UNIVERSIDAD DE TALCA
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA CIVIL EN COMPUTACIÓN

Herramienta para la comparación de modelos de Proceso de software(ProMoCoT)

ARIEL ANDRÉS CORNEJO GONZÁLEZ

Profesor Guía: LUIS GREGORIO SILVESTRE QUIROGA

Memoria para optar al título de
Ingeniero Civil en Computación

Curicó – Chile
Julio, 2020

CONSTANCIA

La Dirección del Sistema de Bibliotecas a través de su encargado Biblioteca Campus Curicó certifica que el autor del siguiente trabajo de titulación ha firmado su autorización para la reproducción en forma total o parcial e ilimitada del mismo.

Curicó, 2022

Dedicado a mi Madre Gloria y Mi tía Luisa

AGRADECIMIENTOS

Siempre recuerdo los sueños, pensar lo que quería ser en el futuro y ver siempre cada uno de ellos de manera distante y lejana. Con el paso de los años, comencé a sorprenderme cuando vi que esos sueños cada día se tornaban una realidad, el salir de mi pueblo natal para estudiar en la ciudad, entrar a la universidad y finalmente uno de mis logros más anhelados, el conseguir la profesión que tanto quise por muchos años. Por esto mismo luego de tantas cosas buenas que me ha dado la vida, es necesario agradecer a todos y cada uno de los que en este largo viaje, han estado a mi lado y me han ayudado en todos los aspectos que se puedan.

Partir estos agradecimientos, por aquellos que siempre me han acompañado, mis amigos. Siempre he dicho que una de las cosas mas bellas que tenemos los seres humanos, es el poder establecer relaciones con otros, por lo mismo doy gracias a cada uno de ellos por los buenos momentos, cada memoria, la ayuda y su motivación que cada día me dieron para seguir adelante. Nunca podre dejar de agradecerles por dejarme ser parte de su vida.

De igual manera agradecer a aquellos que me ayudaron como mentores, tanto en el conocimiento, así como en mi formación como persona, mis profesores. Dar las gracias a mis profesores de básica que motivaron a seguir estudiando, a los de media que me dieron el fervor para seguir alcanzando mis metas y a mis mentores de la universidad, siendo estos últimos los que finalmente moldearon mi visión y más allá de solo conocimiento técnico, me entregaron los pilares finales para establecerme como el profesional que soy hoy. Decir que en especial agradezco a mi profesor guía, sin el esto no estaría siendo escrito, le agradezco su buena disposición, cercanía y por sobre todo el compartir conmigo y todos sus alumnos su sabiduría así como su devoción por el conocimiento, muchas gracias profesor Luis por haberme dejado compartir esta etapa tan crucial con usted.

Dar también agradecimientos, a las personas con las cuales comparto mis lazos más cercanos, mi familia. No imagino el estar en este lugar sin ustedes, cada momento que he compartido desde mi nacimiento ha sido de los mas gratos que puedan existir, su apoyo es el que me siguió moviendo y me hizo creer en mismo, así como formo cada uno de los valores que hoy puedo poner en practica como un profesional. Agradecer en especial a mi madre y mi tía, quienes fueron mis figuras a seguir durante muchos

años, las que me despertaban en la mañana para ir al colegio y en la tarde siempre tenían un abrazo calidad para recibirme. Muchas gracias a la vida por haberme puesto en la vida de dos personas tan maravillosas. Agregar que esto no solamente toma la familia que es unida por lazos de sangre, si no todos aquellos con los que comparto un vinculo tan importante como para considerarlos miembros de la misma.

Finalmente, pero no menos importante agradecerte a ti, quien esta leyendo esta memoria ya sea por conocimiento, para usar de modelo o simplemente curiosidad. Esperar que la disfrutes y decirte que sigas con tus sueños, que de primera fuente confirmo que se pueden alcanzar.

TABLA DE CONTENIDOS

página

ÍNDICE DE FIGURAS

página

ÍNDICE DE TABLAS

página

RESUMEN

En la actualidad, las empresas especializadas en el desarrollo de software realizan proyectos de índoles diferentes, pudiendo ser un software nuevo, una actualización o simplemente dar soporte a algún software ya lanzado. Para lograr esto es necesario que los equipos del las empresa sigan un determinado modelo de proceso de software.

De manera mas precisa, dichas empresas desarrollan Lineas de Proceso de Software (SPrL), que se originan de la derivación de un proceso base cuyo objetivo es abordar distintos tipos de proyectos. Los modelos deben pasar a ser especificados en herramientas dedicadas, para que así puedan ser abstraídos a un nivel mas alto. EPF Composer es una de estas, permitiendo gestionar la creación de procesos de software de manera gráfica y siguiendo un estándar estructural.

Usualmente cuando un modelo de proceso de software sufre demasiadas modificaciones, es complejo determinar su variación respecto al proceso base o algún proceso estándar. Todo ello con el objetivo de reparar, mejorar u optimizar una variación de proceso. Actualmente, existen herramientas que suelen ser efectivas para aplicar comparaciones entre procesos pero basta una mínima perturbación a nivel de código para que los resultados dejen de ser satisfactorios.

Con el objetivo de abordar dicha problemática, se construye herramienta llamada *ProMoCot* que permite realizar la comparación entre modelos de proceso de software a nivel visual. ProMoCot se divide en dos partes fundamentales, un comparador y la plataforma web, que funcionan de manera conjunta.

Para fundamentar ProMoCot se aplica la metodología de investigación *Snowballing* para establecer el estado del arte sobre los métodos de comparación de modelos. Para el desarrollo de ProMoCot se aplica la metodología de desarrollo *Personal Extreme Programming* debido su orientación individual. Para la evaluación del proyecto se usa experimentación en Ingeniería de Software con el objetivo de evaluar su funcionalidad, corrección y utilidad.

Los resultados de la investigación sugieren que ProMoCot permite importar, visualizar y comparar modelos de proceso en un entorno gráfico sin la necesidad que un usuario conozca la formalidad de ingeniería dirigida por modelos. Además ProMoCot proporciona resultados más precisos y esperados que otras herramientas.

1. Introducción

En este capítulo se muestran los conceptos necesarios para introducir el proyecto. Se consideran aspectos fundamentales como los son el contexto y la problemática en la cual se desarrolla el proyecto. Junto con lo anterior se hace mención a la propuesta de solución que se tiene así como sus alcances y los objetivos que se pretenden cumplir con esta.

1.1. Contexto del Proyecto

En la actualidad en el ámbito de desarrollo de software, las empresas definen sus procesos para gestionar sus proyectos de desarrollo de forma sistemática, pudiendo planificar, asignar recursos y controlar el progreso de un proyecto. Los procesos de software se definen a partir de elementos simples llamados elementos del proceso de software. Los nombres de elementos de proceso de software dependen del vocabulario; por ejemplo, rol se puede usar para representar a alguien que realiza una tarea o actividad cuyo resultado es un artefacto o documento. La definición formal del proceso de software permite a las empresas documentar rigurosamente su soporte de procesos y herramientas para el análisis y la gestión formales, además de facilitar la evolución del mismo.

Las compañías de software se involucran en diferentes tipos de proyectos; por ejemplo, desarrollo de software desde cero, desarrollo con un equipo experimentado o novato, desarrollo con tecnologías conocidas o altamente innovadoras, entre otros. Por lo tanto, el mismo proceso no suele ser apropiado para abordar todo tipo de proyectos, o no es igualmente productivo y efectivo en todos los casos. Si el proceso de software definido no se entiende y aplica fácilmente, invertir tiempo y dinero en

su definición no es productivo para las compañías de software. Por lo tanto, una conclusión obvia es que es necesario contar con una serie de procesos, cada uno para cada tipo de proyecto, es decir, una familia de procesos.

En este escenario, las líneas de proceso de software (SPrL) aparecen como un enfoque potencial para enfrentar este desafío. Un SPrL es una línea de productos de software donde los productos son procesos de software. De esta manera, SPrL define un proceso reutilizable y un mecanismo para obtener procesos particulares; este mecanismo corresponde a la adaptación de procesos. Las técnicas MDE, siendo este el paradigma de ingeniería orientada a modelos [39], pueden contribuir al desarrollo de SPrL, esto se ve directamente reflejado en la capacidad de crear bibliotecas de activos reusables, los cuales puede ser usados en las especificaciones de próximas líneas de proceso de software [30].

En el paradigma de MDE, el modelado de procesos de software se refiere a la definición de los procesos como modelos, mas cualquier soporte automatizado opcional disponible para modelar y ejecutar los modelos durante el proceso de software. Un modelo de proceso de software es una representación abstracta de un proceso de software desde una perspectiva particular. Proporciona definiciones del proceso de software que se utilizará, instanciará, promulgará o ejecutará. Por lo tanto, un modelo de proceso de software se puede analizar, validar, simular o ejecutar si se define adecuadamente de acuerdo con estos objetivos. Acuña[1] define un modelo de proceso como la descripción de un proceso expresado en un lenguaje de modelado de proceso adecuado. Con este fin, el Grupo de Gestión de Objetos (OMG) propuso la especificación del Metamodelo de Ingeniería de Procesos de Software y Sistemas 2.0 (SPEM 2.0) que es un estándar para el modelado de procesos de software [17].

Dentro del ámbito de trabajo de MDE existen varias herramientas de software para la abstracción de procesos de software hacia modelos, de los cuales cabe destacar la herramienta EPF Composer [23]. Esta herramienta pertenece a la Eclipse Foundation, siendo parte de la gran variedad de proyectos que existen dentro del Framework Eclipse. Entre sus principales características esta la facilidad con la cual permite la abstracción de un determinado proceso de software, todo esto a través del uso de una interfaz gráfica de usuario para especificar el proceso de software y que considera aspectos fundamentales del modelado de procesos establecido por SPEM 2.0.

1.2. Presentación del Problema

En la actualidad el proceso de software puede presentar variantes dependiendo del ámbito en el cual es aplicado, siendo los más comunes el académico y el empresarial. Una variante de proceso o también conocida como proceso adaptado corresponde a la evolución a la cual puede verse sometida el proceso base base del cual surgió, lo cual también puede ser conocido como generar líneas de proceso de software. La evolución de las variantes provoca diferencias entre estas, haciendo que se desconozca el nivel de similitud que comparten.

Un proceso base usualmente conforma un estándar, por ejemplo RUP [40], Cascada [7], Espiral [11] entre otros; sin embargo la aplicación práctica de estos genera variantes propias de cada ámbito de aplicación. En este sentido, surge el problema de identificar el proceso base de una determinada variante de proceso. La identificación de un proceso base puede ser realizado utilizando distintas técnicas, por ejemplo: comparación textual de proceso, comparación estructural de proceso, comparación de especificación de proceso entre otros. Sin embargo, un proceso de software puede ser especificado de distintas formas utilizando distintas herramientas. Por lo tanto, la comparación de procesos puede ser compleja debido a la variabilidad de herramientas y especificaciones de procesos que existen.

Recientemente, MDE ha sido utilizado en el ámbito de la especificación de procesos a partir de modelos y transformaciones [31]. MDE brinda la posibilidad de abstraer todas las variantes de especificación de proceso en una única estructura estándar de modelo. El modelo de proceso estándar conforma a SPEM cuya especificación puede ser comparable entre modelos. Obviamente la comparación de modelos de proceso no es trivial porque se trabaja con especificaciones SPEM y herramientas propias de MDE que son, en la práctica, poco aplicables para usuarios finales [59]. Además, se debe considerar que existen muchos desafíos en la comparación de modelos que las herramientas especializadas no consideran, por ejemplo: semántica del lenguaje, sintaxis del lenguaje, similitud del lenguaje.

Para orientar la investigación que se llevará a cabo, se define una pregunta de investigación que se pretende responder. La pregunta es :

- ¿Qué estrategia de comparación puede aplicarse entre procesos de software?

En primer lugar se pretende llevar a cabo una comparación de los procesos de software adaptados que son aplicados en cada una de las diferentes empresas. Esto se

pretende llevar a cabo en base a los resultados de las comparaciones que se efectúen entre el proceso adaptado y otros procesos de software que pueden ser estándares o particulares a otras empresas, para de esta manera ilustrar al usuario acerca de cuales son los cambios que ha sufrido su adaptación.

De manera consecuente y para poder llevar a cabo dichas comparaciones, es necesario estipular las estrategias que permitan realizar dicho proceso, es decir, diseñar los pasos que se deben tomar para la comparación de los modelos.

1.3. Objetivos

Objetivo general

- Desarrollar una solución para comparar modelos de proceso de software que permita la identificación de las diferencias explicitas de cada uno.

Objetivos específicos

1. Aplicar un método para la estandarización de modelos de proceso de software, haciendo uso de un extractor e inyector.
2. Caracterizar los principales conceptos del proceso de software e ingeniería dirigida por modelos.
3. Definir un método para la comparación de modelos.
4. Desarrollar una herramienta de software para llevar a cabo la gestión de modelos.
5. Evaluar el desempeño de la herramienta al momento de realizar comparaciones.

1.4. Propuesta de Solución

Para dar solución al problema planteado se propone definir un método para comparar modelos mediante tres diferentes técnicas:

- Comparación estructural de modelos: Esta comparación se refiere a buscar similitudes en la estructura de los modelos comparados, es decir, identificar que componentes están presentes o no en cada uno de los modelos [45].

- Comparación léxica de modelos: Consiste en la búsqueda de similitudes directas en los nombres de los componentes del modelo, es decir una comparación directa de las cadenas de texto [26].
- Comparación de sinonimia de modelos: Consiste en la comparación a nivel de significado de cada una de las palabras que componen los nombres de las partes del modelo [24].

Para llevar a cabo cada uno de las anteriores estrategias, se pretende desarrollar un prototipo, el cual consistirá en una herramienta web que permitirá realizar las siguientes acciones:

- Cargar un modelo de proceso de software exportado desde la herramienta EPF Composer.
- Generar un modelo equivalente especificado completamente en SPEM.
- Comparar el modelo generado con otros de la misma estandarización.
- Comparar este modelo con un modelo estándar o base.
- Visualizar los resultados de la comparación.

1.5. Alcances

Los alcances a considerar en este proyecto son los siguientes:

- En este proyecto se pretende realizar una herramienta de software orientada a una plataforma web, por lo tanto no se espera dar soporte a dispositivos móviles u otras plataformas.
- La comparación de modelos se limitará a la aplicación de tres estrategias: comparación estructural, léxica y de sinonimia de textos.
- Las comparaciones de modelos se limitan a tareas, fases, actividades y disciplinas de estos mismos, no se ejecutarán comparaciones en ningún otro aspecto.
- Los nombres de los atributos deben ser compuestos por palabras en español.

- La herramienta solo aceptará procesos que estén especificados EPF Composer.
- La herramienta hará uso de un extractor para estandarizar los modelos y visualizarlos , no se dará la opción de volver al formato original en el cual venía el proceso.
- La herramienta manipulará modelos de proceso en formato XMI.

1.6. Trabajos Relacionados

A continuación se mostraran trabajos que se relacionan con el contexto del desarrollo del proyecto, los cuales son EMF Compare que consiste en una opción directa al proyecto que se esta llevando a cabo y Textual Concrete Syntax en cuanto a lo que conlleva la acción de comparación directamente.

1. TCS (Textual Concrete Syntax): Este método es utilizado para lograr una estandarización en cuanto a los lenguajes de dominio específico o DSL por sus siglas en ingles, evitando ambigüedades en su sintaxis o similares. Dichos lenguajes son aquellos que son usados para tareas o problemáticas muy especificas, es decir que no son multiproposito [62]. Dentro de MDE el método brinda la opción de lograr modelos completamente estandarizados los cuales poseen menos ambigüedades sintácticas. Pese a los beneficios que entrega, se ve limitado en ciertos aspectos como: una gramática limitada, la generación de errores o el análisis léxico [37].
2. EMF Compare: Herramienta ubicada dentro del entorno de Eclipse, esta permite el realizar comparaciones directas con modelos trabajados en el software Eclipse, obteniendo de esta manera los resultados directos de cuán similares son los modelos en comparación o cualquier procedimiento de esta índole. Pese a que permite comparación directa no es capaz de detectar ambigüedades en el proceso, es decir si existe una fase de nombre similar que apunta al mismo concepto no será capaz de realizar un emparejamiento con su más cercana [21].

2. Antecedentes

En este capítulo se describen los antecedentes correspondientes al contexto del proyecto. Cada apartado es un concepto fundamental dentro del contexto del proyecto, además se introduce conceptos relativos a los apartados para definir cada uno de ellos para poder dar una visión mas profunda del contexto del proyecto.

2.1. Procesos de software

En esta sección se define los conceptos relativos directamente con el proceso de desarrollo de software, para de esta manera introducir los conceptos base necesarios para la entrar en la temática de modelado de proceso de software. A continuación se definen los conceptos clave que rodean esta área.

2.1.1. Proceso de Software

El concepto de proceso de software hace referencia a un conjunto de actividades que al ejecutarlas de una manera secuencial especifica logran como resultado un producto de software en particular. Dichas actividades pueden ser llevadas a cabo para generar un producto de software desde cero, sin embargo en la actualidad muchas de estas son derecha mente ocupadas para realizar modificaciones o ampliaciones de productos ya existentes [57].

Pese a tener una gran variedad en cuanto a como se especifican a estos procesos, se puede identificar algunas actividades fundamentales presentes en cada uno de ellos.

- Especificación del software: Se define la funcionalidad y las restricciones que rodearan al software.

- Diseño e implementación del software: Se realizan las actividades para asegurarse la producción de un software que cumpla con todas las especificaciones realizadas.
- Validación de Software: Se realiza una validación de las funcionalidades del software con el propósito de asegurar que este cumple con lo que el cliente desea.
- Evolución del software: Se considera como una etapa post lanzamiento del producto, donde este se vera sujeto a cambios dependiendo de las necesidades que vallan surgiendo a futuro.

2.1.2. Ciclo de vida de proceso de software

El termino modelo de proceso de software hace referencia a un tipo de descripción simplificada del proceso de software, lo cual consiste en presentar una visión mas a alto nivel del mismo para así poder facilitar su compresión y la manipulación que se le da a este. Normalmente estos modelos son compuestos por varios elementos comunes, como lo son actividades, productos y los papeles refiriendo este ultimo a quienes llevan cabo una determinada actividad dentro del proceso [57]. Gran parte de las estructuras de los modelos tradicionales realizados pueden ser englobadas dentro de 2 estructuras generales:

- Enfoque cascada: Se presentan cada una de las actividades típicas del proceso de software y estas se presentan en fases. Se necesita dar por finiquitado completamente una fase previa para pasar a la siguiente. En la figura 2.1 se ilustra como se relacionan las fases, además de mostrar como es la movilidad dentro de estas.

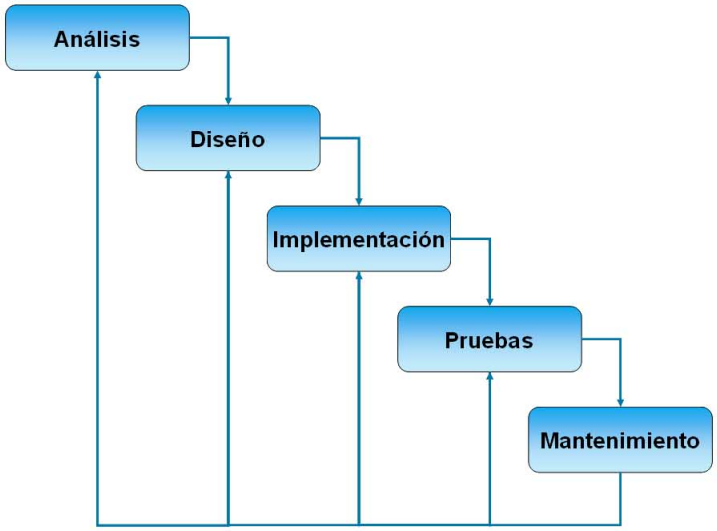


Figura 2.1: Ejemplo de modelo en cascada

- Desarrollo Iterativo: Se le da un enfoque a las actividades de especificación, desarrollo y validación. Se comienza con un prototipo desarrollado en base a especificaciones abstractas, para luego irse refinando a medida que se valla pasando durante mas iteraciones. En la figura 2.2 se ilustra como las actividades e iteraciones se distribuyen y los diferentes prototipos que pueden ser obtenidos a través de estos.

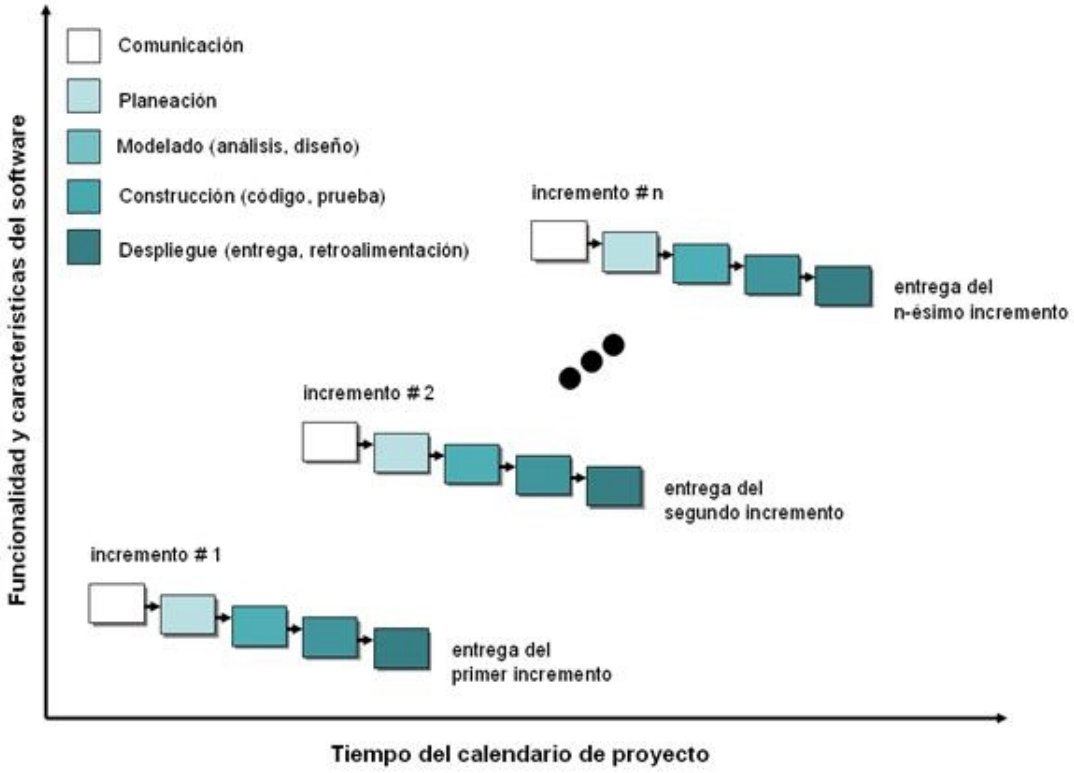


Figura 2.2: Ejemplo de modelo iterativo incremental

Además de los ciclos de vida utilizados para las metodologías ágiles, también es necesario considerar los que corresponde a las metodologías ágiles, como los siguientes ejemplos:

- Scrum: Corresponde a un marco para el desarrollo agil de software, el cual no solamente se limita a este exclusivo medio. Este tipo de proceso privilegia el trabajo colaborativo y en equipo considerando que es la mejor manera para obtener buenos resultados, se incorpora una estrategia de desarrollo incremental y se solapan las diferentes fases de desarrollo, en lugar de realizar un ciclo secuencial [55]. La Figura 2.3 representa el ciclo convencional utilizado en Scrum.

- XP: XP o programación extrema por sus siglas en ingles es un tipo de metodología ágil, la cual se caracteriza principalmente para ser responsiva frente a los cambios de requisitos que tenga el cliente. Además también se da énfasis

Figura 2.3: Ejemplo de metodologia Scrum

en la entrega de pequeños lanzamientos en ciclos de desarrollo cortos y privilegiar la aplicación de pruebas en el código [43]. La Figura 2.4 muestra el ejemplo convencional del ciclo seguido por la metodología para realizar sus lanzamientos.

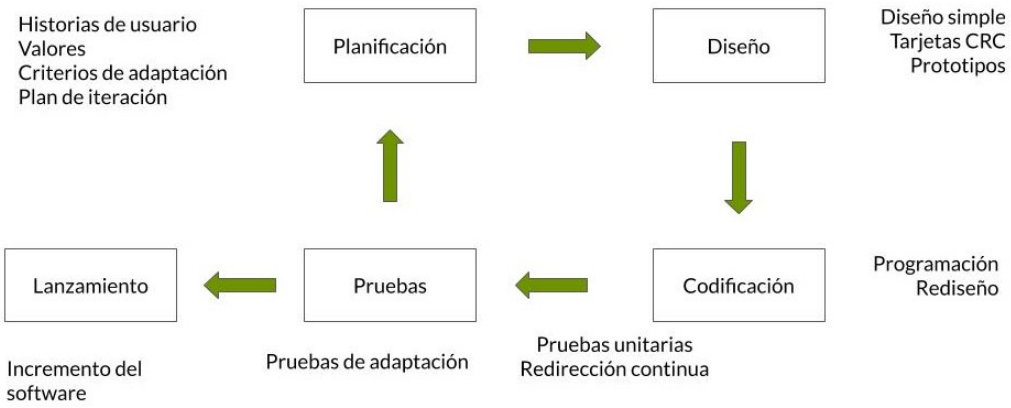


Figura 2.4: Ejemplo de modelo XP

2.2. Modelos de proceso de software

En la presente sección se explican conceptos relacionados con el modelado de procesos de software. Si bien durante la sección anterior se dio a conocer el termino de modelo de proceso de software, es necesario explicar como este es llevado a cabo,

tomando en cuenta las especificaciones de lenguaje y las herramientas utilizadas comúnmente para llevar a cabo dicho tipo de tareas. A continuación se mencionan los conceptos clave en cuanto a lenguajes y herramientas que son utilizados.

2.2.1. Software Process Engineering Metamodel - SPEM (Metamodelo de Ingeniería de Procesos de Software)

Lenguaje del paradigma MDE utilizado para la escritura de los modelos y metamodelos para procesos de software. SPEM es una especificación de OMG [49],que permite representar una familia de procesos de desarrollo de software y sus componentes. Constituye un tipo de ontología de procesos de desarrollo de software. Para ello provee un conjunto de elementos de modelado de procesos para describir cualquier proceso de desarrollo de software. SPEM proporciona una sintaxis y estructura para cada aspecto de los procesos desarrollo, incluyendo:

- Roles.
- Tareas.
- Artefactos.
- Lista de verificación
- Productos de trabajo.
- Técnicas y herramientas.
- Estructuras de trabajo.

2.2.2. Software Process Line - SPrL (Línea de proceso de software)

Deriva de la línea de producto de software, cuyo fin es generar variaciones desde un proceso de software base. Las variaciones del proceso usualmente son determinadas mediante un contexto de proyecto. El contexto de proyecto describe características de un proyecto que son considerados para adaptar un proceso de software. Algunas características de proyecto pueden ser: Tipo de proyecto, Duración del proyecto, Experiencia del equipo, Tamaño del equipo de desarrollo de software, entre otras [6].

2.2.3. EPF Composer

Herramienta de software utilizado para la creación y manipulación de modelos que actúan bajo el paradigma de MDE, hace uso del lenguaje SPEM en conjunto a elementos propios de la herramienta para la realización de los productos, los cuales resultan en un lenguaje llamado UMA, cuyo nombre proviene de la frase "Método de arquitectura unificada", siendo esto un meta-modelo de cómo el software estructura sus métodos y procesos [23]. La herramienta permite el uso de una interfaz gráfica para el modelaje, dando la posibilidad de que el usuario interactúe directo con esta. La figura 2.5 representa un extracto de un modelo realizado haciendo uso de la herramienta EPF, en el cual se puede apreciar la representación de las tareas de que serán llevadas a cabo dentro del proceso de desarrollo de software que es mostrado a través del modelo. Otra característica que se muestra es como es la estructura secuencial con la cual se ejecutan las tareas dentro del modelo. [20].

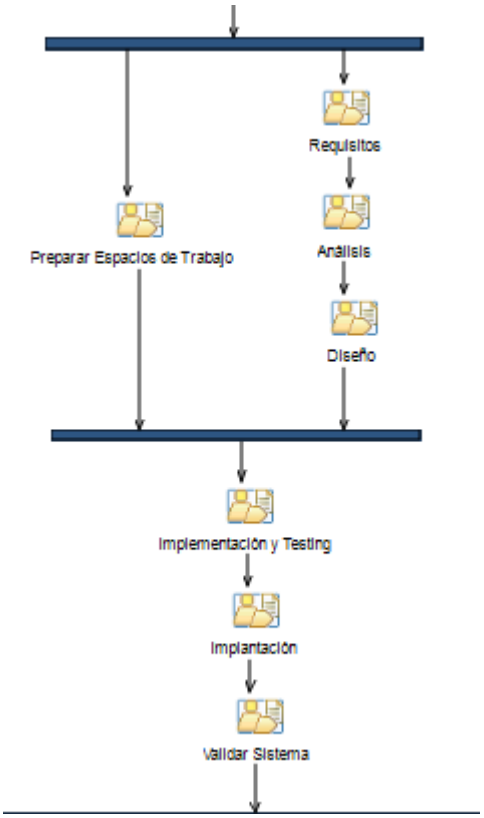


Figura 2.5: Ejemplo de modelo de proceso en EPF [9]

Existen otras herramientas como Magic Draw[1] o Enterprise Architect[2], las cuales también puede ser usadas para hacer tareas relacionadas con modelos, sin embargo estas no son software libre, puesto que piden una unas subscripción de pago, por lo tanto no se considero usar estas dentro del contexto del proyecto.

2.2.4. Model Driven Enginering - MDE (Ingenieria Orientada a Modelos)

Paradigma de ingeniería de software, el cual se centra en la creación y explotación de modelos de dominio (es decir, representaciones abstractas de los conocimientos y actividades que rigen un dominio de aplicación particular). La abstracción de un determinado proceso para su manipulación ayudando a comprender de manera tanto visual como escrita el uso de estos [54]. La figura 2.6 muestra los tres niveles que componen MDE. El nivel 0 se identifica un objeto o artefacto del mundo real. En el nivel 1 se representa el objeto o artefacto a través mediante la abstracción a un modelo. EL nivel 2 define un conjunto de características básicas que permiten crear modelos del nivel 1. El conjunto de características básicas del nivel 2 usualmente es llamado metamodelo.

[1] https://www.magicdraw.com
[2] https://sparxsystems.com/products/ea/

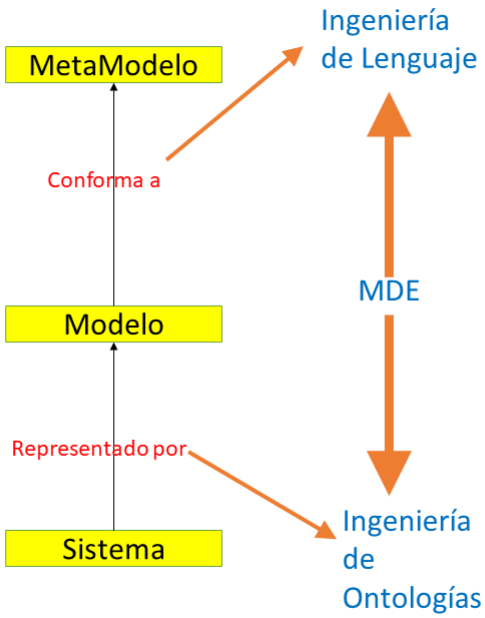


Figura 2.6: Diagrama de niveles de abstracción de MDE. Modificado del trabajo de Bezivin [9]

2.3. Proceso de software a Modelo de Software - Inyector

El extractor inyector es un programa especificado en el lenguaje Java y compilado dentro de un ejecutable de extension jar[3]. Este cumple la función de transformar procesos exportados desde la herramienta EPF Composer. Esto se ve reflejado a través de la transformación de la especificación otorgada en un archivo en formato XML el cual se especifica en SPEM y UMA, hacia una estandarización de formato XMI, la cual solo esta especificada bajo el lenguaje SPEM, eliminando los añadidos gráficos de UMA [8].

2.4. Comparación de modelos y texto

Para esclarecer de mejor manera parte del contexto central del proyecto, se introducen además los conceptos de comparación de modelos y texto. En estos se detalla

[3]https://docs.oracle.com/javase/1.5.0/docs/guide/jar/jar.html

a manera general algunos tipos de estrategias usadas para esta clase de acciones.

2.4.1. Comparación de modelos

La comparación de modelos se refiere al calculo de diferencias y similitudes que puedan existir entre dos modelos de software concretos. Esta acción normalmente es llevada a cabo a través del método de análisis de estructura sobre modelos que compartan una similar, dado a que presenta una manera mas fácil de abordar. Dentro del concepto anterior muchas veces también se lleva a cabo la comparación con el propósito de saber cuales han sido las modificaciones que ha sufrido un determinado modelo con el paso del tiempo, además de comparar los resultados que se esperaban para la estructura del modelo en contraste con los actualmente presentes [45]. Algunas de las estrategias algorítmicas para llevar a cabo la comparación de modelos son las siguientes.

- Algoritmos que se encargan de encontrar diferencias y similitudes dentro de los archivos XML, en los cuales estén expresados los modelos. En este tipo de método se pretende hacer un análisis jerárquico dentro del código en el cual este escrito el modelo. Este análisis jerárquico se hará descendiendo a través de todos los elementos contenidos [13]. Cabe mencionar que usualmente se usa el tipo de archivos XML para la representación de estos modelos. Se utiliza este lenguaje ya que por definición este funciona para definir cualquier lenguaje de marca, es decir que utiliza algún tipo llaves o características similares [12].

- Algoritmos de cálculo de diferencias, basados en como se abordan los componentes de cada uno, en este caso se representa en base a operaciones similares a las aritméticas donde podemos obtener cada una de las partes del modelo como un posible de la operación, siendo el fin obtener de una operador $\triangle$, el cual contenga las diferencias presentes entre ambos modelos, similar a como seria el ver cada modelo bajo la teoría de conjuntos [3]. En la figura 2.7 se puede ve como es aplicada la operación, es apreciable como los elementos son restados directamente desde cada conjunto.

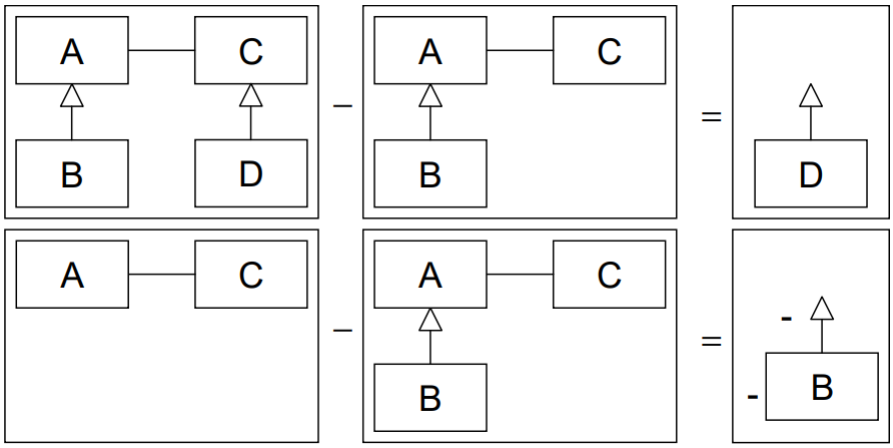


Figura 2.7: Ejemplo de diferencia de modelos [3]

2.4.2. Comparación de texto

El concepto de de comparación de textos, tiene una directa incidencia dentro el ámbito de la similitud textual. La similitud textual hace referencia a la comparación de textos, esto con el propósito de conocer la similitud de significado que existe entre ellos, de la misma manera se puede realizar esta actividad con variantes para la comparación de textos mas cortos, sin embargo estos últimos presentan un nivel de dificultad mas elevado para ser llevados a cabo [4].Para dar una formalización al proceso de similitud textual o comparación texto, se llevaron a cabo varios modelos formales de similitud, de los cuales cabe destacar los siguientes:

- Modelo Geométrico: En este caso las palabras son representadas como objetos en un determinado espacio geométrico, con el fin de poder manipular cada una de estas basándose en propiedades geométricas de las mismas [64].

- Modelo teoría de conjuntos: Se representa las características de los objetos a comparar, en este caso las frases o el texto completo, aplicando de esta manera una función que es capaz de decir que elementos esta presentes en ambos conjuntos, así como los que solo son propios de cada uno en particular [60].

3. Marco Metodológico

En el siguiente capítulo se explican las diferentes metodologías que son utilizada para el desarrollo del proyecto. Además dar a conocer la explicación teórica acerca del proceso que consta a cada una de estas metodologías, también se hace alusión a como son adaptadas dentro del contexto del proyecto, considerando las modificaciones necesarias que necesitan ser llevadas a cabo para el correcto funcionamiento de cada una. Las metodologías a presentarse son Snowballing para la etapa de investigación en torno a la creación del comparador, PXP para el desarrollo de la herramienta web y finalmente la Metodología de evaluación experimental en Ingeniería de Software para lograr llevar a cabo las pruebas necesarias dentro del software.

3.1. Metodología de investigación Snowballing

La metodología de investigación que se aplica para la recolección de información literaria necesaria para el diseño y desarrollo del comparador corresponde a Snowballing. El funcionamiento del método consiste en llevar a cabo un estudio desde una fuente primaria y de esta comenzar a ahondar en las fuentes que sirvieron para la escritura de la misma, haciendo esto para profundizar en los conocimientos que cada una trata de exponer dentro de su contenido [65].

La razón de elegir este tipo de metodología para llevar a cabo esta primera instancia del proyecto se debe a que considerando el estado de inicio del comparador, es que se puede llevar a cabo una gran cantidad de investigación comenzando desde una fuente concreta. Esto puesto que desde un mismo articulo relacionado con la temática de comparación de modelos y meta modelos, se extrae una gran cantidad de artículos anexos en temáticas especificas del mismo, en otras palabras hacien-

do uso de los recursos que son expuestos dentro de de las referencias usadas en el documento.

Considerando lo anterior la metodología es aplicada durante la primera fase del proyecto, recolectando a través de varias fuentes y bibliografías de las mismas, la información necesaria para el desarrollo del comparador, sin embargo cabe mencionar que pese a que se termine la instancia de aplicación de esta, el producto final puede seguir siendo sometidos a cambios dependiendo de los nuevos desafíos que surjan.

La Tabla 3.1 representa las diferentes fuentes de información que fueron visitadas a través de la aplicación de la metodología de investigación. Dentro de esta se detalla el nivel a cual pertenecen así como la fuente original de donde fue sacado como referencia, en caso de no tenerla se denota con un carácter guión.

Cuadro 3.1: Tabla de resumen Snowballing

Código	Nombre	Autor(es)	Nivel	Origen
P01	A survey of approaches to automatic schema matching	ERhard Rahm, Philip A.Bernstein	1	-
P02	An Introduction to Model Versioning	Petra Brosch, Gerti Kappel, Philip Langer, Martina Seidl, Konrad Wieland, and Manuel Wimmer	1	-
P03	Difference and Union of Models	Marcus Alanen Ivan Porres	1	-
P04	Elección entre procesos automáticamente adaptados y Procesos Predefinidos	Felipe Ignacion González Martínez	1	-
P05	A Metamodel Independent Approach to Difference Representation	Antonio Cicchetti, Davide Di Ruscio, and Alfonso Pierantonio	2	P02
P06	Change Detection in Hierarchically Structured Information	Sudarshan S. Chawathe, Anand Rajaraman, Hector Garcia-Molina, and Jennifer Widom.	2	P03

Código	Nombre	Autor(es)	Nivel	Origen
P07	Fundamental theory for typed attributed graph transformation	Hartmut Ehrig, Ulrike Prange, and Gabriele Taentzer	2	P04
P08	a differentiation tool for domain-specific models	Yuehua Lin, Jeff Gray, and Frederic Jouault	2	P04

3.2. Metodología de desarrollo PXP

Para la segunda parte del proyecto, la cual corresponde al desarrollo de la página web que funciona como medio de interacción con el comparador de modelos, se utiliza la metodología de Programación Personal Extrema o PXP por sus siglas en ingles. Dicha metodología tiene como principales características el poder adaptarse a un ambiente de requisitos cambiantes, es decir a que los requisitos previamente establecidos por el usurario sufran algún cambio, se añadan nuevos requisitos o bien se opte por dejar fuera ciertas funcionalidades, junto con esto también tiene dentro de sus características el poner bastante énfasis en las pruebas que son hechas durante el transcurso del desarrollo en el cual se este aplicando la metodología [2].

La elección de esta metodología para llevar a cabo el desarrollo de la esta dada principalmente por la flexibilidad que tiene la misma para la adaptación de los requisitos,esto por que dado a la naturaleza del proyecto es necesario considerar que las especificaciones establecidas en un comienzo se pueden ver modificadas, esto debido a diferentes factores como o el tiempo o dificultades a medida que se va desarrollando la aplicación misma. Junto con lo anteriormente mencionado se añade el gran énfasis en las pruebas que se da, mas que nada por el hecho de que gran parte de los módulos de la aplicación web necesitan un proceso de pruebas bastante exhaustivo para cerciorarse de que lo mismos funcionaran de manera correcta una vez hayan sido implementados

Otro aspecto considerar de dentro de lo que conlleva el uso de PXP como metodología de desarrollo es el de saber cuales son los roles que se verán implicado durante el mismo. En este caso como estamos frente a un variante personal de XP, se considera como roles principales el de Cliente que en este caso es tomado directamente el profesor Guía y el de Programador que es cumplido por el alumno en

cuestión que este llevando a cabo el proyecto como memoria de titulo. En el ultimo caso se considera además de que el desarrollador deberá llevar a cabo el desarrollo de pruebas u similares durante las etapas posteriores a la implementación.

A continuación se muestra la Figura 3.1 correspondiente a la ilustración de todas las partes que son llevadas a cabo dentro de la metodología PXP, esta además hace explicito el como cada uno de las diferentes etapas interactúa, mostrando el camino temporal que se suele tomar para ponerla en practica. Posterior a esto se hace una explicación de cada una de las fases llevadas a cabo, no obstante estas son explicadas en base a como se aplicaron bajo el contexto del proyecto que se esta desarrollando

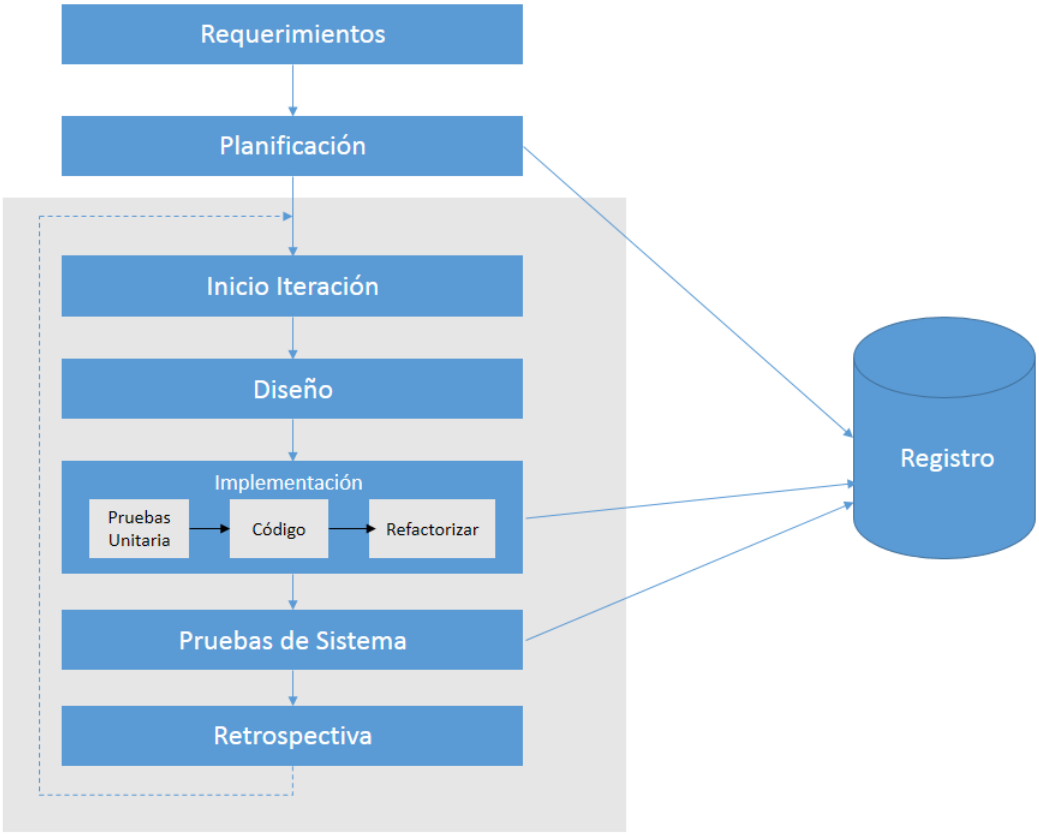


Figura 3.1: Descripción gráfica del modelo de desarrollo PXP

3.2.1. Requisitos

La fase de requisitos consiste en una captura de los mismos, dependiendo de cuales sean las especificaciones que sea necesarias llevar a cabo durante el completo

desarrollo de la aplicación. Para el contexto actual dichos requisitos están capturados como de usuario, esto debido a que los mismos son especificados de esta manera por el Cliente que fue anteriormente mencionado. Estos responden a este formato, diferenciándose a lo que es normalmente usado en PXP, correspondiendo a las historias de usuario que suelen ser especificadas para llevar a cabo durante desarrollos que usen esta clase de metodología o bien su variante de uso no personal. Se decide usar este formato, puesto que de esta manera es mas fácil la especificación de las peticiones del cliente, además este mismo sugiere usar este formato para facilitar la manipulación de las mismas. Dichos requisitos de usuario se utilizaran para generar las tareas, que a su vez generan subtareas, correspondientes a aspectos puntuales de cada una de las funcionalidades dentro de la aplicación, orientando a cada una dentro de su respectiva categoría, es decir, de interfaz o de servidor, también conocido como Front-End o Back-End.

La Tabla 3.2 muestra los requisitos que fueron capturados en esta fase.

Cuadro 3.2: Tabla de requisitos de Usuario

Código Requisito	Nombre	Descripción
RU001	Acceso de Usuarios	Permitir acceder a los usuarios a la plataforma
RU002	Creación de perfil	Permitir la creación de un perfil de usuario
RU003	Importar Modelos	Importar un modelo hacia la pagina en formato XML
RU004	Visualizar Modelos	Visualizar los modelos importados
RU005	Comparar Modelos Almacenados	Comparar modelos con los ya almacenados en la plataforma
RU006	Ver resultados de la comparación	Permitir ver resultados mas aproximados en la comparación de modelos
RU007	Comparar Modelos subidos	Permitir comparar modelos subidos por el usuario

Código Requisito	Nombre	Descripción
RU008	Registrar usuarios	Permitir que le administrador pueda registrar usuarios
RU009	Visualizar estado de usuarios	Permitir al administrador visualizar los usuario activos e inactivos
RU0010	Ver actividad de usuarios	Permitir al administrador ver un *dashboard* de los usos de la aplicación

3.2.2. Planificación

Durante la etapa de planificación se lleva a cabo la definición de todos los sprint, en este caso definiendo su duración formal y cuales serán los requisitos que cada abarcara. Esto se logra través al definir un macro para cada sprint, es decir agrupar una cierta cantidad de requisitos con el mismo enfoque en una Sprint particular.

La Tabla 3.3 muestra los 3 Sprint globales definidos, se considera hacer cada uno de estos con una duración de 4 semanas y como fue mencionado anteriormente, cada uno de estos engloba una o mas funcionalidades con similitudes entre si.Para especificar mas el contenido de estos se considera que el primer Sprint abarca las funcionalidades de gestión de cuenta y visualización de modelos, el segundo todo lo referente a la comparación de modelos y el ultimo solo le referente a visualización de datos, además de ciertas funcionalidades necesarias por el administrador

Cuadro 3.3: Tabla de división de Sprint

Sprint	Fechas	Requisitos trabajados
1	20/04/2020 - 20/05/2020	Se contempla todos los aspectos circundantes a el acceso y registro de usuario, junto a la subida de archivos y la administración de los mismos.

Cuadro 3.3: Tabla de división de Sprint

Sprint	Fechas	Requisitos trabajados
2	20/05/2020 - 20/06/2020	Se abarca todo lo respectivo a la comparación de modelos, considerando la acción misma, así como el ver resultados de las mismas.
3	20/06/2020 - 20/07/2020	Se consideran los aspectos circundantes al *dashboard* de administrador, incluyendo todo los aspectos de gestión y control de estadísticas.

Se considera que cada uno de estos sprint de puede ver modificado en cuanto a fechas o el contenido de los requisitos vistos dentro de cada uno.

Para poder llevar a cabo una mejor gestión del flujo de trabajo del proyecto se opta por utilizar un gestor virtual del método Kanban llamado Trello. Esta aplicación es un software en linea con uso en la nube, el cual permite visualizar a través de una interfaz simplificada donde pueden ser vistos con facilidad las tareas al clasificarlas y agruparlas a través de tarjetas dependiendo de las categorías en que se estén ocupando [35].

Se hace de uso de esta herramienta puesto que es bastante simple la interacción con la misma, además se ha ocupado en reiterados desarrollos pasados.El tablero se clasifica a través de varias tarjetas, cada una de estas posee una tarea general que se relaciona con el requisito respectivo y la iteración que se este desarrollando en ese momento como se muestra en la Figura 3.2 , además cada una de estas tareas generales contiene dentro un listado de sub tareas que deben ser llevadas a cabo durante el desarrollo. Dichas sub tareas corresponden muchas a los aspectos específicos de sistema que deben ser desarrollados ilustrado en la Figura 3.3

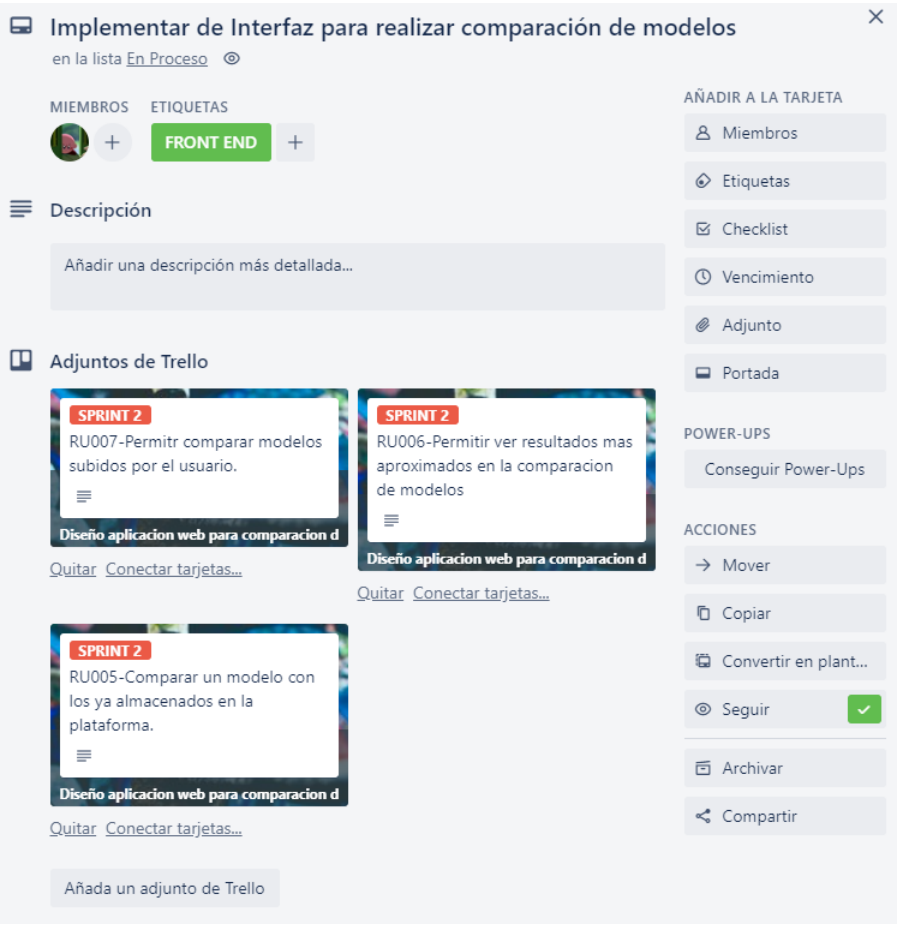


Figura 3.2: Descripción Tarjeta de Trello Requisitos

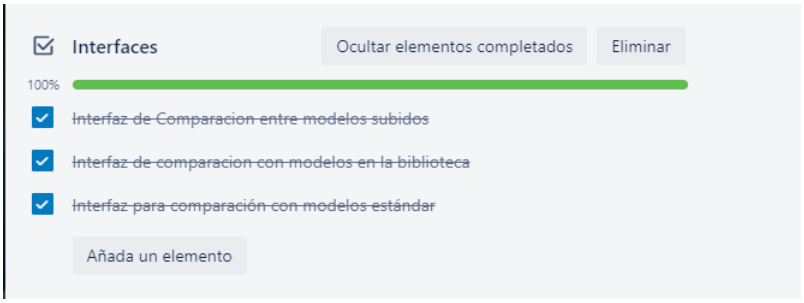


Figura 3.3: Descripción de Tarjeta de Trello lista de subtareas

3.2.3. Iteraciones

Durante las iteraciones se considera el uso de cada una de las fases especificadas en PXP, sin embargo estas sufren cambios con respecto al contexto que se le da al proyecto los cuales son detallados en cada una de estas.

Inicio

Para dar inicio a cada una de las iteraciones, se hace una definición previa de cada una de las tareas que serán usadas. Esto se logra haciendo un listado de sub tareas como los que fueron mostrados en la figura 3.3, dando de esta manera un orden al trabajo que sera llevada a cabo, además se establece una priorización de requisitos durante una reunión con el cliente.La periodización de requisitos corresponde a ver cuales de estos se vuelven cruciales dentro del desarrollo, así como cuales pueden ser modificados o simplemente pasados en alto en caso de problemas de tiempo.

Diseño

En la etapa de diseño se procede a revisar los diferentes diagrama que hayan sido generados como productos, además de crear los mismos si aun no están presentes durante el desarrollo, estos pueden pueden verse modificados dependiendo de como de lo que pidan las tareas especificas de la iteración, es decir si X tarea no se ve solventada con la actual arquitectura propuesta, por ejemplo el que no se vea representado correctamente dentro de algún aspecto de la base de datos, se resolverá modificando el actual estado del diagrama para que en la siguiente fase este sea modificado para satisfacer la característica.

Implementación

La etapa de implementación comprende la codificación que se lleva a cabo para terminar cada una de las tareas que sean dadas durante el desarrollo de la plataforma. Esto se lleva a cabo mediante un control de versiones, correspondiente en este caso a GitHub, donde cada cambio es subido a un repositorio para de esta manera integrar con el código que ya esta completado.

PXP indica que durante esta parte es necesario llevar a cabo pruebas unitarias para cada una de los módulos implementados, sin embargo para el contexto actual

del proyecto se decide usar pruebas de caja negra o simplemente depuración para cada una de las funcionalidades implementadas. Se toma esta decisión en parte por que es necesario que se lleven a cabo las pruebas, para cerciorarse del correcto funcionamiento de cada funcionalidad, sin embargo dado al limitado tiempo que se posee, se opto por una manera un poco mas informal para esto.

Pruebas

En la etapa de pruebas se lleva a cabo una prueba de sistema global de la funcionalidad implementadas.Esto es llevado a cabo con una reunión con el cliente, donde este para estar mas activo durante el desarrollo, indica un set pruebas que son ejecutadas en el mismo momento para asegurar que la funcionalidad esta cumpliendo con los lineamientos que el indico, además de notificar si es necesario reparar algún error dentro del sistema.

Retrospectiva

Finalmente en la retrospectiva se lleva a cabo la retroalimentación de todo el trabajo aplicado durante el sprint,considerando como puntos importantes el tiempo tomado en cada una de las tareas que se llevaron a cabo y teniendo en cuenta las dificultades que se vieron durante el desarrollo. Esta también se lleva a cabo en una reunión con el cliente.

3.3. Metodología de evaluación experimental en IS

Para la evaluación del proyecto, para de esta manera poder evaluar si se cumple con la calidad buscada del mismo, se opta por la metodología de evaluación experimental en Ingeniería de Software. Este método se divide normalmente se estandariza normalmente dentro de 4 fases [61]:

- Definición: Durante esta fase procede a dar una especificación de todos los elementos que serán llevados a cabo dentro del experimento así como también especificar hipótesis o similares. Entre otras cosas también se ven enfoques en cuanto al propósito, la calidad o similares. Dentro del contexto de la investigación en este fase se hará definición de cuales serán los objetos de estudio dentro de los modelos y las posibles comparaciones.

- Diseño o Planificación: Durante esta fase a grandes rasgo se lleva a cabo el diseño del experimento.Dentro de este diseño se especifican como se hará tratamiento de las muestras del experimento así como las herramientas que serán usadas durante el procedimiento. Llevándolo a la delimitación del proyecto durante esta fase definirá cuales serán los modelos que serán usados, las funcionalidades de la herramienta que serán usadas dentro del experimento y las posibles salidas esperadas.

- Ejecución: En esta fase simplemente se lleva a cabo la ejecución del experimento especificado, haciendo acción de la actividad necesaria para correr el experimento en su totalidad. En la herramienta sera la ejecución de la funcionalidad determinada dependiendo cual sea el ámbito seleccionado.

- Análisis: Finalmente en esta fase se lleva cabo una comparación de los resultados esperados, a través de métricas generadas durante la ejecución del experimento, para luego contrastar con los resultados esperados. Lo expuesto anteriormente se vera en el contexto de la investigación a través de la comparación de los resultados esperados de similitud que se obtengan con la herramienta y los modelos sometidos al proceso.

Para aplicar esto en el contexto del proyecto se realizaran experimentos controlados, usando esta misma secuencia lógica para llevarlos a cabo. La principal característica de estos es que presentaran diferentes modelos de proceso, los cuales pueden ser de origen real, es decir que pertenecen a alguna empresa establecida, o bien que estos sean solo simulaciones pequeñas creadas dentro del entorno de EPF Composer.

4. Estado del Arte

En este capítulo se describe el estado del arte sobre las técnicas algorítmicas para llevar a cabo la comparación de modelos, esto considerando conocimiento teórico y otros aspectos circundantes a la temática aplicado Snowballing.

4.1. Técnicas de comparación

En la siguiente sección se muestran las diferentes técnicas que componen el estado del arte actual de la comparación de de modelos. Se introducen además técnicas para la comparación de elementos en lenguajes específicos adicionalmente.

4.1.1. Detección de cambios dentro de información estructurada jerárquicamente

Esta estrategia algorítmica fue previamente mencionada dentro de los antecedentes de los modelos, mencionado como una de las estrategias que son utilizadas para el análisis de diferencias de modelos. A continuación se profundizara la técnica tratando de de abordar como esta es aplicada y como puede ser aplicada dentro de los modelos.

La detección consiste en detectar cambios en estructuras de información jerárquicas, refiriéndose este concepto a lenguajes como HTML o XML, los cuales pertenecen a los lenguajes de estándar establecidos por el World Wide Web Consortium(W3C) [66], en los cuales la información y estructura se distribuye en relaciones jerárquicas.

El algoritmo funciona en base a la comparación sobre los archivos en los cuales se trabaja, esto quiere decir que comparando la estructura en la que se ejecuta se pueden considerar como ha sufrido cambios. Dichos cambios se ilustran en base a los

documento de clase HTML, en los cuales al ser constantemente modificados sobre la marcha y la producción de paginas web se reflejaran variados cambios un vez que los documentos hayan sido sometidos sobre algún cambio que el usuario haya llevado a cabo frente a estos. Las estrategias que serán mencionadas son extraídas del trabajo de Sudarshan S. Chawathe [13].

Para la correcta ejecución de la estrategia algorítmica es necesario tener en consideración algunos aspectos particulares de la estructura que se encuentra presente en los archivos. Estos aspectos en algunos casos deben ser analizados de una manera mas minuciosa para poder llevar a cabo una comparación mas efectiva frente a los modelos. Algunos de los aspectos a tener en consideración son los siguientes:

- Información almacenada : Para una correcta comparación jerárquica es necesario considerar todos los componentes que puedan ser identificados dentro de los documentos por tanto es necesario explorar de alguna manera hasta los niveles mas profundos de la relación jerárquica, lo cual si es representado de alguna manera a través de grafos puede ser visto como una comparación directa a los nodos y sus hijos , dependiendo del grado de profundidad al cual se aspire llegar.

- No considerar los identificadores de objetos: En algunos caso ciertas porciones de datos como por ejemplo identificadores dentro de los mismos nodos deben ser ignorados, de esta manera para evitar incompatibilidad entre la comparaciones llevadas a cabo, puesto que dado a los cambios de versión puede que el estándar se modifique o ocurran situaciones similares.

- Comparación entre versiones nuevas y antiguas: Si bien ya ha sido mencionado varias veces, es necesario recalcar que se busca encontrar las diferencias esencialmente en versiones antiguas y nuevas, de esta se puede ver como ha sido el historial de cambios que los archivos han sufrido con el pasar del tiempo.

Cabe mencionar que todos estos aspectos son mencionados para determinar el algoritmo expuesto durante el trabajo que fue mencionado.

El autor representa a los archivos a comparar como arboles ordenados, en los cuales cada nodo tendrá una etiqueta y un valor, siendo esto lo necesario para establecer la comparación entre las estructuras a comparar. En la figura 4.1 se ve la representación que el autor le da, en donde los arboles representan la estructura jerárquica que

es vista en los documentos y las aristas rayadas representan la comparación llevada a cabo en cada uno de los nodos.

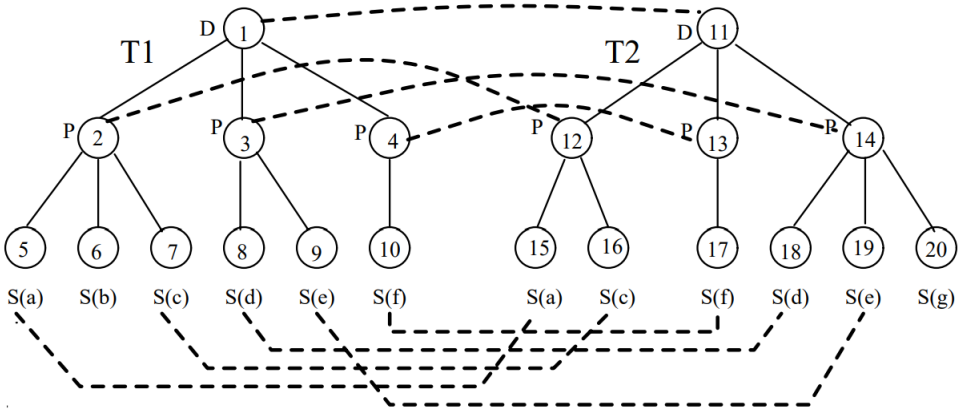


Figura 4.1: Representación del árbol delta

4.1.2. Static identity-based Matching

Esta técnica de comparación se basa en el uso de atributos libres de semántica dentro de los modelos, es decir, tipográficos o lingüísticos, de esta manera es posible establecer similitudes según el nombre o identificador de los elementos que componen los modelos [28]. Las formas en las cuales es puesto en funcionamiento esta técnica corresponden al *matching* tipográfico y el lingüístico.

El *matching* exige el uso de un identificador único par cada elemento dentro del modelo. La comparación se lleva a cabo a través de dichos identificadores, verificando que los identificadores de ambos modelos sean iguales o aplicando un algoritmo de la N-grama [46], a los identificadores, esto considerando establecer un umbral entre cero y uno, donde cero es similitud nula y uno igualdad.

Uno de los principales inconvenientes de esta estrategia es que si los modelos son creados en herramientas diferentes, probablemente poseerán identificadores diferentes, que no responderán al mismo tipo de secuencia, por lo tanto las comparaciones serán insatisfactorias.

El *matching* lingüístico en cambio considera las reglas y restricciones de la composición misma de los modelos,es decir que el nombre único de un elemento o que

una determinada relación tenga un origen y destino [28]. De esta manera es posible establecer el uso de operadores frente a las relaciones de palabras, para de esta manera llevar a cabo las acciones de comparación.

4.1.3. Signature-Based Matching

Esta técnica utiliza para la comparación un *signature*, lo cual se trata de una combinación establecida entre diferentes valores o características de un respectivo elemento del modelo, que en este caso son específicos del lenguaje que se usa para modelar [28]. De igual manera un *signature type* es considerado como un determinado conjunto de elementos usados para determinar así misma una *signature*.

Dentro de MDE para poder llevar a cabo esta clase de comparación, es necesario el uso de un *signature type* particular, el cual puede representar ciertos aspectos de los modelos.Es necesario considerar los modelos expuesto en la Figura 4.2, Estos poseen un nombre para sus respectivas clases, así como una definición de los atributos de cada una, además de los métodos y relaciones con otras estructuras presentes. Para este preciso caso el *signature type* debería venir dado por una clase, en la cual se especifican sus atributos, tipos de los mismos y las relaciones que esta tenga con otros.

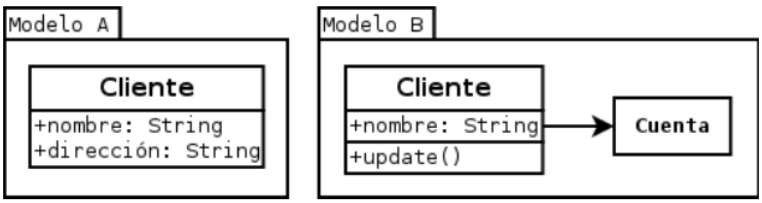


Figura 4.2: Modelo A y B para amplificación

Similar a otras técnicas vistas dentro del ámbito de comparación de modelos, es necesario determinar cuales serán los aspectos para determinar que los elementos son considerados dentro de un *matching*, es decir en el caso de la Figura 4.2 se puede considerar el nombre de la misma como un aspecto de comparación, lo cual daría un *match* entre ambas, pero de la misma manera se podría considerar otras como lo son las relaciones o la cantidad de atributos, lo cual arrojaría una diferenciación dentro de esta comparación.

4.1.4. Similarity-Based Matching

Para esta técnica de comparación se plantea las búsqueda de una *match* estructural entre los modelos, lo cual considerar ver a los mismo como estructuras denominadas como *typed atributed graphs*(TAG), que corresponde a una colección de nodos y arcos con atributos [22].

Cada elemento dentro del modelo posee su propio conjunto de atributos, es decir que tanto los nodos como las aristas pueden responder a tuplas con atributos diferentes. De esta manera para poder llegar a lograr *match* entre estos elementos diferentes presentes dentro de los modelos se usa el *signature based matching* el cual genera dos criterios de evaluación a considerar, esto dependiendo de la composición de los atributos de los mismos elementos [44].

- Node Signature Matching: Considerando dos modelos diferentes *M1* y *M2* con sus respectivos nodos v_{m1} y v_{m2}. Los nodos son equivalentes si el *signature* es equivalente.

- Edge Signature Matching: Considerando dos modelos diferentes *M1* y *M2* con sus aristas propias e_{m1} y e_{m2}. Las aristas son equivalentes si el *signature* es equivalente.

Esto se puede ver reflejado en la Figura 4.3, en la cual la Clase Cliente presente en el modelo A, tiene dos posibles candidatos para *matching* dentro del Modelo B. Se puede ver que dentro de estos nodos representados a través de la clases existen variados atributos, sin embargo uno que es bastante importante es el nombre de la misma clase. Siguiendo ese lineamiento ambas clases Cliente presentes en los modelos son candidatas a *match* usando el criterio entregado por *Node Signature Matching*, considerando que por el mismo las clases Cliente y Cuenta darían una visión negativa bajo el mismo criterio. Dentro del criterio *Edge Signature Matching* podría también considerarse un *match* ya que ninguno de los dos modelos tiene aristas presentes.

Complementando los criterios anteriormente definidos, se define un ultimo aspecto a considerar para determinar el *matching* al cual se le denomina *Edge Similarity* [44].

- Edge Similarity: Considerando que dos elementos en dos modelos *M1* y *M2*, siendo los nodos respectivos v_{m1} y v_{m2}, siendo ambos candidatos de *matching*. El *Edge Similarity* presente en ambos responde a que las aristas que que estén

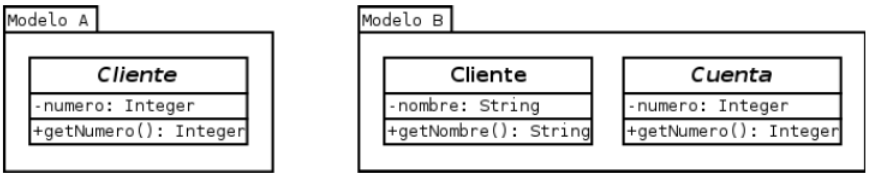


Figura 4.3: Modelo A y B aplicados en *Similiraty-Based Matching*

presentes v_{m1} cumplan *Edge Signature Matching* con las aristas presentes en v_{m2}.

El anterior concepto se puede ver aplicado directamente dentro de la Figura 4.4 , en esta se aprecia como ambos nodos de ambos modelos pueden responder con un *matching* directo, dando como resultado que ambos sean candidatos para el criterio *Node Signature Matching*, sin embargo al aplicar el concepto de *Edge Similarity* podemos ver que ambos no cumplen con el criterio de *Edge Signature Matching* por lo tanto no existirá un *match* entre ambos.

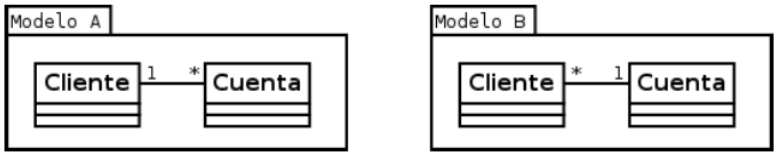


Figura 4.4: Modelo A y B aplicados en *Edge Similarity*

4.2. Técnicas de análisis de documento/texto

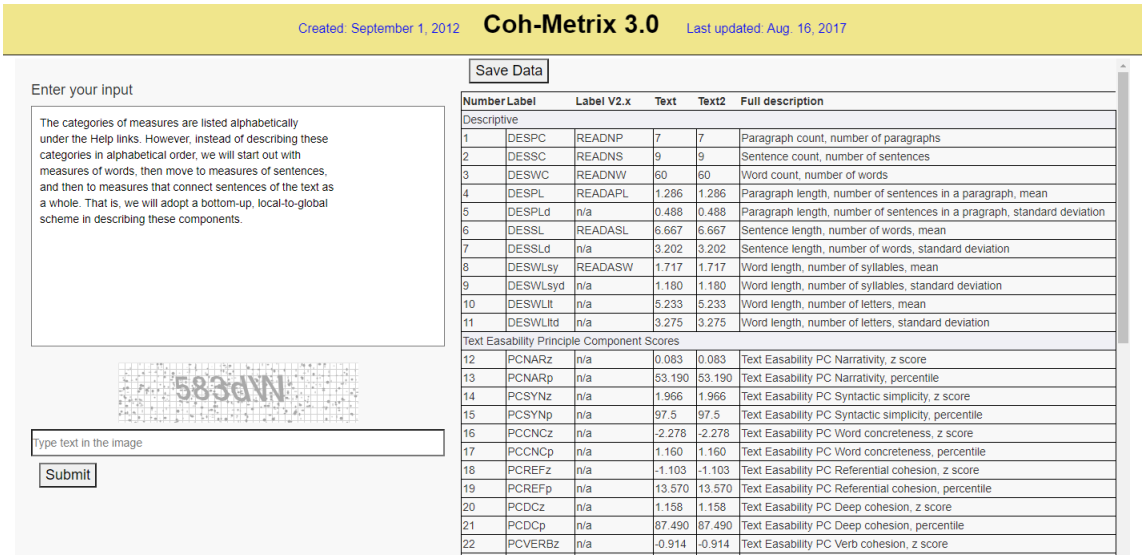
En el siguiente apartado se mencionan técnicas o estrategias utilizadas para la comparación de texto, esto considerando diferentes niveles de profundidad.

4.2.1. Sintaxis

En el ámbito de sintaxis se mencionan los siguientes métodos para llevar a cabo este tipo de tareas. Estos tienen como propósito la comparación a nivel de estructura de las frases, es decir corro bar una sintaxis correcta, dependiendo del lenguaje donde sean utilizados.

Coh-Metrix

Distinto de una técnica, Coh-Metrix hace referencia a una herramienta web [5]. Esta mas que para un análisis de sintaxis hace referencia a la búsqueda de cohesión dentro del texto que se este analizando. Para los autores de esta cohesión hace referencia a características explicitas del texto las cuales ayudan al lector a conectar las ideas del mismo texto [29]. Su uso es bastante simple, basta con usar la interfaz mostrada en la Figura 4.5. Dentro de esta basta con copiar el texto en el recuadro para obtener un resultado. Este contendrá un análisis profundo en varias métricas para definir la cohesión presente en el texto, las cuales pueden ser evaluadas por el lector para la correcta corrección de su escrito.



Number	Label	Label V2.x	Text	Text2	Full description
Descriptive					
1	DESPC	READNP	7	7	Paragraph count, number of paragraphs
2	DESSC	READNS	9	9	Sentence count, number of sentences
3	DESWC	READNW	60	60	Word count, number of words
4	DESPL	READAPL	1.286	1.286	Paragraph length, number of sentences in a paragraph, mean
5	DESPLd	n/a	0.488	0.488	Paragraph length, number of sentences in a pragraph, standard deviation
6	DESSL	READASL	6.667	6.667	Sentence length, number of words, mean
7	DESSLd	n/a	3.202	3.202	Sentence length, number of words, standard deviation
8	DESWLsy	READASW	1.717	1.717	Word length, number of syllables, mean
9	DESWLsyd	n/a	1.180	1.180	Word length, number of syllables, standard deviation
10	DESWLlt	n/a	5.233	5.233	Word length, number of letters, mean
11	DESWLltd	n/a	3.275	3.275	Word length, number of letters, standard deviation
Text Easability Principle Component Scores					
12	PCNARz	n/a	0.083	0.083	Text Easability PC Narrativity, z score
13	PCNARp	n/a	53.190	53.190	Text Easability PC Narrativity, percentile
14	PCSYNz	n/a	1.966	1.966	Text Easability PC Syntactic simplicity, z score
15	PCSYNp	n/a	97.5	97.5	Text Easability PC Syntactic simplicity, percentile
16	PCCNCz	n/a	-2.278	-2.278	Text Easability PC Word concreteness, z score
17	PCCNCp	n/a	1.160	1.160	Text Easability PC Word concreteness, percentile
18	PCREFz	n/a	-1.103	-1.103	Text Easability PC Referential cohesion, z score
19	PCREFp	n/a	13.570	13.570	Text Easability PC Referential cohesion, percentile
20	PCDCz	n/a	1.158	1.158	Text Easability PC Deep cohesion, z score
21	PCDCp	n/a	87.490	87.490	Text Easability PC Deep cohesion, percentile
22	PCVERBz	n/a	-0.914	-0.914	Text Easability PC Verb cohesion, z score

Figura 4.5: Interfaz de usuario de Coh-Metrix [5]

Dentro de su funcionamiento interno se encuentra el uso de la base de datos Psico-Linguistica MRC[14]. Dentro de esta se encuentras ciertas propiedades especificas para cada una de las 150.837 palabras contenidas dentro de esta. De dichas propiedades la herramienta ocupa algunas, las cuales son listadas a continuación:

- Familiaridad: Cuan frecuente aparece la palabra en el escrito.
- Concreción: Cuan concreta es una palabra, es decir cuan no abstracta es.
- Capacidad de imagen: Cuan fácil es llevar a cabo una imagen mental de la palabra en cuestión.

- Sentido de Paivio: Es una clasficacion de sentido establecido por las normas de Paivio [50] y Gilhooly[25]. Esta es multiplicada por 100 para dar un numero entre 100 o 700.

- Edad de adquisición: Basado también dentro de las normas de Gilhooly[25]. Este hace referencia a la edad en cual se obtiene una determinada palabra durante la niñez. De igual manera se multiplica por 100 para producir un numero entre 100 y 700.

Con estas determinadas métricas se calcula la coherencia dentro de las palabras contenidas en un texto, llevado desde el nivel frase, párrafo, hasta la composición completa del mismo.

4.2.2. Semántica

El ámbito semántico se puede encontrar que las metodologías para su análisis tienden a ser muy difíciles, debido a la dificultad que implica el uso de un modelo computacional para esto. Sin embargo existen ciertas técnicas para lograr este tipo de tareas.

Dentro de estas mismas podemos encontrar técnicas que se basan en el uso de similaridad semántica pertenecientes a las palabras presentes en las frases del texto [15]. A continuación se nombra algunas de estas:

- Similaridad de Lesk: En este caso se utiliza una función que superpone los significados de dos palabras, según sean provistas por un determinado diccionario [42].

- Similaridad de Wu y Palmer: Esta métrica hace uso de la taxonomía de WordNey, de esta manera se aplica usando parte de la profundidad del ultimo ancestro común de ambas palabras, combinando esto en la siguiente formula:

$$Sim_{wup} = \frac{2 \cdot depth(LCS)}{depth(concept_1) + depth(concept_2)} \tag{4.1}$$

  Donde *depth* representa la profundidad de las palabras, LCS el ultimo ancestro común de ambas palabras, y *concept* las palabras en si [67].

- Similarida de Resnik: La similaridad de Resnik postula la búsqueda de el retorno de la información contenida del ultimo ancestro común determinado de dos palabras [53]:

$$Sim_{res} = IC(LCS) \tag{4.2}$$

Donde IC representa la información contenida y se calcula como:

$$IC(c) = -\log P(c) \tag{4.3}$$

Donde $P(c)$ representa la probabilidad de encontrar el concepto (c) en un corpus extenso [63].

Este tipo de técnicas de comparación de palabras, prestan el pilar fundamental para lograr un estado superior y llegar a una determinada metodología de la comparación de textos.

En este aspecto se nombra el siguiente modelo de comparación semántica de texto, rescatado desde el trabajo Courtney Cole y Rada Mihalcea [15]. Este propone que al tener bien definidas la semántica de las palabra que corresponden al contexto del texto se puede realizar los siguiente. Dados determinados segmentos de texto T_i y T_j, se determinara el separar las palabras de estos en sets dependiendo de su clase, separados en sustantivos, verbos, adjetivos y adverbios. De esta manera la comparación semántica se lleva a cabo entre los sustantivos y los verbos, mientras que los adverbios y adjetivos solo son sometidos a una comparación léxica. Dichas comparaciones semánticas se llevan a cabo con alguna de las estrategias mencionadas en el punto anterior, para de esta manera fija un porcentaje que dará como exitosa a la misma.

4.3. Herramientas de comparación

En esta sección se muestran herramientas de software utilizadas para la comparación de modelos de proceso de software. Se consideran en este caso las pertenecientes a Eclipse.

4.3.1. EMF Compare

Esta herramienta perteneciente al *Framework* Eclipse, es utilizada directamente para la comparación de modelos. Dichos modelos deben haber sido previamente especificados dentro del software EPF Composer [21].

Las comparaciones llevadas a cabo a través de la aplicación son respondidas a través del motor de comparación propuesto por la misma Eclipse Foundation [59], el cual responde bajo la comparación textual de modelos. Dicho motor contempla dos usos generales para el uso en el aspecto de la comparación que se mencionan a continuación.

- Comparar dos modelos: El motor de comparación se encarga de computar las diferencias presentes en dos textos puntuales, los cuales son la especificación de los modelos.

- Comparar dos modelos con su ancestro común: Usar un modelo del cual hereden ambos en cuestión para poder ver los cambios a los cuales se han sometido sus derivaciones, en punto particulares como la mutación, la adición o la eliminación de elementos a los mismos.

Explicado el uso general del motor, es necesario explicar como es llevada a cabo la comparación dentro del programa, la cual se divide en dos fases: *matching* y *differencing* [28].

La fase de *matching* puede ser llevada a cabo a traves de*Similarity-Based Matching* usando como principales cuatro métricas pertenecientes al elemento como lo es: el nombre del elemento, su contenido,su tipo y las relaciones con otros elementos. También es posible llevar a cabo esta fase usando *Static-Based Matching*, sin embargo no se puede aplicar ambas técnicas juntas para llevar a cabo las comparaciones.

La fase de *differencing* es mostrada de manera gráfica usando un representación de ambos modelos, en base a una estructura de árbol. En la Figura 4.6, se puede apreciar como los modelos sometidos a comparación en este caso, imagen.xmi e imagen2.xmi resaltan la diferencia en el elemento *Work Product* Modelos de requisitos y Modelos de requerimientos, respectivamente señalando como este llevo una transformación en su estructura.

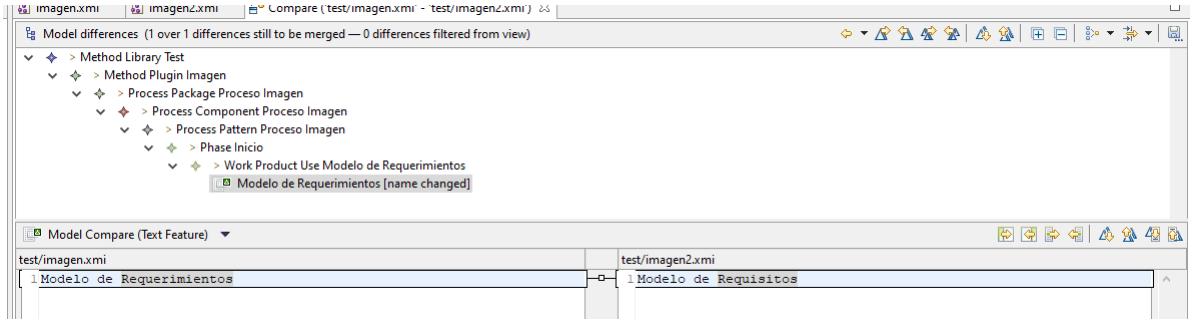


Figura 4.6: Ejemplificación funcionamiento EMF Compare

4.3.2. EMF Diff/merge

EMF Diff/Merge es una herramienta perteneciente al *framework* Eclipse, utilizada para la combinación o mezcla de modelos [18]. Mas precisamente se trata de un componente con la capacidad de comparar y mezclar colecciones de elementos presentes en los modelos, todo esto contando con una interfaz gráfica que se muestra en la Figura 4.7.

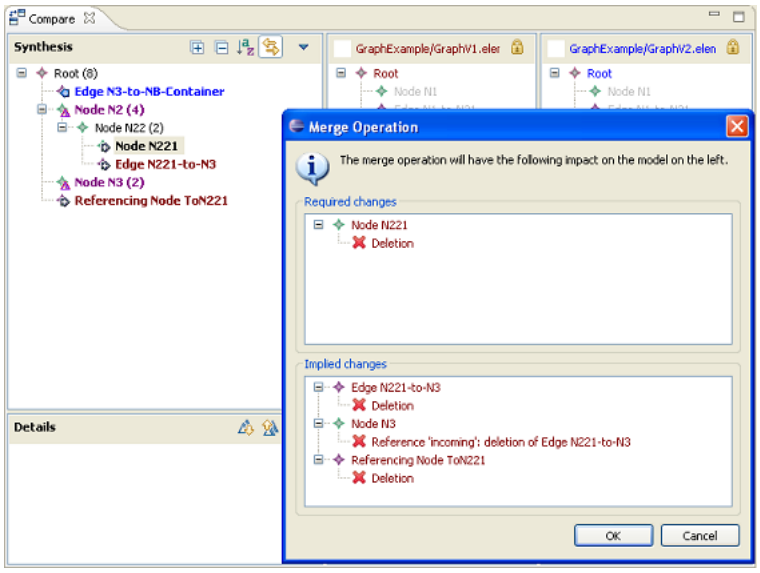


Figura 4.7: Interfaz Grafica EMF Diff/Merge [18].

Su proceso de funcionamiento se ilustra con la Figura 4.8. Primeramente se computan la diferencia que tengan los modelos a mezclar,una vez computadas la

diferencias se procede a especificar estas para finalmente entregar como salida un nuevo modelo, el cual responde a las especificaciones de los dos anteriores [19]

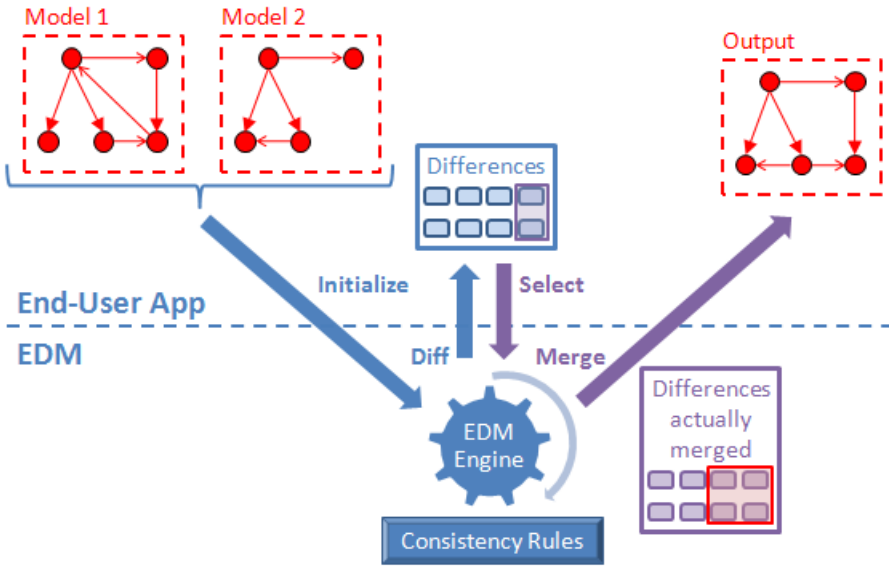


Figura 4.8: Ejemplificación proceso EMF Diff/Merge [19]

5. ProMoCoT

En este capítulo se explica todo lo referente al desarrollo del proyecto, considerando esencialmente los aspectos circundantes a la herramienta web del mismo. Para ver en mayor detalles diagramas, requisitos e interfaces revisar el Anexo **B**.

5.1. Concepción del proyecto

La idea del proyecto surge de las conversaciones con el Profesor Guía, Luis Silvestre Quiroga, comenzando en una primera instancia con la inquietud de este acerca de tener una opción alternativa para la comparación de modelos de proceso de software. El profesor manifiesta que las herramientas actuales o mas usadas, como es el caso de EMF Compare o similares,pese a tener una gran precisión, falla al usarse modelos con estructuras o diferencias léxicas en los nombres de las fases, por lo cual se busca una nueva clase de herramienta capaz de poder cubrir este tipo de necesidades, además de que conserve las virtudes que de por si tiene EMF Compare.

Durante las reuniones llevadas a cabo durante Agosto del 2019, se define cuales serian las principales características del proyecto, considerando la mas importante la capacidad de poder llevar a cabo la comparación de modelos incluyendo las particularidades anteriormente mencionadas. En septiembre 2019 se toma la decisión de que el proyecto se divida en dos partes siendo una de estas un comparador que sirve como la parte lógica de la plataforma y una aplicación web que funciona como la interfaz de usuario para hacer uso de este.

Comparador

Corresponde a toda la funcionalidad de lógica de comparación utilizada en la herramienta. El programa esta escrito en lenguaje Java y hace uso además de un

apoyo por parte de un servidor dedicado en lenguaje Python, siendo este ultimo el que se encarga de llevar a cabo la búsqueda de sinónimos haciendo uso de la herramienta WordNet. Las demás técnicas de comparación son todas realizadas de manera nativa dentro de la lógica implementada en Java y este es utilizado de manera directa por la plataforma web.

Plataforma Web

Encargada de la interacción del usuario con el comparador. La plataforma esta programada en el *framework* de PHP Laravel y se encarga de toda las acciones que el usuario necesite realizar con el comparador, además funciona para gestionar perfiles de usuarios, administrar la carga de modelos y mostrar de manera gráfica los resultados de cada comparación llevada a cabo en la misma.

5.2. Desarrollo del proyecto

A continuación se explica todo el proyecto considerando el trabajo que fue realizado durante el periodo de programación y pruebas de la plataforma web del mismo.

5.2.1. Lenguajes y herramientas para el desarrollo de software

A continuación se describen los lenguajes y herramientas de desarrollo de software que fueran utilizadas para llevar a cabo el proyecto.

Java

Java es un lenguaje de programación multi-propósito, basado en clases y basado en la orientación el paradigma de orientación a objetos [38]. Esta diseñado para ser lo suficientemente simple para que los programadores puedan acostumbrarse y lo dominen con rapidez.

Se elige este lenguaje para el desarrollo de la parte de comparación de la herramienta, puesto que dado a su naturaleza de orientación a objetos presenta una gran facilidad para el trabajo con los modelos, Debido a que estos pueden ser administrados como objetos, para de esta manera poder trabajar correctamente con ellos. Finalmente se considera el uso de este debido a la familiaridad que se tiene con el, puesto que gran parte de proyectos anteriores han sido escritos en este lenguaje.

Dentro del desarrollo se incorpora directamente en el comparador de modelos, es decir que la misma esta completamente programada en este lenguaje, considerando las fases de comparación tanto estructural como sintáctica. Otros aspectos de la herramienta que están escritos en este lenguaje son el extractor-inyector, el cual es utilizado para transformar los modelos de formato XML a XMI, eliminando todo el ruido incorporado por EPF Composer y el *parser* de XMI a JSON, el cual es utilizado para transformar los archivos XMI a un formato JSON especifico, usado en la visualización de modelos dentro de la herramienta.

PHP

PHP es un lenguaje de código abierto orientado especialmente para el desarrollo web y el cual puede ser incrustado dentro del contenido HTML [51]. A diferencia de JavaScript, PHP al ser incluido dentro de HTML responde directamente desde el servidor, funcionando de manera interiorizada.

Se decide utilizar este lenguaje, puesto que en el desarrollo web de la herramienta se hará uso del *framework* Laravel, posteriormente mencionado. Además se selecciona en parte por el gran soporte que tiene por parte de la comunidad y la facilidad para la ejecución de *scripts* dentro del funcionamiento del servidor.

Dentro del desarrollo se incorpora en el funcionamiento de la herramienta web, esto considerando que el *framework* Laravel funciona de manera nativa con PHP, además se incluye el uso de algunas funciones propias del lenguaje como es la ejecución de una terminal interna dentro del servidor.

Python

Python es un lenguaje interpretado, el cual posea la característica de que sintácticamente no es difícil de comprender [58]. Normalmente gracias a esta simpleza se hace bastante fácil el llevar a cabo tareas que en otros lenguajes tomarían mucho mas tiempo.

Se decide hacer uso de este lenguaje específicamente debido a la gran cantidad de librerías relacionadas con las ciencias de datos, en especifico con el procesamiento de lenguaje natural [10].

Dentro de la herramienta, este se usa directamente en servidor dedicado que sirve para la consulta de sinónimos, el cual hace uso de WordNet[52] para poder llevar a

cabo cada una de estas consultas.

Laravel

Laravel es un *framework* de desarrollo web escrito en PHP, este basa su diseño lógico en el Modelo Vista-Controlador (MVC), siendo una de sus principales características el tener un sintaxis de uso simple y la capacidad de llevar a cabo tareas haciendo uso de pocas lineas de código [47].

La decisión de usar este *framework* para llevar a cabo las tareas dentro del proyecto viene dada en parte por su capacidad de ofrecer una opción funcional utilizando MVC, además de tener un gran soporte dentro de la comunidad de programadores, lo cual muestra que no sera difícil buscar soporte o documentación acerca de este.

Dentro del proyecto Laravel es usado como el *framework* principal para la pagina web de comparación de modelos, considerando tanto el front-end como el back-end de la misma.

WordNet

WordNet es una gran base de datos léxica, considerando los idiomas ingles, español, francés, entre otros [52]. Dentro de su contenido se encuentran sustantivos, verbos, adjetivos y adverbios los cuales son agrupados en colecciones de sinónimos cognitivos(*synsets*) los cuales por si solos expresan distintos conceptos y se relacionan mostrado relaciones conceptuales-semánticas entre los mismos.

La decisión de usar WordNet dentro de los lineamientos se debe a la gran cantidad de conceptos que esta posee dentro de su base de datos, prácticamente todas las palabras el lenguaje natural ingles y español están contenidas dentro de ella, lo cual facilitara el uso de la comparación haciendo uso de las relaciones otorgadas por los *synsets* antes nombrados.

Dentro del proyecto WordNet se incorpora en el servidor dedicado de Python, el cual hace uso de este para llevar a cabo la comparación de sinónimos enviados desde el comparador. El uso de esta extensión es provisto por el uso de el componente de Python NLTK[1] el cual dentro de sus extensiones incorpora el uso de WordNet.

[1] https://www.nltk.org/

MySQL

MySQL es un sistema de gestión de bases de datos relacionales[2], el cual esta desarrollado por Oracle, así como licencia de código abierto.

La decisión de usar este gestor de Base de datos relacional, es por el gran soporte de la comunidad que recibe[3]. Es considerado dentro de los gestores mas usados mundialmente, además de haber sido utilizado por el desarrollador con anterioridad en variados proyectos de desarrollo web.

Dentro del contexto del proyecto se incorpora como el gestor de base de datos de la plataforma web. Dentro de este se albergan todas la tablas correspondiente a la herramienta, así como tablas que son propias del *framework* Laravel.

5.2.2. Requisitos

Los requisitos de usuario, son capturados a través de varias entrevistas con el cliente, y cada uno de estos especifica como las funcionalidades que le corresponden a la plataforma web, siendo completamente independientes del funcionamiento del comparador.

Una vez capturados los requisitos se da una especificación formal de cada uno. Dicha especificación considera varias características que se deben tener en cuenta de acerca de los mismos. En la Tabla 5.1 se ejemplifica cuales son los atributos que dan definición a cada requisito para ser aplicado en el desarrollo.

Cuadro 5.1: Tabla de ejemplo especificación de requisitos

Código	Nombre	Descripción	Estabilidad	Tipo	Usuario
RU001	Acceso de Usuarios	Permitir acceder a los usuarios	Alta	Funcional	Investigador

Las 3 primeras características hacen referencia a la especificación dada dentro del común de la captura de requisitos, añadiéndose además otras 3 donde se esclarecen ciertos aspectos puntuales de cada uno de los requisitos. Estos nuevos atributos a considerar definen aspectos que pueden afectar durante la implementación de las

[2]https://www.mysql.com
[3]https://www.oracle.com/mysql/

tareas correspondientes a cada requisito durante el desarrollo. A continuación se detallan mas a fondo cada uno de ellos.

- Estabilidad: Hace referencia a la posibilidad de cambio que tenga el requisito, es decir que tan susceptible es a sufrir una modificación durante el desarrollo. Puede tomar valores de Alta, Media o Baja.
- Tipo de Requisito: Hace referencia al tipo de requisito que se esta especificando, puede ser funcional, de restricción de calidad de software. Cada uno de estos valores tiene su propia incidencia dentro del software.
- Usuario: Hace referencia al tipo de usuario que tiene influencia directa con las funcionalidades que engloba el requisito, considerando como usuarios a estos en la categoría de Administrador e Investigador.

5.2.3. Planificación

En la etapa de planificación se procede a ordenar como serán llevados a cabo los Sprint correspondientes para el desarrollo. Se decide optar por una mecánica de 3 sprint separando cada uno de estos en funcionalidades especificas de la plataforma, además se opta por un tiempo de duración aproximado de 4 semanas por sprint.

Se dividen los requisitos en cantidades equitativas dentro de cada uno de los Sprint, considerando la especificación de cada uno de estos, para de esta manera terminar las funcionalidades mas necesarias primero. Cabe mencionar que esto ultimo se basa en las especificaciones del proyecto, por lo cual se da prioridad a todas las funcionalidades cruciales para la visualización de modelos así como su comparación, siendo esta ultima la mas importante.

La Tabla 5.2 muestra como fue la distribución de cada uno de los requisitos en los diferentes sprint planificados.

Cuadro 5.2: Tabla de división de Sprint

Sprint	Fecha	Requisitos trabajados
1	20/04/2020 - 20/05/2020	RU0001
		RU0002
		RU0003
		RU0004
2	20/05/2020 - 20/06/2020	RU0005
		RU0006
		RU0007
3	20/06/2020 - 20/07/2020	RU0008
		RU0009
		RU0010

5.2.4. Sprint 1

El Sprint 1 del proyecto corresponde a todas las funcionalidades referentes al registro y acceso de usuarios, centrándose específicamente en los usuarios del tipo investigador. Los investigadores son aquellos que pretenden utilizar la aplicación para llevar a cabo las comparaciones de procesos y modelos de proceso. También se consideran las acciones de importación de modelos de proceso a la plataforma, así como su visualización dentro de esta.

Inicio

Se lleva a cabo la priorización de requisitos a partir de la prioridad de cuales son los mas necesarios para el correcto funcionamiento de la plataforma. En este caso se llega al consenso con el cliente de que es mejor comenzar por los requisitos de acceso a la plataforma, por lo tanto se comienza por los de esta índole. En segundo lugar se considera el poder subir los procesos y modelos a la plataforma, teniendo en cuenta su correcto almacenamiento dentro de esta y finalmente se considera la visualización de los mismos. De esta manera se procede a generar las tareas que se llevaran a cabo durante el Sprint, considerando su orden según la prioridad que la mismas tengan.

Entre las tareas se consideran la creación de controladores para acceso y modelos. El primero para administrar el login, así como también la creación de usuario y el segundo para controlar la subidas de los modelos, incluyendo además su correcta visualización.

Diseño

Durante este periodo se elabora la primera versión del modelo relacional considerando las entidades necesarias para su funcionamiento, en este caso se le da prioridad al usuario, modelos y comparaciones para de esta manera almacenar todos los datos referentes a estos. La Figura 5.2 muestra las entidades creadas.

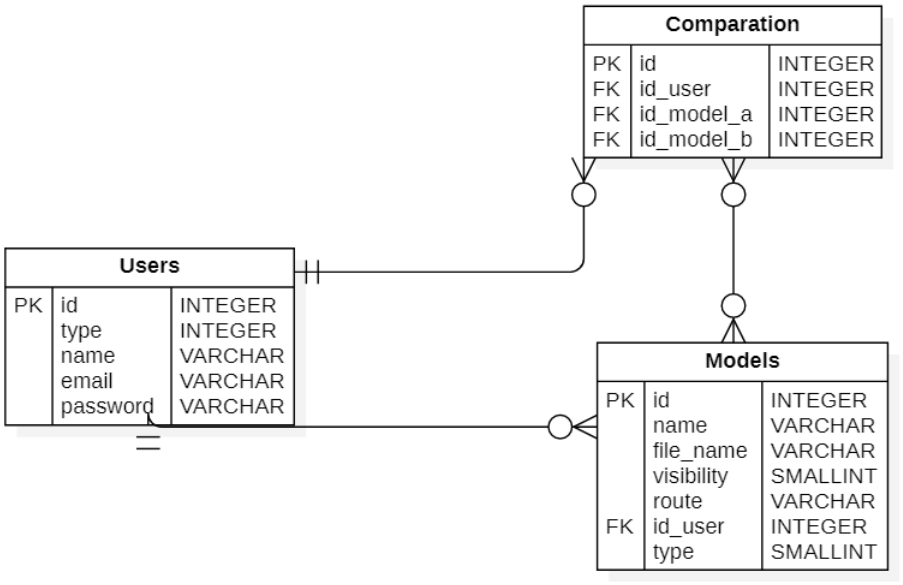


Figura 5.1: Partes principales Modelo Relacional

Se establece la arquitectura física como cliente servidor normal y se define la arquitectura lógica como el modelo vista-controlador o MVC [27]. Dicho modelo divide las partes de la plataforma en Modelo, las vistas y los controladores, permitiendo una implementación mas sencillas y modular para cada elemento, además es parte del funcionamiento básico de Laravel. Adicionalmente se agrega una capa extendida considerando el uso del comparador y además el extractor inyector para la transformación de pro esos en formato XML a modelos en formato XMI. La Figura 5.2 muestra esta capa externa dentro del diagrama de arquitectura Lógica.

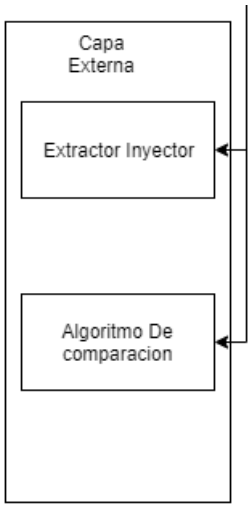


Figura 5.2: Capa externa arquitectura lógica

Implementación

En la fase de implementación se lleva a cabo la programación de los módulos necesarios, para satisfacer las necesidades de cada una de las tareas mencionadas anteriormente. Se generan los controladores de modelos y de acceso, además se la subida de modelo dentro de un nuevo controlador dedicado exclusivamente a eso.

En última instancia para la visualización de modelos se vuelve necesario la implementación de un nuevo elemento Java dentro del proyecto el cual funciona como un *parser* de los archivos XMI[4] de los modelos al formato Json[5]. Esto se crea puesto que la tecnología Jstree[6] utilizada un formato especifico de archivos para poder mostrar su contenido.

Las interfaces implementadas corresponden a dos interfaces iniciales, una para el login y otra para el registro de usuario. También se generan las interfaces para la visualización de modelos teniendo esta la peculiaridad de que se divide en 3 partes diferentes una para visualizar los modelos propios, otra para los de la comunidad y finalmente otra para los modelo estándar. Todas siguen el mismo patrón, sin embargo cambia su contenido. La Figura 5.3 muestra el modelo base de estas interfaces.

[4]https://www.omg.org/spec/XMI/About-XMI/
[5]https://www.json.org/json-en.html
[6]https://www.jstree.com/

Figura 5.3: Interfaz Lista de Modelos

Retrospectiva

Se discute acerca de todas las tareas llevadas a cabo las cuales fueron todas cumplidas, sin embargo se utilizo mayor tiempo durante estas. Este tiempo extra fue invertido principalmente en el hecho de implementar el *parser* para la visualización de modelos, lo cual al no estar considerado principalmente, considero una carga extra durante el desarrollo. Finalmente se discute acerca de las interfaces y si es necesario agregar algo nuevo dentro de estas.

5.2.5. Sprint 2

El segundo Sprint corresponde a las funcionalidades de comparación del proyecto, también pensadas los usuarios de tipo investigador. Se considera dentro de esta la visualización de resultados así como de igual manera la correcta ejecución de estos.

Inicio

Para comenzar se priorizan los requisitos que necesitaran mas atención dentro del Sprint. Se concluye que se comienza con la tarea de implementar la interfaz y controlador para la comparación, dejando en una segunda instancia el la vista de resultados, esto puesto que es necesario seguir este orden lógico.

Diseño

No se consideran nueva modificaciones dentro de los diagramas o arquitecturas, sin embargo el de arquitectura lógica se actualiza para considerar el nuevo elemento añadido que corresponde al *parser* como se muestra en la Figura 5.4.

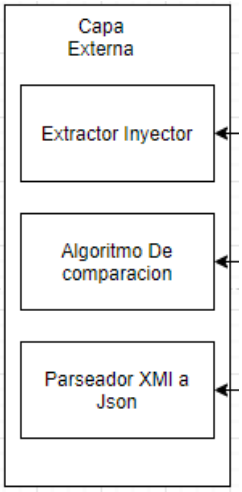


Figura 5.4: Capa externa Lógica *Parser*

Implementación

Durante la implementación se lleva a cabo el desarrollo de un nuevo controlador, el de comparación que se encargara de administrar todo lo referente al proceso de la misma, así como la generación de resultados dentro de la plataforma.Las interfaz se modela en cuanto las necesidades de comparación considerando dos tipos globales de comparación, entre procesos almacenados y entre procesos subidos en el momento de llevarla a cabo. Las interfaces implementadas para cada una se muestran en la Figura 5.5 y la Figura 5.6 respectivamente.

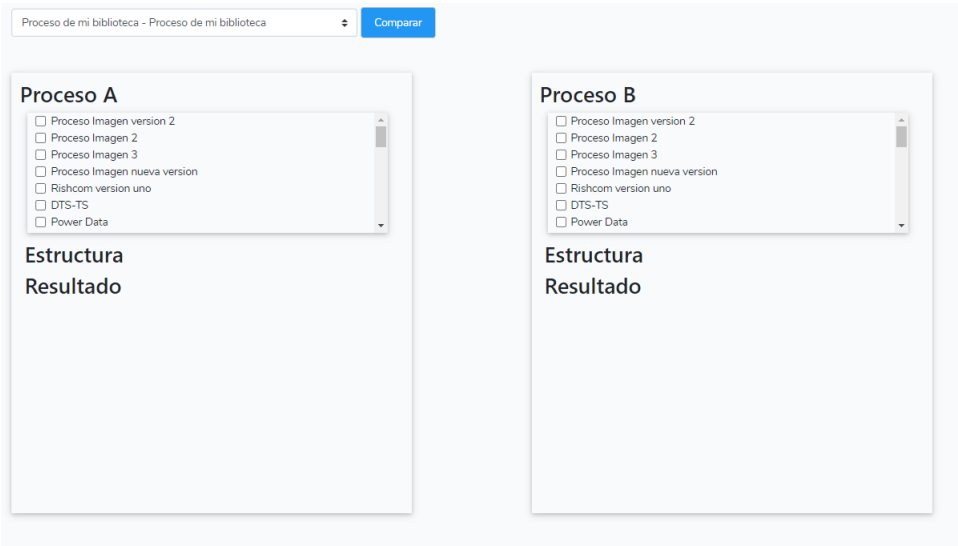


Figura 5.5: Interfaz de comparación Procesos almacenados

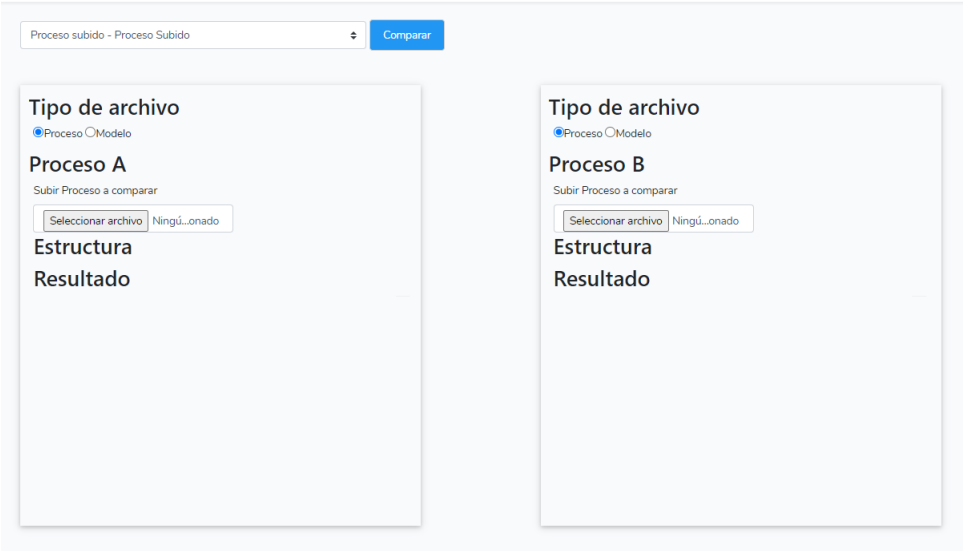


Figura 5.6: Interfaz de comparación Procesos subidos

Uno de los principales inconvenientes dentro del desarrollo es el tiempo que las comparaciones toman para llevarse a cabo dentro de la herramienta, por lo cual se decide recurrir a una nueva estrategia. Dentro del comparador se decide paralelizar la comparación de modelos haciendo uso de la tecnología de Laravel llamada Queues, estos permiten el uso de código en procesos externos llamados *Jobs* [41]. De esta

manera se logra paralelizar la comparación, no obstante los tiempos siguen siendo elevados dada la cantidad de elementos de algunos modelos, debido a esto se decide optimizar el comparador, puesto que este es el que mas tiempo toma, para lograr esto se elimina código redunda te del mismo y además se decide transformar el *Script* de Python en un servidor dedicado, lo cual disminuye considerablemente el tiempo invertido dentro de las comparaciones.

El hecho de llevar a cabo una paralelización implica una nueva modificación en la base de datos, esta para poder llevar a cabo la sincronización al momento de almacenar y buscar los resultados. Además se consideran nuevos módulos dentro de la arquitectura lógica debido a la inclusión de un servidor dedicado haciendo uso del *Script* de Python.

Retrospectiva

Dentro de la retrospectiva se consideran todos los inconvenientes que se tuvieron durante el desarrollo. El principal es el hecho de haber tenido que implementar un método de uso paralelo dentro de la misma aplicación lo cual sumo bastante tiempo al desarrollo, añadiendo casi una semana mas de desarrollo. Otro aspecto a considerar es que es necesario agregar dentro de la interfaz un apartado para administrar las notificaciones.

5.2.6. Sprint 3

El tercer Sprint corresponde a funcionalidades de interfaz, considerando globalmente la principal. También se da prioridad a las acciones de usuario, considerando la capacidad de ver las estadísticas de uso que tiene la pagina y el administrar a los usuarios de menor nivel, es decir los investigadores.

Inicio

Dentro del inicio se consideran la prioridad de tareas similar a las anteriores iteraciones llevadas a cabo, además se añaden nuevas tareas que no estaban consideradas en un comienzo. Una de estas la implementación de notificaciones para avisar al usuario de las comparaciones que ya fueron terminadas, mostrando en estas los resultados dentro de las misma interfaz que se usa en el apartado de comparación. Además se

considera el implementar una interfaz de actividad también para el usuario, puesto que en primera instancia había sido solo considerada para el administrador.

Se considera priorizar principalmente las tareas referentes al usuario para luego poder dar paso a las mismas del administrador.

Diseño

No se hacen grandes cambios en cuanto al diseño, sin embargo se agrega la nueva entidad correspondiente a los Jobs, de esta manera se completa la base de datos para llevar a cabo todas las acciones necesarias. Dicha entidad toma el nombre de *Works* y en esta se consideran las referencias y datos respectivos para el correcto uso y sincronización de los procesos externos una vez que estos son realizados. La Figura 5.7 muestra los atributos así como las referencias que posee, siendo estas la relación con la tabla *User* y la tabla*Comparation*

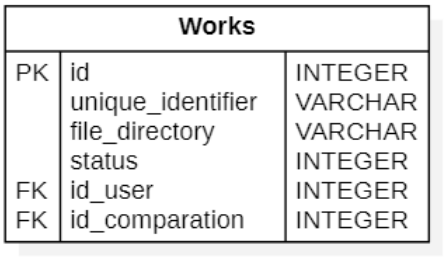


Figura 5.7: Entidad Works dentro del diagrama relacional

Implementación

Se implementa un nuevo controlador encargado de la interfaz principal dentro de las cuales se consideran las notificaciones y las estadísticas mostradas en la misma. Dicho controlador se implementa considerando el uso para los dos clases de usuario que están involucrados dentro de la plataforma, para de esta manera poder llevar a cabo todas las tareas que ambos buscan.

Se opta por usar una interfaz principal similar para el investigador y el administrador, considerando que las métricas de ambos son parecidas. Dentro de estas métricas para el lado del usuario se muestran números relativos a las acciones que este puede tener dentro de la herramienta, aludiendo a la cantidad de procesos,

modelos y comparaciones realizadas dentro de esta, en adición a estas métricas se incluye un cuadro de *feed* donde el investigador puede visualizar las ultimas 5 acciones referentes a los mismos tópicos. En la Figura 5.8 ese muestra parte de dicha interfaz

Figura 5.8: Interfaz Principal Investigador

En el caso del administrador como se menciona en el párrafo anterior, se implementa una interfaz principal similar, con la peculiaridad de que la misma incluye una métrica extra. Dicha métrica contempla la cantidad de usuarios registrados, en este caso de la clase investigador, además de mostrar el usuario registrado mas reciente. El *feed* que también se implementa en la interfaz de investigador se conserva, sin embargo de igual manera se modifica su contenido, para en este caso mostrar la actividad de los usuarios, considerando todo lo referente a las comparaciones llevadas a cabo por estos agrupándolos respecto en mas activos y poco activos. Dicha interfaz es mostrada en la Figura 5.9, considerando de esta manera los aspectos que

son modificados con respecto a la interfaz utilizada para el usuario investigador.

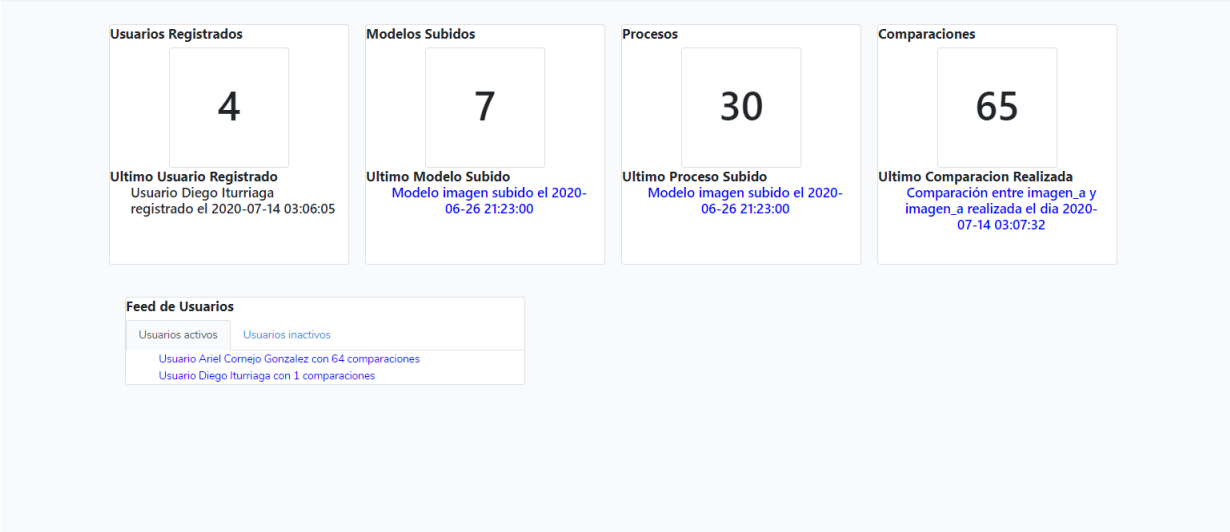


Figura 5.9: Interfaz Principal Administrador

Otro aspecto a considerar dentro de las funcionalidades otorgadas a los usuarios de clase administrador es el de visualización de modelos y procesos dentro de la misma plataforma, incluyendo la capacidad de subir los mismos a la plataforma en sus versiones estándar. Para llevar a cabo dichas acciones se modifica el controlador de modelos que anteriormente había sido implementado, cambiando ciertos parámetros en los datos que son entregados por el mismo. De esta manera el administrador recibe una interfaz donde es capaz de visualizar todos los modelos y procesos almacenado dentro de la plataforma, además de incluir ahora adicionalmente una interfaz para la subida de modelos estándar, siguiendo el diseño mostrado en la Figura 5.3.

Finalmente se procede a la implementación de registro de usuarios dentro de la interfaz del administrador, considerando que dichos usuarios quedaran se registraran bajo la clase de investigadores. Dado a limitaciones de tiempo esta funcionalidad no queda terminada, pero se considera para trabajo futuro dentro de la herramienta.

Retrospectiva

Como retrospectiva final se evalúan algunos aspectos del desarrollo así como posibles funcionalidades que pueden ser consideradas para trabajos futuros a implementar dentro de la herramienta. También se considera la modificación de ciertos

apartados de la interfaz, puesto que en esta instancia de desarrollo no se considero demasiado este aspecto, así como el poder dar uso de la funcionalidad no implementada, es decir, el registro de usuarios a través de la interfaz principal del administrador, no obstante dicha acción puede ser llevada a cabo directamente por los usuarios a través de la pagina de inicio de la herramienta.

5.3. Cierre del proyecto

Para dar finalización al proyecto se procede a poner el mismo en un entorno de producción, para de esta manera comenzar con las pruebas necesarias para la evaluación de la plataforma. Sin embargo debido a algunas incompatibilidades con el servidor con el que se esta trabajando y el poco tiempo con el que se cuenta para llevar a cabo dichas acciones se decide realizar la evaluación de manera local. Para el caso del cliente, la plataforma es instalada en su computador personal y se configura de manera que el pueda hacer uso de este para realizar todas las pruebas que el estime conveniente.

6. Evaluación de ProMoCot

En el siguiente capítulo se documenta como es llevada cabo la evaluación de la herramienta ProMoCot. Se incorpora una descripción de cada una de las etapas vistas en la metodología de evaluación aplicada, así como un resumen de los datos obtenidos a través de esta.

6.1. Fases de Experimentación

Las fases de experimentación descritas corresponden a la metodología de experimentación de Ingeniería de Software [61], exponiendo como estas fueron adaptadas para el contexto en el cual se desarrolla el proyecto.

6.1.1. Definición

Para dar definición a las pruebas es necesario considerar cuales son los aspectos a evaluar por parte de la plataforma, esto implica el como se abordaran las funcionalidades a través de los posibles ambientes de evaluación que se tengan. Dada esta consideración es que se opta por el uso de experimentos controlados [56] dentro de la herramienta. Estos consideran evaluar características esenciales de la herramienta, considerando los objetivos que han sido mencionados dentro de la especificación de este proyecto, además de características esenciales que necesitan también una correcta evaluación.

6.1.2. Diseño de la Experimentación

En el siguiente apartado se detalla como es llevado a cabo el diseño de la experimentación. Se especifican los objetivos de la evaluación, así como la características

que esta tiene dentro de su estructura.

Objetivos de evaluación

Se considera como principal objetivo de los experimentos controlados y la evaluación por parte del usuario, el llevar a cabo una evaluación de las características más determinantes del proyecto. Dichas características corresponden a las principales aspectos funcionales que son prestadas por la herramienta, en este caso principalmente para el lado de los usuarios de tipo investigador, así como aspectos circundantes la gestión procesos estándar considerando al usuario de tipo administrador.

Características de evaluación

Como se menciona en el párrafo anterior la evaluación corresponde al uso de experimentos controlados dentro de la aplicación. Estos corresponden a la realización de mini experimentos, donde el motivo principal es la validación de los resultados obtenidos con respecto a los esperados [56].

Otra característica determinante de la evaluación, es el comprobar aspectos determinados que se buscan en la herramienta. Dichos aspectos corresponden métricas de satisfacción con respecto a lo que se busca tanto con el desarrollo, así como con la experiencia de usuario. Los aspectos a evaluar son los siguientes:

- **Funcionalidad**: Considerada como una de las principales características según los lineamientos de la ISO 9126 [16]. Este hace referencia poder proveer las funciones necesarias para cumplir con los requisitos dados por el usuario, siendo usados en situaciones especificas [34]. Esto dentro del proyecto corresponde a comprobar si cada una de las funcionalidades cumplen con los requisitos entregados por el usuario,considerando principalmente la importación y comparación de modelos, lo cual se establece a través de la prioridad de requisitos que se realiza durante el comienzo del desarrollo de la plataforma.

- **Corrección**: Hace referencia a la aplicación efectiva de un algoritmo, es decir que este resuelva el problema par el que se diseño, produce las salidas deseadas y además tiene un tiempo de ejecución finito [32]. Dentro del proyecto se considera en cuanto a la funcionalidad de comparación, considerando como una

evaluación del comparador, es decir que va ligado también a la funcionalidad de la pagina.

- **Utilidad**: Se refiere a cuan útil puede ser el objeto que se evalúa, en este caso el software [33]. Dentro de los elementos de la herramienta es ver cuanto abarca dentro de lo que busca el usuario para trabajar.

Considerando los aspectos que se mencionan, es que se decide la estructura de los experimentos controlados. Para estos se utiliza un estilo similar a lo que se ve en los Test de caja Negra[48], en los cuales se muestra una entrada y además se especifica cual es la salida esperada. De esta manera los experimentos controlados consideran posibles entradas, siendo estas modelos para comparación o importación hacia la herramienta, esperando obtener resultados que se especifican con antelación. Además de estos, se consideran experimentos paralelos que son llevados a cabo por el cliente, este bajo el rol de usuario Investigador, para que el mismo valide las características que pretende usar de la plataforma.

El formato utilizado para los experimentos Controlados se ilustra en la Figura 6.1.2. En este se indica el código,tipo de usuario, aspecto evaluado, así como las salidas y entradas que pueda tener. También se agrega un apartado de observaciones en caso de ser necesario denotar alguna.

Cuadro 6.1: Ejemplo del formato de los Experimentos Controlados.

Codigo	EC001	
Aspecto Evaluado	Funcionalidad,Corrección	
Tipo Usuario	Investigador	
Descripción	Se realiza una importación a la pagina de un proceso en formato XML, exportado desde EPF Composer.	
Entrada	**Salida esperada**	**Salida obtenida**
Proceso DTS-TS en formato XML	▪ Registro en la BD del proceso subido. ▪ Archivo en formato XMI del mismo proceso. ▪ Proceso dentro la biblioteca del usuario.	▪ Proceso Registrado en BD. ▪ Proceso en formato XMI, nombre compuesto por original mas marca de tiempo. ▪ Proceso en biblioteca de usuario. Editable y visible para este.
Observaciones	El Proceso es subido en opción publica, por lo tanto es visible por otros usuarios. La privacidad es modificable a través de la interfaz.	

Instrumentos de Medición

El instrumento de medición, que en este caso se orienta hacia los experimentos que lleva a cabo el cliente, corresponde a una encuesta. Esta hace uso de la escala Likert [36], la cual corresponde a una evaluación por apreciación de cada tópico, clasificando esta una escala visual análoga que refleje que tan de acuerdo o en desacuerdo se encuentre con el ítem preguntado. Dichos ítem corresponden a la experiencia con el

uso de las funcionalidades de la plataforma y se relacionan de manera directa con los aspectos que se mencionan como fundamentales de la evaluación.

6.1.3. Ejecución de la Experimentación

En esta sección se da a conocer como es llevada a cabo la experimentación.Se consideran como se verán ejecutados los experimentos de manera global, además de también fijar el tiempo que se toma para realizar los mismos.

Protocolo de Experimentación

El protocolo de la experimentación se separa en dos secuencias diferentes, una correspondiente a los experimentos controlados llevados a cabo por el Autor y la otra correspondiente a los experimentos hechos en este caso por el cliente. Ambas se detallan a continuación:

- Protocolo Autor

    1. Definir los experimentos a realizar.
    2. Ejecutar el experimento determinado.
    3. Registrar el resultado del experimento.
    4. Comparar el resultado obtenido con el esperado.

- Protocolo Usuario o Cliente

    1. Registrarse dentro de la plataforma.
    2. Acceder a la plataforma como un usuario Investigador.
    3. Realizar la acción delimitada por el experimento.
    4. Registrar el resultado obtenido.
    5. Informar del resultado obtenido y cualquier inconveniente al desarrollador.
    6. Completar la encuesta de acuerdo a su experiencia con la herramienta.

Los protocolos que se mencionan son vistos de la manera mas genérica posible, por lo tanto responden para cada uno de los experimentos llevados a cabo.

Sesiones de Experimentación

Para llevar a cabo los experimentos, se realizan dos sesiones de experimentación. La primera es llevada a cabo por el Autor, en esta sea realiza un total de 9 experimentos controlados en los cuales principalmente se evaluaran los criterios de Funcionalidad y Corrección de la plataforma. La segunda le corresponde al cliente, como se dice en el apartado de cierre del proyecto la plataforma se configura en su computador para que el pueda hacer uso de sus funcionalidades. Dichas funcionalidades serán revisadas a través de mini experimentos que el mismo cliente decide ejecutar y que son evaluados a través de una encuesta mencionada en el instrumento de medición.

6.1.4. Análisis de la Experimentación

En el siguiente apartado se exponen los resultados de la sesiones de experimentación. Dichos resultados pertenecen tanto a los Experimentos Controlados llevados a cabo por el autor y los de la experimentación realizada por el cliente.

Resultados de los Experimentos Controlados

Para poder clarificar los resultados obtenidos por la experimentación controlados es necesario establecer una escala de éxito sobre los mismos. Esta escala responde bajos los 3 siguientes valores:

- Logrado: Significa que el experimento logró satisfacer completamente los objetivos esperados, sin ningún tipo de percance durante o después de su ejecución.
- Pendiente: Significa que el experimento logró satisfacer parte de los resultados, sin embargo algún aspecto especifico no pudo ser atendido de manera completa, por algún fallo o especificación dentro de los objetos usados para esto.
- No Logrado: Significa que el experimento no logró satisfacer de manera mínima o nula los los objetivos esperados, estos se puede deber a un fallo directo de la plataforma o la incompatibilidad del archivo en cuestión.

Considerando la escala que se menciona, se establece en la Tabla 6.2 un resumen con el éxito de todos los resultados correspondientes a cada uno de los experimentos

controlados que se realizan. Para mayor información acerca de los experimentos controlados revisar el Anexo **A**.

Cuadro 6.2: Tabla de resumen de experimentación

Código Experimento	Estado de éxito	Observaciones
EC01	Logrado	No hay Observaciones destacables.
EC02	Logrado	No hay Observaciones destacables.
EC03	Logrado	No hay observaciones destacables.
EC04	Logrado	No hay Observaciones destacables.
EC05	Logrado	No hay Observaciones destacables.
EC06	Pendiente	La salida muestra las diferencias encontradas, sin embargo la estructura del modelo presenta algunos errores, lo cual produce errores de visualización dentro de la herramienta. Ciertos elementos son especificadas en ingles, lo cual no permite ser realizada la tercera etapa de la comparación.
EC07	Logrado	No hay Observaciones destacables.

Código Experimento	Estado de éxito	Observaciones
EC08	Pendiente	Los modelos posee una diferencia de una frase, la cual es resuelta durante la segunda etapa de comparación, debido a su longitud. Sin embargo se debería considerar que dicha frase fuera resuelta hasta la tercera etapa del comparador.
EC09	Logrado	No hay Comentarios destacables.

Se puede apreciar en los resultados expuestos en la Tabla 6.2, que un su mayoría los experimentos controlados tiene un nivel de éxito máximo, siendo algunas excepciones las que quedan dentro de un nivel de pendiente, es decir que presentan algún tipo de inconveniente luego de su ejecución.

Dentro de los resultados completamente exitosos se puede apreciar que algunas características están completamente funcionales, según lo indicado por los experimentos Controlados llevados cabo.

La característica de importación,visualización y edición de modelos o procesos, es evaluada a través de los cuatro primeros experimentos. Se visualiza en los resultados particulares de estos experimentos, que cada uno de los tópicos recurrentes a la característica esta completamente funcional. Esto se debe en parte al formato que fue utilizado en gran parte de los archivos, además de que en dos casos particulares, estos fueron transformados de formato XML a XMI, a través del extractor-inyector que también es utilizado dentro de la plataforma. De esta manera todos los modelos son visibles de manera correcta dentro de lo que respecta a lo experimentos.

Por otra parte lo que corresponde a la característica de comparación de modelos y visualización de resultados es evaluada a través de los cinco experimentos controlados restantes. En este caso se puede apreciar que dos experimentos quedan en calidad de pendiente. Este caso se debe a ciertos aspectos puntuales dentro de los modelos sometidos a la acción de comparación. Para el caso de error de visualización, esto se debe a que la estructura de los modelos utilizados dentro del experimento **EC06** no

presentan el atributo id, el cual en parte no afecta gran parte del proceso de comparación en cuanto a números, sin embargo dificulta la visualización e identificación precisa de los elementos que fueron encontrados como diferencias. Probablemente esta peculiaridad se puede deber al método que se uso para la transformación de estos modelos o bien a como fueron originalmente definidos. El otro experimento en calidad de pendiente debe esto a un percance durante el proceso de comparación, puesto que la diferencia es encontrada como *match* un paso antes de lo estimado, lo cual produce una confusión al momento de visualizar los resultados de comparación, pero que de todos modos es correcta.

Finalmente considerando los aspectos evaluados dentro de los experimentos controlados, los cuales corresponden a funcionalidad y corrección, se puede decir que se cumple casi por completo con estas. Esto se ve reflejado directamente en el nivel de éxito de las pruebas, como se menciona en la Tabla 6.2, aunque de todas maneras como se explica en los párrafos anteriores, existen ciertos elementos en los cuales no se ve un resultado completamente satisfactorio. No obstante dichos resultados no presentan un error o una incidencia directa en la funcionalidad, si no que se remolcan mas a aspectos de la corrección, pero esto es debido a aspectos que van mas allá del control de la plataforma, por lo tanto se puede decir que pese a existir estos percances dichos aspectos cumplen con lo necesario.

Para el caso del aspecto de utilidad, esta no es directamente evaluada por los experimentos controlados, sin embargo se puede ver una relación con respecto a los aspectos funcionales, no obstante el resultado final se entrega a traves de la encuesta de usuario.

Resultados de los Experimentos de Usuario

A continuación se muestran los resultados de los experimentos llevados a cabo por el usuario. Estos no fueron directamente documentados, pero si son evaluados a través del instrumento de medición correspondiente a una encuesta. Dicha encuesta busca evaluar la funcionalidad de cada una de las características, así como también la corrección y la utilidad de la plataforma. Para ver la encuesta en detalle revisar el Anexo **C**.

La primera funcionalidad de la plataforma que es evaluada dentro de la encuesta es la de gestión de usuario. Los resultados que se pueden ver en la Figura 6.1,

muestran que de los 3 tópicos evaluados solo uno muestra un resultado inferior a **totalmente de acuerdo**. Dicho tópico hace referencia la información de la actividad de usuario. Esto puede ser debido a que aunque en la pantalla principal del usuario se muestra un registro histórico de las acciones que este lleva a cabo en la plataforma, esta no se esta mostrando de manera correcta, no por su contenido, si no mas bien ser refiere una temática de disposición de la interfaz, por lo tanto el usuario puede considerar que aunque sus datos son visibles no los puede ver con completa claridad.

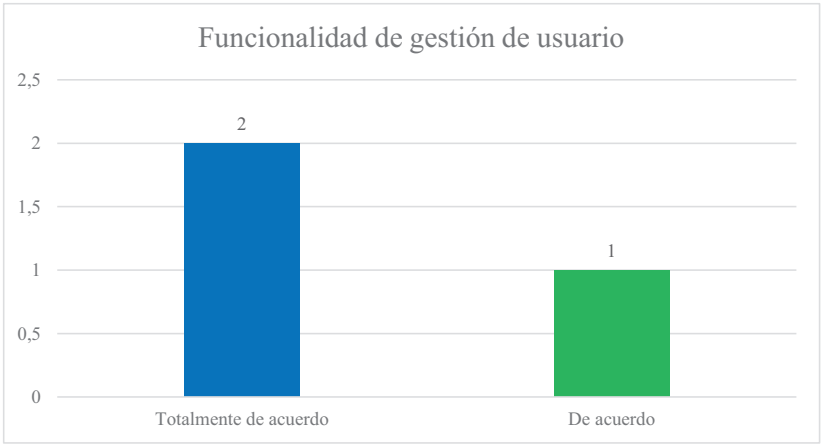


Figura 6.1: Gráfico respuestas Funcionalidad Gestión de Usuario

Los otros aspectos de registro e inicio de sesión son abarcados completamente, por lo tanto se puede decir que están funcionando de manera correcta.

La segunda funcionalidad de la plataforma evaluada dentro de la encuesta es la de gestión de modelos. Los resultados que se pueden apreciar en la Figura 6.2 muestran que solo uno de los tópicos evaluados presenta una respuesta inferior a **totalmente de acuerdo**. Dicha repuesta cae en el tópico de visualización de modelos subidos por otros usuarios, lo cual puede deberse a errores similares a los que se vieron durante los experimentos controlados, es decir el subir un modelo que no haya sido formateado de manera correcta. Esto puede deberse a errores en su estructura como la ausencia de algún atributo faltante o algo similar.

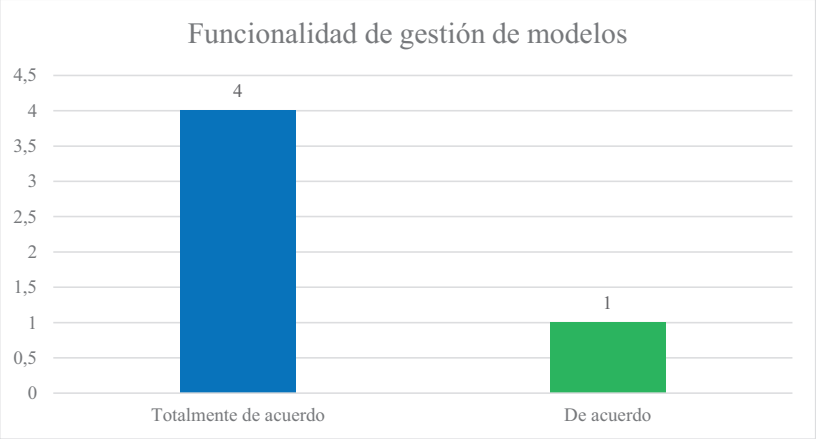


Figura 6.2: Gráfico respuestas Funcionalidad Gestión de Modelos

Para los demás tópicos se ve que que están funcionando de manera correcta según lo que expresa el cliente a través de la encuesta.

La tercera funcionalidad evaluada del sistema corresponde a la comparación de modelos. Como se aprecia en los resultados mostrados en la Figura 6.3 solo uno de los tópicos evaluados se clasifica como **de acuerdo**. Este corresponde a la comparación de modelos entre guardados dentro de la biblioteca, lo cual de igual manera puede ser debido a la estructura de los modelo en los cuales se realiza la comparación, lo cual puede recaer en que los mismos no posean una estructura correcta, lo que no permite ver de manera correcta los resultados. Otro aspecto que puede influir es el hecho de que en el recuadro de modelos o procesos públicos a comparar no se indica directamente que estos son públicos, además se muestran los modelos o procesos del mismo usuario, lo que puede de igual manera provocar confusión.

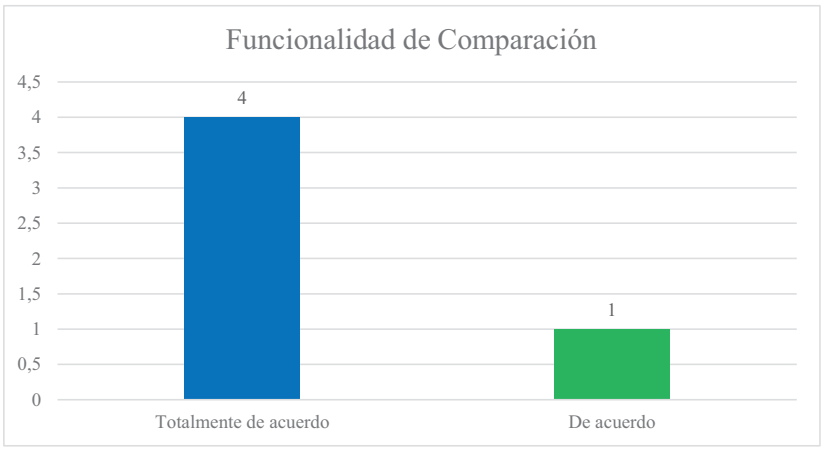


Figura 6.3: Gráfico respuestas Funcionalidad Gestión de Comparación

Luego de que se revisaran las funcionalidades se procede a analizar la corrección del sistema, considerada desde el punto de vista del cliente y evaluado de igual manera por la encuesta. Como se aprecia en la Figura 6.4 ambas preguntas tienen respuestas **de acuerdo**, siendo referente a la comparación de modelos y la visualización de los mismos. De manera similar a las funcionalidades que presentaban este tipo de déficit, se puede deber a aspectos con la estructura de los modelos, que tanto como en comparación como visualización puede provocar que se presenten errores dentro la estructura de visualización de la plataforma. Otro aspecto que se puede recalcar e influir en esto es la disposición de las interfaces de visualización de modelos y resultados, puesto que ambas se muestran en un *modal* que puede considerarse como poco espacio por parte del usuario.

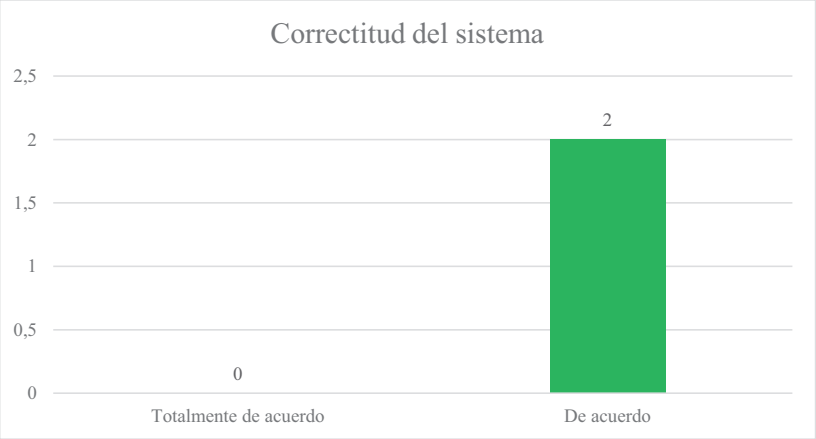


Figura 6.4: Gráfico respuestas Corrección del Sistema

La utilidad del sistema dentro de la encuesta se ve reflejada a través de la Figura 6.5. En esta se denota que una solo pregunta obtuvo la clasificación **de acuerdo** mientras que las otras dos tienen clasificación máxima. El tópico que no presenta clasificación máxima hace referencia a las notificaciones de comparación, estas tienen un pequeño problema debido a que no son mostradas de manera dinámica, lo cual puede confundir al usuario al momento de esperar por una comparación que tome un tiempo superior al estimado.

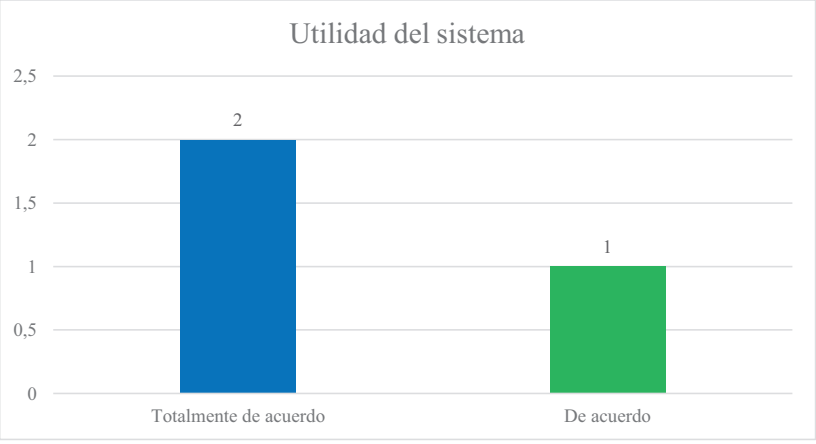


Figura 6.5: Gráfico respuestas Utilidad del Sistema

Por otra parte los otros dos aspectos que si obtuvieron la nota máxima hacen referencia a mostrar los datos históricos , así como las ultimas actividades realizadas, los cuales son correctamente mostrados en la interfaz principal.

Finalmente se pide una evaluación de apreciación general de la plataforma, bajo los 3 aspectos que se tienen como eje: Funcionalidad,Corrección y Utilidad. Los resultados de este apartado de la encuesta se muestran dentro la Figura 6.6, en este se puede apreciar que solo el apartado de Corrección no posee una calificación máxima, lo cual es subsecuente a las razones dadas para tópicos anteriores.

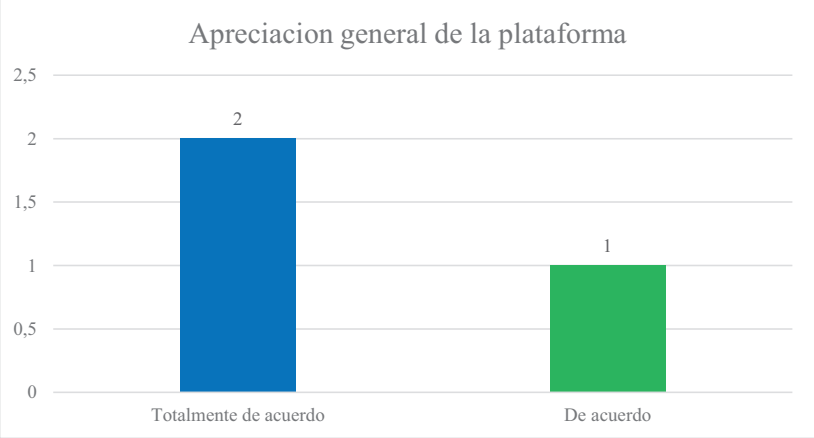


Figura 6.6: Gráfico respuestas Apreciación General del la Plataforma

6.2. Revisión de objetivos de experimentación

Considerando la naturaleza de los principales objetivos planteados en esta evaluación, correspondientes a la evaluación de las características de la plataforma por parte de los Experimentos Controlados y la evaluación propuesta por el cliente, se puede decir lo siguiente.

Para la evaluación correspondiente a los Experimentos Controlados, se evidencia a través de los resultados de estos que en su mayor parte las características de la plataforma funcionan de correcta manera, considerando los aspectos evaluados, en este caso Funcionalidad y Corrección. Se consideran que solo en algunos casos, como los que se indican en su respectivo análisis, no se tuvo un resultado del todo satisfactorio. Sin embargo, pese a estos aspecto puntuales, en su mayoría se cumplen los objetivos de dar una evaluación a las características, considerando obtener resultados positivos.

La evaluación correspondiente a la experimentación llevada a cabo por el usuario, también evidencia resultados positivos, mostrando de similar manera el cumplimento

del logro desde este aspecto. Dichos resultados se reflejan en su respectivo análisis, donde se puede apreciar directamente en los resultados de la encuesta que gran parte de las funcionalidades se están ejecutando de manera correcta.

7. Conclusiones y trabajo futuro

En el presente capítulo se exponen las conclusiones a las cuales se llega con la finalización de este proyecto de memoria. Además se exponen las lecciones aprendidas por el autor durante el proceso y el trabajo futuro considerado para el proyecto.

7.1. Discusión de objetivos del proyecto

En esta sección se discute el cumplimento de los objetivos del proyecto. A continuación se describen los objetivos específicos y cómo estos son abarcados en el desarrollo del proyecto.

- **Aplicar un método para la estandarización de modelos de proceso de software, haciendo uso de un extractor e inyector:**

  Este objetivo hace referencia a la funcionalidad de que la plataforma sea capaz de realizar la transformación, de un proceso salido de EPF Composer en formato XML hacia un modelo en formato XMI, almacenado dentro de la plataforma. Esto se ve solventado a través de lo expuesto en el **Capítulo 5** específicamente. Dentro de este capitulo que expone todo el desarrollo de la plataforma y en el primer sprint se especifica como se lleva a cabo la gestión de modelos. Dentro de esa especificación es donde se expone como el extractor-inyector es incorporado en la lógica de la plataforma, comprobando de esa manera que se cumple este objetivo en especifico.

- **Caracterizar los principales conceptos del proceso de software e ingeniería dirigida por Modelos.**

Este objetivo hace referencia a la presentación de los temas necesarios para la correcta comprensión de los conceptos mencionados en el mismo, bajo el contexto expuesto en el proyecto. Esto se lleva a cabo a través del **Capítulo 2** del documento, donde se presentan los antecedentes que corresponden para ambos conceptos, dando a conocer las temáticas que deben tenerse presente durante la lectura de documento para poder comprender correctamente el proyecto.

- **Definir un método para la comparación de modelos:**

  Esto se refleja dentro del **Capítulo 3**, donde se explica el estado del arte y se documenta los métodos de comparación de modelos. De esta manera se comprueba como parte de la documentación con respecto a estas temáticas incide dentro del proceso del método de comparación.Además en el **Capítulo 1** se detalla como es la estructura que se sigue para la comparación y esta se va documentando con lo anteriormente mencionado.

- **Desarrollar una herramienta de software para llevar a cabo la gestión de modelos**

  El objetivo hace referencia al desarrollo completo de una herramienta de software la cual pueda gestionar modelos, considerando la importación, visualización y comparación de los mismos. Esto se ve reflejado a través de todo el **Capítulo 5**, donde se especifica todas las funcionalidades implementadas para lograr la gestión completa de modelos.

- **Evaluar el desempeño de la herramienta al momento de realizar comparaciones**

  El objetivo hace referencia a la evaluación que es necesaria aplicar dentro de la plataforma para medir de esta manera el desempeño con el cual se están realizando las comparaciones.Esto se ve reflejado en el **Capítulo 6** de la memoria, donde se expone la evaluación que fue realizada a través de la metodología de experimentación en ingeniería de software. De esta manera se evalúa no tan solo los aspectos de comparación de la plataforma, puesto que además se incluyen características de importación de modelos, visualización de los mismos, entre otras. Esto se valida a través de los resultados de los experimentos controlados realizados por el usuario, así como la experimentación que realiza el usuario, donde sus resultados son medidos a través de una encuesta.

7.2. Lecciones aprendidas

En la siguiente sección se expone las lecciones aprendidas por el autor con el desarrollo de este proyecto. En estas se muestran los conocimientos que este adquiere a través de la experiencia que conlleva el desarrollo de este proyecto. Estos se dividen en 3 aspectos fundamentales: Metodológico, Técnico y Personal

- **Metodológico:**

  Dentro del aspecto metodológico se menciona como una de las principales lecciones la modificación y adaptación de las metodologías. El autor considera que esta es la mas importante, puesto que es una de las mas complicadas de llevar cabo dentro de la practica. Se considera esta principalmente al aplicar la metodología de desarrollo, ya que esta se adapta y recorta para quedar en sincronía de como se pretende llevar a cabo la implementación de la plataforma. De esta manera el autor rescata que pese a tener una forma definida, las metodologías pueden ser sometidas a una adaptación dependiendo de su contexto de aplicación.

- **Técnico:**

  En el aspecto técnico es donde mas lecciones y a la vez conocimientos se rescatan. Principalmente estos van desde el ámbito de desarrollo web, donde el Autor aprende a usar el *framework* Laravel, así como también en la implementación del comparador de modelos. En este ultimo se aprende no solo diferentes técnicas de comparación, si no que además se adentra en el paradigma de MDE, el cual previo al desarrollo de este proyecto era ajeno al Autor. Además se obtienen conocimientos en el ámbito de uso de herramientas de software de este mismo paradigma. Finalmente se obtiene conocimiento con respecto a la utilización de Python, específicamente para el manejo de lenguaje natural, haciendo uso de WordNet.

- **Personal:**

  En el ámbito personal se destaca principalmente la experiencia ganada en cuanto a constancia y responsabilidad. Ambos factores fueron cruciales dentro del desarrollo del proyecto, puesto que aplicando ambos a los lineamientos del proyecto, para lograr el objetivo de terminar el desarrollo del así como la escritura

del documento. El autor considera que estos son incluso mas preciados que el mismo conocimiento técnico, puesto que estos tendrán una incidencia directa dentro su vida laboral.

7.3. Trabajo Futuro

En esta sección se exponen los aspectos que se consideran como trabajo futuro dentro del proyecto. Estos corresponden directamente a la plataforma y el comparador.

- Implementar una opción multi-idioma para la comparación. Considerar lectura de elementos con nombres en ingles dentro de la tercera etapa de comparación.
- Implementar una función para la recomendación dentro de las comparaciones. Realizar la adición de Machine Learning o similares para poder dar retroalimentación al usuario en cuanto a los posibles elemento que podrían ser añadidos dentro de su modelo de proceso de software.
- Dar la capacidad de no solo aceptar exportados desde EPF Composer, por ejemplo BPMM[1] o BPEL[2], para de esta no limitar solo al uso de las herramientas de Eclipse.
- Mejorar la interfaz de la aplicación, dando énfasis en la estabilidad y la experiencia de usuario con esta. Considerar el uso de una capa de Front-End diferente y externa a Laravel como ReactJs[3], Vue[4] o Angular[5].
- Implementar la capacidad para que el administrador puede gestionar el registro de usuarios.
- Implementar opciones para que el usuario pueda modificar datos de su perfil.

[1]https://www.softexpert.com/es/solucao/bpmm/
[2]https://www.oracle.com/technical-resources/articles/matjaz-bpel.html
[3]https://es.reactjs.org/
[4]https://vuejs.org/
[5]https://angular.io/

Bibliografía

[1] Silvia Teresita Acuña and Xavier Ferre. Software Process Modelling. In *ISAS-SCI (1)*, pages 237–242, 2001.

[2] Ravikant Agarwa and David Umphress. Extreme programming for a single person team. *Proceedings of the 46th Annual Southeast Regional Conference on XX, ACM-SE 46*, (August):82–87, 2008.

[3] Marcus Alanen and Ivan Porres. Difference and union of models. In *International Conference on the Unified Modeling Language*, pages 2–17. Springer, 2003.

[4] Mn Álvarez Carmona. Detección de similitud semántica en textos cortos. *Instituto Nacional de Astrofísica, Óptica y Electrónica. Tonantzintla, Puebla*, 2014.

[5] Andrew Tackett,Dr. Danielle McNamara. Coh-Metrix. http://cohmetrix.com/, 2017. Accedido Julio 2020.

[6] Ove Armbrust, Masafumi Katahira, Yuko Miyamoto, Jürgen Münch, Haruka Nakao, and Alexis Ocampo. Scoping software process lines. *Software Process Improvement and Practice*, 14(3):181–197, 2009.

[7] Youssef Bassil. A simulation model for the waterfall software development life cycle. *arXiv preprint arXiv:1205.6904*, 2012.

[8] Aldo Sebastián Bertero González. Incorporación de un parser xml-xmi para modelamiento de procesos computacionales. 2012.

[9] Jean Bézivin. Generative and Transformational Techniques in Software Engineering. 4143(June), 2006.

[10] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit*. .ºReilly Media, Inc.", 2009.

[11] Barry W Boehm. A spiral model of software development and enhancement. *Computer*, (5):61–72, 1988.

[12] Tim Bray, Jean Paoli, C Michael Sperberg-McQueen, Eve Maler, Franois Yergeau, et al. Extensible markup language (xml) 1.0, 2000.

[13] Sudarshan S Chawathe, Anand Rajaraman, Hector Garcia-Molina, and Jennifer Widom. Change detection in hierarchically structured information. *Acm Sigmod Record*, 25(2):493–504, 1996.

[14] Max Coltheart. The mrc psycholinguistic database. *The Quarterly Journal of Experimental Psychology Section A*, 33(4):497–505, 1981.

[15] Courtney D Corley and Rada Mihalcea. Measuring the semantic similarity of texts. In *Proceedings of the ACL workshop on empirical modeling of semantic equivalence and entailment*, pages 13–18, 2005.

[16] Danang Ary Dewangga, Apol Pribadi Subriadi, Feby Artwodini Muqtadiroh, et al. Quality measurement of software based on characteristics of functionality, reliability, and maintainability. In *2018 International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)*, pages 192–197. IEEE, 2018.

[17] Víctor Hugo Menéndez Domínguez, María Enriqueta Castellanos Bolaños, et al. Spem: Software process engineering metamodel. 2018.

[18] Eclipse foundation . EMF DiffMerge. https://wiki.eclipse.org/EMF_DiffMerge, 2020. Accedido Julio 2020.

[19] Eclipse Foundation . EMF DiffMerge Proposal. https://www.eclipse.org/proposals/modeling.emf.edm/, 2020. Accedido Julio 2020.

[20] Eclipse Foundation. Eclipse Process Framework Project (EPF). https://www.eclipse.org/epf/, 2018. Accedido Septiembre 2019.

[21] Eclipse Foundation. EMF Compare. https://www.eclipse.org/emf/compare/, 2019. Accedido Septiembre 2019.

[22] Hartmut Ehrig, Ulrike Prange, and Gabriele Taentzer. Fundamental theory for typed attributed graph transformation. In *International conference on graph transformation*, pages 161–177. Springer, 2004.

[23] Eclipse Foundation. Eclipse Process Framework (EPF) Composer 1.0 Architecture Overview. https://www.eclipse.org/epf/composer_architecture/). Accedido Octubre 2019.

[24] Evgeniy Gabrilovich, Shaul Markovitch, et al. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *IJcAI*, volume 7, pages 1606–1611, 2007.

[25] Ken J Gilhooly and Robert H Logie. Age-of-acquisition, imagery, concreteness, familiarity, and ambiguity measures for 1,944 words. *Behavior research methods & instrumentation*, 12(4):395–427, 1980.

[26] Wael H Gomaa and Aly A Fahmy. A survey of text similarity approaches. *International Journal of Computer Applications*, 68(13):13–18, 2013.

[27] Yanette Díaz González and Yenisleidy Fernández Romero. Patrón modelo-vista-controlador. *Revista Telemática*, 11(1):47–57, 2012.

[28] Felipe Ignacio González Martínez. Elección entre procesos automáticamente adaptados y procesos predefinidos. 2015.

[29] Arthur C Graesser, Danielle S McNamara, Max M Louwerse, and Zhiqiang Cai. Coh-metrix: Analysis of text on cohesion and language. *Behavior research methods, instruments, & computers*, 36(2):193–202, 2004.

[30] Julio Ariel Hurtado, María Cecilia Bastarrica, Sergio F Ochoa, and Jocelyn Simmonds. Mde software process lines in small companies. *Journal of Systems and Software*, 86(5):1153–1171, 2013.

[31] Julio A Hurtado Alegría, María Cecilia Bastarrica, Alcides Quispe, and Sergio F Ochoa. An mde approach to software process tailoring. In *Proceedings of the 2011 International Conference on Software and Systems Process*, pages 43–52. ACM, 2011.

[32] ISO 2500. Functional Suitability. https://iso25000.com/index.php/en/iso-25000-standards/iso-25010/58-functional-suitability. Accedido Julio 2020.

[33] ISO 2500. La familia de normas ISO/IEC 25000, year =2020, howpublished = https://iso25000.com/index.php/normas-iso-25000, note = Accedido Julio 2020.

[34] IsoCom n.d. . SO. ISO/IEC 9126-1:2001 - Software engineering – Product quality – Part 1: Quality model. https://www.iso.org/standard/22749.html. Accedido Julio 2020.

[35] Heather A Johnson. Trello. *Journal of the Medical Library Association: JMLA*, 105(2):209, 2017.

[36] Ankur Joshi, Saket Kale, Satish Chandel, and D Kumar Pal. Likert scale: Explored and explained. *Current Journal of Applied Science and Technology*, pages 396–403, 2015.

[37] Frédéric Jouault, Jean Bézivin, and Ivan Kurtev. TCS: A DSL for the specification of textual concrete syntaxes in model engineering. *Proceedings of the 5th International Conference on Generative Programming and Component Engineering, GPCE'06*, (June 2014):249–254, 2006.

[38] Bill Joy, Guy Steele, James Gosling, and Gilad Bracha. The java language specification, 2000.

[39] Stuart Kent. Model driven engineering. In *International Conference on Integrated Formal Methods*, pages 286–298. Springer, 2002.

[40] Per Kroll and Philippe Kruchten. *The rational unified process made easy: a practitioner's guide to the RUP*. Addison-Wesley Professional, 2003.

[41] Laravel. Laravel Queues. https://laravel.com/docs/7.x/queues. Accedido Junio 2020.

[42] Michael Lesk. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In *Proceedings of the 5th annual international conference on Systems documentation*, pages 24–26, 1986.

[43] Patricio Letelier and Mª Carmen Penadés. Metodologías ágiles para el desarrollo de software: extreme programming (xp). 2006.

[44] Yuehua Lin, Jeff Gray, and Frédéric Jouault. Dsmdiff: a differentiation tool for domain-specific models. *European Journal of Information Systems*, 16(4):349–361, 2007.

[45] Yuehua Lin, Jing Zhang, and Jeff Gray. Model comparison: A key challenge for transformation testing and version control in model driven software development. In *OOPSLA Workshop on Best Practices for Model-Driven Software Development*, volume 108, page 6, 2004.

[46] Christopher Manning and Hinrich Schutze. *Foundations of statistical natural language processing*. MIT press, 1999.

[47] Shawn McCool. *Laravel Starter*. Packt Publishing Ltd, 2012.

[48] Srinivas Nidhra and Jagruthi Dondeti. Black box and white box testing techniques-a literature review. *International Journal of Embedded Systems and Applications (IJESA)*, 2(2):29–50, 2012.

[49] Object Management Group (OMG). Software Process Engineering Metamodel (SPEM). 3(2):92–100, 2008.

[50] Allan Paivio, John C Yuille, and Stephen A Madigan. Concreteness, imagery, and meaningfulness values for 925 nouns. *Journal of experimental psychology*, 76(1p2):1, 1968.

[51] PHP. ¿Qué es PHP? https://www.php.net/manual/es/intro-whatis.php. Accedido Julio 2020.

[52] Princeton. WordNet — A Lexical Database for English. https://wordnet.princeton.edu/. Accedido Julio 2020.

[53] Philip Resnik. Using information content to evaluate semantic similarity in a taxonomy. *arXiv preprint cmp-lg/9511007*, 1995.

[54] Douglas C. Schmid. Model-Driven Engineering. 2006.

[55] Ken Schwaber and Mike Beedle. *Agile software development with Scrum*, volume 1. Prentice Hall Upper Saddle River, 2002.

[56] Dag IK Sjøberg, Jo Erskine Hannay, Ove Hansen, Vigdis By Kampenes, Amela Karahasanovic, N-K Liborg, and Anette C Rekdal. A survey of controlled experiments in software engineering. *IEEE transactions on software engineering*, 31(9):733–753, 2005.

[57] Ian Sommerville. *Ingeniería del software*. Pearson educación, 2005.

[58] Mark Summerfield. *Programming in Python 3: a complete introduction to the Python language*. Addison-Wesley Professional, 2010.

[59] Antoine Toulmé and I Inc. Presentation of emf compare utility. In *Eclipse Modeling Symposium*, pages 1–8, 2006.

[60] Amos Tversky. Features of similarity. *Psychological review*, 84(4):327, 1977.

[61] Juan P Uc and E G Gerzon. Aplicación del proceso de experimentación a la Ingeniería de Software Introducción Experimentación en Ingeniería de Software. 8(2013):26–37, 2010.

[62] Arie Van Deursen, Paul Klint, and Joost Visser. Domain-specific languages: An annotated bibliography. *ACM Sigplan Notices*, 35(6):26–36, 2000.

[63] Werner Welte. *Lingüística moderna, terminología y bibliografía*, volume 5. Gredos, 1985.

[64] Dominic Widdows and Dominic Widdows. *Geometry and meaning*, volume 773. CSLI publications Stanford, 2004.

[65] Claes Wohlin. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In *Proceedings of the 18th international conference on evaluation and assessment in software engineering*, page 38. Citeseer, 2014.

[66] World Wide Web Consortium. W3C standards. https://www.w3.org/standards/. Accedido Marzo 2020.

[67] Zhibiao Wu and Martha Palmer. Verb semantics and lexical selection. *arXiv preprint cmp-lg/9406033*, 1994.

ANEXOS

A. Especificación de Experimentos

En el presente anexo se documentan todos los experimentos llevados a cabo durante la experimentación.

A.1. Tarjetas de Experimentación

A continuación se muestran todos los experimentos controlados que son llevados a cabo durante la evaluación de la plataforma. Se muestran además observaciones que se tuvieron en consideración una vez que se llevan a cabo.

Cuadro A.1: Experimento Controlado 01

Codigo	EC01	
Aspecto Evaluado	Funcionalidad,Corrección	
Tipo Usuario	Investigador	
Descripción	Se realiza una importación a la pagina de un proceso en formato XML, exportado desde EPF Composer.	
Entrada	**Salida esperada**	**Salida obtenida**
Proceso DTS-TS en formato XML	▪ Registro en la BD del proceso subido. ▪ Archivo en formato XMI del mismo proceso. ▪ Proceso dentro la biblioteca del usuario.	▪ Proceso Registrado en BD. ▪ Proceso en formato XMI, nombre compuesto por original mas marca de tiempo. ▪ Proceso en biblioteca de usuario. Editable y visible para este.
Observaciones	El Proceso es subido en opción publica, por lo tanto es visible por otros usuarios. La privacidad es modificable a través de la interfaz.	

Cuadro A.2: Experimento Controlado 02

Código	EC02	
Aspecto Evaluado	Funcionalidad,Corrección	
Tipo Usuario	Investigador	
Descripción	Se realiza una importación a la pagina de un proceso en formato XMI, exportado desde EPF Composer y transformado previamente por el extractor-inyector.	
Entrada	**Salida esperada**	**Salida obtenida**
Modelo imagen en formato XMI	▪ Registro en la BD del Modelo subido. ▪ Archivo en formato XMI dentro de las carpetas de la herramienta. ▪ Modelo dentro la biblioteca del usuario.	▪ Modelo Registrado en BD. ▪ Modelo en formato XMI, nombre compuesto por original mas marca de tiempo. ▪ Modelo en biblioteca de usuario. Editable y visible para este.
Observaciones	El Modelo es subido en opción publica, por lo tanto es visible por otros usuarios. La privacidad es modificable a través de la interfaz.	

Cuadro A.3: Experimento Controlado 03

Código	EC03	
Aspecto Evaluado	Funcionalidad,Corrección	
Tipo Usuario	Administrador	
Descripción	Se realiza una importación a la pagina de un proceso estándar XML, exportado desde EPF Composer.	
Entrada	**Salida esperada**	**Salida obtenida**
Proceso Scrum en formato XML.	▪ Registro en la BD del Proceso estándar subido. ▪ Archivo en formato XMI dentro de las carpetas de la herramienta. ▪ Modelo dentro la biblioteca de procesos estándar, visible por el usuario y el administrador, editable para el administrador.	▪ Modelo Registrado en BD. ▪ Modelo en formato XMI, nombre compuesto por original mas marca de tiempo. ▪ Modelo en biblioteca de usuario y administrador. Editable y visible para el administrador, solo visible para el usuario.
Observaciones	El Modelo es correctamente guardado, pero queda en otro directorio dado a su calidad standard.	

Cuadro A.4: Experimento Controlado 04

Código	EC04	
Aspecto Evaluado	Funcionalidad,Corrección	
Tipo Usuario	Administrador	
Descripción	Se realiza una importación a la pagina de un modelo estándar XMI, exportado desde EPF Composer y transformado por el extractor-inyector.	
Entrada	**Salida esperada**	**Salida obtenida**
Modelo XP en formato XMI.	▪ Registro en la BD del Proceso estándar subido. ▪ Archivo en formato XMI dentro de las carpetas de la herramienta. ▪ Modelo dentro la biblioteca de modelos estándar, visible por el usuario y el administrador, editable para el administrador.	▪ Modelo Registrado en BD. ▪ Modelo en formato XMI, nombre compuesto por original mas marca de tiempo. ▪ Modelo en biblioteca de usuario y administrador. Editable y visible para el administrador, solo visible para el usuario.
Observaciones	El Proceso es correctamente guardado, pero queda en otro directorio dado a su calidad estándar.	

Cuadro A.5: Experimento Controlado 05

Código	EC05	
Aspecto Evaluado	Funcionalidad,Corrección	
Tipo Usuario	Investigador	
Descripción	Se realiza una comparación haciendo uso de la función de proceso subidos - proceso subido, con procesos.	
Entrada	**Salida esperada**	**Salida obtenida**
▪ Proceso DTS-TS en formato XML ▪ Proceso DTS-TS' en formato XML	▪ Registro en la BD de la comparación y su work correspondiente. ▪ Salida de de diferencias, se esperan 3 diferencias en dos primeras etapas de comparación. Solo 2 en tercera etapa de comparación.	▪ Comparación registrada en la BD y work correspondiente. ▪ 3 Diferencias en dos primeras fases de comparación, en la tercera solo se aprecian 2.
Observaciones	La tercera diferencia pertenecía a la palabra *Entorno* y *Ambiente*, la cual posee mismo significado, por lo tanto se elimina durante la tercera instancia.	

Cuadro A.6: Experimento Controlado 06

Código	EC06	
Aspecto Evaluado	Funcionalidad,Corrección	
Tipo Usuario	Investigador	
Descripción	Se realiza una comparación haciendo uso de la función de proceso subidos - proceso subido, con modelos.	
Entrada	**Salida esperada**	**Salida obtenida**
▪ Modelo AdaptedRhiscomProcess en formato XMI ▪ Modelo AdaptedRhiscomProcessA en formato XMI	▪ Registro en la BD de la comparación y su work correspondiente. ▪ Salidas sin diferencias.	▪ Comparación registrada en la BD y work correspondiente. ▪ No hay diferencias en ninguna de las 3 fases de comparación.
Observaciones	Los modelos solo poseían una diferencia según EMF Compare, la cual pertenecía a la acentuación de una palabra, esta no es considerada por la metodología del comparador. La estructura interna del modelo no esta completamente especificada en español, además sus elementos no presentan ID, lo que produce errores de visualización.	

Cuadro A.7: Experimento Controlado 07

Código	EC07	
Aspecto Evaluado	Funcionalidad,Corrección	
Tipo Usuario	Investigador	
Descripción	Se realiza una comparación haciendo uso de la función de proceso de mi biblioteca - proceso de mi biblioteca.	
Entrada	**Salida esperada**	**Salida obtenida**
▪ Proceso Imagen almacenado en formato XMI ▪ Proceso Imagen almacenado en formato XMI	▪ Registro en la BD de la comparación y su work correspondiente. ▪ Salidas con 2 diferencias en las dos primeras etapas de la comparación, solo 1 en tercera.	▪ Comparación registrada en la BD y work correspondiente. ▪ Salida con 2 diferencias en las dos primeras etapas, solo 1 en la tercera.
Observaciones	Los modelos poseían dos diferencias, una era completamente diferente , sin embargo la segunda solo se veía reflejada entre las palabras sinónimo. Dichas palabras correspondían a *Peligro* y *Riesgo*, ambas precedidas por *Análisis de...*	

Cuadro A.8: Experimento Controlado 08

Código	EC08	
Aspecto Evaluado	Funcionalidad,Corrección	
Tipo Usuario	Investigador	
Descripción	Se realiza una comparación haciendo uso de la función de proceso de mi biblioteca - proceso publico.	
Entrada	**Salida esperada**	**Salida obtenida**
▪ Proceso Imagen almacenado en formato XMI ▪ Proceso ImagenPublic almacenado de manera publica en formato XMI	▪ Registro en la BD de la comparación y su work correspondiente. ▪ Salidas con 2 diferencias en la primera etapa, 1 para las que sigan.	▪ Comparación registrada en la BD y work correspondiente. ▪ Salida con 2 diferencias solo en la primera etapa, 1 para las siguientes.
Observaciones	Los modelos poseían una diferencia referente a la estructura de una frase, por lo tanto esta al ser tan similar en ambos y por la cantidad de palabras que poseía, fue considerada*match* en la segunda etapa.	

Cuadro A.9: Experimento Controlado 09

Código	EC09	
Aspecto Evaluado	Funcionalidad,Corrección	
Tipo Usuario	Investigador	
Descripción	Se realiza una comparación haciendo uso de la función de proceso de mi biblioteca - proceso estándar.	
Entrada	**Salida esperada**	**Salida obtenida**
▪ Proceso Rhiscom versión 1 XMI ▪ Proceso Rhiscom Standard Test almacenado de manera estándar en formato XMI	▪ Registro en la BD de la comparación y su work correspondiente. ▪ Comparación satisfactoria, sin diferencias en ninguna fase.	▪ Comparación registrada en la BD y work correspondiente. ▪ Salida sin diferencias en ninguna fase.
Observaciones	Los Modelos poseían una estructura casi igual, lo cual permitió una comparación sin diferencias.	

B. Documentación de desarrollo

En el siguiente Anexo se dejan todos los productos del desarrollo de software llevado a cabo.

B.1. Especificación Formal de Requisitos

A continuación la Tabla B.1muestra la especificación formal de requisitos llevada a cabo con cada uno de estos.

Cuadro B.1: Tabla de especificación de requisitos

Código	Nombre	Descripción	Estabilidad	Tipo	Usuario
RU001	Acceso de Usuarios	Permitir acceder a los usuarios	Alta	Funcional	Investigador-Administrador
RU002	Creación de perfil	Permitir la creación de un perfil de usuario	Alta	Funcional	Investigador-Administrador
RU003	Importar Modelos	Importar un Modelo hacia la pagina en formato XML	Media	Funcional	Investigador-Administrador

Código	Nombre	Descripción	Estabilidad	Tipo	Usuario
RU004	Visualizar Modelos	Visualizar los modelos importados	Media	Calidad de Software	Investigador-Administrador
RU005	Comparar Modelos Almacenados	Comparar modelos con los ya almacenados en la plataforma	Media	Funcional	Investigador
RU006	Ver resultados de la comparación	Permitir ver resultados mas aproximados en la comparación de modelos	Media	Funcional	Investigador
RU007	Comparar Modelos subidos	Permitir comparar modelos subidos por el usuario	Media	Funcional	Investigador
RU008	Registrar usuarios	Permitir que le administrador pueda registrar usuarios	Media	Funcional	Administrador
RU009	Registrar usuarios	Visualizar estado de usuarios	Media	Calidad de Software	Administrador

Código	Nombre	Descripción	Estabilidad	Tipo	Usuario
RU010	Ver actividad de usuarios	Permitir al administrador ver un *dashboard* de los usos de la aplicación	Media	Calidad de Software	Administrador

B.2. Diagramas

A continuación se muestran los diagramas concebidos durante el desarrollo. La Figura muestra el diagrama del modelo relacional final, en el cual no son consideradas las entidades nativas del *framework* Laravel Y la Figura muestra el diagrama de la arquitectura Lógica.

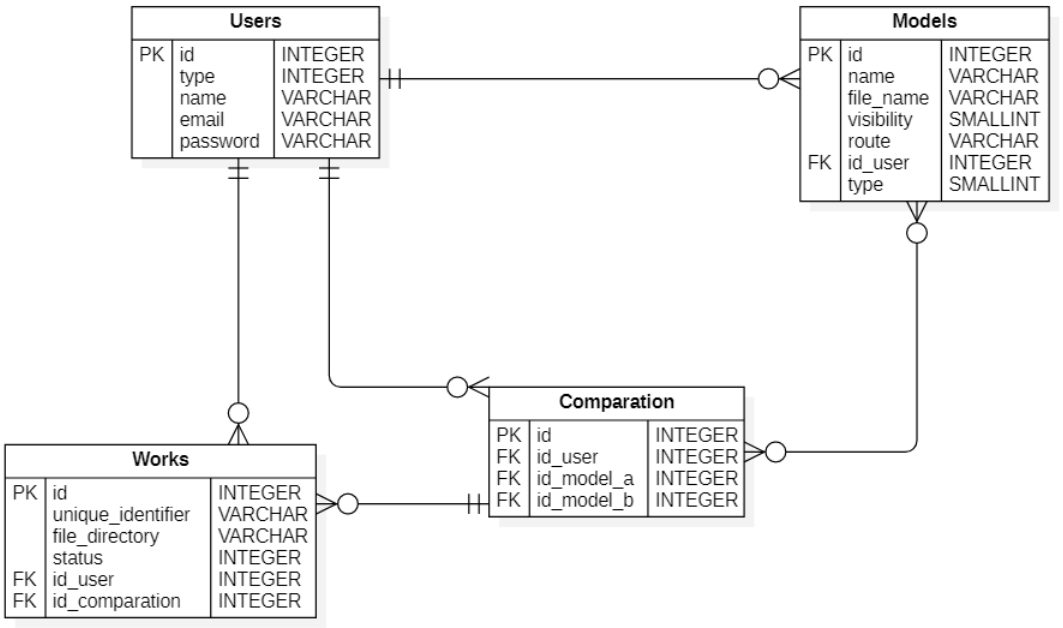


Figura B.1: Diagrama Modelo Relacional Final

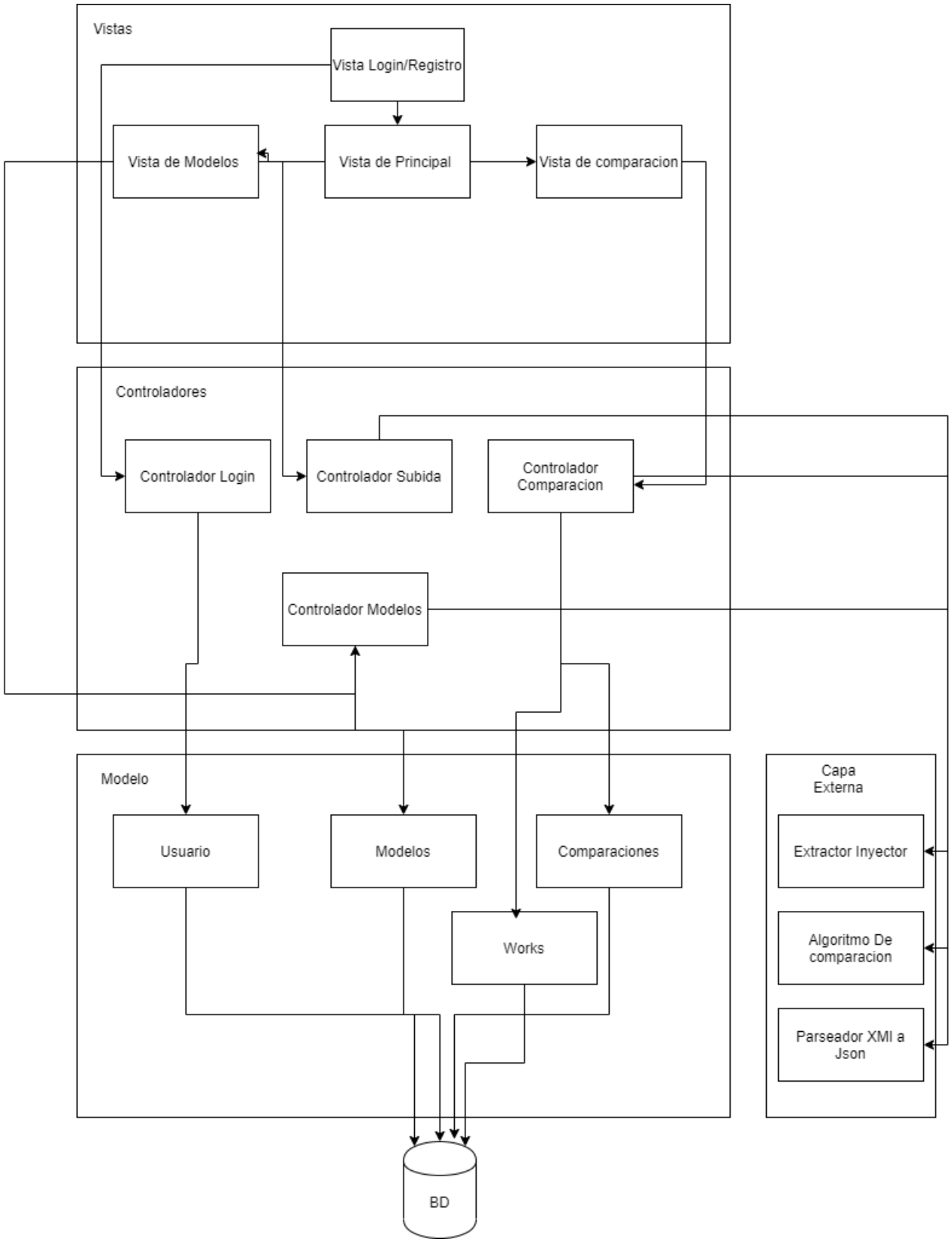


Figura B.2: Diagrama Arquitectura Lógica

B.3. Capturas Plataforma Web

A continuación se dejan algunas capturas de las interfaces implementadas dentro de la plataforma. Estas se clasifican respecto a los usuarios considerados dentro de la plataforma.

B.3.1. Capturas Usuario Investigador

A continuación se muestran las capturas de la interfaz del usuario Investigador.

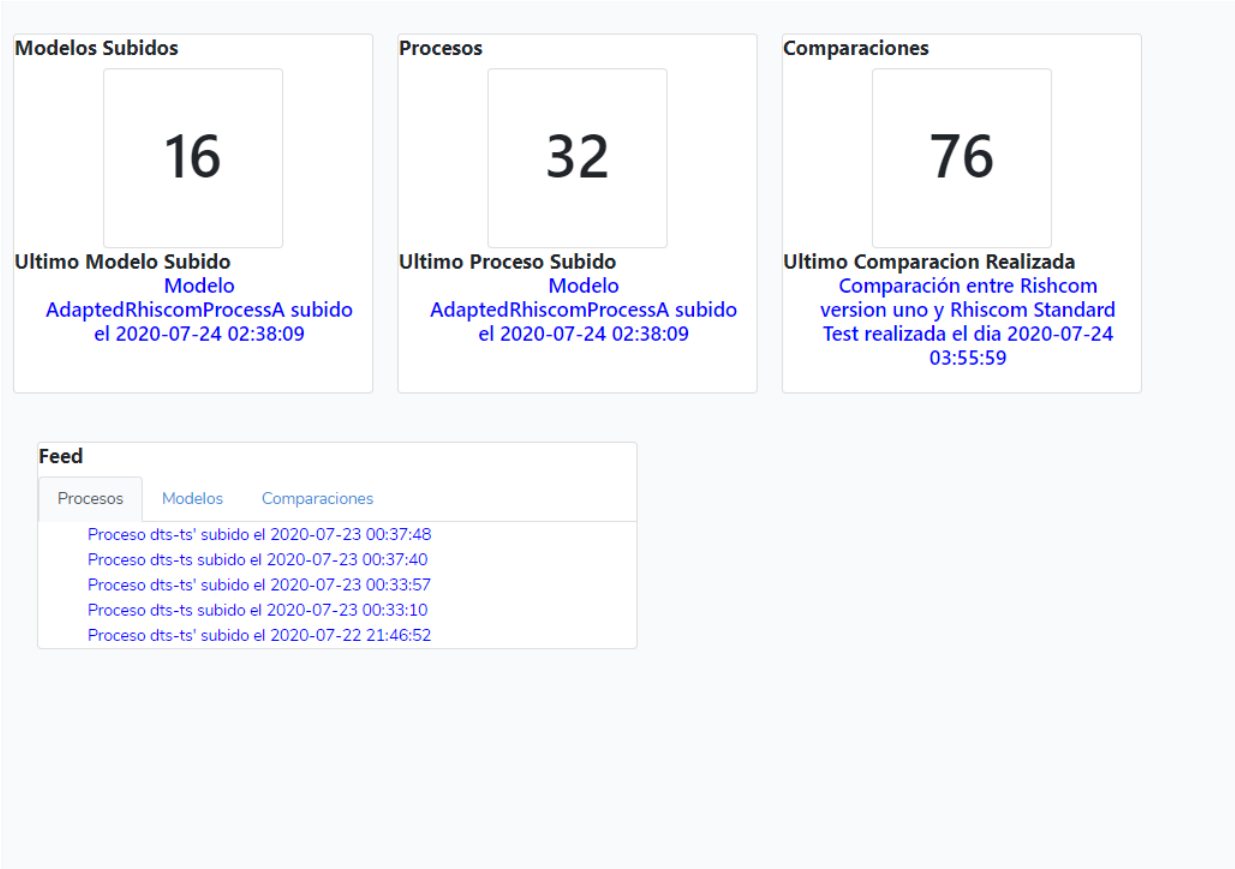


Figura B.3: Interfaz Principal Investigador

Figura B.4: Interfaz Biblioteca de Procesos del Usuario

Figura B.5: Interfaz Biblioteca de Procesos Públicos

Figura B.6: Interfaz Principal Investigador

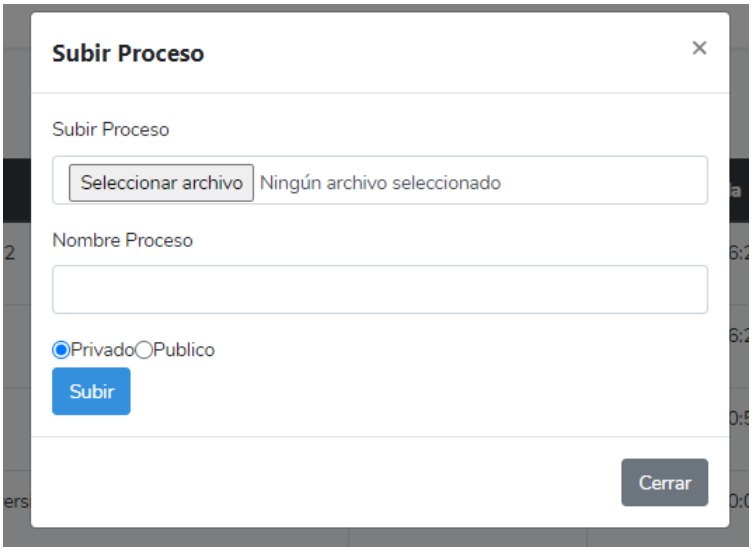


Figura B.7: Modal Subir Proceso o Modelo a la plataforma

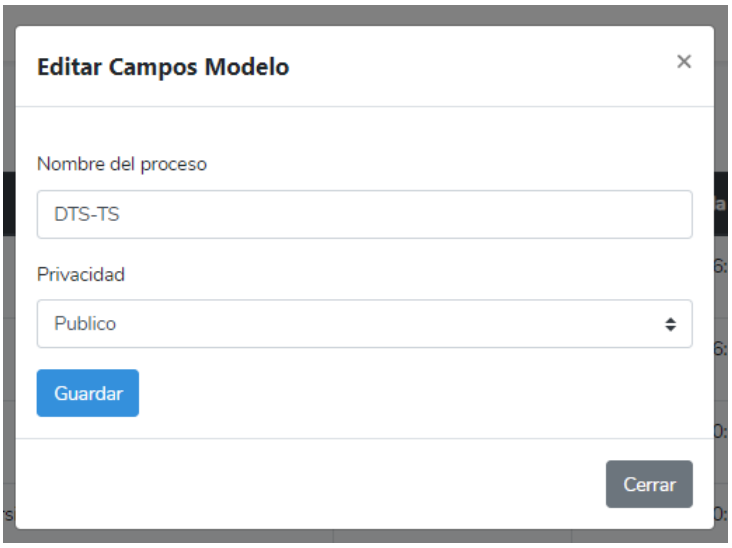


Figura B.8: Modal Editar Proceso o Modelo

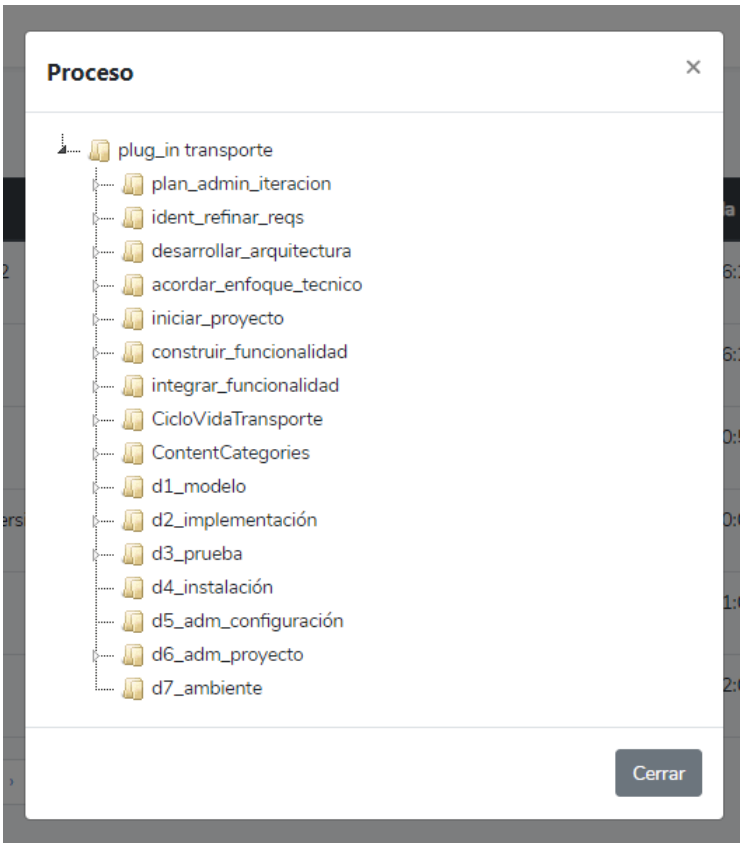


Figura B.9: Modal Visualización de Proceso o Modelo

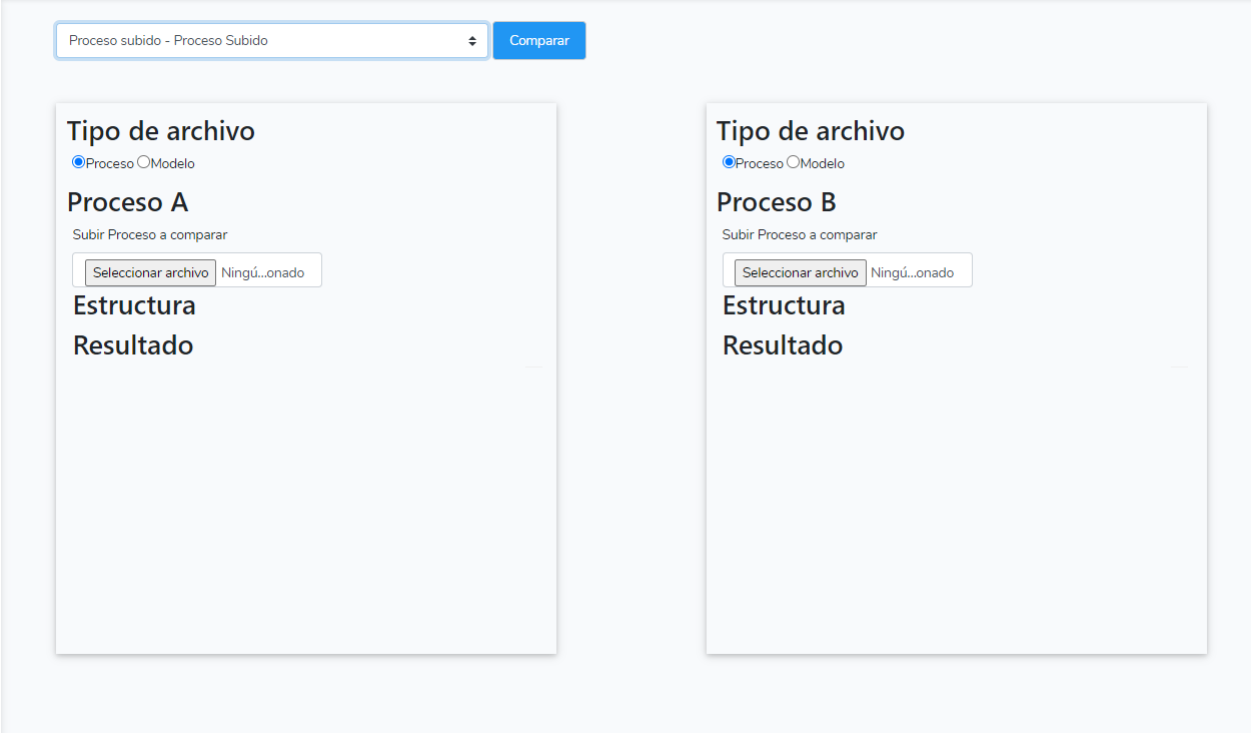


Figura B.10: Interfaz Comparación de Proceso Subido

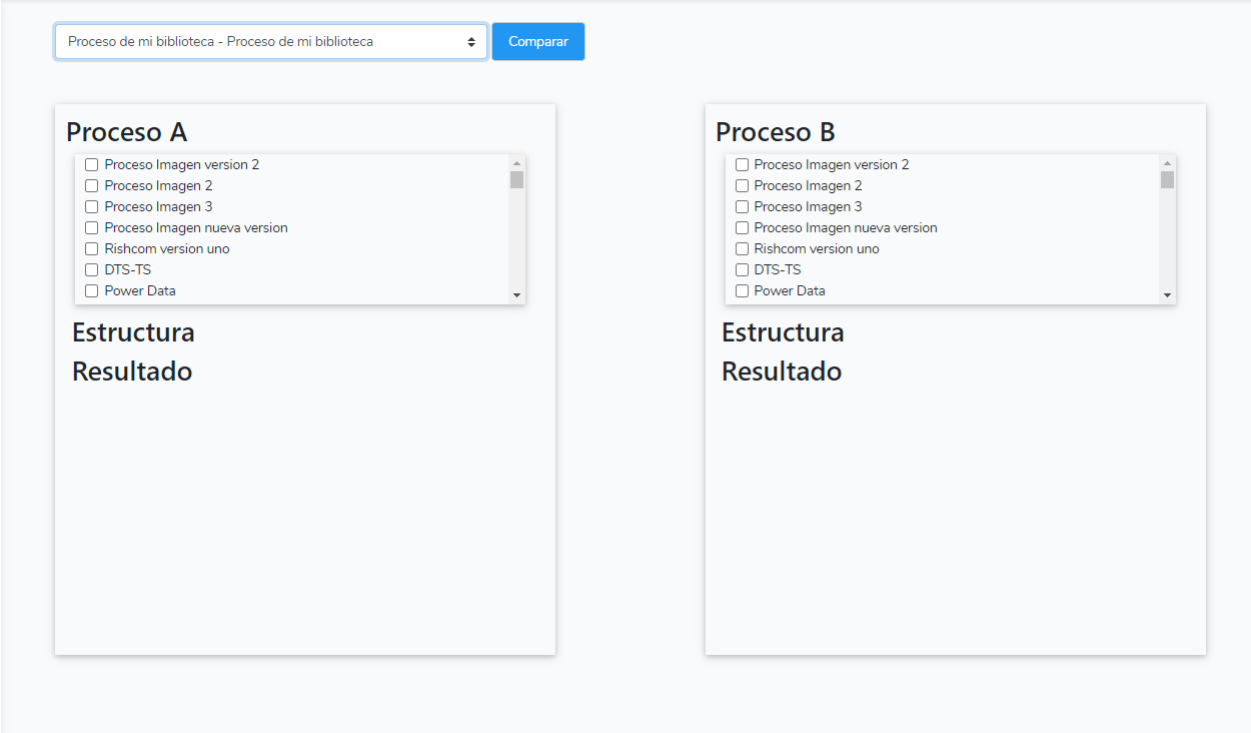


Figura B.11: Interfaz Comparación de Proceso Biblioteca de Usuario

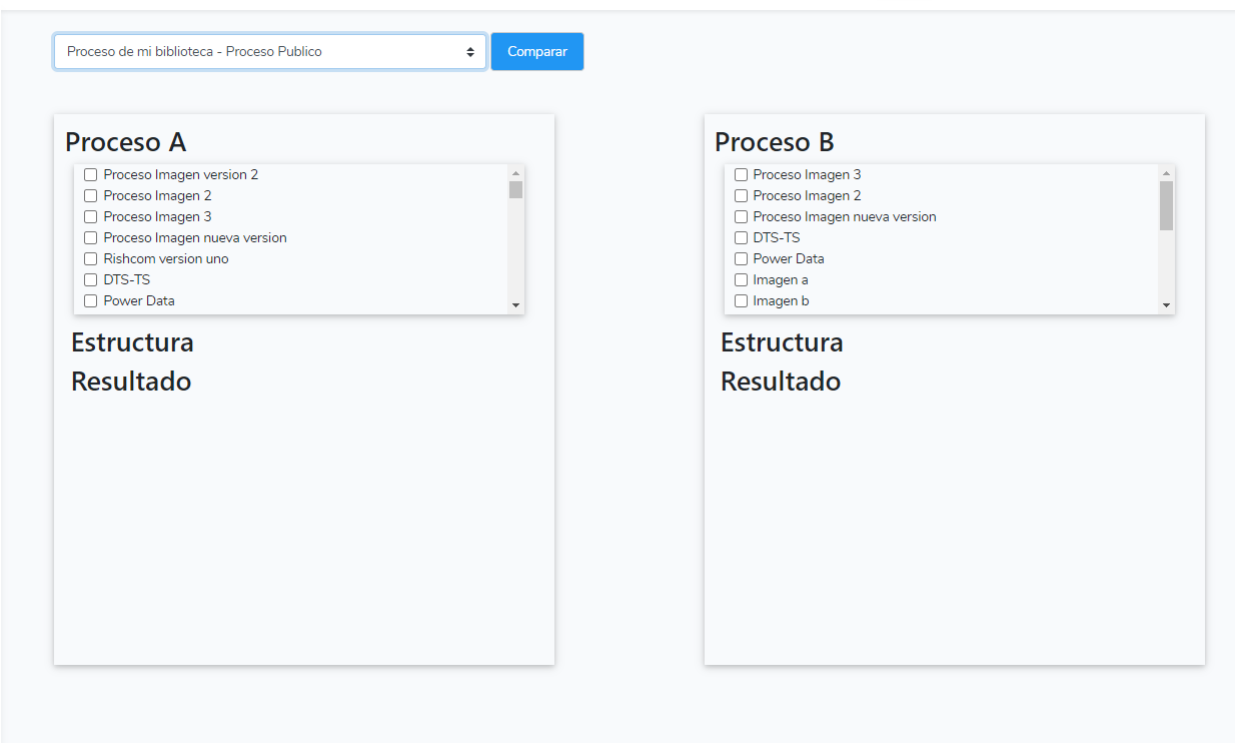


Figura B.12: Interfaz Comparación de Proceso Biblioteca de Usuario contra Proceso Publico

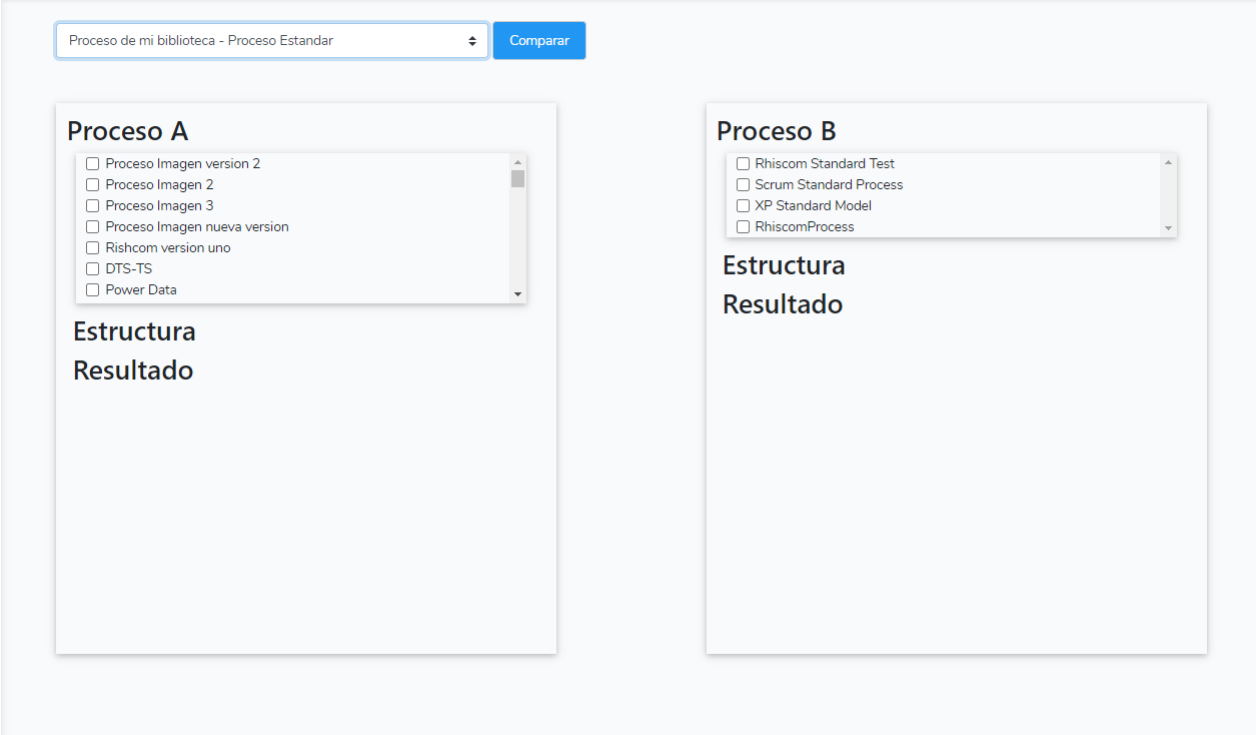


Figura B.13: Interfaz Comparación de Proceso Biblioteca de Usuario contra Proceso Estándar

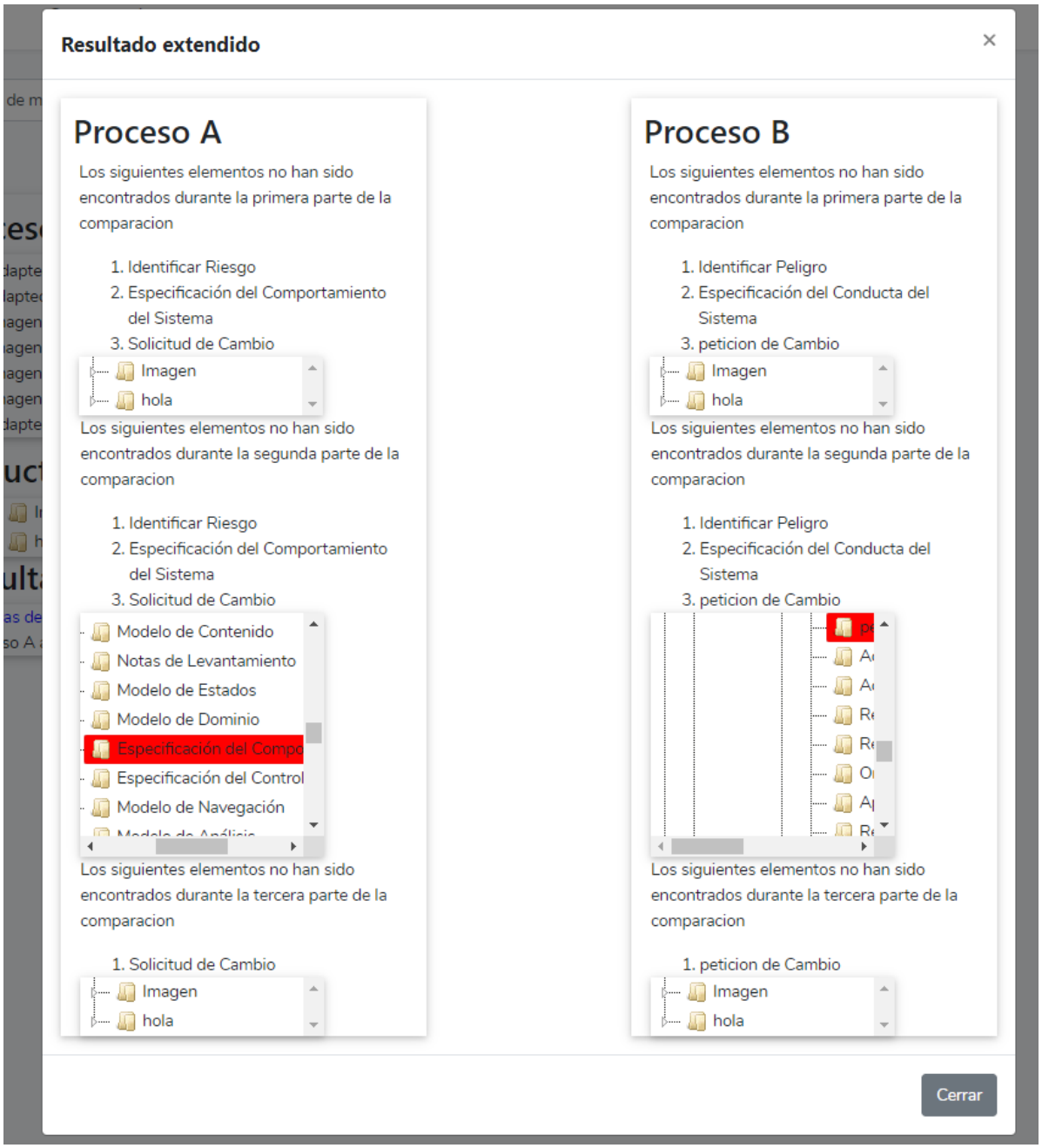


Figura B.14: Modal Resultado Extendido de Comparación

Figura B.15: Apartado Notificaciones Comparación Terminada

B.3.2. Capturas Usuario Administrador

A continuación se muestran las capturas de las interfaces del usuario Administrador.

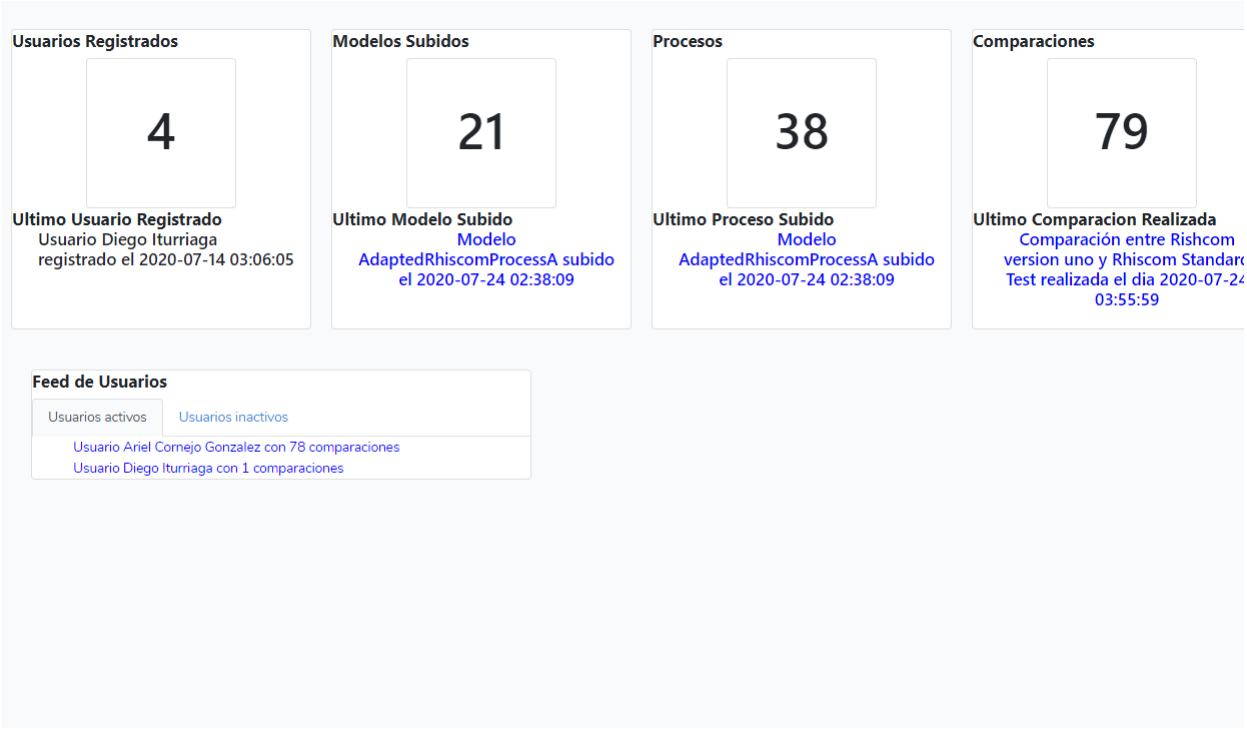
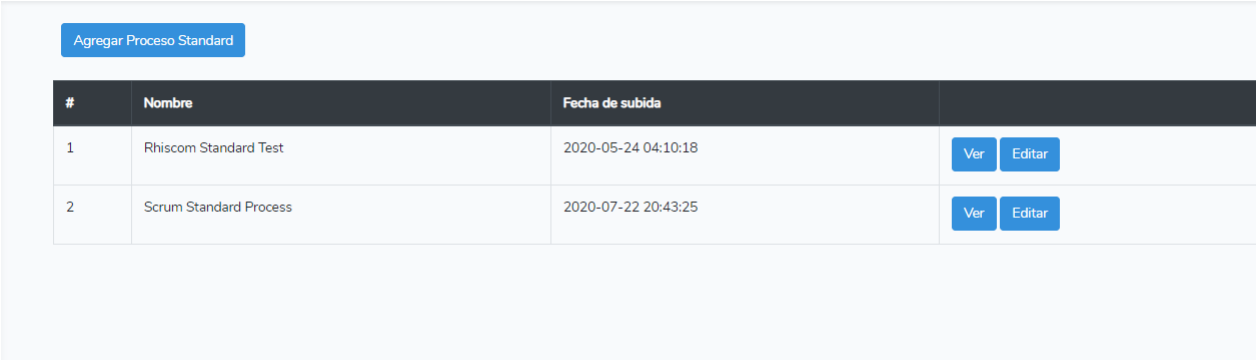


Figura B.16: Interfaz Principal Usuario Administrador

#	Nombre	Privacidad	Autor	Fecha de subida	
1	Proceso Imagen version 2	Privado	Ariel Cornejo Gonzalez	2020-05-04 16:20:14	Ver
2	Proceso Imagen 2	Privado	Ariel Cornejo Gonzalez	2020-05-04 16:20:36	Ver
3	Proceso Imagen 3	Publico	Ariel Cornejo Gonzalez	2020-05-05 00:52:21	Ver
4	Proceso Imagen nueva version	Publico	Ariel Cornejo Gonzalez	2020-05-09 20:05:40	Ver
5	Rishcom version uno	Privado	Ariel Cornejo Gonzalez	2020-05-12 21:03:50	Ver

‹ 1 2 3 4 5 6 7 8 9 10 ›

Figura B.17: Interfaz Biblioteca Total de Modelos



Agregar Proceso Standard

#	Nombre	Fecha de subida	
1	Rhiscom Standard Test	2020-05-24 04:10:18	Ver Editar
2	Scrum Standard Process	2020-07-22 20:43:25	Ver Editar

Figura B.18: Interfaz Biblioteca de Modelos Estándar

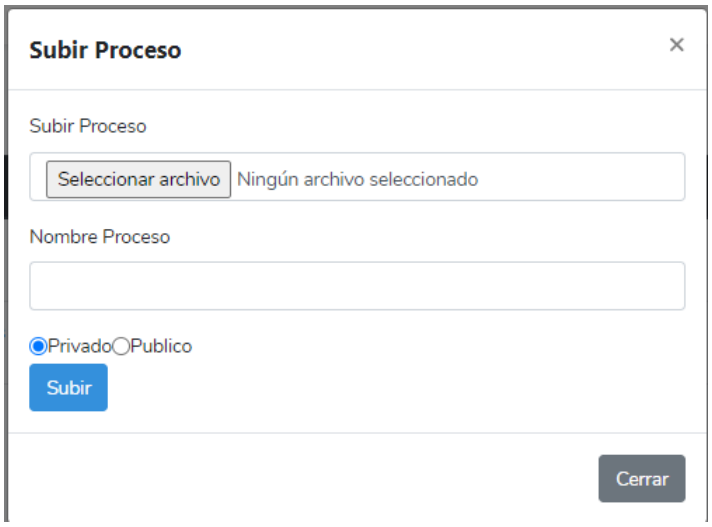


Figura B.19: Modal subida Proceso

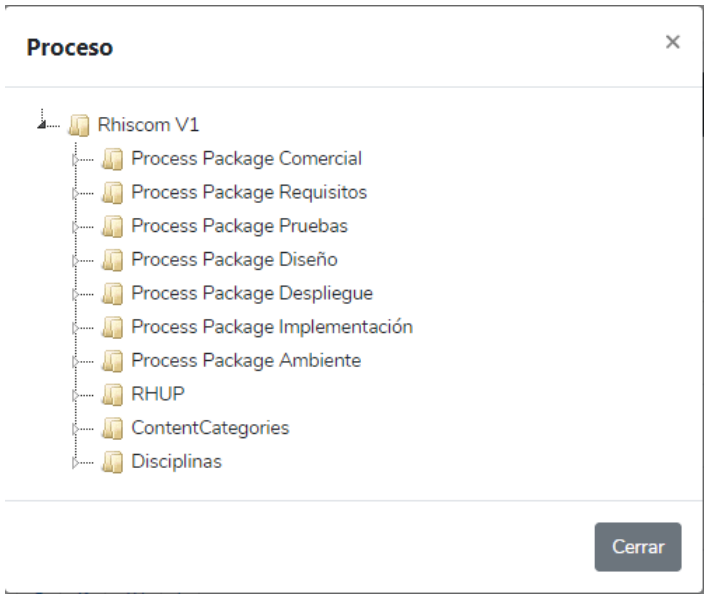


Figura B.20: Modal Visualización Proceso

C. Encuesta De Evaluación de Pro-MoCot

Encuesta de evaluación del la plataforma ProMoCot

Se ha desarrollado una plataforma que permite la comparación de modelos de proceso de software que hayan sido construidos en EPF Composer. En concreto la plataforma permite la importación y visualización de modelos, así como compartir estos de manera publica entre usuarios. Se permite la comparación entre modelos propios o con modelos de otros usuarios y también modelos de proceso de software estándar (RUP, Scrum, XP, etc...), visualizando ademas los resultados de dichas comparaciones .

Esta encuesta se realiza para evaluar la funcionalidad, correcctitud y utilidad del sistema.
* Required

Datos Generales

Ingeniería Civil en Computación - Universidad de Talca

1. Edad *

Marca una opción

Mark only one oval.

- ◯ menor a 18 años
- ◯ 18 - 21 años
- ◯ 22 - 25 años
- ◯ mayor a 25 años
- ◯ Other: _____

2. Genero *

Marca una opción

Mark only one oval.

- ◯ Masculino
- ◯ Femenino

3. Ocupación *

Marcar su principal ocupación para caracterizar a los participantes.

Mark only one oval.

- ◯ Estudiante
- ◯ Empleado dependiente
- ◯ Empleado independiente
- ◯ Other: _____

4. Actividad económica *

Indicar la principal actividad que realiza de acuerdo a su profesión u ocupación (ejemplo: Microempresario, Ingeniero, Motoboy, Estudiante, etc.)

_____

Skip to question 5

Evaluación de la Plataforma ProMoCot

En esta sección, debe seleccionar el grado de deacuerdo o desacuerdo referente a la utilización del sistema.

5. Funcionalidad de gestión de usuario *

Mark only one oval per row.

	Totalmente en desacuerdo	En desacuerdo	Ni en desacuero ni de acuerdo	De acuerdo	Totalmente de acuerdo
La plataforma es capaz de registrar un perfil de usuario.					
La plataforma permite iniciar sesión con las credenciales de usuario.					
La plataforma permite ver los datos de actividad del usuario.					

6. Funcionalidad de gestión de modelos *

Mark only one oval per row.

	Totalmente en desacuerdo	En desacuerdo	Ni en desacuero ni de acuerdo	De acuerdo	Totalmente de acuerdo
La plataforma permite importar un modelo en formato XML o XMI.					
La plataforma permite visualizar los modelos subidos por el usuario.					
La plataforma permite visualizar los modelos subidos por otros usuarios.					
La plataforma permite visualizar los modelos estándar.					
La plataforma permite visualizar la estructura de los modelos					

7. Funcionalidad de Comparación *

Mark only one oval per row.

	Totalmente en desacuerdo	En desacuerdo	Ni en desacuero ni de acuerdo	De acuerdo	Totalmente de acuerdo
La plataforma permite comparar modelos subidos					
La plataforma permite comparar modelos almacenados propios					
La plataforma permite comparar modelos almacenados propios con los de otros usuarios					
La plataforma permite comparar modelos almacenados propios con modelos estándar					
La plataforma permite visualizar de manera gráfica los resultados de la comparación					

Evaluación de la Plataforma ProMoCot

En esta sección, debe seleccionar el grado de deacuerdo o desacuerdo referente a la utilización del sistema.

8. Correctitud del sistema *

Mark only one oval per row.

	Totalmente en desacuerdo	En desacuerdo	Ni en desacuerdo ni de acuerdo	De acuerdo	Totalmente de acuerdo
La plataforma realiza correctamente las comparaciones					
La plataforma muestra la estructura correcta de los modelos					

9. Utilidad del sistema *

Mark only one oval per row.

	Totalmente en desacuerdo	En desacuerdo	Ni en desacuerdo ni de acuerdo	De acuerdo	Totalmente de acuerdo
La plataforma muestra notificaciones en caso de comparaciones muy largas.					
La plataforma muestra datos históricos de las actividades hechas					
La plataforma permite ver un registro de las ultimas actividades realizadas					

Evaluación de la Plataforma ProMoCot

En esta sección, debe seleccionar el grado de deacuerdo o desacuerdo referente a la apreciación general del sistema.

10. Cómo le ha parecido el sistema web respecto a: *

Mark only one oval per row.

	Totalmente en desacuerdo	En desacuerdo	Ni en desacuerdo ni de acuerdo	De acuerdo	Totalmente de acuerdo
Funcionalidad de la plataforma ProMoCot					
Correcctitud de la plataforma ProMoCot					
Utilidad de la plataforma ProMoCot					

Preguntas

En esta sección, se solicita responder las siguientes preguntas.

11. ¿La plataforma presentó algún un error al momento de utilizarse? *

12. ¿Qué cambios o mejoras le haría usted a la plataforma? *

13. ¿Utilizaría la plataforma ProMoCot para realizar comparaciones de modelos de proceso de software? *