Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación
Centro de Ingeniería de Software y Sistemas (ISYS)

Desarrollo de una aplicación web denominada «ProAgil» para automatizar actividades y artefactos de evaluación de usabilidad de indagación e inspección para procesos de desarrollo de software

Trabajo Especial de Grado
presentado ante la ilustre
Universidad Central de Venezuela
por los bachilleres:

Sandra Paola Jimenez Garcilazo
Jesús Aldemaro Díaz Hernández

para optar al título de
Licenciado en Computación

Tutora:
Profa. Jossie Zambrano

Ciudad Universitaria de Caracas

mayo 2015

Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación

Acta

Quienes suscriben, 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 una aplicación web denominada "ProAgil" para automatizar actividades y artefactos de evaluación de usabilidad de indagación e inspección para proceso de desarrollo de software», presentado por los bachilleres: Jesús Aldemaro Díaz Hernández, C.I.: 19.819.020 y Sandra Paola Jimenez Garcilazo, C.I.: 20.638.898, a los fines de optar por el título de Licenciado en Computación, dejan constancia de lo siguiente: Leído el trabajo por cada uno de los miembros del jurado, se fijó el día Jueves 21 de mayo de 2015, a las 08:30 am, para que sus autores lo defendieran en forma pública en la Escuela de Computación, mediante una presentación oral de su contenido, luego de lo cual respondieron 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 veintiún (21) días del mes de mayo del año dos mil quince (2015).

Profa. Jossie Zambrano – Tutora

Profa. Lucía Ojeda – Jurado

Prof. Eugenio Scalise – Jurado

Dedicatoria

A mi Mamá y Papá
Por su amor y su apoyo incondicional en mis decisiones

A Dios
Por la oportunidad de vivir esta experiencia
Sandra Jimenez

A mi ángel mayor, quien me cuida día y noche. Mis logros siempre serán para ti
Aldemaro Díaz

Agradecimientos

A Dios
Por siempre colocar en mi camino a las personas y las situaciones adecuadas para así lograr cumplir mis sueños. Gracias por siempre darme fuerza para enfrentar los retos que se me presenten

A ustedes Mamá y Papá
Gracias por siempre querer lo mejor para mí. Gracias por siempre darme el apoyo incondicional y guiarme en mis decisiones. Gracias por hacer todos los sacrificios necesarios para que mis sueños sean cumplidos. Gracias por educarme con valores para ser lo soy hoy en día
.

A ustedes Familia
A mi padrino José Gregorio por apoyarme, por estar siempre pendiente de mí, aconsejarme y prestarme su ayuda. A mis abuelas, mis tíos y mis primos, siempre representan un apoyo en mi vida

A ustedes profesores
Gracias todos los profesores que tuve durante todos mis estudios académicos, por compartir sus conocimientos y consejos

A mis Amigos
Por hacerme reír y compartir buenos y malos momentos juntos. Gracias Aldemaro por tu compañía y apoyo tanto en lo académico como en mi vida personal, gracias por hacerme reír, llorar de felicidad y aconsejarme en todo momento

A ti Jossie
Gracias por tu dedicación, tu tiempo, tus consejos y ser más que una tutora y adoptarnos como tus hijos. Gracias por ayudarnos a lograr esta meta

Sandra Jimenez

Gracias a Dios, por ser mí guía cada día de mi vida

Gracias a la Universidad Central de Venezuela por permitirme formarme en tus aulas

Gracias a Klari, Klaireth, Klarelys, Mari, Kaki, Nena y Betania, mi pequeña gran familia, por siempre estar, por su apoyo sin condiciones. Dios los bendiga infinitamente

Gracias a Sandra, mi compañera y amiga, por ser la voz de mí conciencia, por enseñarme a ver siempre el lado positivo de las cosas y por darme la certeza de que todo estaría bien. El mayor de los éxitos para ti hoy y SIEMPRE
¡Lo logramos!

Gracias a Jossie, por confiar en nosotros, por su sinceridad y por quedarse para apoyarnos a terminar nuestro TEG. Cumplimos juntos el compromiso ♥

Gracias a Eleonora, por ser la semillita de este trabajo, por sus sabios consejos y por enseñarme que las cosas deben hacerse con el corazón.
Estaré eternamente agradecido

Gracias a mi grupo docente favorito: Lucía, Jossie y Eleonora. Por la confianza, por darme la oportunidad de trabajar junto a ustedes durante cinco semestres, por permitirme aprender a través de la enseñanza

Gracias a mis amigos, simplemente por siempre estar pendiente, por las sonrisas y momentos compartidos

Gracias a Lau, Hailyx, Anak, Kari, Gamar, Key, Mari, Isthar, Steph y Jhonatan por acompañarme en mi camino por la universidad.
El mayor de los éxitos para ustedes

Gracias a la familia de Siete27, por darme la oportunidad de crecer como profesional, en especial a Daniel Thompson por su confianza y sus enseñanzas

El éxito es el resultado de aquello que planeas con amor

Aldemaro Díaz

Resumen

Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación
Centro de Ingeniería de Software y Sistemas (ISYS)

Desarrollo de una aplicación web denominada «ProAgil» para automatizar actividades y artefactos de evaluación de usabilidad de indagación e inspección para procesos de desarrollo de software

Autores
Sandra Paola Jimenez Garcilazo – sandrap1992@gmail.com
Jesús Aldemaro Díaz Hernández – aldemaro.diaz@gmail.com

Tutora
Profa. Jossie Zambrano – jossie.zambrano@gmail.com

En la actualidad, el auge del desarrollo de aplicaciones web y móviles ha aumentado, por esta razón los grupos de desarrollo se ven en la necesidad de incorporar mecanismos para potenciar los aspectos de calidad del software. Uno de estos aspectos de calidad es la usabilidad, la cual se define como la capacidad del producto de software para ser entendido, aprendido, usado y ser atractivo para el usuario, cuando se usa bajo condiciones específicas (ISO/IEC). Entre los artefactos para la evaluación de usabilidad se encuentran los de indagación e inspección. A través del uso de los artefactos de *indagación* se puede determinar requerimientos y características del software a desarrollar. Por otro lado, los artefactos de *inspección* permiten detectar errores o inconsistencias en los sistemas. En muchos casos, para la aplicación de estos artefactos es necesario usar diferentes herramientas de software o generarlos manualmente lo que puede ocasionar desorden en la información de los proyectos. Enmarcados en la línea de investigación del centro ISYS de la Escuela de Computación en la Facultad de Ciencias de la Universidad Central de Venezuela surge este Trabajo Especial de Grado que tiene como objetivo desarrollar la aplicación web "ProAgil" para automatizar actividades y artefactos de evaluación de usabilidad de indagación e inspección para procesos de desarrollo de software. Para el desarrollo de la aplicación se utilizó el método AgilUs, el cual incorpora artefactos de evaluación de usabilidad en sus diferentes etapas.

Palabras Clave
Evaluación de usabilidad, indagación, inspección, procesos de desarrollo de software, AgilUs.

Índice general

Introducción 1

Capítulo 1: Artefactos de evaluación de Usabilidad 4
1.1 Usabilidad 4
1.2 Evaluación de Usabilidad 5
1.2.1 Artefactos de indagación 6
(i) Tormenta de ideas 6
(ii) Análisis de sistemas existentes 7
(iii) Sondeos 8
1.2.2 Artefactos de inspección 9
(i) Evaluación heurística 10
(ii) Lista de comprobación 11
(iii) Guía de estilos 12
Capítulo 2: Proceso de desarrollo de software 13
2.1 Definición 13
2.2 Desarrollo ágil 14
2.3 AgilUs 15
2.3.1 Etapas 15
(i) Requisitos 16
(ii) Análisis 17
(iii) Prototipaje 18
(iv) Entrega 19
Capítulo 3: Tecnologías y herramientas web 20
3.1 Tecnologías del lado del cliente 20
3.1.1 HTML 20
3.1.2 CSS 20
3.1.3 JavaScript 21
3.1.4 jQuery 21
3.1.5 Bootstrap 21
3.1.6 AJAX 21
3.2 Tecnologías del lado del servidor 22
3.2.1 PHP 22
3.2.2 PostgreSQL 23

3.3 Herramientas web 24
3.3.1 Laravel 24
(i) Modelo 26
(ii) Vista 26
(iii) Controlador 27
3.3.2 Git 28
Capítulo 4: Desarrollo de la aplicación web ProAgil 30
4.1 Iteración 1: Análisis global 31
4.1.1 Etapa de Requisitos 31
(i) Tormenta de ideas 31
(ii) Sondeo 33
(iii) Perfiles de usuario 38
(iv) Requerimientos funcionales y no funcionales 38
4.1.2 Etapa de Análisis 40
(i) Casos de uso 40
(ii) Objetos del dominio 45
(iii) Guía de estilos 46
(iv) Prototipo en papel 47
4.2 Iteración 2: Módulo de proyectos y actividades 47
4.2.1 Etapa de Requisitos 48
(i) Análisis de sistemas existentes 48
4.2.2 Etapa de Análisis 51
(i) Prototipo en papel 51
4.2.3 Etapa de Prototipaje 52
(i) Prototipo ejecutable 53
(ii) Evaluación Heurística 55
4.3 Iteración 3: Módulo para generar formularios 56
4.3.1 Etapa de Requisitos 57
(i) Análisis de sistemas existentes 57
4.3.2 Etapa de Análisis 60
(i) Prototipo en papel 60
4.3.3 Etapa de Prototipaje 61
(i) Prototipo ejecutable 62
(ii) Evaluación heurística 65
4.4 Iteración 4: Módulo de componentes gráficos 66

4.4.1 Etapa de Requisitos 67
(i) Análisis de sistemas existentes 67
4.4.2 Etapa de Análisis 69
(i) Prototipo en papel 69
4.4.2 Etapa de Prototipaje 70
(i) Prototipo ejecutable 71
(ii) Evaluación heurística 72
4.5 Iteración 5: Pruebas de aceptación 73
Conclusiones y Recomendaciones 75
Referencias 78
Anexos 80

Índice de figuras

Figura 1 – Tipos de artefactos de Evaluación de Usabilidad 6
Figura 2 – Artefactos de indagación 6
Figura 3 – Ejemplo de tormenta de ideas (Alba et al, 2014) 7
Figura 4 – Artefactos de inspección 9
Figura 5 – Capas de la Ingeniería de Software (IS) 13
Figura 6 – Etapas del método AgilUs 16
Figura 7 – Proceso de funcionamiento de las páginas con PHP 23
Figura 8 – Patrón MVC 25
Figura 9 – Patrón MVC en Laravel 25
Figura 10 – Creación de Modelos en Laravel 26
Figura 11 – Llamada de modelo en Laravel 26
Figura 12 – Imprimir en la vista con código PHP 27
Figura 13 – Imprimir en la vista con código PHP usando Blade 27
Figura 14 – Controlador usando *Routes with Closures* 27
Figura 15 – Controlador en Laravel 27
Figura 16 – Estructura de Git 28
Figura 17 – Ambiente de desarrollo 29
Figura 18 – Iteraciones para la construcción de la aplicación web ProAgil 30
Figura 19 – Tormenta de ideas 32
Figura 20 – Casos de uso: Nivel 1 41
Figura 21 – Casos de uso: Nivel 2 42
Figura 22 – Objetos del dominio 45
Figura 23 – Prototipo en papel: Inicio 47
Figura 24 – Prototipo en papel: Creación de proyecto y detalle de proyecto 52
Figura 25 – Arquitectura de software 53
Figura 26 – Prototipo ejecutable: Crear proyecto (Información de proyecto) 53
Figura 27 – Prototipo ejecutable: Crear proyecto (Iteraciones) 54
Figura 28 – Prototipo ejecutable: Detalle de proyecto 54
Figura 29 – Prototipo ejecutable: Detalle de actividad 55
Figura 30 – Prototipo en papel: Generar formulario de tipo sondeo 61
Figura 31 – Prototipo en papel: Crear lista de comprobación 61
Figura 32 – Prototipo ejecutable: Crear sondeo 62
Figura 33 – Prototipo ejecutable: Listado de sondeos 63
Figura 34 – Prototipo ejecutable: Crear lista de comprobación 63
Figura 35 – Prototipo ejecutable: Listado de lista de comprobación 64
Figura 36 – Prototipo ejecutable: Verificar lista de comprobación 64
Figura 37 – Prototipo en papel: Guía de estilos 70
Figura 38 – Prototipo en papel: Tormenta de ideas 70
Figura 39 – Prototipo ejecutable: Guía de estilos 71
Figura 40 – Prototipo ejecutable: Resultado de crear tormenta de ideas 72

Índice de tablas

Tabla 1 – Ejemplo de Análisis de sistemas existentes 8
Tabla 2 – Ejemplo de evaluación heurística 11
Tabla 3 – Pregunta 9: ¿Cuál de los siguientes temas le parece más atractivo?.....36
Tabla 4 – Guía de estilos 47
Tabla 5 – Análisis de sistema existente *Freedcamp* 49
Tabla 6 – Análisis de sistema existente *Asana* 51
Tabla 7 – Evaluación heurística de la iteración 2 56
Tabla 8 – Análisis de sistema existente *Google Forms* 58
Tabla 9 – Análisis de sistema existente *Checkli*.....60
Tabla 10 – Evaluación heurística de iteración 3 66
Tabla 11 – Análisis de sistemas existentes *Stylify Me* 68
Tabla 12 – Análisis de sistemas existentes *Wordle* 69
Tabla 13 – Evaluación heurística: Iteración 4 73

Introducción

En la actualidad, el desarrollo de aplicaciones y el número de usuarios que las utilizan ha aumentado considerablemente. Por esta razón, las empresas y los equipos de desarrollo de software se ven en la necesidad de incorporar mecanismos eficientes que le permitan construir software de calidad en el menor tiempo posible. Dependiendo de la audiencia a la que va dirigida una aplicación y las necesidades del cliente el grupo de desarrollo debe tomar en cuenta los aspectos de calidad potenciales que deben tener los sistemas. Uno de los aspectos de calidad que se puede considerar al momento de desarrollar es la Usabilidad, este aspecto se puede definir intuitivamente como un atributo de calidad software que indica qué tan fácil de usar es una interfaz de usuario (Nielsen, 2006). Existen diversos artefactos de evaluación de usabilidad que se pueden incorporar en los procesos de desarrollo de software para potenciar este atributo de calidad.

Entre la variedad de artefactos de evaluación de usabilidad están los propuestos por la Fundación SIDAR, los cuales son: artefactos de indagación, de inspección, de prototipaje, y de pruebas. Este Trabajo Especial de Grado (TEG) está basado en los artefactos de indagación e inspección. Los artefactos de *indagación* permiten determinar requerimientos y características del software a desarrollar. Por otro lado, con el uso de artefactos de *inspección* se puede detectar errores o inconsistencias en las aplicaciones. Los artefactos suelen ser diferentes en cuanto a su contenido, estructura e información lo que genera como consecuencia que se utilicen diversas herramientas para generarlos. Adicionalmente, algunos de éstos se generan manualmente o de forma semiautomática utilizando hojas de cálculo, procesadores de palabras, repositorios de documentos compartidos, correo electrónico y otras herramientas de software, lo que puede generar inconsistencia en la información y componentes del proceso de desarrollo. De igual manera, la desinformación se presenta con respecto a las actividades que se deben realizar en los proyectos, ya que en algunos casos no se lleva de manera formal el grupo de actividades que conforman un proyecto y a cuál miembro del equipo de desarrollo pertenece.

Debido a la problemática descrita y con la finalidad de apoyar a los grupos de desarrollo que deseen utilizar evaluaciones de usabilidad en sus proyectos, desde el centro ISYS de la Escuela de Computación de la Facultad de Ciencias en la Universidad Central de Venezuela se propone crear una aplicación web que permita generar y aplicar artefactos de evaluación de usabilidad de manera automatizada y centralizada, específicamente artefactos de inspección e indagación. Esto da pie a enunciar los objetivos del presente TEG.

El objetivo general de este TEG es desarrollar la aplicación web "ProAgil" para automatizar actividades y artefactos de evaluación de usabilidad de indagación e inspección para procesos de desarrollo de software.

Para dar cumplimiento al objetivo general, se presenta los objetivos específicos de este trabajo:

- Analizar herramientas que apoyan el proceso de desarrollo de software para tomar en cuenta las buenas prácticas utilizadas
- Seleccionar las tecnologías y herramientas web para el desarrollo de la aplicación
- Desarrollar un módulo para la configuración de proyectos
- Desarrollar el módulo para la gestión y asignación de actividades al grupo de desarrollo
- Automatizar los artefactos de indagación: tormenta de ideas, sondeo y análisis de sistemas existentes
- Automatizar los artefactos de inspección: evaluación heurística, lista de comprobación y guías de estilos
- Aplicar el método AgilUs para el desarrollo de la aplicación web ProAgil

Para lograr los objetivos mencionados para el desarrollo del presente Trabajo Especial de Grado (TEG) se propone la siguiente estructura organizada en cuatro (4) capítulos de la siguiente forma:

Capítulo 1 – Artefactos de evaluación de Usabilidad: Con la finalidad de contextualizar este trabajo, en el primer capítulo se definen los conceptos de usabilidad y evaluación de usabilidad haciendo énfasis en los artefactos de indagación e inspección en los que se enfoca este TEG los cuales son: tormenta de ideas, análisis de sistemas existentes y sondeos (de indagación), y evaluación heurística, lista de comprobación y guía de estilos (de inspección).

Capítulo 2 – Proceso de desarrollo de software: Este capítulo tiene como objetivo describir el proceso de desarrollo de software haciendo énfasis en el enfoque ágil a través del método AgilUs el cual es utilizado para el desarrollo de esta investigación.

Capítulo 3 – Tecnologías y herramientas web: Corresponde al capítulo de los aspectos técnicos utilizados para el desarrollo de la aplicación web. Se describen las tecnologías del lado del cliente y del lado del servidor utilizadas. Adicionalmente se describen otras herramientas web utilizadas para el desarrollo.

Capítulo 4 – Implementación de la aplicación web ProAgil: En este capítulo se presenta la experiencia del desarrollo de la aplicación web ProAgil teniendo en cuenta

los aspectos teóricos y utilizando un grupo de tecnologías a través del método de desarrollo de software AgilUs. Se describen las iteraciones y los artefactos utilizados detallando algunos de ellos.

Finalmente se presentan las conclusiones, recomendaciones, referencias y anexos que dan soporte a este Trabajo Especial de Grado.

Capítulo 1: Artefactos de evaluación de Usabilidad

El presente capítulo tiene como objetivo detallar los aspectos teóricos relacionados a los artefactos de evaluación de usabilidad. Específicamente los artefactos de indagación e inspección. Para dar cumplimiento al objetivo enunciado, en principio se define el concepto de usabilidad. Seguidamente se presenta el concepto de evaluación de usabilidad haciendo énfasis en los artefactos de indagación e inspección. Los artefactos de *indagación* descritos son: tormenta de ideas, análisis de sistemas existentes y sondeos. Luego, los artefactos de *inspección* que se detallan son: evaluación heurística, lista de comprobación y guía de estilos. Es importante estudiar estos conceptos debido a que son la base conceptual para el desarrollo de la aplicación que sustenta esta investigación.

1.1 Usabilidad

Según la *International Organization for Standardization* y la *International Electrotechnical Commission* (ISO/IEC) en las especificaciones de su estándar 9126, se define la usabilidad como la capacidad del producto software para ser entendido, aprendido, usado y ser atractivo para el usuario, cuando se usa bajo condiciones específicas. Adicionalmente, en este estándar se especifican cuatro (4) características del software relacionadas a este aspecto de calidad las cuales son: aprendizaje, comprensión, operatividad y atractividad. El *aprendizaje* hace referencia al esfuerzo que los usuarios deben hacer para aprender a usar la aplicación. La *comprensión* se refiere al esfuerzo requerido por los usuarios para reconocer la estructura lógica del sistema y los conceptos relativos a la aplicación del software. La *operatividad* agrupa los conceptos que evalúan la operación y el control del sistema. Referente a la *atractividad* verifica qué tan atractiva es la interfaz de usuario de la aplicación.

Existen diversos autores que han hecho estudios sobre la Usabilidad, entre ellos Steve Krug y Jakob Nielsen. Krug (2000) afirmó "la usabilidad realmente significa estar seguro de que algo funciona bien: que una persona con habilidades promedio e incluso por debajo del promedio pueda utilizar una cosa (ya sea un sitio web, un jet de combate, o una puerta rotatoria) sin terminar frustrado". Por otro lado, Nielsen (2006) define la usabilidad como un atributo de calidad que indica qué tan fácil de usar es una interfaz de usuario. Además, afirma que un sistema es usable si cumple con los siguientes cinco (5) atributos: fácil aprendizaje, memorización, eficiencia, tolerancia a errores y satisfacción. El primer atributo, *fácil aprendizaje* especifica que el sistema debe ser fácil de aprender, de tal manera que el usuario pueda comenzar rápidamente a utilizarlo. La *memorización* se refiere al tiempo que le toma al usuario recordar cómo utilizar el sistema desde la última vez que lo utilizó, lo ideal es que este tiempo sea el menor posible. En cuanto a la *eficiencia*, es deseable que al aprender a usar el sistema,

el usuario tenga un nivel de productividad alto. La *tolerancia a errores* busca minimizar la cantidad de equivocaciones posibles cuando el usuario utiliza el software y en caso que ocurra, el sistema debe tener las facilidades necesarias para proveer una solución. El último atributo es la *satisfacción* el cual hace referencia a qué tan a gusto se siente el usuario al utilizar un sistema.

Partiendo de la definición basada en los cinco (5) atributos de Nielsen, Acosta (2013) afirma que la usabilidad es una cualidad que impacta positivamente en la calidad del software ya que el diseño centrado en el usuario resulta en productos de mayor calidad de uso y más competitivos en el mercado. Los sistemas difíciles de usar disminuyen la salud, bienestar, productividad y motivación de los usuarios. Adicionalmente, es necesario considerar las necesidades y demandas de los usuarios en cada etapa del desarrollo de software, lo cual podría incluir: planificación, recopilación de datos de los usuarios, desarrollo de prototipos y aplicación de evaluaciones de usabilidad; esta última consideración será explicada a continuación.

1.2 Evaluación de Usabilidad

En general, la evaluación es un aspecto fundamental a considerar en el diseño centrado en el usuario (DCU) el cual se centra en los deseos, limitaciones y necesidades de los usuarios finales de un software. La evaluación permite saber si un sistema cumple las expectativas de los usuarios y se adapta a su contexto social, físico y organizacional. En particular, la evaluación de usabilidad determina el grado de usabilidad de un software (Acosta, A., 2013). En el mismo orden de ideas (Hwang y Salvendy, 2010) afirman que la evaluación de la usabilidad aseguran que los productos de software sean fáciles de usar, eficientes, eficaces y satisfactorios para los usuarios. Su aplicación constante en el desarrollo de software permite construir la usabilidad del producto final durante su ciclo de vida. Con la finalidad de potenciar este aspecto de calidad en productos de software, es necesario aplicar artefactos de evaluación de usabilidad desde etapas iniciales del desarrollo.

Existen diversos artefactos para la evaluación de usabilidad los cuales pueden ser aplicados por distintos actores (usuarios, especialistas, entre otros) y en diferentes momentos del desarrollo de software, incluso una vez que ya se ha puesto en producción. La Fundación SIDAR, en su recopilación de "Métodos de Usabilidad" (2000), los agrupa en cuatro (4) categorías, a saber: indagación, prototipaje, pruebas e inspección, en la **Figura 1** se ilustra la clasificación de las categorías y los artefactos pertenecientes a cada una. Este Trabajo Especial de Grado se enfoca en la definición de los artefactos de evaluación de usabilidad de *indagación* e *inspección* los cuales se describen a continuación.

Figura 1 – Tipos de artefactos de Evaluación de Usabilidad

1.2.1 Artefactos de indagación

En general, la indagación trata de llegar al conocimiento de algo a través de preguntas y exploración. Los artefactos o técnicas que se agrupan en esta categoría deben proporcionar información acerca de la usabilidad de un producto que aún no se ha empezado a construir (Acosta A., 2013). Los artefactos de indagación se suelen aplicar en el inicio de un proyecto, pero también se pueden utilizar en cualquier fase o momento del proceso de diseño; "lo que define realmente un artefacto de indagación es su naturaleza de intentar averiguar o investigar algo deduciendo o con preguntas, para establecer unos requerimientos" (Solano et al, 2009).

Los artefactos que conforman la categoría de indagación son: tormenta de ideas, análisis de sistemas existentes y sondeos ilustrados en la Figura 2 y se explican a continuación.

Figura 2 – Artefactos de indagación

(i) Tormenta de ideas

Consiste en la generación de ideas por parte de un grupo, liberando la mente de cada integrante para aceptar cualquier idea que se proponga, permitiendo así, la libertad para la creatividad. En el caso del proceso de desarrollo de software, corresponde a los actores involucrados en el desarrollo de las aplicaciones. El resultado de esta actividad será un conjunto de ideas, y la contextualización del sistema a desarrollar. Para llevar a cabo una tormenta de ideas exitosa, al inicio se establecen los objetivos de la sesión y

los participantes. Se debe establecer un acuerdo acerca de las técnicas de registro o grabación a utilizar por ejemplo: audio, vídeo, documento, nube de palabras, entre otros. Durante la sesión, el moderador deberá fomentar la discusión, no permitir la crítica y reunirá los resultados obtenidos al final de cada aspecto tratado. Será importante distinguir entre el consenso del grupo y la opinión de los diferentes participantes. Al culminar la sesión se discuten los resultados obtenidos, dando importancia en los cuales los miembros del grupo hayan tenido mayor coincidencia. Es recomendable aplicar esta técnica en las etapas tempranas del proceso de desarrollo, cuando se conoce poco acerca del diseño real y hay necesidad de nuevas ideas, a fin de determinar parte de los requerimientos y cualidades del software del sistema a desarrollar. En la Figura 3 se presenta un ejemplo del resultado de una tormenta de ideas para la construcción de un sistema de reservas en línea como parte del proyecto de la materia Interacción Humano-Computador de la Escuela de Computación de la UCV. Para su elaboración el grupo de desarrollo se reunió con el preparador y sus compañeros de clases para discutir características relacionadas a sistemas de reservas, las palabras se iban registrando en la herramienta online *Wordle* para finalmente generar la nube de palabras presentada.

Figura 3 – Ejemplo de tormenta de ideas (Alba et al, 2014)

(ii) Análisis de sistemas existentes

Es una técnica que consiste en la revisión de versiones anteriores de un sistema o sistemas similares, con el objetivo de identificar problemas, características y virtudes en cuanto a la usabilidad. Esto permite, prevenir los problemas en el sistema que se construye y obtener una medida de base para la usabilidad del mismo. Entre los principales beneficios de esta técnica se pueden mencionar: identificar problemas que podrán ser evitados en el diseño del nuevo sistema y provee medidas de efectividad, eficiencia y satisfacción que pueden ser usadas como base para el nuevo sistema. Los tópicos a evaluar por cada sistema existente son establecidos por el grupo de desarrollo, sin embargo, se proponen realizar observaciones positivas y/o negativas de los sistemas basadas en las siguientes características: descripción, funcionalidades, personalización, sociabilidad, idiomas, apariencia, accesibilidad, soporte en línea, *look and feel* y opinión como usuario. Una vez que se analicen los sistemas, el equipo de

desarrollo puede hacer reuniones para definir qué aspectos se pueden incluir en el sistema a desarrollar. En la Tabla 1 se muestra un ejemplo de un análisis para la creación de un juego colaborativo (Montenegro et al, 2013). Para generar este artefacto se utilizó el procesador de textos *Word*

1, 2, 3 Bee – http://latrola.net/blok/juegos/escapanumeros.swf	
Tópico a evaluar	**Observaciones (positivas y/o negativas)**
Funcionalidades	Buscar números que están escondidos en una imagen en un tiempo restringido
Personalización	El usuario no puede modificar ningún aspecto de la interfaz del juego
Sociabilidad	No se puede compartir en redes sociales
Apariencia	Interfaz sencilla, con colores sobrios entre negro y gris, y botones grandes. Los números a buscar están a un lado de la pantalla y se oscurecen cuando el número es encontrado
Aspectos de accesibilidad	El juego tiene una lupa que hace *zoom in* a los objetos lo que ayuda a personas con problemas visuales
Soporte en línea	El juego tiene una opción de instrucciones para el usuario
Idiomas	Solo disponible en inglés
Opinión como usuario	Es un juego fácil de jugar y entretenido

Tabla 1 – Ejemplo de Análisis de sistemas existentes

Luego del análisis de sistemas existentes, el equipo de desarrollo decide qué aspectos de los sistemas evaluados se incluirán con la finalidad de potenciar la usabilidad.

(iii) Sondeos

De manera intuitiva, un sondeo es una medición estadística tomada a partir de encuestas destinadas a conocer la opinión pública (Bradburn, N.,2007). Aplicado a la Interacción Humano-Computador y procesos de desarrollo de software, se desea saber la opinión de usuarios para determinar: requerimientos funcionales, no funcionales y aspectos de la interfaz de usuario de algún sistema que está por iniciar. Las mediciones se realizan por medio de muestreos que, usualmente están diseñados para representar las opiniones de una población llevando a cabo una serie de preguntas las cuales se sintetizan a través de tablas de resultados o gráficos. Al realizar sondeos para la creación de un sistema se recomienda que se presenten a los usuarios preguntas con opciones (preguntas cerradas), aunque se puede dar el caso que se requiera alguna pregunta abierta como una opinión de algún aspecto. Las preguntas cerradas pueden ser de selección simple o selección múltiple y las preguntas abiertas pueden ser de una respuesta breve no mayor a una línea o un párrafo. Además, es

necesario que cada pregunta tenga un propósito enfocado al sistema que se está realizando.

El anexo *a: sondeo para el desarrollo de un videojuego colaborativo* presenta un ejemplo de sondeo, el objetivo de éste es obtener información para el desarrollo de un videojuego colaborativo (Figueroa M., Trejo J., 2013). Luego de aplicar el sondeo a treinta y cuatro (34) personas se obtuvieron resultados sobre funcionalidades y la interfaz de usuario. Algunos de los resultados obtenidos *sobre funcionalidades* son*:* el 68% de los usuarios prefiere que el juego sea desarrollado para PC. El 82% de los encuestados prefiere que los personajes suban de nivel y a un 59% no le gustaría la posibilidad de compartir los logros en las redes sociales. Con respecto a la *interfaz de usuario*: el 56% de los usuarios prefiere colores fríos para la interfaz de usuario. Al 91% de los usuarios les gustaría poder personalizar sus personajes y a un 68% les gustaría tener una ayuda para aprender a jugar. Luego de analizar los resultados el equipo de desarrollo pudo tomar decisiones con respecto a requerimientos del juego a desarrollar.

Adicionalmente a los artefactos de indagación descritos, en la siguiente sección se describen los artefactos de inspección para la evaluación de usabilidad en los que se basa este Trabajo Especial de Grado.

1.2.2 Artefactos de inspección

Los artefactos de inspección se asocian evaluaciones de proyectos en proceso o terminados. Estos artefactos al igual que los artefactos de indagación se pueden utilizar en cualquier momento del proceso de desarrollo de software. De manera intuitiva los artefactos de inspección permiten buscar errores o inconsistencias en las interfaces de usuario en una forma detallada. Pueden ser aplicados por expertos, usuarios o desarrolladores. Este tipo de evaluación puede implicar observación, medición y comparación con respecto a características de usabilidad agrupando un conjunto de técnicas que permiten verificar el grado de usabilidad a través de la inspección de interfaces de usuario. Los artefactos de inspección a automatizar en este TEG son: evaluación heurística, lista de comprobación y guías de estilos ilustrados en la Figura 4. Estos artefactos son descritos a continuación

Figura 4 – Artefactos de inspección

(i) 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. Nielsen propone una lista de diez (10) heurísticas, resultando suficiente y aceptable para cualquier evaluación de diseño:

- H1. Diálogo natural y simple
- H2. Hablar el lenguaje del usuario
- H3. Minimizar la carga cognitiva
- H4. Consistencia
- H5. *Feedback*
- H6. Proveer claramente las salidas
- H7. Proveer *shortcuts*
- H8. Mensajes de error descriptivos
- H9. Prevención de errores
- H10. Asistencia al usuario

Cuanto mayor sea el número de especialistas que revisan la interfaz de usuario, mayor será el número de errores que se podrán encontrar, pero el costo se incrementa. El rango ideal de especialistas según Nielsen es entre tres (3) y cinco (5) especialistas. Una vez que se dispone de los especialistas, estos realizan la evaluación de la interfaz de usuario individualmente, fijándose en cada elemento de la misma. Luego de realizar la evaluación, los especialistas se reúnen para discutir los problemas de usabilidad encontrados al realizar la evaluación de forma individual. Luego de la discusión se crea un informe con la recopilación de todos los problemas. En el informe se especifica las heurísticas que no se están cumpliendo, indicando el grado del problema utilizando la siguiente escala:

- 0: No es un problema de Usabilidad
- 1: Problema cosmético
- 2: Problema menor
- 3: Problema mayor de Usabilidad; importante fijar solución
- 4: Usabilidad catastrófica, imperativo fijar solución

Adicionalmente se debe proporcionar una orientación para su solución. La evaluación heurística es posible aplicarla en cualquier momento del ciclo de desarrollo. Se puede proporcionar prototipos en papel o incluso especificaciones de diseño a los expertos y detectar una buena cantidad de problemas de usabilidad antes de que el trabajo real de producción de comienzo.

En la Tabla 2 se muestra un informe de evaluación heurística, con la recopilación de algunos los errores encontrados sobre un prototipo de un juego de memoria colaborativo. Para su realización tres (3) especialistas hicieron el recorrido del

prototipo ejecutable para obtener los errores, a continuación se describen los problemas encontrados

Problema	Heurística	Valoración	Solución
Al iniciar el juego están las cartas y el campo de ingresar el nombre de usuario en una misma interfaz de usuario	H3	3	Mostrar en una primera interfaz de usuario el campo para introducir el nombre de usuario y luego de darle al botón JUGAR mostrar las cartas
No existe ningún tipo de ayuda o instrucciones para ayudar a los usuarios	H10	3	Crear una sección de ayuda para que los usuarios puedan entender qué hay que realizar
No hay un botón para jugar de nuevo	H3	3	Colocar un botón que te permita empezar de nuevo el juego
Las cartas son de un solo color	H3	2	Utilizar distintos tipos de carta (corazones rojos, trébol, etc.)
No se utiliza bien el espacio para la ubicación de todas las secciones	H4	3	Aprovechar todo el espacio de una mejor manera para que no quede tan vacío (Uso horizontal para evitar hacer *scroll*)

Tabla 2 – Ejemplo de evaluación heurística

Después de realizar el informe, se agrupan los errores por tipo, prioridad o cualquier aspecto fijado por el equipo de desarrollo. Posteriormente se resuelven tomando en cuenta la solución indicada por cada problema encontrado o la solución que decida el equipo.

(ii) 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. Es posible utilizar heurísticas o en algunos casos adaptar a los aspectos a los que se enfrenta el usuario del producto en desarrollo. Las listas de comprobación se pueden utilizar en cualquier etapa del ciclo de vida del software, siempre que se cuente con un prototipo del sistema en desarrollo. Para realizar este artefacto, se debe explorar el prototipo y decir cuáles de las heurísticas se cumplen o no, una de las listas conformada por guías para el diseño de la interfaz de usuario es la siguiente:

- Reconocer en lugar de recordar
- Usar indicadores visuales
- Utilizar metáforas adecuadas, y en lo posible, las metáforas estándares
- Los controles del sistema deben estar claramente visibles y sus funciones
- Flexibilidad en la interfaz de usuario
- Proveer *feedback* a las acciones del usuario
- Minimizar las posibilidades de cometer errores
- Permitir al usuario recuperarse de los errores
- Mantener la interfaz de usuario simple, sencilla y organizada
- Mostrar al usuario lo que necesita cuando lo necesite
- Evitar palabras coloquiales y abreviaturas
- Consistencia en apariencia y uso
- Proveer asistencia cuando el usuario lo requiera
- Mantener un formato consistente de una pantalla a otra
- Destacar la información importante

(iii) Guía de estilos

Las guías de estilos para el diseño de interfaces de usuario resumen para el diseño de interfaces usables. El uso de guías de estilos específicas se puede establecer como parte de los requerimientos no funcionales o requerimientos de usabilidad. El equipo de desarrollo debe estar bien familiarizado con estas guías, y se recomienda aplicar evaluaciones de expertos, similar a la lista de comprobación a fin de verificar el cumplimiento de las mismas. Las guías de estilos incorporan buenas prácticas en el diseño de interfaces de usuario teniendo de manera documentada los aspectos gráficos de los sistemas, lo cual sirve de apoyo al equipo de desarrollo. Entre los componentes de una guía de estilos se pueden mencionar: logotipos, tipografía, colores primarios y secundarios, estilos de navegación, estilos de menú e iconografía. En el anexo *b*: Guía de estilos del sistema PortalAsig se ilustra un ejemplo de guía de estilos. El anexo muestra los aspectos gráficos concernientes al sistema PortalAsig el cual es un generador de asignaturas para la Facultad de Ciencias de la Universidad Central de Venezuela. (Villlavicencio M., Díaz O., 2013).

Los artefactos para la evaluación de usabilidad de evaluación e inspección sirven de ayuda para potenciar la usabilidad en productos de software. Estos, no se pueden aplicar sin tener un mecanismo que especifique cuándo es recomendable aplicarlos. Por esta razón deben utilizarse enmarcados dentro de un proceso de desarrollo de software tópico que se presenta en el próximo capítulo.

Capítulo 2: Proceso de desarrollo de software

El presente capítulo tiene como finalidad describir el método de desarrollo de software utilizado para la realización de este Trabajo Especial de Grado. Para dar cumplimiento al objetivo planteado se define el proceso de desarrollo de software haciendo énfasis en el desarrollo ágil a través del método AgilUs describiendo las etapas y artefactos que lo conforman. Este método resultó el oportuno para el desarrollo ya que está enfocado en potenciar la usabilidad de los sistemas a través del uso de evaluaciones de usabilidad descritas en el capítulo anterior.

2.1 Definición

De manera intuitiva, un proceso define quién hace qué, cuándo y cómo lo hace, para alcanzar cierto objetivo. En general, el éxito de las organizaciones depende en gran medida de la definición y seguimiento adecuado de sus procesos. En el caso de una empresa que se dedica al desarrollo de software, se requieren múltiples procesos especializados que abarquen desde la creación hasta la administración de un sistema de software (Weitzenfeld R. y Guardati S., 2007). Específicamente, el proceso de desarrollo de software tiene como propósito producir productos de software de manera eficaz y eficiente que reúna los requisitos del cliente y los usuarios. El proceso de desarrollo de software, junto con las herramientas, métodos y enfoque de calidad conforman las capas de la Ingeniería de Software (IS) según la IEEE en su *Standard Glossary of Software Engineering Terminology* ilustradas en la Figura 5. Formalmente, el proceso de desarrollo de software forma la base para la gestión de los proyectos de software y establece el contexto en el cual se aplican los métodos técnicos, actividades, artefactos, aseguramiento de calidad y los cambios que puedan surgir al desarrollar productos de software (Pressman, 2010).

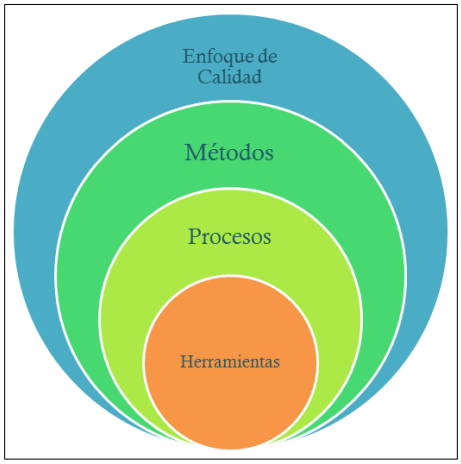


Figura 5 – Capas de la Ingeniería de Software (IS)

2.2 Desarrollo ágil

Actualmente, es necesario contar con métodos de desarrollo de software que den respuesta de manera eficiente ante cualquier cambio que pueda ocurrir en el desarrollo. Por esta razón, en el año 2001 surge el *Manifiesto Ágil* (Beck et al, 2001) que propone cuatro valores principales que se deben tener en cuenta al momento de desarrollar software de manera ágil:

- **Valorar individuos e interacciones** sobre procesos y herramientas: las personas son el principal factor de éxito de un proyecto software. Es más importante construir un buen equipo que construir el entorno de desarrollo.
- **Valorar software funcionando** sobre documentación extensiva: La regla a seguir es "no producir documentos a menos que sean necesarios de forma inmediata para tomar un decisión importante". Estos documentos deben ser cortos y centrarse en lo fundamental.
- **Valorar la colaboración con el cliente** sobre negociación contractual: se propone que exista una interacción constante entre el cliente y el equipo de desarrollo. Esta colaboración entre ambos será la que marque la marcha del proyecto y asegure su éxito.
- **Valorar la respuesta ante el cambio** sobre seguir un plan*:* La habilidad de responder a los cambios que puedan surgir a los largo del proyecto (cambios en los requisitos, en la tecnología, en el equipo, etc.) determina también el éxito o fracaso del mismo. Por lo tanto, la planificación no debe ser estricta sino flexible y abierta

Para el desarrollo ágil se utilizan metodologías que buscan ser más adaptables a los cambios continuos que se presentan durante el desarrollo de un sistema y para esto emplean un enfoque iterativo e incremental, con iteraciones cortas, planificación adaptativa y entrega evolutiva. Se busca lograr que los cambios sean menos costosos, permitiendo que sean incorporados fácilmente (Gabardini, J., & Campos, L. 2004). Así mismo permiten mayor flexibilidad que las metodologías tradicionales de desarrollo, las cuales se enfocan en los detalles del proyecto y son menos capaces de ajustarse a las cambiantes necesidades de los clientes, del mercado y de los desafíos imprevistos que plantea la tecnología (Ekas, L., 2012).

Para el desarrollo de este Trabajo Especial de Grado era necesario seleccionar un método de desarrollo de software ágil que potencie la usabilidad. Por esta razón el método ágil oportuno seleccionado es AgilUs, el cual incorpora en sus actividades y técnicas de evaluación de usabilidad. En la siguiente sección se define el método especificando sus etapas y artefactos.

2.3 AgilUs

AgilUs es un método de desarrollo de software utilizado para generar software usable, en este sentido, en esta sección se define éste método y las cuatro (4) etapas que lo conforman: Requisitos, Análisis, Prototipaje y Entrega.

El método 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 proceso 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.

El desarrollo de software a través de este método se realiza de manera iterativa e incremental incluyendo artefactos y técnicas que ayudan a desarrollar software usable. AgilUs, especifica 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). Se enfoca en cuatro (4) principios mencionados a continuación: Integración la Interacción Humano-Computador (IHC) y la Ingeniería de Software (IS); 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 y el usuario determina la usabilidad. Un software sólo será considerado usable en un contexto específico bajo cierto perfil de usuario. Estos principios son aplicados durante todo el proceso de desarrollo de software al utilizar éste método; por esta razón a continuación se describen las etapas que lo conforman.

2.3.1 Etapas

Este método propone realizar cuatro (4) etapas para el proceso de desarrollo de software, en cada una se aplican técnicas y artefactos de IHC e IS. 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 para crear prototipos ejecutables. Finalmente, se hace la *entrega* en donde se aplican *pruebas* con los usuarios finales. De manera sintetizada se ilustra en la Figura 6 las etapas y artefactos propuestos por este método de desarrollo.

Figura 6 – Etapas del método AgilUs

A continuación se describe las cuatro (4) etapas de AgilUs (Acosta, A., 2013).

(i) Requisitos

Se realiza una indagación del problema a solucionar. Se estudian productos similares existentes. Se genera un perfil de usuario y finalmente se define la lista de requerimientos funcionales y no funcionales a desarrollar. Específicamente los artefactos que propone el método en esta etapa son:

a. ***Análisis de sistemas existentes:*** Es una técnica que consiste en la revisión de versiones anteriores del mismo sistema o sistemas similares, con el objetivo de identificar cualidades positivas y problemas de usabilidad. Esto permite prevenir estos problemas en el sistema que se construye y obtener una medida de base para la usabilidad del mismo.
b. ***Tormenta de ideas:*** Es una técnica a utilizar cuando es necesario liberar la creatividad de un grupo, generar ideas en torno a un tema. Los principios que guían una tormenta de ideas son el aplazamiento de críticas y juicios sobre las ideas, nadie debe criticar una idea. No importa cuán disparatada o absurda sea, ya que esa idea puede permitir un desarrollo posterior más lógico. Así mismo es necesario fomentar la confianza en el grupo, impidiendo inhibiciones en los miembros.
c. ***Sondeo:*** Los sondeos son una técnica de indagación para la evaluación de usabilidad constan de un conjunto de preguntas que se le realizan al usuario para extraer información de lo que espera acerca del sistema. Para logar esto es necesario definir el objetivo del sondeo identificando qué información es la que se pretende obtener. Determinar la escala a utilizar para el sondeo, se puede

seleccionar una o varias escalas. Definir el número y tipo de preguntas, no existe un número ideal de preguntas para un sondeo, se debe recabar la información necesaria para cumplir un objetivo. Una vez recopilados los datos se analizan y se determinan los requerimientos funcionales, requerimientos no funcionales y el perfil de usuario que también forman parte de los artefactos de la etapa de requisitos y se explican a continuación.

d. ***Requerimientos funcionales****:* Los requerimientos funcionales de un sistema describen la funcionalidad o los servicios que se espera que éste provea, dependen del tipo de software y de los posibles usuarios. Es decir, son las acciones que realiza el usuario, por ejemplo: Iniciar sesión, consultar evento, publicar foto.
e. ***Requerimientos no funcionales:*** Son los requerimientos que no se refieren directamente a las funciones del sistema, sino a los criterios de calidad de éste, por ejemplo: usable, fácil de aprender, fácil de memorizar, satisfactorio, con prevención de errores, eficiente, confiable, seguridad, características de la plataforma a utilizar, etc.
f. ***Perfil de usuario:*** Constituye el elemento básico para diseñar un software. Se debe tomar en cuenta las características de los usuarios para el desarrollo de las funcionalidades e interfaz de usuario de la aplicación. Describe las cualidades y posibles limitaciones de la audiencia a la que va dirigida el software.

(ii) 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. Los artefactos para esta etapa son:

a. ***Modelo de objetos del dominio:*** El modelo de objetos del dominio 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.
b. ***Modelo de casos de uso:*** El Modelo de casos de uso 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 sistema. En este modelo se representan los requerimientos funcionales determinados en la etapa de requisitos.
c. ***Guía de estilos:*** Es un documento que recoge directrices relacionados con el aspecto de la interfaz de usuario de una aplicación tales como: Imagen corporativa o logotipo, contraste a utilizar, paleta de colores primarios y/o secundarios, tipo(s) de fuente(s) y características de los estilos de interacción. La guía de estilos debe integrarse de forma cómoda en el proceso de trabajo de

un desarrollador dándole respuestas a situaciones propias dentro de la construcción de la interfaz de usuario. De igual manera debe tener ejemplos aplicables y razonados, que permitan desarrollar criterios mínimos de usabilidad y estética al personal técnico con ejemplos útiles, actuales y materiales para su aplicación directa.

d. ***Prototipo en papel:*** Los prototipos en papel son una forma de crear una imagen palpable de lo que será una aplicación. Se trata de 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.
e. ***Patrones de interacción***: Un patrón de interacción describe una solución exitosa a un problema recurrente concerniente a la interfaz de usuario, en un contexto dado (Acosta, A., Zambrano, N., 2004). Describen aspectos concernientes a la interfaz de usuario; están orientados a presentar soluciones usables a problemas recurrentes que se les presentan a los usuarios cuando utilizan las aplicaciones interactivas. Los patrones de interacción surgen a partir de los modelos de casos de uso y objetos del dominio (Acosta, E., 2013).

(iii) 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. Adicionalmente, se realizan evaluaciones de usabilidad apropiadas a esta etapa: las evaluaciones heurísticas y las listas de comprobación. Los artefactos de esta etapa son:

a. ***Prototipo ejecutable:*** Representa una versión del software representado en el lenguaje de programación seleccionado por el sistema de desarrollo. AgilUs tiene la ventaja que no limita al grupo de desarrollo a trabajar con un lenguaje en particular por lo que se podría aplicar para realizar aplicaciones de distintas naturaleza (aplicación móvil, web, de escritorio,…).
b. ***Lista de comprobación****:* Son recomendaciones que el equipo de desarrollo ha acordado considerar en el diseño de la interfaz de usuario. 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. Es posible utilizar las heurísticas de Nielsen o en algunos casos adaptar a lo aspectos a los que se enfrenta el usuario del producto en desarrollo. Las listas de comprobación se pueden utilizar en cualquier etapa del ciclo de vida del software, siempre que se cuente con un prototipo del sistema en desarrollo.
c. ***Evaluación Heurística:*** Es un método de inspección de usabilidad en donde evaluadores juzgan si cada elemento de la interfaz de usuario sigue ciertos principios o heurísticas de usabilidad establecidos, los evaluadores pueden ser especialistas o usuarios finales. El objetivo de esta técnica es encontrar

problemas de usabilidad en el diseño de la interfaz de usuario, tal que puedan ser atendidos como parte de un proceso de diseño iterativo.

d. ***Pensamiento en voz alta:*** Con esta técnica se solicita de los usuarios que lleven a cabo las tareas con el producto mientras explican lo que piensan cuando trabajan con la interfaz de usuario. El pensamiento en voz alta va a permitir entender la aproximación del usuario a la interfaz y las consideraciones que mantiene en mente mientras hace uso de ella. Algunas de las ventajas de esta técnica son: mejor comprensión del modelo mental del usuario y de su interacción con el sistema, así como de la terminología utilizada por el usuario para expresar una idea o función puede ser susceptible de ser incorporada en el diseño del sistema.

(iv) 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. Los artefactos que se utilizan en esta etapa son:

a. ***Prototipo a liberar:*** Es una primera versión del software, no necesariamente tiene todas las funcionalidades implementadas, pero si es necesario que las interfaces de usuario y los comportamientos ante las interacciones de los usuarios estén bien definidas.

b. ***Protocolo de preguntas:*** En esta técnica se solicita de los participantes que realicen las tareas haciendo uso del producto y explicando lo que piensan mientras trabajan con la interfaz de usuario. De igual manera, se les formula preguntas puntuales y directas (Acosta, E., 2013); por ejemplo "¿Cómo realizaría el registro de un nuevo usuario?". De acuerdo a la respuesta y las acciones que realice el usuario se sabrá qué tan intuitiva está esa tarea en el software y servirá para tomar decisiones por parte del grupo de desarrollo.

c. ***Pruebas de aceptación:*** Las pruebas de aceptación permiten verificar si el sistema satisface o no los requerimientos funcionales y no funcionales en el contexto de la Interacción Humano-Computador, esta prueba se utiliza con la finalidad de determinar el grado de satisfacción del usuario después de hacer uso de la aplicación. Se realiza un sondeo al igual que en la etapa de Requisitos, pero el tipo de preguntas va enfocada hacia el sistema ya hecho, por ejemplo: ¿Se pueden realizar las tareas de manera sencilla?

En el siguiente capítulo se detallan las tecnologías y herramientas web seleccionadas para el desarrollo de la aplicación de éste Trabajo Especial de Grado a través del método de desarrollo de software descrito en éste capítulo.

Capítulo 3: Tecnologías y herramientas web

El propósito de este capítulo es describir las tecnologías para la implementación de la aplicación web de esta investigación. Específicamente se describen las siguientes tecnologías del lado del cliente: HTML, CSS, JavaScript, jQuery, Bootstrap y AJAX. Las tecnologías del lado del servidor descritas son: PHP y PostgreSQL. Luego, se describen las herramientas web, las cuales permiten integrar las tecnologías del cliente y del servidor, en específico el *framework* Laravel y el sistema de control de versiones Git. Finalmente se ilustra a manera de síntesis las tecnologías y herramientas descritas.

3.1 Tecnologías del lado del cliente

Son tecnologías o lenguajes que no dependen del servidor, por lo que las páginas creadas son interpretadas por el navegador y mostradas al usuario. Están enfocadas en dar estructura, estilo y animaciones a las aplicaciones web. A continuación se describe este grupo de tecnologías propuestas, específicamente: HTML, CSS, JavaScript, jQuery, Bootstrap y AJAX.

3.1.1 HTML

HTML es el lenguaje de marcado (*HyperText Markup Language* por sus siglas en inglés) basado en etiquetas para la creación de páginas web. La primera versión fue desarrollada por Tim Berners-Lee a finales de 1991 (W3C, 2012). La última versión de este lenguaje es la 5, la cual incluye nuevas características para ayudar a los desarrolladores de aplicaciones web. Se establecen una serie de nuevos elementos y atributos que reflejan el uso típico de los sitios web modernos. Fue especialmente diseñado con la intención de traer mayor contenido sin necesidad de incorporar componentes adicionales.

3.1.2 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, 2012). Esta tecnología 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. La regla está compuesta por un selector que a su vez se divide en una propiedad y un valor. La versión actual de este lenguaje es la 3 la cual incorpora nuevos módulos. Entre ellos se encuentran bordes, fondos, efectos en el texto, transformación en 2D Y 3D, animaciones, transiciones, degradados, entre otros.

3.1.3 JavaScript

JavaScript según *ECMA International* es un lenguaje de programación interpretado que incorpora dinamismo del lado del lado del cliente a través de la manipulación las etiquetas del documento HTML creando o modificando nuevo contenido. Este lenguaje tiene soporte para eventos que permiten ejecutar funciones ante alguna acción realizada por el usuario

Hay tres formas de incluir JavaScript en un documento HTML:

- Incluyéndolo en la cabecera del documento HTML, dentro de la etiqueta **`<script></script>`**
- Dentro de elementos de HTML
- En un archivo externo referenciando en el atributo **src** de la etiqueta **`<script>`**

3.1.4 jQuery

jQuery (jQuery, 2014) es una librería basada en JavaScript que permite modificar documentos HTML. Simplifica el recorrido del documento HTML, manejo de eventos y animaciones con la finalidad de optimizar el desarrollo web. jQuery toma tareas comunes las cuales requieren muchas líneas de código en JavaScript para su realización y las encapsula en métodos que solo deben ser invocados. De igual manera, permite hacer cambios de manera dinámica en los estilos del CSS.

3.1.5 Bootstrap

Bootstrap (Bootstrap, 2011) es un *framework* basado en CSS y JavaScript/jQuery para hacer diseños web adaptativos (*responsive web design*) además aprovecha los beneficios de jQuery para agregar nuevos componentes para las páginas web dinámicas. Adicionalmente *Bootstrap* permite incluir elementos de interfaz de usuario con estilos predefinidos, por ejemplo botones, alertas, mensajes de error, ventanas modales e íconos. De igual manera permite incluir contenido dinámico y de fácil uso como menú de navegación, acordeones, carruseles, entre otras.

3.1.6 AJAX

AJAX (*Asynchronous JavaScript and XML*, por sus siglas en inglés) es una técnica que involucra varias tecnologías, de manera asíncronas, en el sentido que los datos adicionales que se requieren al servidor se cargan en segundo plano sin interferir con la visualización ni el comportamiento de la página (W3C-c, 2013). Javascript es el lenguaje interpretado en el que normalmente se efectúan las funciones de llamada a AJAX mientras que el acceso a los datos se realiza mediante *XMLHttpRequest*, objeto disponible en los navegadores actuales, que actúa como una interfaz para realizar peticiones HTTP y HTTPS a servidores web. En cualquier caso, no es necesario que el contenido asíncrono esté formateado.

Los componentes de AJAX son los siguientes:

- **HTML**: Tecnología que le da la estructura a los elementos del documento
- **CSS:** Tecnología que le da el estilos a los elementos del HTML
- **Modelo de Objetos del Documento (DOM):** Accedido generalmente con JavaScript o jQuery, para mostrar e interactuar dinámicamente con la información presentada
- **El objeto XMLHttpRequest:** Para intercambiar datos de forma asíncrona con el servidor web
- **XML o JSON:** Es el formato en el que se intercambia la información entre el cliente y el servidor

Las tecnologías descritas del lado del cliente servirán para la creación de las interfaces e interacciones de usuario de la aplicación a desarrollar en este Trabajo Especial de Grado. HTML es la tecnología ideal para crear el *layout* de la aplicación. CSS permite definir atributos gráficos de los contenedores, estilos, tipos de fuentes y demás componentes de la interfaz de usuario. Las funcionalidades de jQuery y Bootstrap serán utilizadas para agregar dinamismo a la aplicación y poder incorporar elementos en la interfaz de usuario de manera sencilla. Con respecto a AJAX, esta tecnología será importante cuando se deseen realizar solicitudes al servidor para ser ejecutadas en segundo plano.

A continuación se describen las tecnologías del lado del servidor encargadas del procesamiento de datos y la interacción con los servidores de bases de datos. Estas tecnologías complementan a tecnologías del lado del cliente previamente descritas.

3.2 Tecnologías del lado del servidor

Son tecnologías o lenguajes ejecutados e interpretados por el servidor, estos se envían al cliente en un formato comprensible para éste. Se encargan del procesamiento de las peticiones de usuarios, mediante la interpretación de un *script* en el servidor web, para generar contenidos dinámicamente en formato HTML. Las tecnologías del lado del servidor a describir en este documento son: PHP y PostgreSQL.

3.2.1 PHP

Preprocesador de Hipertexto (por sus siglas en inglés *Hipertext Preprocesor*) es un lenguaje *script* interpretado de código abierto. Se ejecuta del lado del servidor y está especialmente adecuado para el desarrollo web. El principal objetivo de este lenguaje es permitir los desarrolladores la generación de páginas web dinámica de manera rápida, reduciendo el esfuerzo. PHP tiene la capacidad de conexión con la mayoría de los motores de bases de datos que se utilizan en la actualidad, tales como: MySQL, PostgreSQL, Oracle, ODBC, DB2, Microsoft SQL Server, Firebird y SQLite. Adicionalmente, puede emplearse en los principales sistemas operativos, incluyendo

Linux, muchas variantes de Unix, Microsoft Windows, Mac OS X, RISC OS y algunos otros más. Además, Admite la mayoría de servidores web de hoy en día, incluyendo Apache, IIS, y muchos otros.

Algunas de las características principales del lenguaje son las siguientes:

- Permite aplicar técnicas de programación orientada a objetos
- Soporte a varios sistemas manejadores de bases de datos
- No requiere definición de tipos de variables
- Permite el manejo de sesiones
- Permite realizar funciones de correo electrónico

Al utilizar PHP se crea una instancia del paradigma cliente-servidor, el cual está basado en el mecanismo de petición-respuesta. Lo primero que se realiza es la solicitud de una página al servidor utilizando el protocolo HTTP, el servidor recibe la petición y retorna un documento en formato PHP que es ejecutado y analizado para ser transformado en formato HTML y mostrarlo al usuario en el navegador. El proceso descrito se ilustra en la Figura 7

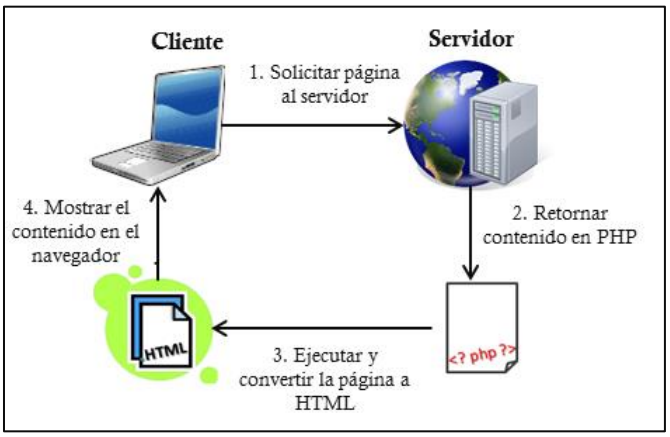


Figura 7 – Proceso de funcionamiento de las páginas con PHP

3.2.2 PostgreSQL

PostgreSQL es un sistema manejador de bases de datos relacionales. Implementa las características necesarias para competir con cualquier otro sistema manejador de base de datos comercial, con la ventaja de tener una licencia libre. La migración de bases de datos alojadas en productos comerciales a PostgreSQL se facilita gracias a que soporta ampliamente el estándar SQL. Cuenta con una serie de características a saber: herencia de tablas, diversidad de tipos de datos que incluyen arreglos, BLOB, tipos geométricos y de direcciones de red. Incluye procesamiento de transacciones, integridad referencial y procedimientos almacenados.

Con respecto a las tecnologías del lado del servidor, permitirán realizar el procesamiento y almacenamiento de la información recibida y mostrada al cliente donde se ejecute la aplicación del presente trabajo. La selección del lenguaje de programación PHP se seleccionó ya que tiene buena documentación y es ampliamente utilizado en la actualidad. Con respecto al sistema manejador de bases de datos PostgreSQL resulta una ventaja su selección ya que es de licencia libre y cumple con los estándares de las bases relacionales actuales lo cual permitirá modelar las entidades bajo un esquema conocido.

Para la integración de las tecnologías del lado del cliente y del lado del servidor descritas previamente, es necesario el uso de herramientas web, por ello, a continuación se describen dos (2) herramientas a utilizar en este trabajo: Laravel y Git.

3.3 Herramientas web

La sección que se presenta a continuación corresponde a las herramientas que permiten la integración de las tecnologías del cliente y del servidor descritas en las secciones anteriores. Laravel es un *framework* basado en PHP para crear una estructura lógica de algún proyecto basado en PHP como lenguaje del servidor y las tecnologías del lado del cliente. Git es otra herramienta descrita en esta sección la cual permite llevar el control de archivos.

3.3.1 Laravel

Laravel (Laravel *versión 4.2*, 2013) es un nuevo y poderoso *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. 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.

El patrón de arquitectura MVC divide las aplicaciones en tres niveles de abstracción. La interacción entre estos tres niveles se ilustra en la Figura 8

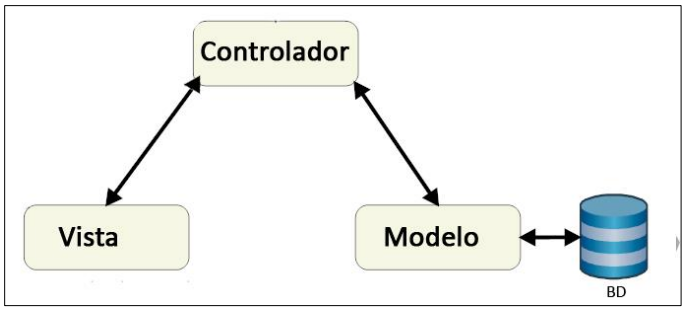


Figura 8 – Patrón MVC

Específicamente cada componente del patrón MVC se describe a continuación:

- **Modelo:** Es el encargado de acceder de forma directa a los datos actuando como intermediario con la base de datos

- **Vista:** Es la encargada de mostrar la información al usuario a través de la interfaz de usuario en el navegador web

- **Controlador:** Es el intermediario entre la vista y el modelo. Se encarga de gestionar las interacciones del usuario solicitando los datos al modelo y enviándola a la vista para a ser presentada al usuario

Siguiendo el mismo orden de ideas, Laravel propone incorporar un cuarto componente denominado **enrutador** el cual es el encargado de buscar el controlador específico dependiendo de la solicitud del usuario, éste se encarga de la gestión con el modelo para posteriormente pasar los datos a la vista para ser mostrados al usuario. Gráficamente se pueden ver los pasos al realizar una solicitud con Laravel en la Figura 9

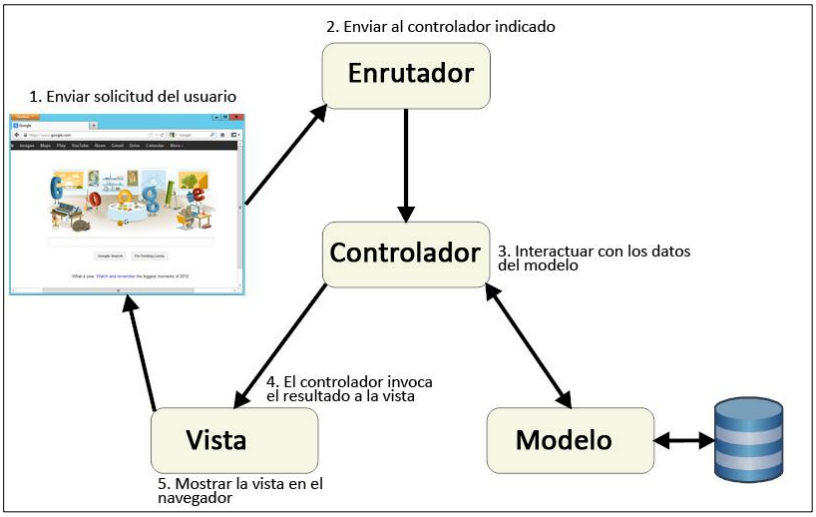


Figura 9 – Patrón MVC en Laravel

En Laravel, el modelo incluye un sistema de correspondencia de datos relacional llamado *Eloquent ORM* que facilita la creación de modelos. Un ORM (De sus siglas en inglés *Object Relational Mapper*) en PHP, es un software que permite tratar la capa de persistencia de los datos, como simples accesos a métodos de una clase u objeto en PHP. La funcionalidad interna del ORM es hacer una correspondencia entre los objetos de PHP y las tablas en la base de datos. En Laravel el ORM se basa en el patrón *Active Record* que es un objeto que incluye funciones sobre los datos como crear, leer, actualizar y eliminar (Wesley, A., 2003).

A continuación se describe la sintaxis para la creación de modelos, vistas y controladores en Laravel.

(i) Modelo

En la Figura 10 se muestra la creación del modelo *User* y el método *enumerate* para listar todos los usuarios

```
class User extends Eloquent{

    public static function enumerate(){

        return DB::table('user')->get();
    }

}
```

Figura 10 – Creación de Modelos en Laravel

El modelo se puede invocar desde el *route* como se muestra en la Figura 11

```
Route::get('/usuario/listar', function(){

    $users = User::enumerate();

    return View::make('enuerateUser')->with('users', $users);
```

Figura 11 – Llamada de modelo en Laravel

(ii) Vista

En cuanto a la vista, Laravel incluye un sistema de procesamiento de plantillas llamado *Blade*, éste sistema de plantillas permite crear código mucho más limpio en las vistas, además de incluir un sistema de caché que lo hace mucho más rápido. El sistema *Blade* de Laravel, permite una sintaxis mucho más reducida en su escritura. En la Figura 12 se muestra cómo se imprime una variable en la vista con PHP y la Figura 13 cómo es la sintaxis con *Blade* la cual es más simplificada.

```
foreach ($users as $user) {
    <p> Nombre: <?php echo $user['name'] ?> </p>

}
```

Figura 12 – Imprimir en la vista con código PHP

```
@foreach ($users as $user)
    <p> Nombre: {{ $user['name'] }} </p>
@endforeach
```

Figura 13 – Imprimir en la vista con código PHP usando Blade

(iii) Controlador

Los controladores contienen la lógica de la aplicación y permiten organizar el código en clases sin tener que escribirlo todo en las rutas. El controlador es una clase PHP que dispone de métodos públicos que son el punto de entrada final de una petición HTTP a una aplicación. Todos los controladores deben extenderse de la clase *BaseController*.

Laravel, propone en el desarrollo usar *'Routes with Closures'*. En el archivo de rutas se especifica a cuál controlador y cuál función se debe acceder según la URL especificada. Por ejemplo, para una aplicación que responda a la URL: `http://mi-aplicacion.com/usuario/listar`, a través del enrutador se especifica el nombre del controlador y la acción a la que debe acceder como se muestra en la Figura 14. Es ruta invocará al controlador mostrado en la Figura 15

```
Route::get('/usuario/listar', UserController@listar)
```

Figura 14 – Controlador usando *Routes with Closures*

```
class UsuarioController extends BaseController {
    public function __construct()
    {
        parent::__construct()
    }
    public function listar()
    {
        //punto de entrada de la peticion HTTP
        ...
    }
}
```

Figura 15 – Controlador en Laravel

Con la finalidad de integrar todos los componentes y archivos generados con el *framework* Laravel se propone utilizar el sistema de control de archivos Git, el cual es una herramienta de integración que permite manejar versiones de archivos y documentos, a continuación se explica cómo funciona.

3.3.2 Git

Git (Git-SCM, 2013) es un sistema para el control de versiones distribuido diseñado por Linus Torvalds, pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando tienen un gran número de archivos de código fuente. Git se basa en el uso de un repositorio local y un repositorio centralizado. Al utilizar Git las actualizaciones de los archivos se hacen en el repositorio local de cada usuario y adicionalmente se deben actualizar en el repositorio centralizado, lo que permite que cada integrante del equipo de desarrollo tenga el proyecto de manera local sin tener que hacer actualizaciones al repositorio central. Para actualizar el repositorio local con los archivos del repositorio centralizado se utiliza el comando *pull* y para agregar los cambios locales al repositorio centralizado se utiliza el comando *push*. Si se desea actualizar el repositorio local con nuevos cambios se utiliza el comando *commit* y *update* para llevar al entorno de trabajo los archivos del repositorio local. De manera ilustrativa, en la Figura 16 se muestran los componentes que intervienen al utilizar este sistema de control de versiones

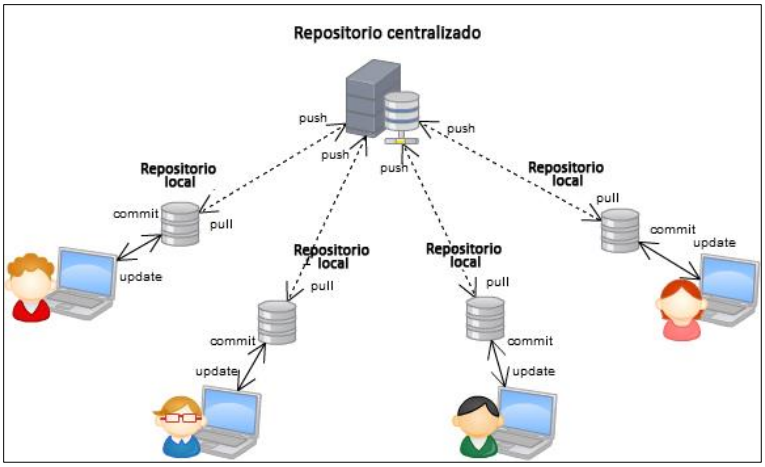


Figura 16 – Estructura de Git

Algunas de las características relevantes de Git son las siguientes (Git-SCM, 2013):

- Gestión distribuida. Git le da a cada programador una copia local del historial del desarrollo entero, y los cambios se propagan entre los repositorios locales. Los cambios se importan como ramas adicionales y pueden ser fusionados en la misma manera que se hace con la rama local
- Gestión eficiente de proyectos grandes, dada la rapidez de gestión de diferencias entre archivos, entre otras mejoras de optimización de velocidad de ejecución
- Los renombrados se trabajan basándose en similitudes entre archivos, aparte de nombres de archivos, pero no se hacen marcas explícitas de cambios de nombre con base en supuestos nombres únicos de nodos de sistema de archivos,

lo que evita posibles, y posiblemente desastrosas, coincidencias de archivos diferentes en un único nombre.

– Re-almacenamiento periódico en paquetes (archivos). Esto es relativamente eficiente para escritura de cambios y relativamente ineficiente para lectura si el reempaquetado (con base en diferencias) no ocurre cada cierto tiempo

El uso de las herramientas web descritas apoyará de manera fundamental al proceso de desarrollo de software. A través del *framework* Laravel se desarrollará una aplicación modular ya que está basada en el patrón de arquitectura MVC el cual permitirá tener separados los componentes del software. Adicionalmente el uso este *framework* es una ventaja ya que se está utilizando actualmente por un amplio número de desarrollares web lo cual facilita la cantidad de documentación que se pueda conseguir. Con respecto a la herramienta Git, permitirá la sincronización e integración del código implementado. Este sistema resulta conveniente ya que se necesita un mecanismo que permita a varios integrantes trabajar de forma remota utilizando una herramienta de control de versiones de código.

La Figura 17 resume las tecnologías y herramientas descritas en el presente Capítulo a través del paradigma cliente-servidor. Del lado del cliente HTML, CSS, JavaScript, jQuery, AJAX y Bootstrap. Del lado del servidor PHP y PostgreSQL. En el medio de ambos grupos de tecnologías las herramientas web descritas: el *framework* Laravel y Git.

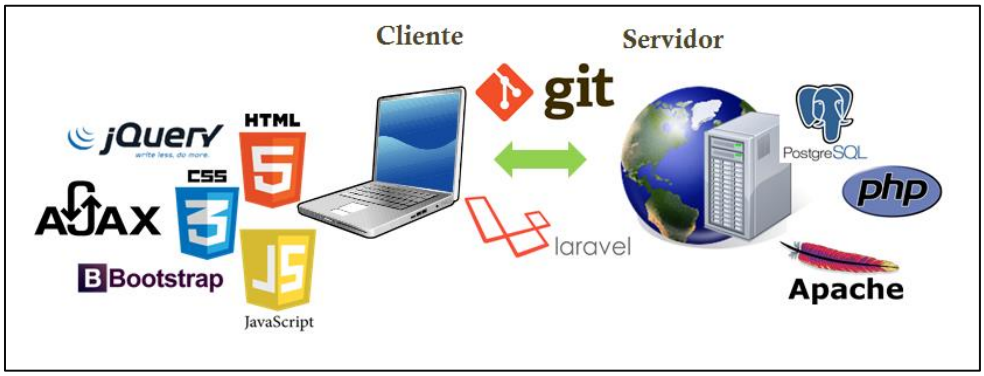


Figura 17 – Ambiente de desarrollo

Una vez descritas las tecnologías y herramientas seleccionadas para el desarrollo de la aplicación del presente trabajo, en el siguiente capítulo se detalla el proceso de desarrollo de software involucrado para dar cumplimiento al objetivo general de esta investigación.

Capítulo 4: Desarrollo de la aplicación web ProAgil

En este capítulo se presenta la experiencia del desarrollo de la aplicación web ProAgil haciendo uso de los aspectos conceptuales descritos, las tecnologías propuestas y el método de desarrollo de software AgilUs.

Las iteraciones utilizadas agrupan los requerimientos de acuerdo a características similares entre ellos. Específicamente, se realizaron cinco (5) iteraciones a saber: análisis global, módulo de proyectos y actividades, módulo generador de formularios, módulo de componentes gráficos y la iteración de pruebas. En la iteración 1, se realiza el recorrido por las etapas de requisitos y análisis con la finalidad de definir el alcance y requerimientos del proyecto. Para las iteraciones intermedias, específicamente la 2, 3 y 4 se instancia el método recorriendo las etapas de requerimientos, análisis y prototipaje, la cual es la que tiene mayor importancia ya que es la etapa donde se desarrollan los prototipos ejecutables de los requerimientos determinados en la iteración 1. Finalmente, en la iteración 5 se describe la experiencia de utilizar las pruebas de aceptación las cuales permitieron validar la aceptación del software. La Figura 18 ilustra las iteraciones utilizadas a través del método AgilUs. A continuación se detalla cada iteración para la construcción de la aplicación web ProAgil especificando los artefactos utilizados.

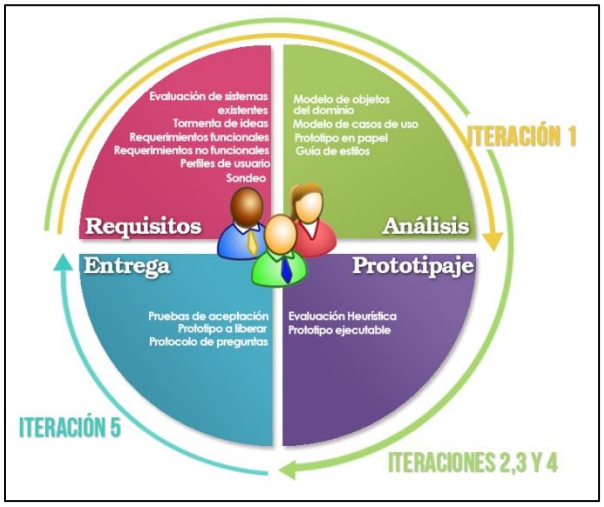


Figura 18 – Iteraciones para la construcción de la aplicación web ProAgil

4.1 Iteración 1: Análisis global

Esta iteración tiene como objetivo determinar el alcance general de aplicación web a desarrollar a través del uso de actividades y técnicas propuestas en AgilUs en sus primeras dos etapas: requisitos y análisis. Es importante resaltar que en las primeras etapas de desarrollo se deben realizar actividades que permitan determinar los requerimientos de los usuarios a través de técnicas de indagación. Los requerimientos determinados se formalizan en la etapa de análisis a través de diferentes artefactos los cuales sirven de comunicación entre el grupo de desarrollo. En esta iteración, en la etapa de requisitos se realiza una tormenta de ideas, se aplica un sondeo a personas que han utilizado artefactos de evaluación de usabilidad, se determinan los perfiles de usuario y se listas los requerimientos funcionales y no funcionales que conforman la aplicación. En la etapa de análisis, los artefactos para la formalización de los requisitos son: casos de uso, objetos del dominio, guía de estilos y prototipo en papel. Los artefactos seleccionados en la etapa de requisitos y análisis permiten dar cumplimiento al objetivo para esta iteración. A continuación se describen los artefactos y resultados obtenidos en cada una de las etapas.

4.1.1 Etapa de Requisitos

Como propone el método de desarrollo de software AgilUs, en las primeras etapas de desarrollo es necesario aplicar técnicas de indagación a los usuarios para definir los requerimientos funcionales y no funcionales, de igual manera es importante determinar la audiencia a la cual va dirigida la aplicación. Por esa razón, en esta iteración se utilizan los siguientes artefactos: tormenta de ideas y sondeo para determinar los perfiles de usuario y requerimientos funcionales y no funcionales para la aplicación web ProAgil.

(i) Tormenta de ideas

El primer artefacto generado para el desarrollo de la aplicación web fue la tormenta de ideas. Gracias a esta técnica se obtuvieron palabras claves relacionadas a la aplicación. Para su realización se hizo una recopilación de las grabaciones de las reuniones del grupo de desarrollo en donde se discuten los tópicos relacionados a la investigación tomando aquellas en las que se hacía énfasis. Para representar el resultado obtenido gráficamente se sintetiza las palabras claves resultantes en la Figura 19

Figura 19 – Tormenta de ideas

Tal como se puede observar en la nube de palabras generada algunas palabras resaltan debido a su importancia con el tópico del proyecto, a continuación se detalla cada una de ellas:

- Proyecto: Se propone que el sistema debe proveer a los usuarios la facilidad de crear proyectos con la finalidad de establecer los artefactos y actividades que estarán presentes en el proyecto. De igual manera, poder invitar a otros usuarios a un proyecto para conformar el equipo de desarrollo de software.

- Actividades: En un proyecto se pueden crear actividades. Es importante que el sistema provea la facilidad de guardar información relacionada a la actividad y asignarla a algún miembro del equipo.

- Estado: Cada actividad asociada a un proyecto debe tener un estado para saber el avance del proyecto, entre los estados que surgieron en la tormenta de ideas se pueden citar: actividad no realizada, actividad en proceso y actividad realizada.

- Artefactos: Adicionalmente a las actividades se propone asociar artefactos a los proyectos los cuales permiten realizar evaluaciones de usabilidad en los proyectos.

- Evaluación Usabilidad: Son los tipos de artefactos que proveen al equipo de desarrollo la facilidad de aplicar técnicas para lograr la usabilidad de los proyectos de software, antes, durante y después de su desarrollo.

La tormenta de ideas generada permitió concretar el contexto de trabajo para los siguientes artefactos a utilizar, entre ellos el sondeo el cual fue realizado a estudiantes

que han utilizado artefactos para la evaluación de usabilidad y se presentan sus resultados a continuación.

(ii) Sondeo

Este artefacto fue aplicado a sesenta y tres (63) personas pertenecientes a la Facultad de Ciencias de la Universidad Central de Venezuela, los cuales son estudiantes y/o preparadores, con experiencia en el uso de técnicas para la evaluación de usabilidad. Este sondeo tiene como objetivo identificar funcionalidades y características de software a ser incorporadas en la aplicación web a desarrollar. Está conformado por diez (10) preguntas con respuestas sugeridas las cuales se detallan a continuación.

Pregunta 1: ¿Qué tan importante considera Ud. llevar las tareas y actividades del proceso de desarrollo de software a través de una herramienta informática?

En esta pregunta las respuestas con mayor porcentaje son: Muy importante con 56% y la opción importante con 37%. Esto permite concluir que de manera general los encuestados considera importante llevar las actividades del proceso de desarrollo de software a través de una herramienta informática. El resultado de esta pregunta se ilustra en el Gráfico 1

Gráfico 1 – Pregunta 1: ¿Qué tan importante considera Ud. llevar las tareas y actividades del proceso de desarrollo de software a través de una herramienta informática?

Pregunta 2: ¿Qué tipo de notificación preferiría cuando se le asigne una tarea que deba realizar?

Con relación a la pregunta 2, el 63% prefiere las notificaciones a través de correo electrónico y el 37% a través de la aplicación (ver Gráfico 2). Posterior a esta pregunta se decide implementar las notificaciones de asignación de tareas a través de correo.

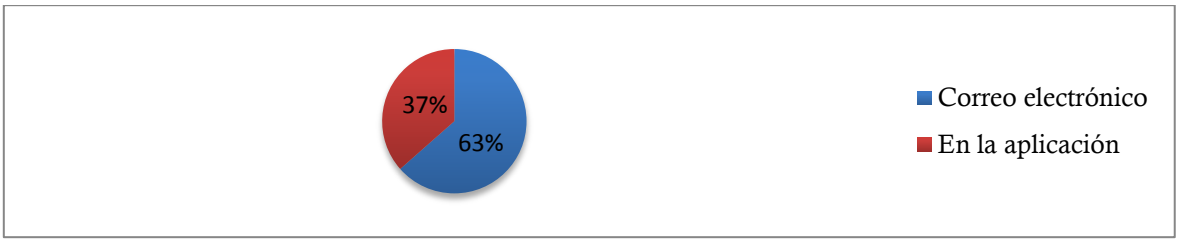


Gráfico 2 – ¿Qué tipo de notificación preferiría cuando se le asigne una tarea que deba realizar?

Pregunta 3: ¿Cuál(es) de las siguientes acciones sobre una tarea considera útil?

En la pregunta 3 las acciones sobre una tarea más relevante (ver Gráfico 3) según los encuestados son: Cambiar estado y comentar tarea. Posterior a esto, ambas funcionalidades son tomadas en cuenta para su implementación.

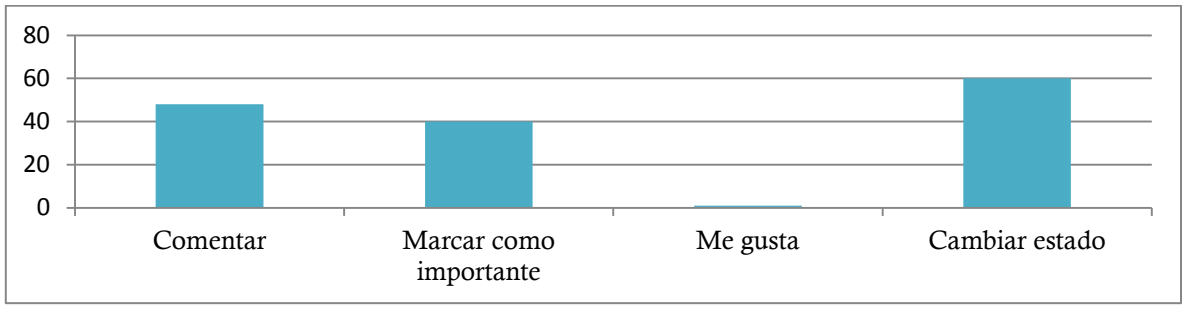


Gráfico 3 – Pregunta 3: ¿Cuál(es) de las siguientes acciones sobre una tarea considera útil?

Pregunta 4: ¿Cuál(es) de las siguientes opciones considera debe ser un filtro para mostrar las tareas de un proyecto?

En esta pregunta se quiso conocer la opinión de los encuestados con respecto al uso de filtros para mostrar las tareas de un proyecto. Las respuestas con mayores coincidencias se ilustran en el Gráfico 4, las cuales son: Filtrar por estado y filtrar por tipo de tarea. Por esta razón, es necesario desarrollar un mecanismo que permita crear tipos de tareas y cambiar el estado de éstas.

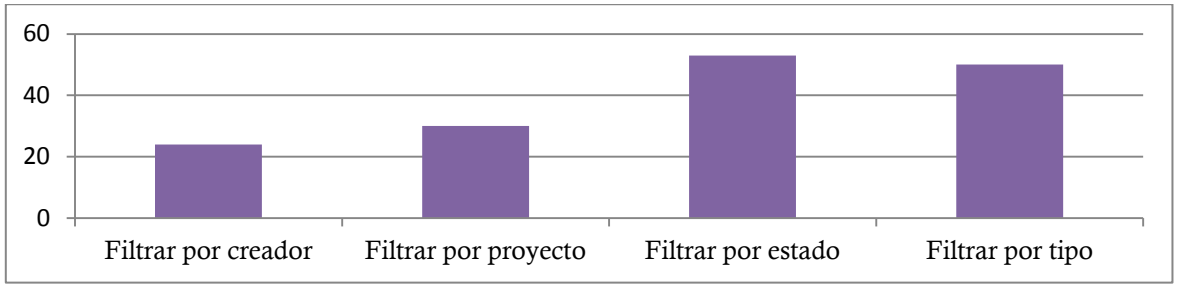


Gráfico 4 – Pregunta 4: ¿Cuál(es) de las siguientes opciones considera debe ser un filtro para mostrar las tareas de un proyecto?

Pregunta 5: ¿Cuál(es) de los siguientes artefactos de evaluación de usabilidad considera que da(n) mejor apoyo al proceso de desarrollo de software?

El objetivo de esta pregunta es identificar si existe interés en el uso de técnicas de evaluación de usabilidad con la finalidad de priorizar los artefactos. Según los encuestados, el artefacto que da mayor apoyo al proceso de desarrollo de software es el análisis de sistemas existentes (ver Gráfico 5). A pesar de que la guía de estilos obtuvo el menor grado de importancia se considera que es un artefacto que da buen apoyo al proceso d desarrollo de software.

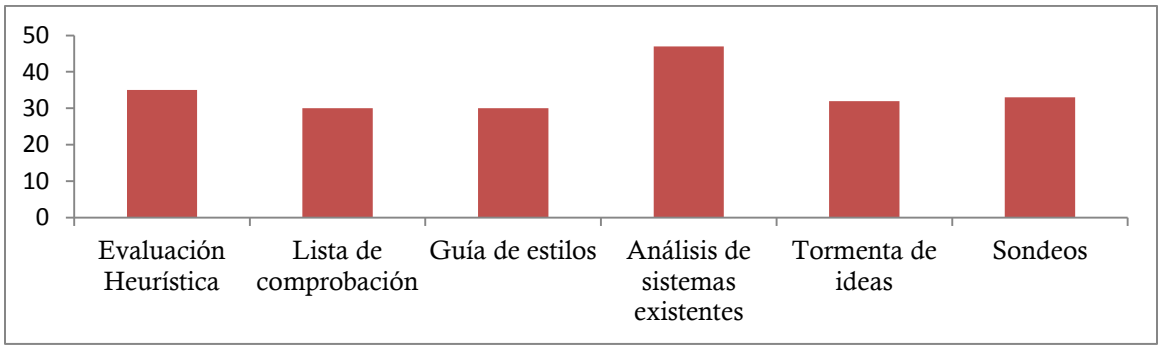


Gráfico 5 – Pregunta 5: ¿Cuál(es) de los siguientes artefactos de evaluación de usabilidad considera que da(n) mejor apoyo al proceso de desarrollo de software?

Pregunta 6: De tener automatizados los artefactos de la pregunta anterior ¿Qué información considera relevante a tomar en cuenta?

En la pregunta 6, la opción Descripción fue la que obtuvo más respuestas de los encuestados. Por esta razón será el atributo que será la información que se mostrará al crear un artefacto en la aplicación. El resultado de esta pregunta se ilustra en el Gráfico 6

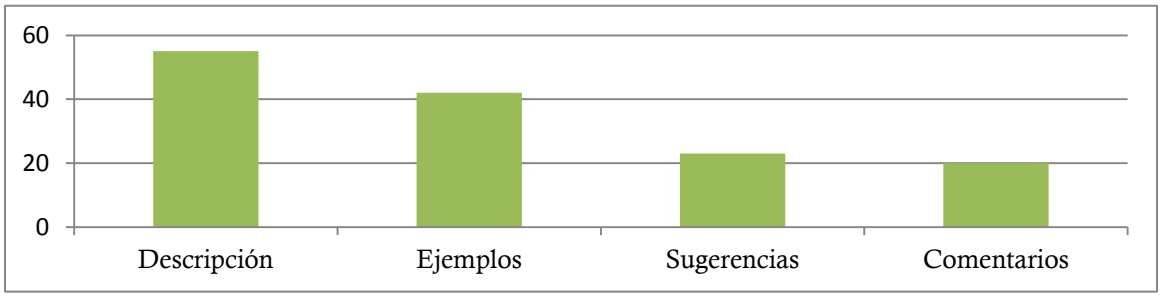


Gráfico 6 – Pregunta 6: De tener automatizados los artefactos de la pregunta anterior ¿Qué información considera relevante a tomar en cuenta?

Pregunta 7: ¿En cuál(es) redes sociales le gustaría compartir alguna de las actividades que esté realizando en el sistema?

Con la finalidad de apoyar el uso de las redes sociales la pregunta 7 hace referencia a aspectos de sociabilidad. Las redes sociales que se considera más importante son Twitter, seguidamente LinkedIn y Facebook (ver Gráfico 7)

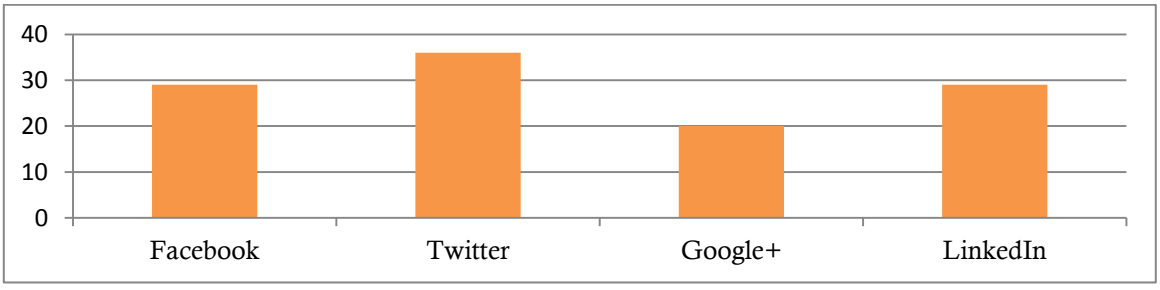


Gráfico 7 – Pregunta 7: ¿En cuál(es) redes sociales le gustaría compartir alguna de las actividades que esté realizando en el sistema?

Pregunta 8: ¿En qué lugar de la interfaz de usuario preferiría ver la lista de proyectos en lo que está trabajando?

La pregunta 8 es realizada con el objetivo de tener un punto de partida con respecto a la interfaz de usuario especifica que el 66% de los encuestados prefiere el listado de los proyectos en el lateral izquierdo (ver Gráfico 8)

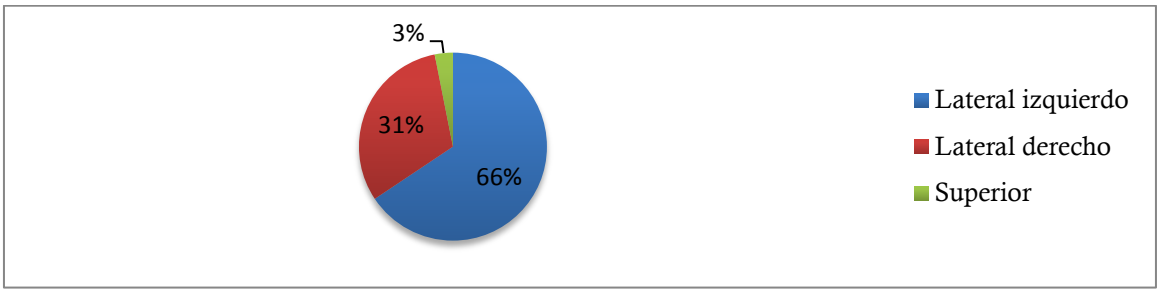


Gráfico 8 – Pregunta 8: ¿En qué lugar de la interfaz de usuario preferiría ver la lista de proyectos en lo que está trabajando?

Pregunta 9: ¿Cuál de los siguientes temas le parece más atractivo?

Esta pregunta también está enfocada en aspectos de la interfaz de usuario. En la Tabla 3 se muestran los temas propuestos en el sondeo. La opción D fue la que tuvo más aceptación por parte de los encuestados, por lo que se seleccionó como base para el tema del proyecto.

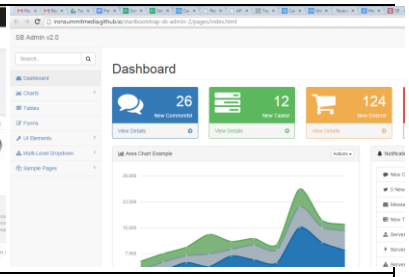
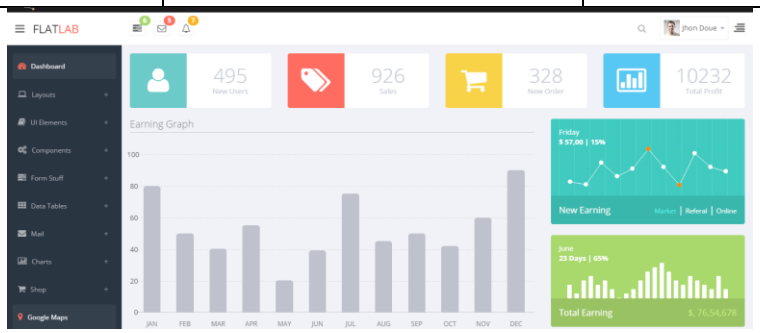
		
(A) 18 personas	(B) 11 personas	(C) 9 personas
		
(D) 25 personas		

Tabla 3 – Pregunta 9: ¿Cuál de los siguientes temas le parece más atractivo?

Pregunta 10: ¿Cuál(es) de las siguientes cualidades de software considera que deben estar presentes en una herramienta que automatice las actividades del proceso de desarrollo de software?

La pregunta 10 se realizó con el objetivo de saber la opinión con respecto a los requerimientos no funcionales del software a desarrollar. Las cualidades de software más relevantes (ver Gráfico 9) según los usuarios son: Usabilidad, seguridad y confiabilidad.

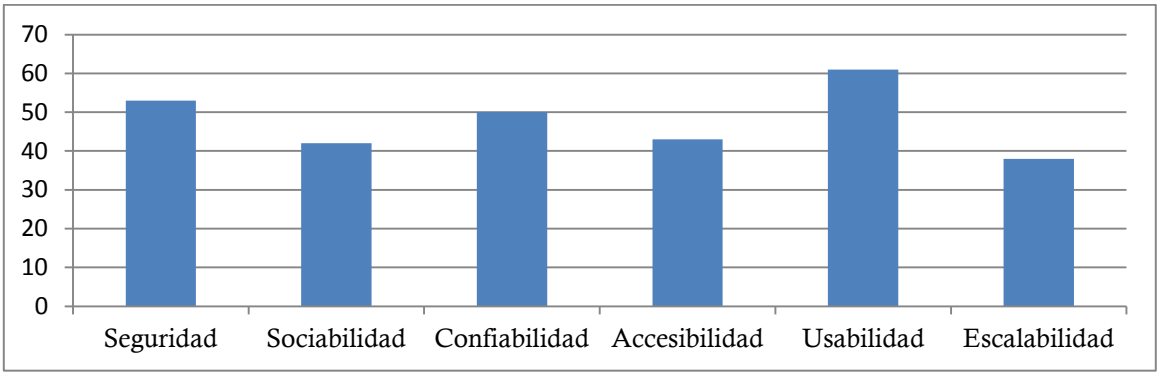


Gráfico 9 – Pregunta 10: ¿Cuál(es) de las siguientes cualidades de software considera que deben estar presentes en una herramienta que automatice las actividades del proceso de desarrollo de software?

Con el análisis del sondeo se pudo definir aspectos importantes a partir de las respuestas obtenidas, específicamente requerimientos funcionales y no funcionales de la aplicación. Tomando en cuenta los resultados de la encuesta uno de los requerimientos funcionales identificado es el envío de notificaciones vía correo electrónico de la asignación de una actividad. Otro requerimiento funcional es que el usuario dependiendo de sus necesidades pueda realizar la configuración de los artefactos que desee incorporar en su proyecto.

Se considera también la posibilidad de crear, filtrar y modificar el estado de actividades. Esto permite fomentar el orden en las actividades de los proyectos.

Otro aspecto considerado a través del análisis de las respuestas del sondeo, es la interfaz de usuario. Se pudo seleccionar el tema (*layout*) de la aplicación el cual permite tener una idea inicial la apariencia de la interfaz de usuario.

Con respecto a los requerimientos no funcionales se le da importancia a la usabilidad de manera prioritaria también se toma en cuenta aspectos seguridad y confiabilidad. Adicionalmente se toman en cuenta otros requerimientos no funcionales.

Una vez analizado el sondeo, se realiza la definición de los perfiles de usuario el cual contempla la audiencia a la que está dirigida la aplicación web ProAgil.

(iii) Perfiles de usuario

Teniendo en cuenta que AgilUs es un método de desarrollo de software centrado en el usuario, es importante definir los perfiles de usuario al cual está dirigida la aplicación desarrollada. ProAgil es una herramienta basada en dos tipos de roles a saber: líder y colaborador. Por esta razón se deben definir dos perfiles de usuario descritos a continuación.

El usuario *líder* es una persona encargada de coordinar y planificar proyectos de software. Debe tener experiencia utilizando métodos de desarrollo de software ágil y realización de artefactos de evaluación de usabilidad, específicamente los artefactos: tormenta de ideas, sondeos, análisis de sistemas existentes, evaluación heurística, lista de comprobación y guía de estilos. De igual manera, debe tener experiencia en el uso de herramientas informáticas para gestión de proyectos que permitan crear y asignar actividades a miembros de su equipo de desarrollo.

El usuario *colaborador* es una persona con conocimientos de métodos de desarrollo de software y artefactos de evaluación de usabilidad específicamente los artefactos: tormenta de ideas, sondeos, análisis de sistemas existentes, evaluación heurística, lista de comprobación y guía de estilos. El colaborador debe ser capaz de poder aplicar cualquiera de los artefactos mencionados siguiendo las indicaciones del líder.

Los siguientes artefactos a generar como parte de la etapa de Requisitos de la primera iteración son los requerimientos funcionales y no funcionales.

(iv) Requerimientos funcionales y no funcionales

La definición de los requerimientos funcionales permite establecer una primera versión de las acciones que puede realizar el usuario en la aplicación. Este artefacto sirvió de entrada para la siguiente etapa, facilitando el modelado de los diagramas en la etapa de análisis. A continuación se listan los requerimientos funcionales determinados, parte de ellos se definieron a través de los resultados alcanzados en la tormenta de ideas y el análisis del sondeo, otros son requerimientos que se cumplen en la mayoría de las aplicaciones web.

- Iniciar de sesión en el sistema. Esta funcionalidad le permite a los usuarios autenticarse para utilizar el sistema
- Realizar registro. Permite a los usuarios crear una cuenta en la aplicación web para poder hacer uso de la misma
- Crear proyecto. Facilita la creación de un proyecto en la aplicación al usuario líder
- Consultar lista de proyectos. Provee la facilidad de consultar los proyectos creados y en los que colabora a los usuarios registrados
- Gestionar actividades. Permite crear, editar y eliminar actividades asociadas a los proyectos de la aplicación web. Adicionalmente gestionar una actividad

también implica cambiar el estado de la misma, es decir, especificar si la actividad está en proceso de realización o si ya está terminada
- Gestionar categoría de actividades. Permite crear, editar y eliminar categoría de actividades asociadas a los proyectos de la aplicación web
- Filtrar actividades. Permite a los usuarios filtrar la lista de actividades por estado o por categoría de actividad
- Generar formulario. Provee la facilidad de crear dinámicamente formularios relacionados a artefactos de evaluación de usabilidad, específicamente los sondeos, la lista de comprobación análisis de sistemas existentes y evaluación heurística
- Guardar y consultar guía de estilos. Permite guardar los elementos gráficos asociados a un proyecto para ser consultados por el equipo de desarrollo cuando se desee
- Generar nube de *tags*. Provee la facilidad de generar una nube de *tags* a partir de una lista de palabras establecidas por el usuario
- Consultar artefactos: Permite a los usuarios consultar los artefactos de evaluación de usabilidad asociados a un proyecto

Por otro lado, los requerimientos no funcionales describen aspectos del sistema que son visibles por el usuario, que no tienen una relación directa con el comportamiento funcional del sistema. A continuación se plantean los requerimientos no funcionales a tomar en cuenta para el desarrollo de la aplicación web, algunos de ellos determinados en el análisis del sondeo y otros propuestos:

- Usabilidad: una de las características más importantes que apuntan a la calidad del software. El software será de fácil uso, ya que las interfaces de usuario son diseñadas pensando en las necesidades de los usuarios. El software contará con metáforas adecuadas y textos de ayuda que faciliten la interacción con la aplicación, por lo que el tiempo de aprendizaje será mínimo
- Seguridad: el acceso al sistema va estar restringido. Existirá un inicio de sesión de los usuarios que permitirá ingresar solo las personas registradas en la aplicación. De igual manera se podrá consultar la información asociada a los proyectos una vez se haya iniciado sesión.
- Confiabilidad: el software será confiable, ya que asegura un nivel de funcionamiento adecuado bajo condiciones normales.
- Sociabilidad: es la capacidad de un software de compartir con otros actores a través de espacios colaborativos o redes sociales, esto se logra en esta aplicación web al permitirle al usuario compartir las preguntas de un sondeo a través de las redes sociales Facebook, Twitter y LinkedIn. También se fomenta esta característica permitiéndole a los usuarios realizar comentarios en las actividades asociadas a los proyectos.
- Mantenibilidad: Hace referencia al conjunto de atributos relacionados con la facilidad de extender, modificar o corregir errores en un sistema software.

- Escalabilidad: Se debe poder agregar en el software funcionalidades, requerimientos o módulos sin perder la calidad de la versión actual.

Una vez aplicados los artefactos de indagación tormenta de ideas y sondeo, se determinaron los requerimientos funcionales y no funcionales. Ahora, se modelan los requerimientos determinados en esta etapa a través del uso de diversos artefactos en la etapa de análisis.

4.1.2 Etapa de Análisis

En esta etapa se realiza el análisis de los requerimientos obtenidos en la etapa de requisitos. Los artefactos utilizados en esta etapa para el modelado son: diagramas de casos de uso y diagrama de objetos del domino; los cuales permiten representar las funcionalidades y entidades de la aplicación respectivamente. Adicionalmente, con la finalidad de definir los componentes gráficos de la aplicación el artefacto oportuno a utilizar fue la guía de estilos. Asimismo, para definir un prototipo de baja fidelidad, se realizó el prototipo en papel correspondiente a la interfaz de usuario inicio de la aplicación. A continuación se describen los resultados de utilizar cada uno de los artefactos mencionados.

(i) Casos de uso

Se utiliza este artefacto ya que permite modelar los requerimientos funcionales de la aplicación determinados en la etapa de requisitos, especificando cuáles son las funcionalidades principales y cuáles son las funcionalidades que se extienden de éstas. De igual manera el modelo de caos de uso, sirve de guía para dar seguimiento de las funcionalidades desarrolladas en la aplicación. A continuación, en la Figura 20 y la Figura 21 se ilustra el diagrama de casos de uso (CU) en los niveles 1 y 2 respectivamente. Adicionalmente se describe cada caso de uso haciendo énfasis en la interacción que tiene el usuario con el sistema.

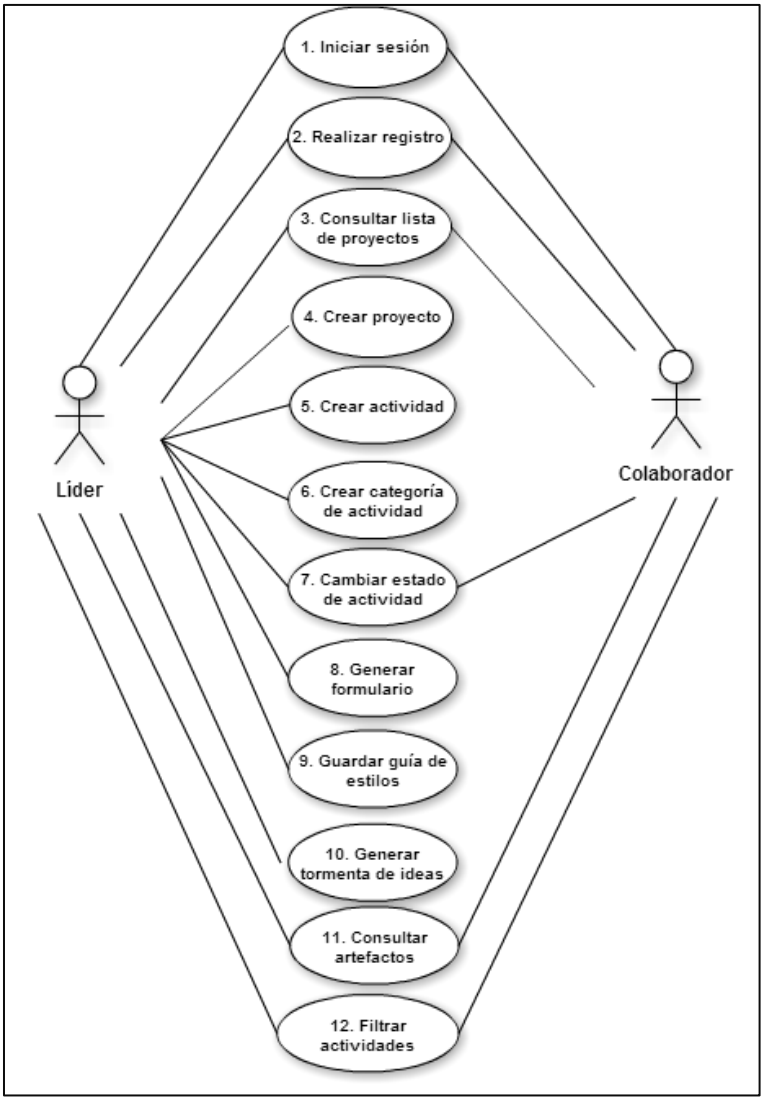


Figura 20 – Casos de uso: Nivel 1

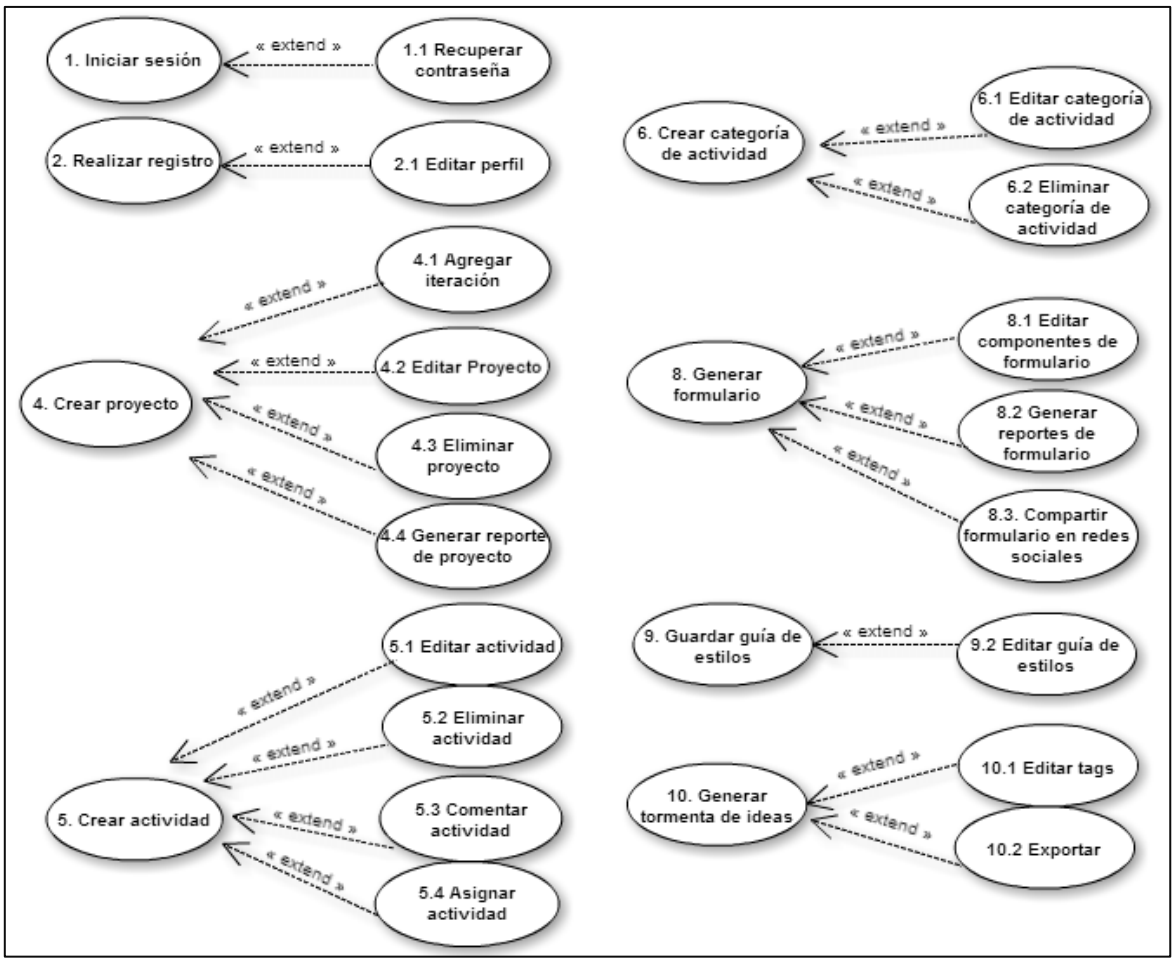


Figura 21 – Casos de uso: Nivel 2

Para la construcción del sistema se establecieron dos tipos de actores correspondientes a los perfiles de usuarios de la etapa de requisitos: líder y colaborador. El actor *líder* crea, gestiona y asigna artefactos y actividades en la aplicación. El actor *colaborador* es a quien se le asignan las actividades y puede consultar y aplicar los artefactos del proyecto. A continuación se describen los casos de uso que provee la aplicación haciendo énfasis en la interacción que tiene el usuario con el sistema:

1. Iniciar Sesión: permite a los actores ingresar al sistema y hacer uso del mismo. El usuario indica su correo electrónico y contraseña y si son correctos los datos suministrados se le da acceso, de lo contrario se muestra un mensaje de error indicando que alguno de los campos está errado. Como extensión de este caso de uso se tiene el caso de uso (CU) 1.1 el cual permite al usuario indicar su correo electrónico para recuperar su contraseña a través de su correo.

2. Realizar registro: facilita a los actores crear una cuenta de usuario en la aplicación. Para dar cumplimiento a ésta funcionalidad el usuario indica sus

datos: nombre, apellido, correo electrónico, contraseña y un avatar para su perfil. Una vez que se registra al usuario se envía un correo notificándole sobre su registro en la aplicación. Extendiendo la funcionalidad de este caso de uso, el CU 2.1 permite al usuario editar los datos de su perfil.

3. Consultar lista de proyectos: permite a los actores consultar los proyectos que ha creado como líder y en los que colabora

4. Crear proyecto: permite al actor líder crear un proyecto en el sistema. Para dar cumplimiento a esta funcionalidad el usuario debe indicar: nombre de proyecto, objetivos y cliente al cual pertenece. Para dar extensión la creación de proyectos se proveen las funcionalidades agregar iteración (CU 4.1), editar proyecto (CU 4.2), eliminar proyecto (CU 4.3) y generar reporte (CU 4.4). Al agregar una iteración al proyecto el usuario especifica los artefactos de evaluación a incorporar y los colaboradores involucrados. Al editar un proyecto se puede cambiar la información del proyecto o los artefactos del mismo. La funcionalidad eliminar proyecto permite borrar la información, artefactos y actividades asociadas a un proyecto. Finalmente, se puede crear un reporte de proyecto en donde se especifiquen las iteraciones y actividades del mismo.

5. Crear actividad: facilita al actor líder crear una actividad asociada a un proyecto. El usuario indica nombre de actividad, breve descripción, fecha tope y categoría de actividad. Esta funcionalidad es extendida por los siguientes casos de uso: Editar actividad (CU 5.1), eliminar actividad (CU 5.2), comentar actividad (CU 5.3) y asignar actividad (CU 5.4). La edición de actividad permite modificar los atributos de una actividad. Eliminar actividad permite desasociar una actividad a un proyecto y asignar actividad permite asociar una actividad a un usuario, adicionalmente se le notifica a través de correo electrónico sobre la asignación de la misma.

6. Crear categoría de actividad: provee la facilidad de crear *tags* para clasificar las actividades asociadas a un proyecto. Para extender la creación de categoría de actividad se presentan los casos de uso editar categoría de actividad (CU 6.1) y eliminar categoría de actividad (CU 6.2). Editar categoría permite cambiar el nombre y eliminar borra la categoría asociada.

7. Cambiar estado de actividad: permite a los actores involucrados cambiar el estado de una actividad. Las actividades pueden tener tres estados: sin empezar, en proceso y completada.

8. Generar formulario: a través de un formulario será posible generar otros formularios dinámicamente, con cierta cantidad de preguntas ya sean de tipo

cerradas o abiertas. Estos formularios pueden ser de tipo sondeo, lista de comprobación, análisis de sistemas existentes o evaluación heurística. Como extensión de este caso de uso se tiene: editar componentes del formulario (CU 8.1), generar reportes del formulario (CU 8.2) y compartir formulario en redes sociales (CU 8.3). Editar componentes del formulario permitirá agregar, editar o eliminar una pregunta u opciones de respuestas del formulario. El CU 8.2 provee la facilidad de generar gráficos que reflejen los resultados una vez aplicado el formulario de tipo sondeo. Los formularios resultantes de tipo sondeo podrán ser compartidos en las redes sociales como Twitter, Facebook o LinkedIn.

9. Guardar guía de estilos: permite guardar elementos de la guía de estilo del proyecto que se ha creado. Se podrá agregar a la guía de estilo elementos como: el logo, la paleta de colores, estilo de botones, iconografía entre otros elementos por definir. El caso de uso que se extienden es el 9.1 que permitirá editar guía de estilo, es decir, será posible agregar, eliminar o modificar algún elemento de la guía de estilo previamente creada.

10. Generar tormenta de ideas: esta funcionalidad permite generar una nube de *tags* a partir de una lista de palabras escogidas por el usuario. De este caso de uso se extiende editar *tags* (CU 10.1), que permite es agregar, editar o eliminar una palabra de la lista. Otro punto de extensión es el CU 10.2 el cual le permite al usuario exportar la tormenta de ideas generada

11. Consultar artefactos: provee a los actores, tanto colaboradores como líderes la posibilidad de consultar los artefactos ya creados con la posibilidad de poder exportarlos

12. Filtrar actividades: consiste en filtrar una lista de actividades dependiendo del estado en el que se encuentre la actividad, permite que los actores solo tengas a su disposición las tareas de un estado específico.

Con el modelado de los requerimientos a través del diagrama de casos de uso se establecen con mayor formalidad las funcionalidades a desarrollar. Continuando con la fase de análisis el siguiente artefacto de modelado utilizado es el modelo de objetos del dominio, el cual describe los objetos involucrados en el sistema que tienen representación en la interfaz de usuario.

(ii) Objetos del dominio

El diagrama de objetos de dominio describe los objetos que existen en el contexto del sistema y se representan en la interfaz de usuario. Este diagrama permite abstraerse de las entidades a tomar en cuenta para la aplicación web. En la Figura 22 se ilustra el diagrama de objetos de dominio de la aplicación ProAgil especificando los objetos y las relaciones entre ellos.

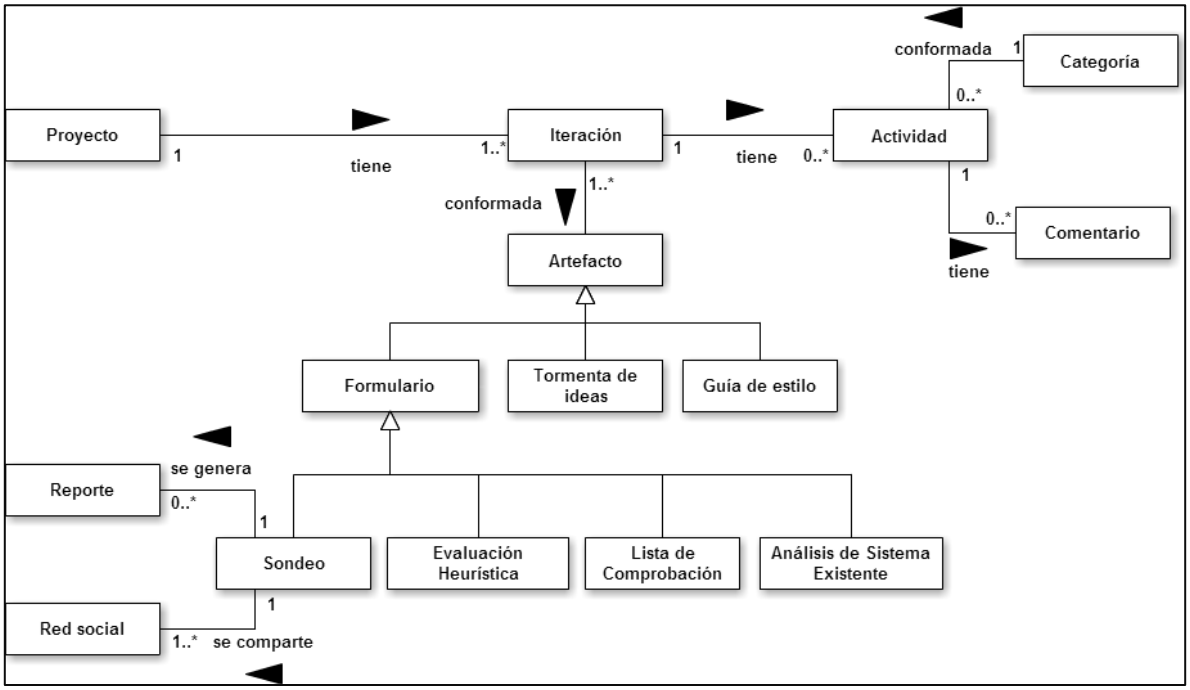


Figura 22 – Objetos del dominio

La figura 22 ilustra los objetos más relevantes de la aplicación ProAgil. Un proyecto está compuesto por iteraciones las cuales están conformadas por actividades y artefactos. Las actividades pueden pertenecer a una categoría y tener comentarios. Un artefacto generaliza los siguientes objetos: formulario, tormenta de ideas y guía de estilos. El objeto formulario a su vez puede ser de tipo: sondeo, evaluación heurística, lista de comprobación y análisis de sistema existente. El sondeo puede generar un reporte de resultados y ser compartido en alguna red social.

A través de este modelo se puede determinar los componentes a representar en la interfaz de usuario, para ello se platea el uso de metáforas a través de iconos con la finalidad de desarrollar una aplicación intuitiva.

Una vez modelado los requerimientos y objetos del sistema, a continuación se presenta la guía de estilos para la aplicación , la cual resume los elementos gráficos a tomar en cuenta para la interfaz de usuario.

(iii) Guía de estilos

La guía de estilos resume los aspectos gráficos de la aplicación ProAgil, específicamente: logo del sistema, paleta de colores, tipografía, estilos de menú, estilos de formulario e iconografía. El uso de este artefacto sirve para establecer los lineamientos gráficos a utilizar en la interfaces de usuario. De igual manera, se utiliza como un material de consulta al momento de incorporar un nuevo componente a la interfaz de usuario. En la Tabla 4 se puede apreciar la guía de estilos propuesta.

Logo

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

Botones

Estilos de interacción de forma

Iconografía

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


Metáforas para los artefactos de evaluación de usabilidad

Tabla 4 – Guía de estilos

(iv) Prototipo en papel

En esta iteración se realiza el prototipo en papel correspondiente a la interfaz de usuario de inicio en la aplicación. El primer planteamiento que surge es presentarle al usuario una interfaz de usuario en donde se muestre sus proyectos y los proyectos en los que es colaborador. En la Figura 23 se ilustra el prototipo generado.

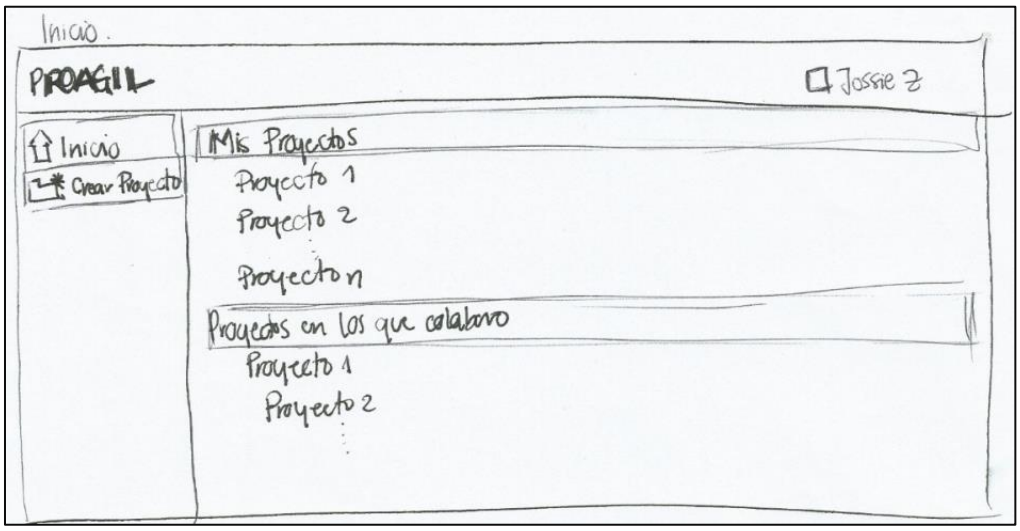


Figura 23 – Prototipo en papel: Inicio

Después de aplicar los artefactos seleccionados para la iteración 1, de análisis global es posible determinar el alcance general de aplicación web ProAgil. El artefacto generado que se considera más importante en esta iteración es el listado de los requerimientos funcionales los cuales se desarrollan en las siguientes iteraciones.

4.2 Iteración 2: Módulo de proyectos y actividades

Esta iteración agrupa los requerimientos relacionados a la creación de proyectos y actividades. El objetivo es desarrollar un componente de software que permita configurar proyetos y actividades en el sistema. Para dar cumplimiento al objetivo planteado se realizan artefactos correspondientes a las etapas de requisitos, análisis y prototipaje del método. En la etapa de requisitos se realiza el análisis de sistemas existentes de herramientas que permitan gestionar proyectos y actividades, especificamente *Freedcamp* y *Asana* con la finalidad de indagar cómo se realiza la gestión de proyectos a través de las paltaformas mencionadas. En la etapa de análisis

se realizan prototipos en papel correspondientes a las interfaces de usuario para las funcionalidades de crear y detalle de proyecto. En la etapa de prototipaje se realiza la evaluación heurística del prototipo ejecutable del módulo creado. A continuación se detalla cada uno de los artefactos mencionados.

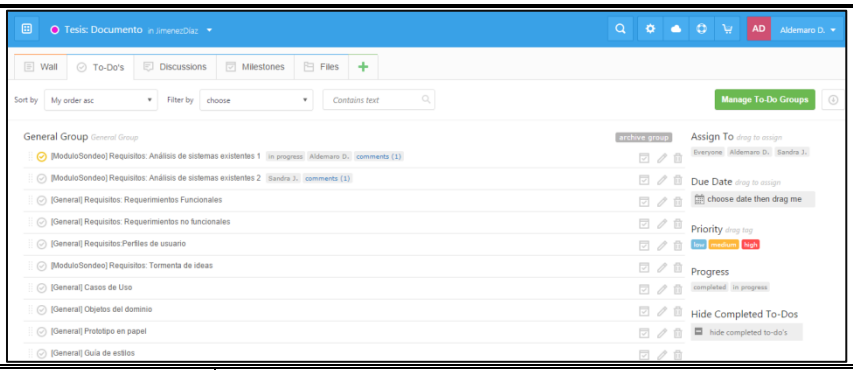
4.2.1 Etapa de Requisitos

En esta etapa se hacen evaluaciones de usabilidad de indagación, específicamente análisis de sistemas existentes. Se utiliza este artefacto ya que le permite al grupo de desarrollo evaluar otros sistemas tomando en cuenta sus aspectos positivos para incorporarlos en el sistema a desarrollar. A continuación se describen los sistemas analizados.

(i) Análisis de sistemas existentes

Se utiliza este artefacto de indagación con la finalidad de analizar las funcionalidades de sistemas similares que servirán para la toma de decisiones sobre aspectos de interfaz de usuario y funcionalidades desarrolladas en esta iteración, específicamente: asignación y creación de actividades, creación de proyectos, cambiar estado de una actividad, invitaciones a usuarios a colaborar en un proyecto, y comentar actividad. A continuación se presentan los sistemas existentes analizados, específicamente *FreedCamp* y *Asana* los cuales permiten gestionar actividades y grupos de desarrollo a través de una herramienta web.

FreedCamp es el primer sistema analizado, el cual se describe en la Tabla 5 especificando varios tópicos de este sistema.

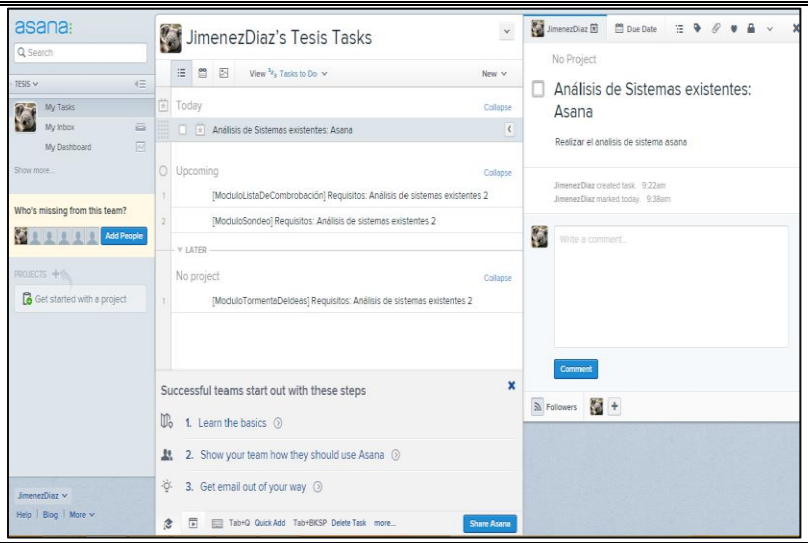
FreedCamp - https://freedcamp.com	
	
Tópico a evaluar	**Descripción/Observaciones**
Descripción	Es una herramienta web gratis para gestionar las tareas de uno o más proyectos de manera colaborativa
Funcionalidades	Permite crear tareas de un proyecto indicando su prioridad (alta, media o baja) y la asignación a un miembro del grupo de trabajo, también se pueden listar, editar y eliminar, además se puede hacer búsquedas y filtros sobre las tareas. Tiene un foro

	de discusión, para difundir nuevas ideas para algún proyecto. Adicionalmente tienen un gestor de tiempo, donde se escribir cuánto tiempo se ha tardado en realizar una tarea. De igual manera permite crear hitos (*milestones*) y subir archivos para compartir con el equipo de desarrollo
Personalización	Al crear un proyecto se le puede asociar un color, de manera que si se tienen varios proyectos sea fácil de identificar a través del color asignado. Permite crear *widgets* de alguna sección personalizada, por ejemplo: la lista de las tareas terminadas
Aspectos de sociabilidad	Los miembros del equipo pueden realizar comentarios sobre sus propias tareas o las tareas de otros. Se puede conectar la cuenta de *FreedCamp* con las redes sociales Google+, Twitter, Facebook y LinkedIn. Finalmente el foro de discusiones le permite al equipo debatir sobre el proyecto en desarrollo
Idiomas	Inglés
Opinión como usuario	La interfaz de usuario es moderna e intuitiva. Los colores utilizados son adecuados, en la parte superior se utiliza el color azul para destacar la barra superior y el resto de la página está en colores blanco y distintos tonos de gris que facilitan la lectura, de igual manera se utilizan colores adecuados para indicar la prioridad de una tarea. Se hace buen uso de metáforas en las opciones del menú y de la barra superior. El uso de indicadores visuales cuando se está cargando alguna sección permite saber el estado del sistema

Tabla 5 – Análisis de sistema existente *Freedcamp*

Este análisis permite tomar en cuenta características del sistema analizado para incorporarlas en la aplicación web ProAgil. El listado de las actividades se realiza como se propone en *FreedCamp*, especificando el estado de la actividad a través de una metáfora en un color relacionado al estado, el título y a quién está asignada. Otro aspecto a tomar en cuenta de este sistema son los filtros los cuales son desarrollados como en este sistema. La sección de comentarios de *FreedCamp* es intuitiva y por esta razón se implementa esta funcionalidad de manera similar.

La otra aplicación que se toma en cuenta es *Asana*, la cual es una herramienta de gestión de tareas de manera compartida. En la Tabla 6 se describen los tópicos tomados en cuenta.

Asana - https://app.asana.com/	
	
Tópico a evaluar	**Descripción/Observaciones**
Descripción	Es una herramienta web gratis de gestión de tareas, permite a los equipos compartir, planificar, organizar y seguir el progreso de las tareas en las que cada miembro está trabajando
Funcionalidades	Permite realizar registro con una dirección de email, o bien, utilizar la cuenta de Gmail o Google Apps, una vez registrado, es posible crear un espacio de trabajo (*workspace*), al cual se le puede asignar un nombre. Invitar a compañeros de trabajo a través de un correo electrónico para enviar el vínculo con la invitación del proyecto. Es posible crear y asignar tareas, cada una se puede asignar a miembros del grupo, agregar una fecha de vencimiento, añadir archivos, marcarla como completada, y seguir su actividad y progreso. Otras características incorporada son los *tags* que se pueden agregar a cada área de trabajo, la "sincronización del calendario", y los "atajos de teclado"
Personalización	Al crear una tarea de un proyecto es posible relacionarla a un *tag*. Al crear un proyecto se le puede asociar un color y además agregar como favorito. Es posible cambiar el tema del espacio de trabajo
Aspectos de sociabilidad	Los miembros del equipo pueden realizar comentarios sobre sus propias tareas o las tareas de otros. Provee una bandeja de entrada que muestra toda la actividad en tareas y proyectos que están siguiendo
Idiomas	Inglés
Opinión como usuario	La interfaz de usuario es moderna y contiene metáforas adecuadas e intuitivas, utiliza indicadores visuales cuando el usuario intenta cargar una sección, además le permite al usuario mostrar solo lo que el mismo desea ver, ya que contiene botones que permiten ocultar secciones, logrando tener disponibilidad de

	lo necesario. Provee varias opciones visibles para el soporte o ayuda al usuario. Hace especial énfasis en el uso del teclado para trabajar con los proyectos, la mayoría de la acciones se pueden hacer con atajos del teclado.

Tabla 6 – Análisis de sistema existente *Asana*

Con respecto a *Asana* se considera la forma de la creación de etiquetas (*tags*) para incorporarlos de manera similar a ProAgil. Esta característica se utiliza en la creación de actividades. De igual manera se toman en cuenta metáforas estándares utilizadas en este sistema tales como: inicio, perfil de usuario, configurar y agregar.

Una vez realizada la indagación con el análisis de sistemas existentes *Freedcamp* y *Asana*, la siguiente es la etapa de análisis donde se realizan los prototipos en papel de las interfaces de usuario para las funcionalidades del módulo de creación de proyectos y actividades.

4.2.2 Etapa de Análisis

Esta etapa está basada en la construcción del artefacto prototipo en papel de las funcionalidades de proyectos y actividades. Se selecciona solamente este artefacto ya que previamente en la iteración 1 se modelaron las funcionalidades a través de los diagramas, los cuales se consultan y a partir de ellos se realiza la construcción de prototipo en papel. Además el prototipo en papel permite tener una idea rápida de los componentes de interfaz de usuario.

(i) Prototipo en papel

En esta etapa se realiza prototipos en papel asociados a las funcionalidades de los proyectos y actividades. En la Figura 24 se ilustran los prototipos de las interfaces de usuario más importantes de esta iteración las cuales son: creación de proyecto y detalle de proyecto. Para la creación de proyecto se plantea una interfaz de usuario con un formulario en donde se especifique información y artefactos asociados a un proyecto. Para el detalle de proyecto se propone una interfaz de usuario en donde se listan los artefactos y actividades asociadas a un proyecto.

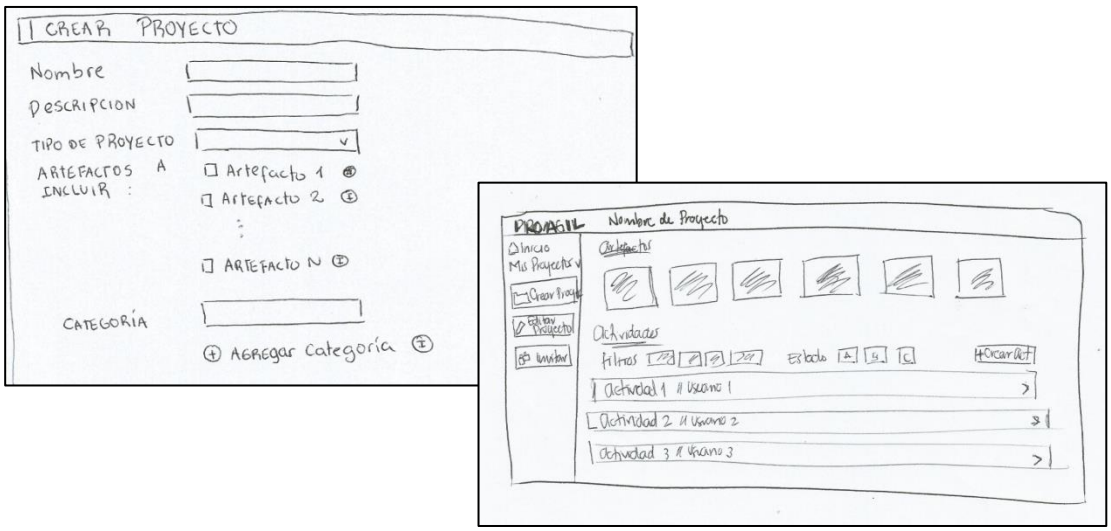


Figura 24 – Prototipo en papel: Creación de proyecto y detalle de proyecto

El uso de prototipos en papel facilita la creación de los prototipos ejecutables a lo largo de todo el desarrollo ya que proveen una guía inicial de la interfaz de usuario y sirven de punto de partida para crear el prototipo ejecutable en la etapa de prototipaje presentada seguidamente.

4.2.3 Etapa de Prototipaje

En esta etapa se desarrolla el prototipo ejecutable de las funcionalidades asociadas a la configuración de proyectos y actividades. Para la realización de los prototipos ejecutables de la aplicación se utilizan las tecnologías y herramientas web descritas en el Capítulo 3, en la Figura 25 se ilustran las diversas tecnologías que conforman la aplicación. A efectos del documento se presentan los prototipos ejecutables de creación y detalle de proyecto las cuales se consideran como las interfaces de usuario más importantes de la iteración, ya que resultan ser las más retadoras para el desarrollo de la aplicación. Adicionalmente, se realiza la evaluación heurística correspondiente a las funcionalidades implementadas.

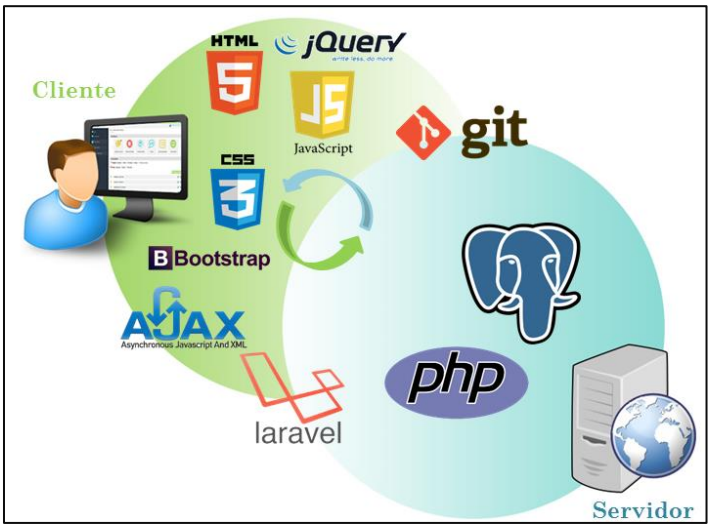


Figura 25 – Arquitectura de software

(i) Prototipo ejecutable

Corresponde al desarrollo de las funcionalidades asociadas a los proyectos y actividades. A continuación se presentan las interfaces de usuario correspondientes a la creación y detalle de proyecto. Estas interfaces de usuario surgen a partir de la evolución de los prototipos en papel propuestos en la etapa de análisis y se detallan a continuación.

- **Creación de proyectos**

  Corresponde a la interfaz de usuario para la configuración inicial de proyectos en el sistema. Esta interfaz de usuario permite indicar la información del proyecto descrita en la Figura 26. Adicionalmente como se ilustra en la Figura 27, el usuario puede agregar las iteraciones asociadas al proyecto. Por cada iteración el usuario especifica los colaboradores y los artefactos de evaluación de usabilidad que desea incorporar. Esto permite que la configuración de proyectos sea adaptable a las necesidades del equipo de desarrollo.

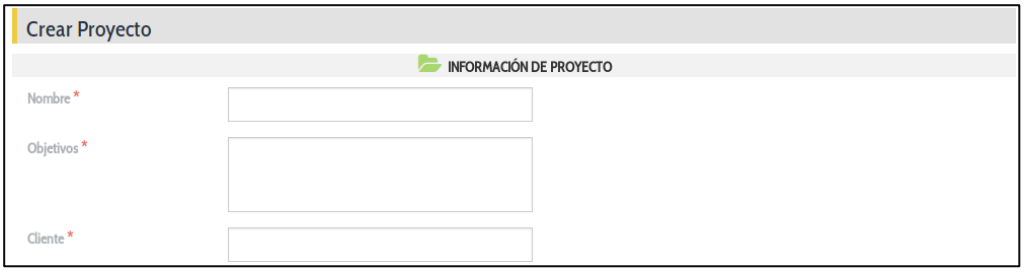


Figura 26 – Prototipo ejecutable: Crear proyecto (Información de proyecto)

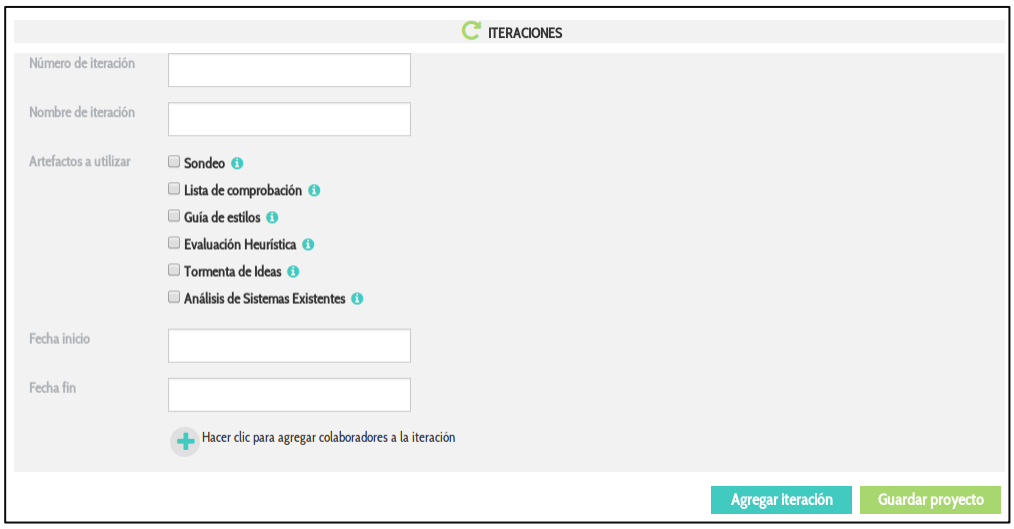


Figura 27 – Prototipo ejecutable: Crear proyecto (Iteraciones)

– **Detalle de proyecto**

En la Figura 28 se ilustra el prototipo ejecutable del detalle de proyecto. En esta interfaz de usuario se listan los artefactos de evaluación de usabilidad y las actividades asociadas a cada iteración del proyecto. La creación de este prototipo ejecutable resulta retadora ya que se debe presentar de manera intuitiva los elementos asociados a un proyecto en una interfaz de usuario fácil de utilizar.

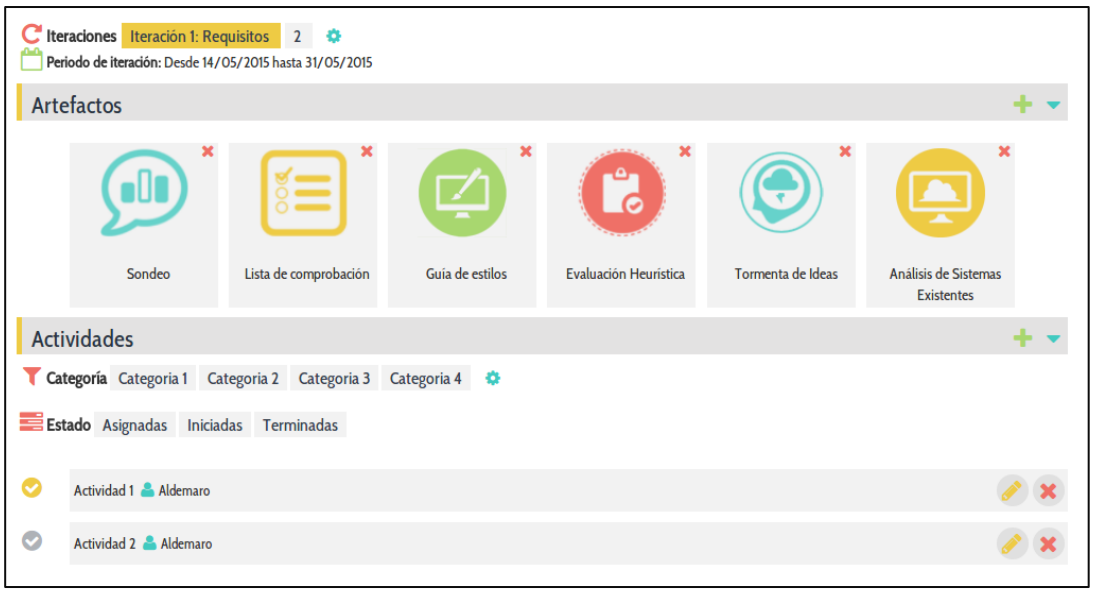


Figura 28 – Prototipo ejecutable: Detalle de proyecto

Al hacer clic sobre cualquier actividad se muestra en detalle (ver Figura 29). Adicionalmente, el usuario en sesión puede comentar la actividad potenciando la sociabilidad dentro de la aplicación.

Figura 29 – Prototipo ejecutable: Detalle de actividad

(ii) Evaluación Heurística

Corresponde a la evaluación de usabilidad del prototipo ejecutable desarrollado en la iteración 2. Con la aplicación de este artefacto de inspección se obtuvieron los problemas que se detallan en la Tabla 7. Para su realización dos usuarios en el rol de especialistas de Interacción Humano-Computador hacen recorridos por las interfaces de usuario para identificar los problemas basados en las heurísticas de Nielsen. Cada recorrido con una duración aproximadamente 20 minutos.

#	Problema	Heurística	Valoración	Solución
1	El usuario crea un proyecto sin seleccionar artefactos, al ver el detalle de proyecto en la sección de "Artefactos" se muestran 2 flechas sin ningún contenido	H1	2	Indicarle al usuario con mensaje descriptivo que no ha seleccionado ningún artefacto para el proyecto
2	Al crear un proyecto no se le indica al usuario cuáles campos son obligatorios antes de presionar el botón "Guardar"	H9	2	Indicarle al usuario con un "*" que los campos con obligatorios
3	Al seleccionar una "Fecha tope" para una actividad los meses y días salen en inglés	H2	1	Que el calendario salga en español
4	El formato de fecha al crear	H2	1	El formato debería ser el

	una actividad se presenta en formato Año/Mes/día			estándar para Latinoamérica día/mes/año
5	Al editar una actividad en el campo “Asignar actividad” no se selecciona a quien tiene asignada la actividad sino el primero de la lista	H5	2	Es necesario que salga seleccionado el usuario al que se le asignó una actividad
6	Al editar una actividad en el campo de texto de fecha tope aparece la fecha junto con 00:00:00. Adicionalmente no se marca en el calendario la fecha guardada sino la fecha actual	H5	3	Sólo debería aparecer la fecha. Adicionalmente al abrir el calendario para seleccionar una fecha debería estar seleccionada la fecha guardada

Tabla 7 – Evaluación heurística de la iteración 2

La detección de los problemas especificados permite solucionarlos oportunamente incorporando mecanismos de validación en las vistas para mostrar los contenidos correctamente. La corrección de cada problema se realiza según la valoración dándole prioridad a los problemas con valoración más alta. Para el problema con valoración 3 se cambia el tipo de dato del atributo y se establece la fecha guardada en el *datepicker* al editar una actividad. Los problemas con valoración 2 (1, 2 y 5) se agregan los mensajes descriptivos e indicadores de campos obligatorios para el usuario. Para el problema 5 se realiza la validación correspondiente para que se cargue el responsable de una actividad al editar la misma. Para resolver los problemas de valoración 1 (3 y 4) se configura el *datepicker* en español para que las fechas tengan el formato correcto.

Una vez completada la iteración de proyectos y actividades, a continuación se describe la iteración 3 la cual correspondiente al módulo para generar formularios que permitan auomatizar cuatro (4) de los seis artefactos de evaluación de usabilidad propuestos en esta investigación.

4.3 Iteración 3: Módulo para generar formularios

Esta iteración tiene como propósito desarrollar un componente de software que permita dar cumplimiento al requerimiento funcional generar formularios, los cuales servirán para registrar información para la realización de algunas evaluaciones de usabilidad, especificamente: sondeo, análisis de sistema existente, evaluación heurística y lista de comprobación. Con la finalidad de dar cumplimiento al objetivo, en la etapa de requisitos se realiza el análisis de varios sistemas existentes. En la

etapa de análisis se realiza el protipo en papel de la interfaz de usuario para generar formularios de tipo sondeo y listas de comprobación. Luego, en la etapa de prototipaje se desarrollan los prototipos ejecutables para generar formularios y aplicar los artefactos de evaluación de usabilidad. Por último, se realiza la evaluación heurística del módulo desarrollado. Seguidamente se describen cada una de las etapas de la iteración.

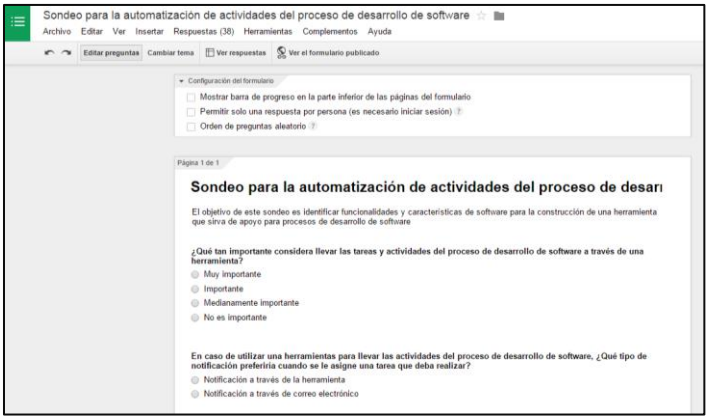
4.3.1 Etapa de Requisitos

Corresponde a la primera etapa de la tercera iteración, donde se realizan técnicas de evaluación de usabilidad de tipo indagación específicamente el artefacto utilizado es análisis de sistemas existentes, con el objetivo de identificar características de sistemas similares que permitan generar formularios.

(i) Análisis de sistemas existentes

Con la finalidad de identificar cualidades y aspectos de interacción de otros sistemas que poseen funcionalidades similares a las que se desea desarrollar, se analizaron varios sistemas que permiten generar formularios. A efectos de este documento se presentan dos (2) de los sistemas evaluados: *Google Forms* y *Checkli*. Particularmente *Google Forms* permite generar formularios de manera dinámica para realizar encuestas. Por otro lado, *Checkli* permite crear listas de cosas por hacer y verificación de tareas. A continuación, se detalla el análisis de los sistemas utilizando una tabla con tópicos a evaluar, descripción y/u observaciones de cada uno de los tópicos.

El primer sistema analizado es la aplicación *Google Forms*, descrita en la Tabla 8

Google forms - www.google.com/Forms	
	
Tópico a evaluar	**Descripción/Observaciones**
Descripción	Es una herramienta web para crear, difundir y generar reportes de encuestas
Funcionalidades	Permite crear encuestas utilizando distintos componentes

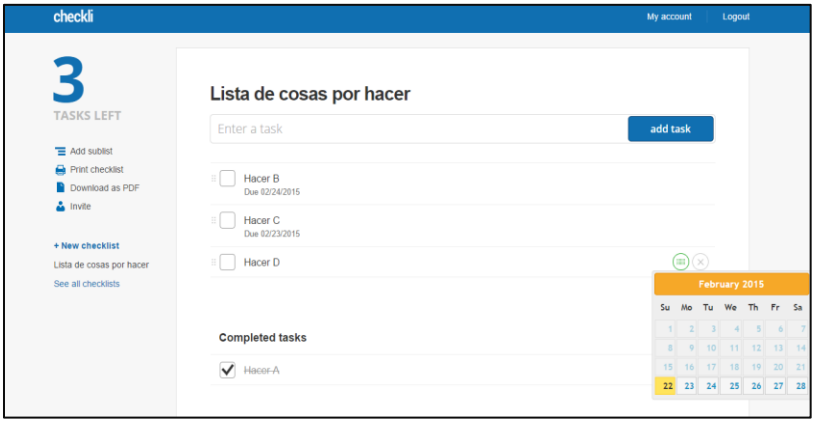
	de formularios, específicamente: Campos de texto, selección simple (*radio button*), selección múltiple (*checkbox*), selección de una lista (*select*), escala, cuadrícula, fecha y hora. Una vez generada la encuesta, permite generar un enlace para recopilar las respuestas de las personas que la contesten. Otra funcionalidad que provee es generar reportes de los resultados a través de gráficos de barra o gráficos de torta.
Personalización	La creación de formularios es un aspecto de personalización, ya que cada usuario que realice una encuesta en este sistema utiliza los componentes según sus necesidades. Otro aspecto de personalización que permite es que se puede seleccionar alguno de los temas predefinidos para la presentación de la encuesta. Adicionalmente, permite seleccionar si se desea que se presenten las preguntas de manera aleatoria. Provee la opción de presentar una barra de progreso para que el usuario sepa el porcentaje de preguntas que ha contestado.
Aspectos de sociabilidad	Una vez que se crea una encuesta, permite compartirla en Google+, Twitter y Facebook para que los usuarios la contesten, además permite enviar el enlace de la encuesta por correo electrónico
Idiomas	En el idioma que en el que se tenga configurada la cuenta de correo electrónico de Google (gmail)
Opinión como usuario	Es una herramienta muy útil para generar encuestas con distintos elementos lo que permite utilizar varios tipos de preguntas. La edición de las preguntas es sencilla e intuitiva. Utiliza metáforas adecuadas para editar y eliminar contenido. La sección de reportes de las respuestas es muy útil y se presenta de manera sencilla. La única opción que no se presenta con *shortcut* es para agregar una nueva pregunta, para hacerlo hay que ir a la opción "insertar" del menú principal.

Tabla 8 – Análisis de sistema existente *Google Forms*

Una vez realizado el análisis de la aplicación *Google Forms,* se detectaron algunos aspectos de la interfaz de usuario, datos a solicitar de importancia y estilos de interacción a tomar en cuenta para el desarrollo. *Google Forms* utiliza una interfaz de usuario con combinaciones de tonos grises claros y oscuros, lo que proporciona un aspecto formal a la interfaz de usuario. También es necesario que el usuario deba establecer parámetros generales de la encuesta como nombre y descripción, por otro lado en cuanto a la creación de preguntas existe una sección más oscura que se despliega cada vez que el usuario desee agregar una nueva pregunta. Otra funcionalidad importante que provee la aplicación es que permite al usuario compartir

la encuesta, una vez elaborada, a través de las redes sociales. Todos estos aspectos son tomados en cuenta para el desarrollo del módulo de generación de formularios.

También se realizó el análisis de sistema existente para la creación de formularios que generen listas de comprobación. Básicamente, se tomó en cuenta sistemas que permiten a los usuarios crear listas de cosas por hacer, para luego verificar cuáles se cumplieron y cuáles no. El análisis del sistema *Checkli*, se muestra en la Tabla 9.

<table>
<tr><td colspan="2">Checkli – https://www.checkli.com/</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><th>Tópico a evaluar</th><th>Descripción/Observaciones</th></tr>
<tr><td>Descripción</td><td>Checkli es una herramienta web para crear y verificar listas de cosas por hacer (TODO list) e invitar a usuarios a participar en una lista</td></tr>
<tr><td>Funcionalidades</td><td>Para poder crear listas de cosas por hacer es necesario crea una cuenta en la aplicación, luego de haberlo hecho es posible crear una o más listas de cosas por hacer (tareas) de manera sencilla. Permite crear y editar los nombres de las listas y de las tareas. Las tareas pueden tener fechas para su realización, pueden ser marcadas como terminadas o pueden ser eliminadas, incluso cada tarea puede tener una sub-lista de tareas. Permite invitar a otros usuarios y compartir la lista de tareas vía correo electrónico. Provee un sistema de filtro de las lista de tareas ya sea por las compartidas, las completadas y las que le pertenecen al usuario que ha iniciado sesión. Finalmente, la herramienta tiene un buscador que permite filtrar las listas de tareas, dependiendo de lo que el usuario desee y sus necesidades</td></tr>
<tr><td>Personalización</td><td>Provee la funcionalidad de establecer fechas a las tareas, lo que permite un control más real de las tareas por hacer. Permite al usuario seleccionar una foto como perfil. El</td></tr>
</table>

	hecho de que permita crear lista de cosas por hacer de cualquier ámbito hace que la aplicación sea totalmente personalizable
Aspectos de sociabilidad	Se puede invitar a otros usuarios a formar parte de la lista vía correo electrónico
Idiomas	Inglés
Opinión como usuario	Es una herramienta con una guía de estilo bastante minimalista y simple, utiliza colores como el azul y blanco, los cuales son bastante sobrios. Es posible utilizarla la herramienta en cualquier área dependiendo de las necesidades de quien la utilice. Permite llevar una lista de tareas sencillas a verificar. Es una aplicación muy básica pero cumple con su objetivo, podría tener muchas mejoras a fututo como por ejemplo, permitir definir colores a las tareas, e incluso una versión para dispositivos móviles ya que es bastante sencilla

Tabla 9 – Análisis de sistema existente *Checkli*

En el caso de la aplicación *Checkli* permite crear tareas o cosas por hacer de una manera muy simple con pocos parámetros, ya que solo se le solicita al usuario el nombre de la tarea y la fecha tope para su realización. Luego de crear un conjunto de tareas es posible ver una lista de tareas donde en cada caso se muestra con una casilla vacía aquellas tareas que no se han realizado y con una casilla con una marca de verificación aquella que se hayan seleccionado como completadas. Cada uno de estos aspectos descritos, son tomados en cuenta para el desarrollo del formulario para la aplicación del artefacto de evaluación de usabilidad de lista de comprobación.

4.3.2 Etapa de Análisis

Con el propósito de modelar las características tomadas en cuenta en la etapa de requisitos a través del análisis de sistemas existentes. En esta etapa se crean los prototipos en papel de las interfaces de usuario que permiten generar formularios, específicamente se presentan los prototipos para los artefactos de tipo sondeo y listas de comprobación.

(i) Prototipo en papel

Corresponde a la creación de prototipos de baja fidelidad, con la finalidad de establecer una primera visión o guía de las interfaces de usuario para la creación de formularios, por simplicidad se presentan los prototipos para los artefactos sondeo y lista de comprobación. El artefacto evaluación heurística y análisis de sistemas existentes tienen estructura similar al sondeo. Con respecto a la lista de comprobación tiene estructura distinta a las mencionadas por lo cual también se presenta su prototipo.

En la Figura 30 se aprecia el prototipo para generar un formulario del artefacto de tipo sondeo. En principio se especifican los datos del artefacto, luego se plantea crear una fila con las especificaciones de cada pregunta del sondeo.

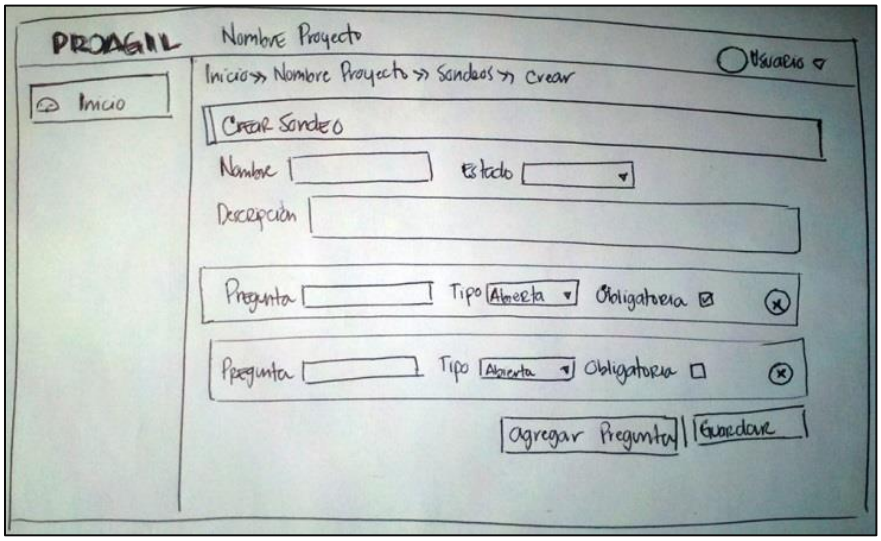


Figura 30 – Prototipo en papel: Generar formulario de tipo sondeo

Seguidamente, se presenta en la Figura 31 el prototipo en papel para crear listas de comprobación. Se propone una interfaz de usuario en donde se debe introducir el nombre de la lista de comprobación, además se deben seleccionar los principios que serán verificados una vez creada.

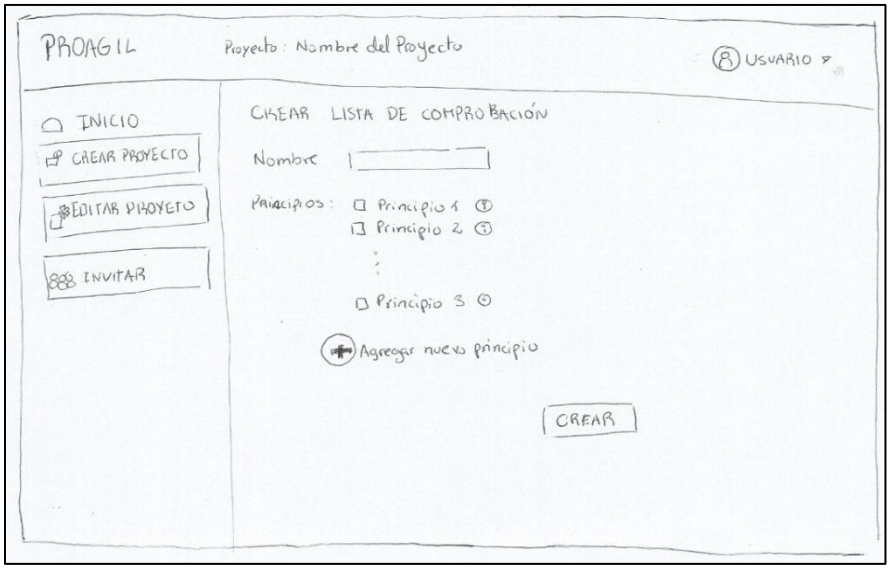


Figura 31 – Prototipo en papel: Crear lista de comprobación

4.3.3 Etapa de Prototipaje

Esta tercera etapa se refiere al desarrollo del requerimiento funcional de generación de formularios. Como ya se ha descrito, esta funcionalidad provee la facilidad de crear

dinámicamente formularios relacionados a artefactos de evaluación de usabilidad, específicamente: sondeos, lista de comprobación análisis de sistemas existentes y evaluación heurística. A continuación, se muestran en detalle las interfaces de usuario implementadas para la creación de sondeos y la lista de comprobación las cuales poseen características importantes a resaltar.

(i) Prototipo ejecutable

En esta sección se muestran las interfaces de usuario desarrolladas correspondientes a los artefactos sondeo y lista de comprobación.

- **Crear sondeo**

  La creación de sondeo es una funcionalidad disponible para los usuarios con el rol de líderes de proyecto. Para su desarrollo se evoluciona el prototipo en papel incorporando elementos que permitan mantener la interfaz de usuario sencilla y organizada. En esta interfaz de usuario se indican los datos que identifican el sondeo. Luego de indicar los datos, se pueden agregar la cantidad de preguntas necesarias. Por cada pregunta se puede especificar: tipo (abierta o cerrada) y un *checkbox* que permite especificar si es una pregunta obligatoria. En la Figura 32 se ilustra la interfaz de usuario para crear sondeos. El resultado de un sondeo generado con los elementos de formulario se presenta en el anexo *d: Ejemplo de sondeo creado*

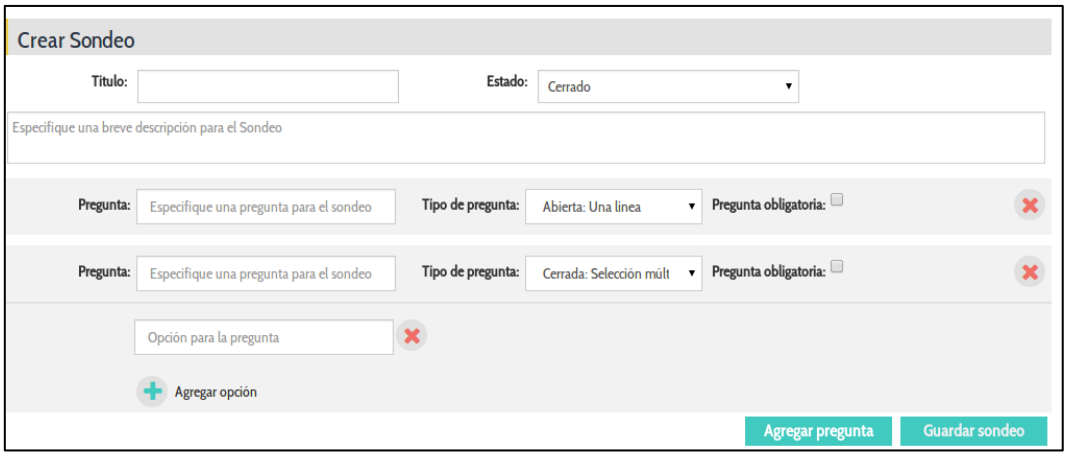


Figura 32 – Prototipo ejecutable: Crear sondeo

- **Listado de sondeos**

  Se refiere a la interfaz de usuario donde se listan todos los sondeos creados en una iteración. Las acciones que se pueden hacer sobre un sondeo son: responderlo, ver los resultados y compartirlo en Twitter, Facebook o LinkedIn con la finalidad de incorporar funcionalidades de sociabilidad en la aplicación, se tomaron en cuenta estas redes sociales basadas en los resultados del sondeo en la iteración 1. La Figura 33 ilustra el listado de sondeos incluyendo los enlaces de compartir.

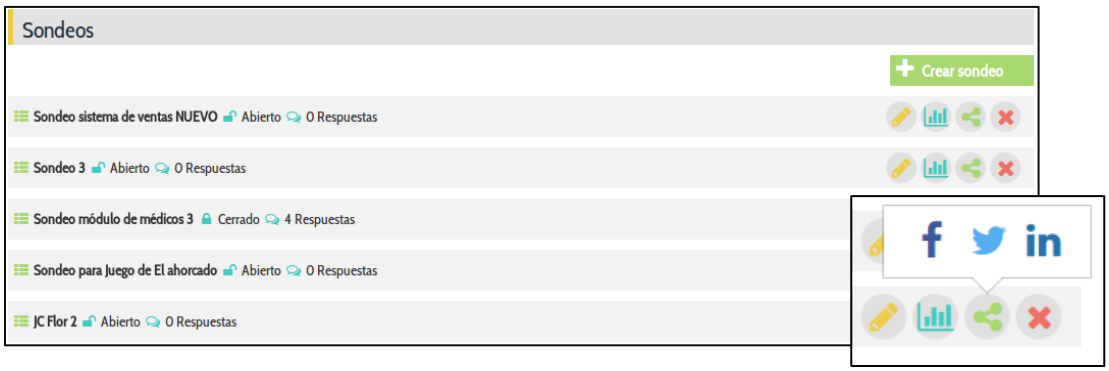


Figura 33 – Prototipo ejecutable: Listado de sondeos

– **Crear lista de comprobación**

Esta interfaz de usuario ilustrada en la Figura 34 provee la facilidad de crear listas de comprobación seleccionando principios o estableciendo unos personalizados. Para ello, el usuario indica los datos de la lista y los principios que desea verificar, por defecto el sistema provee las diez (10) heurísticas de Nielsen, adicionalmente con la finalidad de que el sistema se adapte a las necesidades del usuario puede especificar principios adicionales para realizar la verificación. Se establece una fecha tope de realización de la verificación con el motivo de darle un plazo al usuario para que realice la verificación de la lista de comprobación.

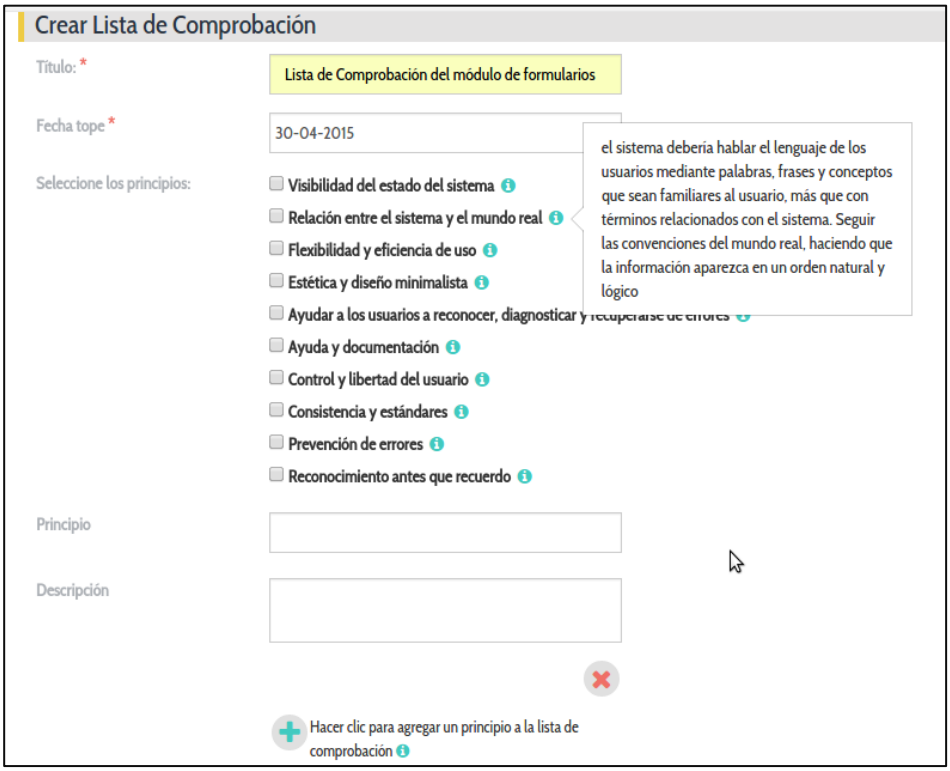


Figura 34 – Prototipo ejecutable: Crear lista de comprobación

- **Listado de listas de comprobación**
  Se refiere al conjunto de listas de comprobación creadas en la iteración al cual pertenece el artefacto. En la Figura 35, se observar la interfaz de usuario de un listado de listas de comprobación. Para cada lista de comprobación se podrá apreciar el estado, el cual puede ser *"Por Verificar"*, utilizando la metáfora de una casilla vacía si ningún colaborador o líder de proyecto ha realizado la verificación de la lista de comprobación respectiva, o *"Verificada"* utilizando la metáfora de una casilla marcada como verificada que indica que cualquier usuario ha realizado la verificación de la lista de comprobación. La utilización de la metáfora de la casilla vacía o marcada se determinó gracias al análisis del sistema *Checkli*.

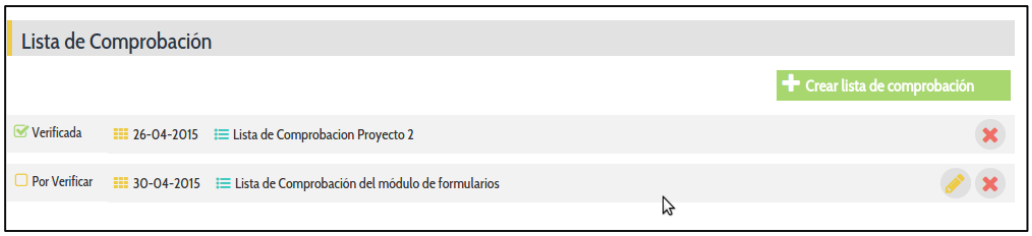


Figura 35 – Prototipo ejecutable: Listado de lista de comprobación

- **Verificar lista de comprobación**
  Cualquier usuario que pertenezca a la iteración, podrá realizar la verificación de una lista de comprobación específica, para ello se desplegará la interfaz de usuario que se muestra en la Figura 36. En esta sección el usuario debe para cada uno de los principios, especificar si el principio se cumple o no. Esta funcionalidad se realiza con la finalidad de verificar cada uno de los principios establecidos previamente, en la creación de la listas de comprobación.

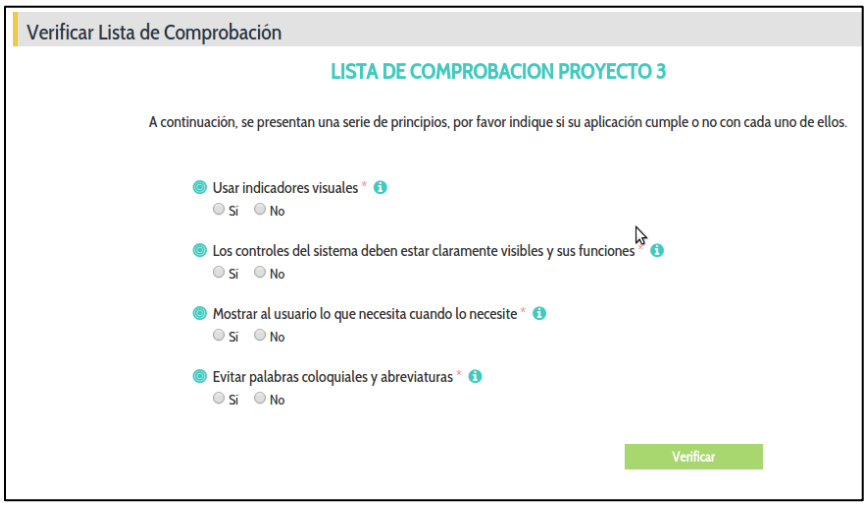


Figura 36 – Prototipo ejecutable: Verificar lista de comprobación

(ii) Evaluación heurística

Concierne a la evaluación de las funcionalidades desarrolladas en la iteración 3, para su realización se hace uso de las heurísticas propuestas por Nielsen. Con la aplicación de este artefacto de inspección realizado por dos expertos de evaluación de usabilidad, que evaluaron las interfaces de usuario creadas en el módulo de generación de formularios, durante aproximadamente treinta (30) minutos. Se obtuvieron problemas de desarrollo y usabilidad en las interfaces de usuario que involucran los artefactos: sondeo, evaluación heurística, análisis de sistemas existentes y lista de comprobación. De manera ordenada se describen los problemas encontrados, la heurística que viola, la valoración y una posible solución al problema encontrado en la Tabla 10.

#	Problema	Heurística	Valoración	Solución
1	Al verificar una lista de comprobación se mantiene con el estado "Por verificar"	H3	3	Una vez verificada la lista de comprobación debe cambiar al estado "Verificada"
2	Al darle clic para eliminar una lista de comprobación no aparece un mensaje de confirmación para el usuario.	H9	3	Debe aparecer un mensaje que le indique al usuario si realmente desea eliminar la lista de comprobación/item de la lista de comprobación
3	En el editar de lista de comprobación al eliminar un item de la lista no se borra	H3	3	Una vez que el usuario especifique que quiere eliminar el elemento se debe borrar
4	Cuando se crea un nuevo sondeo la palabra "titulo", "linea" y "parrafo" en el select del tipo de pregunta, les falta la tilde	H1	0	Colocar el tilde a las palabras
5	Cuando no hay sondeos debe indicar un mensaje descriptivo que no hay sondeos creados aún	H4, H5	1	Indicar con un mensaje corto que "no hay sondeos creados aún"
6	En la sección crear sondeo el botón "Guardar Sondeo" la letra es de color gris y el agregar pregunta la letra es de color blanco	H1, H4	1	Cambiar el color de las letras a blanco en los 2 botones. Una recomendación podría ser cambiar el color del botón "Guardar Sondeo" a verde
7	Cuando no hay Evaluación de Sistemas existente la	H4, H5	1	Indicar con un mensaje corto que "no hay Evaluación

	sección aparece en blanco			Heurística creados aún”
8	Cuando no hay Evaluación Heurística la sección aparece en blanco	H4, H5	1	Indicar con un mensaje corto que “no hay Evaluación Heurística creados aún”
9	En el editar el análisis de sistema existente no es visible la opción de editar la imagen si no presionas la opción de editar de la sección	H6,H7	2	Mostrarle al usuario debajo del nombre un label “Interfaz”, y si no ha subido la imagen indicarle que no ha cargado la imagen aún y si existe imagen mostrarla
10	En el editar de la evaluación heurística el botón cancelar en la sección del nombre no funciona y las letras del botón guardar ésta en color gris	H6	3	El botón debe funcionar y cancelar la edición
11	Cuando se visualiza el listado de la lista de comprobación como usuario colaborador se rueda el contenido	H4	3	Definir un tamaño al listado que permita visualizar el título de las lista de comprobación sin que se ruede

Tabla 10 – Evaluación heurística de iteración 3

Para solucionar los problemas se agrupan según la valoración de mayor a menor. Los problemas de valoración 3 (1, 2, 3, 10 y 11) se resuelven de manera separada tomando en cuenta las soluciones planteadas, para resolverlos es necesario hacer cambios las funciones de cada problema. Para el problema 9 de valoración 2 se implementa la funcionalidad de mostrar y editar la imagen del sistema existente. Con respecto a los problemas de valoración 1 (5, 6, 7 y 8) se agregan las validaciones correspondientes en las vistas y se agregan los mensajes descriptivos faltantes. El problema de valoración 0 se corrige arreglando los errores de tildes en las palabras indicadas.

Posteriormente, se realiza la iteración 4 correspondiente a los componentes gráficos de la aplicación. A continuación se detalla su desarrollo.

4.4 Iteración 4: Módulo de componentes gráficos

Esta iteración tiene como objetivo desarrollar los requerimientos relacionadas a componentes gráficos de la aplicación, específicamente la automatización de los artefactos: guía de estilos y tormenta de ideas. Los artefactos del método utilizados en esta iteración para dar cumplimiento al objetivo planteado son: en la etapa de

requisitos se analizan sistemas existentes de guías de estilos y tormentas de ideas; en la etapa de análisis se realizan prototipos en papel concernientes a las interfaces de usuario de tormentas de ideas, guías de estilos y resultados de sondeo; en la etapa de prototipaje se desarrollan los requerimientos funcionalidades correspondientes y se hace una evaluación heurística sobre los mismos.

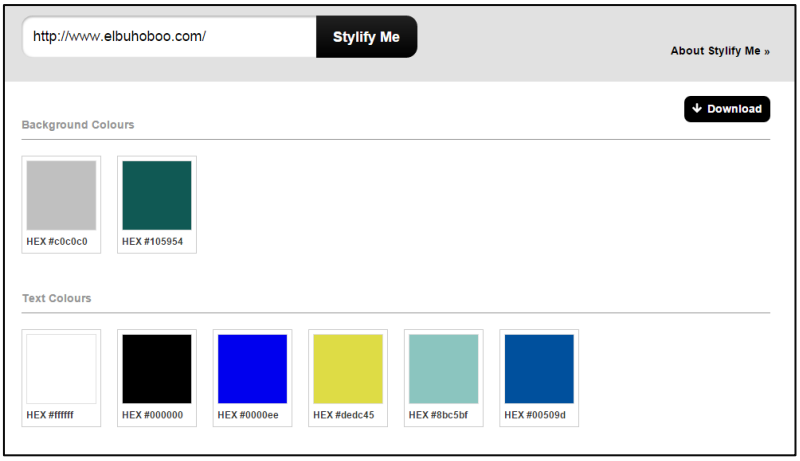
4.4.1 Etapa de Requisitos

En esta etapa se realizan análisis de sistemas existentes relacionados a los artefactos a desarrollar en la presente iteración con el objetivo de identificar características de este tipo de software.

(i) Análisis de sistemas existentes

Con la finalidad de evaluar componentes de otros sistemas se realizó el análisis de las diversas herramientas web para que permitan crear componentes gráficos. En esta sección se presentan las herramientas *Stylify Me* y *Wordle,* las cuales están relacionadas a los artefactos guía de estilos y tormenta de ideas respectivamente. A continuación se describe cada uno.

El primer sistema analizado por el grupo de desarrollo es *Stylify Me*, este sistema permite obtener una visión general de la guía de estilos de un sitio web al introducir la URL del mismo. A continuación en la Tabla 11, se describen las observaciones encontradas.

Stylify Me - http://stylifyme.com/


Tópico a evaluar	Descripción/Observaciones
Descripción	*Stylify Me* es una herramienta web que permite obtener una visión general de la guía de estilos de un sitio web.
Funcionalidades	A través de la especificación de la URL de un sitio web se

	pueden obtener algunos componentes pertenecientes a la guía de estilos, específicamente: Colores de fondo, colores de textos, tipografías, dimensiones de imágenes y una imagen previa del sitio. Adicionalmente permite descargar la guía de estilos generada en formato PDF.
Personalización	No tiene aspectos de personalización
Aspectos de sociabilidad	Permite compartir la guía de estilos en las redes sociales Twitter y Facebook. Adicionalmente se especifican sus perfiles en ambas redes sociales
Idiomas	Inglés
Opinión como usuario	Es una herramienta sencilla y fácil de utilizar. Resulta muy útil cuando se quieren saber los componentes gráficos de un sistema existente. La interfaz de usuario es simple y organizada. Se podría mejorar la generación de la guía de estilos con una opción que le permita al usuario especificar qué elementos de interfaz de usuario desea incorporar

Tabla 11 – Análisis de sistemas existentes *Stylify Me*

Posterior al análisis de la aplicación *Stylify Me*, se consideran algunos componentes importantes para la presentación de una guía de estilos, tales como: la paleta de colores, tipografía y una imagen de la interfaz de usuario inicial del software. Adicionalmente se toma una estructura de interfaz de usuario similar para la presentación de los colores de la guía de estilos en cuadrados en las guías de estilos generadas a través de la aplicación ProAgil.

Para el desarrollo del artefacto tormenta de ideas, se analiza el sistema *Worldle*, a través de este sistema se pueden generar nube de *tags* de un grupo de palabras dadas. En la Tabla 12 se describe el sistema.

Wordle - http://www.wordle.net/


Tópico a evaluar	Descripción/Observaciones
Descripción	*Wordle* es una aplicación web que permite crear nubes de palabras. Provee una cantidad diversa de tipos de fuentes y formas para crear las nubes
Funcionalidades	Admite agregar una lista de palabras para conformar una nube de palabras. Es posible indicar el tipo de fuente, la disposición de las palabras, así como también el color del fondo, y adicionalmente se puede agregar formato a las palabras (mayúsculas, minúsculas o letra capital). Provee la funcionalidad de imprimir el resultado o guardarlo en la galería pública de nubes de ideas del sitio web
Personalización	Es posible escoger distintas formas para la nube de palabras, cambiar la fuente, el color y el fondo.
Aspectos de sociabilidad	Tiene la opción de publicar la nube de palabras que genera la herramienta en la galería disponible de manera pública para todos los usuarios que ingresen en el sitio web
Idiomas	Inglés
Opinión como usuario	*Wordle* es una aplicación web sencilla, con pocas funcionalidades orientadas a generar nubes de ideas, posee un diseño simplista donde se utiliza el color blanco para el fondo y negro para la fuente. En cuanto a la interfaz de usuario no se hace uso de metáforas y no provee *feedback* al usuario, específicamente en el menú principal de la aplicación, ya que no hay manera de indicarle en que sección se encuentra

Tabla 12 – Análisis de sistemas existentes *Wordle*

Después de analizar este sistema las características tomadas en cuenta para ProAgil son: proveerle al usuario un campo de texto de tamaño moderado en donde pueda indicar las palabras que conforman la tormenta de ideas. Incporporar un mecanismo que le permita al usuario poder descargar la tormenta de ideas generada.

Inmediatamente del análisis de sistemas existentes como parte de la etapa de requisitos, se realizan los prototipos en papel para los artefactos tormenta de ideas y guía de estilos.

4.4.2 Etapa de Análisis

Con la finalidad de tener una idea inicial de las interfaces de usuario correspondientes a esta iteración, a continuación se muestran los siguientes prototipos en papel correspondientes a la creación de guía de estilos y tormenta de ideas.

(i) Prototipo en papel

- **Crear guía de estilos**

  En la Figura 37 se muestra el prototipo para la creación de la guía de estilos. Se propone una interfaz de usuario compuesta por pestañas en donde se

especifican de manera separada elementos que conforman la guía de estilos que el usuario puede establecer tales como: los colores, tipo de fuentes y una imagen de la interfaz de usuario.

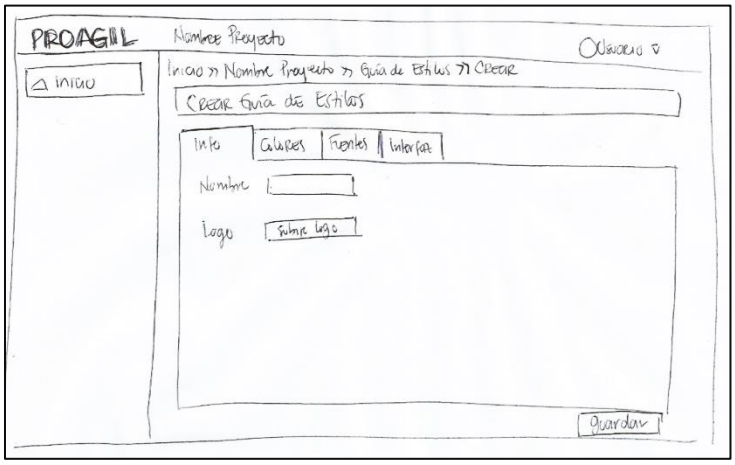


Figura 37 – Prototipo en papel: Guía de estilos

- **Crear tormenta de ideas**

  Para el requerimiento crear tormenta de ideas se propone una interfaz de usuario en donde el usuario indica las palabras para generar una nube de *tags* de la forma propuesta en la Figura 38.

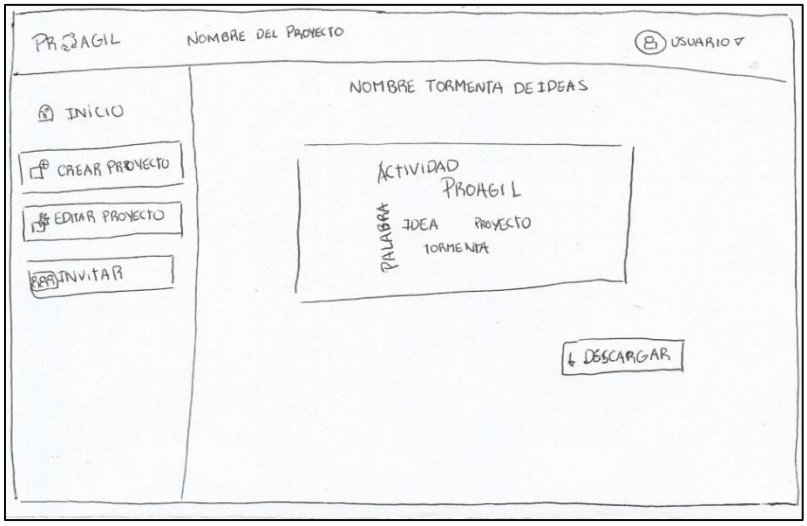


Figura 38 – Prototipo en papel: Tormenta de ideas

4.4.2 Etapa de Prototipaje

A partir de los prototipos en papel se realizó un refinamiento haciendo uso de las tecnologías propuestas para construir el prototipo ejecutable de la presente iteración, específicamente las siguientes interfaces de usuario: creación de guía de estilos y tormenta de ideas. Seguidamente se muestran cada una.

(i) Prototipo ejecutable

- **Crear guía de estilos**
  Corresponde a la funcionalidad que permite crear el artefacto guía de estilos. En esta interfaz de usuario a través del uso de *tabs* se especifican los elementos que componen una guía de estilos. Los *tabs* propuestos son: "Información", "Paleta de colores", "Tipografía" e "Interfaz de usuario", los cuales son los componentes que se determinaron usar al hacer el análisis de los sistemas en la etapa de requisitos de la iteración actual. En la Figura 39 se ilustra la interfaz de usuario para la creación de la guía de estilos. En el anexo *e: Ejemplo de guía de estilos generada* se muestra la interfaz de usuario que especifica el detalle de la guía de estilos creada.

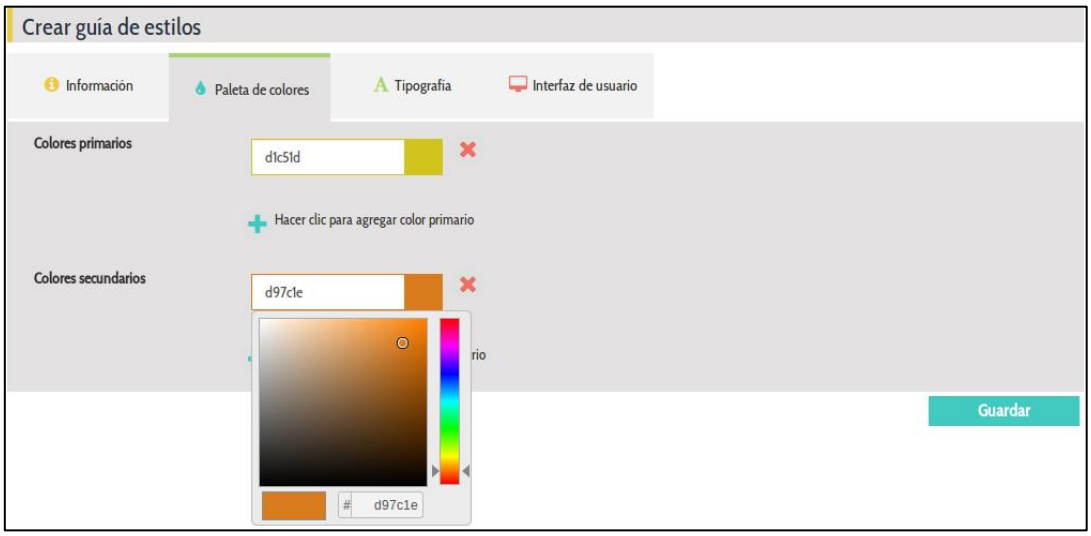


Figura 39 – Prototipo ejecutable: Guía de estilos

- **Crear tormenta de ideas**
  En cuanto a la funcionalidad del artefacto de indagación tormenta de ideas, cuando el usuario líder de proyecto desee crear una, deberá introducir el nombre y las ideas para generar la tormenta. Luego que se genera la tormenta de ideas como se muestra en la Figura 40, el usuario tendrá la posibilidad descargarla en formato de imagen *png*, y almacenarla en su dispositivo electrónico.

Figura 40 – Prototipo ejecutable: Resultado de crear tormenta de ideas

Elaborado el prototipo ejecutable de la iteración 4 se procede a realizar la evaluación heurística de las funcionalidades desarrolladas.

(ii) Evaluación heurística

Corresponde a la evaluación de usabilidad del prototipo ejecutable desarrollado en la iteración 4. Para su realización dos usuarios en el rol de especialistas de Interacción Humano-Computador hicieron recorridos de quince (15) minutos aproximadamente por las interfaces de usuario para identificar los problemas. En la Tabla 13 se presentan los problemas encontrados

#	Problema	Heurística	Valoración	Solución
1	Al crear una tormenta de ideas existe un botón "Regresar"	H3	2	Utilizar el botón "Volver" utilizado en el resto de la aplicación para realizar la acción y mantener la consistencia
2	Al darle clic para eliminar una tormenta de ideas no aparece un mensaje de confirmación para el usuario	H9	3	Debe aparecer un mensaje que le indique al usuario si realmente desea eliminar la tormenta de ideas
3	Al enviar las respuestas del en el formulario del sondeo para los usuarios finales una vez completado, el mensaje no le da las gracias al usuario	H2	1	Mostrar al usuario un mensaje agradeciendo por llenar el formulario de encuesta

4	En los reportes del sondeo los resultados en el mismo orden que cuando se realizan las preguntas al usuario	H3	1	Mostrar los resultados de cada encuesta en el mismo orden en que se hacen las preguntas en el sondeo, si es posible enumerarlas
5	En los reportes del sondeo los resultados de las preguntas de tipo línea o párrafo se ven muy pegadas	H4	1	Colocar las respuestas de los usuarios al mismo nivel de los gráficos

Tabla 13 – Evaluación heurística: Iteración 4

Los problemas de esta iteración se agrupan igualmente según su valoración. Para el problema 2 de valoración 3 se agrega el mensaje de confirmación correspondiente al hacer clic en el botón de eliminar tormenta de ideas. Con respecto al problema 1 de valoración 2 se elimina el botón con el texto "regresar" y se realiza dicha acción a través del enlace "volver" para mantener la consistencia. Para los problemas de valoración 1 (3, 4 y 5) se realizan los cambios de acuerdo a las soluciones propuestas. Para el problema 3 se agrega un mensaje agradeciendole al usuario por responder el sondeo. Con respecto al problema 4 se realizan los cambios en las funciones para listar las preguntas en el mismo orden que se crean ya que se previamente se mostraban según el tipo. El problema 5 se resuelve realizando cambios de algunas clases en el archivo CSS.

Luego de de realizar los prototipos ejecutables en las iteraciones descritas, a continuación se presenta la iteración cinco (5) relacionada a las pruebas de aceptación de la aplicación.

4.5 Iteración 5: Pruebas de aceptación

Con la finalidad de tener *feedback* de usuarios con respecto a su experiencia interactuando con la aplicación se aplica el artefacto de pruebas de aceptación a quince (15) personas que han aplicado los artefactos de indagación e inspección en los que se enfoca esta investigación. La prueba está compuesta por las siguientes cinco (5) preguntas con una escala del 1 al 4 donde 1 es el número de apreciación más bajo y 4 el más alto.

Pregunta 1 – ¿Considera que la aplicación podría apoyar al proceso de desarrollo de software (PDSW)?
Pregunta 2 – ¿La creación de los artefactos de evaluación de usabilidad se realizó de manera sencilla?
Pregunta 3 – ¿Considera que la aplicación es usable?
Pregunta 4 – ¿El diseño de la aplicación le parece consistente?
Pregunta 5 – ¿Considera que la aplicación es atractiva?

En el Gráfico 10 se presentan los resultados obtenidos por cada pregunta de la prueba de aceptación

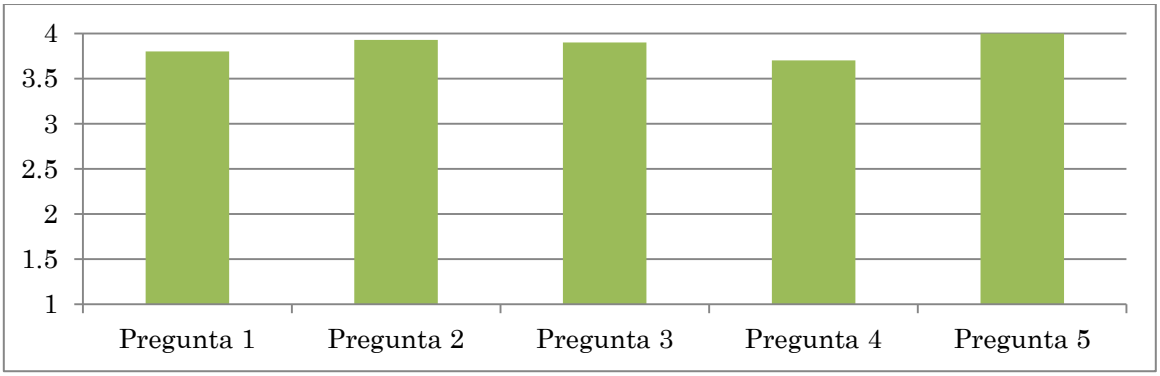


Gráfico 10 – Pruebas de aceptación

Con los resultados obtenidos en la prueba de aceptación es posible concluir que la aplicación ProAgil tiene una alta aceptación con un margen mayor al 95% por parte de los usuarios encuestados. Logrando desarrollar un software que apoya el proceso de desarrollo de software a través de la creación de artefactos de evaluación de usabilidad de indagación e inspección en una aplicación web usable y atractiva. Adicionalmente, con la aplicación de este artefacto se detectan errores de funcionalidad y navegación en algunas interfaces de usuario, los cuales son corregidos posteriormente.

Con las pruebas de aceptación se culmina este capítulo correspondiente al desarrollo de la aplicación web ProAgil a través del método AgilUs. Este método permitió concebir una aplicación usable a través del uso de técnicas sencillas pero útiles para generar software de calidad. Dicho esto, a continuación se presentan las conclusiones y recomendaciones para la aplicación desarrollada.

Conclusiones y Recomendaciones

El uso de herramientas tecnológicas para dar apoyo al proceso de desarrollo de software es una tendencia que provee mecanismos para organizar, planificar y potenciar aspectos de calidad de las aplicaciones. Esta investigación propone desarrollar una herramienta que permita potenciar la usabilidad en proyectos de software a través del uso de evaluaciones de usabilidad. Por esta razón, el objetivo general planteado inicialmente fue desarrollar la aplicación web "ProAgil" para automatizar actividades y artefactos de evaluación de usabilidad de indagación e inspección para procesos de desarrollo de software, el cual se cumplió ya que se desarrolló una aplicación que permite configurar actividades y crear artefactos de evaluación usabilidad de indagación e inspección para proyectos de software.

A través del desarrollo de esta investigación, se han obtenido resultados relacionados con los objetivos planteados inicialmente, los cuales se describen a continuación.

- Analizar herramientas que apoyan el proceso de desarrollo de software permitió evaluar plataformas que para organizar actividades de grupos de desarrollo. Las herramientas analizadas fueron *Asana* y *FreedCamp*, las cuales permiten de manera general crear y asignar tareas. La evaluación de estos sistemas permitió definir los componentes para el módulo de creación de proyectos y actividades especificados en la iteración 2 del proceso de desarrollo.

- Seleccionar las tecnologías y herramientas web a utilizar para el desarrollo de la aplicación fue un objetivo logrado de manera satisfactoria. Las tecnologías del lado del cliente como HTML, CSS, jQuery y Bootstrap permitieron construir interfaces de usuario usables. Con respecto a las tecnologías del lado del servidor, PHP resultó de gran utilidad ya que es un lenguaje conocido por el grupo de desarrollo. El manejador de bases de datos PostgreSQL al principio resultó complicado para la creación de entidades pero luego se convirtió en una tarea sencilla de realizar. Las herramientas web seleccionadas resultaron ser las más retadoras para el proceso de desarrollo. Con relación al *framework* Laravel fue fácil de entender y es muy potente para el manejo de los datos en el modelo. De igual manera el acceso a las URL a través del enrutador sirvió de gran ayuda para el proceso de implementación. Al principio resultó complejo realizar la integración con el sistema manejador de bases de datos lo cual era indispensable para el acceso a los datos. Posteriormente, fue sencillo de utilizar, Con respecto a Git, fue una herramienta clave para la sincronización del código a lo largo de todo el proceso de desarrollo.

- Desarrollar un módulo para la configuración de proyectos. Este objetivo se cumplió en la iteración 2 del proceso de desarrollo. El sistema le provee al usuario la facilidad de configurar proyectos en la aplicación especificando iteraciones, artefactos de evaluación de usabilidad, actividades y colaboradores del proyecto.

- Se desarrolló el módulo para la gestión y asignación de actividades al grupo de desarrollo como parte de las funcionalidades durante en la iteración 2. La aplicación le permite al usuario crear actividades especificando un período para su realización, asociarlas a categorías para ser filtradas y asignarlas a algún miembro del equipo de desarrollo para su realización

- Se automatizó de manera satisfactoria los seis (6) artefactos de indagación e inspección propuestos: tormenta de ideas, sondeos, análisis de sistemas existentes, evaluación heurística, lista de comprobación y guía de estilos. En la configuración de proyectos el usuario especifica los artefactos que desea incluir, los cuales ya están precargados en el sistema. Se pudo reutilizar el código para la creación y edición de los sondeos, análisis de sistemas existentes y evaluación heurística ya que a nivel de estructura son artefactos similares, solo fue necesario especificar los componentes de cada artefacto de evaluación de usabilidad. Para el desarrollo de la guía de estilos resultó retador la incorporación de un selector de colores (*colorpicker*) que funcionara de manera dinámica, ya que no se puede limitar al usuario a utilizar una cantidad específica de colores debido a que los sistemas pueden variar. La implementación de la tormenta de ideas se logró gracias a la incorporación de un *plugin* hecho con la tecnología jQuery, que permite generar una nube de *tags* a partir de un conjunto de palabras, el *plugin* resultó ser intuitivo y se pudo integrar manera adecuada en el sistema configurando ciertos parámetros convenientemente. Con respecto a la lista de comprobación se logró su desarrollo satisfactoriamente logrando especificar al usuario principios de usabilidad por defecto, de los cuales él puede seleccionar algunos o todos, o si el usuario lo desea puede crear unos personalizados, para así luego proceder a realizar la verificación de la lista de comprobación. Adicionalmente se logró implementar la exportación de cada uno de los artefactos de evaluación de usabilidad.

- Aplicar el método AgilUs para el desarrollo de la aplicación web resultó ser una ventaja ya que es ampliamente conocido por el equipo de desarrollo. Su uso permitió utilizar distintos artefactos enfocados en potenciar la usabilidad en la aplicación lo cual se logró satisfactoriamente, permitiendo crear un software usable para grupos de desarrollo. Algunas actividades que no forman parte del método como por ejemplo la definición de las entidades (anexo *c*) de bases de

datos se realizaron en la etapa del prototipaje ya que era un elemento necesario a pesar de no estar vinculado para la concesión de Usabilidad en la aplicación.

Con la finalidad de extender las funcionalidades creadas en la aplicación web ProAgil, se propone las siguientes recomendaciones para trabajos futuros

- Incorporar nuevos artefactos que permitan evaluar otros aspectos de calidad en proyectos de software. Esto con la finalidad de que la aplicación tenga mayor adaptación a las necesidades del cliente o usuario. Por ejemplo historias de usuario y registro de casos de prueba que forman parte de otros métodos ágiles

- Desarrollar un módulo de administración de la aplicación, que permita configurar la información de los artefactos

- Desarrollar una versión móvil de la aplicación que permita realizar las funcionalidades principales de la aplicación web, con el objetivo de tener acceso a la aplicación desde un dispositivo móvil siguiendo la tendencia actual de desarrollo móvil.

Referencias

Abud, M. (2005). *Calidad en la Industria del Software. La Norma ISO-9126*

Acosta, A. (2013). *AgilUs: una metodología ágil de desarrollo de software que incorpora la evaluación de la usabilidad*

Acosta, A.; Zambrano, N. (2004). *Patterns and Objects for User Interface Construction. Journal of Object Technology. ETH Zurich.* http://bit.ly/WzRcnf. (Visitado en julio 2014)

Acosta, A.; Zambrano, N. (2004). *Patterns and Objects for User Interface Construction*. Journal of Object Technology. ETH Zurich. http://bit.ly/WzRcnf. (Visitado en julio 2014)

Álvarez, M. (2001). *Qué es PHP.* http://www.desarrolloweb.com/articulos/392.php

Apache. Servidor Apache http://www.apache.org/ (visitado julio 2014)

Bahit E. (2011). *POO y MVC en PHP.*

Beck K. y otros. (2001). *Manifesto for Agile Software Development.*

Bradburn, M.;Seymour S., (1998). *Polls and Surveys: Understanding What They Tell Us*

Díaz J; Jiménez S. (2015). *Tópicos para la automatización de artefactos de indagación e inspección para procesos de desarrollo de software. Trabajo de Seminario. Universidad Central de Venezuela*

Diccionario de la Lengua Española. (2008). http://www.rae.es/rae.html. (visitado julio 2014)

Ecma International. http://www.ecma-international.org/ (visitado julio 2014)

Ekas, L (2012). *5 beneficios de las metodologías ágiles en el desarrollo de sistemas de software*

Gabardini J; Campos L. (2004) *Balanceo de metodologías orientadas al plan y ágiles*

Git. Acerca del control de versiones (2013). http://git-scm.com (visitado en septiembre 2014)

Gutierrez D. (2011). *Métodos y procesos de Ingeniería de Software.*

Hwang, W; Salvendy, G. (2010). *Number of People Required for Usability Evaluation: the 10±2 rule.*

ISO/IEC (2014). www.iso.org. (visitado febrero 2015)

jQuery. (2015). https://jquery.com (visitado septiembre 2014)

Korth, H. (2005). *Fundamentos de bases de datos, 4ta edición.*

La Fundación SIDAR - Acceso Universal. *www.sidar.org* (visitado en octubre 2014)

Laravel (2013). *http://laravel.com/docs/4.2* (visitado en julio 2014)

Manchón, E. (S/F). *Diseño centrado en el usuario.*

Manual Oficial PHP. *http://php.net/* (visitado en agosto 2014)

Nielsen, J. (1995). *Ten Usability Heuristics.* *http://bit.ly/WC49x5*. (Visitado en julio 2014)

Pressman (2010). *Ingeniería de software, 7ma edición.*

Ramos, M. (2007). Sistemas Gestores de Bases de Datos. McGrall-Hill

Rodriguez L. (2008). *Corrientes heterodoxas de desarrollo. Madurez o involución*

Solano A. y otros (2009).*Generación de Métodos de Indagación colaborativos para evaluar Usabilidad de software*

Sommerville, I. (2005). Ingeniería de Software, Pearson Educación

Villavicencio M; Díaz O. (2013). *Portal generador de sitios web de asignaturas para la Facultad de Ciencias de la Universidad Central de Venezuela. TEG, Escuela de Computación, UCV*

Viviani, A. (2012). *Desarrollo de una aplicación web para la elaboración automatizada del diseño de los exámenes escritos de los cursos de extensión de la Escuela de Idiomas Modernos de la Facultad de Humanidades y Educación de la Universidad Central de Venezuela. TEG, Escuela de Computación, UCV*

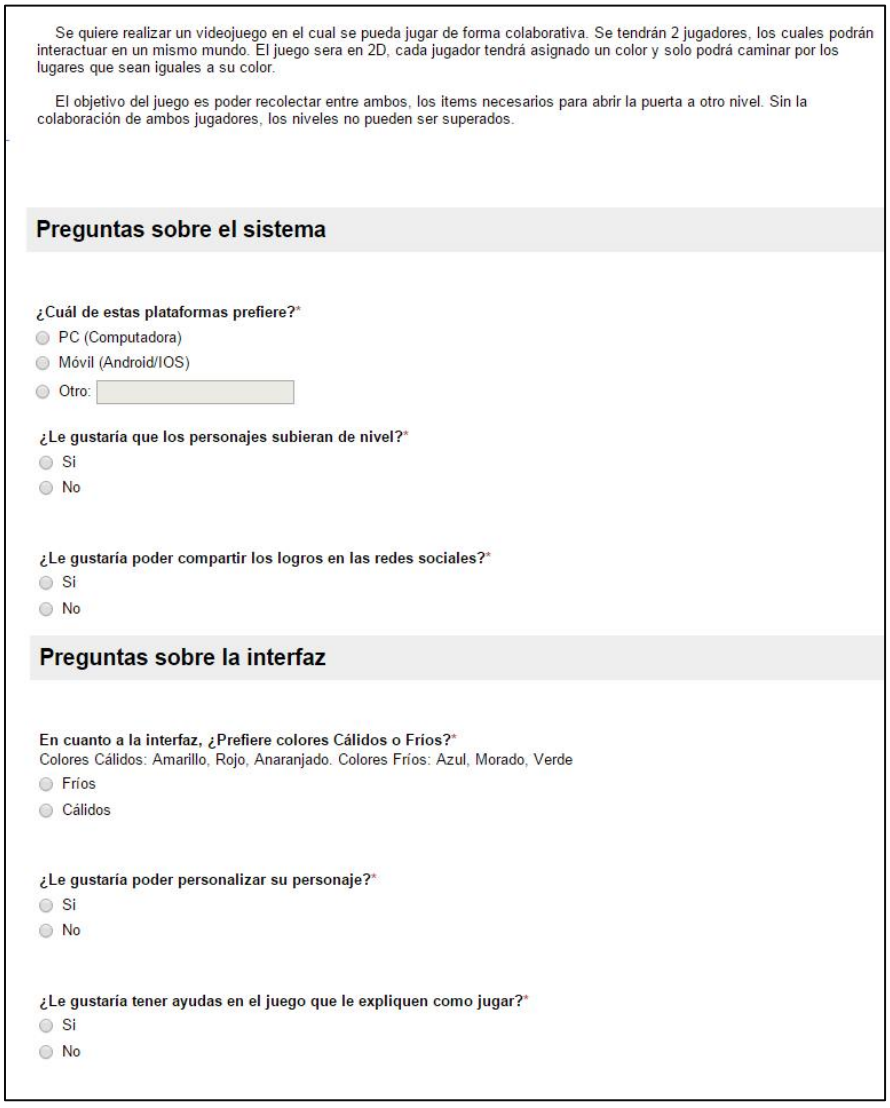
W3 Schools (2014). *http://www.w3schools.com/*. (visitado en agosto 2014).

W3C (2014). *http://www.w3c.es/* (visitado agosto 2014).

Weitzenfeld A. (2008). *Ingeniería de software: el proceso para el desarrollo de software.*

Wesley, A. (2003). *Patterns of Enterprise Application Architecture*

Anexo α – Sondeo para el desarrollo de un videojuego colaborativo



Se quiere realizar un videojuego en el cual se pueda jugar de forma colaborativa. Se tendrán 2 jugadores, los cuales podrán interactuar en un mismo mundo. El juego sera en 2D, cada jugador tendrá asignado un color y solo podrá caminar por los lugares que sean iguales a su color.

El objetivo del juego es poder recolectar entre ambos, los ítems necesarios para abrir la puerta a otro nivel. Sin la colaboración de ambos jugadores, los niveles no pueden ser superados.

Preguntas sobre el sistema

¿Cuál de estas plataformas prefiere?*
- PC (Computadora)
- Móvil (Android/IOS)
- Otro:

¿Le gustaría que los personajes subieran de nivel?*
- Si
- No

¿Le gustaría poder compartir los logros en las redes sociales?*
- Si
- No

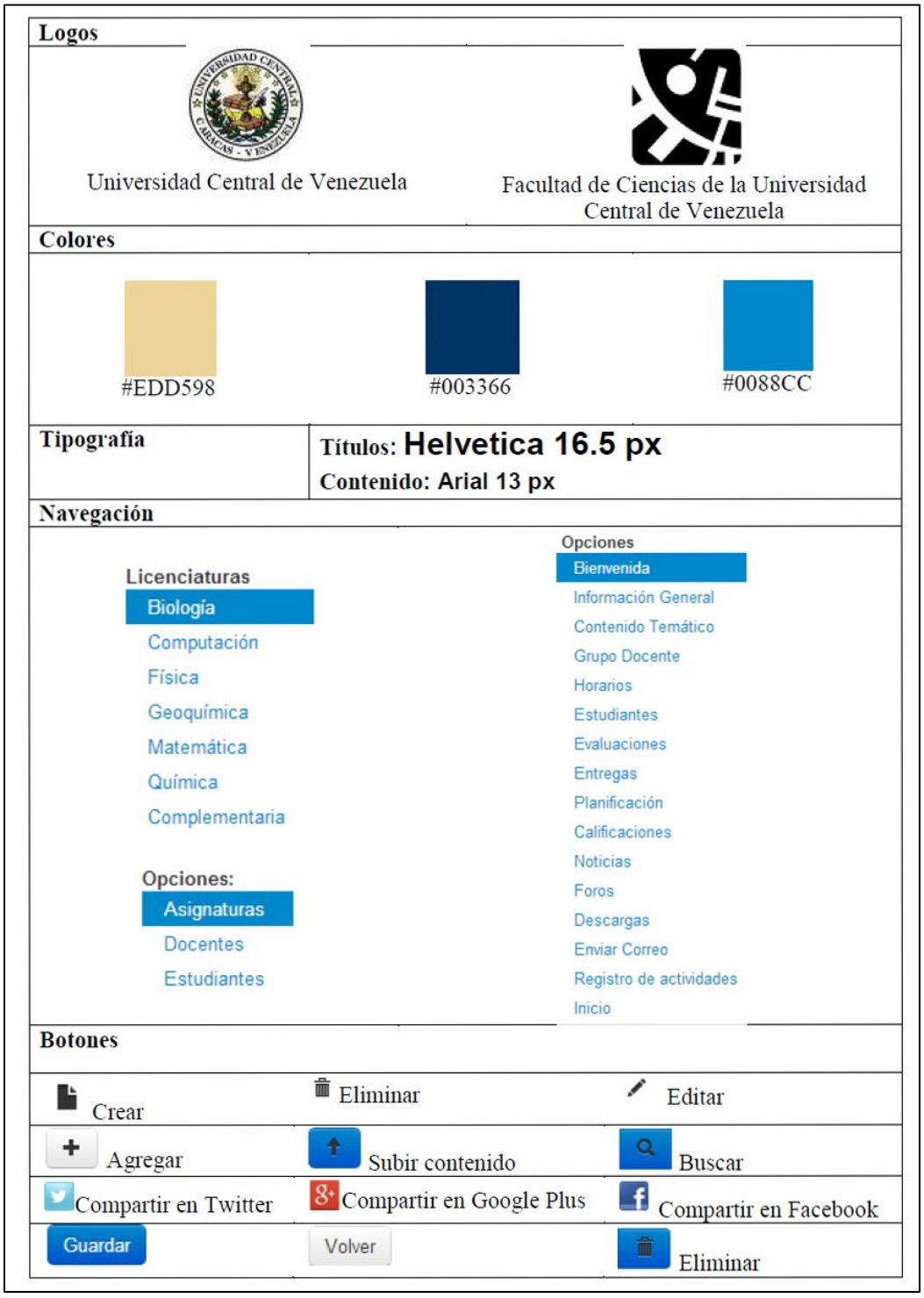
Preguntas sobre la interfaz

En cuanto a la interfaz, ¿Prefiere colores Cálidos o Fríos?*
Colores Cálidos: Amarillo, Rojo, Anaranjado. Colores Fríos: Azul, Morado, Verde
- Fríos
- Cálidos

¿Le gustaría poder personalizar su personaje?*
- Si
- No

¿Le gustaría tener ayudas en el juego que le expliquen como jugar?*
- Si
- No

Anexo *b* – Guía de estilos del sistema PortalAsig[1]



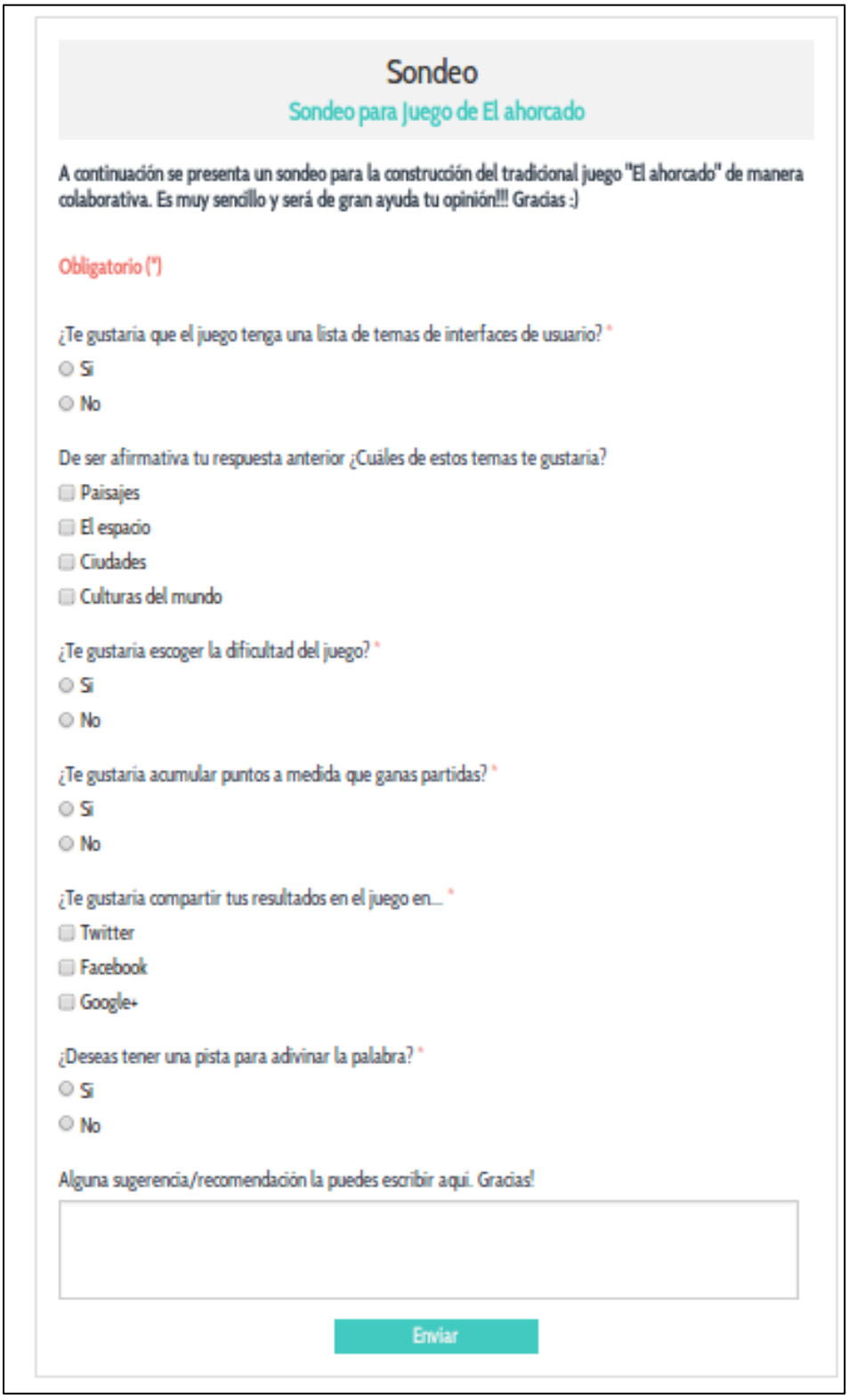
1: PortalAsig – disponible en: http://www.ciens.ucv.ve/portalasig/

Anexo *c* – Documento de definición de entidades de base de datos

Entidad	Atributos
files	id, client_name, server_name
admin_user	id, first_name,last_name, enabled, email, password, type
project_type	id, name, enabled
project	id, name, description, enabled, type_id (web, mobile, stand-alone)
artefact	id, name, description, icon, enabled, friendly_url
artefact_belongs_to_project	id, status, project_id, artefact_id
category_activity_belongs_to_project	id, name, project_id
user	id, first_name, last_name, enabled, password, email, type, avatar, twitter_account, facebook_account, notifications
user_invitation	id, email, project_id, user_role, token
user_role	id, name, enabled
user_belogns_to_project	id, user_id, project_id, user_role_id
activity	id, title, description, date, enabled, status, category_id
activity_belogns_to_project	id, activity_id, project_id, user_id
activity_comment	id, comment, date, user_id, activity_id
existing_system	id, name, enabled, interface
existing_system_topic	id, name, enabled
existing_system_topic_belogns_to_existing_system	id, observation, project_id, existing_system_topic_id, existing_system_id
storm_idea	id, image,, text, enabled, project_id
heuristic	id, enabled, name, description
heuristic_evaluation	id, name, project_id

heuristic_valoration	id, name
heuristic_evaluation_item	id, problem, solution, heuristic_evaluation_id, heuristic_id, heuristic_valoration_id,
comprobation_list	id, enabled, project_id
comprobation_list_item	id, rule, enabled
comprobation_list_item_belogns_to_comprobation_list	id, active, comprobation_list_id, comprobation_list_item_id
probe	id, title, description, status, project_id
probe_template_element	id, question, requerid, form_element, enabled, order probe_id
probe_template_element_value	id, value, probe_template_element_id, probe_template_option_id
probe_template_option	id, enabled, name, probe_template_element_id
style_guide	id, name, logo, interface, project_id
style_guide_color	id, hexadecimal, type, style_guide_id
style_guide_font	id, name, size, style_guide_id

Anexo *d* – Ejemplo de sondeo creado



Sondeo

Sondeo para Juego de El ahorcado

A continuación se presenta un sondeo para la construcción del tradicional juego "El ahorcado" de manera colaborativa. Es muy sencillo y será de gran ayuda tu opinión!!! Gracias :)

Obligatorio (*)

¿Te gustaria que el juego tenga una lista de temas de interfaces de usuario? *

- Si
- No

De ser afirmativa tu respuesta anterior ¿Cuáles de estos temas te gustaria?

- Paisajes
- El espacio
- Ciudades
- Culturas del mundo

¿Te gustaria escoger la dificultad del juego? *

- Si
- No

¿Te gustaria acumular puntos a medida que ganas partidas? *

- Si
- No

¿Te gustaria compartir tus resultados en el juego en... *

- Twitter
- Facebook
- Google+

¿Deseas tener una pista para adivinar la palabra? *

- Si
- No

Alguna sugerencia/recomendación la puedes escribir aqui. Gracias!

Enviar

Anexo *e* – Ejemplo de guía de estilos generada