UNIVERSIDAD CENTRAL DE VENEZUELA
FACULTAD DE CIENCIAS
ESCUELA DE COMPUTACIÓN
CENTRO DE INGENIERÍA DE SOFTWARE Y SISTEMAS

Desarrollo de la aplicación web ProAgil. Módulos: diagramación y prototipaje

Trabajo especial de grado presentado ante la
Ilustre Universidad Central de Venezuela por
Br. María Francis Malavé Briceño C.I. 21123806

Tutora: Profa. Jossie Zambrano

Ciudad Universitaria de Caracas, Julio 2015

Universidad Central de Venezuela

Facultad de Ciencias

Escuela de Computación

ACTA

Quienes subscriben, miembros del jurado designado por el Consejo de la Escuela de Computación de la Facultad de Ciencias de la Universidad Central de Venezuela, para examinar el Trabajo Especial de Grado titulado: "Desarrollo de la aplicación web ProAgil. Módulos: diagramación y prototipaje", presentado por la bachiller: María Francis Malavé Briceño, C.I: 21.123.806, a fin de optar por el título de Licenciada en Computación, dejan constancia de lo siguiente: Leído el trabajo por cada uno de los miembros del jurado, se fijó el día Jueves 06 de agosto de 2015, a las 05:00 pm, en la Escuela de Computación, para que su autora lo defendiera en forma pública, mediante una presentación oral de su contenido, luego de la cual respondió a las preguntas formuladas. Finalizada la defensa pública del Trabajo Especial de Grado, el jurado decidió **aprobarlo**. En fe de lo cual se levanta la presente Acta, en Caracas a los seis (06) días del mes de Agosto del año dos mil quince (2015).

Profa. Jossie Zambrano – Tutora

Profa. Lucía Ojeda – Jurado

Prof. Eugenio Scalise – Jurado

Dedicatoria

A mi mamá y a mi papá
Todos mis logros siempre han sido gracias y por ustedes

A mi tío José Briceño
Eres un angelito que siempre nos cuidará. Sé lo mucho que significaba para ti que todos obtuviéramos un título. El mío te la dedico.

Agradecimientos

A Dios, por bendecirme cada día y por colocar en mi camino a personas con las que he aprendido y vivido momentos inolvidables.

A mis padres, Rafaelina y Robert. Gracias por su apoyo y amor incondicional, por todos los sacrificios que han hecho por mí, por sus consejos. Sin ustedes no lo hubiera podido lograr. Los amo.

A mis hermanos, Laura Alejandra y Robert José. Gracias por consentirme y ser los mejores hermanos mayores del mundo. Los quiero mucho.

A mi familia. Todas las situaciones que hemos vivido solo ratifican nuestro amor y lo unidos que somos en las buenas y en las malas. Gracias por siempre estar ahí.

A mis compañeros. Grinde, Mar, Isthar, Hai, Aldemaro, Alexis, Jorge, Juan y Cano. Son los mejores amigos del mundo, gracias por su apoyo y por compartir conmigo momentos tan bonitos. Les deseo el mayor de los éxitos.

A la Universidad Central de Venezuela. Fuiste mi segundo hogar por muchos años. Qué agradecida y orgullosa estoy de haberme formado como profesional en "la casa que vence la sombra". ¡Por siempre Ucvista!

A Jossie, por depositar su confianza en mí, por su tiempo, compromiso y guiarme en este retador pero satisfactorio recorrido.

Resumen

Las metodologías ágiles para el desarrollo de software, hoy en día utilizan artefactos y técnicas que permiten especificar, estructurar y maquetar los requerimientos necesarios. Esta investigación se centra en tres de ellos: *Diagrama de casos de uso*; son una secuencia de interacciones que se desarrollan entre un sistema y sus actores en respuesta a un evento que inicia un actor sobre el propio sistema. Además sirven para especificar el comportamiento de un sistema mediante su interacción con los usuarios y otros sistemas. *Diagrama de objetos de dominio*; útil para determinar cuáles son los objetos inherentes a la problemática y sus relaciones, y los *Prototipos de interfaz* de usuario; que permiten crear y refinar el diseño y maquetación. Para ello se debe contar con una herramienta centralizada que permita construirlos, generarlos y compartirlos. Por esta razón este Trabajo Especial de Grado tiene como objetivo desarrollar una aplicación web denominada «ProAgil» específicamente los módulos: diagramación y prototipaje, aplicando la Metodología de Desarrollo de Software AgilUs, para proveer en los usuarios de un proyecto el uso y la disponibilidad de este tipo de artefactos de gran importancia para el desarrollo de software.

Palabras clave: ProAgil, AgilUs, Desarrollo ágil, prototipos, casos de uso, objetos del dominio.

Índice

Introducción

Actualmente, con el auge que ha surgido con las aplicaciones web es común que grande empresas apuesten por este tipo de tecnología, beneficiándose de las ventajas que ellas ofrecen, que van desde la reducción de costos en tiempo y recursos hasta la automatización de procesos. Sin embargo, con la competitividad que existe en la web, las organizaciones deben ingeniárselas para que los usuarios se sientan atraídos por sus aplicaciones. Uno de los factores o atributos determinantes que influyen en la atracción de usuarios en la calidad, utilidad y facilidad que éstas deben proveer. Esta característica es lo que se conoce como usabilidad, la cual se define como la capacidad del producto software para ser entendido, aprendido, usado y ser atractivo para el usuario, cuando se usa bajo condiciones específicas.

Para poder evaluar el grado de usabilidad de un software se deben tomar en cuenta artefactos y técnicas que ayuden a determinar si el sistema cumple con las expectativas de los usuarios. Los artefactos deben aplicarse bajo un método de desarrollo, con la finalidad de saber cuándo y cómo aplicarlos. En la actualidad se tiende aplicar metodologías ágiles que emplean un enfoque iterativo e incremental en el desarrollo del software y que son flexibles a los imprevistos que surgen en el ciclo de vida de la producción.

Es importante destacar que al momento de realizar un software es retador tener una buena comunicación con el grupo de desarrollo cuando no se lleva seguimiento de los comentarios, discusiones y tareas asignadas a cada miembro, por lo que se pierde trazabilidad y desmejora la organización de la producción repercutiendo en el producto final.

Por todo ello, hoy en día existen muchas herramientas que permiten gestionar proyectos, facilitando la organización de la información y mejorando la

comunicación del equipo de trabajo, de manera que se trabaja en un entorno colaborativo y centralizado.

Tal es el caso de la aplicación ProAgil, la cual es una propuesta hecha por Centro de Ingeniería de Software y Sistemas de la Escuela de Computación de la Facultad de Ciencias. ProAgil es una aplicación web que permite generar y aplicar artefactos de evaluación de usabilidad de manera automatizada y centralizada. Está fundamentada bajo el enfoque de metodologías ágiles, que permiten adaptarse a los cambios que surgen durante el proceso de desarrollo del software; es por ello que en ella se aplican artefactos que facilitan la puesta en práctica de cualidades de usabilidad con la finalidad de que el producto final sea satisfactorio y fácil de usar para el usuario que lo utilice. Tomando en cuenta esta característica, ProAgil está compuesta por dos módulos: Artefactos de indagación y artefactos de inspección.

A través de los artefactos de indagación se pueden determinar los requerimientos y características del software a desarrollar. En el caso de los artefactos de inspección permiten encontrar errores o inconsistencias en el sistema. Dentro de las metodologías ágiles también existen otros artefactos que permiten especificar, estructurar y diseñar el software. Como por ejemplo los diagramas de casos de uso, los diagramas de objetos de dominio y el prototipo de interfaz de usuario.

Los diagramas de casos de uso permiten al desarrollador la posibilidad de identificar las interacciones entre el usuario y las funcionalidades que provee el sistema. Por su fácil interpretación ayuda a la comunicación entre los desarrolladores y el cliente. Con los diagramas de objetos de dominio se determinan los objetos y las relaciones entre ellos, por lo que se delimita el sistema y contribuye en el modelado de datos. Por otro lado está el prototipo de interfaz de usuario, en la cual básicamente se realiza un diseño de baja fidelidad de las interfaces de usuario del sistema, con la finalidad de tener una primera versión de la estructura y organización del software. En resumen el modelado va a ayudar a

determinar las funcionalidades y entidades que estarán presentes en el sistema, tomando en cuenta las necesidades que requiere el cliente.

En tal sentido, el presente trabajo se centra en desarrollar un módulo para la aplicación ProAgil que automatice los artefactos de diagramación y prototipaje para el desarrollo de software. Este Trabajo Especial de Grado se encuentra estructurado de la siguiente manera.

En el capítulo 1, *Planteamiento del problema,a* se describe brevemente la definición y funcionalidades principales de la aplicación ProAgil, para luego definir los conceptos básicos de la metodología AgilUs y finalizar con la problemática de los módulos de diagramación y prototipaje.

El capítulo 2, *Desarrollo de los módulos de diagramación y prototipaje,* corresponde a la descripción detallada de las iteraciones que se llevaron a cabo para la construcción del software, basándose en el método de desarrollo AgilUs, así como los artefactos y actividades generados en cada fase.

Seguidamente se exponen las conclusiones y recomendaciones y finalmente se presentan las referencias bibliográficas consultadas.

CAPÍTULO I. Planteamiento del Problema

Actualmente en la producción de software es importante incorporar aspectos de usabilidad, siendo esta una cualidad que se refiere a la capacidad de un software a ser funcionalmente correcto, eficiente de usar y fácil de aprender (Nielsen, J., 2006), de manera que contribuya a obtener productos de calidad. Una de las formas que facilitan incorporar este aspecto es el uso de metodologías ágiles en el desarrollo de software, las cuales buscan ser más adaptables a los continuos cambios que se presentan durante el desarrollo de un sistema y para esto emplean un enfoque iterativo e incremental, además de estar centrados en la incorporación del usuario en tempranas etapas de desarrollo del software.

Asimismo, cabe destacar que otro aspecto clave para la construcción de software es gestionar su desarrollo, mecanismo que va a permitir establecer la planificación y organización de todo el proceso de desarrollo, permitiendo al equipo de trabajo cumplir con los objetivos planteados. Para la gestión de desarrollo del software se utilizan herramientas que permitan llevar el control y seguimiento del proyecto. Generalmente, estas herramientas cuentan con funcionalidades que permiten configurar el proyecto; que van desde los artefactos, actividades o técnicas a utilizar en el desarrollo así como la asignación de estas a los miembros del equipo de manera que se tenga control y coordinación sobre los recursos y personas involucradas.

En la Escuela de Computación de la Facultad de Ciencias, el Centro de Ingeniería de Software y Sistemas propone el desarrollo de una aplicación que permita la configuración de proyectos, la gestión y asignación de actividades al grupo de desarrollo de manera centralizada y colaborativa, ofreciendo artefactos y técnicas que faciliten incorporar aspectos de usabilidad en sus proyectos. En el presente capítulo se describe el software, denominado ProAgil, además se introduce la metodología ágil que se utiliza de base para el desarrollo de la aplicación.

1.1 ProAgil

ProAgil es una aplicación web que permite generar y aplicar artefactos de evaluación de usabilidad de manera automatizada y centralizada, específicamente artefactos de inspección e indagación. (Jimenez, S. & Diaz, A., 2015). Una de las características de la aplicación es que permite gestionar proyectos, por lo que potencia la comunicación entre el equipo de desarrollo, además de poder configurar ayuda a fomentar la organización y planificación del trabajo al tener la información centralizada.

Proagil establece dos actores principales: El lider del proyecto quién es el encargado de crear, gestionar y asignar los artefactos de la aplicación; y el colaborador, a quien se le asigna las actividades y artefactos.

Los artefactos están categorizados en dos módulos principales: Módulo de indagación y módulo de inspección. A continuación se explica brevemente en qué consisten los módulos mencionados.

1.1.1 Módulos de indagación

Los artefactos o técnicas que se agrupan en esta categoría proporcionan información acerca de la usabilidad de un producto que aún no se ha empezado a construir. En estas técnicas se debe hacer un gran trabajo de hablar con los usuarios y observarlos usando el sistema en tiempo real, o analizando respuestas que ellos dan a preguntas verbales y/o escritas (Acosta, A., 2013)

Los artefactos que conforman la categoría de indagación son: tormenta de ideas, análisis de sistemas existentes y sondeos.

- Tormenta de ideas: consiste en la generación de ideas por parte de un grupo multidisciplinario, liberando la mente para proponer cualquier idea, permitiendo, así, la libertad para la creatividad. En el caso del proceso de desarrollo de software, corresponde al equipo de desarrollo

realizar esta técnica, incorporando al usuario final. En la figura 1.1 se muestra un ejemplo de tormenta de ideas.

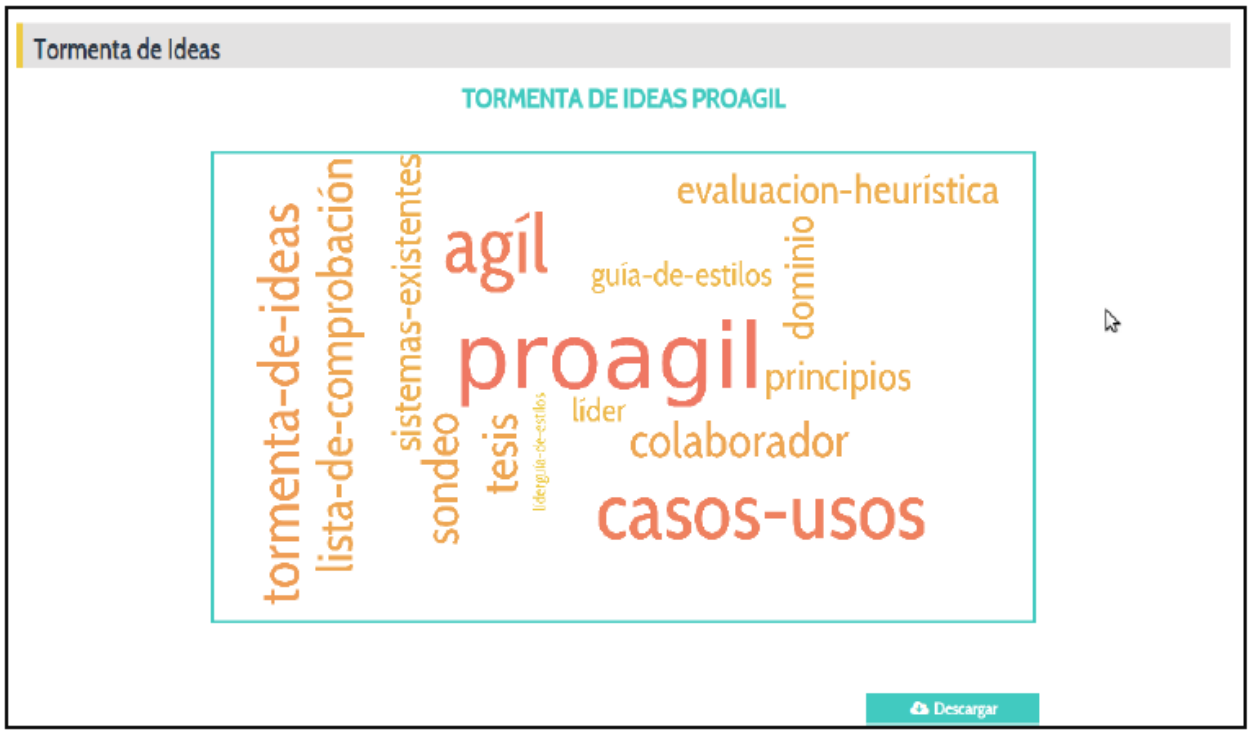


Figura 1.1: Tormenta de ideas

- Análisis de sistemas existentes: consiste en la revisión de versiones anteriores del mismo sistema, así como sistemas de la competencia o afines, con el objetivo de identificar problemas de usabilidad. En la figura 1.2 se observa la interfaz de usuario para la creación del análisis de un sistema existente.

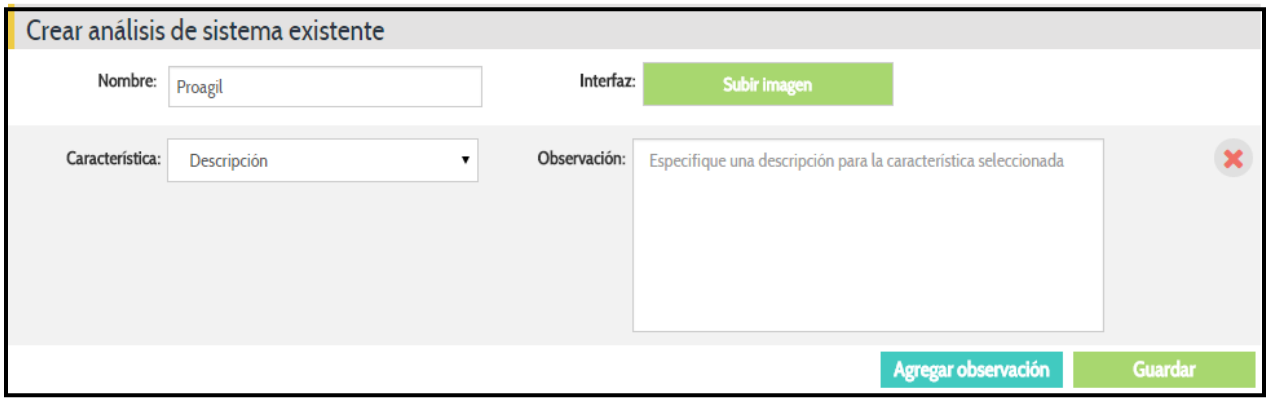


Figura 1.2: Análisis de sistemas existentes

- Sondeos: son una serie de preguntas que se realizan al usuario con el objetivo de determinar los requerimientos funcionales, no funcionales y aspectos de la interfaz de usuario de algún sistema que está por iniciar. Un ejemplo se muestra en la figura 1.3.

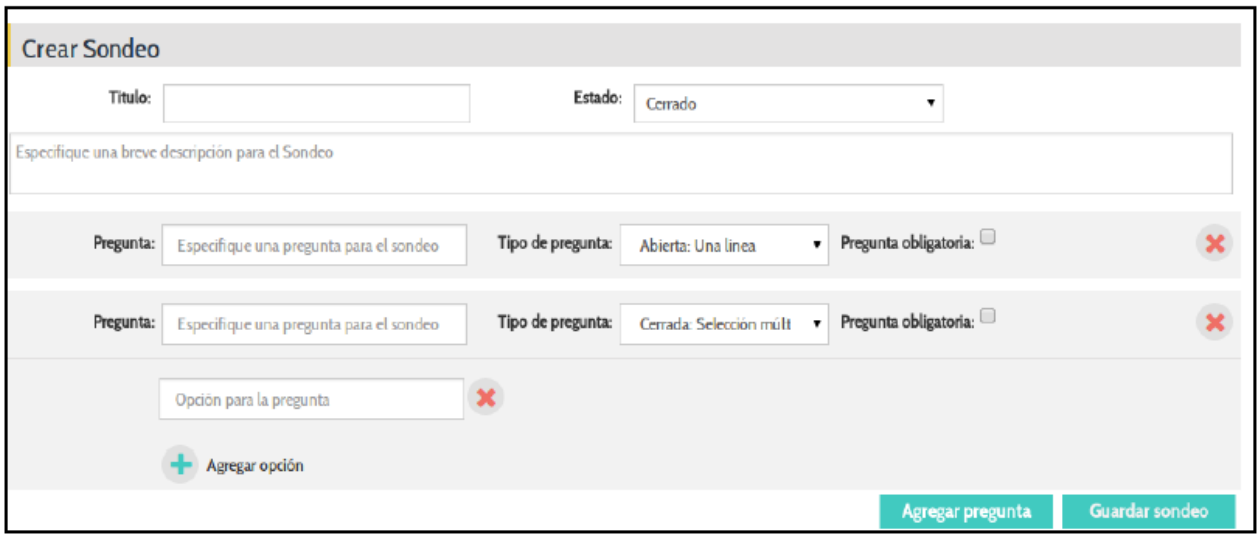


Figura 1.3: Sondeo

1.1.2 Módulo de inspección

Los artefactos de inspección se asocian a evaluaciones de proyectos en proceso o terminados. De manera intuitiva los artefactos de inspección permiten buscar errores o inconsistencias en las interfaces de usuario en una forma detallada.

Los métodos de inspección permiten determinar la madurez conceptual del producto que se está construyendo (Acosta, A., 2013)

Los artefactos automatizados en ProAgil son:

- Evaluación Heurística: es un artefacto de inspección de usabilidad basado en la detección de problemas de usabilidad en interfaces de usuario. Para su aplicación, un grupo de inspectores con conocimiento en aspectos de usabilidad evalúan la interfaz de usuario teniendo en cuenta principios o heurísticas comúnmente aceptadas. En la figura 1.4 se observa la interfaz de usuario que permite realizar la evaluación heurística.

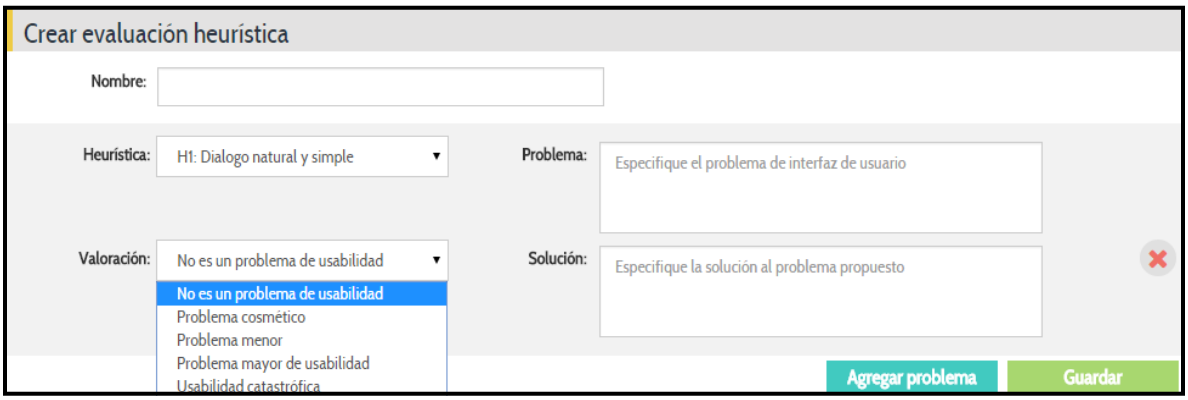


Figura 1.4: Evaluación heurística

- Lista de comprobación: este artefacto está conformado por recomendaciones que el equipo de desarrollo ha acordado considerar en el diseño de la interfaz de usuario. Para utilizarlo, hay que comenzar decidiendo qué tipo de listas se van a utilizar para juzgar los atributos y los métodos de interacción de la interfaz de usuario tal como se puede apreciar en la figura 1.5.

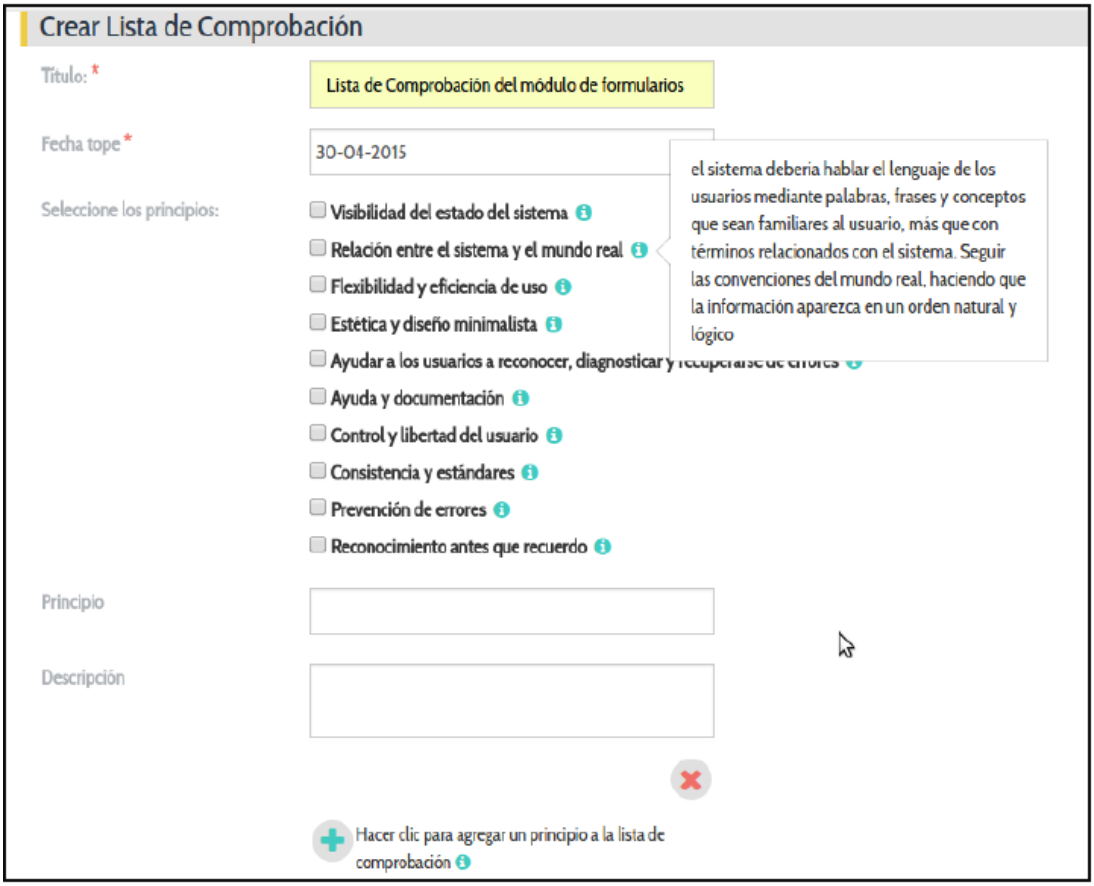


Figura 1.5: Lista de comprobación

- Guías de estilo: una guía de estilo es un documento que recopila directrices relacionadas con el aspecto de la interfaz de usuario que ofrece una aplicación, tales como: logotipo, la gama de colores a utilizar en la interfaz de usuario, fuentes, distribución de los elementos en los diálogos y aspectos de los estilos de interacción (menú, botones, enlaces,. . .). En la figura 1.6 se visualiza la interfaz de usuario para crear una guía de estilo.

Figura 1.6: Guía de estilos

ProAgil se desarrolló bajo el enfoque de la metodología ágil AgilUs la cual permite estructurar y aplicar evaluaciones de usabilidad en todas las etapas del desarrollo del software que se presenta a continuación.

1.2 Metodología AgilUs

La metodología AgilUs es el resultado de una de las líneas de investigación desarrolladas en el Centro de Ingeniería de Software y Sistemas (ISYS) de la Escuela de Computación, Universidad Central de Venezuela (Acosta, A., 2013). Propone incluir la usabilidad como un aspecto fundamental durante el ciclo de desarrollo de software. Para esto, es necesario que los usuarios finales sean incluidos en el desarrollo del proyecto. Se fundamenta en el análisis centrado en el usuario y en la participación de especialistas, con el objetivo de evolucionar el software, a fin que éste alcance el mayor grado de usabilidad una vez culminado su desarrollo. AgilUs es una metodología de desarrollo iterativa e incremental incluyendo artefactos y técnicas que ayudan a desarrollar software usable. Se centra en que la construcción y desarrollo de las interfaces de usuario no debe ser una adición estética que se da al final del desarrollo del sistema sino, muy por el

contrario, el desarrollo de interfaces de usuario debe guiar las decisiones en Ingeniería de Software. (Acosta, A., 2013).

Los principios propuestos por la metodología AgilUs son:

- Integrar la Interacción Humano-Computador (IHC) y la Ingeniería de Software (ISW).
- Considerar la usabilidad desde el principio del desarrollo para impactar positivamente la calidad del producto final.
- La usabilidad determina la utilidad. Mientras más usable sea un software, su utilidad va a ser mayor.
- El usuario determina la usabilidad. Un software sólo será considerado usable en un contexto específico bajo cierto perfil de usuario específico.

1.2.1 Etapas

AgilUs propone realizar cuatro etapas para el desarrollo de software, en cada una se aplican técnicas y artefactos de IHC (Interacción Humano-Computador) e ISW (Ingeniería de Software). Se comienza por definir los requisitos del usuario, saber qué es lo que se desea desarrollar y el tipo de audiencia a la que va dirigida la aplicación. Luego, se hace un análisis de todos los requisitos y se le da formalidad utilizando diagramas y prototipos de baja fidelidad para pasar a la etapa de prototipaje en donde se evoluciona el prototipo inicial. Finalmente, se hace la entrega que permite la aceptación por parte de los usuarios. A continuación se describe las etapas de AgilUs y los artefactos generados en cada una de ellas (Acosta, A., 2013). De manera general se puede ver en la Figura 1.7.

- **Requisitos:** Se realiza el análisis global del problema a solucionar, se estudian productos similares existes, se genera un perfil de usuario, y se define la lista de requerimientos funcionales y no funcionales a desarrollar.

    - Tormenta de ideas
    - Evaluación de sistemas existentes

- Perfiles de usuario
- Requerimientos funcionales y no funcionales

- **Análisis:** Se lleva a cabo el análisis de la solución a desarrollar, se emplean diagramas de casos de uso y modelo de objetos del dominio, siguiendo la notación de Lenguaje Unificado de Modelado (UML), para definir las funcionalidades que tendrá el producto a desarrollar.

    - Modelo de Casos de uso
    - Prototipo en papel
    - Modelo de Objetos del dominio
    - Guía de estilo
    - Patrones de interacción

- **Prototipaje:** Se implementa un prototipo rápido de la interfaz de usuario a partir de los patrones de interacción, el cual va evolucionando hasta convertirse en el producto final, se genera la guía de estilos, y se realizan evaluaciones de usabilidad apropiadas a esta etapa: las evaluaciones heurísticas y las listas de comprobación.

    - Evaluación Heurística
    - Lista de comprobación
    - Prototipo ejecutable

- **Entrega:** Se aplican las pruebas al sistema para certificar que la aplicación desarrollada sea un software usable y sin errores, finalmente se pone en producción la aplicación.

    - Protocolo de preguntas
    - Pruebas de aceptación
    - Aplicación a liberar

Figura 1.7: Etapas de AgilUs

Tanto en AgilUs como en otros métodos de desarrollo se propone crear diagramas o bocetos que permitan formalizar a través de una notación las funcionalidades y entidades que estarán presentes en un software, por lo general esto se aplica en el análisis y diseño del sistema. Hoy en día, una de las notaciones más conocidas es UML[1]. Dicha notación se convirtió en un estándar para poder representar de forma gráfica los requerimientos de un sistema, además de poder comunicarse y aportar ideas dentro del equipo de desarrollo. UML ofrece distintos diagramas que puedan especificar, documentar y representar un sistema. Entre los más utilizados se encuentran el modelo de casos de uso y el modelo de objetos de dominio, descritos a continuación.

1.3 Modelo de casos de uso

El modelo de casos de uso (MCU) es usado para definir los posibles escenarios que se pueden ejecutar en un sistema. Las interacciones entre el usuario y el sistema son definidas a través de secuencias de acciones que describen el comportamiento del

[1] Lenguaje de Modelado Unificado (*Unified Modeling Language)* es un lenguaje estándar, basado en una notación gráfica, que se utiliza para modelar software (OMG (Object Management Group), 2015)

sistema. En el MCU se modelan los requerimientos funcionales determinados en la etapa de requisitos. En la figura 1.8 se muestra un ejemplo del modelo. Se representa a través de actores los cuales interactúan con uno o más Caso de Uso, esto se hace con el uso de asociaciones.

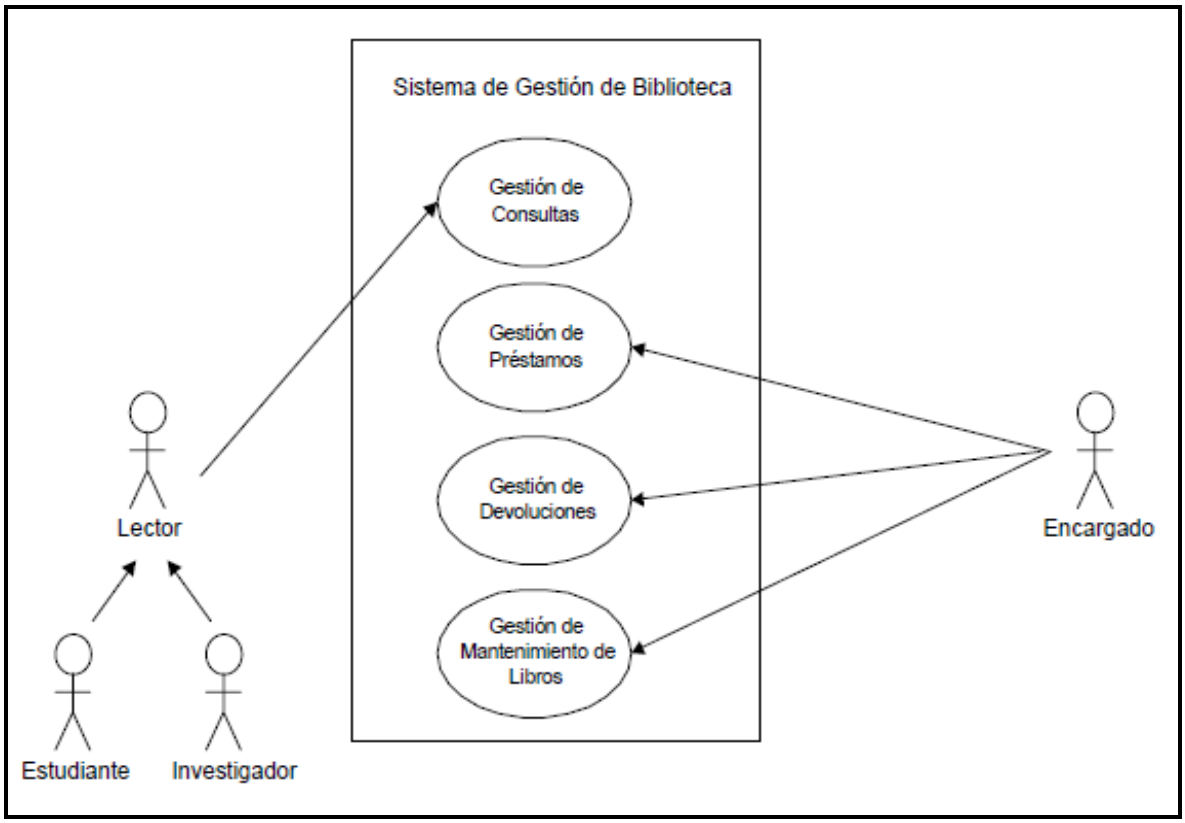


Figura 1.8: Diagrama de casos de uso

La principal ventaja que tiene el modelo de casos de uso es la facilidad para interpretarlos, lo que hace que sean especialmente útiles en la comunicación con el cliente. Además permite al grupo de desarrollo saber cuáles son las funcionalidades del sistema y los actores que intervienen en el mismo.

1.4 Modelo de objetos de dominio

El modelo de objetos del dominio (ODD) es una representación gráfica e intuitiva del sistema, es útil para determinar cuáles objetos van a tener alguna representación de la interfaz de usuario. Tiene como idea central conceptualizar el dominio del problema en términos de objetos que interactúan, se modifican y

responden a acciones, construyendo un modelo que simula el comportamiento del mundo real. En la figura 1.9 se muestra un ejemplo del modelo.

Utilizando la notación UML, un modelo del dominio se representa con un conjunto de diagramas de clases en los que no se define ninguna operación. Pueden mostrar:

- Objetos del dominio o clases conceptuales.
- Asociaciones entre las clases conceptuales.
- Atributos de las clases conceptuales.

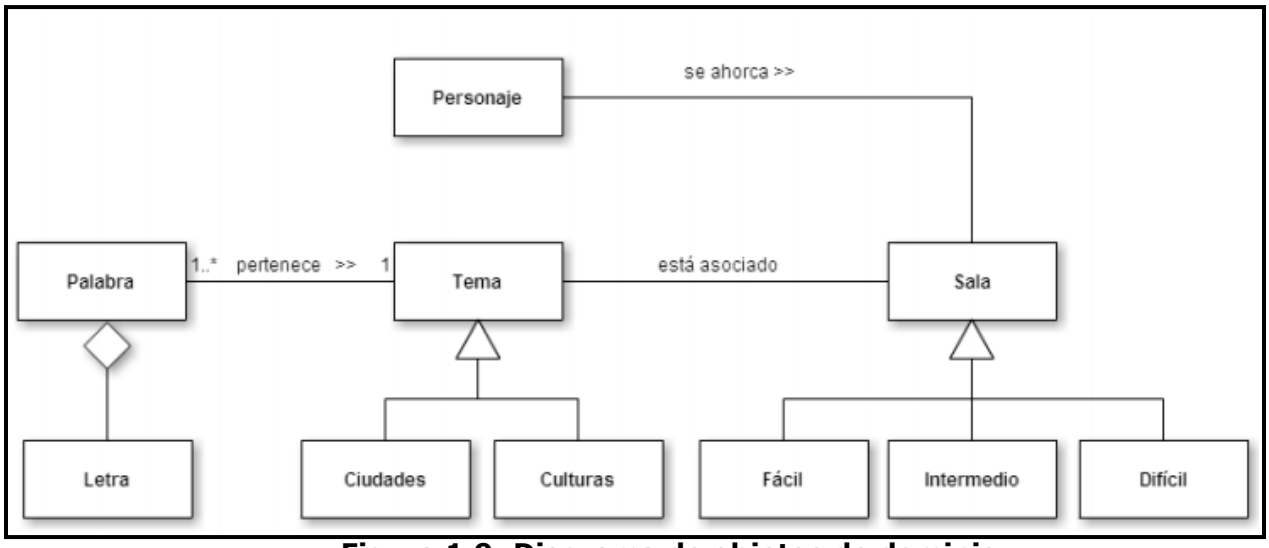


Figura 1.9: Diagrama de objetos de dominio

Estos diagramas van a describir los objetos del sistema y su relación entre ellos permitiendo así delimitar el sistema para así verificar y validar la comprensión del dominio del problema, además contribuye a analizar el modelo de datos.

El modelado de software va a mejorar la productividad del equipo de desarrollo porque va a facilitar la compresión del sistema, además de modular y facilitar el mantenimiento del software. Otra ventaja es que por la facilidad de comprensión de los diagramas, se consigue una mejor comunicación tanto en el equipo de trabajo como con el cliente promoviendo el trabajo en equipo.

1.5 Prototipo de interfaz de usuario

Además del modelado, en la fase de análisis de un sistema también se realiza el diseño de la interfaz, esto se logra a través de un prototipo de interfaz de usuario que nos permita visualizar como estarían dispuestos los elementos que interactuarían con el usuario. Es una forma de crear una imagen palpable de lo que será una aplicación. Es un prototipo de baja fidelidad cuya elaboración no toma en cuenta aspectos técnicos (plataforma de desarrollo) o aspectos gráficos (colores y tipos de fuente). Sirven como una primera aproximación a la maquetación final de las interfaces de usuario sin necesidad de ser tan formales. Pueden ser realizados a mano o en cualquier herramienta que permita estructurar las secciones de la interfaz de usuario. La figura 1.10 muestra un ejemplo de un prototipo de interfaz de usuario para una aplicación móvil.

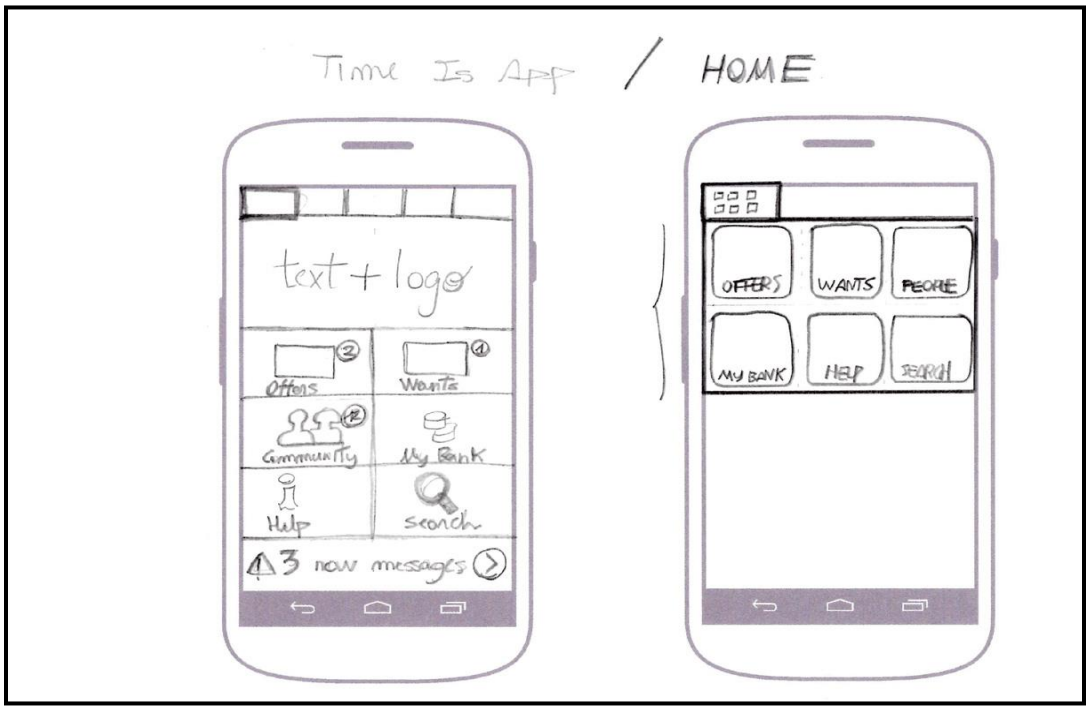


Figura 1.10: Prototipo de interfaz de usuario

El principal propósito del prototipo de interfaz de usuario es obtener y validar los requerimientos esenciales y obtener una retroalimentación del usuario que nos permita mejorar el diseño de interfaz, asimismo ahorra tiempo y dinero, ya que

permite a los desarrolladores probar las interfaces de productos antes de escribir código o comenzar el desarrollo. Esto también permite la modificación fácil y de bajo costo a los diseños existentes, que hace de esta técnica útil en las primeras fases de diseño.

Es por ello que el aporte de esta investigación es la incorporación de los artefactos: diagrama de casos de uso, diagrama de objetos de dominio y prototipo de interfaz de usuario a la aplicación ProAgil.

En el siguiente capítulo se procede a detallar las iteraciones realizadas para el desarrollo de la aplicación así como los artefactos generados al aplicar la metodología AgilUs.

CAPÍTULO II. Desarrollo del módulo de diagramación y prototipaje

El desarrollo de la aplicación web se llevó a cabo a través de la metodología AgilUs. Como fue descrita en el Capítulo I, AgilUs utiliza un enfoque iterativo e incremental lo que permite que se puedan realizar varias iteraciones en una misma etapa. El proceso de desarrollo de la aplicación se dividió en tres iteraciones: Análisis global; donde se describe el análisis general del sistema para determinar las funcionalidades principales, módulo de diagramación; el cual contiene el análisis y diseño general de los diagramas de casos de uso y diagramas de objetos de dominio, y por último el Módulo de prototipaje; se detalla el análisis, diseño e implementación del prototipo de interfaz de usuario. Es importante precisar que las iteraciones están basadas por objetivos, por lo que cada iteración finaliza con una versión del módulo.

A continuación se describen los artefactos generados en cada fase de la iteración.

2.1 Iteración I: Análisis Global

Se inicia con un análisis del sistema, el cual va a permitir conocer en general, lo que se requiere que haga el software. Se prosigue a explicar los artefactos generados en cada fase.

2.1.1 Requisitos

En esta fase se identifican los requerimientos funcionales y no funcionales del sistema, se genera el perfil de usuario y se hace un análisis general de los requisitos del sistema.

- **Tormenta de ideas:** Se realiza una reunión donde se intercambian ideas entre el grupo de desarrollo para definir las características que requiere el sistema. En la figura 2.1 se muestra un resumen de las propuestas sugeridas.

Figura 2.1: Tormenta de ideas

Entre las más destacadas se encuentran:

a. Drag: Es un estilo de interacción que permite arrastrar los elementos dentro del campo de trabajo, por lo que el usuario podrá colocar los elementos que conforman los diagramas o el prototipo de interfaz de usuario en el lugar que desee.
b. Librerías: Se propone utilizar librerías que permitan añadir funcionalidades a los diagramas y al prototipo de interfaz de usuario
c. Diagrama: El sistema incorpora dos tipos de diagramas: diagramas de casos de uso y diagramas de objetos de dominio
d. Prototipo: Son los diseños de interfaz de usuario

- **Perfil de usuario:** El usuario de ProAgil es una persona con conocimiento en el área de ingeniería de software y de tecnologías web que este familiarizado con artefactos y técnicas que se aplican en el desarrollo de software ágil.

- **Requerimientos funcionales:**

Los requerimientos funcionales permiten determinar las áreas funcionales y objetivos que cumplirá el sistema. Entre los requerimientos funcionales definidos se encuentran:

  - Crear diagramas
  - Crear prototipos
  - Eliminar diagramas
  - Eliminar prototipos
  - Exportar diagramas y prototipos
  - Cambiar el tamaño de los elementos

- **Requerimientos no funcionales:**

Se toma en cuenta la usabilidad como aspecto intrínseco o cualidad que debe cumplir el sistema refiriéndose a la satisfacción del usuario al utilizarlo, tomando en cuenta metáforas y textos que faciliten su interacción con el software.

2.1.2 Análisis

Continuando con el ciclo de desarrollo de AgilUs, la segunda etapa es Análisis, en la cual se hace un estudio de la solución a desarrollar. En esta etapa se genera la guía de estilo, el diagrama de casos de uso y el diagrama de objetos de dominio.

- **Guía de estilos**

La guía de estilos es un documento con una serie de lineamientos relacionados al aspecto de la interfaz de usuario. Como se observa en la figura 2.2 se encuentra el logo del sistema, la paleta de colores, el tipo de tipografía y los iconos utilizados.

Logo

PROAGIL

Paleta de colores

#A8D76F #F8D347 #EF6F66 #41CAC0

E3E1E2 #F2F2F2 #FFFFFF

#2A3541 #36404C

Tipografía

Cabin Condensed **Cabin Condensed**

Normal Negrita

En tamaño 20px, 16px y 12px

Iconografía

Íconos de *glyphicons* incluidos en Bootstrap e iconos de *font awesome*

Metáforas para los artefactos de evaluación de usabilidad

Figura 2.2: Guía de estilo

- **Diagrama de Casos de Uso**

Se diseña el modelo de casos de uso que permite ver la interacción existente entre el usuario y los requerimientos del sistema. En la figura 2.3 se observa el diagrama de los casos de usos generados.

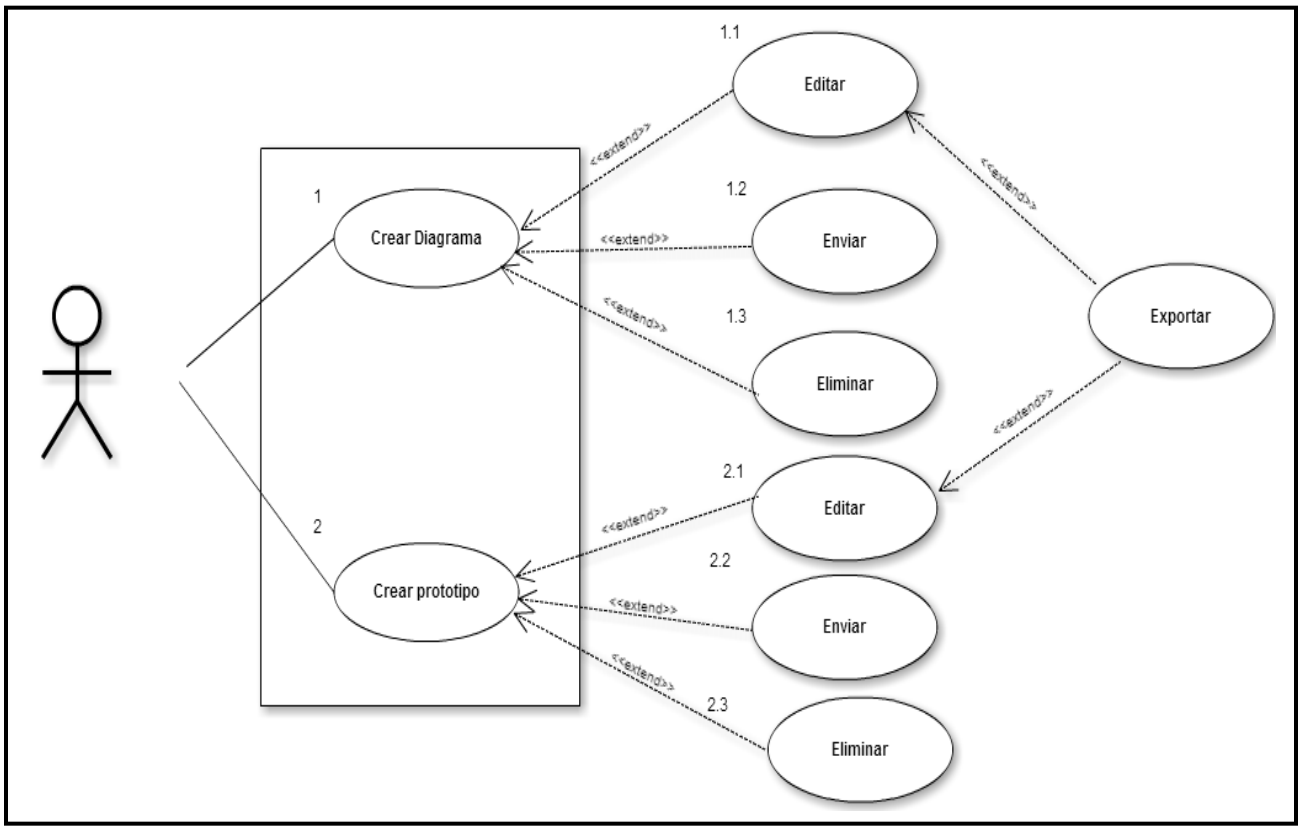


Figura 2.3: Nivel 1 del Diagrama de casos de uso

Se define como actor principal al usuario, que estaría representando a los estudiantes que deseen utilizar la plataforma para realizar los diagramas y el prototipo. Entre los casos de uso que el usuario interactúa se encuentran:

1. Crear diagrama: Permite al usuario crear los diagramas que desee, proporcionando un nombre al diagrama. Los diagramas pueden ser de dos tipos: Diagramas de casos de uso y Diagramas de objetos de

dominio. En la extensión del caso de uso se encuentran: Editar (CU 1.1), permite al usuario realizar distintas acciones como agregar, eliminar entidades del diagrama así como guardarlo, además se puede exportar el diagrama en formato PNG; enviar (CU 1.3), el usuario puede enviar el diagrama a los integrantes del equipo mediante un correo y eliminar (CU 1.3) en el cual se suprime.

2. Crear prototipo: En este caso de uso el usuario podrá crear prototipos. Al igual que en los diagramas, se extiende dos casos de uso: Editar (CU 2.1), en el cual se puede borrar, añadir elementos al prototipo, además de guardarlo y poder exportarlo en formato PNG; enviar (CU 2.2), a través de un correo electrónico el usuario envía el prototipo al equipo de desarrollo y eliminar (CU 2.3), donde el usuario elimina todo el prototipo.

- **Diagrama de Objetos de dominio**

En el modelo de objetos de dominio se definen los objetos del sistema y las relaciones entre ellos para de esta manera determinar el dominio del sistema. En la figura 2.4 se observa los objetos de la aplicación, entre ellos el usuario, diagrama; los cuales se dividen en diagramas de casos de uso y diagrama de objetos de dominio, prototipo y las entidades.

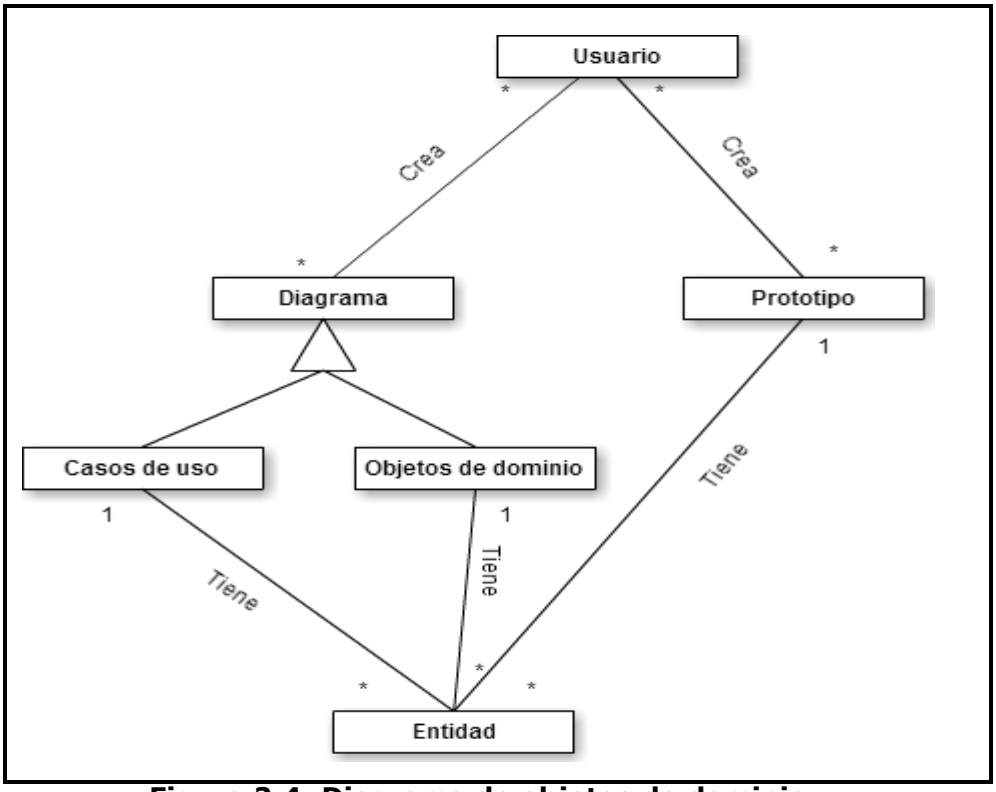


Figura 2.4: Diagrama de objetos de dominio

2.2 Iteración II: Módulo de diagramación

En la segunda iteración se presenta el análisis del módulo de diagramación, el cual está dividido en diagrama de casos de uso y diagramas de objetos de dominio. El objetivo es analizar sistemas existentes que ofrezcan funcionalidades o características que la aplicación pueda aplicar. Además se realiza un primer diseño de la interfaz para luego aplicar evaluaciones de usabilidad que permitan mejorar el la aplicación. Los artefactos que surgieron en cada fase son.

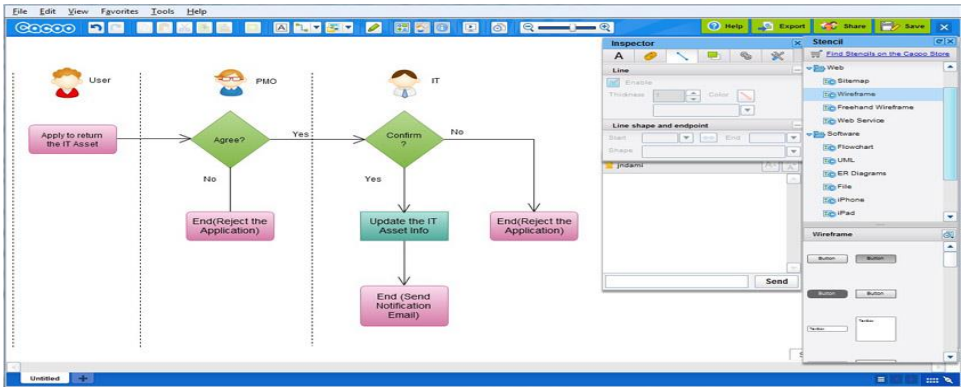
2.2.1 Requisitos

En la presente etapa se realiza la evaluación de sistemas existentes con el propósito de analizar funcionalidades de sistemas que se puedan incorporar a la aplicación. A continuación se explican.

- **Evaluación de sistemas existentes**

La finalidad de la evaluación de sistemas existentes es poder encontrar aspectos positivos o características que puedan ser usadas en el desarrollo del sistema así como aspectos negativos para prevenir incorporarlas en la construcción del software. En esta investigación estudiamos diversos sistemas pero tomamos en cuenta la evaluación de Cacoo porque es una herramienta que nos aporta mayores beneficios para nuestro módulo. La evaluación se puede apreciar en la siguiente tabla 1.1.

Cacoo https://cacoo.com



Tópico a evaluar	Observaciones
Descripción	Cacoo es una herramienta de dibujo en línea que permite a los usuarios crear una variedad de diagramas tales como mapas, prototipos, diagramas UML y de red
Funcionalidades	Permite crear distintos tipos de diagrama, personalizar cada elemento como su tamaño, color, tipo de fuente. También se puede exportar en PDF o en formato JPG y compartirlo en distintas redes sociales. Además es colaborativo por lo que los varios usuarios pueden modificar un mismo diagrama en tiempo real
Idioma	Español e inglés
Opinión como usuario	La herramienta es bastante intuitiva, utiliza un buen uso de los colores. A pesar de que ofrece gran variedad de botones no recarga la vista del usuario. Fácil de usar y bastante eficiente.

Tabla 1.1: Evaluación de sistemas existentes Cacoo

La evaluación de *Cacoo* permite tomar en cuenta características que se desean incorporar a los módulos. Se toma en cuenta la disposición de los elementos; en la barra superior se encuentran herramientas de edición, al igual que en la barra derecha se observan los elementos de cada diagrama categorizados por su tipo. Otro aspecto que se considera son las metáforas como las de rehacer y deshacer, exportar y aumento/reducción de la pantalla, lo que permite que el usuario reconozca en vez de recordar.

2.2.2 Análisis

En esta etapa se muestra una primera versión de la interfaz de usuario tanto para el diagrama de casos de uso como el diagrama de objetos de dominio.

- **Prototipo en papel**

El propósito del prototipo en papel es diseñar un prototipo de interfaz de baja fidelidad con el objetivo de encontrar problemas y de esta manera poder refinar el diseño en la etapa de desarrollo. En la figura 2.5 se presenta una primera versión de la interfaz para el diagrama de casos de uso. En la figura 2.6 se observa el prototipo de interfaz de usuario para el diagrama de objetos de dominio, la cual presenta la misma estructura que la figura anterior, manteniendo la consistencia entre las interfaces.

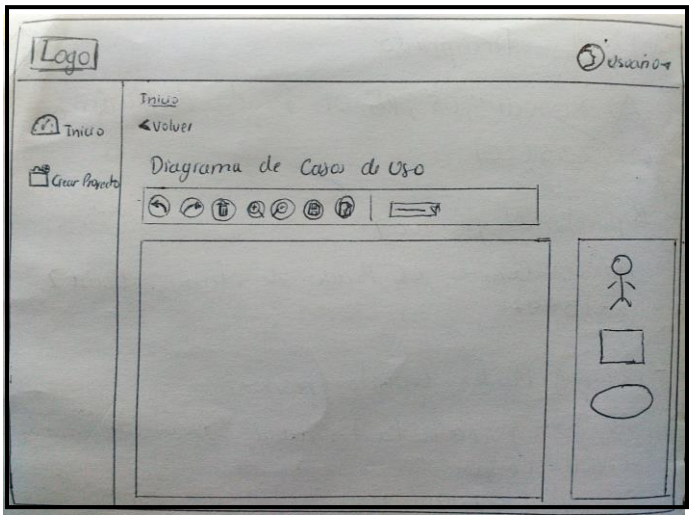


Figura 2.5: Prototipo en papel del diagrama de casos de uso

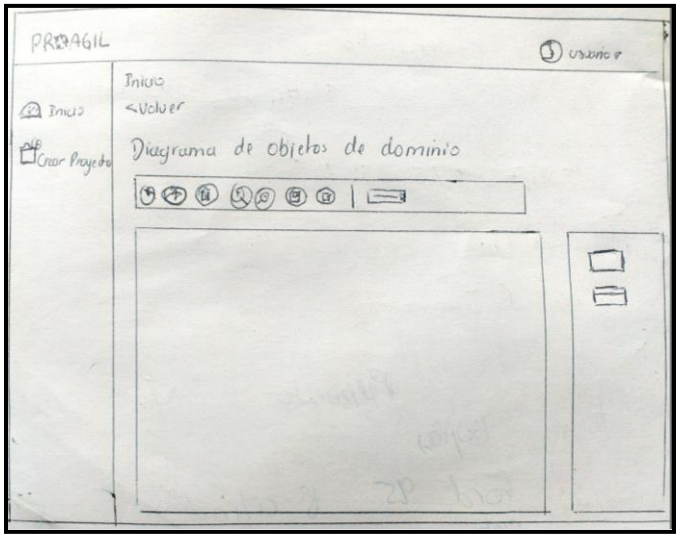


Figura 2.6: Prototipo en papel del Diagrama de objetos de dominio

2.2.3 Prototipaje

La presente etapa consiste en el desarrollo de un prototipo ejecutable. Se describe las tecnologías y herramientas utilizadas para la construcción de la aplicación ProAgil, además de mostrarse las interfaces de usuario más importantes del módulo. Para finalizar se presentan las evaluaciones heurísticas correspondientes a las funcionalidades implementadas.

- **Tecnologías y herramientas web**

Para la implementación de los módulos de diagramación y prototipaje se utilizaron diferentes tecnologías y herramientas web. A continuación se describirán brevemente.

Las tecnologías web se dividen en dos categorías: Las tecnologías del lado del cliente; son las que no dependen del servidor, por lo que las páginas creadas son interpretadas por el navegador y mostradas al usuario, y las tecnologías del lado del servidor; que son ejecutados e interpretados por el propio servidor y que se envían al cliente en un formato comprensible para éste.

2.3 Tecnologías del lado del cliente

Dentro de las tecnologías del lado del cliente se utilizaron:

2.3.1 HTML5

HTML es el lenguaje de marcado (HyperText Markup Language por sus siglas en inglés) utilizado para la creación de páginas web. (W3C, 2013). La sintaxis de HTML está basada en una serie de etiquetas. Un etiqueta está compuesta por caracteres especiales:<,> y /. Estos son interpretados como elementos que van a representar el contenido de la página web. Usualmente las etiquetas vienen en pares, por lo que hay una etiqueta de inicio y una etiqueta de cierre. Además las etiquetas pueden contener atributos las cuales proveen información adicional al elemento.

2.3.2 CSS

Hojas de estilo en cascada (Cascading Style Sheets por sus siglas en inglés) es un lenguaje usado para definir el estilo de un documento escrito en HTML separando así el contenido de la presentación. (W3C España, s.f)CSS funciona a base de reglas, es decir, declaraciones sobre el estilo de uno o más elementos. Las

hojas de estilo están compuestas por una o más de esas reglas aplicadas a un documento HTML o XML. La regla tiene dos partes: un selector y la declaración. A su vez la declaración está compuesta por una propiedad y el valor que se le asigne.

2.3.3 Javascript

Javascript es un lenguaje de programación interpretado que incorpora dinamismo del lado del cliente a través de la manipulación del DOM o del browser. Es el responsable del comportamiento de la página. (Notas Docencia, 2012)

2.3.4 JQuery

jQuery es una librería de JavaScript que permite modificar documentos HTML. El propósito de jQuery es hacer el uso de javascript mucho más fácil (Notas docencia, 2012). jQuery toma una gran cantidad de tareas comunes las cuales requieren muchas líneas de código en javascript para su realización y las introduce en métodos que puedes llamar con una simple línea de código

2.3.5 JointJs

JointJs es una librería hecha en HTML5 y Javascript para la visualización e interacción de diagramas y gráficos. Puede usarse para crear diagramas estáticos o herramientas interactivas de diagramación (client IO, s.f.). Entre sus características se encuentran:

- Trabaja bajo la arquitectura MVC
- Provee elementos básicos de diagramas (rectángulo, circulo, elipse, texto, imágenes)
- Permite personalizar los elementos
- Permite conectar elementos a través de links

2.4 Tecnologías del lado del servidor

En el lado del servidor, se utilizó la siguiente tecnología:

2.4.1 PHP

Preprocesador de Hipertexto (por sus siglas en inglés *HyperText Preprocesor*) es un lenguaje script interpretado de código abierto que se incrusta dentro del HTML, se ejecuta del lado del servidor, está especialmente adecuado para el desarrollo web, y posee una gran cantidad de librería y mucha documentación. El principal objetivo de este lenguaje es permitir rápidamente a los desarrolladores la generación de páginas web dinámica, permitiendo a los programadores trabajar con una diversidad de base de datos sin mayor esfuerzo.

2.5 Otras

Se aplicaron herramientas que nos ofrecieran facilitar la comunicación y organización del equipo de trabajo. Entre ellas se encuentran:

2.5.1 Laravel

Laravel es un framework de PHP desarrollado por Taylor Otwell para aplicaciones web, con sintaxis expresiva y elegante. Tiene como filosofía "el desarrollo debe ser una experiencia agradable y creativa para que sea verdaderamente enriquecedora". Laravel busca eliminar el sufrimiento en el proceso de desarrollo facilitando las tareas comunes utilizadas en la mayoría de los proyectos web, como la autenticación, enrutamiento, sesiones y almacenamiento en caché. Es accesible, pero potente, ofreciendo herramientas poderosas necesarias para aplicaciones de gran envergadura. Este framework utiliza el patrón de arquitectura modelo-vista-controlador (MVC, en inglés *model-view-controller*). Según Bahit

(2011) el patrón MVC es un patrón de arquitectura de software encargado de separar la lógica de negocio de la interfaz del usuario y es el más utilizado en aplicaciones web, ya que facilita la funcionalidad, mantenibilidad y escalabilidad del sistema, de forma simple y sencilla.

2.5.2 GIT

Git es un sistema para el control de versión es distribuido diseñado por Linus Torvalds, pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando estas tienen un gran número de archivos de código fuente. Profundizar en el conocimiento de sus características abre a los desarrolladores un nuevo y liberador enfoque para la gestión del código fuente. Git incluye herramientas específicas para navegar y visualizar un historial de desarrollo no lineal. Una presunción fundamental en Git es que un cambio será fusionado mucho más frecuentemente de lo que se escribe originalmente, conforme se pasa entre varios programadores que lo revisan

2.6 Sistemas Manejadores de Bases de Datos

Los servidores de bases de datos se utilizan en todo el mundo en una amplia variedad de aplicaciones, principalmente son utilizados para bases de datos con múltiples usuarios. En la implementación de la aplicación Proagil se utilizó Postgresql

2.6.1 Postgresql

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente. Es el sistema de gestión de bases de datos de código abierto más potente del mercado. (Martinez, R., 2010)

PostgreSQL utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando.

- **Prototipo ejecutable**

Corresponde al desarrollo de las funcionalidades principales del módulo implementadas con las tecnologías descritas en la sección anterior.

- **Crear diagrama de casos de uso**

Corresponde a la interfaz de usuario para la funcionalidad de creación del diagrama de casos de uso. En la figura 2.7 se observa la interfaz donde se le indica al usuario colocar un título al diagrama. Una vez guardado el título se redirecciona a la interfaz de usuario con los elementos para crear el diagrama.

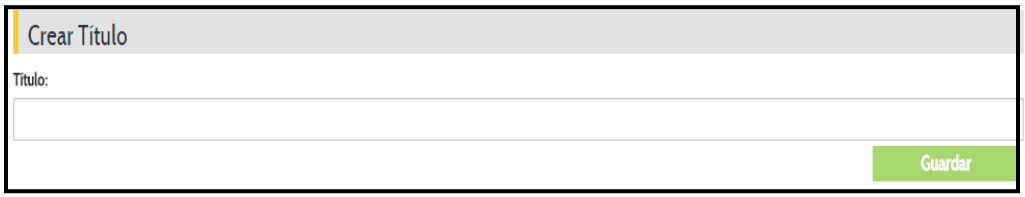


Figura 2.7: Interfaz de usuario para crear titulo

- **Editar diagrama de casos de uso**

En la figura 2.8 se muestra la interfaz perteneciente a la edición del diagrama. La estructura de la interfaz está dividida de la siguiente manera: En la parte superior se muestra el título del diagrama; el cual editable, luego viene la barra de herramientas donde se despliegan las opciones de edición del diagrama. Finalmente la parte inferior está dividida en dos secciones; primero el lienzo donde los elementos podrán ser arrastrados y editados; y una plantilla donde estan dispuestos los elementos del diagrama.

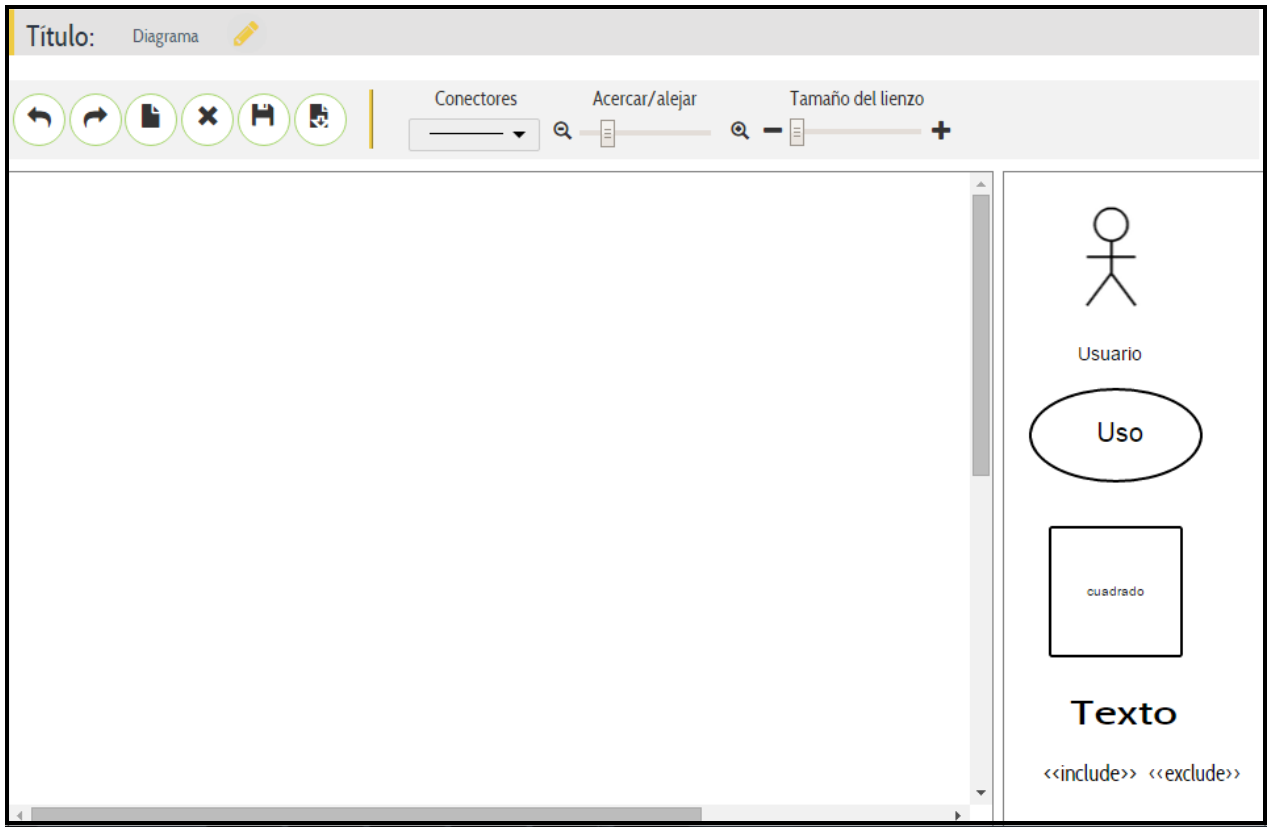


Figura 2.8: Interfaz de usuario para editar diagrama

- **Crear título del diagrama de objetos de dominio**

En la figura 2.9 de aprecia la interfaz de usuario para crear el título del diagrama de objetos de dominio.

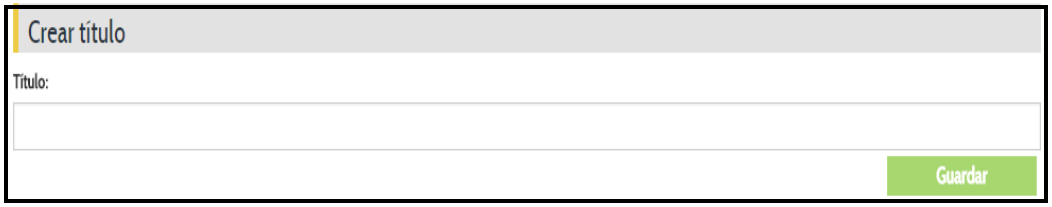


Figura 2.9: Interfaz de usuario para crear título del diagrama

- **Editar diagrama de objetos de dominio**

Al igual que la estructura para la creación del diagrama de casos de uso (ver figura 2.8), la estructura para la edición del diagrama de objetos de

dominio se mantiene igual, con la finalidad de que haya consistencia entre las interfaces. La principal diferencia son los elementos dispuestos en la plantilla (ver figura 2.10).

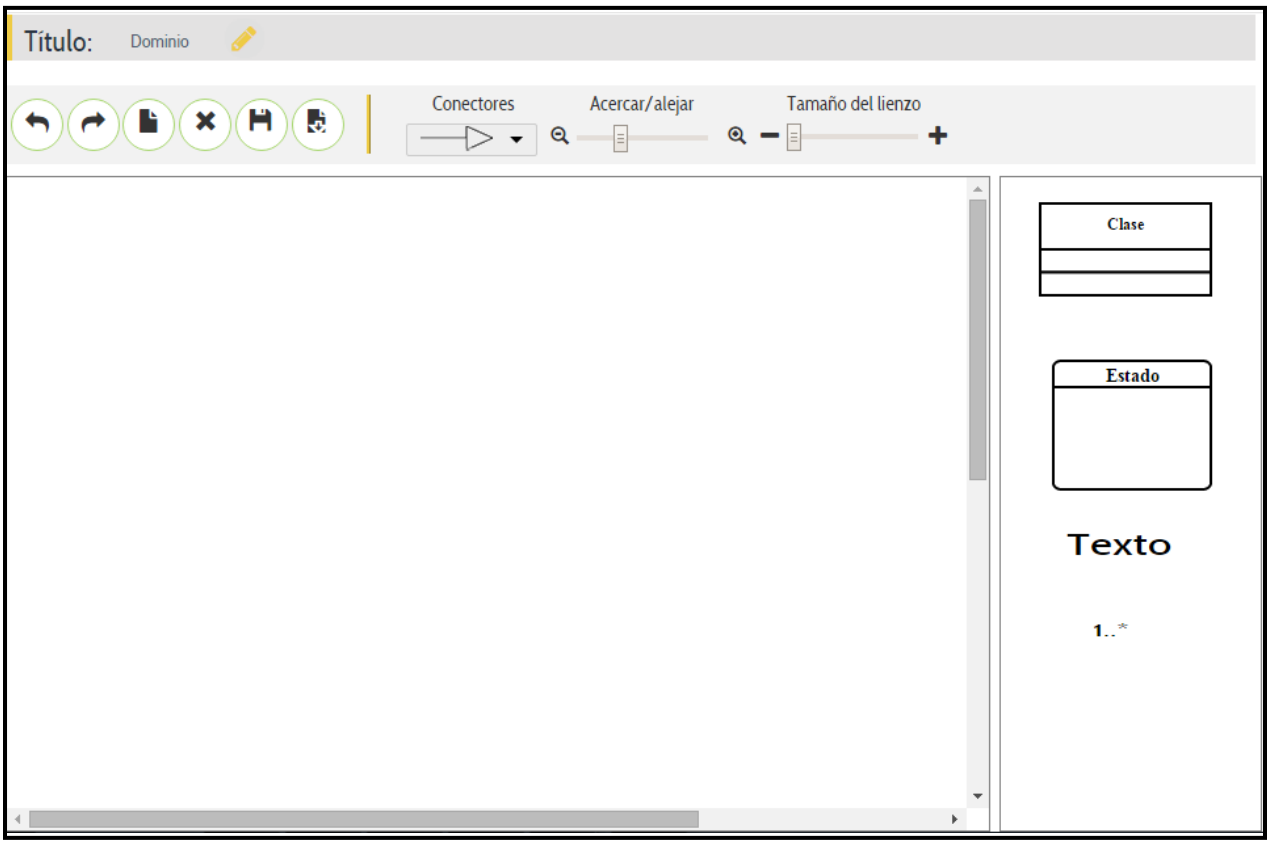


Figura 2.10: Edición diagrama de objetos de dominio

- **Evaluación heurística**

Es una técnica que permite encontrar problemas de usabilidad en la interfaz de usuario, para ello especialistas evalúan cada uno de los elementos de la interfaz tomando en cuenta una lista de principios o heurísticas basadas en las propuestas por Nielsen. En la figura 2.11 se detallan los problemas encontrados por dos especialistas en el área de Ingeniería del software, en el recorrido por los módulos.

Problema	Heurísitca	Valoración	Solución
En los artefactos diagramas de caso de uso y objeto de dominio, cuando el usuario se encuentra en los listados de cada uno respectivamente, el título es "Diagramas"	H3: Minimizar la carga cognitiva	Problema cosmético	El título debería ser "Diagramas de Casos de uso" o "Diagramas de Objeto de dominio", para que el usuario sepa en donde se encuentra específicamente
Cuando el usuario intenta crear un diagrama de caso de uso o objeto de dominio, el título es "Crear diagrama"	H3: Minimizar la carga cognitiva	Problema cosmético	El título debería ser "Crear Diagramas de Casos de uso" o "Crear Diagramas de Objeto de dominio", para que el usuario sepa en donde se encuentra específicamente
Al crear un diagrama de caso de uso, objeto de dominio o un prototipo, el usuario solo tiene la posibilidad de crear el título	H6: Proveer claramente las salidas	Problema mayor de usabilidad	Si la acción es crear un diagrama proveer al usuario la posibilidad de crear el título y el diagrama también, ya que solo puedes crear el título. O una vez creado el título, la siguiente página que se le muestre al usuario sea el editor de diagrama en vez del listado de diagramas o prototipos
En los editores de diagramas y prototipos, no se entiende bien para que sirven los zooms de la lupa y el "-" "+"	H3: Minimizar la carga cognitiva	Problema cosmético	Especificar mejor para que sirven
Al presionar el botón guardar del editor no se muestra un mensaje indicando el estado del sistema	H5: Feedback	Problema cosmético	Debe indicarle al usuario que se guardó el diagrama o el prototipo
El botón "Volver" en las páginas de editar diagrama y prototipo, llevan a la lista de proyectos	H7: Proveer shortcuts	Problema cosmético	Debería llevar a los listado de diagrama y prototipo
Las metáforas de limpiar y eliminar se confunden	H4: Consistencia	Problema cosmético	Utilizar una mejor metáfora para la acción Limpiar puede ser una brochita
En el prototipo se indica la palabra Browser	H2: Hablar el lenguaje del usuario	Problema cosmético	Utilizar preferiblemente la palabra Navegador

Figura 2.11: Evaluación Heurística del módulo de diagramación

Los problemas de la evaluación fueron solucionados tomando en cuenta las sugerencias planteadas por los especialistas. Para los dos primeros problemas se cambiaron los títulos para cada tipo de diagrama. El tercer problema se solucionó brindando una interfaz de usuario para crear el título y luego al guardarlo se redirecciona a la interfaz para crear el diagrama. En el cuarto problema se colocaron títulos que le indiqué al usuario la funcionalidad de cada botón, para el quinto problema una vez guardado el diagrama se le indica la acción al usuario a través de un mensaje en la pantalla. Con respecto al problema seis se modifica el enlace para

que lleve al listado de los diagramas. En cuanto al problema siete y ocho, se cambió la metáfora de eliminar todo el diagrama y la palabra navegador respectivamente.

2.6.2 Entrega

Una vez solucionados los problemas presentados en la fase de prototipaje se presenta una nueva versión del módulo para hacerle las pruebas y de esta manera certificar que el módulo sea usable y satisfaga los requerimientos planteados..

2.7 Iteración III: Módulo de prototipaje

En la última iteración del desarrollo de la aplicación web se encuentra el módulo de prototipaje, en el cual se realiza análisis a sistemas existentes que permitieran poder refinar el sistema tomando en cuenta cualidades de dichos sistemas. Se realiza una primera maqueta de lo que sería el interfaz de usuario de módulo para luego realizar una serie de evaluaciones de usabilidad que permitan refinar la aplicación para su posterior entrega final.

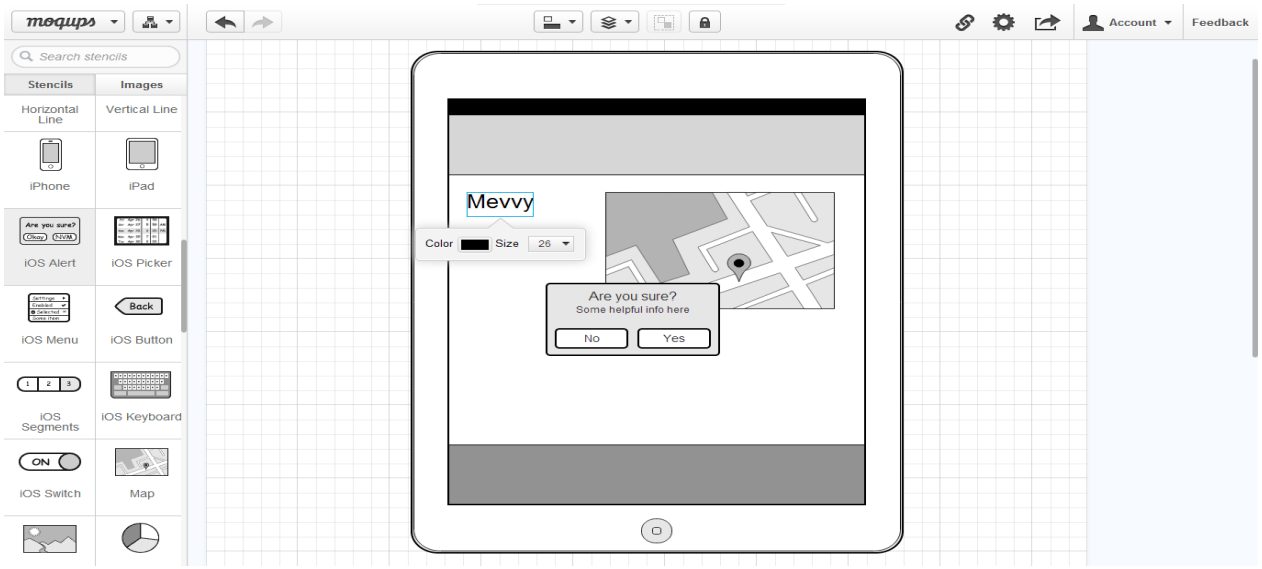
2.7.1 Requisitos

Esta etapa está basada en la evaluación de sistemas existentes con la finalidad de incluir características del sistema evaluado al módulo de prototipaje.

- **Evaluación de sistemas existentes**

En este artefacto se evalúa el sistema Moqups, el cual permite crear maquetas de interfaz de usuario. Una de las características que se toman en cuenta para el módulo de prototipaje es la incorporación de elementos de interfaz tanto para web como para móvil, funcionalidad que provee el sistema evaluado. En la tabla 1.2 se observa la evaluación hecha al sistema Moqups.

Moqups https://moqups.com/



Tópico a evaluar	Observaciones
Descripción	Es una aplicación en HTML5 que permite crear esquemas, maquetas o prototipos de interfaces
Funcionalidades	Permite crear distintos prototipos de interfaz para páginas web, tabletas y teléfonos. Se pueden personalizar cada elemento como su color, tamaño, grosor y el texto. También se pueden cargar imágenes y exportar en formato PDF y PNG.
Idioma	Inglés
Opinión como usuario	Es fácil de usar. No está sobrecargada y ofrece bastantes elementos para usar en el prototipo de interfaz.

Tabla 1.2: Evaluación de sistemas existentes Moqups

2.7.2 Análisis

En la presente etapa se genera el artefacto de prototipo de papel, donde se visualiza un primer boceto de lo que sería la interfaz de usuario para el prototipo

- **Prototipo en papel**

Se muestra la interfaz de usuario asociada al prototipo. Se continúa con el mismo estilo de las interfaces de los diagramas para mantener consistencia entre cada una de ellas. En la figura 2.12 se observa el resultado de la interfaz de usuario.

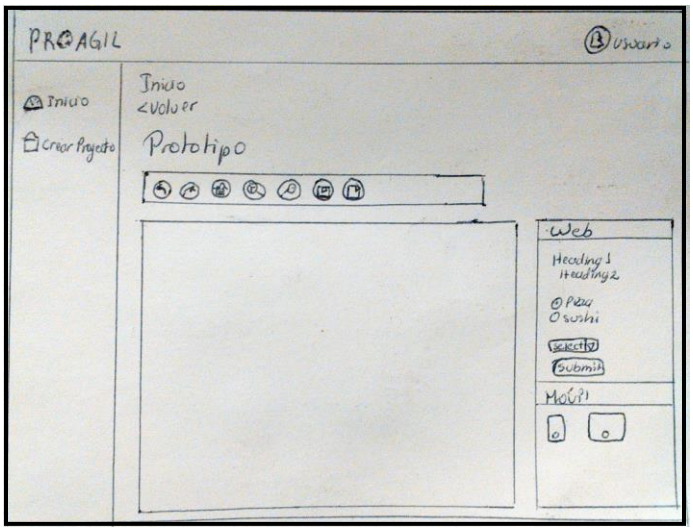


Figura 2.12: Prototipo de interfaz del prototipaje

2.7.3 Prototipaje

En esta etapa se implementa una primera versión del prototipo ejecutable el cual se va refinando, tomando en cuenta la evaluación heurística hasta convertirse en el producto final.

- **Prototipo ejecutable**

Las interfaces de usuario de las funcionalidades más importantes del módulo son:

- **Crear título del prototipo**

Para seguir con la consistencia de las interfaces de usuario se mantiene la creación del título del prototipo (Ver figura 2.13).

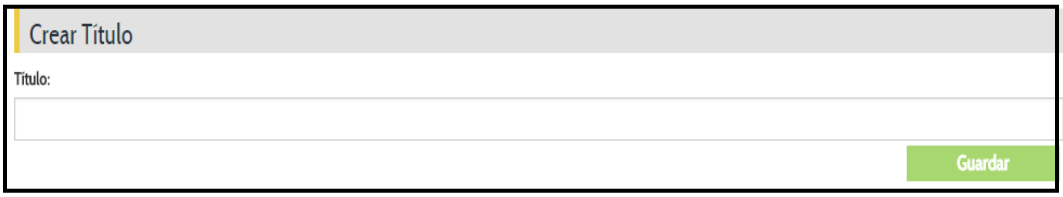


Figura 2.13: Crear título del prototipo

- **Editar diagrama de objetos de dominio**

En la figura 2.14 se aprecia la barra de herramientas en la parte superior y en la parte inferior está dividida del lado izquierdo por el lienzo y del lado derecho por la plantilla con los elementos, la cual a su vez esta categorizada por dos grupos, web y móvil, cada una con subcategorías.

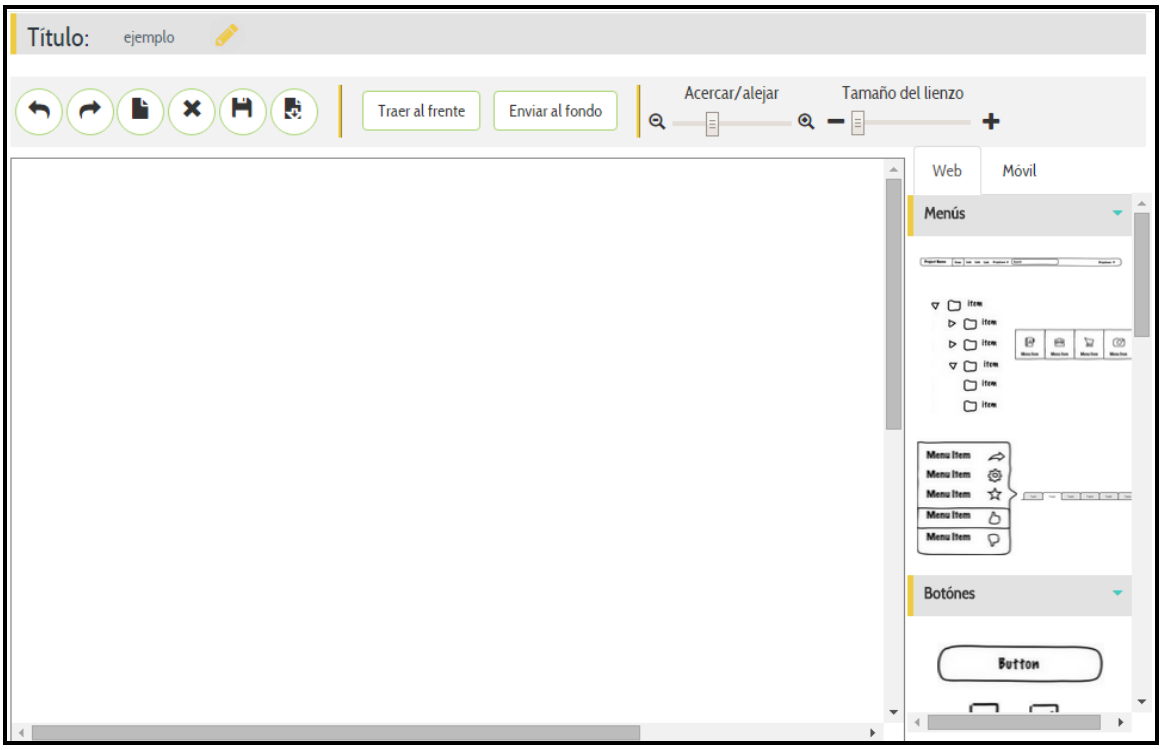


Figura 2.14: Edición del prototipo

- **Evaluación heurística**

Para la evaluación de usabilidad del módulo de prototipaje se cuenta con dos usuarios en el rol de especialistas de Interacción Humano-Computador, los cuales realizaron recorridos por el módulo con el objetivo de evaluar y encontrar errores. En la figura 2.15 se aprecia los problemas encontrados y las soluciones propuestas por ellos.

Problema	Heurísitca	Valoración	Solución
El titulo se muestra en un campo de texto. Se intuye que es para editar el nombre del prototipo pero no existe un botón para guardar en caso de que se modifique	H3: Minimizar la carga cognitiva	Problema mayor de usabilidad	Agregar un botón que permita guardar el título del prototipo en caso que se edite
Para aumentar el tamaño de la zona de trabajo se indica como Tamaño del canvas	H2: Hablar el lenguaje del usuario	Problema cosmético	Indicar una palabra menos técnica, por ejemplo: zona de trabajo, prototipo...
Al querer eliminar un elemento elemento del prototipo hay que hacer clic varias veces en el botón de X	H4: Consistencia	Problema mayor de usabilidad	Eliminar el elemento al hacer clic una vez
Al hacer clic en Nuevo Canvas se borra lo se tenía previamiente	H5: Feedback	Problema mayor de usabilidad	En caso de que se tengan elementos en la zona de trabajo indicarle al usuario con un alerta que se eliminaran los elementos que ha insertado

Figura 2.15: Evaluación Heurística para el módulo de prototipaje

Tomando en cuenta las soluciones planteadas para el primer problema se agrega un botón editar para el título que puede ser guardado o cancelado por el usuario. Con respecto al segundo problema se cambió el título a tamaño de lienzo, utilizando un término más familiar para el usuario. En cuanto al cuarto problema se resolvieron los inconvenientes que no permitían borrar el elemento en el primer clic. Finalmente el cuarto problema se soluciona advirtiendo al usuario a través de una ventana emergente si está seguro de eliminar todo el diagrama

2.7.4 Entrega

En la fase de entrega se realizan pruebas generales, para las 3 iteraciones ya que todas las funcionalidades de los módulos guardan estrecha relación, con la finalidad de verificar si la aplicación cumple con los objetivos planteados.

- **Lista de comprobación**

En éste artefacto se cuenta con una lista de principios, acordados por el equipo de desarrollo, para el diseño de la interfaz de usuario. Se pide a los usuarios manipular la aplicación y luego verificar si cumple con la lista de comprobación. En la figura 2.16 se muestra la lista y si la aplicación llega a cumplir con cada una.

¿Se cumple?	Regla
✔	Usar indicadores visuales:
✔	Utilizar metáforas adecuadas, y en lo posible, las metáforas estándares:
✔	Los controles del sistema deben estar claramente visibles y sus funciones:
✔	Permitir al usuario recuperarse de los errores:
✔	Mantener la interfaz simple, sencilla y organizada:
✔	Evitar palabras coloquiales y abreviaturas:
✔	Consistencia en apariencia y uso:

Figura 2.16: Lista de comprobación para los módulos: diagramación y prototipaje

Con la lista de comprobación se verifica que los módulos de diagramación y prototipaje cumplen con los principios planteados por el equipo de trabajo, concluyendo de manera satisfactoria con los objetivos propuestos.

De esta manera culmina el capítulo, explicando las iteraciones aplicadas para el desarrollo de los módulos de diagramación y prototipaje. A continuación se presentan las conclusiones y recomendaciones para la aplicación desarrollada.

Conclusiones y recomendaciones

Para que el equipo de desarrollo obtenga un software que cumpla de manera satisfactoria los requisitos propuestos desde etapas tempranas de desarrollo es necesario que lleven control sobre los artefactos generados en cada fase, particularmente en metodologías que dentro de sus fases requieran utilizar diagramas y prototipos para la construcción del software. Esto se cumple si se tiene de forma centralizada herramientas que permitan generarlos. Es por ello que el objetivo que se planteó inicialmente en el presente trabajo de grado era "Integración del módulo de diagramación y el de prototipaje en la aplicación ProAgil" el cual se logró satisfactoriamente cumpliendo con una serie de objetivos específicos cuyos resultados se explicarán a continuación.

- Analizar herramientas de diagramación y prototipaje para tomar en cuenta características que sirvan en la plataforma propuesta. En la fase de requisitos se realizó evaluaciones de sistemas como *Cacoo* y *Moqups* que permitieron la elección de cualidades y características que se pudieran incorporan en la aplicación como poder arrastrar los elementos dentro del campo de trabajo además del formato de exportación en PNG.

- Desarrollar un módulo con componentes basados en librerías que permitan realizar diagramas de casos de uso y diagramas de objetos de dominio, así como prototipos. Se utilizó una librería basada en JavaScript que permitió facilitar el desarrollo del módulo de diagramación y prototipaje. En un principio fue difícil adaptar la librería a las características que requería el módulo pero gracias a mecanismos que ofrecen los lenguajes utilizados para la implementación se pudo lograr lo que se planteó al inicio del desarrollo.

- Aplicar el método de desarrollo de software AgilUs. La metodología AgilUs permitió desde etapas tempranas del desarrollo incorporar al usuario lo cual fue de gran ayuda para el refinamiento del software. La comunicación constante y el *feedback* entre el usuario y el equipo contribuyó a establecer y determinar problemas en el sistema, los cuales gracias a la agilidad de la metodología se pudieron resolver y poder mejorar la implementación de la aplicación. La manera en la que está estructurada cada etapa de la metodología permite tener en claro que es lo que se quiere desarrollar y a través de sus actividades y artefactos garantizar la usabilidad del sistema.

Para la implementación de la aplicación se utilizaron lenguajes de programación y herramientas web que permiten la implementación del módulo de diagramación y prototipaje. Se empleó el *framework* Laravel, que aunque no se tenía experiencia con el mismo fue posible su adaptación rápidamente. Permitió mantener una mejor estructura del código debido a la implementación del patrón Modelo Vista Controlador que ofrece. En cuanto a las tecnologías tanto del lado del cliente como el del servidor fueron fáciles de utilizar porque ya eran conocidas por el grupo de desarrollo. El control de versión Git contribuyo en la gestión y control de los cambios hechos en el código fuente.

Para finalizar las recomendaciones propuestas para mejorar y expandir las funcionalidades del sistema desarrollado son:

- Integrar un módulo que muestre los avances del proyecto con el objetivo de mantener informados al grupo de desarrollo sobre el progreso del software.

- Incorporar nuevos elementos dentro del prototipo de interfaz de usuario con la finalidad de adaptarse a los estándares de los dispositivos
- Añadir técnicas o artefactos que permitan mejorar la producción de software como por ejemplo un registro de casos de prueba.

.

Referencias bibliográficas

Acosta, A. (2013). *AgilUs: una metodología ágil de desarrollo de software que incorpora la evaluación de la usabilidad.* client IO. (s.f.). *JointJs*. Obtenido de JointJs: http://www.jointjs.com/

Definicion.de. (s.f.). *plugin*. Obtenido de Definicion.de: http://definicion.de/plugin/

Jacobson, I. (2005). *Casos de Uso.* Obtenido de http://www.ivarjacobson.com/uploadedFiles/Pages/Knowledge_Centre/

Martinez, R. (10 de Octubre de 2010). *PostgreSQL - es*. Obtenido de Sobre PostgreSQL: http://www.postgresql.org.es/sobre_postgresql

Mora, S. (31 de Octubre de 2002). *Universidad de Alicante*. Obtenido de Programación de aplicaciones web: historia, principios básicos y clientes web: http://rua.ua.es/dspace/bitstream/10045/16995/1/sergio_lujan-programacion_de_aplicaciones_web.pdf

Notas de docencia. (2012). *Notas docencia: Javascript.* Aplicaciones con Tecnología en Internet.

Notas docencia. (2012). *Notas docencia: jQuery.* Aplicaciones con Tecnologías en Internet.

Otto, M,. (s.f.). *Bootstrap*. Obtenido de Bootstrap: http://getbootstrap.com/

Torvalds L. (s.f.). *Empezando acerca del control de versiones.* Obtenido de git scm: http://git-scm.com/book/es/v1/Empezando-Acerca-del-control-de-versiones

W3C. (2 de Marzo de 2013). *What is HTML*. Obtenido de The World Wide Web Consortium: http://www.w3.org/community/webed/wiki/HTML/Training/What_is_HTML%3F

W3C España. (s.f). *Guía Breve de CSS*. Obtenido de World Wide Web Consortium España: http://www.w3c.es/Divulgacion/GuiasBreves/HojasEstilo

W3C. (s.f). *Media queries.* Obtenido de World Wide Web Consortium: http://www.w3.org/TR/2001/WD-css3-mediaqueries-20010404/