaje de modelado y XSLT propuestos, donde se obtuvo una buena respuesta debido a la sintaxis compacta y la curva de aprendizaje baja.

Dentro de la cuarta categoría "Effective testing of Android apps using extended IFML models" (Pan et al., 2020) se expone la importancia de las pruebas basadas en modelos y como pueden mejorar la confiabilidad de las aplicaciones móviles, presentándolos como una técnica más eficiente y efectiva para la prueba de aplicaciones móviles en comparación a los métodos tradicionales, para esto se hizo uso de Adamant, Monkey, Gator y métodos manuales

en un experimento controlado a pequeña escala, encontrando que ambos enfoques lograron una eficacia comparable en términos de costo, no obstante Adamant probó ser más efectiva, con una alta tasa de cobertura de código y detección efectiva de errores, además de que la calidad de los modelos usados para representar el comportamiento esperado de una aplicación tiene un impacto en las pruebas de GUI. Siguiendo esta línea *"Applying MDD in the content management system domain"* (Priefer et al., 2021) se expone una investigación empírica de métodos mixtos sobre la aplicación de Model-Driven Development (MDD) en el dominio de los sistemas de gestión de contenido (CMS), se introduce JooMDD, una infraestructura de MDD para extensiones de CMS desarrollada en Joomla. Se llevo a cabo un experimento que involucró a 14 desarrolladores con experiencia, se evaluó la calidad del código generado por JooMDD frente al generado manualmente, concluyendo que el código de JooMDD era igual o superior en calidad, y que la herramienta permitía un enfoque más eficiente en la lógica de negocio, por otro lado también se realizaron 3 casos de estudio donde El primero se enfocó en una extensión de galería de imágenes, demostrando un desarrollo más rápido y eficiente que el manual. El segundo abordó la migración de una extensión existente de Joomla a JooMDD, comparando la calidad del código generado con el original, obteniendo resultados favorables para JooMDD. El tercero se centró en crear una extensión de directorio de negocios utilizando JooMDD, en la cual los desarrolladores reportaron una mayor concentración en la lógica de negocio en lugar de los detalles de implementación, en general, se concluyó que JooMDD puede mejorar la eficiencia y calidad del desarrollo, permitiendo a los desarrolladores centrarse en la lógica comercial. Sin embargo, se señaló una limitación en el tamaño de la muestra, lo que podría afectar la representatividad de los resultados en una población más amplia de desarrolladores de extensiones de CMS. En consonancia con esto *"MDD: A Unified Model-driven Design Framework for Embedded Control Software"* (Su et al., 2021) se enfoca en buscar una solución a los desafíos que enfrentan las herramientas de diseño actuales en

cuanto a las funcionalidades de simulación y la generación de código que han ido aumentando gracias a la creciente complejidad de los requisitos de control, sobrecarga de tiempo y la degradación de velocidad de emulación, además de la generación de contenido redundante que interfiere en la generación de un código de alta calidad, se hizo uso de varias herramientas tales como: Simulink, SCADE, Tsmart y Polychrony para realizar un análisis cuantitativo de la velocidad de simulación y la calidad de la generación de código, por otra parte, también utilizaron MDD para la realización de un desarrollo colaborativo de las diferentes herramientas, concluyendo que el marco unificado impulsado por modelos ayuda a abordar los desafíos que enfrentan las herramientas de diseño. Así mismo *"Evaluating Model-Driven Development Claims with respect to Quality: A Family of Experiments"* (Panach et al., 2021) presenta 6 réplicas de un experimento base con el fin de estudiar el impacto de la complejidad de un problema en la calidad de software en el contexto de un MDD, se definieron 2 tipos de replicaciones, las estrictas que se apegaron más al experimento base y la de objetos donde se buscaba aumentar más la complejidad del problema, dichas replicaciones se realizaron durante 3 años consecutivos en diferentes ubicaciones geográficas, se realizó una comparación de calidad entre un método tradicional y MDD haciendo uso de casos de prueba ejecutados en un sistema funcional dando como resultado que en problemas complejos MDD produce un software de mayor calidad. En el mismo ámbito "*A Family of Experiments to Compare Two Model-Driven Development Tools vs a Traditional Development Method"* (Panach et al., 2022) presenta 7 repeticiones en 56 unidades de muestra que buscan comparar el método tradicional frente al MDD haciendo uso de las herramientas INTEGRANOVA y WebRatio, se hizo la comparación en Idoneidad Funcional donde se pudo observar que MDD supero al método tradicional especialmente en cuanto a problemas complejos, aquí se encontraron diferencias para problemas simples, siendo MDD más eficiente, para los problemas complejos no se encontró gran diferencia, en satisfacción se encontró que el método tradicional tiene más

intención de uso, esto debido a la falta de flexibilidad por parte de MDD, ya que frente al desarrollo tradicional permite un desarrollo más personalizado en comparación a MDD que solo permite la generación de lo que se puede generar a partir de modelos, en consecuencia se pudo observar que al hacer uso de MDD se mejora la productividad y la calidad del software, además de reducir los costos y tiempo de desarrollo, ya que se automatiza gran parte del proceso, por otro lado se menciona que en la adopción de MDD la curva de aprendizaje es un desafío al momento de comprender las herramientas MDD.

Respuestas a preguntas de investigación

RQ1. ¿Cuáles son los desafíos y soluciones más relevantes en el desarrollo de aplicaciones web utilizando MDD y MDA considerando las mejores prácticas de ingeniería web?

Los desafíos principales en el desarrollo de aplicaciones web mediante MDD y MDA incluyen la transformación entre niveles de abstracción, el manejo de requisitos no funcionales (NFR), la generación de interfaces de usuario y las pruebas basadas en modelos. Las soluciones abarcan desde el uso de herramientas de transformación ATL y QVT hasta la integración de NFR en modelos con perfiles UML y reglas de transformación semiautomáticas. La automatización de la generación de interfaces y pruebas mediante MDA también se resalta como una solución efectiva en varios contextos.

RQ2. ¿Cómo se logra la automatización de la transformación y generación de interfaces de usuario en aplicaciones web mediante IFML y MDD, y cuál es su impacto en la eficiencia y calidad del proceso de desarrollo?

La automatización de la transformación y generación de interfaces de usuario en aplicaciones web a través del uso de IFML y MDD ha sido objeto de estudio en varios artículos. Por ejemplo, uno de los enfoques propone una transformación semiautomática que convierte

modelos de procesos (BPMN) en modelos de interacción (IFML) en el contexto de MDA. Además, se han planteado abordajes para la generación automatizada de interfaces de usuario a partir tanto de modelos BPMN como de IFML. Estas estrategias enfatizan la eficiencia en la creación de interfaces adaptables, lo que a su vez contribuye positivamente a la calidad del proceso de desarrollo.

RQ3. ¿Cómo se comparan las herramientas y metodologías de desarrollo web basado en MDD y MDA y cuáles son las implicaciones para la implementación de aplicaciones web?

Las herramientas y metodologías de desarrollo web basadas en Model-Driven Development (MDD) y Model-Driven Architecture (MDA) son comparables en términos de enfoque y resultados. En MDD, se utiliza UML y BPMN para representar modelos y se aplican transformaciones con ATL y QVT. MDA también utiliza estos lenguajes y técnicas. MDD aborda la gestión de requisitos no funcionales y MDA se centra en la generación de interfaces y pruebas de aplicaciones. Ambos enfoques mejoran la eficiencia y calidad del desarrollo web, automatizando partes del proceso y reduciendo costos y tiempo. Sin embargo, pueden requerir ajustes manuales y enfrentar desafíos en la curva de aprendizaje y la adaptación. Las implicaciones para la implementación de aplicaciones web radican en la necesidad de una comprensión sólida de los modelos y las herramientas, junto con la posibilidad de ajustes y refinamientos manuales para garantizar resultados óptimos en casos específicos.

RQ4. ¿Cómo mejoran las aplicaciones basadas en modelos que usan IFML de las aplicaciones web desarrolladas con MDD/MDA, y cómo se comparan con las aplicaciones tradicionales?

Las aplicaciones basadas en modelos que utilizan IFML en el marco de MDD/MDA ofrecen mejoras en la eficiencia y calidad del desarrollo. Estos enfoques permiten transformaciones precisas, generación automática de código y adaptación de interfaces de

usuario, lo que resulta en una mayor agilidad y menor riesgo de errores en comparación con métodos tradicionales. Sin embargo, se enfrentan desafíos como la curva de aprendizaje y la gestión de requisitos no funcionales.

Capítulo 3: Análisis

UI/UX

La interfaz de usuario (UI) es lo que ayuda al usuario a tener interacciones con el sistema de software de acuerdo al servicio requerido y a las características del sistema. En el caso de una aplicación web, la interfaz de usuario abarcaría el desarrollo de las diferentes pantallas que permiten cada una de las funcionalidades que la aplicación debe cumplir, por lo tanto, se trabaja los elementos visuales como tipografía, imágenes, botones y colores, esto se realiza con el fin de obtener una representación de las pantallas y la interacción que tendría el usuario dentro de cada una (V. Sharma & Tiwari, 2021).

La experiencia de usuario (UX) se encarga de manejar la comunicación que debe existir entre los usuarios y el sistema. Para mejorar este punto se suelen utilizar diferentes pruebas, las cuales ayudan a identificar las interacciones que tienen los usuarios para posteriormente corregir errores o crear mejoras que faciliten los procesos que realiza el usuario. A diferencia de UI, la experiencia de usuario en una página web puede incluir elementos externos a un sistema, ya que, la experiencia que el usuario tiene con cierta actividad depende también de elementos como el tiempo o condiciones en las que se recibe un producto o en las que se realiza una actividad (V. Sharma & Tiwari, 2021).

Como se explicó anteriormente, UI y UX son conceptos que se encargan de mejorar la forma en la que el usuario realiza diferentes funcionalidades dentro del sistema, al igual que en otras áreas, estos elementos tienen consideraciones específicas en el caso de trabajar sobre una aplicación web. En este trabajo se va a trabajar estos conceptos mediante el lenguaje IFML, ya que, este ayuda a controlar y manejar el contenido tanto de UI como de UX y a su vez se pueden utilizar herramientas que faciliten su comprensión e implementación.

Requerimientos funcionales y no funcionales

Uno de los pasos previos a crear los diferentes diagramas que representan el flujo y funcionalidades de la aplicación suele ser la especificación de los requerimientos de software, este proceso es independiente a las diferentes metodologías y estándares que se utilicen para el desarrollo de una aplicación web. Para obtener una mejor comprensión de los requerimientos del sistema se suele trabajar con agrupaciones de los diferentes tipos de requerimientos que se muestran dentro del estándar («IEEE Recommended Practice for Software Requirements Specifications», 1998). Una de las formas más comunes es la especificación de requerimientos funcionales y no funcionales, ya que esto permite diferenciar, las funcionalidades específicas y que cumplen con cierto flujo, de las funcionalidades que son aplicables en casi todo el sistema.

Tabla 4

Requerimientos funcionales

Código	Nombre	Descripción
RF01	Crear paciente	El sistema debe permitir la creación de nuevos pacientes con los atributos: nombre, fecha de nacimiento, peso y altura.
RF02	Listar pacientes	El sistema debe permitir al usuario visualizar un listado con los pacientes que se han agregado al sistema.
RF03	Actualizar paciente	El sistema permite modificar los datos de los pacientes
RF04	Eliminar paciente	El sistema debe permitir la eliminación de un paciente ya registrado.

Código	Nombre	Descripción
RF05	Filtrar paciente	El sistema debe permitir el filtro de pacientes por los campos: nombre, fecha de nacimiento, peso y altura.

Nota. Se especifican los requerimientos para pacientes, pero el resto del sistema debe cumplir con los mismos requerimientos para cada una de las entidades que se manejen dentro del sistema.

Tabla 5

Requerimientos no funcionales

Código	Nombre	Descripción
RNF01	Interfaz de usuario	El sistema debe manejar una interfaz de usuario sencilla e intuitiva para facilitar el manejo por parte del usuario.
RNF02	Mantenimiento del sistema	El sistema debe contar con una documentación y estructura que permita su actualización o soporte de manera rápida.
RNF03	Consistencia	Si las funcionalidades dentro de una pantalla son similares en diferentes módulos debe mantener un diseño de la vista y forma de usar similar.
RNF04	Autoajustable	La aplicación debe poder ajustarse a los diferentes tamaños de pantalla dependiendo del dispositivo en el que se habrá el aplicativo.

Código	Nombre	Descripción
RNF05	Confiabilidad	El sistema debe ser accesible en cualquier momento que el usuario lo requiera y considerar posibles escenarios de fallos o problemas de rendimiento.

UML

El Lenguaje Unificado de Modelado (UML), tiene como objetivo brindar herramientas para el análisis, diseño e implementación de diferentes sistemas de software, basándose en métodos orientados a objetos, lenguajes de modelado y lenguajes de descripción arquitectónica. Para lograr cumplir con este objetivo UML debe cumplir con diferentes requisitos de semántica y sintaxis que ayuden a obtener buenos resultados al momento de interpretar la información entre los modelos y herramientas utilizadas, estos requisitos son (OMG, 2017):

- Establecer las reglas para la definición de conceptos y combinación de los atributos y relaciones que debe contener un modelo, esto debe estar basado en la Facilidad de Meta-Objeto o MOF por sus siglas en inglés, ya que, este estándar se encarga de especificar la sintaxis abstracta de UML.
- Especificar detalladamente como las herramientas deben interpretar los conceptos de UML independientemente de la tecnología que se utilice.
- Especificar de manera comprensible para los usuarios cada uno de los conceptos y reglas de combinación que tiene UML.

Para lograr una mejor representación de los sistemas de software UML define varios diagramas que se pueden crear para representar la estructura, comportamiento y las

interacciones del sistema. Estos diagramas se encuentran dentro de diferentes categorías, las cuales se visualizan en la Tabla 4 (*What is UML | Unified Modeling Language*, 2023).

Tabla 6

Diagramas UML

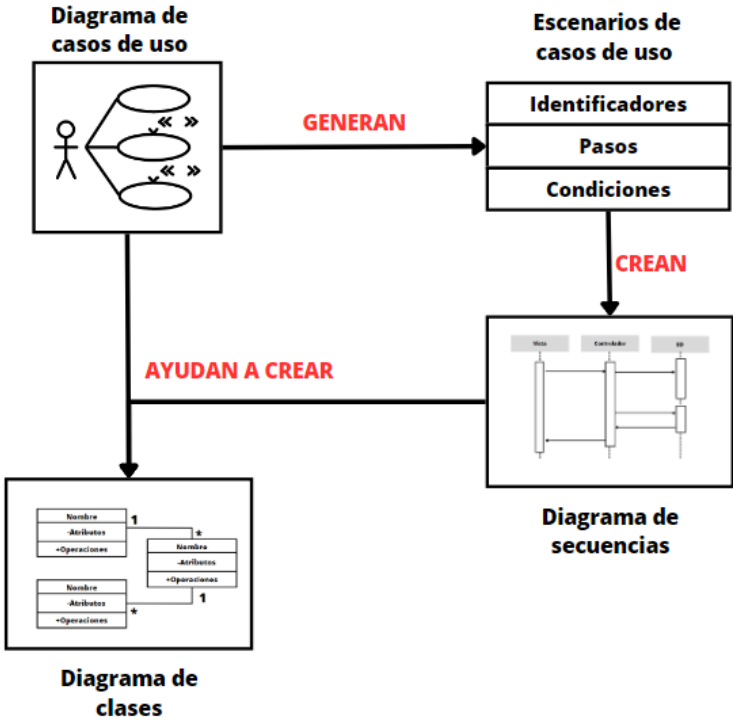
Categoría	Diagramas
De Estructura	• Diagrama de clases • Diagrama de objetos • Diagrama de componentes • Diagrama de estructura compuesta • Diagrama de paquetes • Diagrama de implementación
De Comportamiento	• Diagrama de casos de uso • Diagrama de actividades • Diagrama de máquina de datos
De Interacción	• Diagrama de secuencia • Diagrama de comunicación • Diagrama de tiempo • Diagrama de descripción general de interacción

Cada uno de estos diagramas tienen un objetivo diferente y también cada uno maneja su respectiva notación tanto para los objetos que intervienen como para los elementos relacionales. En este trabajo se utilizarán tres de estos diagramas, elegidos de acuerdo a su importancia en diferentes metodologías y a la relación en el tema presentado. Los diagramas a utilizar son: diagrama de clases (diagrama estructural), diagrama de casos de uso (diagrama de comportamiento) y diagrama de secuencia (diagrama de interacción) los cuales se detallan en la Figura 2, para la realización de los diagramas anteriormente mencionados se hará uso de la herramienta case de modelado ASTAH que permite la visualización de diseños de software,

logrando sistemas seguros y protegidos haciendo uso de diagramas UML, ER, diagramas de flujo, diagramas de flujo de datos, mapas mentales, entre otros. (*Premier Diagramming, Modeling Software & Tools*, s. f.)

Figura 2

Relación entre los diagramas UML



Nota. En esta figura se muestra una vista general de los diagramas que se van a usar en el ejemplo del consultorio médico y la relación que hay entre cada diagrama.

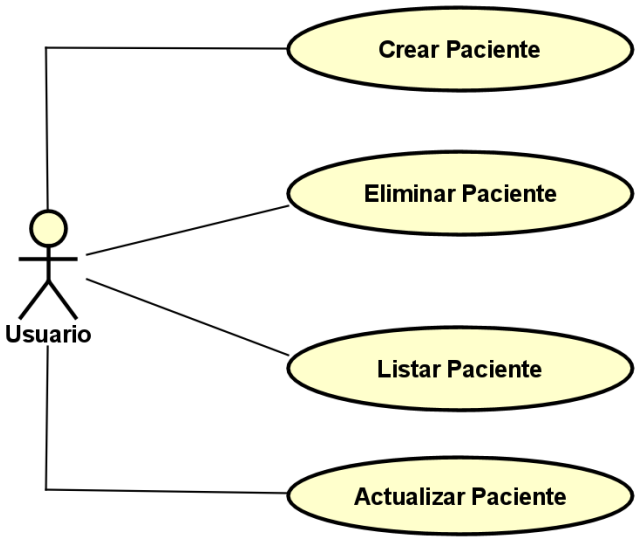
Diagrama de Casos de Uso

Este diagrama es uno de los más utilizados y sirve esencialmente para representar los requisitos funcionales de un sistema, esto se logra mediante la representación de un actor (cualquier persona o cosa que interactúa con el sistema) y los diferentes casos de uso (acciones o funciones que el actor puede hacer dentro del sistema) que existen de acuerdo al sistema a implementar, además de estos elementos, también se deben considerar las

relaciones que pueden existir entre ellos, ya que, gracias a esto se puede dar una mayor comprensión de las funcionalidades que se representan en el diagrama (Microsoft 365 Team, 2019).

Figura 3

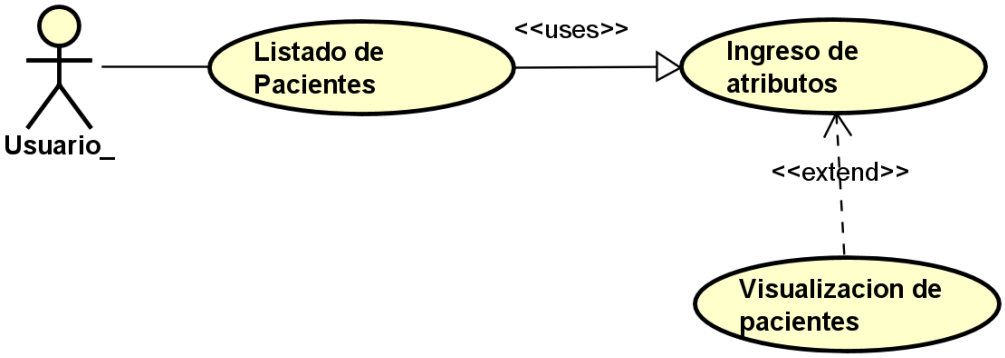
Diagrama de caso de uso Usuario - Consultorio Médico



Nota. En esta figura se muestra los casos de uso que va a realizar un usuario dentro de la aplicación del Consultorio Médico

Figura 4

Diagrama de caso de uso Usuario - Listado Pacientes



Nota. En esta figura se muestra el caso de uso listar pacientes de manera detallada.

Haciendo uso del diagrama de casos de uso, se establece las interacciones con el sistema de cada caso de uso.

1- **Nombre del caso de uso:** Crear paciente

Actor: Usuario

Descripción: El usuario se quiere registrar un nuevo paciente dentro del sistema del consultorio médico

Precondición: El usuario no puede estar registrado en el sistema

Postcondición: Se crea un paciente nuevo en el sistema

Descripción extendida:

Tabla 7

Caso de uso para crear paciente

Acciones del actor	Acciones del sistema
1- El usuario selecciona la vista para crear pacientes	
	2- El sistema muestra el formulario para la creación de pacientes
3- El usuario ingresa el nombre, fecha de nacimiento, peso y altura	
	4- El sistema valida que los datos ingresados cuenten con el formato preestablecido y no vacíos

Acciones del actor	Acciones del sistema
	5- El sistema crea un nuevo paciente y lo guarda en el sistema
	6- El sistema muestra el paciente creado

2- **Nombre del caso de uso:** Eliminar paciente

Actor: Usuario

Descripción: El usuario del consultorio médico elimina un paciente del sistema

Precondición: Debe existir al menos un usuario ya registrado

Postcondición: El paciente es eliminado del sistema del consultorio médico

Descripción extendida:

Tabla 8

Caso de uso para eliminar paciente

Acciones del actor	Acciones del sistema
1- El usuario selecciona la vista de listar pacientes	
	2- El sistema muestra la lista de pacientes existentes
3- El usuario busca el paciente a eliminar por medio del nombre, fecha de nacimiento, peso o altura	

Acciones del actor	Acciones del sistema
	4- El sistema muestra el listado de los pacientes según los filtros seleccionados
5- El usuario selecciona y elimina el paciente	
	6- El sistema elimina al paciente de la base de datos y actualiza la lista de pacientes

3- **Nombre del caso de uso:** Listar pacientes

Actor: Usuario

Descripción: El usuario pide el listado de todos los pacientes registrados en el sistema del consultorio médico

Precondición:

Postcondición: Se muestra la lista de los pacientes registrados en el sistema

Descripción extendida:

Tabla 9

Caso de uso para listar paciente

Acciones del actor	Acciones del sistema
1- El usuario selecciona la vista para listar pacientes	
	2- El sistema muestra la lista de pacientes existentes

Acciones del actor	Acciones del sistema
3- El usuario pide filtrar los resultados de la lista de paciente según el nombre, fecha de nacimiento, peso o altura	
	4- El sistema recupera los pacientes que ya se encuentran registrados en el sistema según el filtro aplicado
	5- El sistema muestra una tabla con la lista de los pacientes registrados en el sistema

4- **Nombre del caso de uso:** Actualizar paciente

Actor: Usuario

Descripción: El usuario del consultorio médico pide editar los datos de un paciente que ya está registrado en el sistema

Precondición:

Postcondición: Se actualizan los datos del paciente seleccionado

Descripción extendida:

Tabla 10

Caso de uso para actualizar paciente

Acciones del actor	Acciones del sistema
1- El usuario selecciona la vista para listar pacientes	

Acciones del actor	Acciones del sistema
	2- El sistema muestra la lista de pacientes existentes
3- El usuario busca el paciente a eliminar por medio del nombre, fecha de nacimiento, peso o altura	
	4- El sistema muestra el listado de los pacientes según los filtros seleccionados
5- El usuario actualiza los campos requeridos	
	6- El sistema valida que los datos ingresados cuenten con el formato preestablecido y actualiza al paciente en la base de datos
	7- El sistema muestra el listado de pacientes actualizado

Diagrama de Secuencias

Un diagrama de secuencia sirve para especificar la estructura y secuencias que existen entre los objetos y mensajes dentro de un sistema, además, esta representación se realiza de forma cronológica, con lo cual se puede evidenciar el tiempo de vida que tiene un objeto dentro de la funcionalidad que se esté construyendo. Al igual que en el diagrama de casos de uso, el diagrama de secuencias también cuenta con su notación para identificar a cada uno de sus elementos, pero en este caso son más, debido a la necesidad de obtener una representación

cronológica, mostrar los eventos internos y externos del sistema, las respuestas de cada objeto y sus relaciones (Microsoft 365 Team, 2019).

A continuación, se muestran los diagramas de secuencias de cada caso de uso propuesto anteriormente.

Figura 5

Secuencia para crear paciente

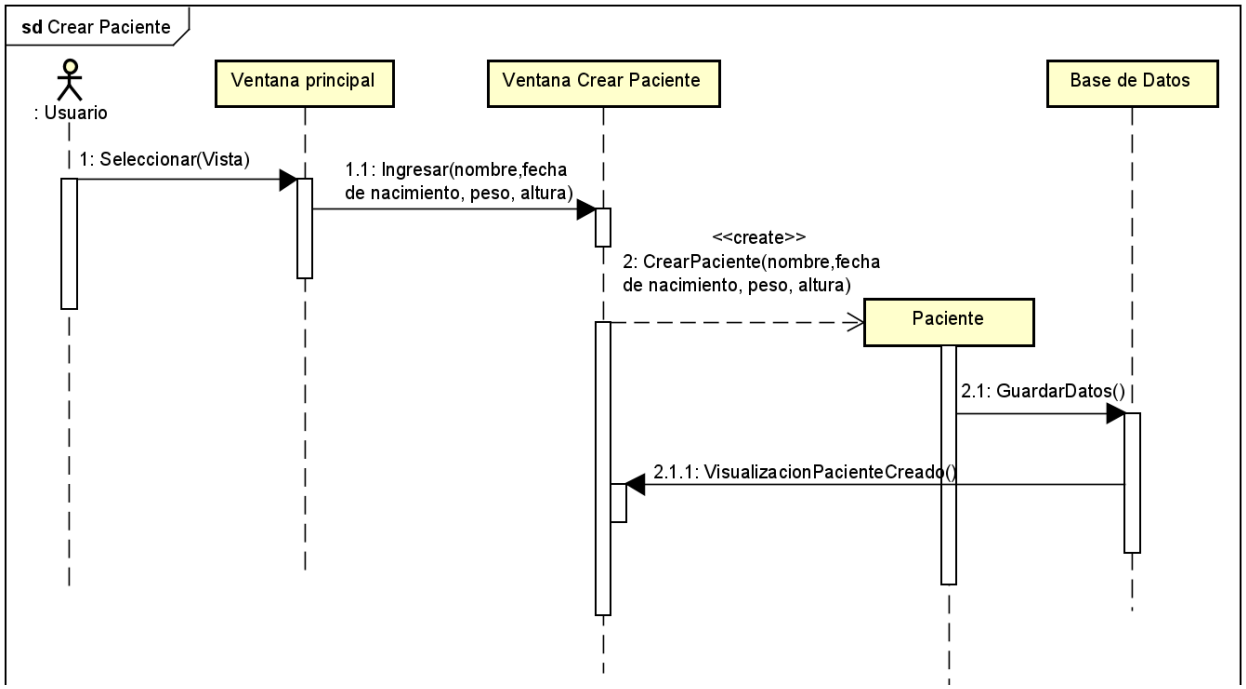


Figura *6*

Secuencia para eliminar paciente

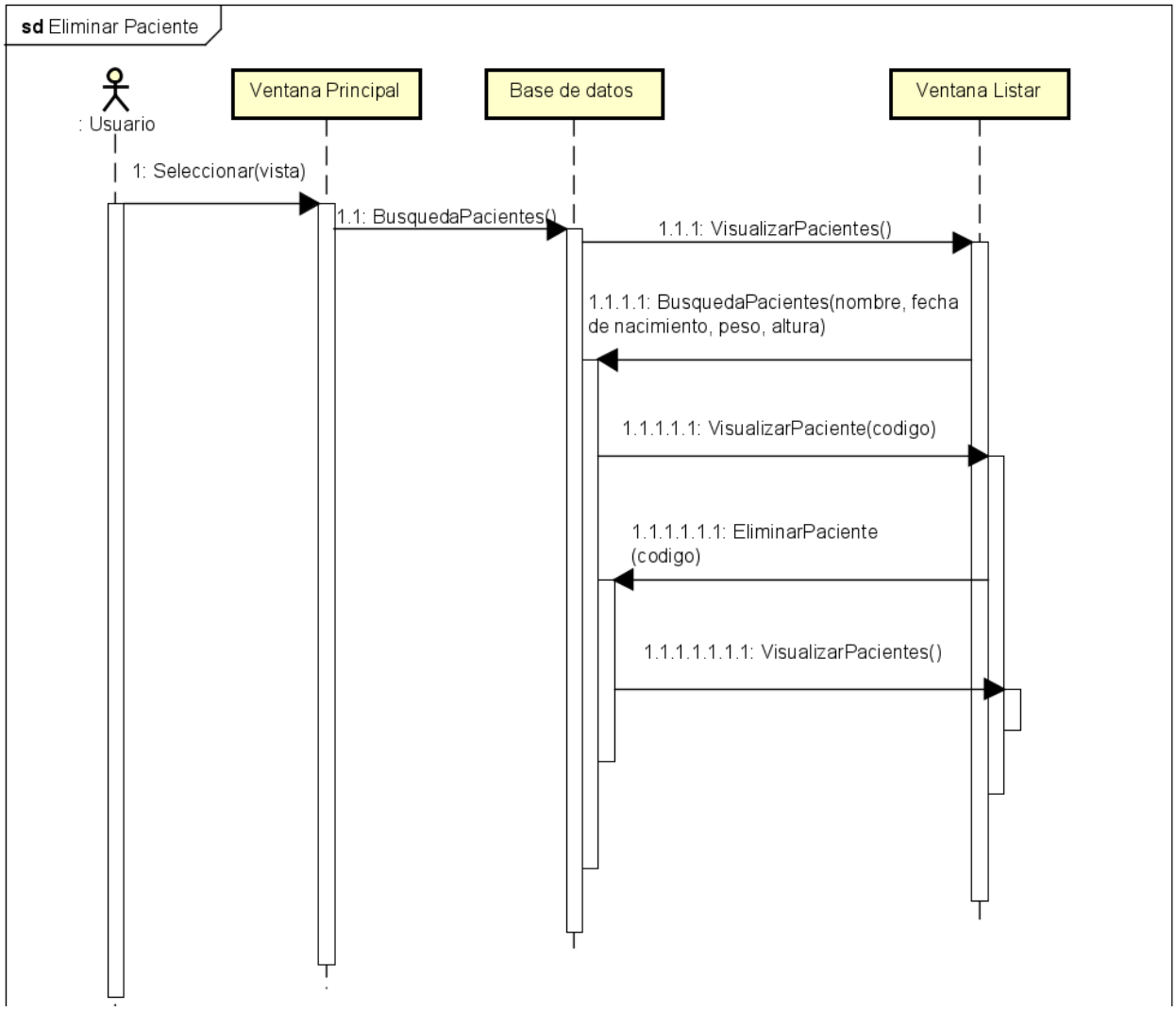


Figura *7*

Secuencia para listar pacientes

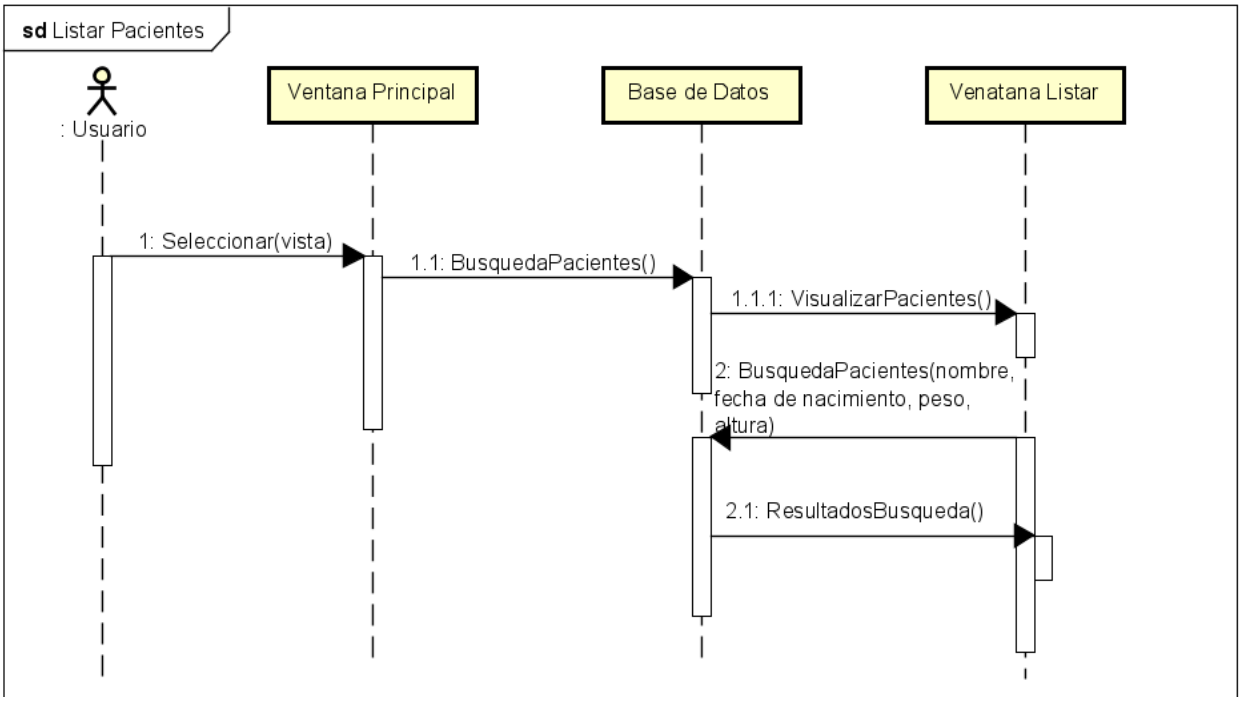


Figura 8

Secuencia para actualizar paciente

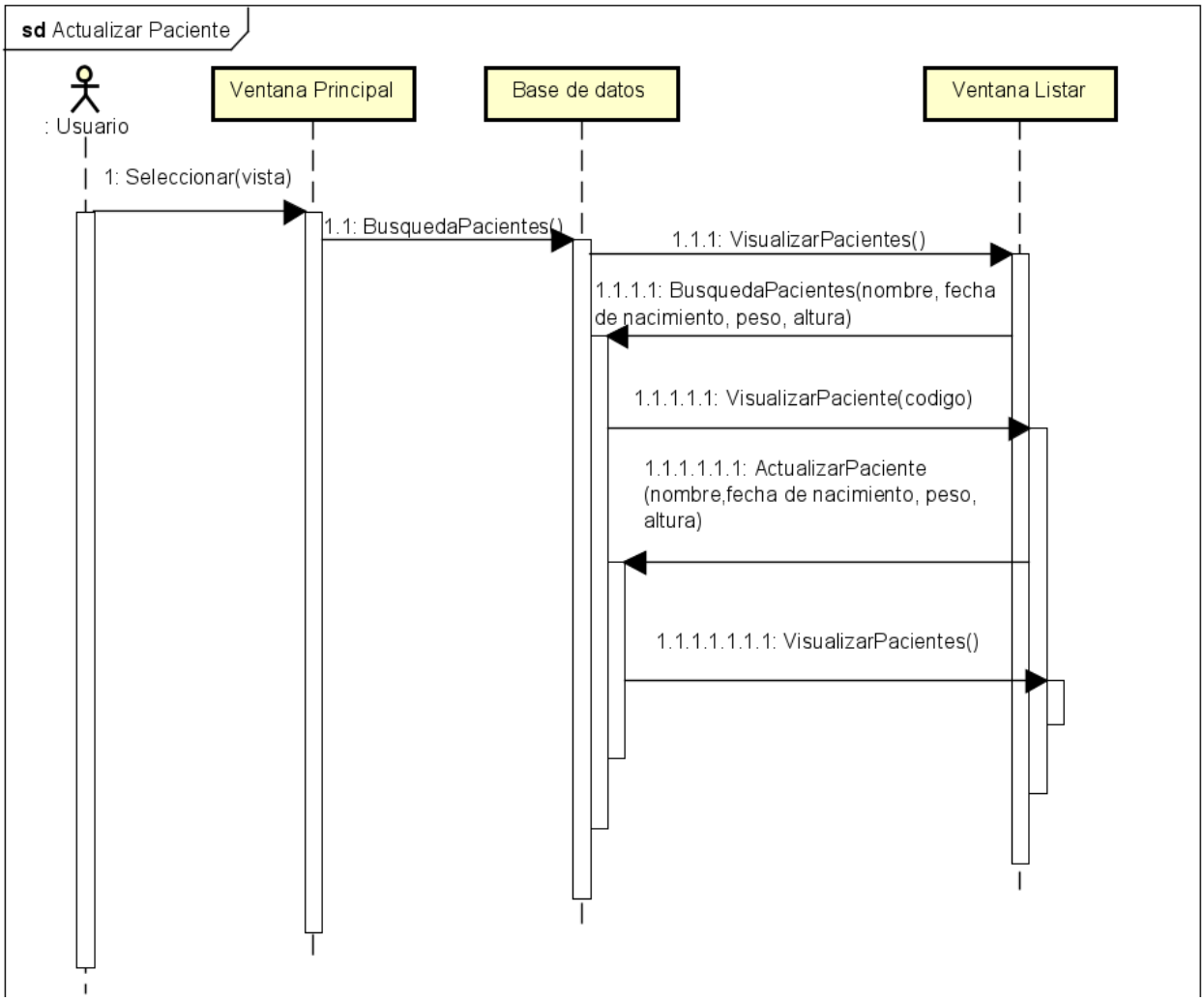


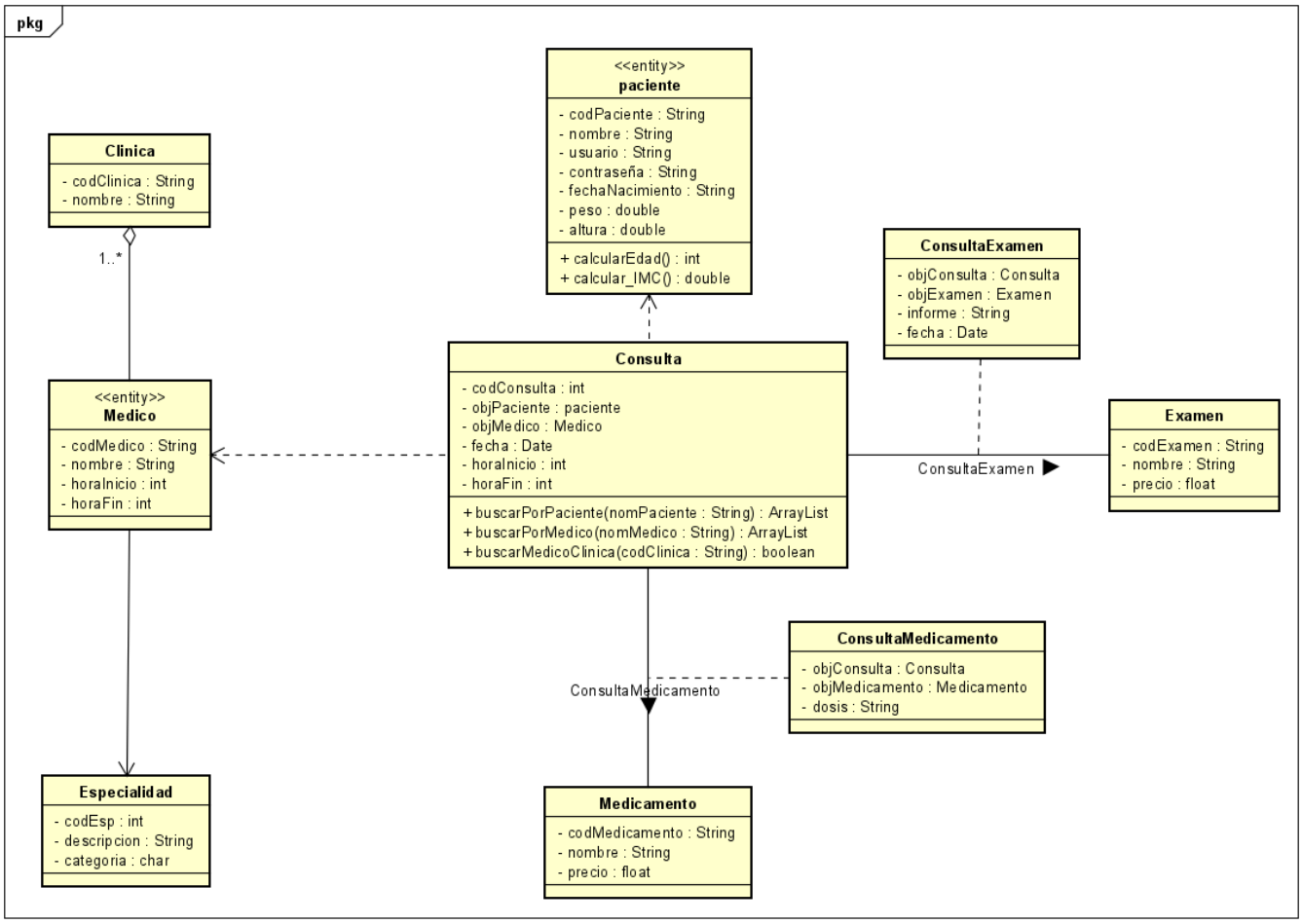
Diagrama de Clases

El diagrama de clases es una representación orientada a objetos de la estructura de un sistema, debido a esto, su uso es muy común durante las etapas de análisis y diseño de software. Para lograr esta representación estructural del sistema, se utilizan clases, las cuales contienen el nombre de la clase, atributos y los métodos necesarios y al igual que en los diagramas vistos anteriormente, este diagrama también cuenta con diferentes tipos de relaciones que ayudan a comprender de mejor manera esa dependencia que puede existir

entre las clases (Microsoft 365 Team, 2019). A continuación, se muestra el diagrama de clases del consultorio médico.

Figura 9

Diagrama de Clases



IFML

El Lenguaje de Modelado de Flujo de Interacción sirve para describir las interfaces de aplicaciones, independientemente de la plataforma en la que se construya y del dispositivo para el que esté orientado. Para realizar esto se deben tomar en cuenta varios aspectos como la inclusión de la estructura y contenido de la vista, los eventos que pueden existir dentro del

sistema y las transiciones que se dan y los parámetros adoptados en base a los métodos de entrada y salida (Torres et al., 2018).

IFML fue presentado como un estándar de la OMG en el año 2013, y al igual que con otros de sus estándares, se han creado varias herramientas que ayuden a realizar el proceso que se describe en cada uno de ellos. Específicamente para el estándar IFML existen varias herramientas que ayudan a cumplir su objetivo, entre las más importantes se tiene a WebRatio, IFMLEdit.org y el complemento de IFML para el IDE Eclipse, de entre estas herramientas la más destacable es WebRatio debido a la cantidad de elementos que cumplen con el estándar IFML, elementos extendidos y de manejo de eventos, y a su capacidad de generar código a partir de los modelos creados (Torres et al., 2018).

En el caso de aplicaciones web, WebRatio Platform permite usar el lenguaje IFML para la creación del front-end de una aplicación, disminuyendo los costos y tiempo, y a su vez se puede construir los sistemas para los diferentes dispositivos en los que se desea ejecutar la aplicación, además de esto WebRatio cuenta con herramientas y elementos específicos dependiendo del tipo de aplicación que se va a implementar, es por esto que en el presente trabajo se hará uso de esta herramienta para la creación de las interfaces de usuario (*IFML General Overview*, s. f.).

Diagramas de los modelos creados en la herramienta WebRatio

Para la creación de los modelos necesarios de la gestión de cada una de las entidades para el sistema de consultas médicas se utilizaron diferentes elementos para la representación de páginas, mensajes, interacciones (botones) y validaciones.

Figura 10

Diagrama de Gestión de Consultas Médicas

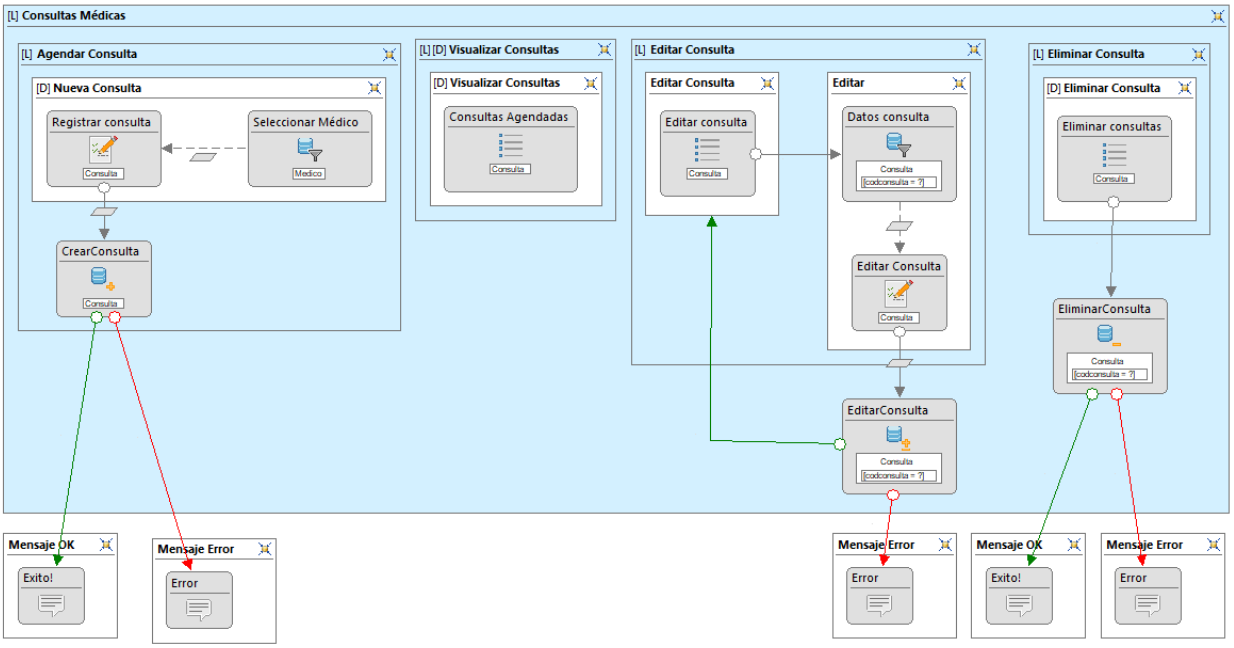


Figura 11

Diagrama de Gestión de Clínicas

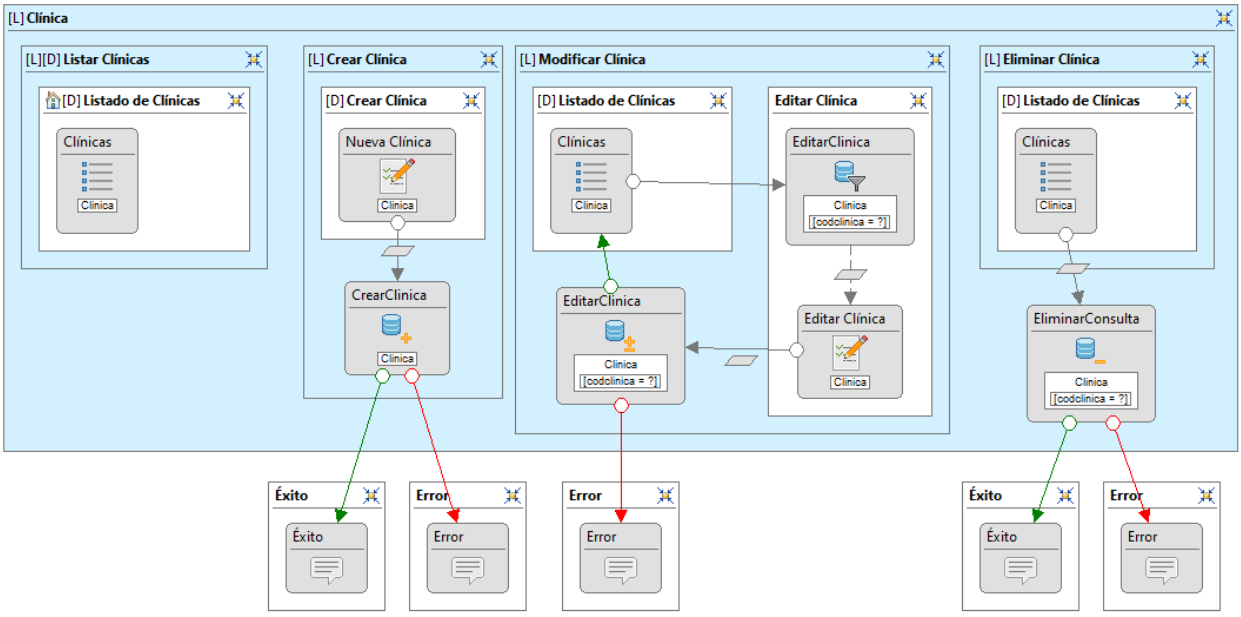


Figura 12

Diagrama de Gestión de Especialidades

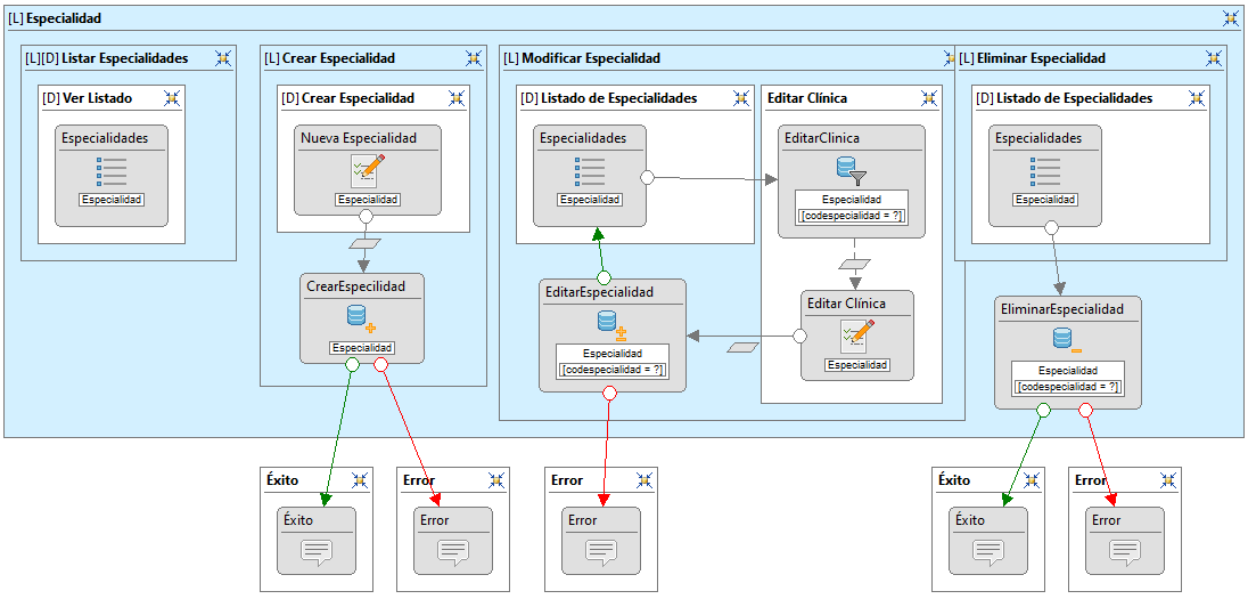


Figura 13

Diagrama de Gestión de Exámenes

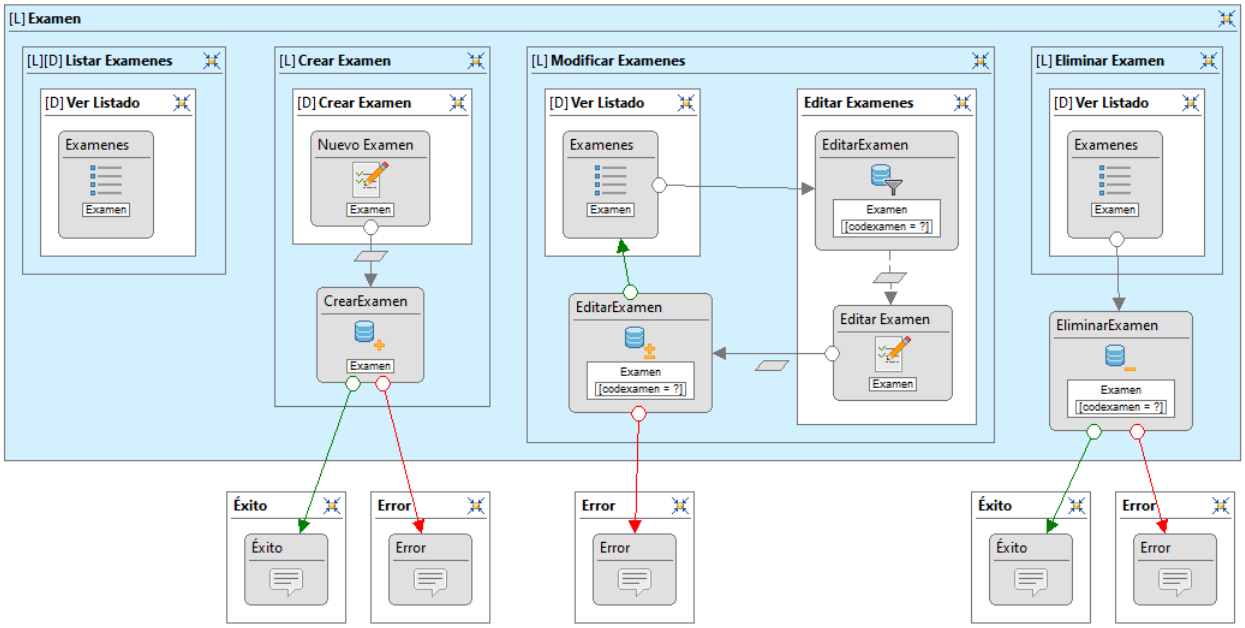


Figura 14

Diagrama de Gestión de Medicamentos

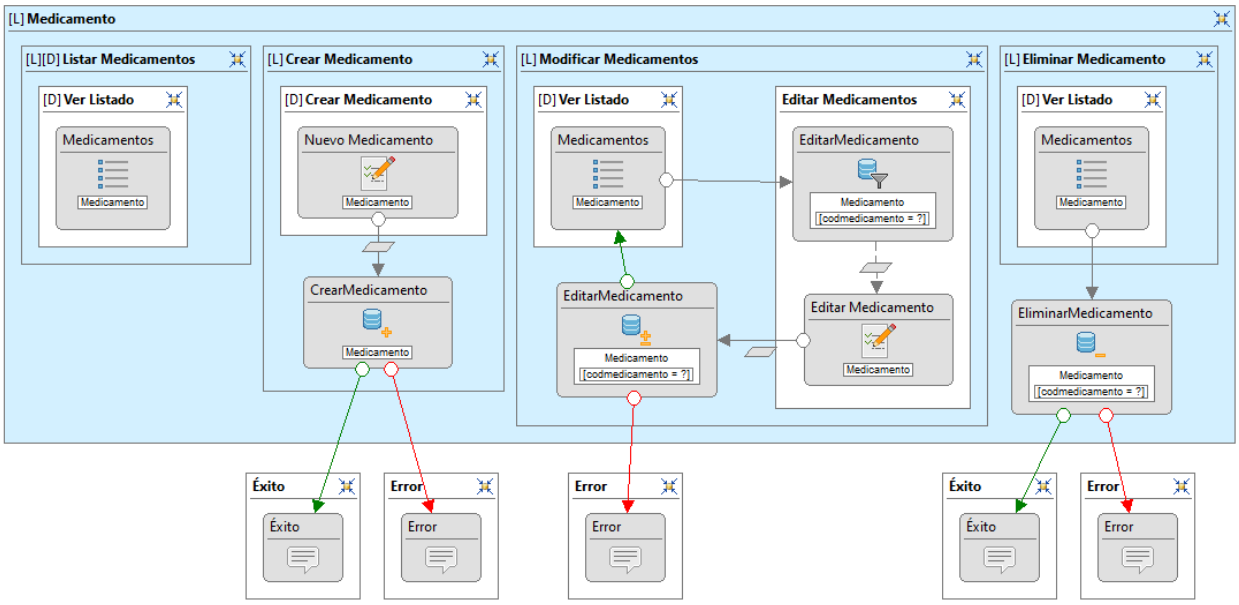


Figura 15

Diagrama de Gestión de Médicos

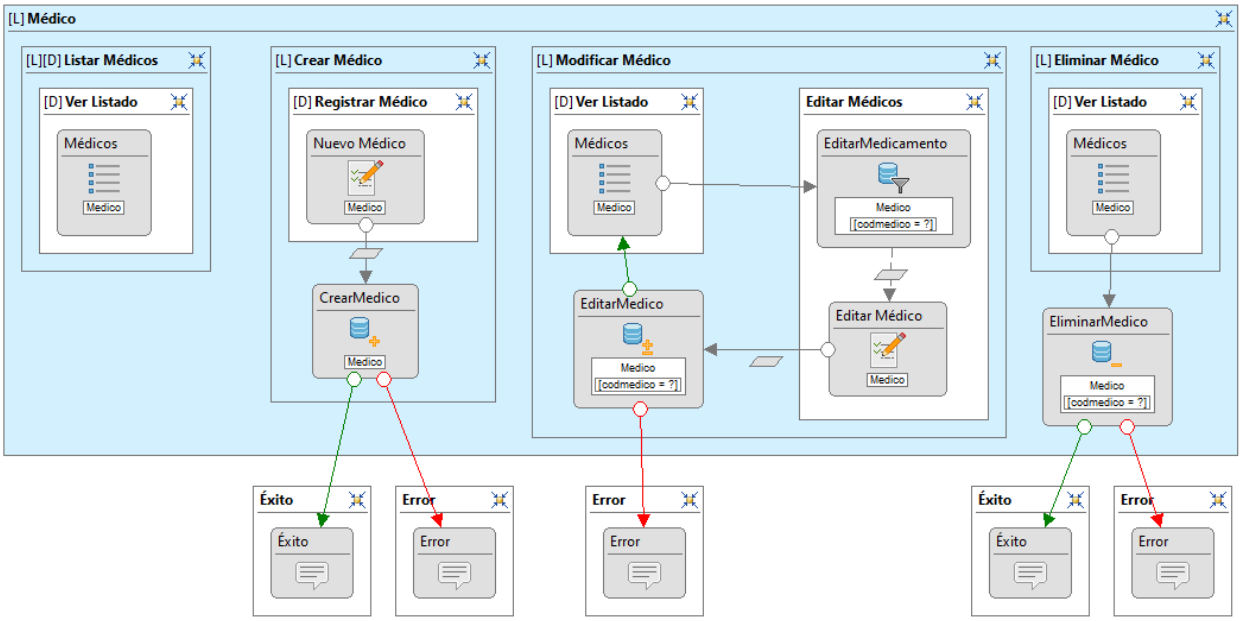
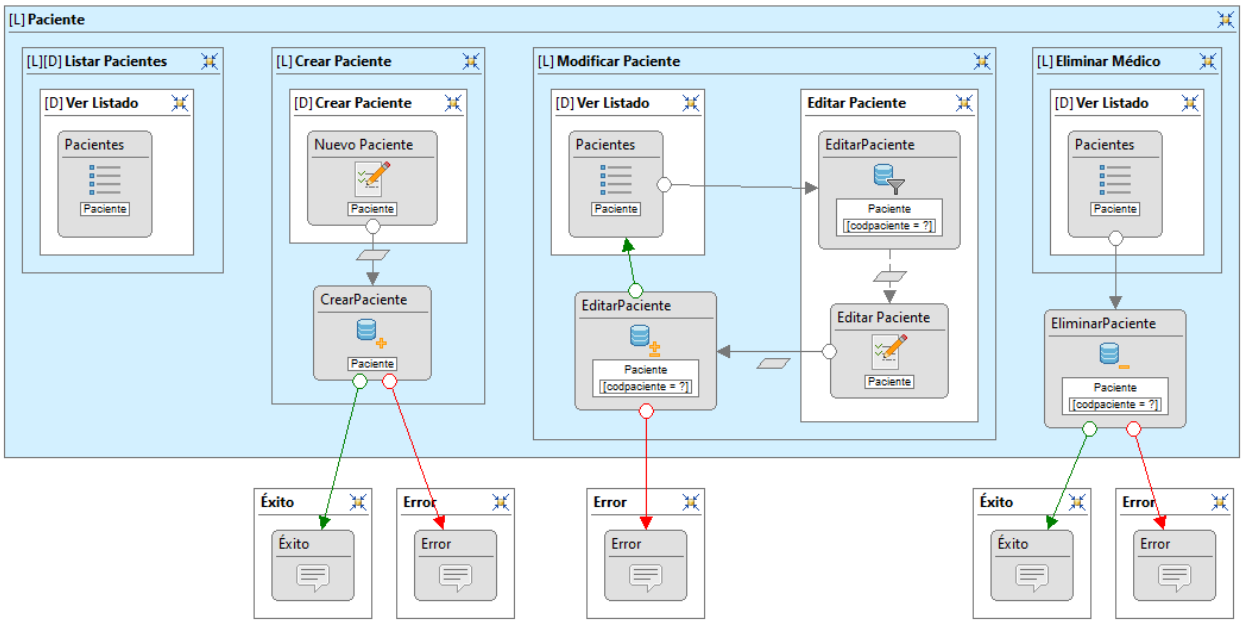


Figura 16

Diagrama de Gestión de Pacientes



Aplicativo generado con la herramienta WebRatio

Otra ventaja importante en la generación de interfaces con WebRatio permite la descarga de diferentes layouts, servicios y componentes en la tienda de la aplicación, lo cual permite cambiar rápidamente el estilo de todos los componentes que se hayan seleccionado, algunos de estos layouts son Bootstrap, AdminLTE, entre otros. A continuación, se expone un ejemplo de las pantallas para la realización de operaciones CRUD de los pacientes, tomar en cuenta que las demás pantallas siguen el mismo modelo.

Figura 17

Pantalla para crear paciente - WebRatio

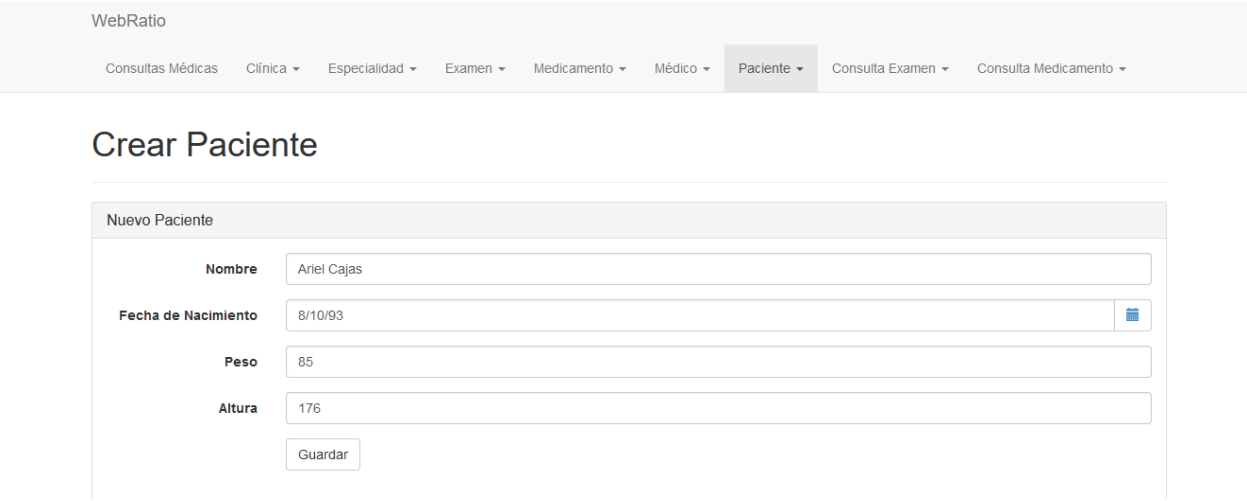


Figura 18

Pantalla para visualizar pacientes - WebRatio

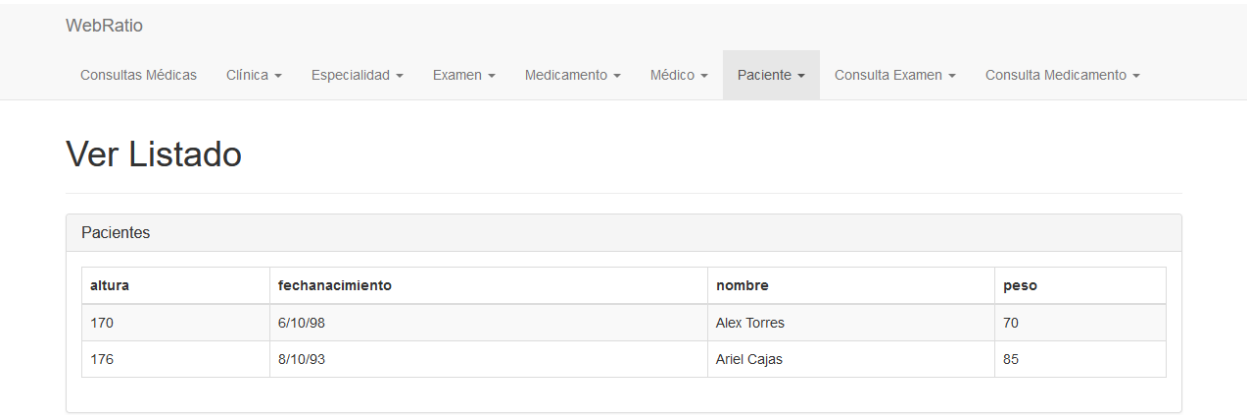


Figura 19

Pantalla para editar paciente - WebRatio

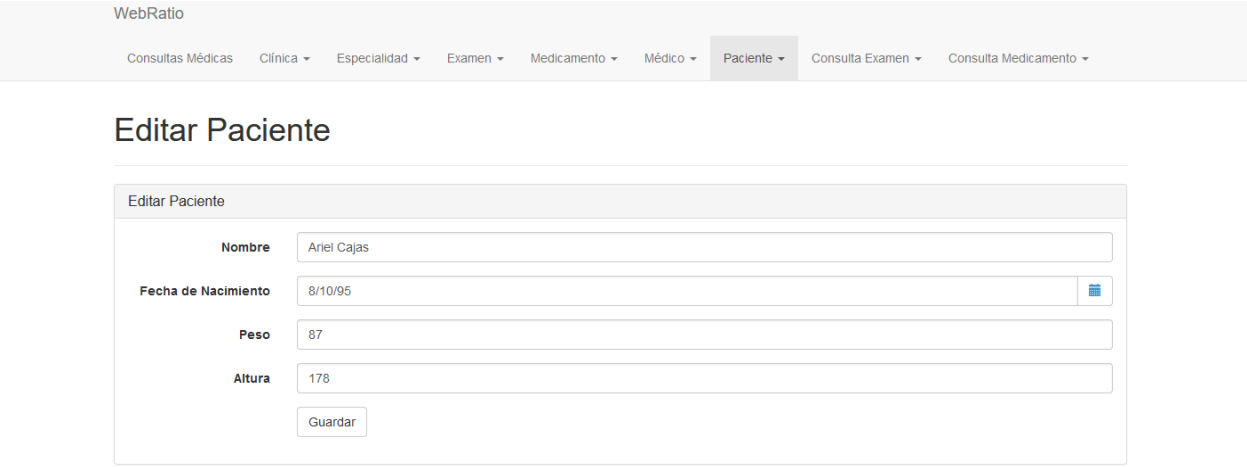
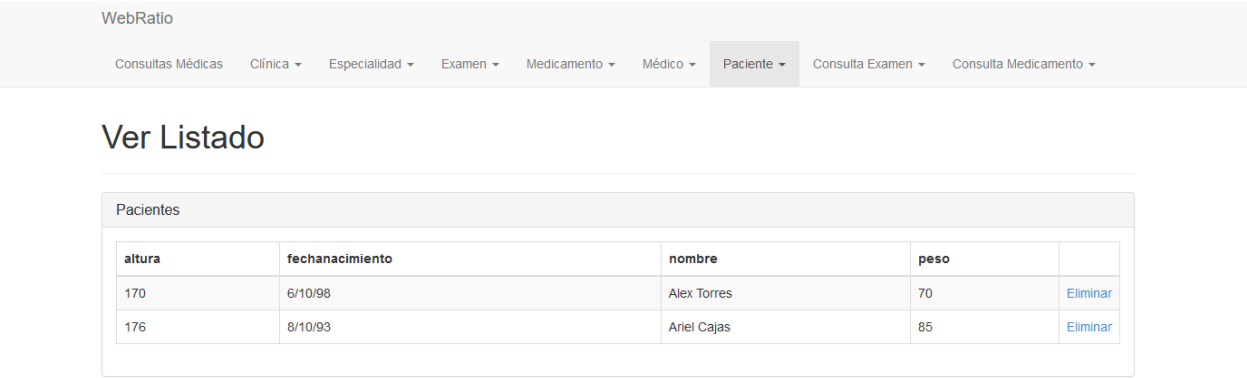


Figura 20

Pantalla para eliminar paciente - WebRatio



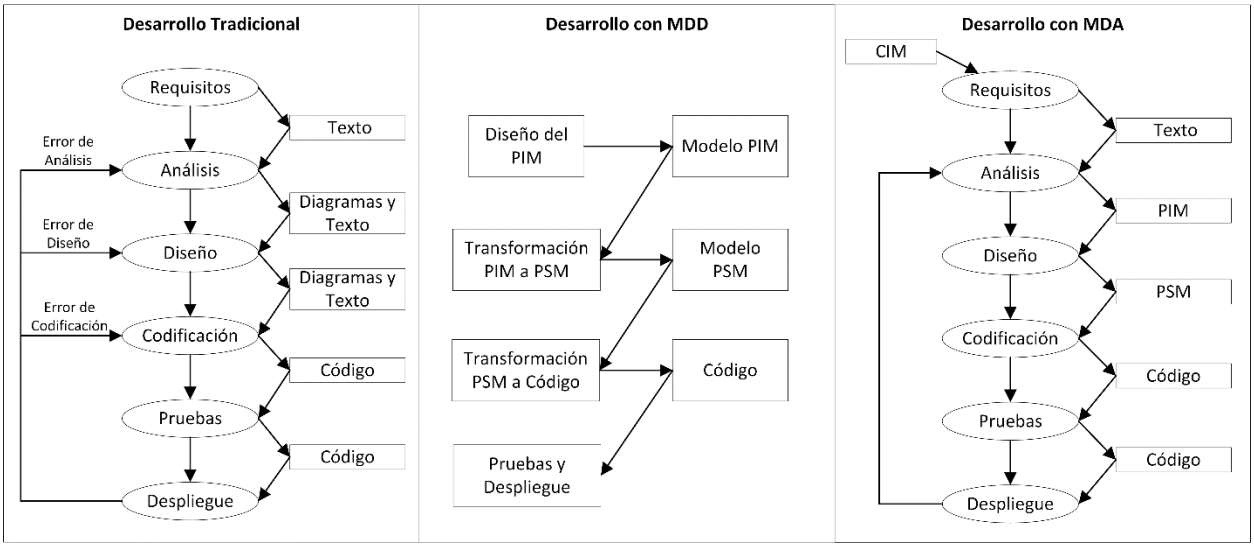
Capítulo 4: Diseño

Modelos de transformación

Los modelos de transformación global son esenciales en la ingeniería de software, facilitando la evolución de CIM a PIM a PSM. Estos modelos permiten la transición de conceptos abstractos a detalles técnicos, primero a nivel independiente de la plataforma y luego a una implementación específica, asegurando una transición fluida y ordenada en el desarrollo de software.

Figura 21

Pasos en el proceso de desarrollo Tradicional, MDD y MDA



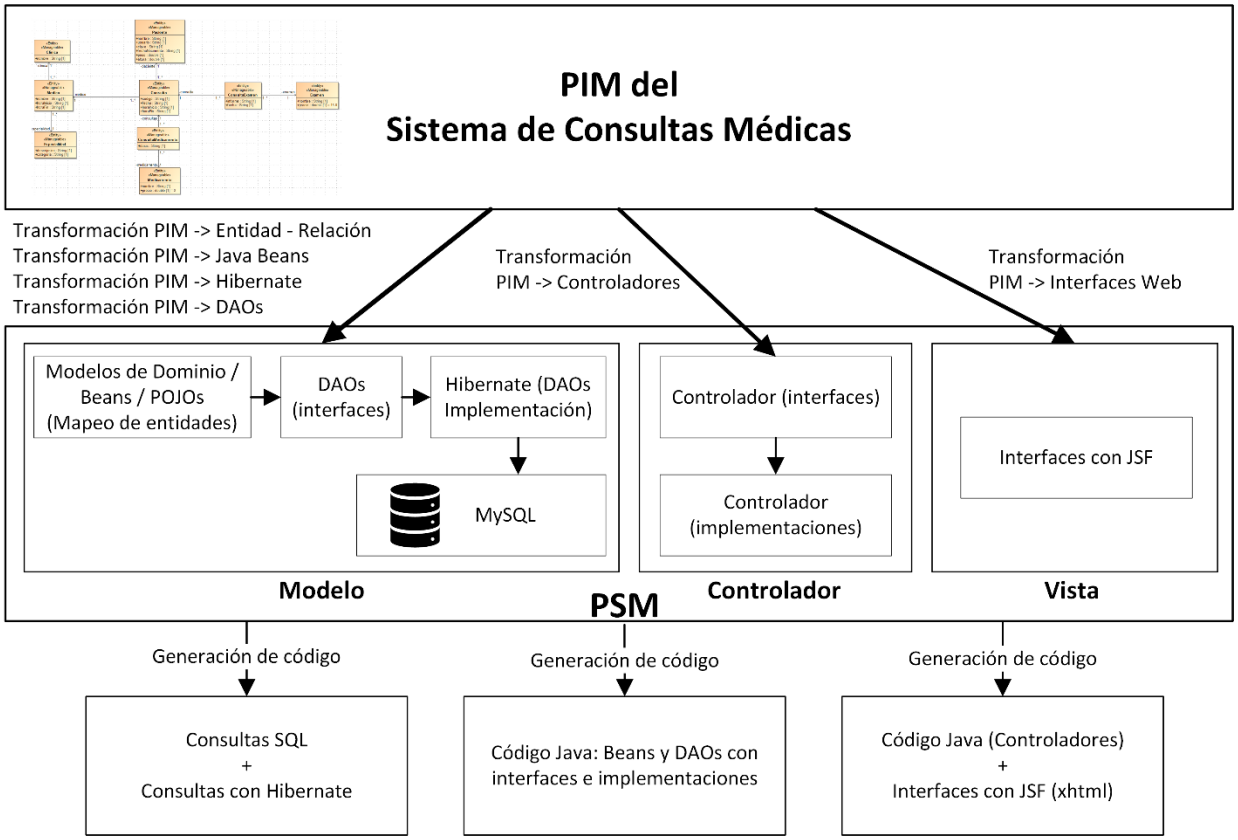
CIM a PIM

Para obtener los diferentes modelos de manera correcta siguiendo el proceso de desarrollo de aplicaciones con MDA se debe pasar por un proceso de transformación, el cual debe permitir obtener el siguiente modelo a partir del anterior, específicamente, en este caso pasar de los modelos CIM a los modelos PIM, lo cual representa pasar de algo general a un nivel más específico y donde se representen todos los requerimientos del sistema. Este proceso suele no incluirse dentro de las herramientas que implementan MDD o MDA, debido a

que al ser CIM modelos muy abstractos y generales es más difícil su transformación. Pero en el caso de ser implementado lo que se platea es el uso de metodologías para el modelado CIM, con lo cual se definen las reglas de transformación para obtener los modelos PIM (Li et al., 2019).

Figura 22

Modelos y transformación del sistema del Consultorio Médico



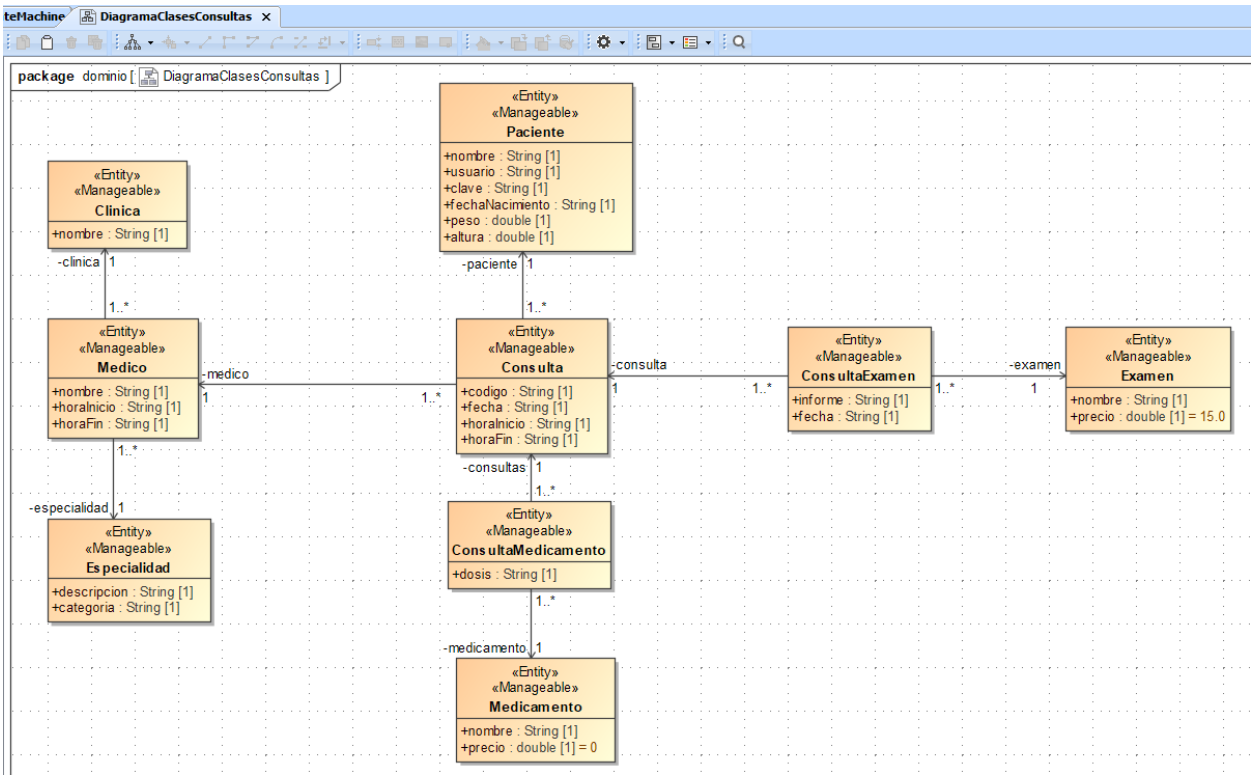
PIM a PSM

Esta transformación es la más utilizada tanto en herramientas MDD como MDA, ya que se utiliza como punto de partida modelos PIM como: diagrama de casos de uso, diagrama de secuencias y diagrama de clases, mismos que se han presentado anteriormente. Al utilizar una herramienta específica para la creación de estos diagramas se cambia ligeramente la notación con la que se crea cada uno de ellos para especificar los tipos de datos, multiplicidades,

relaciones entre las clases, etc. Esto con el objetivo de aplicar las reglas de transformación que establece la herramienta utilizada, por ejemplo en el caso de transformar una clase a los modelos PSM, la herramienta tiene establecido previamente que lo que generará es una tabla en el modelo entidad relación, una clase y controlador o servicio dentro de la aplicación y también una vista que muestre y permitía manipular todos los atributos de esa clase, para el caso de las relaciones, se generan las claves primarias y foráneas, además de si son obligatorias o no de acuerdo a la dependencia que se haya definido en el diagrama de clases, en las clases y controladores se crearán los campos correspondientes junto con las definiciones necesarias para manejar las relaciones y dentro de las vistas, se puede representar con una nueva vista o con un selector, esto dependiendo de la especificación de la relación y de la herramienta utilizada (Pons et al., 2010).

Figura 23

PIM – Diagrama de clases del Consultorio Médico

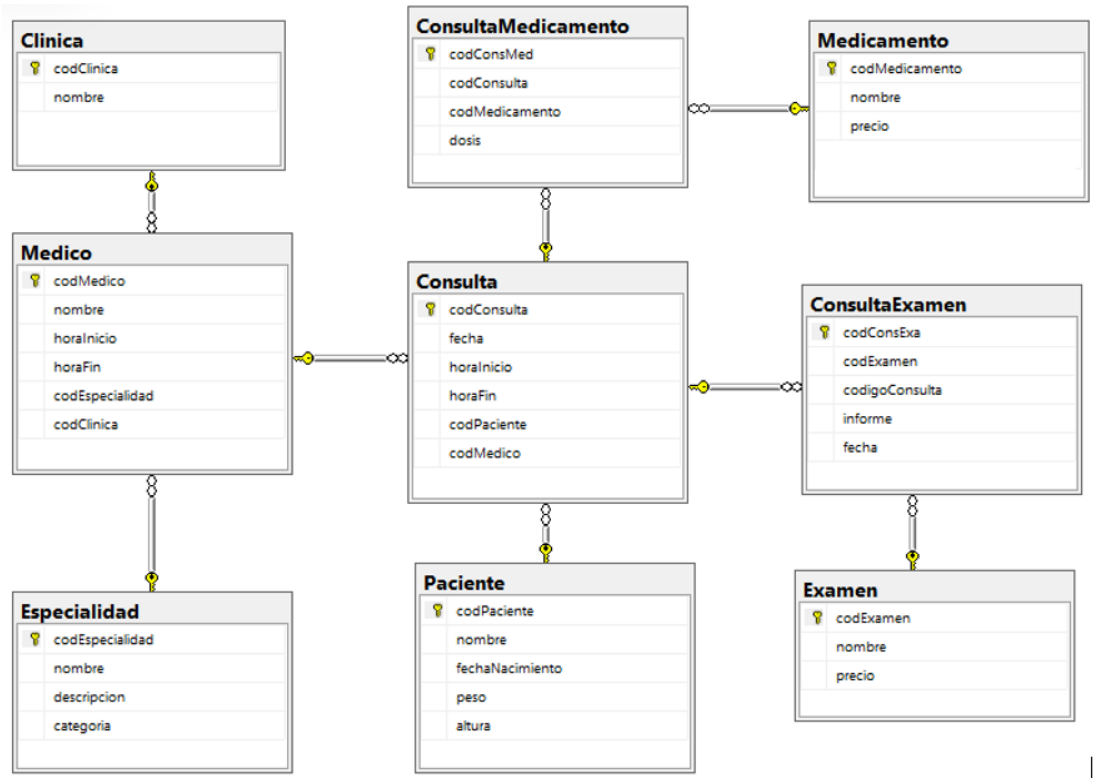


Xomega – Herramienta MDD

Las técnicas más utilizadas en programación son C# y .NET, las cuales han buscado integrar las tecnologías MDD en el espacio de .NET a través del proyecto Xomega. Este proyecto permite crear aplicaciones mediante operaciones CRUD utilizando una base de datos o un modelo de objetos. Además, Xomega es compatible con diversos entornos relacionados con .NET, como WPF o ASP.NET.(Kalnins et al., 2016). SQL Server es una opción sólida para trabajar con .NET gracias a su integración nativa, escalabilidad, seguridad y rendimiento, la base de datos usada para el aplicativo se visualiza en la Figura 24.

Figura 24

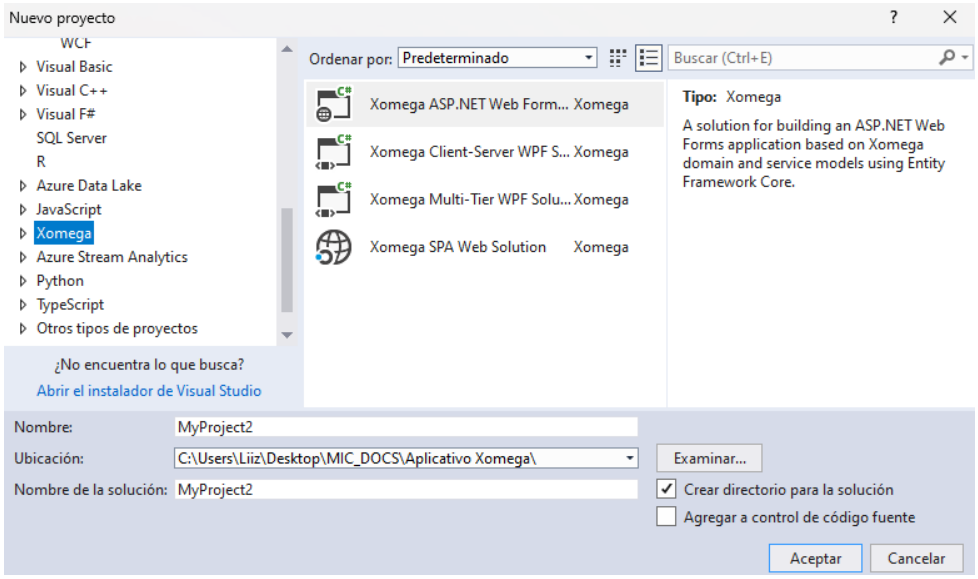
Base de datos SQL Server



Visual Studio 2017 es un IDE desarrollado por Microsoft, que permite el desarrollo de programas en diversos lenguajes de programación, entre los cuales se encuentra C#, en este IDE se creó el proyecto como se puede visualizar en la Figura 25.

Figura 25

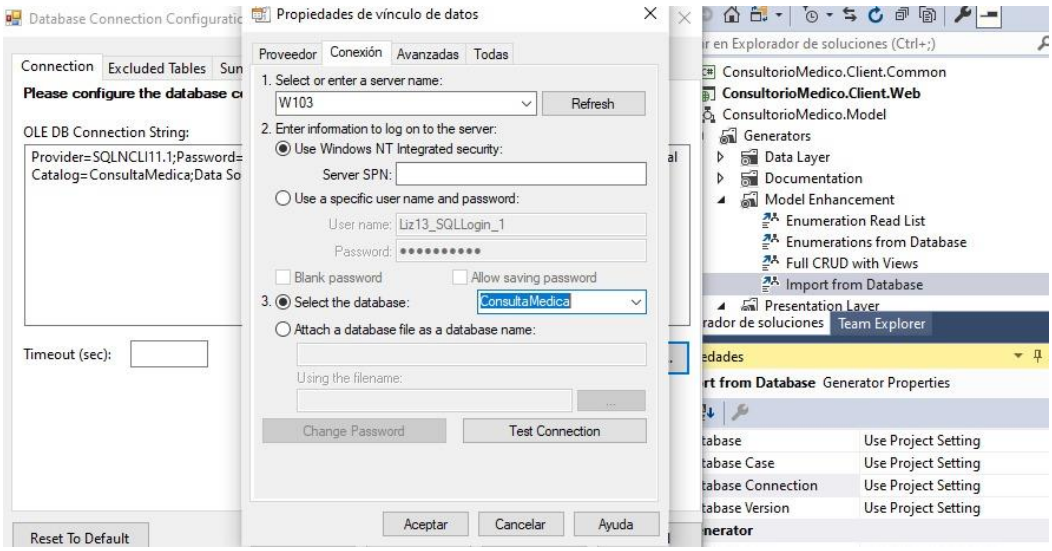
Creación del proyecto Xomega



Realizar la conexión con la base de datos dentro del modelo, mediante la configuración de la conexión de la base de datos.

Figura 26

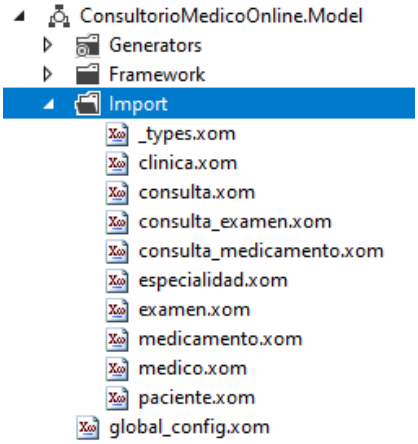
Conexión con la base de datos



Una vez cargada la base de datos importar la base desde el SQL Server tal como se muestra en la Figura 27.

Figura 27

Importación de tablas desde SQL Server



Escoger la tabla de la cual se requiere generar el CRUD y realizar los pasos que se muestran desde la Figura 28 a la 30.

Figura 28

Full CRUD with Views

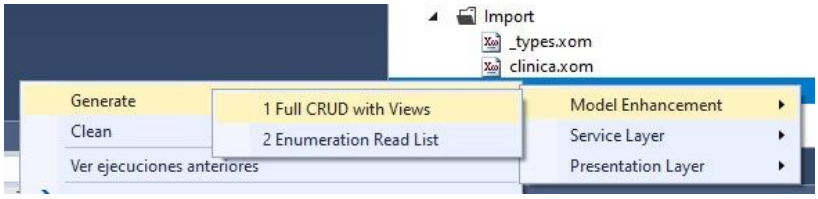


***Figura* 29**

Service Implementations

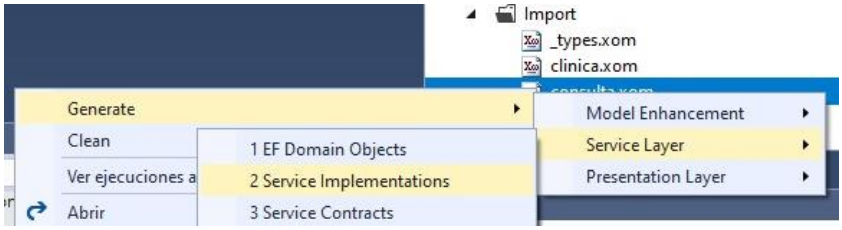
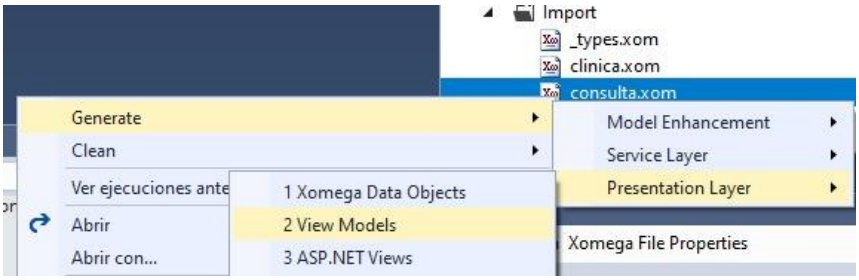


Figura 30

View Models



Finalmente se compila la solución para comprobar que no existan errores y se ejecuta la solución.

Figura 31

Lista de vistas del Consultorio Médico - Xomega

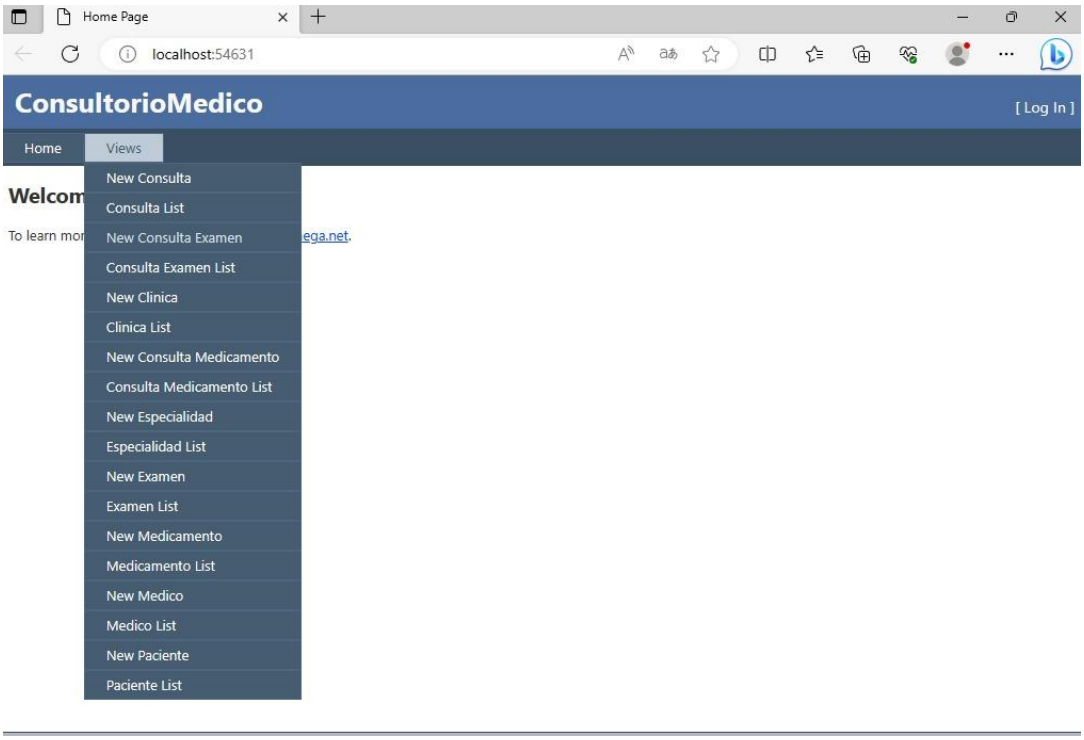


Figura 32

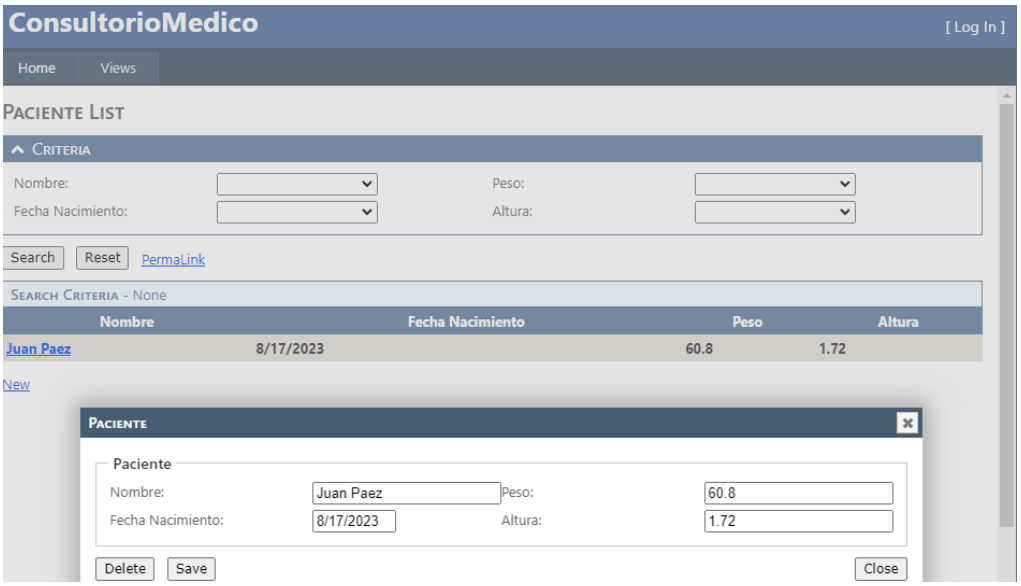
Pantalla crear pacientes - Xomega

Figura 33

Pantalla listar y buscar pacientes - Xomega

Figura 34

Pantalla actualizar y eliminar pacientes - Xomega



AndroMDA – Herramienta MDA

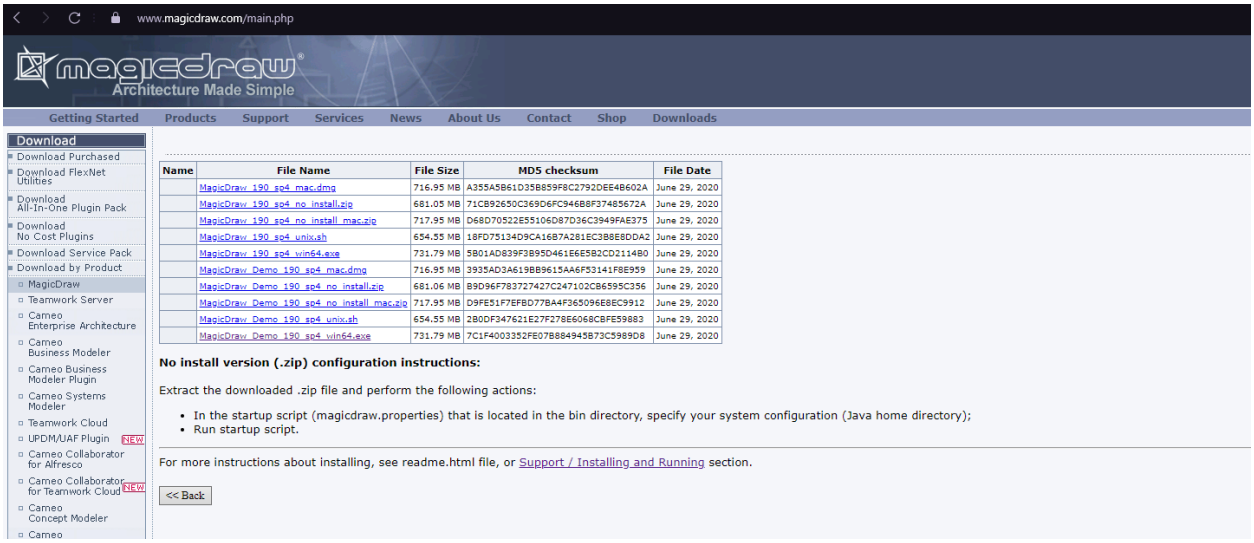
Para el caso de MDA existen diferentes herramientas que pueden ayudar en la generación de aplicaciones con base en esta metodología y como ya se ha mencionado anteriormente, estás metodologías crean sistemas independientemente de las plataformas o lenguajes de programación. Por lo cual, la elección de las herramientas basadas en MDA dependerá de los requerimientos que se deban cumplir en el sistema final. Para MDA, la más utilizada es MagicDraw junto con el framework AndroMDA, debido a que este framework está directamente relacionado con los conceptos especificados para MDA dentro de la OMG, así mismo, los estándares como UML y otras necesarias para la creación de diagramas dentro de MagicDraw están basados en lo que se describe dentro del sitio oficial o de los artículos de la OMG, esta información se toma de acuerdo a la versión de MagicDraw que se utilice y por ende, a la versión de los estándares que se estén considerando.

MagicDraw es una herramienta de Ingeniería de Software Asistida por Computadora o CASE por sus siglas en inglés. Cuenta con una página propia con su documentación, soporte

información y registro para la descarga de plugins, la herramienta MagicDraw o una demo, tal como se muestra en la Figura 35, se puede descargar para Windows, MacOS o sistemas Linux.

Figura 35

Página de descarga para la herramienta CASE MagicDraw



Por otra parte, AndroMDA es un framework que trabaja sobre java, se puede utilizar mediante la descarga de sus dependencias en proyectos generados con la herramienta Maven. Cuenta con su propia página, donde se describe el funcionamiento, versiones y las dependencias necesarias para levantar el entorno necesario para la generación de aplicaciones con AndroMDA, además se destaca la generación de código para Java y C#, diferenciándose únicamente en la configuración para el entorno necesario.

Por lo tanto, es ideal la combinación de MagicDraw y AndroMDA para obtener un ejemplo preciso del funcionamiento de MDA en el desarrollo de aplicaciones, ya que, MagicDraw ayuda a la creación de los modelos y sobre todo tiene la capacidad de generar código a partir del modelo generado, mientras que AndroMDA se complementa, debido a que la creación de aplicaciones incluye también un archivo xml, a partir del cual se puede trabajar

dentro de la herramienta CASE y posteriormente generar la aplicación basada en esos modelos siguiendo las especificaciones de la OMG para MDA.

Figura 36

Configuración inicial del proyecto en AndroMDA

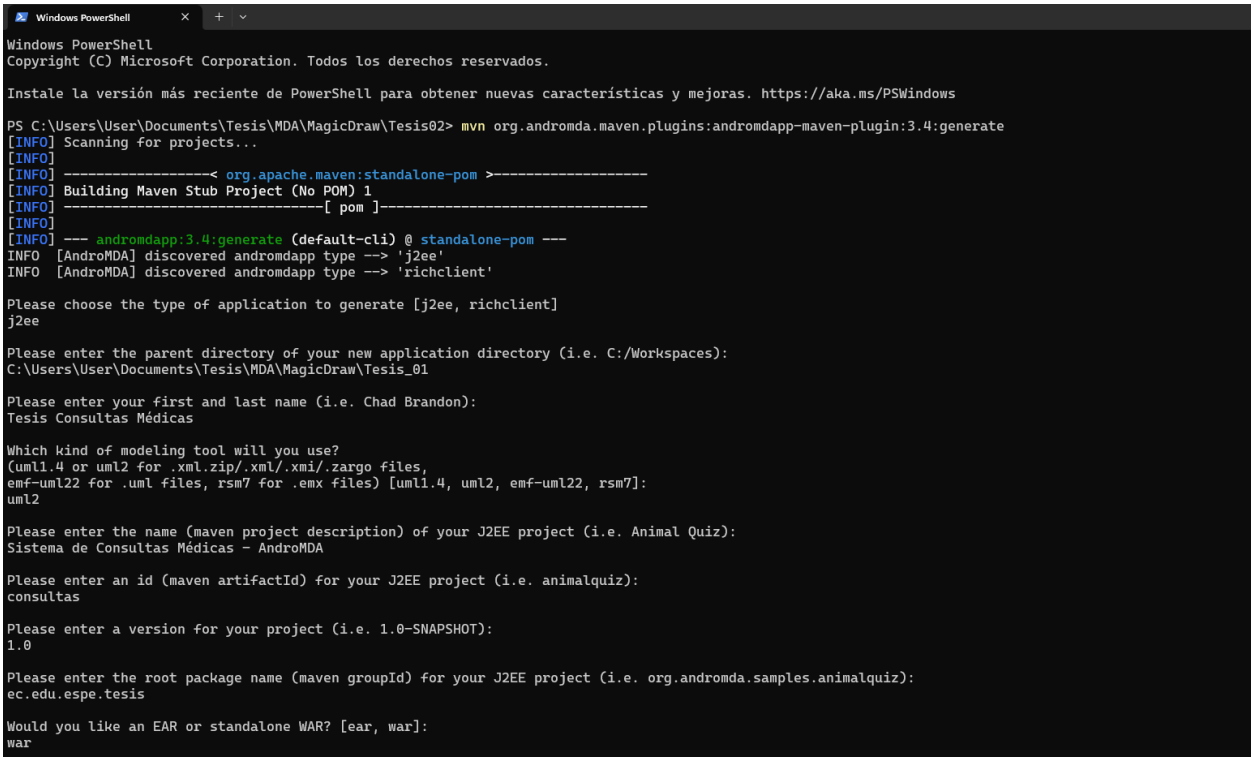


Figura 37

Elección de lenguaje y base de datos para el proyecto

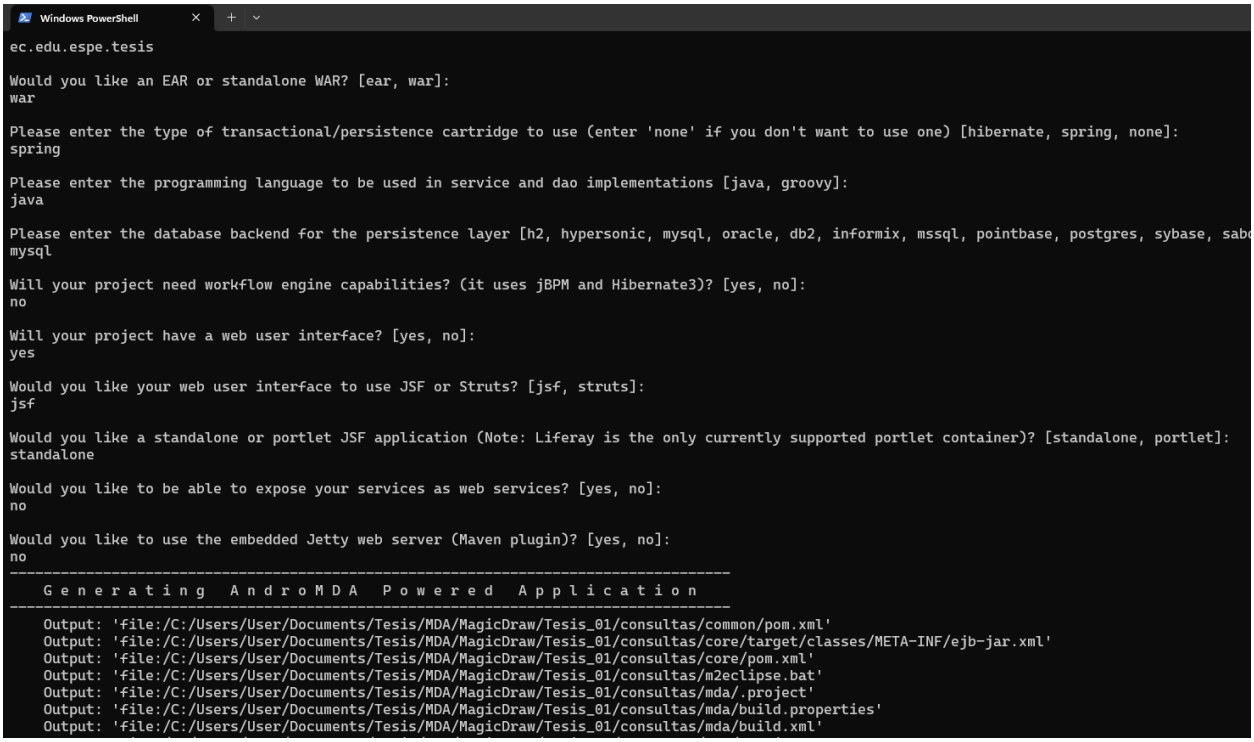


Figura 38

Archivos UML generados con AndroMDA

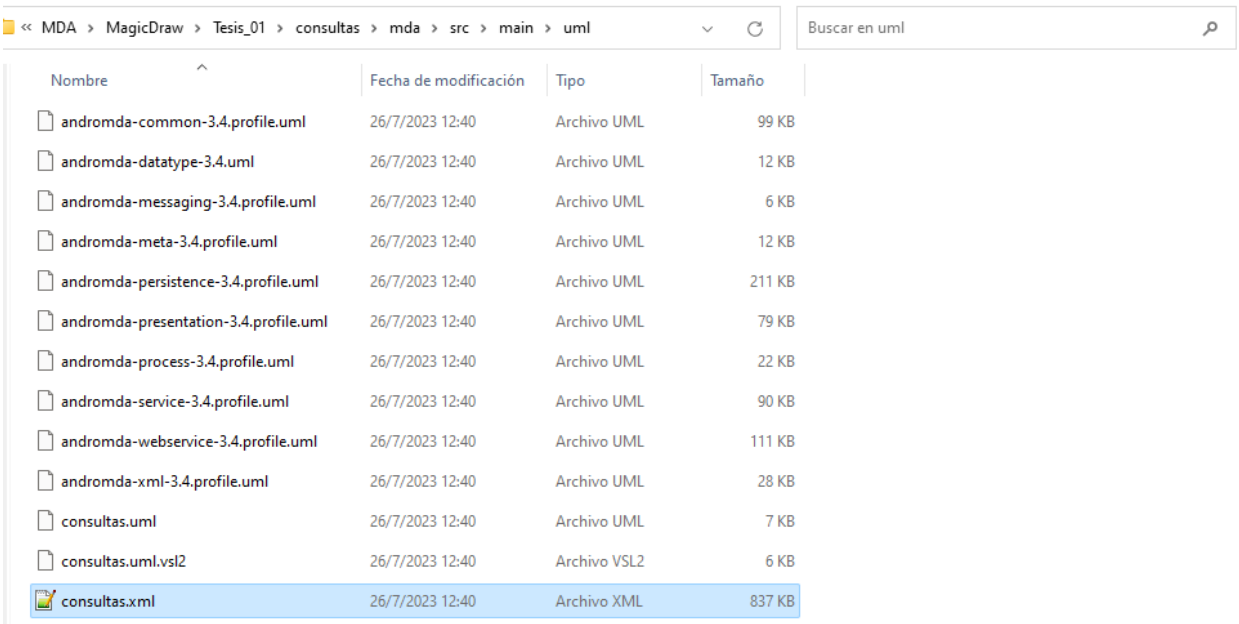


Figura 39

Creación de las clases y atributos necesarios para el modelo

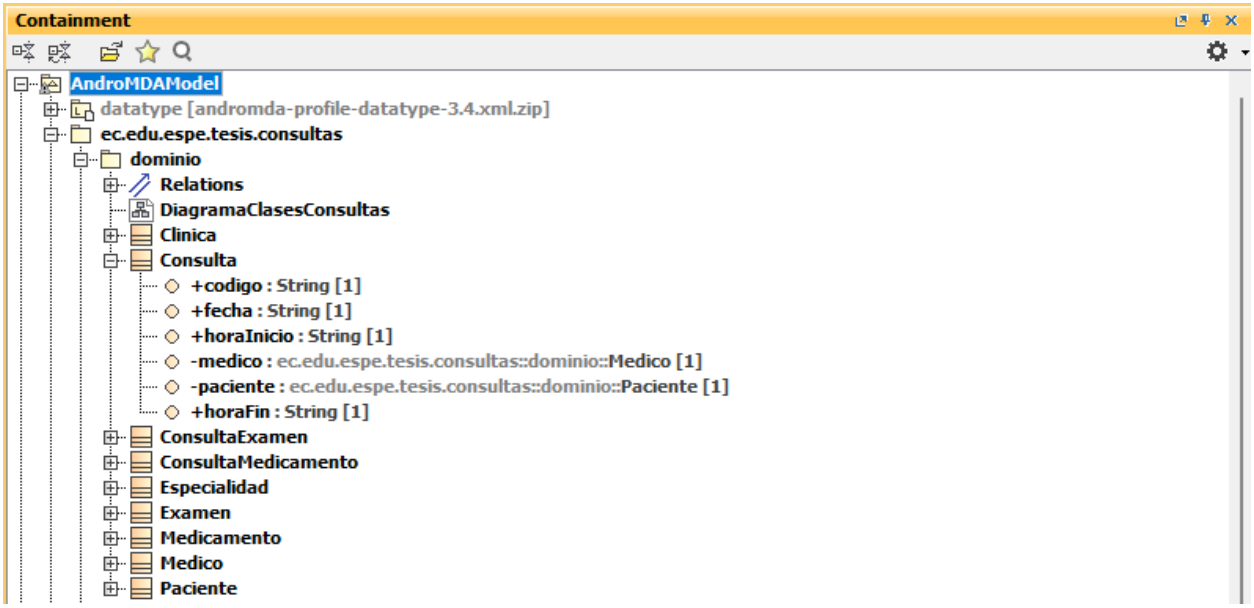


Figura 40

Exportación del modelo y generación de la aplicación.

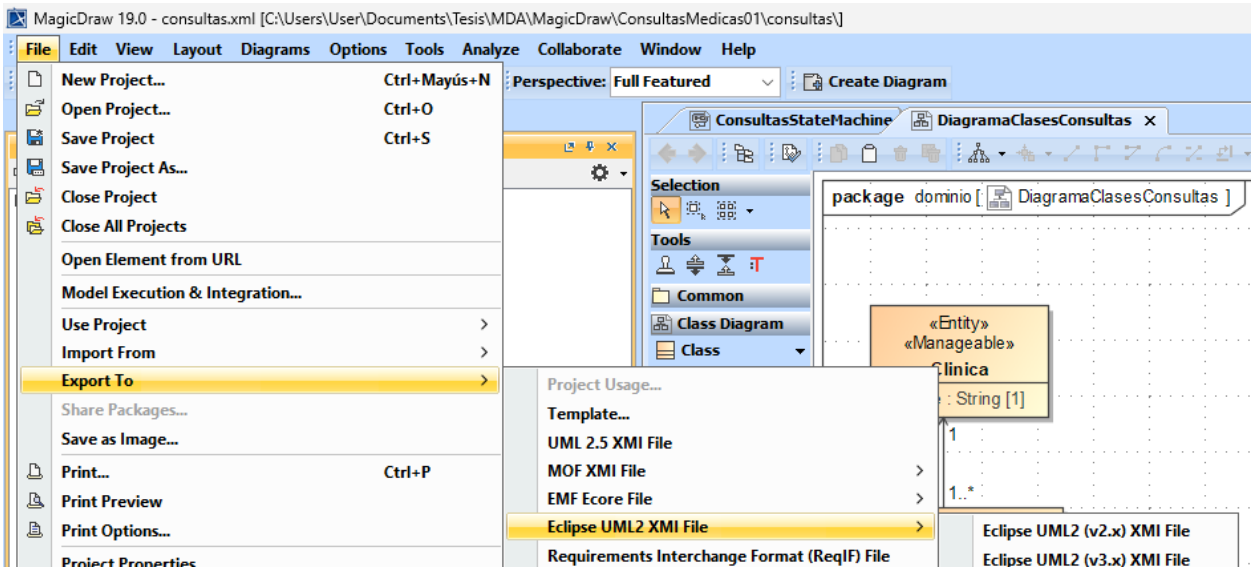


Figura 41

Beans y clases generadas para cada modelo incluido.

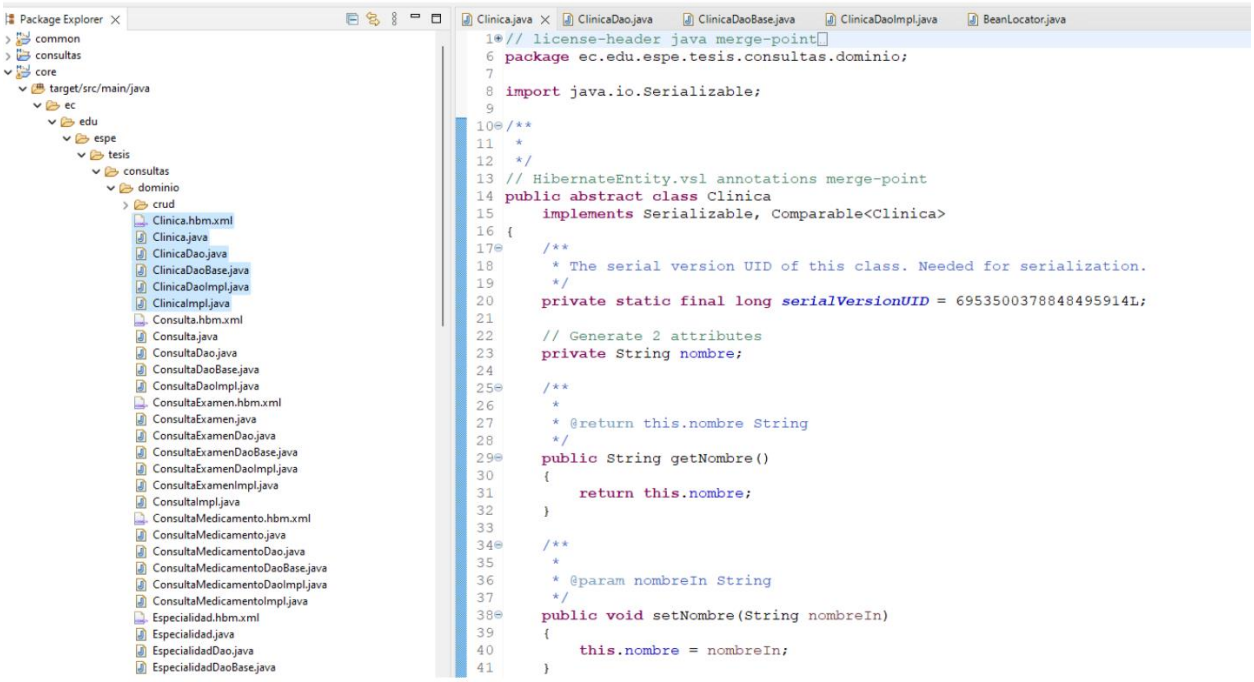


Figura 42

Controladores y archivos JSF para las interfaces del modelo

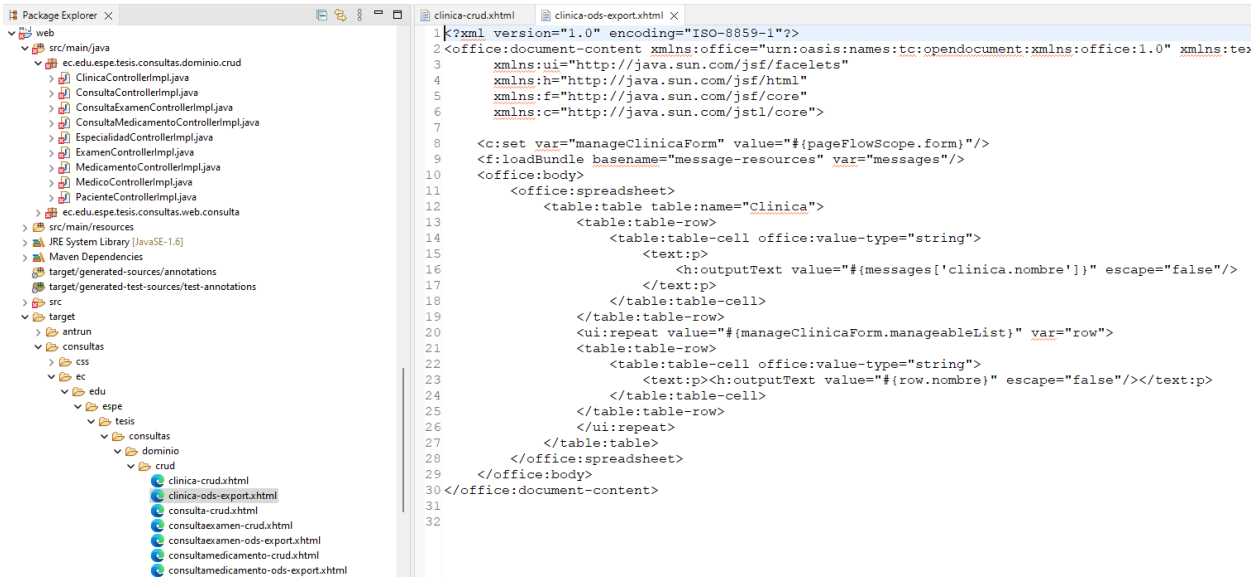
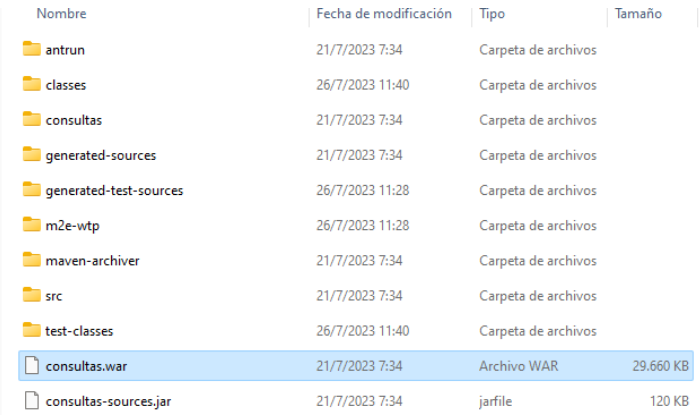


Figura 43

Archivo .war



Nota. Para la obtención del archivo .war se hizo uso del comando nvm para su ejecución en un servidor de aplicaciones

Figura 44

Implementación de la aplicación en el servidor Apache Tomcat.

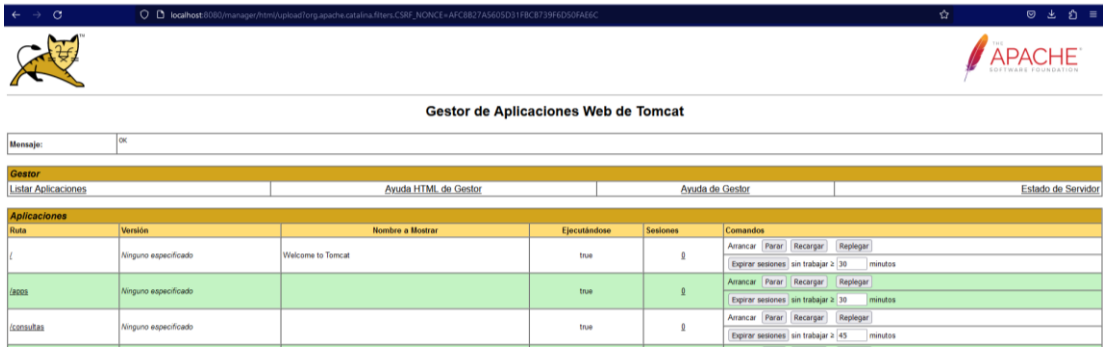


Figura 45

Pantalla listar pacientes - AndroMDA

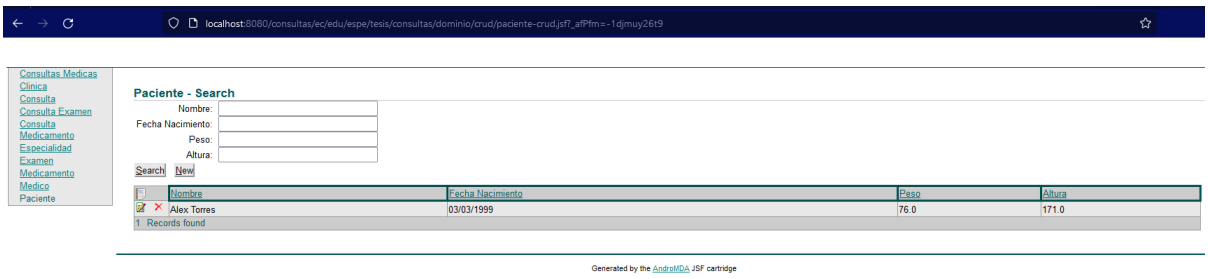


Figura 46

Pantalla crear paciente - AndroMDA

Capítulo 5: Implementación

Configuración de la infraestructura

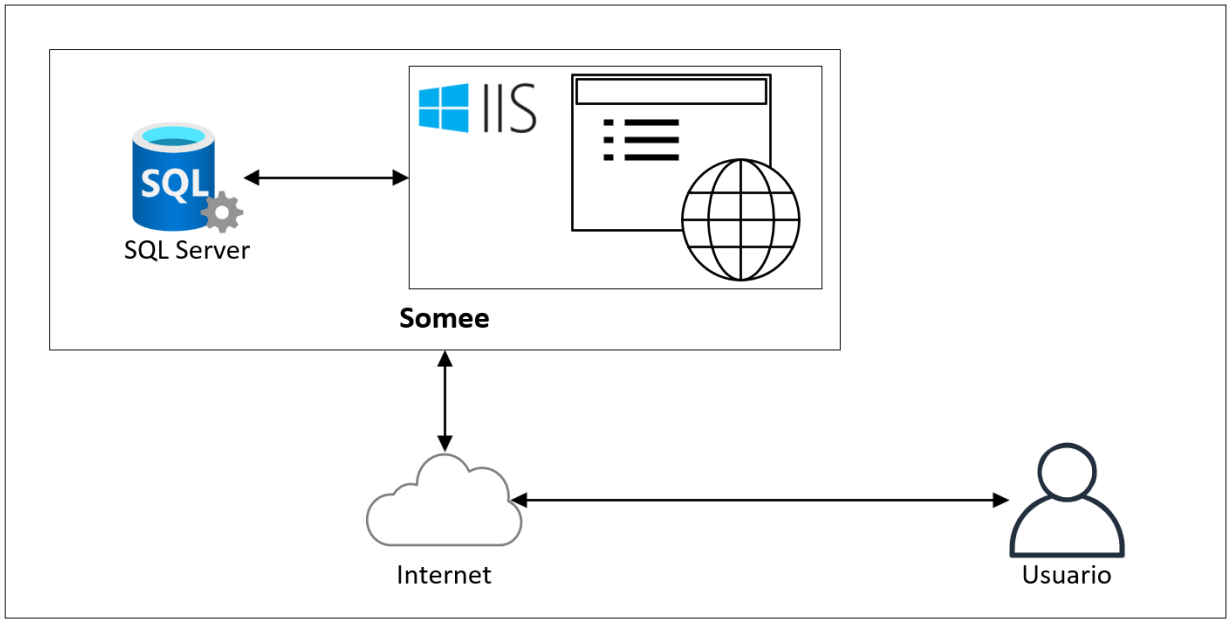
En este capítulo se exponen las configuraciones necesarias de las infraestructuras para alojar y ejecutar los aplicativos generados tanto en MDD como en MDA del Consultorio Médico.

MDD

Posterior a la generación del aplicativo y habiendo revisado la funcionalidad del mismo se procedió a seleccionar las herramientas para poder realizar la implementación del aplicativo, para esto se tomó en cuenta experiencias anteriores, facilidad de integración, presupuesto y disponibilidad de hosting compatibles con el aplicativo generado, como se planteó anteriormente se hizo uso del framework de ASP.NET sobre el ambiente de desarrollo Visual Studio Community 2017, el script se generó en Power Designer y posteriormente se ejecutó en SQL Server 2017.

Figura 47

Arquitectura Interna del Sistema Web



MDA

Una vez generado el proyecto en AndroMDA se procede a construir y empaquetarlo, tal y como se vio durante el proceso de creación del proyecto en la Figura 37, se seleccionó un .war para el tipo de archivo a construirse. Esto implica que para el despliegue se debe contar con la versión de jdk correspondiente y un servidor de aplicaciones como parte del entorno dónde se vaya a desplegar la aplicación. Al tener únicamente estos dos requisitos para su despliegue se vuelve más sencillo la configuración del entorno, por lo cual, para su despliegue se podría utilizar una máquina virtual con un sistema operativo Windows, Linux o Mac OS, incluso se podría hacer con el uso de contenedores. Además, es importante mencionar que la base de datos utilizada es MySQL, lo cual también debe ser considerado para el despliegue de la aplicación.

Despliegue

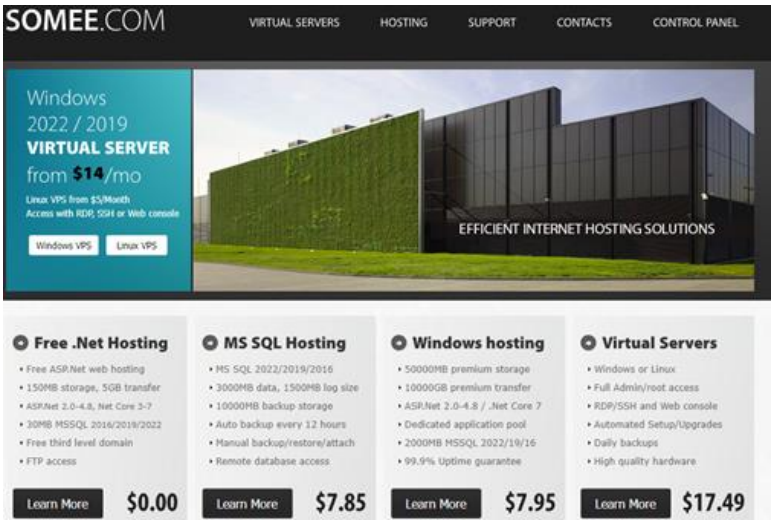
Tal y como se mencionó anteriormente, la aplicación de MDD y MDA han sido creadas para lenguajes de programación y bases de datos diferentes por lo cual requieren de un entorno diferente para cada una de ellas. A continuación, se muestra la configuración de estos entornos y el despliegue de cada una de las aplicaciones.

MDD

Subsiguientemente al desarrollo del proyecto se procede con el hosting y enlazado a un dominio, el hosting seleccionado fue Somme, ya que entre sus características más importantes se cuenta con que permite el alojamiento web de proyectos desarrollados en ASP.Net, almacenamiento de una base de datos MSSQL, además de un dominio de tercer nivel todo de forma gratuita.

Figura 48

Página principal de Somee

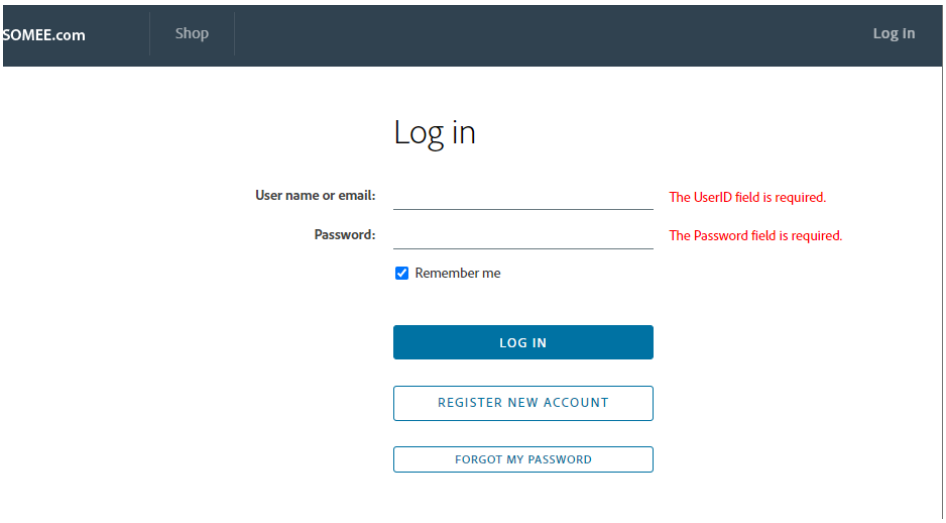


Nota. Se hizo uso de la versión gratis de Somee ya que cuenta con las características necesarias para el deployment el proyecto.

Para el deployment de la aplicación generada en Xomega, se creó una cuenta en la página de Somee.

Figura 49

Login Somee.com



Se crea el sitio web dentro de la pestaña Websites y la base de datos dentro de la pestaña MS SQL

Figura 50

Creación del sitio web

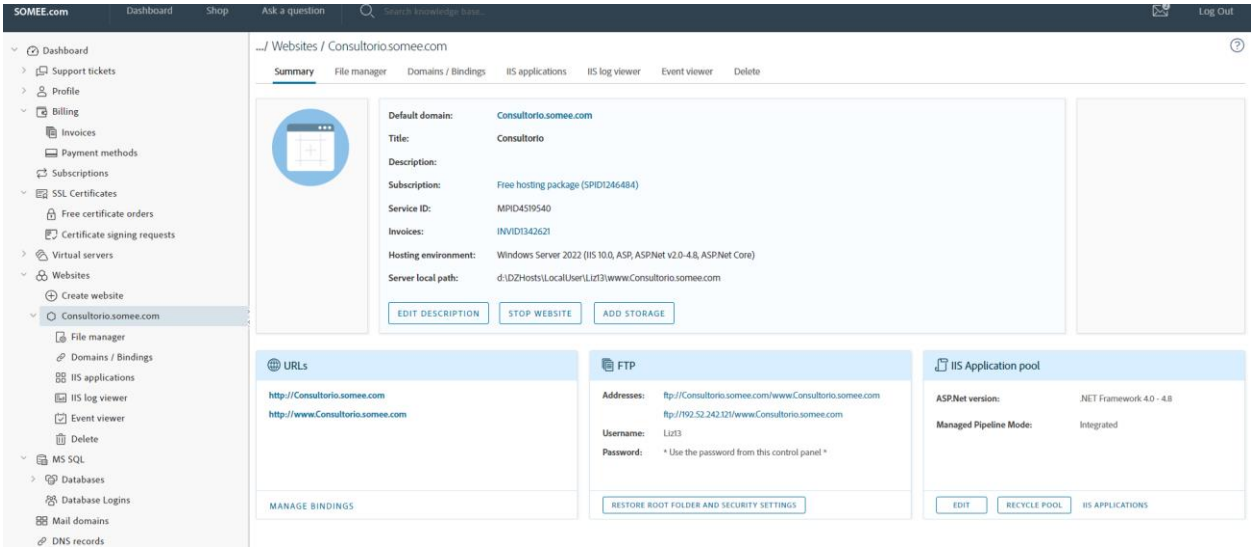
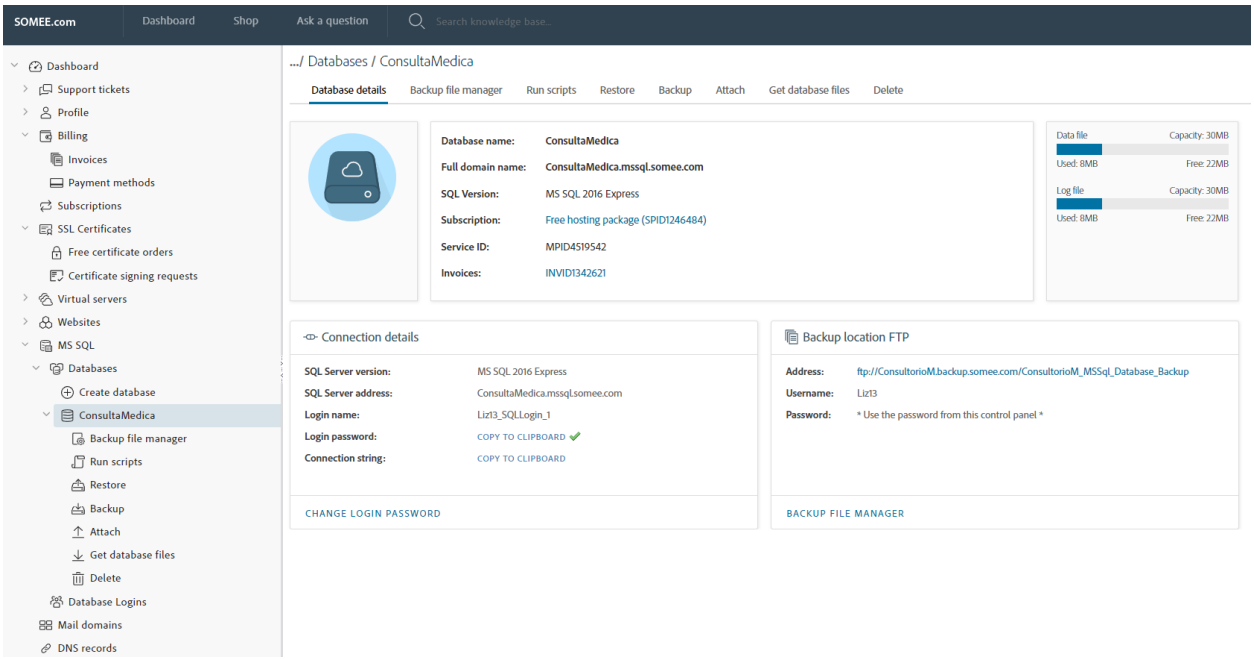


Figura 51

Creación de la base de datos



Realizar la conexión por medio de SQL Server a la base de datos creada en Somee, buscar el nombre de la base de datos creada y cargar el script de creación de las tablas.

Figura 52

Conexión con la base alojada en Somee

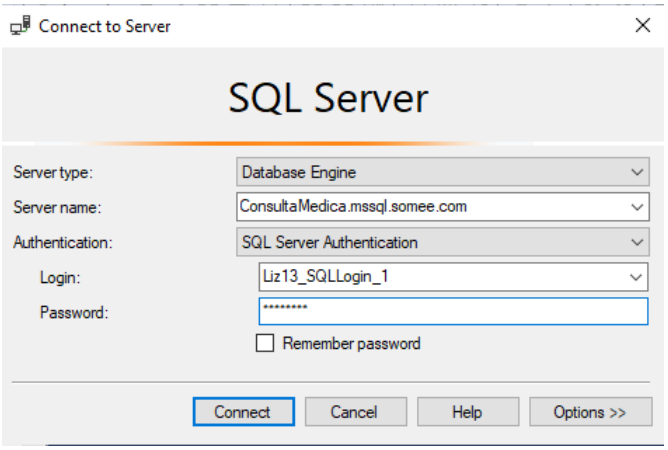
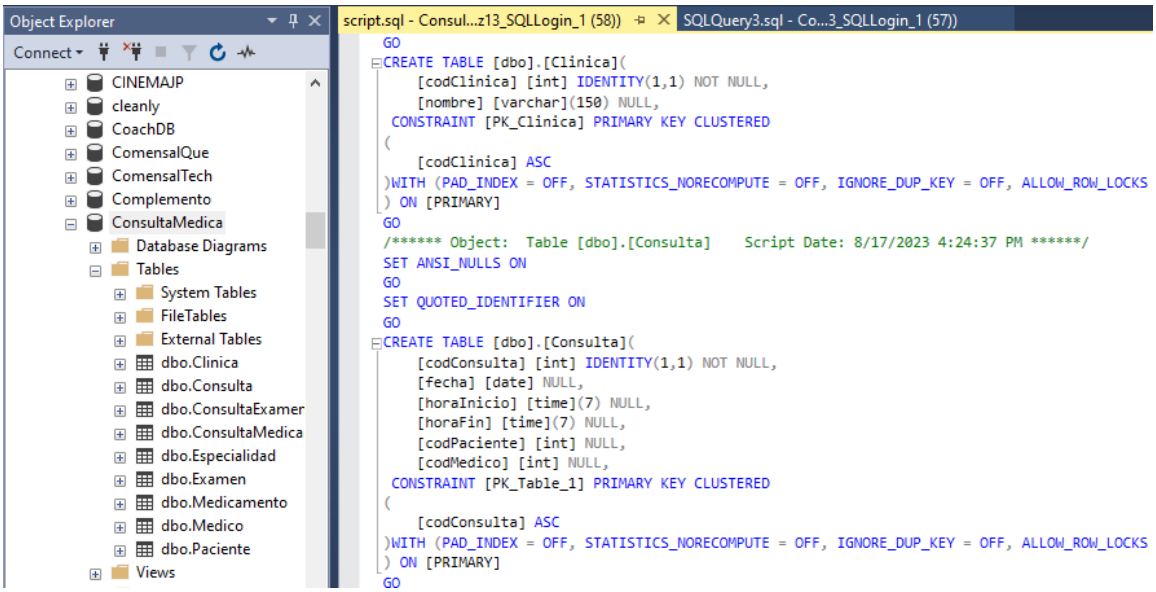


Figura 53

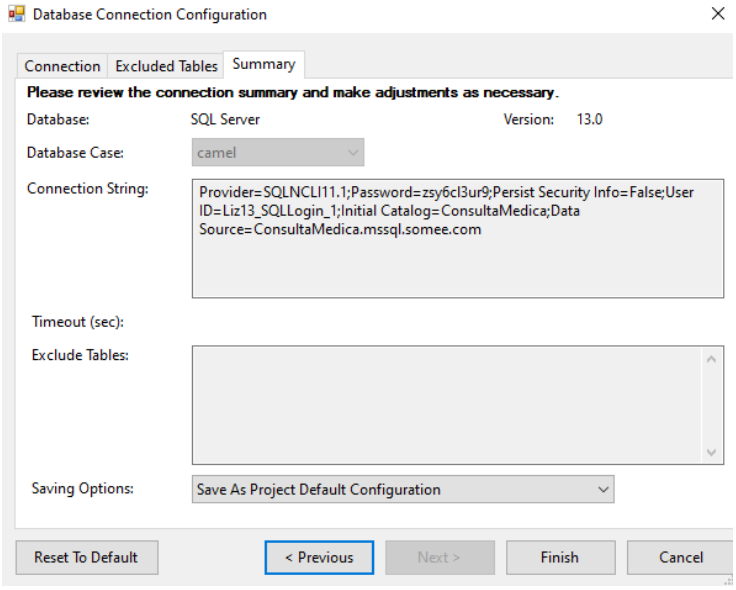
Carga de la base de datos



Conectar el proyecto a la base de datos alojada en Somee a través de la configuración de la conexión con la base de datos.

Figura 54

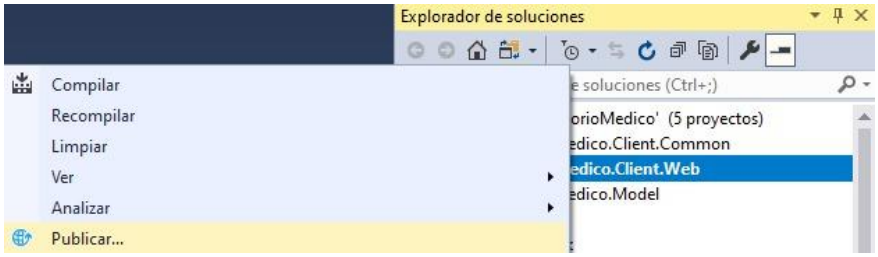
Conexión con Somee desde el proyecto Xomega



Una vez establecida la conexión con la base de datos, se procede a la generación de los archivos correspondientes al modelo importado. A continuación, se efectúa la generación de las operaciones CRUD para las tablas importadas. Subsiguientemente, se ejecuta la aplicación con el fin de verificar la correcta generación del proyecto y tras confirmada la validez del proyecto, se selecciona el proyecto Client.Web y se procede a la publicación de la aplicación.

Figura 55

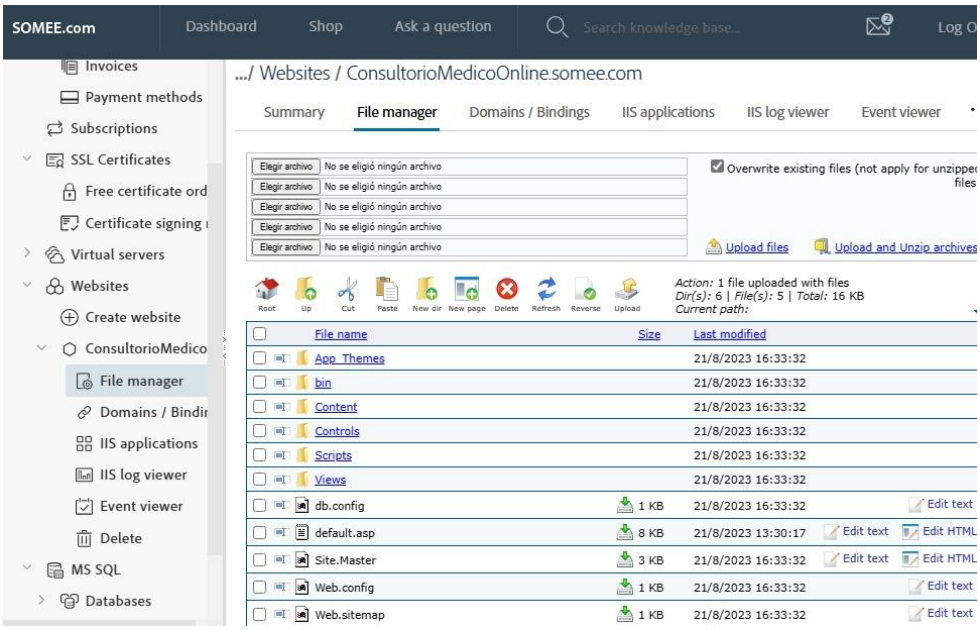
Publicación de la aplicación



Se ingresa a la carpeta donde se publicó el proyecto y almacenar todo el contenido de la ruta en un archivo .zip, consecuentemente en la página de Somee en Websites ingresar en la pestaña File manager y cargar él .zip anteriormente creado.

Figura 56

Carga de los archivos en File manager - Somee



Ingresar al link del sitio web creado con anterioridad dentro de la página de Somee (http://consultoriomedicoonline.somee.com/) y comprobar el funcionamiento del aplicativo.

Figura 57

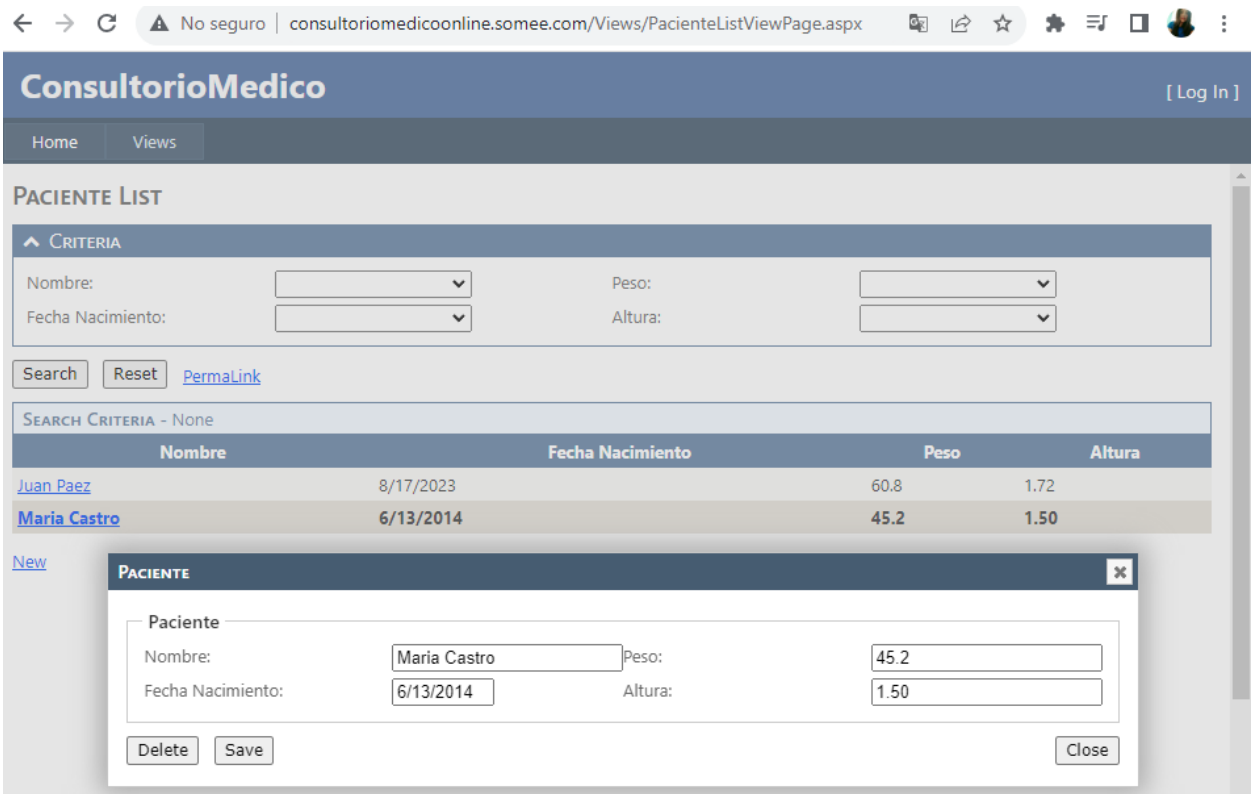
Pantalla crear paciente - Xomega

Figura 58

Pantalla listar y buscar pacientes - Xomega

Figura 59

Pantalla eliminar y actualizar pacientes - Xomega



MDA

Tomando en cuenta los requerimientos del entorno, conocimientos, previos y facilidades de uso, se decidió usar los servicios de Amazon Web Services AWS para el despliegue de la aplicación.

Figura 60

Arquitectura para el despliegue de la aplicación en AWS

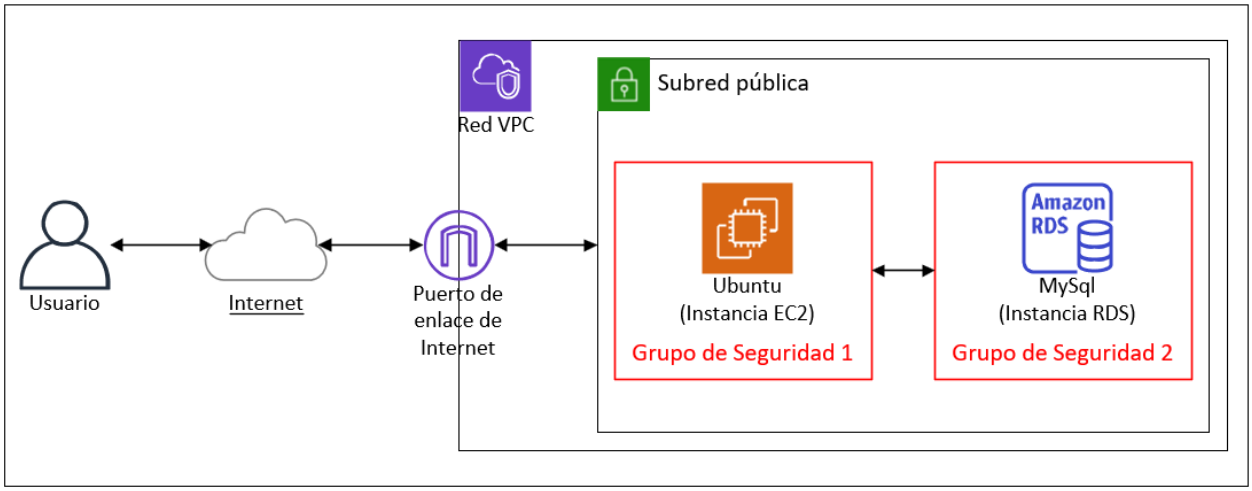


Figura 61

Detalles de la VPC (Red para AWS)

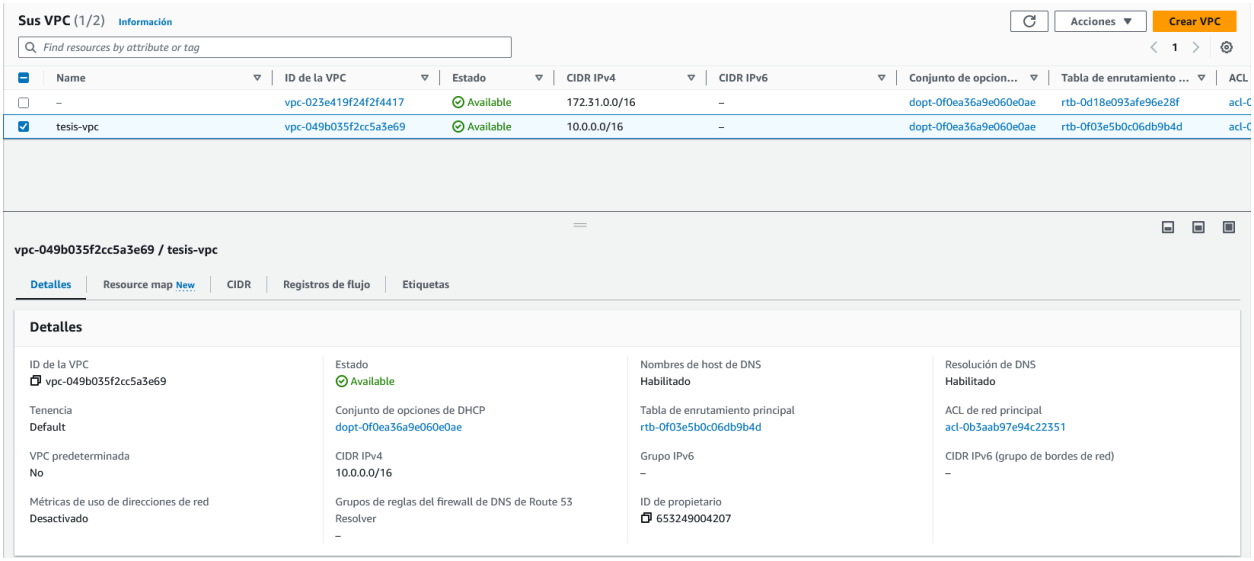


Figura 62

Gateway para el acceso a internet de la VPC

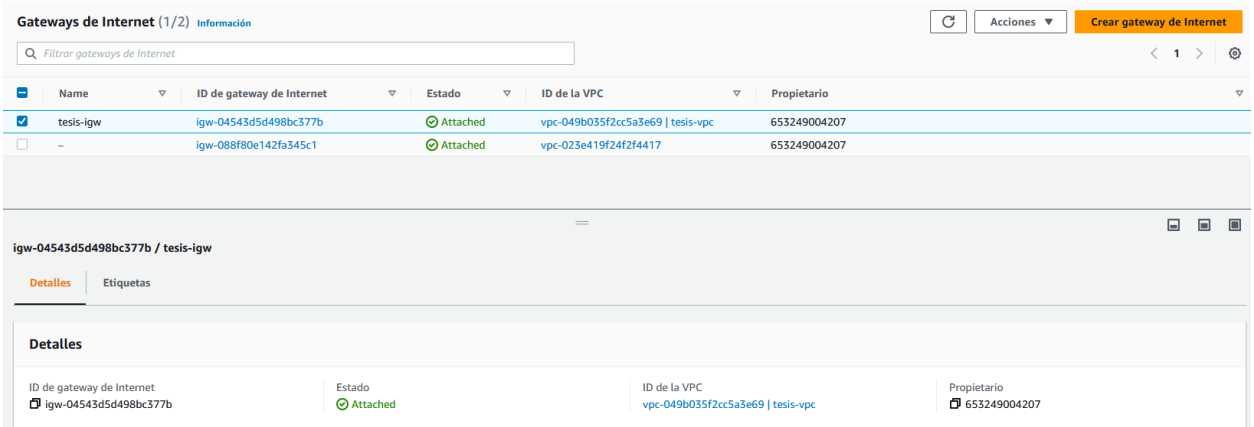


Figura 63

Grupo de seguridad para EC2 en puerto 8080

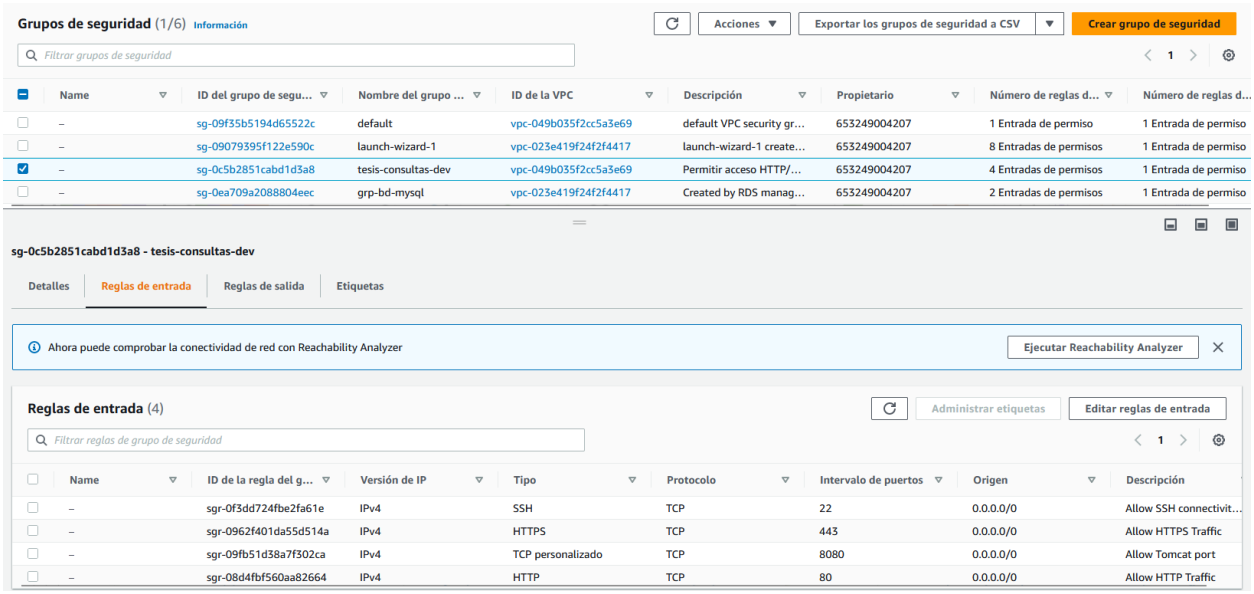


Figura 64

Grupo de seguridad para RDS en el puerto 3306.

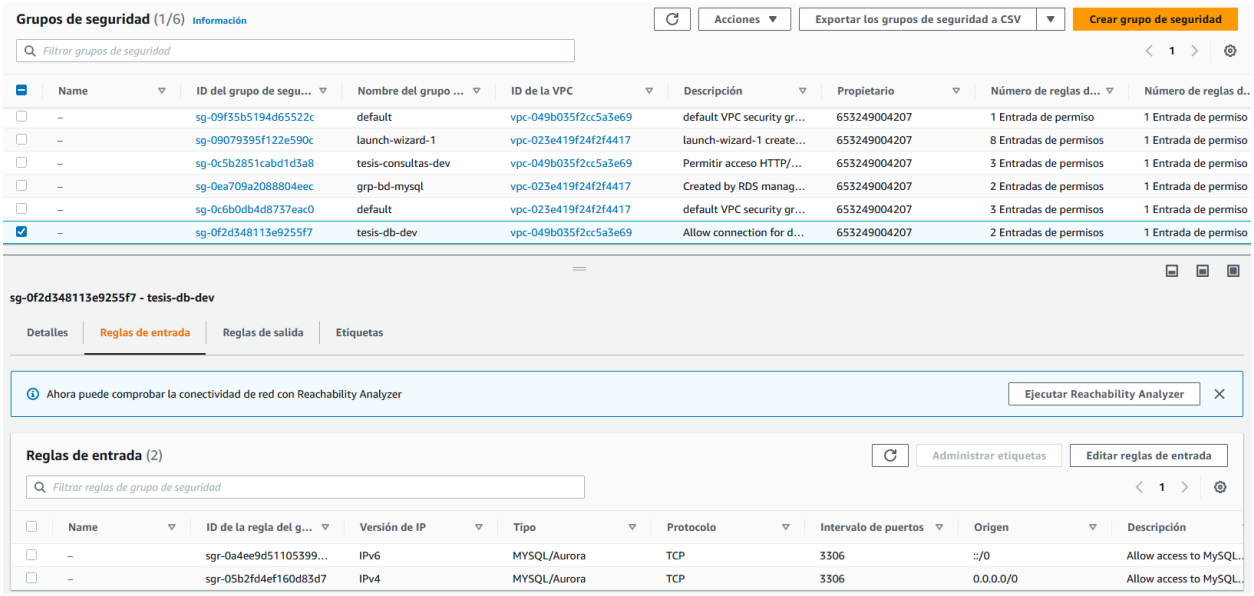


Figura 65

Detalles de la instancia EC2

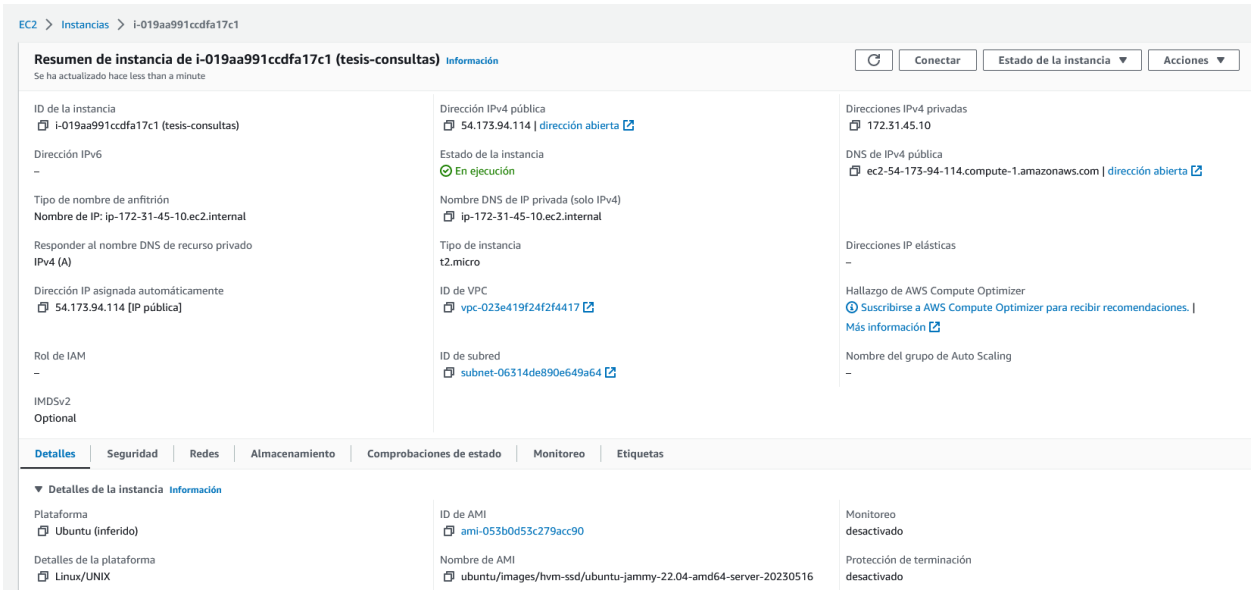


Figura 66

Detalles de la instancia RDS

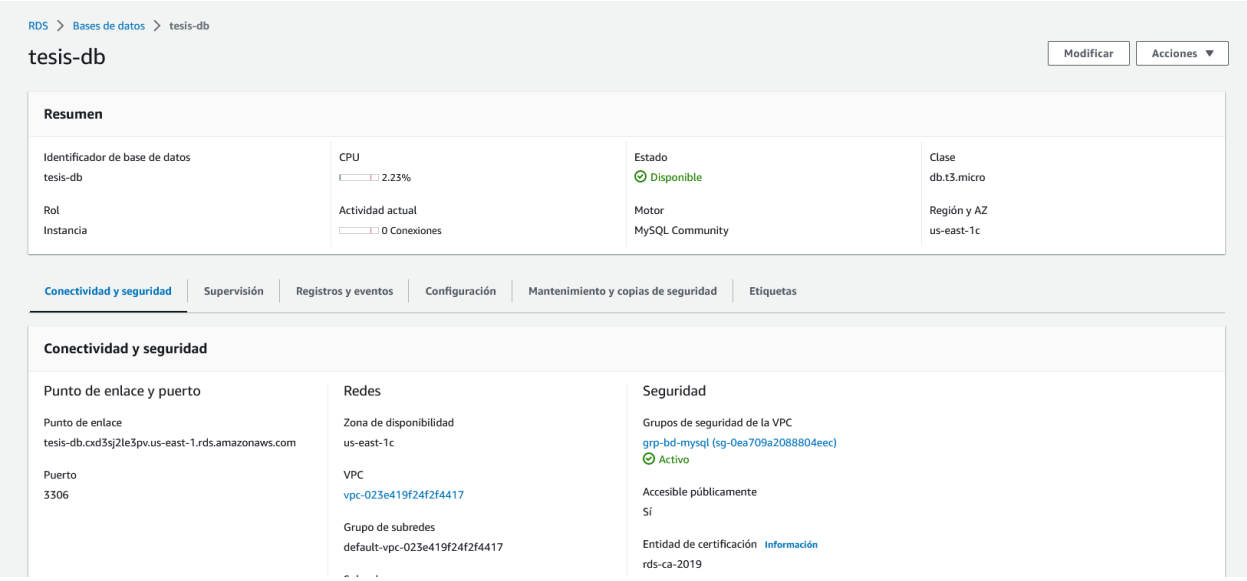


Figura 67

Página de gestión de aplicaciones de Tomcat en EC2.

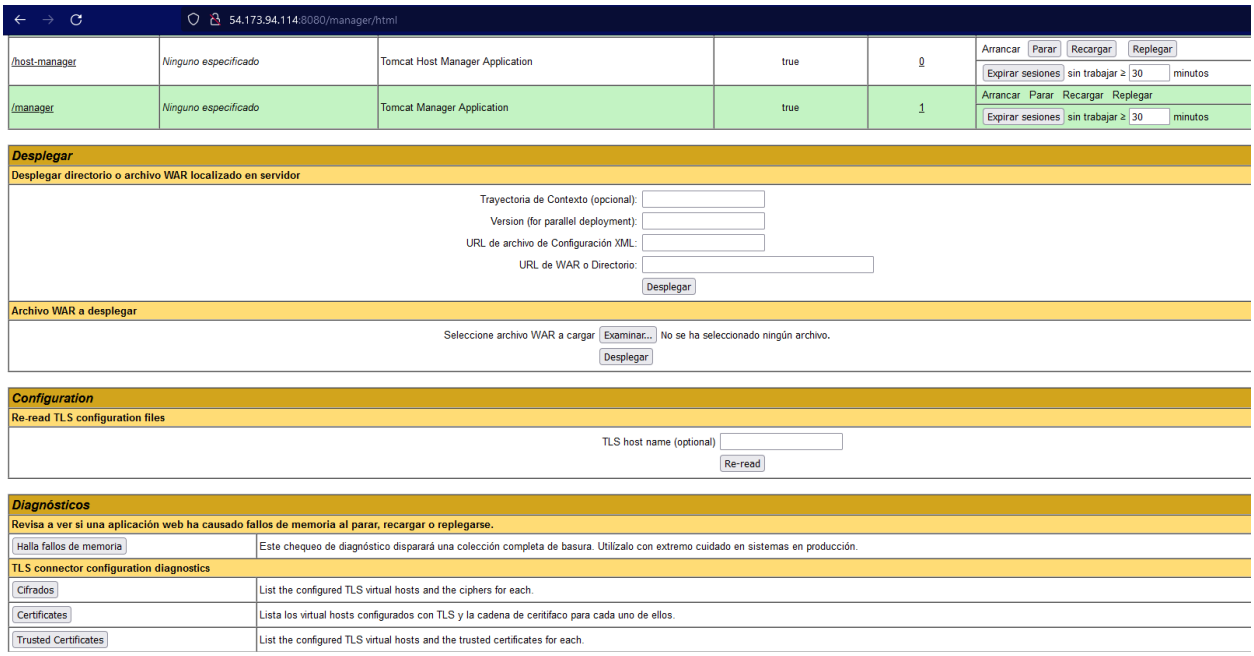


Figura 68

Despliegue de la aplicación en Tomcat

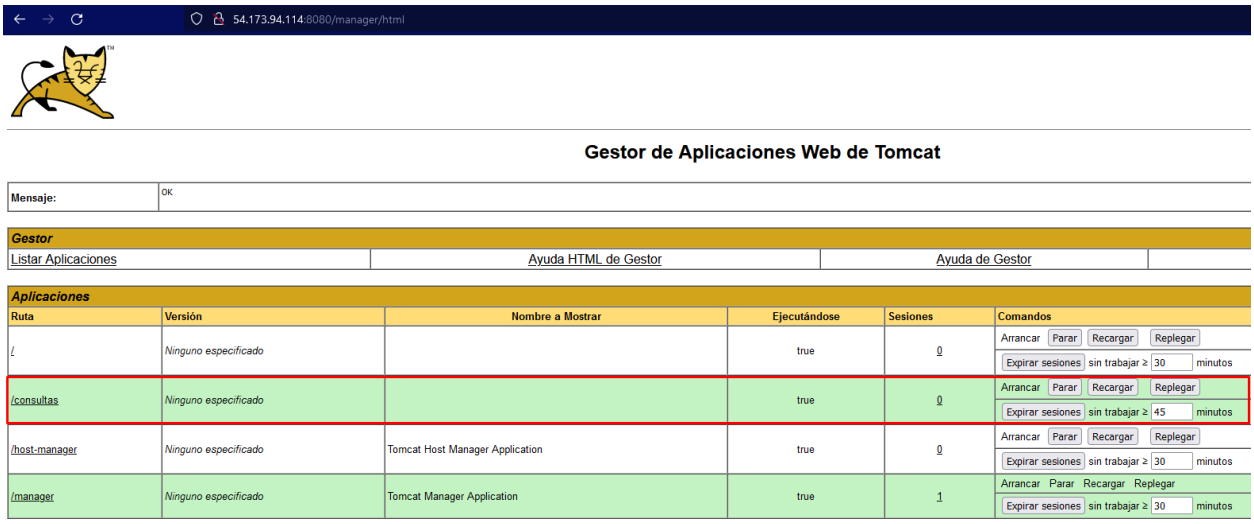


Figura 69

Página principal generado con MDA

Nota. El sistema fue desplegado de acuerdo a la arquitectura presentada en la Figura 60. y se encuentra disponible mediante la siguiente url: http://54.173.94.114:8080/consultas/ec/edu/espe/tesis/consultas/web/consulta/consultas-medicas.jsf?_afPfm=ay875l381

Pruebas

Para la fase de pruebas se consideraron diferentes factores que pueden modificar los tipos de pruebas, niveles de aceptación y escenarios que se planteen durante este proceso, en este trabajo los aspectos más importantes a considerar es que es un aplicativo web y el código ha sido generado con la ayuda de herramientas MDD y MDA, lo cual implica que se deberían hacer pruebas de funcionalidad para verificar que el código generado tenga un flujo adecuado y cumpla con el funcionamiento esperado, adicional a esto se pueden agregar pruebas de integración dado que complementan la verificación del correcto flujo de la aplicación durante la verificación de cada una de las funcionalidades, otro tipo de pruebas que se han trabajado son las pruebas de rendimiento, ya que, esto ayudará a verificar el comportamiento que tiene el aplicativo ante cargas de trabajo mayores y la eficiencia que tiene el código generado.

Se ha realizado la ejemplificación de las pruebas con la opción de pacientes, pero se podría realizar con cualquiera de las otras opciones, dado que, cada opción tiene un funcionamiento similar, diferenciándose únicamente en los campos, tipos de datos y ciertas validaciones.

Pruebas de funcionalidad

Estas pruebas se realizan con el fin de verificar que cada una de las opciones (crear, listar, actualizar, eliminar) funcione correctamente en cada módulo del aplicativo. En la Tabla 5 se puede ver cada una de las funcionalidades generadas para la gestión de pacientes, tomando en cuenta los pasos expuestos en los diagramas establecidos en el Capítulo 3, los resultados esperados, resultados obtenidos y notas adicionales que pueden ser relevantes durante cada una de las pruebas, en cada aplicativo.

Tabla 11

Resultados de las pruebas de funcionalidad en la gestión de Pacientes

Funcionalidad	Resultados Esperados	Resultados obtenidos	Notas adicionales
Crear Paciente	Se agrega un nuevo paciente al listado.	El paciente ingresado se refleja correctamente en la lista.	**MDD:** Para poder observar el paciente creado se debe dirigir a la vista Paciente List. **MDA:** Se muestra un mensaje de confirmación tras haber creado un nuevo paciente o un mensaje de error si no se pudo agregar el registro.
Listar Paciente	Se listan todos los pacientes existentes con sus respectivos atributos.	Listado de todos los pacientes con cada uno de sus atributos.	**MDD/MDA:** La vista incluye inputs que permiten el filtrado de los resultados.
Actualizar Paciente	Se actualizan los campos del paciente seleccionado.	Se pueden ver los cambios del paciente que se seleccionó para editar.	**MDD:** Se muestran los cambios realizados en la lista de pacientes. **MDA:** Se muestra un mensaje de confirmación tras haberse editado el paciente o un mensaje de error si no se pudo editar el registro.

Funcionalidad	Resultados Esperados	Resultados obtenidos	Notas adicionales
Eliminar Paciente	Se elimina el paciente que se seleccionó.	El paciente seleccionado para eliminar ya no forma parte del listado de pacientes.	**MDD:** Se despliega un mensaje para la confirmación de la eliminación del registro. **MDA:** Se muestra un mensaje de confirmación tras haberse eliminado el paciente o un mensaje de error si hubo algún problema al realizar la eliminación.

En ambas herramientas, MDD y MDA, se comprobó satisfactoriamente que las opciones del CRUD funcionaron correctamente en cada módulo del aplicativo, según lo establecido en los diagramas. Las funcionalidades generadas para la gestión de pacientes cumplieron con las verificaciones esperadas en ambas herramientas.

Pruebas de integración

La realización de pruebas de integración en una aplicación es esencial para garantizar la coherencia y el funcionamiento adecuado de componentes interconectados. En MDD se debe probar la interacción precisa entre "Full CRUD with Views", la "Service Implementation" y el "View Model" y la base de datos asegurando un funcionamiento fluido y coherente en la aplicación. Así mismo en MDA se debe verificar que la integración entre las vistas, controladores, DAOs y base de datos sea correcta. Esto se puede verificar a la par que se realizan las pruebas de funcionamiento y se lo ha hecho únicamente comprobando que la información en las vistas corresponda a lo reflejado en la base de datos, puesto que, esto

asegura que todo el flujo e integración se esté haciendo correctamente desde las vistas hasta la base datos.

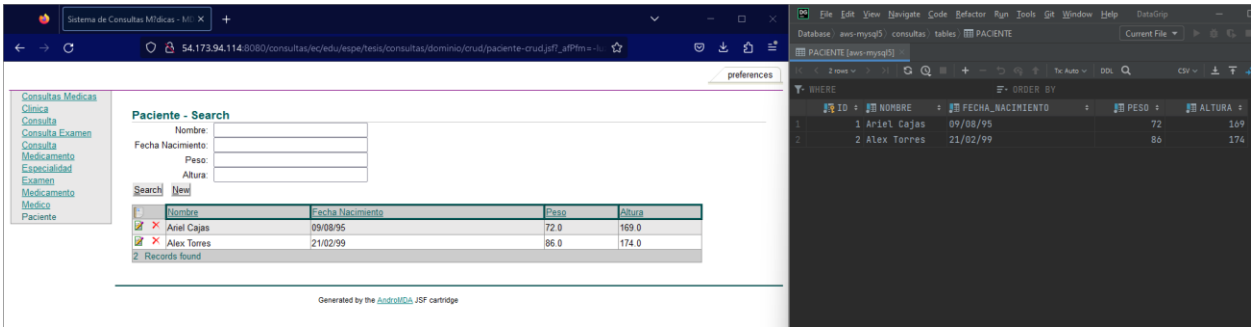
1. Listar pacientes, todos los pacientes en el listado de la vista corresponden a los registros de la tabla paciente dentro de la base de datos.

Figura 70

Listado de pacientes en la interfaz y base de datos - MDD

Figura 71

Listado de pacientes en la interfaz y base de datos - MDA



2. Crear paciente, tras guardar un nuevo paciente desde la vista ese registro se verifica en la tabla de pacientes dentro de la base de datos.

Figura 72

Registro de pacientes en la base antes del guardado - MDD

Figura 73

Guardado del paciente y actualización en la base de datos - MDD

Figura 74

Registro de pacientes en la base antes del guardado - MDA

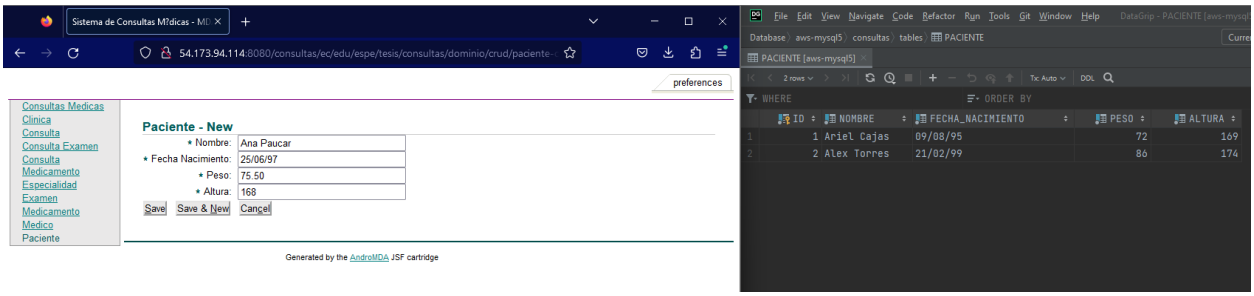
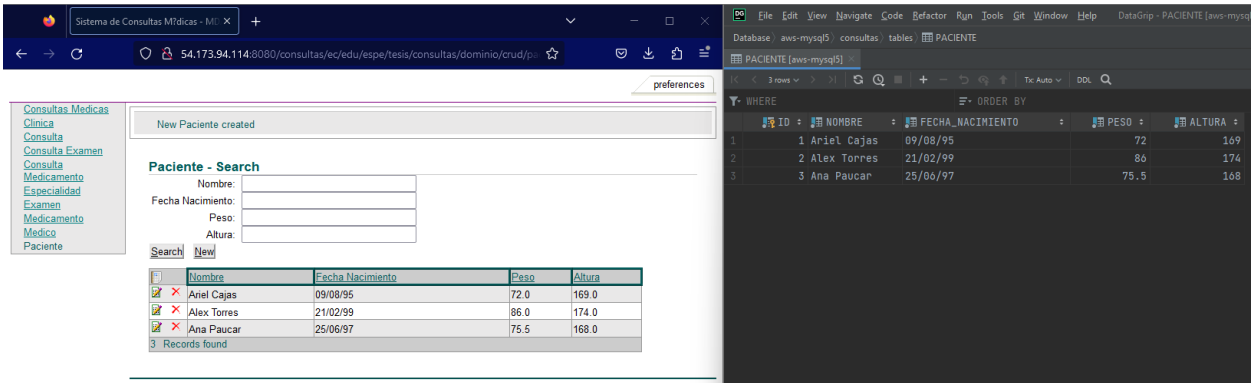


Figura 75

Guardado del paciente y actualización en la base de datos - MDA



3. Editar paciente, tras editar un paciente desde la vista se pueden verificar los cambios en el registro correspondiente en la base de datos.

Figura 76

Actualización paciente y registro anterior en la base de datos - MDD

Figura 77

Guardar cambios paciente y actualización en la base de datos - MDD

Figura 78

Actualización paciente y registro anterior en la base de datos - MDA

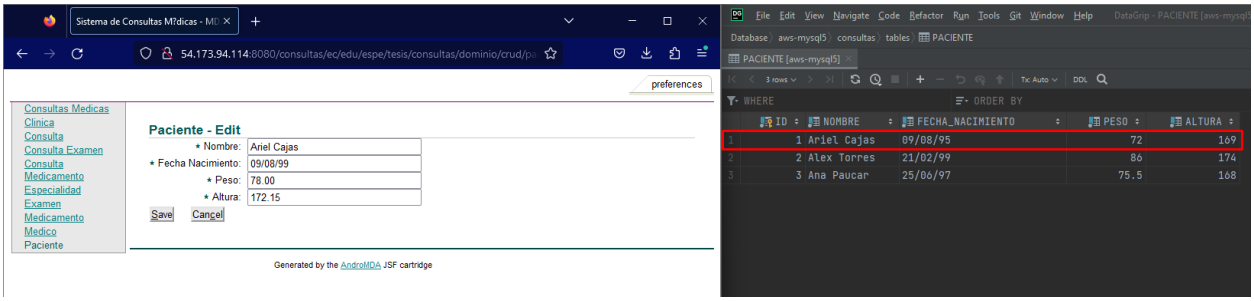
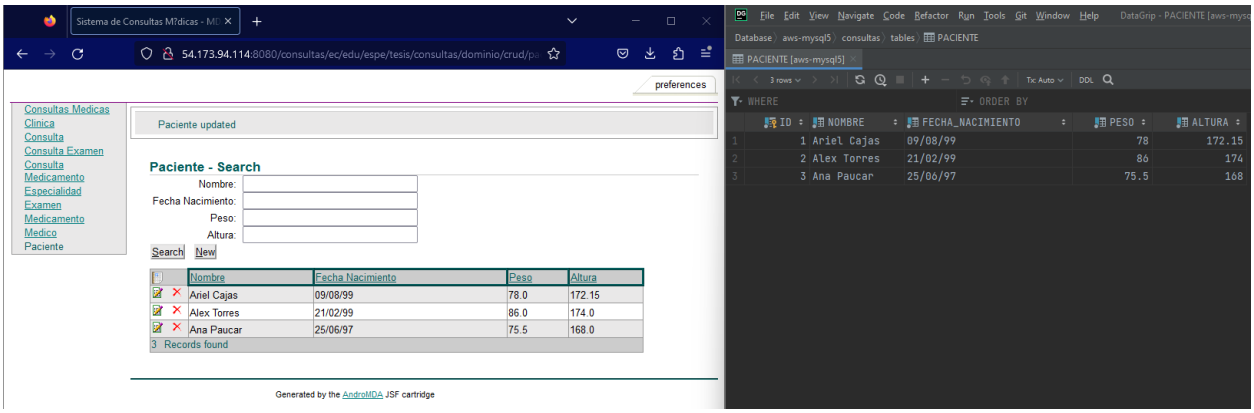


Figura 79

Guardar cambios paciente y actualización en la base de datos - MDA



4. Eliminar paciente, el paciente seleccionado para la eliminación en la base de datos ya no debe estar dentro de la tabla paciente en la base de datos.

Figura *80*

Selección del paciente para la eliminación de la base de datos - MDD

Figura 81

Eliminación del paciente y actualización en la base de datos – MDD

Figura 82

Selección del paciente para la eliminación de la base de datos - MDA

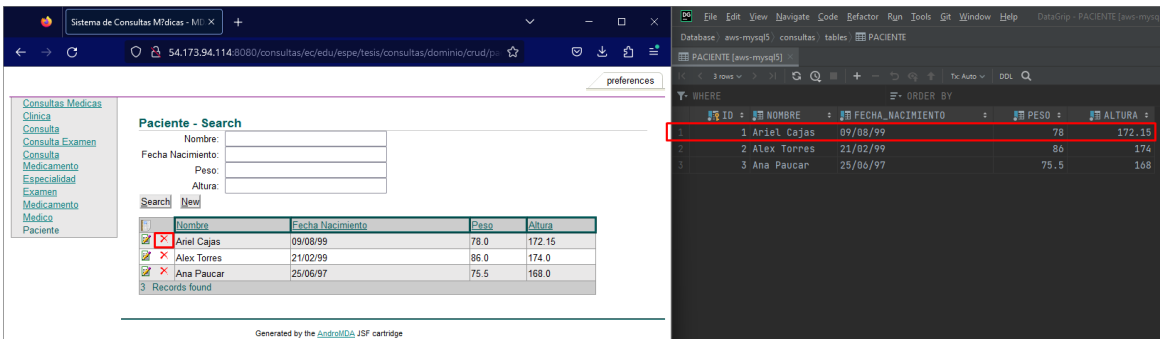
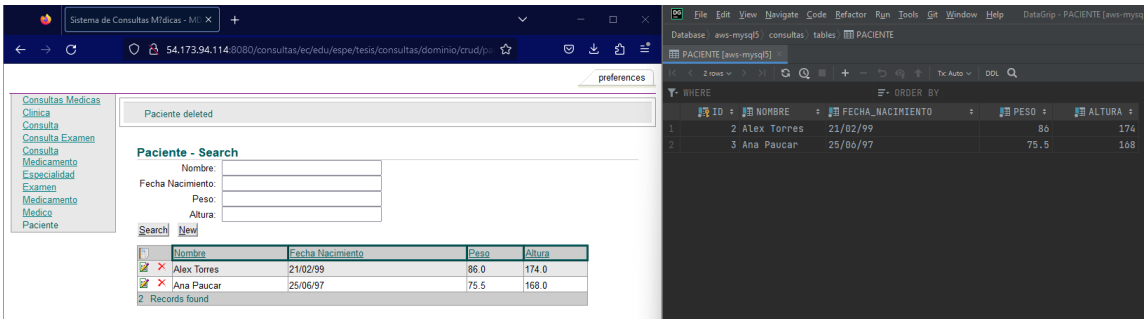


Figura 83

Eliminación del paciente y actualización en la base de datos - MDA



En MDD, la interacción entre "Full CRUD with Views", la "Service Implementation", el "View Model" y la base de datos se verificó exitosamente, asegurando un funcionamiento coherente y fluido. En MDA, la integración entre vistas, controladores, DAOs y la base de datos también se confirmó como correcta. Ambas herramientas demostraron una comunicación exitosa entre los componentes y la base de datos, garantizando una correcta integración.

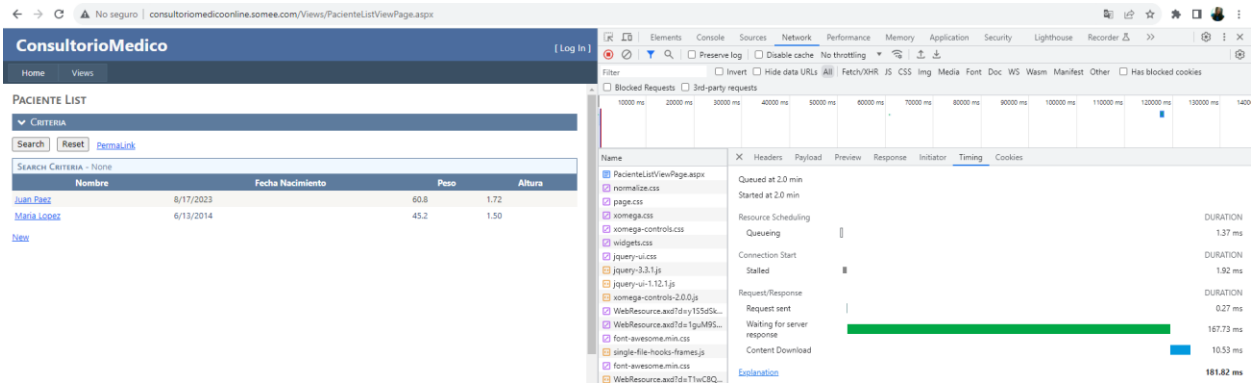
Pruebas de rendimiento

Las pruebas de rendimiento ayudan a visualizar cual es el comportamiento que tiene la aplicación bajo diferentes cargas de trabajo, esto suele ser necesario para tener una estimación del rendimiento de la aplicación en el caso de tener una carga de datos significativa o varios usuarios conectados al mismo tiempo e interactuando con la aplicación. Generalmente el comportamiento del sistema se define mediante los tiempos de respuesta de la aplicación y los fallos que podrían suscitarse, pero también se deben considerar factores como el consumo de memoria y de procesamiento, ya que esto refleja la optimización que tiene el código que se ha generado.

El primer escenario a analizar es con solo los 2 registros creados inicialmente dentro de la tabla de pacientes, para este análisis se ha verificado los tiempos en el tab de Red, dentro de la consola del navegador, con lo cual se puede visualizar cada una de las peticiones que el navegador hace a la aplicación y cuanto es el tiempo que se tarda en devolver la información (Esperando) y cuanto es el tiempo que se tarda en recibir la vista correspondiente (Recibiendo).

Figura 84

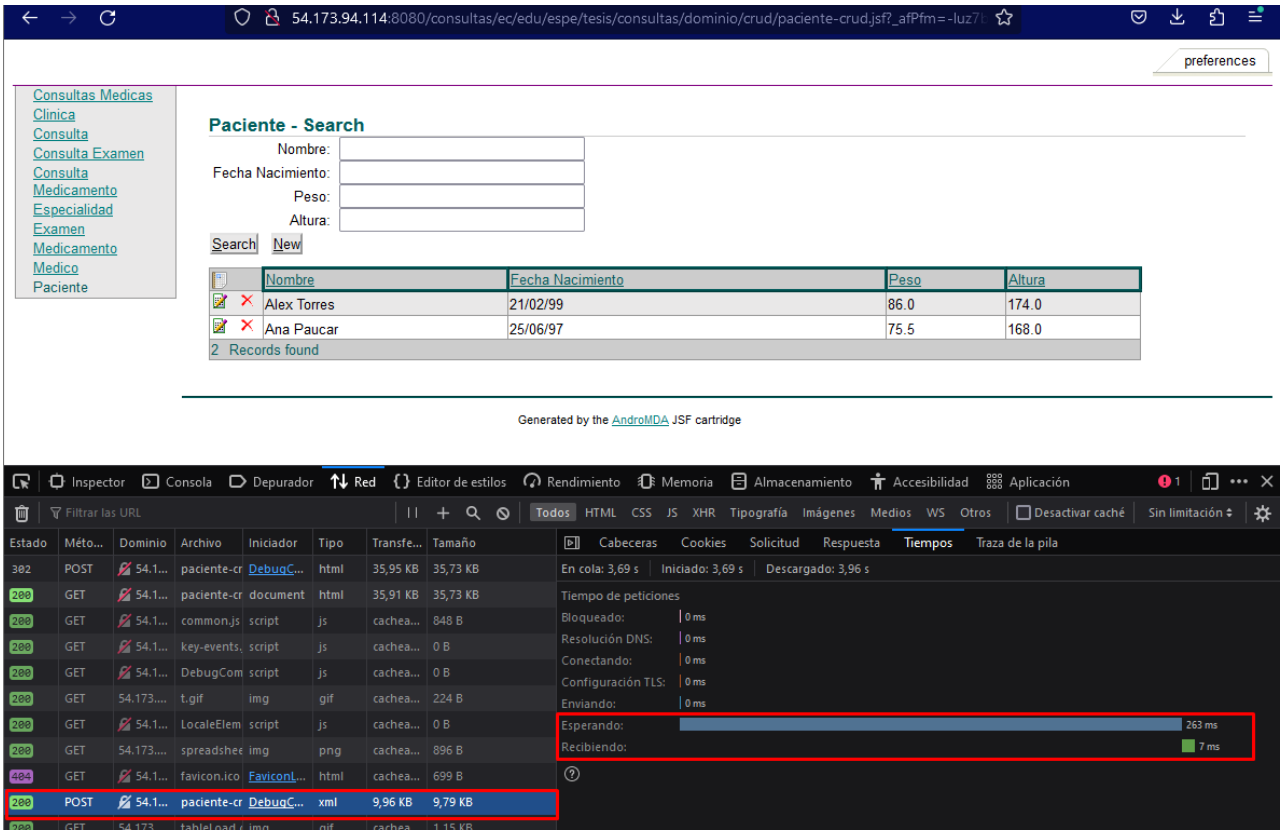
Tiempo de respuesta con 2 registros en la tabla de pacientes - MDD



Nota. La aplicación tarda 167.73 ms en devolver la información y el tiempo de renderizado es de 10.53 ms.

Figura 85

Tiempo de respuesta con 2 registros en la tabla de pacientes - MDA

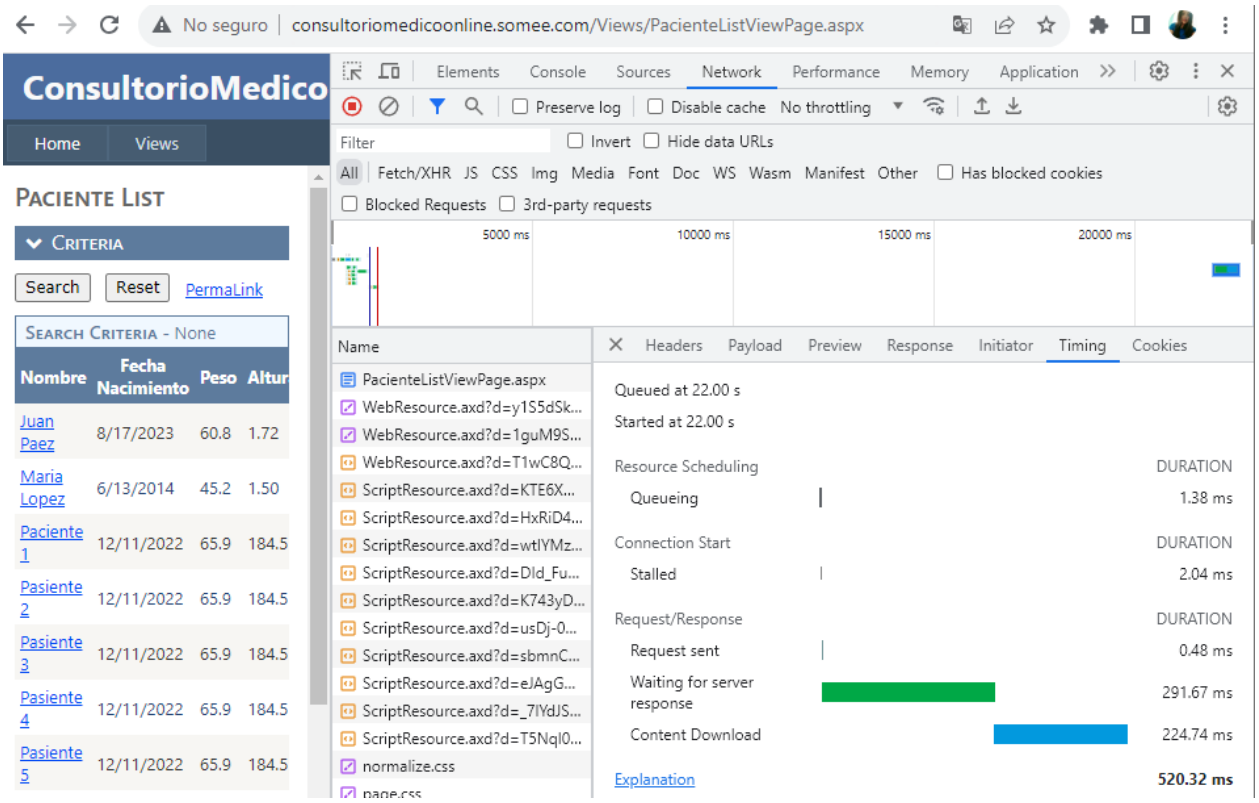


Nota. En este primer escenario la aplicación tarda 263 ms en devolver la información.

Para el siguiente caso se ingresó mediante la aplicación 20 registros, proceso en el cual no existió ningún error ni en MDD o MDA.

Figura 86

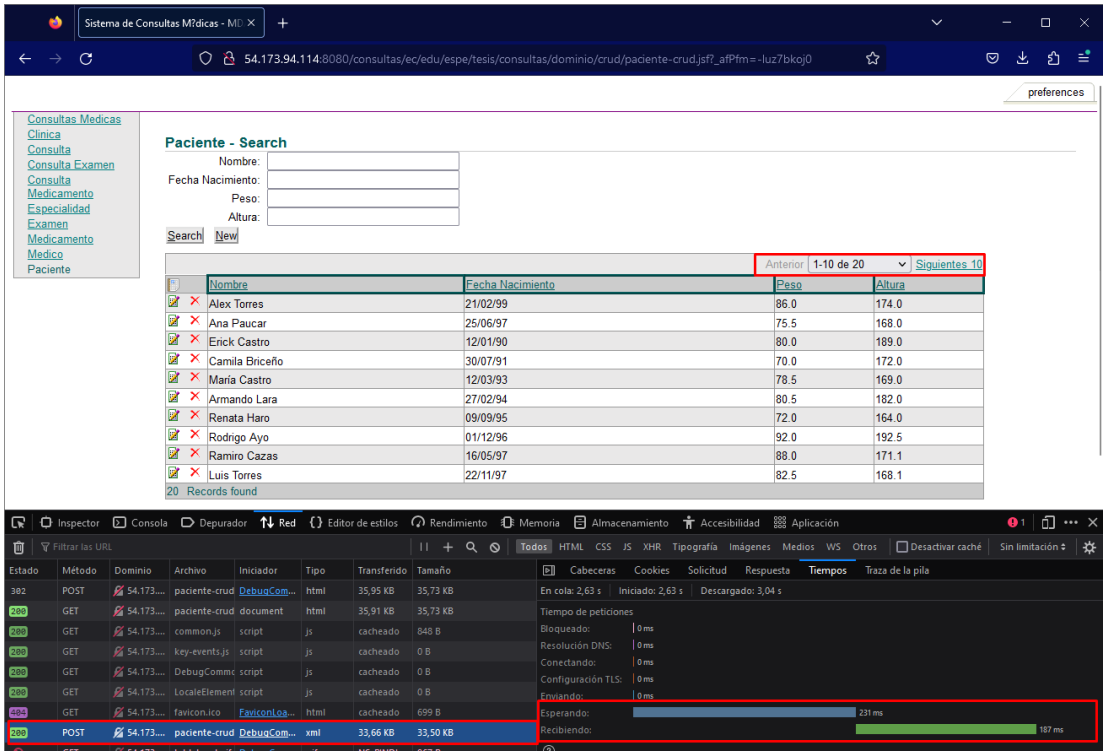
Tiempo de respuesta con 20 registros en la tabla paciente - MDD



Nota. El tiempo de respuesta de la aplicación es de 291.67 ms, mientras que el tiempo de renderizado es de 224.74 ms.

Figura 87

Tiempo de respuesta con 20 registros en la tabla paciente - MDA

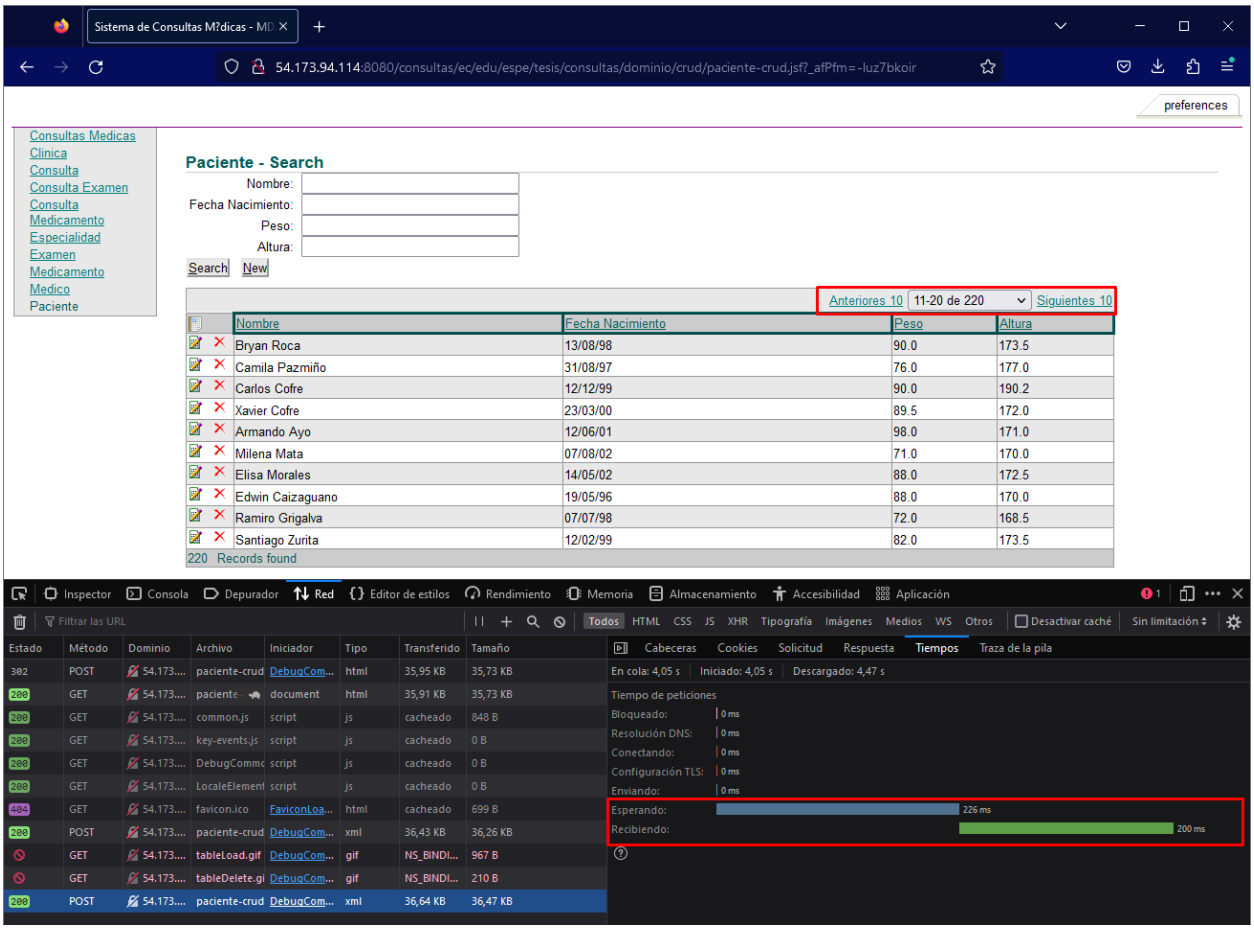


Nota. El tiempo de respuesta de la aplicación es de 231 ms, mientras que el tiempo de renderizado es de 187 ms.

Por defecto AndroMDA genera las vistas con una paginación de 10 registros por vista tal y como se puede apreciar en la Figura 88 al solicitar ver los siguientes registros se vuelve a hacer una petición a la base de datos, en este caso con 226 ms en el tiempo de respuesta del servidor, lo cual disminuye de la anterior petición.

Figura 88

Tiempo de carga para los siguientes 10 registros



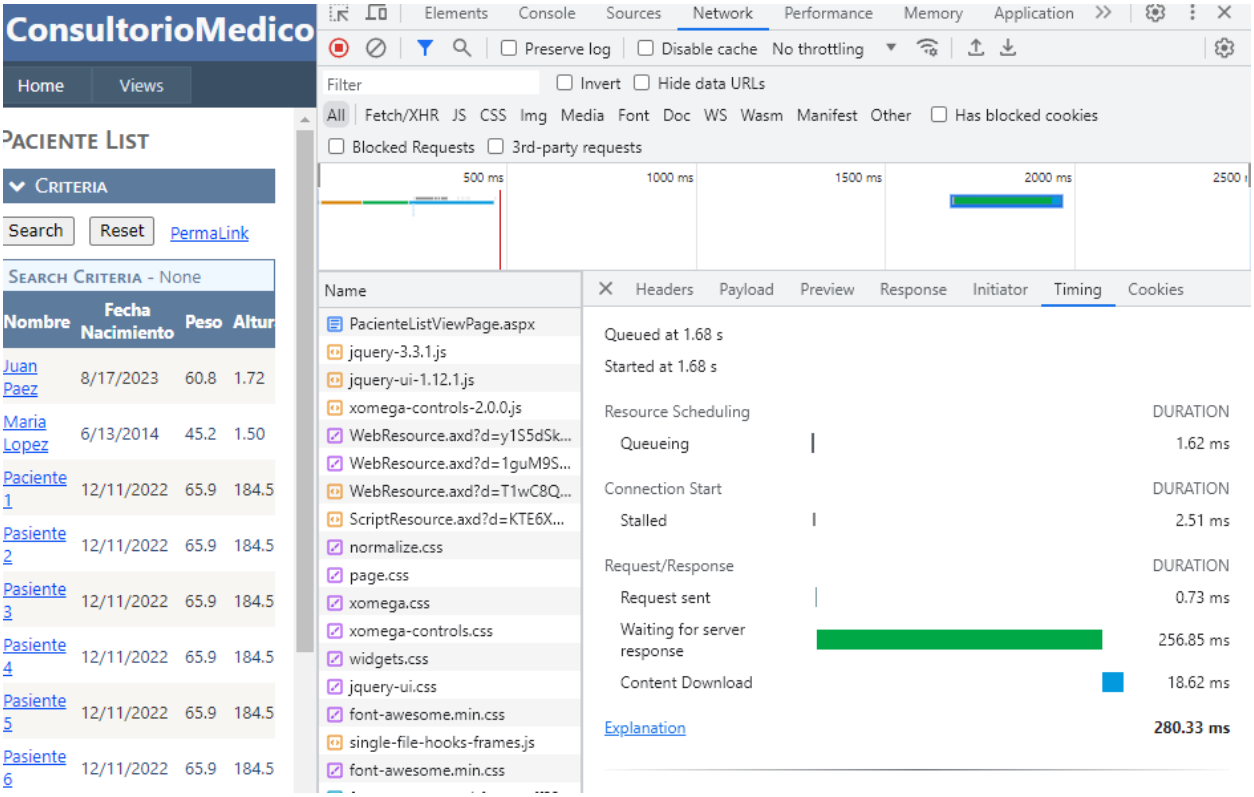
Por último, al realizar la consulta de 200 registros más para comprobar el tiempo de respuesta.

Para esto se debe tomar en cuenta que AndroMDA cuenta con una paginación de 10 registros por vista, en cambio Xomega presenta una paginación de 20 registro por vista, esto ayuda a que los tiempos de respuesta por parte del servidor sean similares y no tengan picos muy altos, ya que, siempre se estará trayendo solo 10 o 20 registros, los cuales pueden estar en cualquier intervalo permitido por la aplicación y dentro de la cantidad de registros totales en la base de datos, lo cual significa que la aplicación tendrá un comportamiento similar a pesar de

que exista una gran cantidad de registros en cada una de las opciones del sistema, ya que el tiempo de respuesta y renderización se mantendrán en la media.

Figura 89

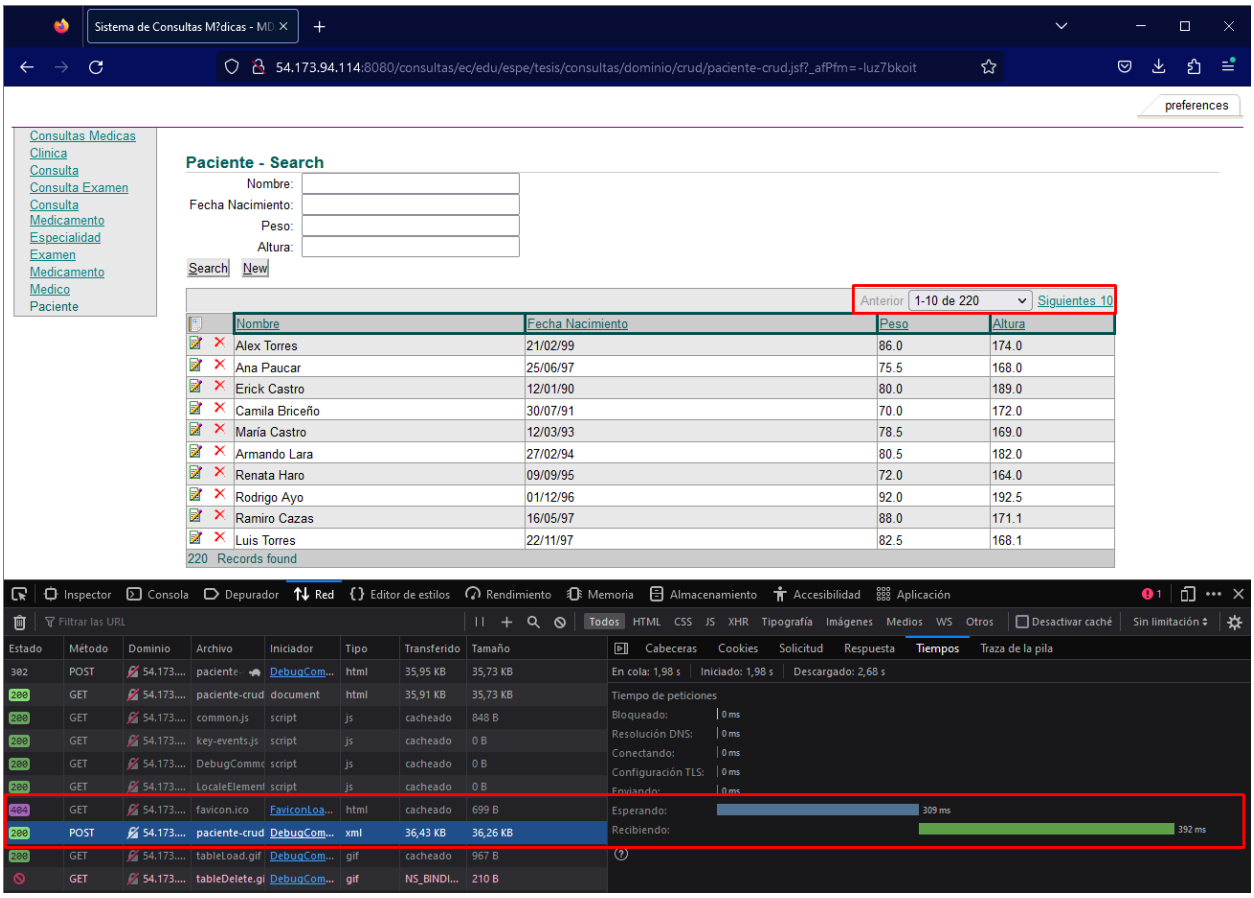
Tiempo de carga del listado de pacientes con 220 registros - MDD



Nota. El tiempo de respuesta de la aplicación es de 256.85 ms, mientras que el tiempo de renderizado es de 18.62 ms.

Figura 90

Tiempo de carga del listado de pacientes con 220 registros - MDA



Nota. El tiempo de respuesta de la aplicación es de 309 ms, mientras que el tiempo de renderizado es de 352 ms.

Se evaluaron diferentes escenarios de carga. En el primer escenario, con 2 registros, MDD mostró un tiempo de respuesta bajo (167.73 ms) al igual que MDA (231 ms). A medida que se incrementó el número de registros a 20, MDD mantuvo un tiempo de respuesta más bajo (291.67 ms) y MDA (226 ms). Además, se observó que MDA generó vistas con paginación, lo que contribuyó a tiempos de respuesta más consistentes, mientras que MDD presentó un listado continuo sin paginación.

En el escenario final con 220 registros adicionales, MDD género vistas con paginación, no obstante, continuó manteniendo un tiempo de respuesta más bajo (256.85 ms) en comparación con MDA (309 ms), demostrando un rendimiento más eficiente a pesar del aumento en la cantidad de registros.

En general, tanto Xomega y AndroMDA, mostraron cumplir con las pruebas de funcionalidad, integración y rendimiento en lo que respecta a la gestión de pacientes en la aplicación. Sin embargo, en términos de rendimiento, MDD demostró un mejor desempeño en términos de tiempos de respuesta, especialmente en situaciones de carga más alta. La generación de vistas con paginación en MDD y MDA permitió un mejor control de los tiempos de respuesta. En comparación con MDA, MDD se destacó por ofrecer una integración fluida, un rendimiento más eficiente y tiempos de respuesta más bajos, lo que podría indicar una ventaja en términos de usabilidad y experiencia del usuario en situaciones de carga más intensas.

Tiempo de desarrollo

Para calcular un tiempo de desarrollo de las aplicaciones generadas con MDD y MDA se debe considerar el tiempo necesario para crear los diferentes diagramas UML que se necesitan para la generación del código, tanto con Xomega como con AndroMDA, otro aspecto a considerar es el tiempo que se debe invertir para poder manejar cada una de las herramientas, ya que, a pesar de que sean diagramas similares a UML y el lenguaje de programación del código generado puede ser conocido, como java, c#, jsf, etc. es necesario tener un conocimiento sobre el uso de las herramientas para la generación de código, puesto que esto puede definir las funcionalidades y flujo que tendrán cada una de las aplicaciones web. Tanto para MDD como para MDA el tiempo requerido es similar, dado que en los dos casos se tuvo que aprender sobre el uso de la herramienta y, como se basaron en los mismos diagramas UML para la creación de los modelos no existió mayores diferencias en la generación de la aplicación.

Tomando en cuenta la aplicación generada y la cantidad de tablas en la base de datos, el tiempo que tomó la generación de cada aplicación fue menor al que se hubiese requerido para crear una aplicación similar con una metodología tradicional, además es muy importante recordar que los diagramas que se han generado también forman parte de la documentación de la aplicación. Si se anula el factor del tiempo del desarrollador tanto para aprender a utilizar una herramienta MDD o MDA y el tiempo para aprender uno o varios lenguajes de programación para crear una aplicación web de forma tradicional, y solo se toma en cuenta el desarrollo de la aplicación, se tendría un escenario en el que la aplicación generada con XOmega o AndroMDA se puede crear en horas mientras que la aplicación tradicional puede requerir de días para ser completada o incluso requerir a más de un desarrollador, debido a los diferentes lenguajes de programación que se podría utilizar y a la creación manual de la base de datos.

Conclusiones y Recomendaciones

Conclusiones

Los estudios encontrados por medio de la SLR respaldan la elección de enfoques MDD y MDA para la aplicación web propuesta, especialmente con el uso del lenguaje IFML para diseño UX/UI. Estos enfoques ofrecen mejoras en calidad, productividad y eficiencia del desarrollo, aunque presentan desafíos en cuanto a la curva de aprendizaje y flexibilidad. En general, la implementación basada en MDD, MDA e IFML es una buena opción para lograr interfaces de usuario adaptables y una rápida implementación de funcionalidades.

IFML es un lenguaje excelente para en la creación de interfaces de usuario efectivas, su uso permite modelar flujos de interacción a un nivel muy específico, simplificar el proceso de diseño y facilitar la adaptabilidad de las interfaces. La adopción de IFML potencia la creación de interfaces de usuario coherentes y centradas en el usuario, a pesar de los retos identificados en la integración y resolución de conflictos entre módulos.

Las herramientas MDD y MDA utilizadas permiten seleccionar entre diferentes opciones para configurar la aplicación que se genera, algunas de estas configuraciones pueden ser la base de datos con la que se conectará la aplicación, el lenguaje de programación con el que se generará el código, el framework con el que se trabaja ese lenguaje de programación, entre otras, esto permite a los usuarios seleccionar las opciones más convenientes de acuerdo a los requerimientos de la aplicación.

Dado que el código generado por las herramientas XOmega y AndroMDA están en lenguajes de programación muy conocidos como lo son c# y java, existe un buen soporte para su implementación, lo cual facilita realizar el despliegue de las aplicaciones para su publicación en internet con un dominio o dirección IP pública, además, existen diferentes proveedores que

permiten levantar una plataforma que soporte tanto la base de datos como la aplicación generada.

La generación de una aplicación web con herramientas MDD o MDA tiene varias ventajas como la rapidez con la que se puede crear la aplicación, la documentación previa de la aplicación que se va a generar, la facilidad con la que se pueden hacer cambios y cubrir nuevas funcionalidades y la facilidad de uso que tienen estas herramientas una vez que se tiene un conocimiento base de las mismas, pero también puede presentar aspectos que pueden ser considerados como desventajas de acuerdo a las necesidades del usuario o a los requerimientos del sistema, como las limitantes en la implementación de funcionalidades complejas, establecer flujos específicos o los cambios que se puedan necesitar en una funcionalidad que ya ha generado la aplicación.

Recomendaciones

Para lograr una experiencia de usuario óptima, se recomienda adoptar el lenguaje IFML como herramienta central para modelar las interfaces de usuario ya que proporciona una manera estructurada y comprensible de representar cómo los usuarios interactúan con la aplicación, además de definir de manera clara y detallada la secuencia de acciones, flujos de navegación y comportamientos esperados en la interfaz.

Es muy recomendable tener claro cuál es el entorno en el que se desea trabajar y los requerimientos que debería poder cubrir en el futuro la aplicación que se va a generar, ya que, a pesar de que estas herramientas permitan la selección entre diferentes especificaciones de la aplicación, una vez que se haya creado a diferencia de agregar nuevas funcionalidades, será muy difícil modificar alguna de las opciones de creación del proyecto que ya se ha seleccionado, incluso en un escenario como ese, se podría necesitar volver a crear el proyecto con las especificaciones correctas.

Durante la selección de la herramienta a utilizarse también se debe considerar la facilidad de despliegue que se tendrá en base a la configuración del proyecto, dado que, en el caso de ser necesaria la publicación de la aplicación, se tendrá un mayor soporte, facilidades y costes en la infraestructura que se requiere para este proceso.

Se debe considerar cada una de las ventajas y desventajas de generar una aplicación web con herramientas MDD o MDA antes de utilizarlas, ya que, si no se llegan a cumplir los requerimientos de la aplicación, se habrá invertido tiempo innecesariamente en el manejo de las herramientas y en la creación de los diagramas y modelos, por ejemplo, si la aplicación requiere de funcionalidades simples y manejo de varias entidades sería muy buena opción utilizar MDD o MDA, pero si la aplicación necesita la gestión de pocas entidades y debe ser modificada sería contraproducente utilizar alguna de estas herramientas, ya que se pasaría por la fase de documentación obligatoriamente y la aplicación podría ser muy robusta para su funcionalidad y por ende difícil de modificar.

Bibliografía

About the Interaction Flow Modeling Language Specification Version 1.0. (s. f.). Recuperado 9 de mayo de 2023, de https://www.omg.org/spec/IFML/1.0

Ahmad, S. I., Rana, T., & Maqbool, A. (2022). A Model-Driven Framework for the Development of MVC-Based (Web) Application. *Arabian Journal for Science and Engineering*, *47*(2), 1733-1747. https://doi.org/10.1007/s13369-021-06087-4

Amado, D. (2021). *Sistemas Colaborativos mediante el Desarrollo de Software Dirigido por Modelos*.

Ameller, D., Franch, X., Gómez, C., Martínez-Fernández, S., Araújo, J., Biffl, S., Cabot, J., Cortellessa, V., Fernández, D. M., Moreira, A., Muccini, H., Vallecillo, A., Wimmer, M., Amaral, V., Böhm, W., Bruneliere, H., Burgueño, L., Goulão, M., Teufl, S., & Berardinelli, L. (2021). Dealing with Non-Functional Requirements in Model-Driven Development: A Survey. *IEEE Transactions on Software Engineering*, *47*(4), 818-835. https://doi.org/10.1109/TSE.2019.2904476

Codina, L. (2018). *Revisiones bibliográficas sistematizadas*.

Da Silva, E., Maciel, R., & Magalhães, A. (2020). Integrating Model-Driven Development Practices into Agile Process: Analyzing and Evaluating Software Evolution Aspects: *Proceedings of the 22nd International Conference on Enterprise Information Systems*, 101-110. https://doi.org/10.5220/0009392501010110

Delgado Olivera, L. de la C., & Díaz Alonso, L. M. (2021). *Modelos de Desarrollo de Software*. *15*(1), 37-51.

Guamán Coronel, D. A. (2021). *Identificación uso y evaluación de herramientas de Model Driven Development MDD* [BachelorThesis]. http://dspace.utpl.edu.ec/jspui/handle/20.500.11962/28025

IEEE Recommended Practice for Software Requirements Specifications. (1998). *IEEE Std 830-1998*, 1-40. https://doi.org/10.1109/IEEESTD.1998.88286

IFML General Overview. (s. f.). http://www.webratio.com. Recuperado 6 de junio de 2023, de https://my.webratio.com/learn/learningobject/ifml-general-overview-v-72?link=ln174a

IFML primer | IFML: The Interaction Flow Modeling Language. (s. f.). Recuperado 17 de mayo de 2023, de https://www.ifml.org/ifml-primer/

Joo, H. (2017). *A Study on Understanding of UI and UX, and Understanding of Design According to User Interface Change*. *12*(20).

Kalnins, A., Kalnina, E., Sostaks, A., Celms, E., & Tabernakulovs, I. (2016). LINQ as Model Tansformation Language for MDD. *Baltic Journal of Modern Computing*, *4*. https://doi.org/10.22364/bjmc.2016.4.4.21

Kristiadi, D. P., Udjaja, Y., Supangat, B., Prameswara, R. Y., Warnars, H. L. H. S., Heryadi, Y., & Kusakunniran, W. (2017). The effect of UI, UX and GX on video games. *2017 IEEE International Conference on Cybernetics and Computational Intelligence (CyberneticsCom)*, 158-163. https://doi.org/10.1109/CYBERNETICSCOM.2017.8311702

Li, Z., Zhou, X., & Ye, Z. (2019). A Formalization Model Transformation Approach on Workflow Automatic Execution from CIM Level to PIM Level. *International Journal of Software Engineering and Knowledge Engineering*, *29*(09), 1179-1217. https://doi.org/10.1142/S0218194019500372

Llamuca-Quinaloa, J., Vera-Vincent, Y., Tapia-Cerda, V., Llamuca-Quinaloa, J., Vera-Vincent, Y., & Tapia-Cerda, V. (2021). Análisis comparativo para medir la eficiencia de desempeño entre una aplicación web tradicional y una aplicación web progresiva. *TecnoLógicas*, *24*(51), 164-185. https://doi.org/10.22430/22565337.1892

Mena Roa, M. (2021, agosto 6). *¿Cuántos sitios web hay en el mundo?* Statista Infografías. https://es.statista.com/grafico/19107/numero-de-sitios-web-existentes-en-internet

Microsoft 365 Team. (2019, septiembre 24). *La guía sencilla para la diagramación de UML y el modelado de la base de datos*. https://www.microsoft.com/es-ww/microsoft-365/business-insights-ideas/resources/guide-to-uml-diagramming-and-database-modeling

Nieves-Guerrero, C., Ucán-Pech, J., & Menéndez-Domínguez, V. (2014). UWE en Sistema de Recomendación de Objetos de Aprendizaje. Aplicando Ingeniería Web: Un Método en Caso de Estudio. *Revista Latinoamericana de Ingenieria de Software*, *2*(3), 137. https://doi.org/10.18294/relais.2014.137-143

OMG. (2017). Unified Modeling Language, v2.5.1. *Unified Modeling Language*.

OMG. (2023). *Model Driven Architecture (MDA)* | *Object Management Group*. https://www.omg.org/mda/

Pachay Quiñónez, M. E. (2019). *Generación de una APP Web dirigida por modelos* [Thesis, Ecuador - PUCESE - Escuela de Sistemas y Computación]. http://localhost/xmlui/handle/123456789/1892

Palomo, S. R. G., & Gil, E. M. (2020). *Aproximación a la ingeniería del software*. Editorial Centro de Estudios Ramon Areces SA.

Pan, M., Lu, Y., Pei, Y., Zhang, T., Zhai, J., & Li, X. (2020). Effective testing of Android apps using extended IFML models. *Journal of Systems and Software*, *159*, 110433. https://doi.org/10.1016/j.jss.2019.110433

Panach, J. I., Dieste, Ó., Marín, B., España, S., Vegas, S., Pastor, Ó., & Juristo, N. (2021). Evaluating Model-Driven Development Claims with Respect to Quality: A Family of Experiments. *IEEE Transactions on Software Engineering*, *47*(1), 130-145. https://doi.org/10.1109/TSE.2018.2884706

Panach, J. I., Pastor, Ó., & Juristo, N. (2022). A Family of Experiments to Compare Two Model-Driven Development Tools vs a Traditional Development Method. *IEEE Transactions on Software Engineering*, *48*(12), 4802-4817. https://doi.org/10.1109/TSE.2021.3127350

Parada, W. V., Giraldo, F. A., & Londoño R, M. I. (2019). MDD based case tool for Automatic Generation of ChatBot. *2019 XLV Latin American Computing Conference (CLEI)*, 1-7. https://doi.org/10.1109/CLEI47609.2019.235059

Parra Arévalo, J. P. (2020). Análisis del desarrollo de software en no desarrolladores. *Editorial Universitaria San Mateo*.

Pinzon, O. (2017). *Ingeniería Web: Una Metodología para el Desarrollo de Aplicaciones Web Escalables y Sostenibles*.

Pons, C., Giandini, R., & Pérez, G. (2010, febrero 22). *Desarrollo de Software dirigido por modelos: Conceptos teóricos y su aplicación práctica*. https://ri.conicet.gov.ar/handle/11336/127598

Premier Diagramming, Modeling Software & Tools. (s. f.). Astah. Recuperado 28 de junio de 2023, de https://astah.net/

Priefer, D., Rost, W., Strüber, D., Taentzer, G., & Kneisel, P. (2021). Applying MDD in the content management system domain: Scenarios, tooling, and a mixed-method empirical assessment. *Software and Systems Modeling*, *20*. https://doi.org/10.1007/s10270-021-00872-3

Quintero, J. B., & Anaya, R. (2007). MDA Y EL PAPEL DE LOS MODELOS EN EL PROCESO DE DESARROLLO DE SOFTWARE. *Revista EIA*, *8*, 131-146.

Roig Hervás, D. (2021). Tecnologías Low-Code y No-Code: Un caso práctico para estudiar su potencial y limitaciones. *Ingeniería del agua*, *18*(1), ix. https://doi.org/10.4995/ia.2014.3293

Sajji, A., Rhazali, Y., & Hadi, Y. (2022). A methodology for transforming BPMN to IFML into MDA. *Bulletin of Electrical Engineering and Informatics*, *11*(5), Article 5. https://doi.org/10.11591/eei.v11i5.3973

Sajji, A., Rhazali, Y., & Hadi, Y. (2023). Graphical User Interfaces Generation from BPMN (Business Process Model and Notation) via IFML (Interaction Flow Modeling Language) up to PSM (Platform Specific Model) Level. *International Journal of Advanced Computer Science and Applications (IJACSA)*, *14*(2), Article 2. https://doi.org/10.14569/IJACSA.2023.0140251

Sebastián, G., Gallud, J. A., & Tesoriero, R. (2020). Code generation using model driven architecture: A systematic mapping study. *Journal of Computer Languages*, *56*, 100935. https://doi.org/10.1016/j.cola.2019.100935

Sharma, P. (2021, noviembre 12). Costo de desarrollo de aplicaciones web: Desglose de precios de 2021. *Cynoteck*. https://cynoteck.com/es/blog-post/web-app-development-cost/

Sharma, V., & Tiwari, A. K. (2021). *A Study on User Interface and User Experience Designs and its Tools*. *12*(6).

Silega, N., Noguera, M., Rogozov, Y. I., Lapshin, V. S., & González, T. (2022). Transformation From CIM to PIM: A Systematic Mapping. *IEEE Access*, *10*, 90857-90872. https://doi.org/10.1109/ACCESS.2022.3201556

Soude, H., & Koussonda, K. (2022). A Model Driven Approach for Unifying user Interfaces Development. *International Journal of Advanced Computer Science and Applications*, *13*(7). https://doi.org/10.14569/IJACSA.2022.01307107

Su, Z., Wang, D., Yang, Y., Yu, Z., Chang, W., Li, W., Cui, A., Jiang, Y., & Sun, J. (2021). MDD: A Unified Model-driven Design Framework for Embedded Control Software. *IEEE*

Transactions on Computer-Aided Design of Integrated Circuits and Systems, *PP*, 1-1. https://doi.org/10.1109/TCAD.2021.3132564

Tatnall, A. (Ed.). (2020). *Encyclopedia of Education and Information Technologies*. Springer International Publishing. https://doi.org/10.1007/978-3-030-10576-1

Torres, V., Fabregat, R., & Rodríguez-Abitia, G. (2018). *Comercio Electrónico Móvil en México y España*.

What is UML | Unified Modeling Language. (2023). https://www.uml.org/what-is-uml.htm

Wijekoon, H., & Merunka, V. (2022). *Combining Model Driven Development and Agile Software Development*.

Yigitbas, E., Jovanovikj, I., Biermeier, K., Sauer, S., & Engels, G. (2020). Integrated model-driven development of self-adaptive user interfaces. *Software and Systems Modeling*, *19*(5), 1057-1081. https://doi.org/10.1007/s10270-020-00777-7

Yousaf, N., Azam, F., Butt, W. H., Anwar, M. W., & Rashid, M. (2019). Automated Model-Based Test Case Generation for Web User Interfaces (WUI) From Interaction Flow Modeling Language (IFML) Models. *IEEE Access*, *7*, 67331-67354. https://doi.org/10.1109/ACCESS.2019.2917674