“SWIRL”, METODOLOGÍA PARA EL DISEÑO Y DESARROLLO DE APLICACIONES WEB

Jimmy Rolando Molina Ríos
María de Las Nieves Pedreira-Souto

Ingeniería y Tecnología

3ciencias

“SWIRL”, METODOLOGÍA PARA EL DISEÑO Y DESARROLLO DE APLICACIONES WEB

Jimmy Rolando Molina Ríos
María de Las Nieves Pedreira-Souto

Editorial Área de Innovación y Desarrollo,S.L.

ÁREA DE INNOVACIÓN Y DESARROLLO, S.L.

C/ Els Alzamora, 17- 03802- ALCOY (ALICANTE) info@3ciencias.com

Primera edición: **septiembre 2019**

ISBN: **978-84-120756-4-9**

DOI: **http://dx.doi.org/10.17993/IngyTec.2019.55**

ÍNDICE DE CONTENIDOS

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

CAPÍTULO I: INTRODUCCIÓN

1.1. Resumen

En el presente libro se describe una guía detallada de las etapas, fases y componentes necesarios para aplicar la metodología de desarrollo web basada en el modelo iterativo, SWIRL. Esta metodología surgió de la fusión de varios conceptos de las metodologías más conocidas dentro de la ingeniería web para el desarrollo de páginas o aplicaciones web de calidad. El proceso de desarrollo iterativo beneficia en gran magnitud a los interesados y al proyecto de software final, las iteraciones que se realizan presentan un gran beneficio en proyectos de aplicaciones web grandes o pequeños. La aplicación y desarrollo de software web se han vuelto más común en el último periodo de tiempo, debido a los beneficios que ofrece a los usuarios finales (personas o empresas) dentro del mercado laboral y profesional. La metodología propuesta busca comprender diversos factores a considerar al momento de realizar un proyecto web, como lo son las etapas más comunes del proceso de desarrollo web, el ciclo de vida en el modelo iterativo, y las buenas prácticas empleadas dentro de la gestión de proyectos de software.

1.2. Introducción

El desarrollo de aplicaciones web es un ámbito el cual ha incrementado en los últimos años, la necesidad de las empresas por incrementar ventas, gestionar de manera ordenadas los registros, y sobre todo superar la competitividad del mercado, ha ocasionado que migren de los sistemas de información tradicionales, como son las aplicaciones de escritorio, a sistemas de información basadas en la web, empleando comúnmente páginas o aplicaciones web.

Al momento de implementar una aplicación web, el programador o equipo de desarrollo suele tener ciertas dudas, ¿cómo diseño la página web?, ¿el proceso que estoy realizando, generará una aplicación de calidad?, ¿la aplicación realizada cumplirá las expectativas del cliente?, ¿cuántos es el tiempo de desarrollo y costo adecuado para esta aplicación? Varias de estas interrogantes son comunes en equipos con y sin experiencia. Para lograr resolver algunas de las interrogantes los jefes de proyecto hacen uso de metodologías de desarrollo, las cuales deben abarcar desde las etapas de análisis hasta la etapa de cierre del proceso de desarrollo de un software.

Antes de comenzar a leer las fases que posee la metodología SWIRL, es fundamental tener un concepto nítido sobre la definición de metodologías de desarrollo web, la cual "es un proceso de desarrollo estandarizado que define un conjunto de actividades, métodos, recomendaciones, valoraciones y herramientas automatizadas que los desarrolladores y directores deben seguir para desarrollar y mejorar de forma continua los sistemas de información" (Alarcón, 2006, p. 44). Es decir, una metodología no es más que el conjunto de métodos y técnicas para lograr un buen desarrollo y obtener el producto deseado.

El uso de metodologías durante el desarrollo de un proyecto de software se ha implementado desde varios años atrás, empezando con las metodologías tradicionales y con el tiempo evolucionando hacia las metodologías híbridas. Con el pasar de los años, las metodologías tradicionales quedaron obsoletas, debido a que no consideraban todos los aspectos del software que podían llegar a perjudicar al desarrollo normal del proyecto. Posteriormente, con la aparición de las metodologías ágiles, se dio más inclusión al usuario, brindando la facilidad de integrar cambios repentinos al software (Domínguez-Mayo, Escalona, Mejías, Ross, y Staples, 2012). Finalmente, debido al incremento en el uso y desarrollo de páginas web, se crearon las metodologías híbridas, las cuales son la mezcla de diversas técnicas y metodologías, para lograr un proceso idóneo y mejorado.

No hay que olvidar que las metodologías se basan en modelos pasados, simplemente agregando la característica necesaria para lograr obtener un software de calidad.

Es por ello, que en el presente libro se propone una nueva metodología de desarrollo para aplicaciones basadas en la web, tomando como enfoque principal las técnicas y procesos empleados en el modelo iterativo.

Este tipo de modelo de metodología, según Brhel, Meth, Maedche, y Werder (2015) software development has been characterized by two major approaches: agile software development, which aims to achieve increased velocity and flexibility during the development process, and user-centered design, which places the goals and needs of the system's end-users at the center of software development in order to deliver software with appropriate usability. Hybrid development models, referred to as user-centered agile software development (UCASD implementa divisiones en las funcionalidades del sistema generando un mayor refinamiento del producto. Para el desarrollo de la metodología propuesta, se investigó la necesidad y factibilidad de una nueva metodología hibrida e iterativo. Una vez constatada la necesidad mediante investigación bibliográfica, se procedió al establecimiento y definición de

cada una de las etapas, mediante la combinación de algunas metodologías ágiles como SCRUM (Satpathy, 2013), XP (Penadés y Letelier, 2006); metodologías híbridas como SNAIL (Molina Ríos, Zea Ordóñez, Redrován Castillo, Loja Mora, Valarezo Pardo, , y Honores Tapia, 2018), y adicionalmente áreas del conocimiento consideradas por la GUÍA PMBOK (Melendez, 2013) para la gestión de proyectos de software.

1.3. Necesidad de una nueva metodología

Es indiscutible la existencia de una gran gama de metodologías para el desarrollo de software, cada una enfocándose en resolver un problema que presentan actualmente las aplicaciones web. Según Molina, et al. (2018) en el desarrollo de software web han surgido diversos problemas como los cambios en las especificaciones de requerimiento, la falta de comunicación y deficiencia de seguridad, y el uso de algunas metodologías no satisface ni resuelve dichos problemas.

Es por esa razón, que se ha creado una nueva metodología híbrida, la cual combina etapas y técnicas de diversas metodologías junto al modelo de desarrollo iterativo, lo cual brinda una mayor flexibilidad por parte de la metodología ante los requerimientos cambiantes del usuario.

Además, el uso de las páginas web como recurso comercial, es más común hoy en día, y las metodologías existentes no brindan la posibilidad de satisfacer las necesidades comerciales, y técnicas SEO para el uso de las aplicaciones dentro del e-commerce.

El desarrollo de esta metodología nace principalmente de la necesidad de contemplar las fases de mayor importancia de otros modelos de desarrollo de software, cruciales para la correcta implementación de software web; conjuntamente de las etapas de modelo de negocios, contemplado como requisito necesario para las empresas contemporáneas, y de lanzamiento de la aplicación.

CAPÍTULO II: FASES DE LA METODOLOGÍA SWIRL

2.1. ¿Qué es la metodología SWIRL?

La metodología SWIRL por sus siglas en español, (Software Web Iterativo Relacional Lógico), es una metodología de desarrollo enfocada en las aplicaciones basadas en la web, que combina el enfoque híbrido e iterativo.

Se basa en la ejecución de iteraciones dentro de sus fases, permitiendo la integración total del usuario durante el proceso, e incorporando la fase de modelo de negocios dentro de su ciclo de vida. Dichas iteraciones constantes simulan la forma de un remolino, de donde nace el nombre de esta metodología.

2.2. Objetivos

- Integrar las técnicas SEO y modelos de negocios.
- Integrar al cliente durante todo el proceso.
- Permitir las modificaciones y mantenimiento posteriores del software.

2.3. Características

- Metodología iterativa basada en validaciones con el usuario.
- Empleada en proyectos grandes o pequeños.
- Reduce los costos de recursos humanos, tiempo, de implementación.
- Los requisitos puedes ser modificados en cualquier etapa del proyecto.
- Los interesados deben estar bien definidos.
- Incluye Joint Application Design (JAD), técnicas SEO y modelo de negocios.

2.4. Ventajas y desventajas

Tabla 1. Ventajas y desventajas de la metodología SWIRL.

Ventajas	Desventajas
Disminuye el tiempo y costo de recursos dentro del proceso de desarrollo.	Se debe terminar una iteración, y ejecutar una revisión para la siguiente.
Entrega constante de resultados.	Los usuarios deben estar definidos.

Ventajas	Desventajas
Es un modelo de desarrollo flexible y adaptable a las necesidades.	Poco adecuada para equipos de desarrollo distribuidos.

Fuente: elaboración propia.

2.5. Modelo SWIRL

La metodología SWIRL se basa en el modelo iterativo considerando en cada iteración cinco criterios inmersos en el desarrollo de proyectos: el costo, tiempo, calidad, alcance y comunicaciones. Cada una de ellas es definida por una de las partes interesadas; es decir, el cliente y el product manager (gerente del proyecto).

El cliente es encargado de definir el alcance y la calidad dentro de la fase de análisis en cada iteración; mientras que el product manager es el encargado de establecer los criterios de tiempo y costo, estos se definen dentro de la fase de planificación. El criterio de comunicaciones es más maleable, es decir, puede ser establecido por el gerente, por el cliente o por ambos.

Por ejemplo: *En la fase de análisis el cliente determina el alcance y la calidad del producto. En la siguiente fase, la planificación, el gerente delimita el tiempo y los costos, además lleva a cabo la primera reunión con el cliente en la cual se coordina el plan de comunicaciones, este puede ser realizado por el gerente y validado por el cliente o viceversa.*

El modelo que emplea SWIRL es presentado y explicado en la Ilustración 1. Cada ciclo de las 5 fases es considerado una iteración, y como en las demás metodologías ágiles, cada iteración debe presentar los entregables correspondientes planteados en la fase de clausura. La fase de lanzamiento no es considerada dentro de cada iteración, debido a que es el proceso de publicación de la aplicación a un servidor web, para lo cual el software debe ser correctamente validado y aceptado por el cliente, por lo cual solo se realiza al finalizar.

Ilustración 1. Modelo SWIRL. **Fuente:** elaboración propia.

2.6. Ciclo de vida

El ciclo de vida de la metodología SWIRL posee 6 fases las cuales son análisis, planificación, modelado, implementación, verificación y pruebas, y lanzamiento. Los ciclos cortos de iteración tienen la finalidad de verificar y corregir los errores que se presenten en la implementación de la funcionalidad actual con periodo de entre 15 días a 1 mes, de duración dependiendo de la dificultad.

Es importante recalcar, que durante todas las iteraciones realizadas el cliente y el equipo de desarrollo deben estar inmersos, de manera que se logre la revisión continua de las funcionalidades implementadas del software durante cada ciclo.

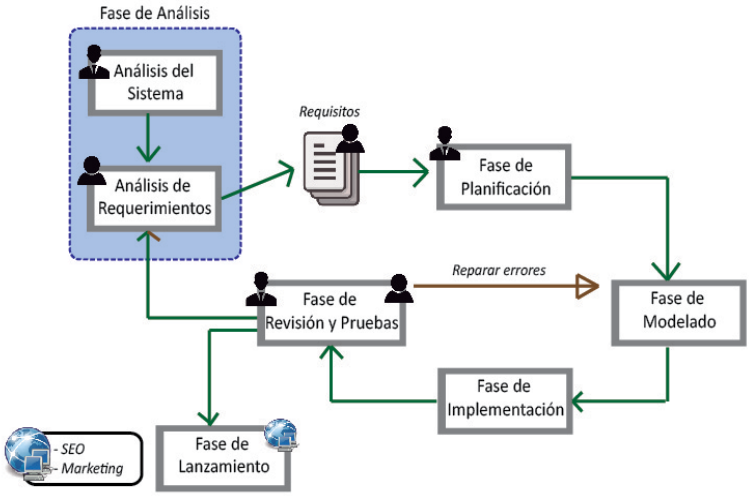

Ilustración 2. Ciclo de vida SWIRL. **Fuente:** elaboración propia.

Los pasos que seguir para el correcto uso de la metodología son:

1. El gerente define el sistema y el modelo de negocios.
2. El cliente delimita los requerimientos y la calidad del software.
3. El gerente planifica las actividades, comunicaciones y entregables de cada iteración.
4. Se diseñan los modelados e implementa las funcionalidades.
5. Se verifica y se realizan las pruebas de control, en caso de encontrar algún error dentro de la implementación se verifican los modelados y se vuelve a codificar.

6. Una vez validado, regresa al punto 2 para la revisión de requisitos y continuar las iteraciones hasta culminar todas las funcionalidades.

7. Finalmente, cuando la aplicación se encuentre verificada por el cliente y sin presencia de errores, se procede al lanzamiento.

2.6.1. Fase I: Análisis

Dentro de esta fase se definen variables relevantes dentro del desarrollo de la aplicación web, como la definición del sistema actual y futuro, el modelo de negocios que presenta la empresa o el cliente, la identificación de interesados, y el análisis sustancial, la elicitación y definición de requerimientos funcionales y no funcionales.

Esta fase, es considerada primordial dentro del ciclo de vida, debido que el proceso de análisis brindará mayor conocimiento sobre la naturaleza de la aplicación a desarrollar. Los elementos que se ejecutan dentro de esta fase son:

Tabla 2. Actividades de la Fase de análisis.

FASE I: Análisis	
1. Definición del sistema	Reconocimiento general del sistema. Estudio de la factibilidad. Análisis de la factibilidad del sistema. Perspectiva del producto. Funcionalidad del sistema. Análisis de riesgos.
2. Modelado de negocio	Descripción del modelo de negocio. Evaluación. Mejora / Innovación (Opcional)
3. Identificación de interesados	Identificación de roles y stakeholders. Definición de las actividades.
4. Requerimientos	Elicitación. Análisis. Validación.

Fuente: elaboración propia.

2.6.2. Fase II: Planificación

Dentro de esta etapa, el gerente debe considerar varios factores para lograr realizar los planes de tiempo, comunicación, riesgos, de calidad, entre otros; necesarios para que el proceso de desarrollo y las iteraciones se puedan realizan de manera sencilla. Las actividades dentro de esta fase son:

- Historias de usuarios.
- Definición de entregables.

- Gestión de cronograma, riesgos y comunicaciones.
- Estructura de desglose de trabajo.
- Velocidad del proyecto y estimación de esfuerzo.
- Gestión de cambios y de la calidad.

2.6.3. Fase III: Modelado

En esta fase se realizan los diseños correspondientes a las bases de datos, modelos conceptuales, y modelos navegacionales. Además, dentro de esta metodología se aplica el uso de diagramas UML (Lenguaje de Modelado Unificado) para el modelado de datos. Las actividades son:

- Diseño del modelo conceptual.
- Diseño del modelo navegacional.
- Diseño de interfaz abstracta de usuario.
- Modelado de diagramas UML.

2.6.4. Fase IV: Implementación

En esta fase se realiza la codificación de cada una de las funcionalidades, según las iteraciones especificadas en la planificación. Se debe emplear una nomenclatura general, independientemente del lenguaje de programación a emplear. Las actividades dentro de esta fase son:

- Codificación.
- Estándares.
- Definición de módulos e implementación.
- Diseño y codificación de páginas principales.
- Interconexiones.

2.6.5. Fase V: Revisión y pruebas

Las funcionalidades que se encuentren implementadas son evaluadas y testeadas para la verificación de su correcta funcionalidad. Adicionalmente en cada iteración se debe realizar el cierre, juntamente con la presentación de entregables.

- Control integrado de cambios.
- Pruebas de integración.
- Detección y corrección de errores.
- Pruebas del sistema y de aceptación.
- Control del alcance.
- Cierre.

2.6.6. Fase VI: Lanzamiento y Marketing

Esta etapa no es considerada dentro de cada iteración, se la realiza al final del proyecto, para ello la aplicación debe ser aceptada por el cliente y no poseer errores de ejecución. Además, en este punto se trabaja en el SEO del portal con la finalidad de posicionarlo correctamente en los buscadores.

- Selección y alojamiento del hosting.
- Preparación del dominio.
- Configuración del certificado SSL (Opcional)
- Campaña Marketing / SEO.

Nota:

Esta fase es adicional a las demás metodologías, con la finalidad de incorporar procesos de marketing y SEO dentro de la aplicación web.

La configuración del certificado SSL es recomendable, ya que brinda al usuario una certeza de seguridad al emplear la aplicación o página web.

CAPÍTULO III: FASE DE ANÁLISIS

Un proyecto de software web requiere mayor esfuerzo que un proyecto de software tradicional. Esto se debe a las complejas y peculiares características que posee una aplicación web, a pesar de comprender etapas similares al desarrollo tradicional, el enfoque y prioridad que se asigna a cada una varía de acuerdo con las necesidades contemporáneas de los proyectos grandes y pequeños de aplicaciones web.

El éxito del desarrollo de un proyecto web radica en las técnicas y herramientas que se aplican durante la elicitación de requerimientos de usuario. A diferencia de las aplicaciones de escritorio, el proceso de diseño y desarrollo web requiere mejorar en la implementación y monitoreo del ciclo de vida completo (Nieves-Guerrero, Ucán-Pech, y Menéndez-Domínguez, 2014).

3.1. Fase de análisis

La fase de análisis dentro de la metodología SWIRL, se enfoca en la definición, elicitación y fundamentalmente análisis del sistema, los procesos de negocio y en gran relevancia, los requerimientos que el proyecto requiere.

Antes de comenzar a recolectar los requerimientos del usuario, es fundamental conocer el negocio y el sistema que desea implementar. Es decir, es indispensable que el gerente defina y estudie el sistema de negocio actual, al igual que el modelado de negocios de la empresa. Este proceso facilita el posterior análisis de los requerimientos funcionales y no funcionales.

Similar a algunas metodologías que emplean herramientas UML dentro del proceso de elicitación de requerimientos, se emplean diagramas de caso de uso, de manera que el equipo logre comprender de mejor manera los requerimientos y necesidades que el usuario posee.

Esta fase consta de 4 subetapas, las cuales buscan resolver los problemas comunes de análisis dentro de la primera fase del proceso de desarrollo. Estas fases son:

- Definición del sistema.
- Modelado de negocio.
- Identificación de interesados.
- Requerimientos.

¿Por qué es importante la etapa de análisis?

Dentro de esta etapa no solo se recolectan y priorizan los requerimientos, antes de esto es necesario definir el sistema a desarrollar; el modelo de negocio que implementa la empresa para la cual se realizará la aplicación web, especificando las fortalezas y debilidades que este brindaría; y finalmente la identificación de stakeholders. Analizar el sistema y sus requerimientos brinda una visión más profunda de los procesos y funcionalidades que el sistema implementará, beneficiando tanto el proceso como el equipo que desarrolla el proyecto.

3.1.1. Objetivos de la fase de análisis

El análisis del diseño y funcionalidades del sistema es definido en 4 subetapas con la finalidad de cubrir los siguientes objetivos:

- Analizar la viabilidad del proyecto a desarrollar en base a sus funcionalidades, restricciones y riesgos.
- Identificar los procesos requeridos dentro del modelo de negocio.
- Definir los roles de las partes interesadas y sus actividades de interacción con el sistema.
- Delimitar los requerimientos relevantes dentro del sistema mediante su priorización y refinamiento.

3.1.2. Estrategias

Algunas buenas prácticas que se pueden aplicar dentro de estas etapas, para obtener resultados exitosos y aplicaciones finales que satisfagan las necesidades del cliente.

1. Identificar la perspectiva del producto conforme la necesidad que posee el cliente.
2. Analizar los riesgos en base a cada una de las funcionalidades que el sistema va a efectuar.
3. Reconocer la empresa y las necesidades que esta posee.
4. Plasmar las actividades del proceso de negocio mediante el uso de diagramas de actividades.
5. Segmentar los roles y tipos de usuarios finales del sistema, definiendo las posibles actividades a realizar.

6. Considerar el análisis FODA en función de las características actuales del negocio y las futuras.
7. Emplear la entrevista como herramienta de elicitación de requerimiento.
8. Organizar jerárquicamente los requerimientos en base a las funcionalidades del sistema.
9. Implementar la matriz de trazabilidad en función de los requerimientos y los casos de usos.

¿Cómo saber si el análisis de requerimientos es correcto?

El proceso de elicitación y análisis de requerimientos cumple un rol fundamental en todo el ciclo de vida, es por ello, que identificar los requisitos correctos es primordial y más aún priorizarlos de manera jerárquica. Este proceso se conoce como ingeniería de requerimientos, para un mayor conocimiento se puede revisar las siguientes referencias:

1) (Cravero, Sepúlveda, López, y Mondéjar, 2013)

2) (Nahuel, Ariste, y Giandini, 2016)

Varios de los problemas que se logran solucionar al aplicar algunas de las estrategias planteadas son:

- Genera aumento en costo, tiempo y talento humano.
- El tiempo de proceso incrementa, y junto al mismo el tiempo de entrega.
- Los requerimientos no se encuentran totalmente definidos, por lo que se presentan constantes cambios.
- El cliente no logra satisfacer la problemática principal.
- Presentación de riesgos no previstos en fases intermedias del ciclo de vida.

Los problemas presentados anteriormente, son los que comúnmente se presentan durante el proceso de diseño y desarrollo de aplicaciones web.

3.2. Definición del sistema

Un aspecto de gran relevancia dentro de la etapa de análisis es el reconocimiento del sistema, desde sus funcionalidades hasta sus usuarios. El conocer el sistema y las características del mismo, ayuda al equipo de desarrollo a tener una idea más clara del diseño que se desea implementar.

3.2.1. Reconocimiento general del sistema

Reconocer de manera global las características principales del sistema es un delimitador necesario antes de comenzar con el proceso de análisis y planificación. El reconocimiento general se realiza en base a las especificaciones del sistema actual que emplea la empresa o el cliente, es decir, el sistema de información de la empresa puede encontrarse regulado por un sistema de software ambiguo, o simplemente no poseer un sistema informático y realizar la gestión de forma manual o en aplicaciones ofimáticas como Excel comúnmente.

Analizar el sistema con el cual se rige la empresa, es crítico para lograr un mejor entendimiento de la problemática que se debe resolver. Los parámetros que considerar para el reconocimiento general del sistema son los siguientes:

- Nombre o razón social.
- Visión y misión.
- Políticas de seguridad (Opcional).
- Organigrama.
- Ubicación geográfica.
- Actividades: dentro de este parámetro se consideran los servicios que oferta la empresa, o las actividades a las que se dedica.
- Perspectiva del sistema actual: se enfoca en el sistema de almacenamiento e información que la empresa utiliza actualmente, especificando las funcionalidades del mismo.
- Reportes / Solicitudes: se especifican los reportes generados dentro de la organización, aún si estos no serán considerados dentro de los requerimientos de la nueva aplicación.

¿Qué sucede si la aplicación no va enfocada a una empresa?

En ciertas ocasiones, las aplicaciones web no son realizadas para cubrir las necesidades de una empresa, en su lugar, son enfocadas a cubrir una necesidad de la sociedad, como puede ocurrir en aplicaciones exclusivamente informativas. En estos supuestos, el reconocimiento se realiza basado en la sociedad, omitiendo parámetros relativos a organizaciones, como son visión y misión, políticas de seguridad, y nombre o razón social. El resto de los parámetros se definen en relación a las características del mercado al cual va dirigido.

3.2.2. Estudio de la factibilidad

Burneo-Valarezo, Delgado Victore, y Vérez (2016), indican que el estudio de factibilidad, es una representación significativa dentro de la gestión de proyectos, teniendo como prioridad la definición del alcance, objetivos y beneficios. El éxito de un proyecto de software no radica simplemente en la elicitación correcta de los requisitos, al contrario, para conocer si el proyecto tendrá éxito se requiere realizar un estudio de factibilidad del sistema informático, de manera que se logre posteriormente tomar decisiones idóneas para el desarrollo del mismo.

Al igual que el desarrollo de otros proyectos, el estudio de factibilidad ayuda al gerente a tomar decisiones referentes a la viabilidad del proceso, y las especificaciones del sistema en algunas etapas de desarrollo.

3.2.2.1. ¿Cómo ayuda al gerente del Proyecto?

Al realizar el estudio de factibilidad el gerente contará con una especificación más clara de recursos requeridos durante el desarrollo del sistema. Mediante este estudio, el gerente será capaz de:

- Identificar y clasificar los usuarios del sistema en función a su actividad.
- Definir los objetivos y beneficios esperados del proyecto.
- Definir el alcance general del sistema, en base a sus restricciones y funcionalidades a desarrollar.
- Conocer el costo total de inversión del proyecto.

3.2.2.2. ¿Qué stakeholders se encuentran inmersos?

Dentro de este proceso simplemente intervienen el gerente del proyecto y cliente.

- Gerente del proyecto: es el encargado de identificar, definir y categorizar las necesidades planteadas por el cliente, en objetivos y alcance del sistema. Además de determinar los recursos y costo de inversión para el desarrollo del proyecto.
- Cliente: es el encargado de informar al gerente las necesidades y problemáticas de la empresa, definiendo las funcionalidades deseadas dentro de la aplicación.

3.2.2.3. ¿Cómo realizo un buen estudio de la factibilidad?

El proceso de un estudio adecuado de factibilidad se estipula por el uso de buenas prácticas, las cuales guían al gerente a determinar de manera sencilla los componentes necesarios para el estudio. A continuación, se plantean algunas buenas prácticas durante el proceso de estudio de factibilidad:

1. Identificar el ámbito en el que se ejecutará el sistema: definir cuáles son los usuarios finales de la aplicación, y clasificarlos en función de la actividad o rol que cumplen dentro de la organización.

2. Plantear los objetivos del proyecto en función a las necesidades: el cometido del sistema debe ser estructurado en relación con el cliente, cubriendo en total las necesidades que plantee de manera que se logre su completa satisfacción. Se recomienda plantear un objetivo general y un mínimo de tres objetivos específicos.

3. Determinar los beneficios en base a los objetivos: el planteamiento de los beneficios esperados debe ser una respuesta positiva a los objetivos propuestos.

4. Delimitar el alcance considerando las restricciones y funcionalidades: el alcance del sistema no es simplemente la especificación de las funcionalidades o características de la aplicación, por el contrario, es necesario informar dentro del alcance las restricciones, que hace y que no hace la aplicación, y estas deben cumplir con los objetivos y beneficios.

5. Calcular el costo de inversión empleando un modelo: los modelos de estimación guían al gerente a calcular y analizar los costos del proyecto en base a diferentes cualidades.

Nota:

Un correcto estudio de factibilidad se realiza en base a la experiencia, adoptando los resultados y experiencias de los proyectos pasado. Para saber si nuestro proyecto es beneficioso es necesario realizar el análisis de factibilidad.

3.2.2.4. Elementos del estudio de factibilidad

Un correcto estudio de la factibilidad consta de indicadores que evalúan características específicas del sistema, de manera que se logre analizar los usuarios, beneficios, funcionalidades del sistema, entre otros. La metodología SWIRL propone seis indicadores para el estudio de factibilidad, los cuales son:

- Usuarios del sistema.
- Beneficios esperados.
- Objetivos del proyecto.
- Alcance del proyecto.
- Costo de inversión.

3.2.2.4.1. Usuarios del sistema

Especificar los usuarios que interactuarán con el sistema ayuda al gerente del proyecto a determinar de manera idónea las actividades específicas que realizara cada uno de ellos. Además, ayuda al equipo desarrollador a diseñar interfaces de usuarios adecuadas según las necesidades de cada uno de ellos.

Los usuarios que considerar dentro de esta estructura son los que interactúan directa o indirectamente con las páginas y sus contenidos de la aplicación web. Para ello, SWIRL define dos niveles de usuarios, en el primer nivel se encuentran los usuarios principales, y secundarios; mientras que, en el segundo nivel se encuentran los tipos de usuarios "suscriptores".

- Usuarios de primer nivel: dentro de este nivel se establecen los usuarios que interactúan directamente con la aplicación, es decir, los que empleen la mayoría de sus funcionalidades. Este nivel es el más común dentro de la gestión de proyectos, debido a que clasifican a los usuarios según las actividades que realicen dentro del sistema.

Dentro de este nivel se establecen dos tipos de usuarios, los usuarios principales y los secundarios.

- Principal: son los que interactúan y tienen acceso a todas las funcionalidades del sistema, como los administradores o superadministradores.
- Secundarios: son los que interactúan directamente con el sistema, pero solo poseen acceso a ciertas funcionalidades del mismo.
- Usuarios de segundo nivel: dentro de este nivel se establecen los usuarios que interactúan indirectamente con el sistema, acceden al mismo por un corto periodo de tiempo y a una o varias funcionalidades en específico.

- Los usuarios de segundo nivel se conocen como los suscriptores, los cuales solo acceden en ocasiones y no administran datos dentro del sistema.

Este nivel de usuarios es opcional, y depende específicamente del tipo de aplicación web que se realice, debido a que existen aplicaciones en las que solo interactúan los usuarios principales y secundarios.

Por ejemplo: *En un sistema web bancario existen tres tipos de usuarios que interactúan con la aplicación, los administradores y gerente, los cajeros, y los clientes. Por consiguiente, se subdividen en dos niveles, dentro de los usuarios de primer nivel se encuentran, los usuarios que hacen uso diario del sistema y gestionan información:*

Principal: *los administrativos y el gerente.*

Secundario: *los cajeros.*

Los usuarios de segundo nivel o considerados como suscriptores son los clientes y personas naturales, debido a que solo ingresan a la aplicación en ciertas ocasiones y durante un tiempo limitado, por ejemplo, para consultar la cuenta (clientes), ver información relevante del banco (personas naturales).

Para la identificación de cada usuario del sistema se establece la siguiente plantilla, en donde el nombre o indicador del usuario se encuentra codificado mediante el uso de la sigla US- (Usuario del sistema), procedido del nombre asignado. Es requerido una plantilla por cada usuario identificado.

Tabla 3. Plantilla para usuarios del sistema.

Usuario:	Se define el nombre del usuario o cargo dentro de la empresa (por ejemplo, US-administrador, US-gerente)
Tipo de usuario:	Identifica la clasificación en la cual se encuentra el usuario <<principal, secundario, o suscriptor>>
Nivel:	Asigna el nivel de usuario en cada uno, indicando el grado de interacción con el sistema <<primer nivel o Segundo nivel>>
Tipo de interacción:	En esta sección se asigna la interacción combinando el tipo de interacción <<directa o indirecta>>, seguido del periodo de tiempo <<temporal o parcial>>
Actividades:	Se deben indicar las actividades que realiza dicho usuario dentro del sistema, es necesario indicar las más significativas para poder comprender claramente los requerimientos posibles.
Contenido de interacción:	Se especifican las posibles páginas web o contenido esperado con el que debe interactuar el usuario.
Habilidades:	Los requisitos con los que debe cumplir el usuario para usar la aplicación web.

Fuente: elaboración propia.

Por ejemplo: *Tomando en cuenta el ejemplo propuesto del sistema bancario se establece la plantilla para el usuario cajero y cliente.*

Tabla 4. Ejemplo usuario de sistema: cajero.

Usuario:	US-cajero.
Tipo de usuario:	Secundario
Nivel:	Primer nivel.
Tipo de interacción:	Directamente temporal.
Actividades:	Ingresar transacciones. Registrar nuevos clientes. Revisión de cuentas de usuarios.
Contenido de interacción:	Página cliente, página transacción, página detalle de la cuenta.
Habilidades:	Tener habilidades de TI. Conocer el funcionamiento básico del sistema.

Fuente: elaboración propia.

Tabla 5. Ejemplo usuario de sistema: cliente.

Usuario:	US-cliente.
Tipo de usuario:	Suscriptor.
Nivel:	Segundo nivel.
Tipo de interacción:	Indirectamente parcial.
Actividades:	Verificación de la cuenta. Visualización de información relativa a la entidad bancaria.
Contenido de interacción:	Página detalle de la cuenta, página principal, página información del banco.
Habilidades:	Tener conocimiento básico sobre uso de navegador web.

Fuente: elaboración propia.

3.2.2.4.2. Beneficios esperados

La realización de un proyecto web posee la finalidad de resolver un problema de gestión de información de alguna empresa o entidad, por lo que su implementación y uso genera beneficios a la misma. El análisis de los beneficios del proyecto es un punto clave para conocer la eficiencia, calidad y sobre todo viabilidad del proyecto planteado. Estos deben estimarse en base al cumplimiento de los objetivos y deben satisfacer favorablemente el alcance del proyecto. Es necesario clasificar los beneficios tangibles e intangibles.

- Beneficios tangibles: son las ventajas obtenidas, las cuales pueden ser medidas de manera monetaria o mediante la satisfacción del cliente. Considerados como una acción que afecta directa y positivamente a la empresa tras la implementación de la aplicación.
- Beneficios intangibles: son ventajas obtenidas y que pueden ser medidas de manera cualitativa, pero no medidos en términos monetarios, por lo que su medición es compleja.

Por ejemplo: *Tomando en cuenta el ejemplo propuesto del sistema bancario se establece la plantilla para la especificación de beneficios esperados.*

Tabla 6. Ejemplo de beneficios tangibles e intangibles.

Beneficios Tangibles	**Beneficios Intangibles**
Automatización de procesos. Mejorar la respuesta a la atención al cliente. Aumentar la eficiencia de la gestión de información y recursos.	Rápido acceso a la información para la toma de decisiones. Incremento de efectividad por parte de la organización. Incremento del control del uso de recursos.

Fuente: elaboración propia.

3.2.2.4.3. Objetivos del proyecto

Estipular los objetivos del proyecto ayuda al gerente del proyecto a definir el alcance, limitaciones y requerimientos bases que la aplicación web debe satisfacer. Determinar los objetivos es un trabajo que requiere la colaboración del gerente de proyecto y del cliente, para ello se aplican reuniones.

Los objetivos para plantear se clasifican en objetivo general y específicos, para ello se debe considerar las siguientes pautas.

- Objetivo general: Debe ser definido en base al resultado final esperado para la aplicación, el cual debe lograr satisfacer las necesidades y resolver el problema que presenta el cliente.
- Se debe establecer un objetivo general contestando las preguntas ¿qué voy a hacer?, ¿cómo lo conseguiré?, y ¿qué intento resolver?

- Objetivo específico: Definido en base a las áreas o requerimientos más importantes a implementar en la aplicación web. Deben ayudar a cumplir el objetivo general, y se establecen como mínimo tres objetivos específicos.

3.2.2.4.4. Alcance del proyecto

El alcance del proyecto consta de la especificación de las funciones que realizará el proyecto, las actividades que no realizará, las limitaciones y en caso de existir dependencias estas también se establecen. Es importante que el alcance se encuentre fundamentado en base a los requerimientos y necesidades del cliente, de manera que cumpla con las expectativas esperadas.

El gerente del proyecto y el cliente deben concordar con el alcance definido, para ello se genera al final de la etapa de Análisis el acta de constitución como un documento entregable, el cual especifica las características que poseerá la aplicación web.

¿Por qué debe estar bien definido el alcance?

Definir un buen alcance es crítico para lograr que el proyecto tenga éxito, debido a que es la base para desarrollar el cronograma, presupuesto y priorización de requerimientos.

¿Quiénes participan?

Al momento de definir el alcance deben intervenir tanto el cliente como el equipo de desarrollo, de manera que todas las partes interesadas comprendan la finalidad, funciones y actividades que la aplicación web ejecutará.

Entonces, es importante que dentro de la definición del alcance se encuentren los siguientes elementos:

- Funcionalidades del sistema.
- Reportes generados (en caso de existir).
- Requerimientos del sistema.
- Tareas y reportes que no realiza la aplicación.
- Limitaciones (opcional).
- Dependencias con otros sistemas (opcional).

3.2.2.4.5. Costo de inversión

El costo del software es un tema de gran importancia durante el desarrollo de un proyecto de software, debido a que el costo generado durante el ciclo de vida debe

ser competitivo en el marcado actual. Como señala Dillibabu y Krishnaiah (2005) which is involved in software development for an embedded system, client-server and Internet environment. The embedded systems group is involved in developing software for major car manufacturers. This study is based on a sample of ten projects, of which eight are development projects and two are porting projects. The actual effort on the projects has been collected from the metrics database of the company. The Lines of code for the various projects have been enumerated using "Code Count" tool to achieve the logical source lines of code for each project. The standard questionnaire has been used to collect the required data to arrive at the various scale factors and effort multipliers. The calibration of the COCOMO (Constructive Cost Model "para un proyecto de software, la estimación comprende el tamaño, el esfuerzo y la estimación del cronograma"(p. 298), por lo que las características a considerar dentro de la estimación son:

- Estimación de tamaño: se hace referencia a las funcionalidades que implementará el sistema, y si el mismo es pequeño o grande. Dentro de esta estimación ingresa el modelo de punto de objeto, el cual consiste en evaluar cada uno de los recursos y componentes del software para realizar la estimación, por ejemplo, interfaces gráficas, componentes, dependencias, informes emitidos, entre otros.

- Estimación de esfuerzo: para ello se consideran el talento humano y las actividades planificadas. Dentro de este suele ingresar la estimación por líneas de código.

- Estimación del cronograma: se evalúa en relación con las actividades a desarrollar y el tiempo establecido para la ejecución de cada una de ellas.

La metodología SWIRL emplea el software COCOMOII para la estimación del costo de inversión. Esta herramienta brinda al gerente una idea más específica del monto económico que se debe invertir para la realización de la aplicación web. Pero debido a que la estimación se realiza en una etapa previo a la planificación los costos serían considerados como supuestos, es decir, la estimación inicial empleada no brinda un monto exacto del costo.

Esta subetapa genera un entregable durante cada iteración del ciclo de vida, esto debido a que debe seguir una serie de fases. Cabe recalcar que durante cada iteración simplemente se realiza la fase 2 de la estimación. SWIRL deja a criterio del gerente el modelo de estimación empleados durante las fases 1 y 2, pero establece el uso del modelo por puntos de objetos y líneas de código en la fase 3.

- Fase 1 de estimación: la estimación se realiza al inicio del proyecto, en la etapa de análisis. Se consideran las posibles actividades y funcionalidades con las que contará el sistema, y debido a que se realizan suposiciones por no contar con una planificación establecida aún, se emplea la estimación pesimista para el análisis de la factibilidad.

El modelo de estimación empleado durante esta fase queda a criterio de gerente del proyecto, pudiendo usar registro de aplicaciones desarrolladas anteriormente para su estimación.

- Fase 2 de estimación: la estimación se realiza después de implementar la etapa de planificación, y se modifica durante cada iteración, generando un nuevo entregable por cada ciclo empleado.

Este monto obtenido es considerado más preciso, por lo que el valor empleado para el análisis es la estimación realista. El modelo de estimación empleado durante esta fase debe ser igual al modelo empleado en la fase 1 de estimación.

- Fase 3 de estimación: esta estimación se realiza durante el cierre del proyecto, es decir, al finalizar el ciclo de vida. Dentro de esta fase SWIRL requiere el uso del modelo de estimación mediante puntos de objetos, y líneas de código empleado en COCOMOII.

Adicionalmente, se genera por defecto el entregable de cierre de estimación de costos, con el modelo empleado en las fases anteriores, es decir, esta fase requiere dos estimaciones.

¿Qué modelo de estimación es el más adecuado?

Existe una gran diversidad de modelos para la estimación del costo de software, cada una se caracteriza por ser empleada bajo ciertas circunstancias, depende del gerente seleccionar una de ellas. El proceso de selección requiere que el gerente posea conocimiento sobre cada uno de los modelos, para un mayor conocimiento se puede revisar las siguientes referencias:

1) (Antúnez Barbosa, Valdovinos Rosas, Marcial Romero, Ramos Corchado, y Herrera Arriaga, 2016)

2) (Tansey y Stroulia, 2007)

El uso de la herramienta COCOMOII es muy útil al momento de estimar el costo de un software por líneas de código, objetos, o tiempo (Kazemifard, Zaeri, Ghasem-Aghaee, Nematbakhsh, y Mardukhi, 2011); dando como resultados tres distintos montos:

- Estimación optimista.
- Estimación más probable.
- Estimación pesimista.

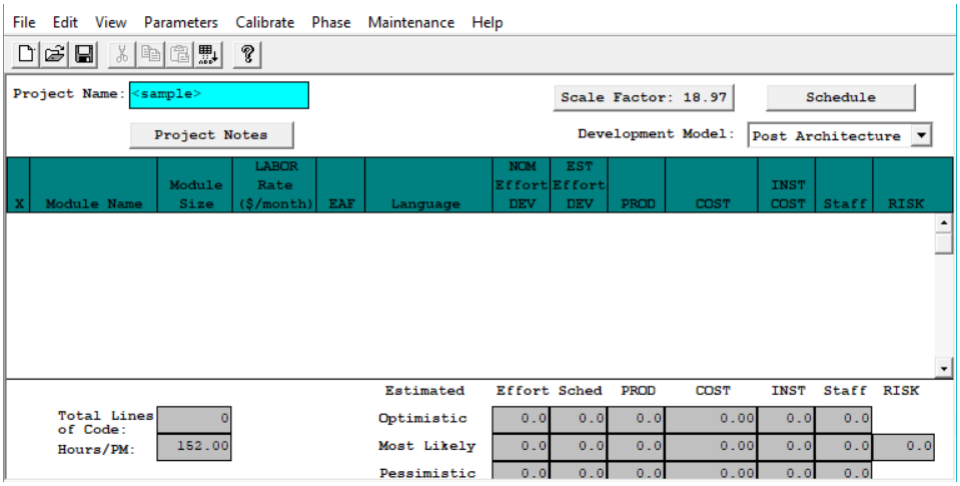

Ilustración 3. Interfaz gráfica de herramienta COCOMOII. **Fuente:** elaboración propia.

3.2.3. Análisis de la factibilidad del sistema

Para considerar favorable la aprobación de un proyecto es recomendable evaluar la factibilidad técnica y económica (Padilla, 2011), no obstante dentro de un proyecto de software es requerido un tipo de factibilidad en específico, el cual es la factibilidad operativa, debido a que el talento humano y el desarrollo del proyecto no se consideran asuntos diferentes.

La fase de análisis de factibilidad hace referencia a la evaluación de los recursos necesarios y que se poseen para el desarrollo del proyecto, y en estimación a los resultados verificar si la ejecución del mismo resulta en pérdida o en beneficio para la empresa desarrolladora. El análisis puede estar estimado en base a el conocimiento y experiencia del gerente del proyecto; mientras que la evaluación respectiva se realiza en base a los elementos que se requieren dentro del proceso de desarrollo.

¿Qué debo hacer si mi análisis es desfavorable?

Dentro de la metodología SWIRL, cuando el análisis de factibilidad del proyecto indica que el desarrollo del proyecto es perjudicial o poco favorable, es recomendable redefinir y mejorar los objetivos y alcance del mismo, conversando con el cliente. Además, elaborar un diseño tentativo que implemente otras alternativas de desarrollo, las cuales beneficien al equipo.

3.2.3.1. ¿Cómo ayuda al gerente del Proyecto?

La toma de decisiones sobre la realización de un proyecto web, es una de las principales responsabilidades del gerente del proyecto. Esto se debe a los diversos factores que se deben considerar para tomar la decisión más adecuada en cuanto al desarrollo del mismo.

Peña, Rodríguez, y Piñero (2016) indican que "el análisis de factibilidad debe documentarse como un informe para la alta gestión. Su resultado pretende determinar de manera objetiva si el proyecto puede continuar o se puede optar por otras alternativas para resolver el problema identificado" (p. 70).

3.2.3.2. ¿Cómo realizo un buen estudio de la factibilidad?

El proceso de análisis de factibilidad se basa en función del conocimiento y experiencia que posea el gerente del proyecto, además, también se puede usar la experiencia a expertos en caso de ser necesario. A continuación, se plantean algunas buenas prácticas a considerar para el proceso de análisis de factibilidad:

1. Identificar todos los recursos, técnicos, recurso humano e informático, empleados para el desarrollo del proyecto.
2. Realizar la estimación de los costos de cada recurso en base al mercado actual, con valores actualizados.
3. Considerar aún si se posee experiencia, la opinión de expertos, esto ayudará a dar mayor seguridad antes de comenzar el desarrollo del proyecto.
4. Determinar los beneficios y posibles pérdidas durante el ciclo de vida completo del proyecto.

Nota:

Un correcto análisis de factibilidad se realiza en base a la experiencia, adoptando los resultados y experiencias de los proyectos pasado. Para saber si nuestro proyecto es beneficioso es recomendable realizar la revisión o consulta a expertos.

3.2.3.3. Elementos del análisis de factibilidad

El análisis de factibilidad se realiza en relación con tres indicadores, la factibilidad económica, la factibilidad técnica, y la factibilidad operativa, generando de esta forma más información al gerente para la toma de decisión.

3.2.3.3.1. Factibilidad operativa

"La factibilidad operativa identifica si el proyecto puede ser operado a través de los recursos de la organización, además de los recursos que participaran en el proyecto" (Molina Ríos, Honores Tapia, y Zea Ordóñez, 2015, p. 55), los sistemas automatizados hoy en día tienen una extraordinaria acogida comparada a épocas anteriores, y todo esto los ha llevado a enfrentarse con la época actual. Sin embargo, existen muchos problemas en los proyectos de software, como por ejemplo retrasos en la entrega y lo más importante problemas de calidad. Esto ha hecho que la ingeniería de software difunda una serie de esquemas para evaluar y mejorar la calidad en muchos aspectos y se busquen procesos de mejora la cual los lleva a plantear procesos, metodologías y mejorar estándares de calidad. Es por ello, que existen muchas herramientas automatizadas que han salido a flote con el único objetivo de ayudar a precisar y emplear un proceso de desarrollo de software seguro. Tal vez suene erróneo, pero a pesar de todas las metodologías existentes, aún existen procesos de desarrollo informales, parciales y muchos de los casos no son confiables. La Ingeniería de Requerimientos (IR).

Es decir, el objetivo de analizar la factibilidad operativa es de comprobar que la empresa que empleará la aplicación web sea capaz de gestionar de manera adecuada el mismo, para ello es recomendable realizar las siguientes preguntas:

- ¿La empresa cuenta con las máquinas y requerimientos de instalación adecuados?
- ¿Los empleado o usuarios finales se encuentran capacitados para hacer uso de la aplicación?
- ¿Se han considerado todas las políticas de la empresa?

Dentro de este punto, se deben establecer todas las condiciones bajo las cuales la aplicación puede ejecutarse, y considerar el comportamiento de los usuarios finales.

Para el análisis de la factibilidad operativa SWIRL hace uso de la siguiente plantilla.

Tabla 7. Plantilla para análisis de factibilidad operativa.

No	Actividad	Priorización
1	Se describe el requerimiento o necesidad considerada (por ejemplo, los usuarios finales necesitan capacitación para su uso).	Se establece el grado de prioridad de la actividad establecida <<Alto, Medio o Leve>>

Fuente: elaboración propia.

3.2.3.3.2. Factibilidad técnica

Es la evaluación del sistema en función de su puesta en marcha y desarrollo, para ello se consideran los recursos de software y hardware empleados durante el proceso de desarrollo del sistema.

Siguiendo a Molina Ríos, et al. (2015), los sistemas automatizados hoy en día tienen una extraordinaria acogida comparada a épocas anteriores, y todo esto los ha llevado a enfrentarse con la época actual. Sin embargo, existen muchos problemas en los proyectos de software, como por ejemplo retrasos en la entrega y lo más importante problemas de calidad. Esto ha hecho que la ingeniería de software difunda una serie de esquemas para evaluar y mejorar la calidad en muchos aspectos y se busquen procesos de mejora la cual los lleva a plantear procesos, metodologías y mejorar estándares de calidad. Es por ello, que existen muchas herramientas automatizadas que han salido a flote con el único objetivo de ayudar a precisar y emplear un proceso de desarrollo de software seguro. Tal vez suene erróneo, pero a pesar de todas las metodologías existentes, aún existen procesos de desarrollo informales, parciales y muchos de los casos no son confiables. La Ingeniería de Requerimientos (IR, "la factibilidad técnica hace referencia a la posibilidad que tiene el proyecto de satisfacer los requerimientos del equipamiento tecnológico con el que cuenta la organización" (p. 56).

Es decir, el gerente del proyecto considera los recursos tecnológicos con los que cuenta la organización y los necesarios para el proceso de desarrollo. Para ello, es recomendable realizar las siguientes preguntas:

- ¿Los recursos de hardware de la empresa son suficientes para comenzar con el proyecto?
- ¿Los recursos de software con los que cuenta la empresa son suficientes para comenzar con el proyecto?
- ¿Hay necesidad de adquirir nuevos recursos de hardware y software?
- ¿Cuáles son las nuevas adquisiciones requeridas?

Para un análisis más resumido y fácil de comprender por las partes interesadas se hace uso de la siguiente plantilla:

Tabla 8. Plantilla para análisis de factibilidad técnica.

Recursos de Hardware		Recursos de Software	
Cant.	***Descripción***	***Cant.***	***Descripción***
1	Se describe cada uno de los recursos de hardware empleados.	1	Se describe cada uno de los programas, herramientas o recursos de software que se requieran.

Fuente: elaboración propia.

Nota:

En caso de ser necesaria la adquisición de recursos adicionales en la descripción es necesario indicar que este no se posee y es necesario su adquisición, para ello se emplea (adquirir).

3.2.3.3.3. Factibilidad económica

Siguiendo a Molina Ríos, et al. (2015), los sistemas automatizados hoy en día tienen una extraordinaria acogida comparada a épocas anteriores, y todo esto los ha llevado a enfrentarse con la época actual. Sin embargo, existen muchos problemas en los proyectos de software, como por ejemplo retrasos en la entrega y lo más importante problemas de calidad. Esto ha hecho que la ingeniería de software difunda una serie de esquemas para evaluar y mejorar la calidad en muchos aspectos y se busquen procesos de mejora la cual los lleva a plantear procesos, metodologías y mejorar estándares de calidad. Es por ello, que existen muchas herramientas automatizadas que han salido a flote con el único objetivo de ayudar a precisar y emplear un proceso de desarrollo de software seguro. Tal vez suene erróneo, pero a pesar de todas las metodologías existentes, aún existen procesos de desarrollo informales, parciales y muchos de los casos no son confiables. La Ingeniería de Requerimientos (IR, indica que la factibilidad económica ”aborda todas las actividades que nos ayudan a identificar los beneficios y costos del proyecto, y a saber si el proyecto es económicamente viable”(p. 56).

Para obtener la factibilidad económica del proyecto se consideran los costos de los recursos humanos, tecnológicos y materiales, y mediante este determinar la viabilidad del proyecto a desarrollar.

Para ello, es necesario identificar por separados los recursos empleados dentro del proyecto y que generan un gasto económico.

A continuación, se presentan las plantillas empleadas para la identificación de cada uno de ellos.

Tabla 9. Plantilla para análisis de recursos humanos.

Recursos Humanos			
Cant.	***Cargo***	***Costo Individual***	***Costo Total***
1	Se establece el cargo que ocupa dentro del equipo de desarrollo <<programador, analista, administrador, tester, etc.>>	Es necesario indicar el costo individual de cada cargo empleado, se recomienda basarse en el mercado actual.	El monto económico generado por todo el equipo dentro del mismo cargo.

Fuente: elaboración propia.

Dentro de la especificación de recursos materiales, se establecen los accesorios necesarios para lograr un buen desempeño durante el proceso de desarrollo de la aplicación.

Tabla 10. Plantilla para análisis de recursos materiales.

Recursos Materiales			
Cant.	***Descripción***	***Costo Individual***	***Costo Total***
1	Se identifican cada uno de los materiales empleados y por emplear durante el ciclo de vida del software.	Es necesario indicar el costo de cada material empleado, se recomienda basarse en el mercado actual.	El monto económico generado por todo los materiales empleados.

Fuente: elaboración propia.

Dentro de los recursos tecnológicos se establecen los recursos de hardware y software.

Tabla 11. Plantilla para análisis de recursos humanos.

Recursos Tecnológicos			
Hardware			
Cant.	***Descripción***	***Costo Individual***	***Costo Total***
1	Se identifican cada uno de los recursos de hardware empleados y por emplear.	Costo de cada recurso usado.	Monto total generado.
Software			
Cant.	***Descripción***	***Costo Individual***	***Costo Total***
1	Se identifican cada uno de los recursos de software empleados y por emplear.	Costo de cada recurso usado.	Monto total generado.

Fuente: elaboración propia

El presupuesto es uno de los puntos fundamentales que se requiere analizar para conocer si el proyecto económicamente es viable o no, formulado en un cierto periodo de tempo y estipulando si los objetivos han sido alcanzados o no, para ello se hace uso del flujo de pagos.

Tabla 12. Plantilla para análisis de flujo de pagos.

Flujo de pagos	
Recursos	Costos
Recursos humanos	Total calculado con anterioridad.
Recursos Tecnológicos	Total calculado con anterioridad.
Recursos Materiales	Total calculado con anterioridad.
Imprevistos (%)	Porcentaje de la sumatoria de los recursos empleados, el porcentaje lo establece el gerente.
Total	

Fuente: elaboración propia

3.2.4. Perspectiva del producto

Plantear que se espera de la aplicación web brinda al usuario final la seguridad de que los requerimientos se cumplirán con satisfacción al final del proyecto. La definición de la perspectiva del producto es realizada mediante la especificación de información relevante del sistema y que ayude a ambas partes interesadas, los usuarios y el equipo de desarrollo, a comprender el funcionamiento de la aplicación web. Para ello, se deben considerar los siguientes puntos durante su definición.

- Las actividades principales que la aplicación web implementará.
- Las restricciones con las que el sistema cuenta.
- En caso de existir validaciones por tipos de usuario o de otra índole se deben especificar.
- ¿De qué manera beneficia al usuario final?
- ¿Qué mejoras generará su desarrollo?

Es importante recalcar, que las perspectivas son planteadas en tiempo futuro y como suposiciones, debido a que son las expectativas que el equipo de desarrollo tiene sobre la aplicación web, y se espera que se cumplan al finalizar el proceso de desarrollo.

3.5.2. Funcionalidad del sistema

Una de las normativas de mayor significancia dentro del desarrollo de aplicaciones web es la especificación de funciones del sistema, es decir, que actividades es capaz de realizar, bajo qué circunstancias o tipos de usuarios, en qué periodo de tiempo se espera una respuesta. Al momento de especificar las características anteriormente especificadas, el desarrollador bosqueja una idea de un diseño tentativo de la aplicación, considerando las diversas condiciones o restricciones que el cliente especifica.

¿No es lo mismo que describir la perspectiva del sistema?

La funcionalidad del sistema difiere de la perspectiva, en la definición de actividades. Cuando se define la perspectiva del sistema, se establecen acciones esperadas del sistema que logren satisfacer al cliente y cumplir con los objetivos establecidos. Mientras que, al establecer la funcionalidad del sistema se definen las actividades que la aplicación web realizará, no necesariamente son respuesta a los objetivos, pueden ser indiferentes.

3.2.5.1. ¿Por qué establecer las funcionalidades de la aplicación?

El gerente del proyecto tiene la obligación de definir las funciones y actividades que realizará el sistema, de manera que, tanto el cliente como el grupo de desarrolladores comprenda claramente el sistema. Algunos beneficios que ofrece la correcta definición de las funcionalidades de la aplicación web para los clientes son:

- Posee una idea del producto final.
- Conoce una breve explicación del manejo del sistema.
- Conoce los reportes que gestiona y las decisiones que se pueden tomar en base a los mismos.
- En futuras revisiones, el cliente identificará de manera rápida los posibles cambios o nuevos requerimientos, debido a que conoce cuales han sido efectuados.

Nota:

Una buena práctica al momento de definir las funcionalidades es entablar una reunión con el cliente definiendo las necesidades, y una reunión con los desarrolladores con la finalidad de definir las actividades básicas del sistema.

Algunos de los beneficios que posee el grupo de desarrolladores al identificar todas las funcionalidades de la aplicación son:

- Facilidad para diseñar prototipos de interfaz de páginas.
- Facilidad durante la etapa de codificación de la aplicación.
- Mayor conocimiento en los modelos para testear la aplicación.
- Conocer el funcionamiento completo del sistema.

3.2.5.2. ¿Cómo establecer las funcionalidades del sistema?

Algunas buenas prácticas empleadas para describir las funcionalidades de la aplicación sin omitir las de mayor importancia son:

1. Identificar los módulos del sistema.
2. Determinar cada una de las posibles actividades desarrolladas por cada uno de los tipos de usuarios.
3. Establecer los reportes o archivos generados por la aplicación para la ayuda en la toma de decisiones del cliente.
4. Definir las restricciones en caso de ser necesario.

3.2.6. Análisis de riesgos

3.2.6.1. ¿Cómo ayuda al gerente del Proyecto?

Mantener una gestión de los riesgos dentro de un proyecto cumple el cometido de incrementar el impacto de los eventos positivos y minimizar la probabilidad de eventos que afecten negativamente el proyecto (Arteaga, Barrera, y Chaparro, 2013). Lo cual, permite al gerente del proyecto tener conocimiento de la probabilidad de acciones que puedan afectar el proceso de desarrollo, permitiendo planificar un cronograma eficiente para solucionar y manejar correctamente los riesgos, sin que afecte de manera grave el proyecto.

3.2.6.2. ¿Por qué analizar los riesgos del proyecto?

Analizar los riesgos del proyecto es la primera etapa dentro de la gestión de riesgos de la cual se encarga el gerente del proyecto. El análisis nace de la respuesta a la necesidad que posee el gerente para entender el ámbito en el que se desarrollará el proyecto, y los posibles eventos que afecten de manera negativa, el ciclo de vida normal. Para ello, Martínez (2007) indica que los altos mandos consideran tres

componentes fundamentales del riesgo, los cuales son ¿algo malo puede ocurrir?, ¿qué posibilidad hay que suceda?, ¿cuáles son las consecuencias que puede generar?

¿Qué sucede si solo se consideran riesgos superficiales?

El proceso de análisis de riesgos debe poseer un enfoque profundo en los eventos que se pueden presentar durante el desarrollo del proyecto. Si solo se consideran riesgos poco relevantes o superficiales, la planificación y actividades delegadas pueden llegar sufrir retrasos. Es por ello que se recomienda considerar los eventos pesimistas, y poco pesimistas.

3.2.6.3. Elementos del análisis de factibilidad

Es indiscutible que la gestión de riesgos se presenta en diferentes etapas, las más importantes se presentan en las etapas de análisis y planificación. Esto se debe a que durante las primeras etapas los riesgos solo se determinan y analizan de manera cuantitativa y cuantitativa; por otro lado, dentro de la etapa de planificación se detallan actividades y cronogramas para lograr prevenir los riesgos determinados anteriormente. Dentro de la etapa de análisis se establecen los siguientes elementos:

- Desafíos del proyecto.
- Identificación de los riesgos.
- Análisis cuantitativo y cualitativo.
- Limitaciones, restricciones, supuestos y dependencias.

3.2.6.3.1. Desafíos del proyecto

Una característica peculiar que menciona Rodríguez (2011) es que "los riesgos afectan a las dimensiones básicas del proyecto, es decir, las especificaciones, los costes o la duración del trabajo" (p. 71). Con ello, se presentan desafíos que son necesarios de sobrellevar para lograr concluir de manera exitosa el proceso. Para lograr definir cuáles son los desafíos que se presentan en el proyecto, SWIRL propone la descripción de los desafíos en base a las etapas y tipos de usuarios. Se debe realizar la plantilla de identificación por cada desafío identificado.

Por ejemplo: *Siguiendo con el ejemplo propuesto del sistema bancario se establecen los siguientes desafíos:*

- Entendimiento con los usuarios

- Usabilidad de la interfaz de usuario.

- Estimación adecuada de los recursos

Tabla 13. Plantilla de identificación de desafíos.

<<D-01>>	Se define un indicador para el desafío.
Descripción:	Se describe el desafío considerando el usuario, la acción y evento en el cual se presenta.
Etapa de desarrollo:	Etapa del proceso de desarrollo en el que se puede presentar el desafío, en caso de ser implementación o pruebas se especifica el módulo.
Módulo:	Especifica el módulo o funcionalidad en el cual se presenta el desafío. Este indicador es opcional.
Tipo de usuario:	El o los tipos de usuarios que se encuentran vinculados con el desafío.
Grado de afectación:	Presenta de manera gradual la intensidad de amenaza, en cuestión de dinero y tiempo. <<Inferior, Medio, Superior>>
Grado de solución:	Presenta de manera gradual la intensidad de facilidad de la posible solución. En base al costo de su ejecución. <<Inferior, Medio, Superior>>

Fuente: elaboración propia.

3.2.6.3.2. Identificación de los riesgos

Definir los riesgos que se pueden presentar en el proyecto según la experiencia del gerente del proyecto y en base a colaboración con expertos se listan y analizan. Es importante conocer a que se considera riesgo dentro del proyecto, para lo cual se toma la definición de Huidobro, Heredia, Salmona, y Alvarado (2009), el cual menciona que al hablar de riesgos "se hace referencia a eventos que de ocurrir pueden impactar positiva o negativamente sobre el proyecto"(p. 30).

Durante la identificación de riesgos se deben especificar ciertos parámetros para cada uno de ellos, lo cual brinda al gerente del proyecto una idea más clara al momento de realizar la planificación.

Tabla 14. Plantilla de identificación de desafíos.

Riesgo No:	**<< RG- >>**		**Fecha:**		
Descripción:					
Etapa	**Categoría**	**Stakeholders afectados**	**Actividad**	**Probabilidad**	**Impacto**

Fuente: elaboración propia.

Un riesgo puede presentarse en varias etapas del ciclo de vida, por lo que debe especificarse el grado de impacto y la probabilidad respectiva dentro de cada etapa.

- << Riesgo No >>: Expresa el indicador del riesgo que se está indicando, sirve para identificar el riesgo en futuras tablas.
- << Fecha >>: Se ingresa la fecha actual en la cual se consideró el riesgo.
- << Descripción >>: Breve descripción del riesgo considerando que abarque de manera general el mismo.
- << Etapa >>: Se indican todas las etapas en las cuales el riesgo se pueda presentar.
- << Categoría >>: Se establece el tipo de riesgo fundamentado en la etapa, las categorías las asigna el gerente, por ejemplo, una categoría considerada sería el alcance; ya que el riesgo afecta directamente a la definición del alcance.
- << Stakeholders afectados >>: Se enlistan las partes interesadas afectadas por el riesgo durante la etapa designada.
- << Actividad >>: Se especifican las actividades que pueden provocar la aparición del riesgo.
- << Probabilidad >>: Este valor se expresa en porcentaje (%), y es estimado por el gerente del proyecto. Representa la posibilidad que el riesgo ocurra durante una cierta etapa.
- << Impacto >>: Este valor al igual que la probabilidad se mide mediante un valor numérico, con la diferencia que es decimal y va de un rango entre 0 a 1. La respuesta del proyecto ante este riesgo.

Por ejemplo: *Haciendo uso del ejemplo propuesto del sistema bancario se plantean como ejemplo 2 riesgos.*

- El usuario no encuentra fácil de aprender la interfaz.

- El grupo de desarrollo desconozca las herramientas.

Tabla 15. Ejemplo1 de identificación de riesgo.

Riesgo No:	RG-1		**Fecha:**	29/05/2019	
Descripción:	Diseño ineficiente de interfaz de usuario				
Etapa	**Categoría**	**Stakeholders afectados**	**Actividad**	**Probabilidad**	**Impacto**
Modelado	Diseño	Cliente, Equipo de desarrollo	Al diseñar los prototipos de las Interfaces.	45%	0.75
Pruebas y Revisión	Diseño	Cliente	Al revisar el diseño en las reuniones, el cliente no comprende el uso de la app.	28%	0.30

Fuente: elaboración propia.

Tabla 16. Ejemplo2 de identificación de riesgo.

Riesgo No:	RG-2		**Fecha:**	29/05/2019	
Descripción:	Desconocimiento de las herramientas de desarrollo.				
Etapa	**Categoría**	**Stakeholders afectados**	**Actividad**	**Probabilidad**	**Impacto**
Implemen-tación	Codifica-ción	Equipo de desarrollo	Al programar la aplicación el equipo no conozca el lenguaje o las herramientas	16%	0.45

Fuente: elaboración propia.

¿Cómo evalúo el porcentaje y el impacto?

Ambos valores son obtenidos mediante técnicas de estadísticas, experiencia, opinión de expertos, entre otros. La probabilidad indica la posibilidad que el riesgo ocurra; mientras que el factor indica que tan grave, o tan leve, serían las consecuencias al ocurrir dichos riesgos.

3.2.6.3.3. Análisis cuantitativo y cualitativo

Analizar cuantitativa y cualitativamente los riesgos es de gran ayuda para el gerente, debido a que mediante este logra conocer los riesgos con mayor posibilidad de ejecución, y los que indican mayor efecto sobre el proyecto.

- Análisis cualitativo: para este análisis se evalúa la probabilidad y el impacto del riesgo, con la finalidad de obtener la magnitud y prioridad (Del Carpio Gallegos, 2006).

- Para ello se pueden emplear herramientas como matriz de probabilidad e impacto, evaluación del juicio de expertos, entre otros.

- Análisis cuantitativo: para este análisis se emplea el análisis obtenido en las variables cualitativas, considerando el impacto y probabilidad como variables numéricas.

Dentro del análisis los riesgos son clasificados dentro de tres grupos, los cuales abarcan las categorías definidas durante la etapa de identificación. Las categorías son:

- Riesgos técnicos, de calidad o rendimiento: dentro de este se relacionan los problemas con los recursos tecnológicos, diseños del sistema y problemas generales en el desarrollo de la aplicación.

- Riesgos en la gerencia de proyectos: riesgos relacionados con ámbitos de planificación, y análisis, sobre todo, los cuales logran afectar directamente al gerente o son ocasionados por ellos.

- Riesgos del sistema en relación con el usuario: riesgos basados en las reuniones con el usuario, y fundamentadas en las necesidades y modificaciones provistas por el cliente. Pueden incluirse dentro del mismo, riesgos en actividades del cliente-gerente, o cliente-grupo desarrollador.

Para ello, SWIRL hace uso de la siguiente plantilla que permite analizar cuantitativa y cualitativamente los riesgos definidos en la sub-fase anterior.

Tabla 17. Plantilla de análisis cuantitativo y cualitativo.

Grupo de riesgos	**Riesgo No**	**Análisis cuantitativo**		**Análisis cuantitativo**	
		Probabilidad	***Impacto***	***Magnitud***	***Prioridad***
Riesgos técnicos, de calidad o rendimiento					
Riesgos en la gerencia de proyectos					

Grupo de riesgos	Riesgo No	Análisis cuantitativo		Análisis cuantitativo	
		Probabilidad	*Impacto*	*Magnitud*	*Prioridad*
Riesgos del sistema en relación con el usuario					

Fuente: elaboración propia.

Los parámetros considerados para el análisis respectivo son:

- << Grupo de riesgos >>: son las clasificaciones emitidas dentro de la metodología.
- << Riesgo No >>: permite identificar fácilmente el riesgo anteriormente definido.
- << Probabilidad >>: Este valor se expresa en porcentaje (%), y es estimado por el gerente del proyecto. Representa la posibilidad que el riesgo ocurra durante una cierta etapa.
- << Impacto >>: Este valor al igual que la probabilidad se mide mediante un valor numérico, con la diferencia que es decimal y va de un rango entre 0 a 1. La respuesta del proyecto ante este riesgo.
- << Magnitud >>: Se calcula mediante el establecimiento del impacto dentro de un rango. El impacto entre 0 – 25% son considerados (Muy leve); entre 0,26 – 0,50 (Leve); 0,51 – 0,75 (Moderado); 0,75 – 1 (Severo)
- << Prioridad >>: Se calcula mediante el establecimiento de la probabilidad dentro de un rango. El impacto entre 0 – 25% son considerados (Muy baja); entre 26% – 50% (Baja); 51% – 75% (Alta); 75% – 100% (Muy Alta)

3.2.6.3.4. Limitaciones, restricciones, supuestos y dependencias

Las limitaciones y restricciones, supuestos y dependencias, se establece con la finalidad de que tanto el cliente como el gerente conozcan las tareas que la aplicación es y no es capaz de hacer.

Limitaciones:

- Expresan las actividades límite o tope que puede realizar, para una buena especificación de estas, se debe considerar las actividades que no realiza el programa.

Restricciones:

- Son las condiciones o requerimientos tanto para el uso de la aplicación como para la instalación de la misma. Estos se establecen mediante el gerente del proyecto.

Supuestos:

- Es la especificación del funcionamiento de cada módulo, de manera que se describa la tentativa planteada. Se pueden realizar supuestos para el producto, para los usuarios e inclusive para el equipo de desarrollo.

Dependencias (opcional):

- Este punto se considera en aplicaciones que deben vincularse con navegadores, sitios web, u otras aplicaciones.

3.3. Modelado de negocios

El modelo de negocios es considerado por algunas metodologías de desarrollo web como una técnica de entendimiento de los procesos de negocio de la organización del cliente, sin mencionar, que logra describir en base a la visión de la organización, las actividades incluidas en el proceso, las responsabilidades y roles de cada usuario del mismo (Baum, Daniele, Martellotto, y Novaira, 2004).

¿Cómo hacer un modelo de negocios?

El modelado de negocios es un tema que no interviene únicamente en los proyectos de software, es empleado en otros ámbitos como finanzas, marketing, entre otros. Sin embargo, se puede hacer uso de algunos modelos, patrones y estructuras de otros modelos de negocios, para un mayor conocimiento se puede revisar las siguientes referencias:

1) (Marciszack, Castro, Sánchez, y Delgado, 2016)

2) (Rodriguez, Bazán, Ambrosi, y Diaz, 2016)

Para lograr realizar un buen modelado de negocios, es fundamental que se entienda la definición por completo, un modelo es la representación gráfica de un objeto empleado dentro del proceso de negocio o actividades comúnmente realizadas; mientras que negocio es la actividad que permite generar ingresaos a una organización. En base a lo explicado, se concluye que un modelo de negocios consiste en la definición de las actividades comunes que realiza la organización, detallando las siguientes preguntas:

- ¿Cuál es la organización de la empresa?

- ¿De qué manera interactúan las personas o actores dentro de la misma?
- ¿Qué objetivos busca cumplir?
- ¿Cuáles son las actividades que realiza la empresa?
- ¿De qué manera son efectuadas las actividades?

3.3.1. Fase de descripción

3.3.1.1. Identificación del proceso de negocio y mantenimiento

Dentro de esta etapa de modelado se requiere el diseño de modelos que logren convertir al sistema en modular, mantenible, y reusable para posteriores modificaciones. Una característica peculiar del modelo de negocio es que busca alcanzar los objetivos y alcance definidos mediante la separación en actividades específicas, de ahí proviene la identificación del proceso de negocio (Pinciroli y Zeligueta, 2017).

El modelo empleado para identificar los procesos de negocios son los diagramas de flujo, el uso de los mismos permite no solo identificar las actividades que se ejecutan durante los procesos, sino brinda un conocimiento de que actividades realizar al momento que exista alguna modificación. El uso de diagramas de flujos permite a las partes interesadas comprender e identificar las actividades que requieren modificaciones durante las etapas de mantenimiento.

En el diseño de mapas de procesos se emplean tres componentes:

- Componente de inicio: se representa mediante un rectángulo con puntas ovaladas, y comúnmente se especifica el proceso de negocio a realizar.
- Componente de actividades: se representa mediante un rectángulo, y en él se especifican todas las actividades que la organización realiza dentro de un proceso específico.
- Componente de decisión: se representa mediante un rombo, y permite identificar las actividades alternativas en caso de existir modificaciones o decisiones durante el proceso de negocios.

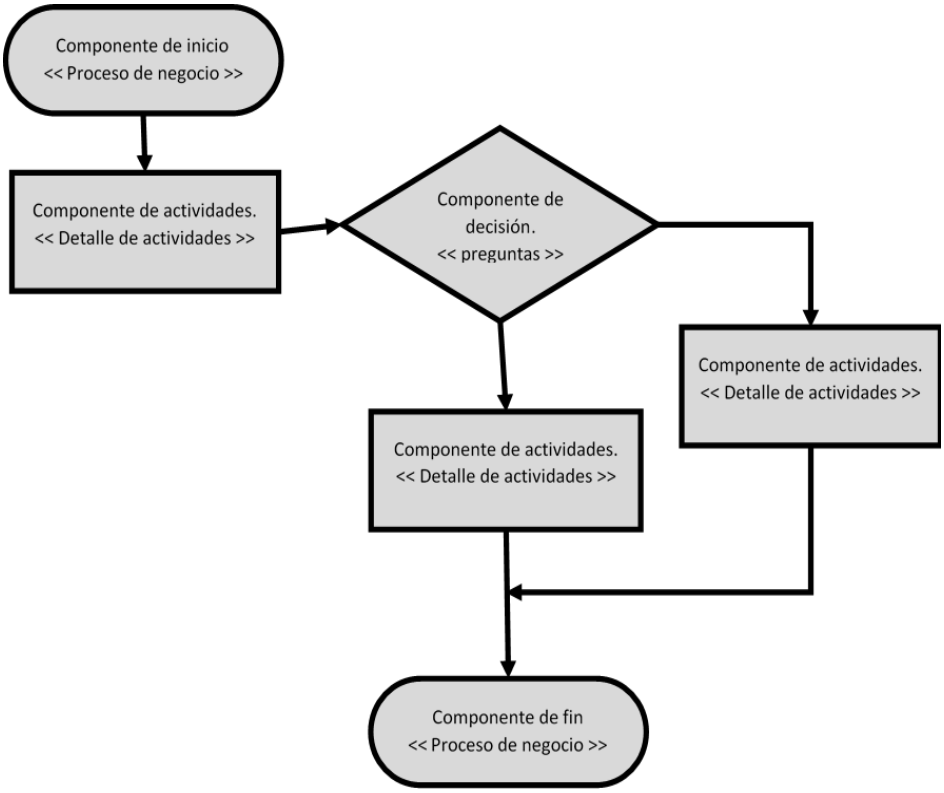

Ilustración 4. Plantilla de mapas de procesos. **Fuente:** elaboración propia.

El mantenimiento se realiza en base al mapa de proceso especificado, para ello se debe realizar lo siguiente:

- Identificar el proceso de negocio en el que se desea realizar el cambio.
- Determinar las actividades que pueden incurrir en la modificación propuesta.
- Emplear el mapa de proceso para identificar las actividades alternativas que se pueden emplear.
- En caso de no presentarse actividades alternativas es necesario redefinir el mapa de procesos considerando las nuevas modificaciones realizadas.
- Ejecutar las actividades identificadas.

3.3.1.2. Modelado de usuario final

Los usuarios que intervienen en el modelo de negocio son conocidos comúnmente como actores, estos permiten identificar las actividades que son requeridas dentro del proceso de negocios, y especificando las interacciones que estas poseen con cada uno de los tipos de usuarios.

Empleando el modelo de dependencias estratégicas, se obtienen conocer las dependencias de las actividades de negocio y los actores que la ejecutan. Para ello se consideran tres tipos de actores, el actor que es dependiente; el actor del cual se depende en las actividades; y finalmente el actor del cual se centra la relación de dependencia (Estrada, Martínez, Pastor, y Sánchez, 2002). Los trabajos de investigación actuales en ingeniería de requisitos buscan mecanismos que permitan establecer la relación entre la funcionalidad esperada de un sistema de información y los procesos de negocios a los que éste dará soporte. Este enfoque permitirá asegurar que el sistema de información a desarrollar sea realmente útil en las tareas de los actores organizacionales. Los trabajos de investigación en esta área han determinado que las metas organizacionales son una buena base para establecer la relación entre los objetivos perseguidos por el negocio y los requisitos del sistema de información a desarrollar, ya que todos estos requisitos (funcionales y no funcionales.

El modelado de usuario final es una característica importante que considerar durante el desarrollo de una aplicación web, ya que definir qué usuarios interactuarán de manera interna con la aplicación, y que actividades realizan dentro de la organización ayuda a comprender el posible prototipo de diseño de interfaz del software.

Para ello el modelo se compone de los siguientes elementos:

- Red de contactos.
- Segmentación de usuarios.
- Propuesta de valor.

3.3.1.2.1. Red de contactos (opcional)

La red de contacto es un diagrama que permite identificar las relaciones entro los actores identificados en el proceso de negocio, para ello se hace uso de los grafos de usuarios. Dentro de esto se identificarán dos elementos:

- Nodos: serán los usuarios y actores dentro del proceso de negocio, es preferible que se abrevie lo más posible, y se determinen todos los que sean necesario, desde los dependientes, hasta los intermedios.
- Relaciones: representa la forma de interactuar entre los usuarios, o nodos, es indispensable que se establezca el método o forma de comunicación en la parte superior de la relación.

Nota:

El diseño de la red de contactos es una etapa que puede ser omitida, es decir su diseño dentro del proceso de desarrollo mediante la metodología SWIRL es opcional. Este diagrama se realiza con la finalidad de facilitar el entendimiento del proceso de negocio, y la posterior segmentación de los usuarios dentro de los procesos de negocios. Además, de permitir al gerente del proyecto conocer los usuarios finales a los cuales va dirigida la aplicación web.

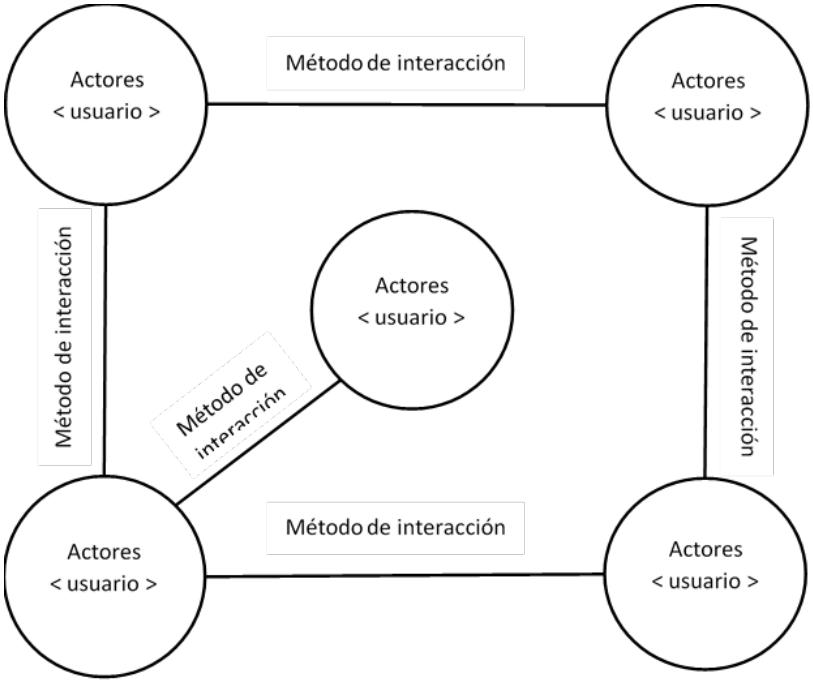

Ilustración 5. Plantilla de mapas de procesos. **Fuente:** elaboración propia.

3.3.1.2.2. Segmentación de usuarios

La segmentación de usuarios consiste en la identificación de las categorías y niveles de usuarios que la organización identifica dentro del proceso de negocio, para ello se debe definir las categorías principales, y posteriormente identificar los usuarios y roles que ingresan dentro de cada una de estas.

Para la segmentación se especifican los siguientes parámetros:

- Segmentación.
- Tipo de interacción con el sistema.
- Los usuarios que se encuentran inmersos en la categoría.
- Los roles y actividades que desempeñan dentro de la aplicación.
- Los objetivos estratégicos que se busca alcanzar.
- Los procesos de negocios relacionados o efectuados debido a los usuarios de la categoría en específica.

Tabla 18. Plantilla de segmentación de usuarios.

Segmentación	Se identifica la categoría de usuario definido dentro de la organización.
Tipo de interacción:	El grado con el que estos usuarios intervienen en el sistema puede ser <<interacción directa o indirecta>>
Usuarios:	Se identifica cada uno de los usuarios que intervienen dentro de esta categoría.
Roles:	Determina que actividades realizan cada uno de los usuarios dentro esta categoría.
Objetivos estratégicos:	Se plantean los objetivos que buscan alcanzar esta categoría de usuarios.
Proceso de negocio:	En esta sección se relacionan los procesos de negocios en los cuales intervienen las actores.

Fuente: elaboración propia.

3.3.1.2.3. Propuesta de valor

La propuesta de valor consiste en la definición de las actividades realizadas mediante el uso de diagramas de caso de uso. Se requiere la creación de una tabla descriptiva y su posterior implementación mediante los casos de uso.

La descripción de la propuesta mediante la tabla debe contener:

- << Propuesta >>: Identificador para la descripción.
- << Proceso de negocio >>: Descripción del proceso de negocio al cual se hace referencia.
- << Actividades >>: Determinación de las actividades realizadas de manera secuencial.

- << Actores >>: Identificación de los actores.
- <<Herramienta>>: Herramientas empleadas para la ejecución de las actividades.

Tabla 19. Plantilla de propuesta de valor.

Propuesta:	<< >>
Proceso de negocio:	<< >>
Actividades:	<< >>
Actores:	<< >>
Herramienta	<< >>

Fuente: elaboración propia.

En base a la plantilla especificada se debe realizar un caso de uso especificando los actores y actividades.

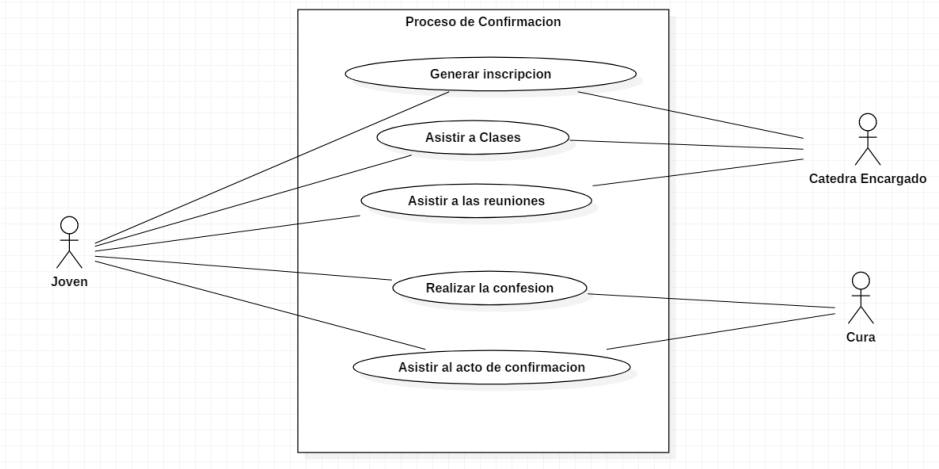

Ilustración 6. Plantilla de caso de uso. **Fuente:** elaboración propia.

3.3.1.3. Definición de procesos de negocios

En palabras de Salgado (2011):

El modelado de procesos de negocio es la base para comprender mejor la operación de una organización, documentar y publicar los procesos buscando una estandarización en la organización, buscar eficiencias en la operación e integrar soluciones en arquitecturas orientadas a servicios.

3.3.1.3.1. Descripción textual y gráfica del proceso de negocios

Para lograr una mejor comprensión se necesita identificar los agentes que intervienen en los procesos, como los departamentos, personas, herramientas, entre otros, de manera que se encuentren conectados directamente con la organización. Al determinar los actores, se especifican cada una de las actividades o acciones que dichos usuarios realizan. Como resultado se obtiene un listado de las actividades que realizan cada uno de los actores que intervienen en el proceso de negocios.

3.3.1.3.2. Especificación de actividades de negocio

La especificación de las actividades del negocio se hace en base al proceso de negocio especificado anteriormente, y usando el diagrama de actividades para plasmar de manera visiblemente entendible cada una de las actividades (Ariste, Ponisio, Nahuel, y Giandini, 2015).

Dentro de una organización se presenta una gran variedad de actividades las cuales deben ser representadas mediante una serie de flujos de trabajo de actividades de cada proceso. Dichos diagramas permiten identificar de manera sencillas, las actividades y los roles relacionados a cada proceso.

Para tener fundamentos en el desarrollo de los diagramas de actividades se debe realizar una plantilla para definir, las restricciones, las actividades y los actores que intervienen en cada proceso. El diagrama de actividades se realiza uno por cada proceso identificado.

Tabla 20. Plantilla de actividades de negocio.

Proceso de negocios:	Se identifica el proceso de negocio al cual se hace referencia.
Restricciones:	Se establecen las restricciones posibles al momento de realizar las actividades del proceso.
Actividades:	Se define cada una de las actividades incurridas dentro del proceso de negocio. Estas actividades deben estar enlistadas de manera secuencial.
Postcondiciones:	Se establecen las actividades alternativas en caso de una decisión en alguna actividad.
Actores:	Identificar que actores son los que intervienen dentro de estas actividades.

Fuente: elaboración propia.

Por consiguiente, se debe realizar el diagrama de actividades en base a la plantilla establecida anteriormente.

3.3.1.4. Reglas de negocio

El uso de modelos de negocios permite identificar los objetivos, actores, recursos y finalmente reglas del negocio, de esta manera el gerente del proyecto logra entender la razón del planteamiento de requisitos y no simplemente la funcionalidad o que actividades debe realizar el sistema (Leonardi y Mauco, 2004).

Una regla de negocio es una condición que debe ser efectuada cuando se realiza alguna actividad dentro de la organización, puede ser impuesta por las políticas de la empresa, o la toma de decisiones del gerente.

Al definir las reglas de negocios se deben considerar algunos aspectos importantes, los cuales se generan desde el punto de vista de los usuarios (Amaolo, 2007).

- Las reglas de negocio deben definir de manera independiente los procesos de negocios.
- Deben ser expresadas en un lenguaje natural comprensible, sin buscar formalismos.
- Deben ser comprendidos por las personas de negocios, sin conocimiento de actividades específicas como programación o análisis de diseño.

Para ello se deben especificar las condiciones y las acciones a realizar en caso de que suceda.

Tabla 21. Plantilla para definición de reglas de negocios.

Regla	Condiciones	Acciones
Se describe la regla de negocio planteada.	Se describe la situación bajo la cual se puede ocasionar la regla.	Se describen los sucesos resultantes de la evaluación de la condición.

Fuente: elaboración propia.

3.3.2. Fase de evaluación

3.3.2.1. Importancia para el gerente

La fase de evaluación dentro del modelado de negocios ayuda al gerente del proyecto a conocer las razones y circunstancias en las que se determinan los requerimientos. Al definir las reglas de negocio el gerente conocer mejor el funcionamiento de la organización y ayuda a los miembros del equipo de desarrollo a conocer las funcionalidades básicas que debe efectuar la aplicación.

3.3.2.2. Análisis FODA

"El análisis FODA consiste en realizar una evaluación de los factores fuertes y débiles que en su conjunto diagnostican la situación interna de una organización, así como su evaluación externa; es decir, las oportunidades y amenazas" (Ponce, 2005). Esta herramienta ayuda al gerente a realizar un análisis organizacional en relación con factores determinantes de la organización para lograr el éxito y el cumplimiento de los metas del proyecto.

A través de este análisis se logra diagnosticar y tomar decisiones en torno a las fortalezas y debilidades encontradas en la organización. Al realizar el análisis FODA es necesario considerar los siguientes factores:

- Debilidades.
- Amenazas.
- Fortalezas.
- Debilidades.

Cada uno de los factores deben ser considerados en base a la función de los procesos de negocios actuales de la organización (Salazar Morales, y Rivero Ceballos, 2013).

Tabla 22. Análisis FODA.

Fortalezas	**Oportunidades**
Factores internos que posee la organización, y pueden ser aprovechados para alcanzar el éxito de los objetivos.	Factores externos a la organización que permiten a la organización alcanzar las metas de manera directa o indirecta.
Debilidades	**Amenazas**
Factores (personas u objetos) que influyen de manera negativa en la organización, y generan un resultado perjudicial al proyecto. Su reconocimiento permite minimizarlas mediante la planificación y gestión de riesgos.	Factores o situaciones negativas externas a la organización, las cuales afectan negativamente a la misma.

Fuente: elaboración propia.

3.3.3. Fase de innovación (Opcional)

La fase de innovación del proceso es opcional, y consiste en la identificación de modificaciones dentro del modelo de negocios actual, las cuales brindan a la organización a generar mejores resultados y beneficios tangibles e intangibles. La propuesta puede ser planteada para indicar al desarrollador las alternativas de

diseño que puede implementar y que debe contemplar (no obligatoriamente) dentro del diseño y codificación de la aplicación.

3.4. Identificación de interesados

La gestión de los interesados es una etapa fundamental dentro del desarrollo de una aplicación web, sobre todo, ya que es importante que se conozca los usuarios finales a quienes va dirigida la aplicación, quienes intervienen durante la fase de análisis, planificación, modelado, y las que siguen.

Al momento de identificar los interesados el gerente debe incluir todas las personas quienes intervienen directa o indirectamente en el proceso de desarrollo del proyecto, además de identificar los grupos que pueden ser afectados. Un indicador clave en la gestión de los stakeholders (interesados) es plantear y clasificar los roles que desempeñan cada uno de ellos, de acuerdo con las actividades que realizan en el proyecto.

3.4.1. ¿Por qué es importante esta fase?

"Es importante que las organizaciones realicen un mapeo de los stakeholders y analicen la relación que sostienen con ellos, con el propósito de fortalecer y consolidar estos vínculos" (Acuña, 2012).

El objetivo que propone resolver esta fase es la determinación del nivel de implicación de cada uno de los interesados, para conseguir un proyecto exitoso. Para ello, es necesario una categorización de los interesados, en las cuales se debe tener presente los siguientes niveles:

- Interesado no consciente: cuando no conoce nada del proyecto, ni el impacto que este puede efectuar sobre la organización o el grupo de trabajo.
- Interesado resistente: cuando no es consciente del proyecto, y se resiste al cambio de perspectiva o idea.
- Interesados neutrales: son los interesados que son conscientes de la planificación e impacto del proyecto, pero no demuestran apoyo hacia el mismo., ni directa ni indirectamente.
- Interesado a favor: son los interesados conscientes y que a más de eso apoyan las actividades realizadas dentro del proyecto.

- Interesado que lidera: interesado que participa activamente en el desarrollo del proyecto, para lograr asegurar que el proyecto sea un éxito.

3.4.2. Identificación de roles y stakeholders

La identificación de los stakeholders se realiza con la finalidad de establecer los roles que cada participante en el proyecto realiza, y establecer las actividades que deben efectuar dentro de la etapa de la planificación. Dentro del documento se debe incluir los roles y elemento comúnmente necesarios para la planificación de estrategias como:

- El nivel de implicación que cada usuario posee dentro del proyecto.
- La relación existente entre cada uno de los interesados.
- Definir los medios y herramientas de comunicación para cada uno.
- Los documentos y la información que se debe dar acceso a cada parte interesada.

Algunas buenas prácticas al momento de identificar los stakeholders consisten en:

1. Identificar todas las personas que intervengan en el proyecto.
2. Conocer y comprender las necesidades y expectativas que cada uno posee.
3. Analizar cada uno de los aspectos planteados anteriormente.
4. Integrar estrategias de organización para las necesidades de cada uno de los interesados.
5. Verificar las relaciones existentes entre cada uno de ellos.
6. Identificar su importancia dentro del proyecto.

Existe una gran variedad de modelos para lograr identificar de manera adecuada a los identificados, entre ellos se plantea la necesidad de dependencias, responsabilidades, el dinamismo en las actividades que cada uno posee, y los roles respectivos que ejecutan en cada etapa.

SWIRL emplea su propia plantilla para lograr la identificación de los stakeholders y los roles-funciones que cada uno desempeña. Es fundamental recalcar, que la matriz se realiza evaluando en cada una de las etapas, y otra matriz general. La matriz general permite al gerente conocer cuáles son todos los interesados con los que cuenta en

el proyecto, y que responsabilidad tienen de manera general en el proyecto. Por otro lado, la matriz de stakeholders por etapas, permite tanto al gerente como a todos los interesados conocer la función o rol que desempeña una persona en específico dentro de cada etapa. Es decir, se debe realizar una matriz de los interesados que intervienen en la etapa de análisis, otra matriz para identificar a los que intervienen en la etapa de planificación y así durante todas las etapas del proceso de desarrollo.

Tabla 23. Plantilla de matriz general de stakeholders.

MATRIZ GENERAL DE STAKEHOLDERS			
Nombre	***Rol***	***Profesión***	***Responsabilidades***
Nombre del interesado, el cual realice fácilmente su identificación	Indica el cargo o función que desempeña el interesado durante todo el proyecto.	El titulo o profesión que desempeña.	Se describe de manera general las responsabilidades que posee durante todo el proyecto.
----------	En caso de existir más de un rol por persona se debe agregar en otro indicador	----------	----------

Fuente: elaboración propia.

Las componentes que intervienen dentro de la matriz específica de interesados son el rol designado, la responsabilidad, la función que ejerce, y el impacto que tienen en la etapa específica del proyecto.

Cada matriz específica debe ser comunicada a todos los equipos de trabajo del proyecto, sin incluir al cliente, de manera que logre conocer las actividades que realiza cada grupo de trabajo dentro del proyecto. La matriz general de stakeholders se convierte en entregable al finalizar esta fase, por lo que debe ser entregada al cliente, logrando de esta manera que el cliente conozca las responsabilidades que poseen y los miembros del equipo de trabajo. Hay que recordar, que la metodología SWIRL se basa en la comunicación constante con el cliente, y a inclusión dentro de todas las fases del proceso de desarrollo del proyecto, logrando obtener un producto exitoso.

Tabla 24. Plantilla de matriz específica de stakeholders.

MATRIZ ESPECÍFICA DE STAKEHOLDERS.					
Etapa:	**Fase a la cual se hace referencia.**		**Fecha:**	**Fecha actual.**	
Nombre	***Rol***	***Profesión***	***Actividades***	***Responsabilidades***	***Impacto***
<<>>	Cargo o función dentro del proyecto.	Titulo o profesión que ejerce.	Todas las actividades para desarrollar en la etapa.	Responsabilidad durante la fase actual.	Porcentaje de inclusión dentro de la fase.

Fuente: elaboración propia.

Los roles de mayor significancia que deben ser incluidos por obligación dentro de la matriz son:

- Gerente del proyecto.
- Analista de requerimientos.
- Desarrollador.
- Diseñador.
- Tester.
- Cliente.
- Usuario.

3.4.3. Definir las actividades

Definir las actividades que cada uno de los interesados debe realizar se realiza en base a grupos de interesados definidos por el gerente del proyecto, esta identificación difiere de la determinación de actividades en la matriz específica de stakeholders, debido a que se deben asignar grupos de trabajos y definir las actividades durante el proceso completo del proyecto, sin separarlas por etapas.

Para ello, se debe emplear la siguiente matriz:

Tabla 25. Plantilla de matriz específica de stakeholders.

ACTIVIDADES		
Nombre	***ROL***	***Actividades***
Nombre del interesado, el cual realice fácilmente su identificación	Descripción del rol o función que cumple.	Todas las actividades que desarrollar durante todo el proyecto.

Fuente: elaboración propia.

Nota:

Al definir las actividades, hay que tener en cuenta que es diferente a la identificación de actividades dentro del modelado de negocio, ya que en el modelado se definen las actividades de los usuarios de la organización; mientras que, en la definición de actividades de los interesados, se debe estimar las actividades que cada uno de los stakeholders del proyecto debe realizar, desde el cliente hasta el equipo de desarrollo.

3.5. Requerimientos

"Los requerimientos son necesidades que deben satisfacer los sistemas a ser construidos, y que la satisfacción de determinados conjuntos de requerimientos define el éxito o fracaso de los proyectos" (Báez y Barba, 2000, p. 211). Para lograr capturar adecuadamente los requisitos en el desarrollo de software, SWIRL considera las fases implicadas en la ingeniería de requerimientos, las cuales abarcan:

- Elicitar los requerimientos que solicita el cliente.
- Asegurar la consistencia y factibilidad de las necesidades.
- Validar los requerimientos en base a las necesidades solicitadas.

3.5.1. Elicitación

La elicitación de requerimientos consiste en la recolección de los requisitos en base a las necesidades y problemas que el cliente menciona durante la reunión con el analista (Arias, 2005)6,8]]}}}],"schema":"https://github.com/citation-style-language/schema/raw/master/csl-citation.json"} , durante este proceso el analista debe considerar algunos puntos claves, para ello puede realizar las siguientes preguntas:

- ¿Cuál es la problemática que intenta resolver?
- ¿Qué necesidades presenta su organización?
- ¿Qué informes o reportes desea que el sistema realice?
- ¿Cuáles son los módulos y funcionalidades que el sistema debe contener?
- ¿Qué restricciones debe poseer la aplicación?
- ¿Quiénes serán los usuarios finales de la aplicación?

3.5.1.1. ¿Cómo elicitar los requerimientos para la aplicación?

Para elicitar los requerimientos se requieren varias herramientas de manera que la determinación de los mismos sea preciso y logre resolver las necesidades del cliente. Se pueden emplear métodos interactivos como son las entrevistas con el cliente, encuestas a los usuarios finales, la selección de muestras, investigación en datos físicos o impresos. Por otro lado, se pueden emplear técnicas en las que no interfiere el usuario como es la observación de campo de los encargados de la toma de decisiones, elaboración de prototipos, entre otros (Kendall y Kendall, 2005).

SWIRL hace uso de técnicas de recolección de requerimientos las cuales incluyen la entrevista y brainstorming-prototipos.

- Entrevista: esta técnica se basa en una reunión cara a cara con el cliente, el gerente del proyecto y el analista. El encargado de la recolección es el gerente mientras que el analista es el encargado de la identificación de requisitos poco ambiguos, concretos y que definan el comportamiento que el sistema busca.

La entrevista puede ser requerida en segunda instancia, ya que puede que no se hayan obtenido todos los requerimientos o simplemente se necesite aclarar el comportamiento final.

- Brainstorming-prototipos: esta técnica se basa en una reunión con el cliente, pero en lugar de establecer todos los requerimientos que él desea, se muestra prototipos creados de proyectos pasados, con lo cual el cliente tiene una idea más clara si la aplicación resolverá o no su problema.

Es importante recalcar, que dentro de la metodología SWIRL la reunión con el cliente para la elicitación de requerimientos es constante, se realiza mediante reuniones al finalizar cada iteración del proyecto. Sin embargo, en las iteraciones no se establece una reunión larga con requerimientos aun no precisos, sino, se identifican algunas modificaciones en los mismos, considerando cuales requerimientos se descartan o se modifican.

¿Qué sucede si la elicitación no es la correcta?

En caso de que la recolección de requerimientos no sea la más adecuado y se hayan indicado requisitos ambiguos, poco factibles, entre otros, pueden llegar a afectar el desempeño de los grupos de trabajo durante el desarrollo del sistema.

3.5.1.2. Elementos de la elicitación de requerimientos

Para lograr definir de manera clara la funcionalidad y el comportamiento del sistema es importante que el analista identifique cuatro tipos de requerimiento, los requerimientos funcionales y no funcionales, que son los más comunes y que define con el claridad el cliente; los requerimientos de usuario, los cuales se definen mediante la experiencia y necesidad dentro de la organización; y finalmente los requisitos navegacionales, estos requerimientos se establecen en función de la manera en que las páginas de la aplicación web se comunican e interactúan entre sí.

3.5.1.2.1. Requisitos funcionales

Los requerimientos funcionales son todos las actividades y el comportamiento que el software debe tener ante determinadas circunstancias. Además de eso, deben ser capaces de describir la interacción que posee el sistema con el entorno en donde trabajará, y con los usuarios finales que accederán.

Para una identificación de requerimientos más clara se deben haber identificado los actores, y posteriormente especificar, que acciones realizarán dentro del sistema, y de qué manera interactuarán con la aplicación. La identificación de requerimientos se realiza mediante dos plantillas, la presentación general de los requerimientos la presentación de requerimientos específicos.

Tabla 26. Plantilla general de requerimientos funcionales.

REQUERIMIENTOS FUNCIONALES	
Código	***Descripción***
Empleado para identificar con facilidad, se codifica empleando << RF-01>>	Se describe el requisito funcional del sistema.

Fuente: elaboración propia.

Para cada uno de los requerimientos planteados en la plantilla especifica se necesita realizar un desglose de requerimientos.

Tabla 27. Plantilla especifica de requerimientos funcionales.

ESPECIFICACIÓN DE REQUERIMIENTO FUNCIONAL			
Fecha de elaboración:	<<fecha actual>>	**Versión:**	<< >>
RF-01	descripción del requerimiento		

ESPECIFICACIÓN DE REQUERIMIENTO FUNCIONAL			
Categoría:	<<completo, incompleto, realizable, necesario, o priorizable>> se pueden emplear más de una categoría para identificar el requisito.		
Fecha:	fecha de modificación	**Responsable**	encargado de su realización
Observación:	<< >>		
Actores:	actores que incurren en el desarrollo del mismo		
Post-condición	acción bajo circunstancias. Especificar la condición.		

Fuente: elaboración propia.

Por ejemplo: *Ejemplos de requerimientos funcionales, son:*
- La aplicación tiene tres tipos de usuarios los cuales deben ser validados.
- Debe tener un módulo de contabilidad.
- La aplicación debe generar reportes diarios sobre los ingresos generados.

3.5.1.2.2.Requisitos no funcionales

Estos requisitos hacen referencia a las características propias que posee la aplicación, como el rendimiento, o seguridad, es decir, no se especifica el comportamiento ni que funciones cumple, sino la manera en que efectúa dichas funciones.

Tabla 28. Plantilla de requerimientos no funcionales.

REQUERIMIENTOS NO FUNCIONALES	
Código	***Descripción***
Empleado para identificar con facilidad, se codifica empleando << RNF-01>>	Se describe el requisito funcional del sistema.

Fuente: elaboración propia.

Cada requerimiento no funcional debe ser definido de manera específica mediante la plantilla indicada a continuación.

Tabla 29. Plantilla especifica de requerimientos no funcionales.

ESPECIFICACIÓN DE REQUERIMIENTO FUNCIONAL	
RNF-01	Descripción del requerimiento
Prioridad:	<<alta, media, baja>> prioridad de ser implementado.
Descripción:	Descripción detallada del requerimiento.
Responsable:	Encargado del cumplimiento del requisito.

ESPECIFICACIÓN DE REQUERIMIENTO FUNCIONAL	
Comentario:	Información adicional para su implementación.

Fuente: elaboración propia.

Por ejemplo: *Ejemplos de requerimientos no funcionales, son:*
- Seguridad.
- Rendimiento.
- Fiabilidad.

3.5.2.1.3. Requisitos de usuario

Los requerimientos de usuario se definen en base a las acciones que el usuario final debe ser capaz de realizar, a diferencia de los requerimientos funcionales, los requerimientos de usuarios solo centran su descripción en la forma en que el usuario ejecuta una acción determinada, es decir, ¿qué hace el usuario? Y no ¿qué hace el sistema? Para ello se definen métodos de representación como casos de usos, descriptores de escenario, y en ocasiones tablas de evento-respuesta.

El escenario que se debe identificar es la descripción textual o en casos narrativa, de las actividades que los actores deben efectuar, indicando el uso y acceso al sistema. Esta debe ser expresada desde el punto de vista del actor, y detallar cada una de las actividades necesarias para realizar las funciones establecidas. La identificación gráfica de los mismos se realiza en la fase de análisis mediante el uso de casos de uso.

Para definir los requisitos de usuarios, SWIRL emplea las tablas de evento-respuesta, en las cuales se especifican las acciones que realiza el usuario, y la respuesta que el sistema emite para ello.

Por ejemplo: *Ejemplos de requerimientos de usuario, son:*
- Hacer una transacción en línea.
- Visualizar los valores de la cuenta bancaria.
- Registrarse como usuario nuevo.

¿Cómo reconozco un requerimiento de usuario?
Para reconocer si un requerimiento es de usuario o funcional, el analista debe realizar las siguientes preguntas ¿Es únicamente el usuario final que realiza esta acción?, ¿Puedo establecer el procedimiento para su realización?, ¿No es una característica que debe tener el sistema?, Un ejemplo de requerimiento de usuario en el caso de sistema bancario, sería que el usuario pueda "hacer una transacción en línea.".

Tabla 30. Plantilla de tabla evento-respuesta.

Evento	Actividad	Respuesta
La acción que el usuario es capaz de realizar	El detonador, o las actividades que realiza el usuario.	El retorno de una página, o la respuesta del sistema ante esa petición.

Fuente: elaboración propia.

3.5.1.2.4. Requisitos navegacionales

Recogen las necesidades que el usuario presenta dentro de la navegación del sistema, dentro de estos se especifican las condiciones y restricciones para navegar entre páginas, y si es necesario los elementos que los usuarios necesitan para poder acceder a la misma. Este tipo de requerimientos son fundamentales en el desarrollo de aplicaciones web, debido a que tanto el equipo de desarrollo como el usuario, deben tener conocimiento de cómo navegar a través de las páginas de la aplicación.

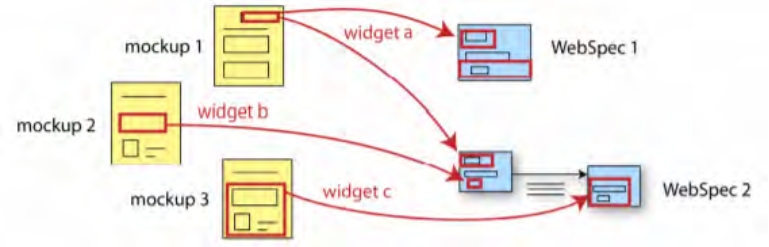

Ilustración 7. Plantilla de caso de uso. **Fuente:** (Grigera, Rivero, Robles, Giacosa, y Rossi, 2012)& Rossi, 2012.

Por ejemplo: *Ejemplos de requerimientos navegacionales, son:*

- Las páginas principales deben ser capaces de re direccionar a páginas oficiales de la empresa.

- La descarga y visualización se debe realizar en una nueva página, mediante el clic en un botón.

3.5.2. Análisis

3.5.2.1. ¿Por qué analizar los requerimientos del proyecto?

El proceso de análisis de requerimientos es la segunda etapa dentro de la ingeniería de requerimientos, y se basa en la evaluación y refinamiento de los requerimientos establecidos con anterioridad. Al finalizar esta etapa se logra identificar los requisitos que sean obsoletos o poco factibles para realizar.

Es importante que el analista defina, modele, refina, y priorice los requerimientos que posee el proyecto, de manera que se consigue un proceso de desarrollo mucho más claro y comprensible para el equipo de desarrollo durante la etapa de implementación.

3.5.2.2. Elementos del análisis de requerimientos

Dentro de la metodología SWIRL el análisis de requerimientos se fundamenta en tres procesos:

- Refinamiento de requerimientos.
- Priorización de requerimientos.
- Identificación de dependencias (opcional).
- Diseño de casos de usos.

3.5.2.2.1. Refinamiento de requerimientos

En esta etapa se identifican los requerimientos que se encuentren incompletos, o no sean factibles para el proyecto. Esta acción es realizada en una reunión entre el gerente, analista y desarrolladores del proyecto, con la finalidad de preparar al equipo para el desarrollo de cada una de las tareas. Es fundamental que se identifiquen que los requisitos sean claros, no ambiguos, y completos.

3.5.2.2.2. Priorización de requerimientos

Los requisitos funcionales recolectados anteriormente son analizados respectivamente con la finalidad de facilitar la implementación de cada uno de los mismos, considerando el tiempo que se posee para la realización de cada uno.

Las priorizaciones de los mismos se realizan mediante tres indicadores los cuales son Alto (se implementará primero), medio (se implementarán después de finalizar los requerimientos con alta prioridad), baja (en caso de no poseer suficiente tiempo o presentarse algún retraso pueden obviarse su implementación). Adicionalmente del rango de prioridad se incluye un numero de implementación el cual representa la secuencia en la cual los desarrolladores deberán codificarlos.

Para ello, la metodología SWIRL define la siguiente plantilla, la cual ayuda al analista a identificar, los requisitos óptimos, y eliminar los que se consideren ambiguos o pocos factibles.

Tabla 31. Plantilla de priorización de requerimientos.

Código	Descripción	Indicador de priorización	# implem.
Identifica el código del requerimiento RF o RNF	Breve descripción del requerimiento	La prioridad que se debe dar al requerimiento. <<Alto, Medio, Bajo>>	Orden para implementar. <<1, 2, 3, …>>

Fuente: elaboración propia.

3.5.5.2.3. Identificación de dependencias

Este proceso es opcional, y consiste en la especificación de requerimiento que dependan de actores, herramientas u otros requerimientos. Para ello se realiza una descripción textual de cada una de ellas, especificando la actividad en la cual ocurre la dependencia, y posterior a ello, la dependencia como tal.

Por ejemplo: *Ejemplo de identificación de dependencias:*

- Para acceder a la página de descargas de informes, se necesita que el usuario cajero haya creado un balance mensual de las transacciones. <<el requerimiento informe depende del actor cajero y el requerimiento balance mensual.>>

3.5.2.2.4. Diseño de casos de usos

El diseño de diagramas de caso de uso permite graficar cada uno de los requerimientos funcionales y los requerimientos de usuario, su diseño se basa en las actividades que realiza el usuario y se debe ejecutar en base a la visión de los actores. Es necesario que antes de diagramar los casos de uso se realice la especificación de los usuarios y la plantilla de casos de usos.

La plantilla de descripción de actores permite identificar todos los actores que intervienen con el sistema y su nivel y actividades dentro del mismo.

Tabla 32. Plantilla para descripción de actores.

Actor	Nombre	**Identificador:**	AC-01
Descripción:	Descripción del actor		
Roles:	El rol que cumple dentro del sistema, un mismo usuario puede llegar a desempeñar varios roles.		
Relación:	Describe la relación con otros actores, se debe especificar cada una de ellas.		
Referencias:	Encargado del cumplimiento del requisito.		

Fuente: elaboración propia.

Una vez realizadas las fichas de actores, se procede a realizar las fichas para caso de uso. SWIRL al ser una metodología versátil emplea la siguiente plantilla, que considera las acciones que se realizan en un escenario, y en caso de existir alguna postcondición en caso de alguna determinada circunstancia se debe indicar.

En la ficha de descripción de un caso de uso se identifican los siguientes elementos:

- CU-XX: El nombre del CU debe comenzar por un verbo y ser lo más corto posible, pero que, a su vez, describa lo que el CU hace.
- Actores: Los actores que interactúan directamente con el sistema, tanto los primarios quienes inician el CU, como los secundarios que interactúan con el sistema luego que este ha iniciado.
- Propósito: intención del caso de uso.
- Tipo.
- Descripción: Sin dar detalles del cómo, la descripción del caso de uso resume todo lo que el caso de uso hace, así como la necesidad de la existencia de actores secundarios, para los casos que aplique.
- Precondiciones: Las precondiciones son elementos opcionales de los CU.
- Curso normal y curso alternativo de eventos.

Tabla 33. Plantilla para descripción de caso de uso.

CU-XXX	Nombre del caso de uso
Actores	Todos los actores que intervienen.
Propósito	Objetivo del caso de uso.
Tipo	<< Primario, Secundario, Opcional >>
Descripción	Describe el caso de uso.
Pre-condiciones	¿En qué estado debe encontrarse el sistema para que el CU se pueda iniciar?
Curso Normal de los eventos Describe los detalles de la conversión interactiva entre los actores y el sistema. Un aspecto esencial de la sección es explicar la secuencia más común de eventos: la historia normal de las actividades y el término exitoso de un proceso. No incluye situaciones alternativas.	
Acción de los actores	**Respuesta del sistema**
Cursos Alternativos	Describe importantes opciones o excepciones que pueden presentarse en relación con el curso normal. Si son éstas son complejas, se pueden expandir y convertir en nuevos casos de uso.

Fuente: elaboración propia.

Una vez que tanto los actores como los casos de uso se encuentren detallados se procede a diagramar el caso de uso mediante el uso de herramientas de modelado UML.

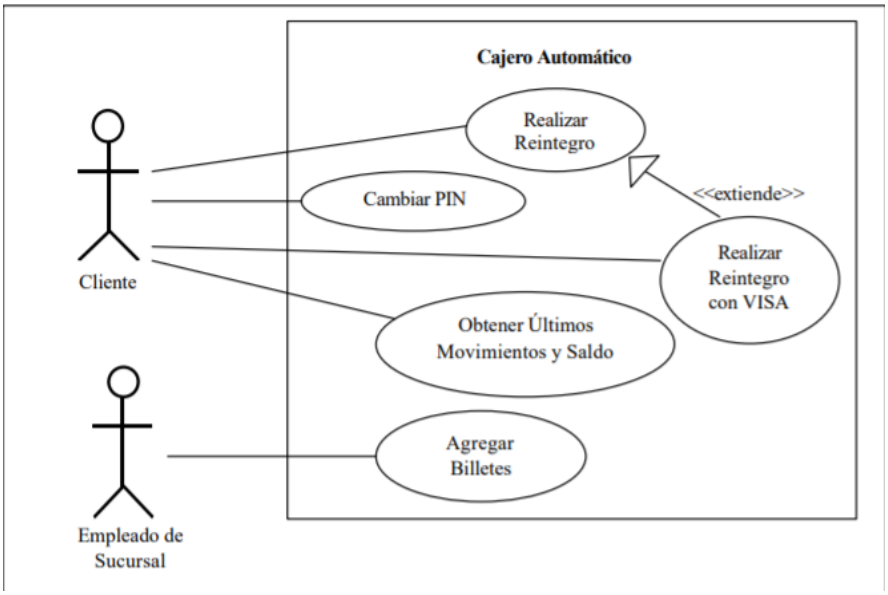

Ilustración 8. Ejemplo de diagrama de caso de uso. **Fuente:** (Ferré y Sánchez, s. f.)

3.5.3. Validación

La validación de los requerimientos debe ser realizada antes del inicio de codificación e implementación del sistema, dentro de esta etapa se busca asegurar que los requisitos cumplan los objetivos planteados en la fase de análisis, y que el sean consistentes, para lograr satisfacer las necesidades del usuario final.

Para la validación se emplean técnicas que permiten identificar si existe correspondencia entre los objetivos, y los requerimientos. Una de estas técnicas es la evaluación mediante una matriz de trazabilidad, comparando los requerimientos, con los actores, o en ocasiones los casos de usos con los actores. Otra técnica que se puede aplicar es el uso de prototipo de interfaz de usuario, de manera que se logre entender de manera comprensible la aplicación propuesta en base a los requerimientos, esta técnica requiere mayor esfuerzo debido a que es necesario poseer un prototipo para su ejecución.

SWIRL emplea la técnica de validación mediante matrices de trazabilidad, entre caso de uso y requerimientos; y entre requerimiento y actores. Debe ser realizado por el

analista de requerimientos, y verificado con el gerente del proyecto y juntamente con el cliente.

3.5.3.1. Matriz de trazabilidad

La trazabilidad en la Ingeniería de Software es una práctica de control que ayuda a obtener el producto en el dominio de la solución lo más exacto y fiable posible a las necesidades expresadas por el cliente. En el desarrollo de aplicaciones web, los requerimientos poseen características particulares, las cuales no están contempladas explícitamente en los estándares vigentes (Ferraro, Medina, Dapozo, y Estayo, 2016, p. 1).

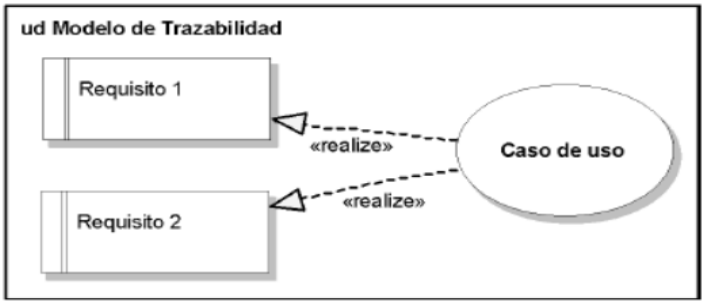

Ilustración 9. Modelo de trazabilidad. **Fuente:** (Ferré y Sánchez, s. f.)

Tabla 34. Matriz de trazabilidad requerimiento-actores.

# RF	Actores Requerimientos	Actor 1	Actor 2	Actor 3
	Requerimiento 1…		x	
	Requerimiento 2…	x		x

Fuente: elaboración propia.

Donde:

- #RF: Indica el código de Requerimiento Funcional al cual se hace referencia.
- Requerimiento: en esta columna se deben listas todos los requerimientos según el código que representen.
- Actores: De manera horizontal se detalla cada uno de los actores.

CAPÍTULO IV: FASE DE PLANIFICACIÓN

La etapa de planificación es la fase que posee un impacto mayor dentro del desarrollo del proyecto, conteniendo los análisis más relevantes sobre las actividades y la gestión de recursos con la finalidad de la reducción del costo y tiempo durante el desarrollo (Arévalo y Atehortúa, 2012). Dentro de la etapa de planificación, la dirección del gerente y la toma de decisiones sobre las actividades y procesos que se llevarán a cabo, son de suma importancia.

4.1. Objetivos

Los objetivos que provee esta etapa están relacionados principalmente con el gerente del proyecto y su toma de decisiones. Sin embargo, la fase tiene como objetivo principal la facilitación y desarrollo de calidad del proyecto de software, logrando entregar un producto que cumpla las necesidades del cliente. A más de lo mencionado, algunos objetivos que pretende cumplir esta fase son:

- Identificar los recursos, tanto de talento humano como tecnológicos, con los que cuenta la organización para el desarrollo del proyecto.
- Establecer un plan de actividades en base a tiempos y recursos definidos, gestionando de manera adecuada el cronograma del proyecto.
- Identificar y prevenir los posibles riesgos mediante actividades realizables por el equipo de trabajo.
- Definir la interacción y comunicación de todos los miembros e interesados dentro del proyecto.

4.2. Actividades en la fase de planificación

La etapa de planificación es una fase breve, más corta que la etapa de análisis, debido a que en esta simplemente se organizan los recursos, tiempo, y actividades que incurren tanto positiva como negativamente dentro del proyecto. Las actividades principales dentro de esta etapa es la gestión del tiempo y riesgo, dando como resultado entregable al cliente, y al equipo de desarrollo.

Las actividades por las que está compuesta esta fase son:

- Historias de usuarios.

- Definición de entregables.
- Gestión de cronograma.
- Gestión de riesgos.
- Gestión de comunicaciones.
- Gestión de cambios.
- Gestión de calidad.

Dentro de cada una de las subetapas se generarán entregables, la finalidad de ello es que el equipo de trabajo conozca mediante planes cada una de las actividades y decisiones que el gerente ha tomado sobre el proyecto. Algunos de los entregables que se presentan dentro de esta etapa son:

- Cronograma.
- Plan de prevención de riesgos.
- Plan de contingencias de riesgos.
- Plan de comunicaciones.
- Plan de adquisiciones.
- Estructura de desglose de trabajo.

4.3. ¿Quiénes intervienen?

Las actividades inmersas dentro de esta etapa son realizadas completamente por el gerente del proyecto, él es el encargado de gestionar correctamente los recursos materiales, humanos y tecnológicos que posee la empresa para cumplir las expectativas y requerimientos que el cliente posee. La finalidad de esta etapa es que el gerente sea capaz de gestionar planificaciones adecuadas, que ayuden a las partes interesadas a conocer cómo se está llevando a cabo el proyecto.

Sin embargo, aunque la mayoría de las actividades son realizadas por el gerente, el cliente debe intervenir al finalizar la etapa. SWIRL, busca el desarrollo de proyectos web de calidad, enfocándose en una de las características más importantes, la cual es la inclusión de requerimientos dinámicos por parte del cliente en cualquier etapa. Considerando eso, el cliente debe encontrarse inmerso en cada una de las fases, por lo que al concluir la etapa de planificación el gerente debe entregar las planificaciones

al cliente con la intención de concordar las gestiones desarrolladas, principalmente el cronograma, y el plan de comunicaciones.

¿El equipo de trabajo no interviene en esta etapa?

Es deber del gerente mantener informados a todos los miembros del equipo de trabajo sobre las decisiones que realiza con respecto al proyecto. Sin embargo, el equipo de trabajo o los demás miembros del proyecto no intervienen activa ni directamente en la toma de decisiones y gestión de los recursos.

4.4. Historias de usuarios

Al igual que otras metodologías ágiles, SCRUM emplea las historias de usuario para describir las funcionalidades que va a tener la aplicación, y la manera en que los usuarios interactuarán con las mismas. Para definir claramente las historias de usuario, se deben especificar los requerimientos priorizados y refinados por el analista en la etapa anterior, es decir, en el análisis.

La historia de usuario no es más que una tarjeta de indicaciones y especificaciones de las actividades a realizarse para dicha funcionalidad, sin embargo, consta de tres etapas o actividades que el gerente debe realizar para lograr una buena definición de las mismas.

1. Definición, mediante el uso de plantillas las cuales sean entendibles por el usuario, gerente y equipo de trabajo.
2. Revisión, se basa en la conversación entre el gerente y el cliente sobre las historias definidas.
3. Aceptación, es la etapa final, y busca la validación y confirmación de cada una de las historias definidas.

¿Qué pautas seguir para realizar una correcta historia de usuario?

Como es común en las metodologías ágiles, y sobre todo usada en la Scrum, las historias de usuario poseen diferentes modalidades para representarlas. Sin embargo, la descripción elementos y características que posee son las mismas. Una correcta definición de historias de usuario beneficia en gran alcance al proyecto, es por ello, que se deben considerar algunas indicaciones como las que se presentan en las siguientes referencias:

1) («Historia de usuario - Scrum Manager BoK», 2014)

2) (Alex y Menzinsky, 2017)

Para la descomposición de las historias de usuarios se emplean técnicas diversas, SWIRL hace uso de la metodología SPIDR (Cohn, s. f.) para crear mejores historias de usuarios y separarlas en función de 5 campos, spikes (o descomposición de las

actividades complejas); paths (la manera en que se realiza); interfaces (identificar las interfaces en las cuales se realiza la historia); Data (identifica los tipos de datos que pueden ser soportados); rules (resuelve de la mejor manera las reglas del negocio).

En otras palabras, dentro de SWIRL para identificar los usuarios se deben seguir las siguientes pautas, en base a la plantilla proporcionada.

1. Refinar las historias de usuario mediante el entendimiento de la funcionalidad principal del sistema ante la situación en específico.
2. Definir diversos caminos para ejecutar las historias grandes, se realiza mediante diagramas de flujos para tener una mejor perspectiva de las soluciones.
3. Las historias deben ser divididas en función de diferentes interfaces que cubran la funcionalidad especificada.
4. En caso de ser necesario se debe dividir las historias por los tipos de datos que brindan en la aplicación.
5. Comúnmente, las historias se vuelven complejas por el uso de reglas de negocio incomprensibles o estrictas, al dividir cada una de las historias según la regla que efectúan el equipo de desarrollo logra comprender en más detalle el comportamiento y funcionalidad del sistema.

La plantilla de definición de historias de usuario es la siguiente:

Tabla 35. Plantilla de historia de usuario.

Historia de Usuario	
Número:	Usuario:
Nombre historia:	
Prioridad en negocio:	Riesgo en desarrollo:
Requerimiento Funcional:	Iteración asignada:
Programador responsable:	
Descripción:	
Observaciones:	

Fuente: elaboración propia.

A continuación, se describe cada uno de los elementos que componen la plantilla de las historias de usuario:

- Número: número secuencial que se realiza en función de identificar las historias creadas.
- Usuario: El usuario, o actor que interviene dentro de la funcionalidad al momento de emplear el sistema. Es decir, que tipo de usuario efectuará esa funcionalidad.
- Nombre de historia: breve descripción de la historia de usuario se recomienda que se relacione a los requerimientos funcionales de manera directa.
- Prioridad en negocio: Estima la necesidad y prioridad que dicha funcionalidad posee en la aplicación web, para ello se consideran las reglas de negocio y las indicaciones realizadas por el cliente. Su descripción esta detallada por tres indicadores <<Alta, Media, Baja>>
- Riesgo en desarrollo: La posibilidad de que exista algún retraso dentro de la misma, y ocasione riesgos dentro de la planificación, es una idea y estimación del gerente y analista. Su descripción esta detallada por tres indicadores <<Alta, Media, Baja>>
- Requerimiento funcional: es necesario especificar el requerimiento funcional que busca resolver la historia de usuario, se hace uso de la codificación establecida en la etapa de análisis.
- Iteración asignada: indica la iteración en la cual se desarrollará dicha funcionalidad. Se suele ubicar en funcionalidad de la priorización de requerimientos realizada.
- Programador responsable: Se identifica el miembro del grupo de desarrollo encargado de realizar y evaluar dicha funcionalidad en la aplicación.
- Descripción: Se detalla de manera extendida la funcionalidad a realizar. Debe contestar a las preguntas ¿Quién lo va a hacer?, ¿Qué va a hacer?, y ¿Cómo lo va a hacer? Además, debe estar definido en un lenguaje entendible para el usuario y para el gerente, sin tecnicismos.
- Observaciones: en caso de presentarse indicaciones adicionales, u observaciones que el equipo de desarrollo deba considerar durante la implementación del mismo.

Por ejemplo: *Siguiendo con el ejemplo del sistema bancario se estable una historia de usuario.*

Tabla 36. Ejemplo de historia de usuario.

Historia de Usuario	
Número: 1	**Usuario:** Cajero
Nombre historia: Visualizar transacciones diarias.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Baja
Requerimiento Funcional: RF-01	Iteración asignada: 1
Programador responsable: Michael Arias	
Descripción: Como cajero quiero poder visualizar las transacciones realizadas durante todo el día mediante la generación de un reporte.	
Observaciones: En el informe deben estar especificados el monto, hora y cliente de quien se realizó la transacción.	

Fuente: elaboración propia.

4.5. Definición de entregables

El gerente del proyecto está obligado a emitir una serie de documentos, conocidos como entregables, los cuales brindan al cliente y al gerente conocimiento sobre las actividades que se realizaron, realizan, y realizarán durante el desarrollo del proyecto. Los entregables presentan un elemento identificativo de la comunicación entre el cliente y el grupo del proyecto, además de encontrarse inmerso en cada etapa del proyecto, debido a que estos son generados en cada una de las iteraciones.

SWIRL no define los entregables específicos dentro de cada iteración, pero si considera necesario que se presenten como mínimo los siguientes documentos:

- Priorización de los requerimientos.
- Cronograma de actividades.
- Planificación de actividades por iteraciones.
- Plan de contingencia de riesgos.

Adicional a ellos, el gerente puede agregar los entregables que considere necesarios, sin embargo, es imprescindible que especifique mediante un documento el periodo en el que se realizará la entrega respectiva de cada uno.

4.6. Gestión de cronograma

Gestionar el tiempo es una de las prioridades que debe considerar el gerente de proyecto, ya que debe distribuir los recursos humanos acorde a las tareas que mejor desempeñen para lograr abarcar el desarrollo completo dentro del plazo establecido con el cliente. Es importante recalcar, que el cronograma es uno de los elementos fundamentales para que un proyecto de software pueda llegar a tener éxito, y la responsabilidad total recae sobre el gerente del proyecto.

El gestionar el tiempo involucra diversas actividades, SWIRL considera como primordiales tres de ellas:

- La elaboración del cronograma.
- El establecimiento de la holgura.
- La planificación de las iteraciones y sus actividades.

Es importante que el gerente considere que el tiempo establecido en cada una de las actividades sea el más acorde, para ello se debe considerar la holgura dependiendo de la complejidad de la actividad y la cantidad de personas que intervienen en su ejecución.

¿Qué técnicas puede el gerente emplear?

El gerente al ser el responsable total de la gestión del cronograma, debe contar con posibles herramientas que verifiquen las gestiones que realiza, una de ellas y que es muy importante es la experiencia, en base a la experiencia adquirida el gerente puede estimar el tiempo acorde. Otros métodos empleados son la reutilización de proyectos anteriores, como base y guía para la nueva planificación; y finalmente, la ayuda de expertos en gestión de proyectos.

4.6.1. Elaboración del cronograma

El cronograma es un documento crucial dentro del desarrollo del proyecto, además de indicar la distribución del tiempo que se emplea para cada actividad, brinda al equipo de desarrollo una idea de la secuencia de las actividades a realizar, y los encargados dentro de cada una. Es importante que al elaborar un cronograma se consideren los siguientes puntos:

- Descripción de la actividad.
- Tiempo estimado (en horas).
- Personal encargado.
- Recursos requeridos.
- Secuencia de la actividad.

Una buena práctica para generar un cronograma es el uso de herramientas de gestión de proyecto, las cuales permiten elaborar cronogramas, y en base a ellos establecer diagramas de PERT, y de GANTT. Los cuales generan una mejor comprensión de la secuencia de cada una de las actividades. SWIRL mantiene como opcional la inclusión de diagramas de GANTT y diagramas de PERT.

4.6.2. Estimación de la holgura

La holgura es un tema de gran relevancia dentro de la gestión de cronogramas. Al estimar la duración de una actividad siempre se debe considerar el peor de los casos, debido a que pueden existir inconvenientes que provoquen un retraso en el flujo normal del mismo. Al considerar un tiempo de holgura en cada actividad se ayuda al equipo de desarrollo a trabajar de forma más eficiente, sin encontrarse presionados. El tiempo estimado de holgura se establece acorde a la complejidad de las actividades y la distribución de la cantidad de personas involucradas en su ejecución; no se puede establecer la misma holgura a todas las actividades o a todo el proyecto, esto se genera de manera específica, en ciertas etapas.

4.6.3. Descripción de las iteraciones

Al ser la metodología SWIRL, una metodología híbrida basada en el modelo iterativo requiere la ejecución de iteraciones o vueltas, lo que en metodologías como la SCRUM se consideran SPRINTS. Definir las iteraciones, no es más que establecer las actividades e historias de usuarios que se realizará en cada una de las iteraciones del ciclo de vida del proyecto. Para ello se realiza una plantilla para cada iteración estimada.

¿Cómo saber que actividades realizar primero?

Para conocer esto, se busca el orden establecido en la priorización de requerimientos. Acorde al grado de importancia dado se estable en que iteración se la procede a desarrollar. Este trabajo es netamente del gerente, ya que el analista fue el encargado de refinar y priorizar los requerimientos.

Tabla 37. Plantilla de priorización de requerimientos.

No Iteración		**Duración total (días)**		
Observaciones				
#	**Historia de usuario**	**Encargado**	**Duración**	**Prioridad**

Fuente: elaboración propia.

Donde:

- No Iteración: Numero de la iteración o ciclo en la que se realizaran las actividades.
- Duración total: Tiempo estimado de la duración de toda la iteración, esta debe ser expresada en días. No siempre es la sumatoria de la duración de cada una de las actividades, ya que dos o más actividades pueden realizarse en paralelo.
- #: Especifica la secuencia de ejecución que tiene cada una de las actividades. En caso de existir actividades desarrolladas al mismo tiempo (en paralelo), el número de secuencia es el mismo para ambos.
- Historia de usuario: Se especifica el nombre o descripción de la historia de usuario a la que se hace mención, también puede describir actividades.
- Encargado: Los miembros del equipo que ejecutan la actividad o llevan a cabo la realización de la historia de usuario.
- Duración: Es el tiempo que se demora en realizar cada una de las historias o actividades, este debe estar en días, y debe considerarse la dificultad y cantidad de miembros involucrados.
- Prioridad: El nivel de prioridad que se estableció en la fase de análisis de requerimiento.

4.7. Gestión de riesgos

El riesgo del proyecto es un evento incierto que, en caso de que ocurra, tendrá un efecto negativo o positivo sobre el objetivo del proyecto. La administración del riesgo del proyecto es un proceso sistemático que identifica, analiza y responde a los riesgos del proyecto (Lledó y Rivarola, 2007, pág. 111).

4.7.1. Plan de prevención de riesgo

El plan de prevención de los riesgos es un documento en el que se especifican las respuestas a cada uno de los riesgos identificados en la etapa de análisis. Mediante la realización del mismo se establece un plan preventivo en caso de que ocurra algún riesgo, y, sobre todo, informa a los miembros del equipo la manera en la que deben actuar cuando ocurra.

Tabla 38. Plantilla de plan preventivo de riesgos.

ID	Riesgo	Actividad detonante	Estrategia de prevención

Fuente: elaboración propia.

Donde:

- ID: identificador con el cual se estableció el riesgo en la etapa de análisis.
- Riesgo: Descripción detallada del riesgo.
- Actividad detonante: Tarea o actividad que puede generar que el riesgo ocurra.
- Estrategia de prevención: Descripción de la estrategia empleada para prevenir el riesgo

4.7.2. Estrategias de minimización

El plan de minimización es también conocido como plan de mitigación de riesgos, y consiste en estrategias que se emplean en caso de que el riesgo ocurra, y no pueda ser solucionado de inmediato o a lo largo del proyecto. Estas estrategias no evitan que el riesgo ocurra, pero si disminuyen el impacto del daño dentro del mismo.

Tabla 39. Plantilla de plan de mitigación de riesgos.

ID	Riesgo	Estrategia de minimización

Fuente: elaboración propia.

Donde:

- ID: identificador con el cual se estableció el riesgo en la etapa de análisis.
- Riesgo: Descripción detallada del riesgo.

- Estrategia de minimización: Descripción de la estrategia empleada en caso de que el riesgo se efectúe, no logra resolver el riesgo, sino reducir su impacto.

4.7.3. Plan de contingencia de riesgos

El plan de contingencia es empleado cuando el riesgo no puede ser ni eliminado (evitado) o mitigado. Significa, efectuar actividades que emplean el riesgo y buscan una solución para que el proyecto siga sin demoras ni inconvenientes.

Tabla 40. Plantilla de plan de contingencia.

ID	Riesgo	Actividad

Fuente: elaboración propia.

Donde:

- ID: identificador con el cual se estableció el riesgo en la etapa de análisis.
- Riesgo: Descripción detallada del riesgo.
- Actividad: Descripción de la estrategia empleada en caso de que el riesgo no pueda ser eliminado ni mitigado.

4.8. Gestión de comunicaciones

Mantener una comunicación efectiva con todos los miembros del equipo de trabajo, ayuda a incrementar el entusiasmo y eficiencia de los miembros durante el proceso general de desarrollo. Las actividades que debe cumplir esta gestión implican la determinación de los medios de comunicación entre los interesados, repartir la información necesaria, y finalmente comunicar de manera comprensible el estado en el que se encuentra el proyecto (Lledó, 2013).

Esta etapa debe desarrollarse pensando en las siguientes interrogantes:

- ¿Qué quiero informar?
- ¿Cómo quiero comunicarme con todos los miembros?
- ¿Qué herramientas emplearé para la comunicación?
- ¿Cuál es la finalidad de la comunicación?

4.8.1. Plan de comunicaciones

El plan de comunicaciones es un documento en el que se establecen las diversas maneras que se emplearán dentro del proyecto para realizar las comunicaciones. SWIRL emplea el uso de las reuniones para mantener constante las comunicaciones, estas reuniones son breves, y se realiza de manera individual con cada grupo de trabajo inmerso en el proyecto.

En esta etapa se logra determinar las necesidades de información de las personas involucradas en el proyecto de esta manera busca definir cómo afrontar las comunicaciones. Además, de identificar las técnicas de comunicación con el equipo de trabajo, se detallan los métodos de comunicación con el cliente, la frecuencia con la que se realizan las reuniones, la finalidad de cada una de ellas, entre otras.

4.8.2. ¿Quién es el encargado de esta planificación?

El director del Proyecto debe tomar en cuenta que en un proyecto intervienen comunicaciones internas y externas, a que nos referimos con esto, el establecer el manejo de los roles que va a desempeñar cada integrante, en el cual se describa en lo que se va desempeñar, de qué manera lo va hacer, y cuanto tiempo va a tardar en dicha tarea, una vez manejado de forma interna se lo puede llevar al resto del equipo de trabajo para así poder formar una discusión sobre los cambios a realizar.

4.8.3. ¿Cuáles son las técnicas que se pueden aplicar?

No todas las comunicaciones del proyecto deben ser escritas, o de manera general se deberían establecer en que momentos deben ser de forma escrita o también vía telefónica o personalmente, esto llega a depender del ambiente en que se trabaje dentro de la compañía o grupo de proyecto.

Podemos nombrar algunas de las formas que existe para poder comunicarse con el director del Proyecto o si es necesario con algún miembro del equipo del proyecto.

- Correo Electrónico.
- Memorando.
- Comunicado informativo.
- Vía telefónica.
- Personalmente.

4.9. Estructura de desglose de trabajo (EDT/WBS)

La estructura de desglose de trabajo o también conocida como EDT es una subdivisión jerárquica de las actividades y tareas que se presentan dentro del proyecto, de manera que se logre descomponer el proyecto en tareas fáciles, y con poco tiempo de implementación requerido (Romano y Yacuzzi, 2011).

La descomposición de trabajos se suele hacer mediante la clasificación en niveles, en donde se representan las actividades generales en los niveles superiores y en los inferiores tareas sencillas de realizar. La representación gráfica de un EDT generalmente es en mapas jerárquicos, representando los niveles en cada subdivisión, sin embargo, no representa la persona que realiza cada una de ellas.

¿De qué manera perjudica el EDT en la gestión del proyecto?

Una correcta subdivisión ayuda al gerente del proyecto a definir de una manera estratégica el tiempo y recursos dentro del proyecto. Además de ayudar a los miembros del equipo de trabajo, a realizar tareas poco complejas y rápidas, manejando de manera adecuada el tiempo. En caso de realizar una subdivisión redundante o errónea durante el EDT el proyecto puede presentar demoras, extensión en el tiempo de entrega, e inclusive inconvenientes durante el desarrollo de sus etapas.

Para realizar un correcto desglose de tareas, se recomienda el uso de prácticas como:

- Emplear el nivel superior para la realización de identificación de módulos o subproyectos.
- Considerar el uso de actividades realizadas por terceros, incluyendo los usuarios finales para no sobrecargar al equipo.
- Identificar el propósito y enfoque de cada uno de los niveles del EDT.

Una representación posible de la estructura de desglose de trabajo puede ser:

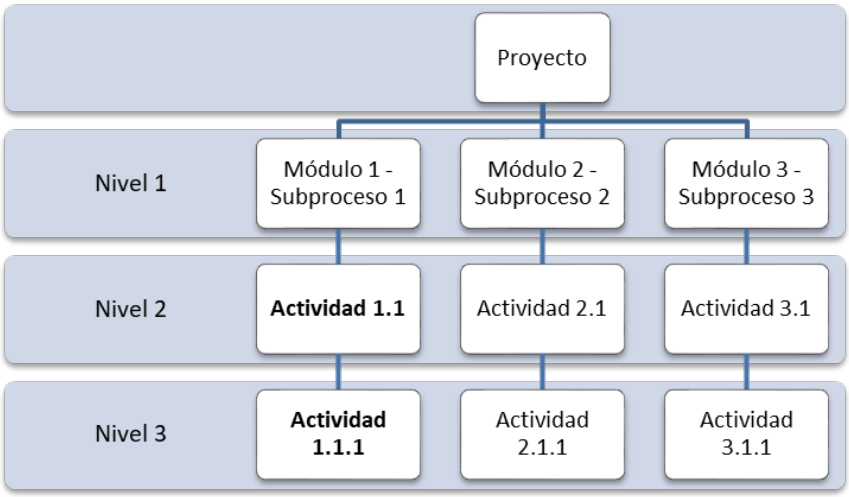

Ilustración 10. Modelo de Estructura de Desglose de Trabajo. **Fuente:** elaboración propia.

4.10. Gestión de cambios

Llevar un control de las modificaciones realizadas es una buena práctica que se ha implementado en proyectos ágiles, considerando las características versátiles que poseen actualmente. Es por eso, que el gerente del proyecto tiene la obligación de llevar un registro detallado de cada cambio o modificación realizado siguiendo la siguiente matriz.

Tabla 41. Plantilla de gestión de cambios.

No	Descripción del cambio	Fase – iteración	Encargado	Fecha

Fuente: elaboración propia.

Donde:

- No: Número secuencial de los cambios.
- Descripción del cambio: Breve descripción del cambio solicitado se debe explicar que se modifica, porque se modifica y quien solicita el cambio.
- Fase – iteración: Etapa del proyecto en donde se realizó el cambio, seguido de la iteración en la cual se realizó el cambio, por ejemplo <<planificacion-3>>, es decir se realizó en la etapa de planificación durante la iteración 3.

- Encargado: Personal encargado de realización la corrección o modificación correspondiente.
- Fecha: La fecha en la que se solicita el cambio, y si se solicita una fecha límite se ubica entre paréntesis.

4.11. Gestión de calidad

La calidad que ofrece la aplicación al cliente es fundamental para definir el éxito o fracaso de un proyecto. La calidad es la forma en como la aplicación cumple con los requerimientos establecidos y los indicadores de calidad de métricas como la ISO 9126, ISO 2500, entre otras.

Por ejemplo: *La calidad de la aplicación puede evaluarse en base a la usabilidad mantenibilidad, fácil aprendizaje, entre otros.*

4.11.1. Plan de aseguramiento y control de calidad

El plan de aseguramiento de calidad implica las actividades que el gerente especifica con la finalidad de que al finalizar el proceso de desarrollo el producto cumpla con dichas expectativas. Una buena práctica para la realización del aseguramiento de la calidad es el benchmarking, comparando el proyecto con otros proyectos ya existentes. Sin embargo, el uso de prototipado ayuda al gerente y clienta a identificar si la aplicación cumplirá o no con los requerimientos establecidos.

Nota:

Para controlar la calidad es necesario considerar las siguientes indicaciones:

- Realizar modelos UML que permitan el fácil acceso a la información y estructura como diagrama de flujo, base de datos, etc.

- Realizar validación en campos requeridos.

- Mediante la inspección de los resultados.

4.11.2. Plan de adquisiciones

Siguiendo a Lledó y Rivarola (2007) "la administración de las adquisidores del proyecto incluye los procesos necesarios para adquirir los bienes y servicios externos a la organización a los fines de lograr el alcance del proyecto" (p. 123).

Los procesos de las adquisiciones están planteados por una serie de actividades, las cuales son seis:

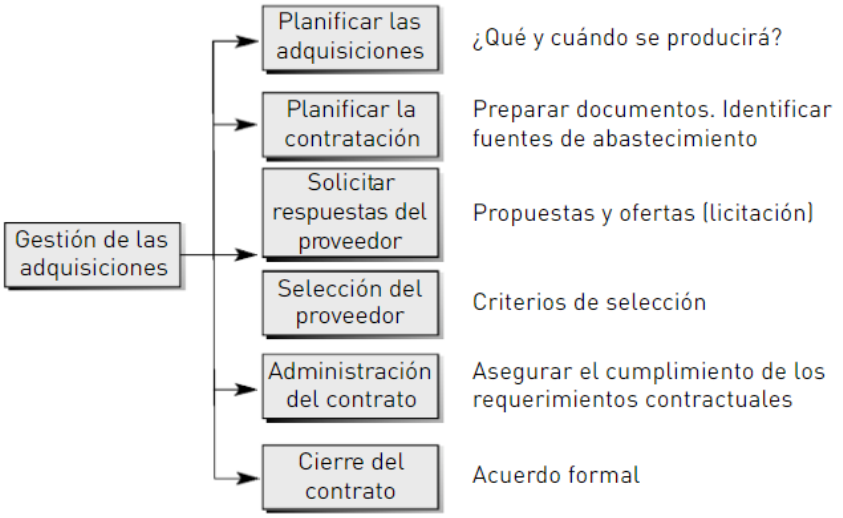

Ilustración 11. Modelo de Estructura de Desglose de Trabajo. **Fuente:** (Lledó y Rivarola, 2007, p. 124)

Para determinar las adquisiciones que posee el proyecto se define la siguiente plantilla, en la cual se definen las tramitaciones adecuadas en cada una de las fases del proyecto:

Tabla 42. Plantilla de plan de adquisiciones.

Encargado:			
Proveedor:			
Etapa:		**Fecha:**	
No	**Adquisición**	**Encargado**	**Fecha**

Fuente: elaboración propia.

CAPÍTULO V: FASE DE MODELADO

La fase de modelado o comúnmente conocida como diseño, es más complejo dentro del desarrollo web que dentro del desarrollo de aplicaciones de escritorios. Esto se debe a que en esta fase se incluye el diseño conceptual y diseño navegacional, de la aplicación orientada a la web. De manera que se consiga un entendimiento claro del desempeño de la aplicación, y sus componentes; además de la navegación entre cada una de las páginas de las que se encuentra compuesta.

5.1. Objetivos

Los objetivos que busca alcanzar SWIRL al incluir esta fase son:

- Diseñar una posible solución que brinde satisfacción a los requerimientos y problemáticas del usuario.
- Modelar las funcionalidades y el comportamiento general del sistema, para brindar facilidad en futuras modificaciones.
- Documentar los procesos y diagramas que representan el comportamiento del sistema, de manera que todos los miembros del equipo de desarrollo logren entenderlo.
- Brindar conocimiento sobre los componentes empleados para el desarrollo del sistema.
- Facilitar la búsqueda y solución de fallas en la etapa de verificación y pruebas.

5.2. Actividades en la fase de planificación

Las actividades primordiales dentro de la etapa son el diseño del modelo de la base de datos, el diseño del modelo de navegación, y uso del lenguaje de modelado unificado UML para el diseño de diagrama. Las actividades completas que comprende esta fase son:

- Diseño del modelo conceptual de base de datos.
- Definición del diccionario de datos.
- Diseño del modelo navegacional.
- Presentación del modelo y correcciones del mismo al cliente.

- Diseño de interfaz abstracta de usuario.
- Presentación del prototipo y correcciones del mismo al cliente.
- Diseño de diagramas UML.
- Generación de entregables y reunión con el cliente.

5.3. ¿Quiénes intervienen?

Las actividades son realizadas principalmente por el analista de software, es será el encargo de diseñar desde el modelo navegacional de la aplicación web, y las interfaces de usuario; hasta la evaluación y modificación de los diseños, para que logren cumplir las expectativas del cliente.

De manera intermedia, el jefe del equipo de desarrollo o programador interviene durante el proceso, él es encargado del diseño y definición de la base de datos, la arquitectura del sistema, y en conjunto con el analista diseñar los diagramas UML. Un rol fundamental es el diseño del diccionario de datos, para un mejor entendimiento de las funciones y nombres asignados por parte del equipo de desarrollo e inclusive los interesados.

¿Cómo interviene el cliente en esta etapa?

Aunque no es de gran relevancia, el cliente también interviene en esta etapa, pero en menor grado que el analista y el desarrollador, ya que su tarea es revisar y aceptar o solicitar modificaciones con respecto de los diseños.

5.4. Diseño de modelo conceptual

Diversas metodologías de desarrollo web orientadas a objetos emplean el modelo conceptual como una ayuda para expresar el comportamiento del sistema de manera clara y fácil de entender (Silva y Mercerat, 2001).

Una definición general del modelo conceptual es el hecho de que se modela la presentación de los datos, generalmente creado en base a los requerimientos y funcionalidades que la aplicación debe cumplir, y como se piensa realizar dichos modelos de manera física.

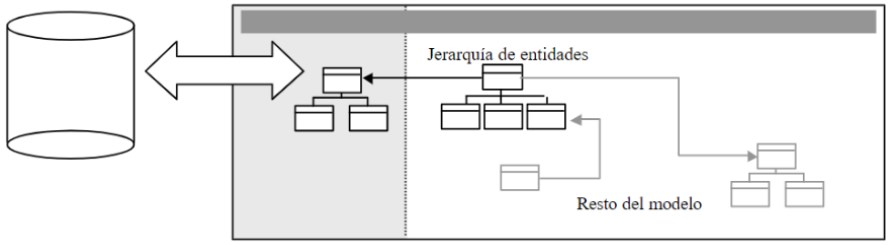

Ilustración 12. Diseño de la capa conceptual. **Fuente:** (Silva y Mercerat, 2001)

5.4.1. Diseño de la base de datos

Aquí se hace referencia al modelo lógico y modelo físico que tendrá la base de datos de la aplicación web, para su modelado es necesario diseñar dos diagramas, el primero es el modelo entidad-relación (conceptual), y en base al mismo diseñar el modelo relacional (físico).

Esta etapa es de suma importancia, y no consiste en la implementación de la base de datos en un sistema gestor de base de datos, sino, se centra en el diseño y especificación de cómo será. Su beneficio principal recae en el equipo de trabajo, debido a que, mediante la realización formal del diseño, cada miembro comprende las funciones, características y en caso de que existiera, información adicional sobre el programa.

- Diseño Entidad-Relación

El modelo de datos entidad-relación o E-R, está basado en una percepción del mundo real que consta de una colección de objetos básicos, llamados entidades, y de relaciones entre estos objetos.

El modelo entidad relación también llamado MER es un tipo de diagrama de flujo que enseña cómo las "entidades", personas, objetos o conceptos, se relacionan entre sí dentro de un sistema.

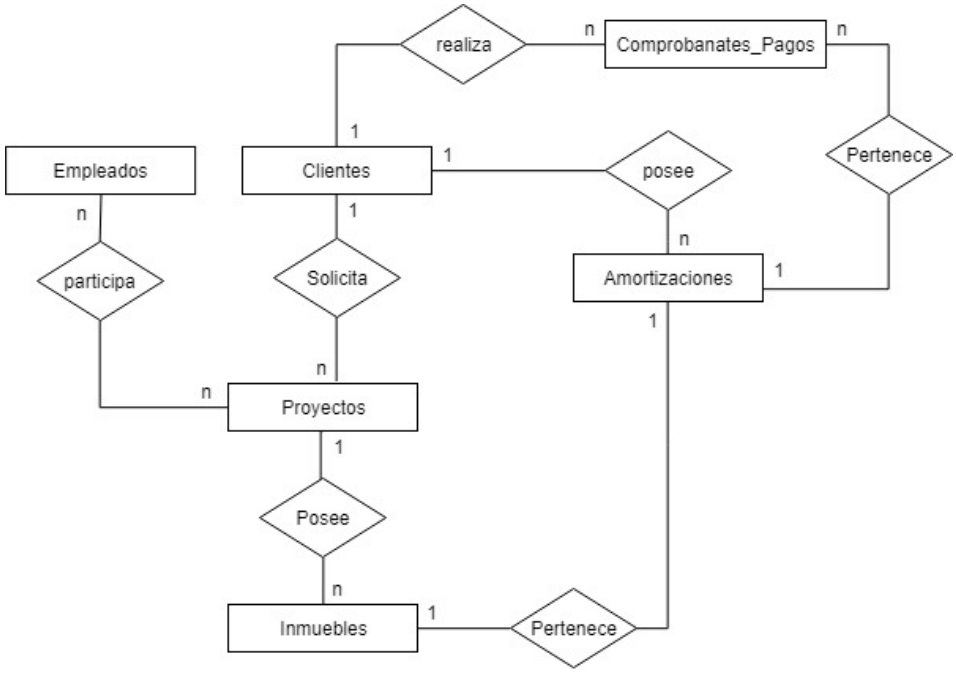

Ilustración 13. Modelo entidad relación. **Fuente:** elaboración propia.

- Modelo relacional

El modelo ERE es un modelo de datos conceptual de alto nivel donde permite facilitar las tareas de diseño conceptual de la base de datos. Es necesario traducirlo a un esquema que sea compatible con un SGBD, ya que El Modelo Relacional es utilizado por la mayoría de los SGBD existentes en el mercado. Este modelo brinda una mejor visión sobre cómo se va a implementar la base de datos, y los parámetros a emplear. De manera que si existe algún error durante la implementación o ejecución de la aplicación el modelo relacional brinda el conocimiento suficiente para encontrar una solución, o la causa principal del problema.

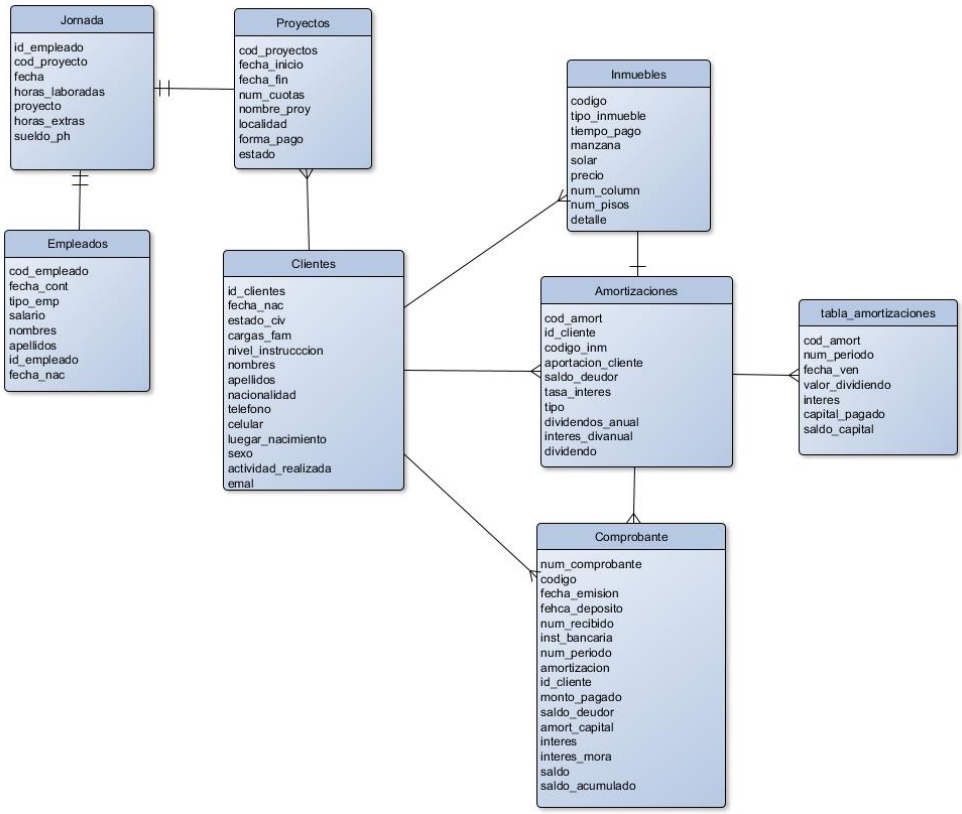

Ilustración 14. Modelo relacional. **Fuente:** elaboración propia.

5.4.2. Diccionario de datos

El diccionario de datos consiste en la elaboración descriptiva de los componentes del sistema, de manera que las partes interesadas, es decir, el cliente y el equipo de trabajo, pueden entender de manera clara que datos y tipos de entradas necesita el sistema, que resultados ofrece, y cuál es la información que se deberá almacenar.

Por esa razón, es fundamental que el gerente y desarrollador del proyecto tengan presente la elaboración adecuada y clara de la base de datos, para ayudar no solo a los miembros del equipo de desarrollo, en el proceso de implementación, sino como ayuda necesaria en las fases de revisión y pruebas.

Nota:

Un diccionario de datos se construye con la finalidad de mantener informado a los miembros del equipo, adicionalmente busca los siguientes objetivos:

- Estructurar ordenadamente los elementos dentro de la aplicación.

- Definir los datos (tipos y características) que se almacenarán en la base de datos.

- De qué manera se relacionan las tablas.

5.4.3. ¿Cómo realizar un diccionario de datos?

Al momento de realizar un diccionario de datos se debe considerar que elementos que se deben incluir. Un diccionario está compuesto por dos partes principales, los datos y sus descripciones. Al hablar de datos se hace referencia a los campos requeridos, son datos que al agruparse generan un significado potencial para el usuario. Mientras que, al hablar de descripciones se hace referencia a como se representan y que relaciones poseen con otros datos.

A pesar de solo estar compuesto de estos dos componentes, un diccionario de datos es extenso e informativo, porque cada elemento se subdivida en pequeñas y relevantes características como se explica a continuación.

Tabla 43. Plantilla de gestión de cambios.

DATO	
Identificador:	Representa el código con el que se diferenciara.
Representación	Breve descripción de que información representa.
Alias:	Forma abreviada de mencionarlo.
Longitud de dato:	El tamaño del dato que se piensa ingresar.
Posibles valores:	Se debe especificar si acepta cualquier valor.
Valores por defecto:	Especificar si se establecen valores por defecto.
Tipo de dato:	El tipo de dato al que representa, <<entero o decimal>>
DESCRIPCIÓN	
Relación con componentes:	De qué manera se relaciona con las demás estructuras de datos. <<secuencial, selección, iteración, opcional>>
Notación en la base de datos:	Es la representación que los analistas emplean para identificarlos con facilidad en los diagramas conceptuales

Fuente: elaboración propia.

5.5. Diseño navegacional

La capa de navegación es una característica propia de las metodologías de desarrollo web. La implementación de diseños que representen de manera gráfica el

comportamiento del sistema es de gran utilidad, para la comprensión y aceptación del cliente sobre las planificaciones realizadas sobre el proyecto.

SWIRL considera los mismos componentes que las otras metodologías de desarrollo web para modelar la navegación dentro del sistema. Entre ellos se encuentran los nodos, que representan las páginas o contenedores a desarrollar; los enlaces, los cuales representan los redireccionamientos de una página a otra; y las estructuras de acceso.

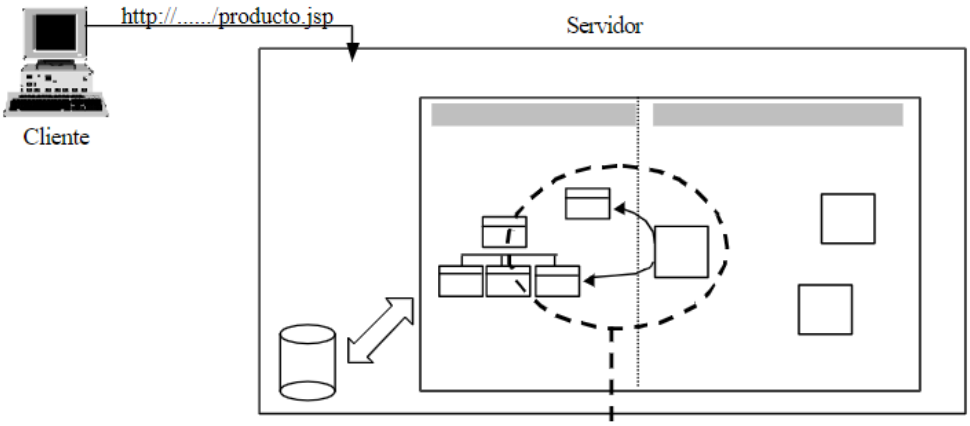

Ilustración 15. Diseño de la capa navegacional. **Fuente:** (Silva y Mercerat, 2001)

Nota:

Al igual que la capa de modelo conceptual, el diseño navegacional brinda mayor beneficio al equipo de proyecto durante las etapas de implementación y verificación y prueba. Esto se debe a que, en caso de existir un error durante pruebas de conexiones, pruebas de respuesta a usuarios específicos, o pruebas de direccionamiento en las páginas; mediante el uso del diagrama de navegación se consigue comprender el punto específico o al menos aproximado de donde se encuentra el error.

Para diseñar un modelo navegacional no se posee una nomenclatura fija, simplemente debe cumplir la condición de que sea entendible por el cliente, y por el equipo de desarrollo.

5.5.1. Diseño de enlaces navegacionales

El diseño de enlaces navegacionales se enfoca en la especificación de las redireccionamientos de páginas y contenidos que se realizan dentro de cada página. Para ello se realizan relaciones entre las páginas que lo requieran con la finalidad de conocer que respuestas el sistema efectuará. Es importante que los enlaces navegacionales se diseñen en base a dos conceptos:

1. Representen los enlaces entre las páginas para realizar una funcionalidad.
2. La navegación sea defina en base a cada usuario que interactúa directamente con el sistema.

Para el diseño navegacional general del sistema se emplea el uso de páginas (contenedores) y enlaces (direccionamiento).

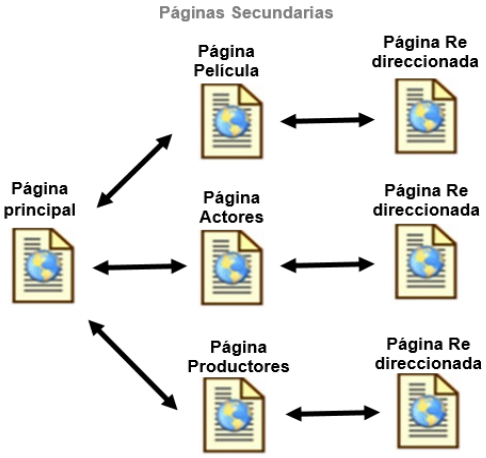

Ilustración 16. Enlace navegacional de la aplicación. **Fuente:** elaboración propia.

Para el diseño navegacional por usuarios, se emplea el uso de actores. Este sirve para conocer de qué manera el usuario puede trabajar con el sistema y cuáles son las respuestas esperadas, es decir, como puede navegar a través de la aplicación.

Ilustración 17. Enlace navegacional del usuario. **Fuente:** elaboración propia.

5.5.2. Descripción textual del modelo

Consiste en describir narrativamente las navegaciones realizadas como respuesta a una funcionalidad del sistema. Se debe especificar cada una de las navegaciones del

usuario. Y para mayor comprensión separarlas por cada acción que realiza el usuario. Para la descripción textual se debe tener en cuenta:

- El actor que efectúa dicha acción.
- La acción que activa la navegación.
- El enlace y redireccionamiento a otra página.
- La acción de corte, es decir la acción que permita finalizar la navegación o retornar al inicio.

5.5.3. Definición de la arquitectura del sistema

La arquitectura del sistema es definida por el analista del sistema, para ello es necesario que conozca el comportamiento del sistema para detallar el patrón de diseño más adecuado considerando las características que provee.

Un patrón se define como un modelo que sirve para obtener un elemento igual. En el mundo de los sistemas la experiencia y la investigación han arrojado muchos patrones que hoy en día son utilizados para resolver problemas semejantes de la mejor forma. Un patrón de diseño es la solución recurrente para problemas comunes.

Para definir la arquitectura a emplear deben tenerse en cuenta los siguientes elementos:

- Las interfaces o páginas por implementar.
- Los componentes o elementos como bases de datos, reportes, archivos, entre otros.
- Las actividades que dan secuencia a la gestión de nuevas páginas.

Los patrones tienen la finalidad de diseñar la comunicación entre los objetos y asignar de manera sencilla las responsabilidades a cada uno de los usuarios. Para la representación de lo mencionado anteriormente se pueden emplear una serie de patrones:

- Patrones de comportamiento: Estructura el comportamiento que poseen los componentes del sistema, de esta manera se logra conocer cuales elementos deben ser acoplados y cuáles no, reduciendo el tiempo de ejecución.
- Bstract Factory: Este patrón busca la comunicación entre elementos de diferentes tipos, identificando el tipo que se emplea para su comunicación.

- Builder: Esta arquitectura se concentra en la existencia de una sola clase creadora, la cual construye las demás clases y procesos complejos.
- Factory Method: Similar al builder, consta de la creación de subelementos centralizadas en un componente, con la excepción de que el usuario no elige el tipo del elemento.
- Prototype: Se basa en la creación de elementos nuevos en base a prototipos o elementos creados con anterioridad.

¿Qué pautas seguir para realizar un correcto modelado de patrón de diseño?

El proceso de selección y elaboración de patrones de diseño, requiere gran habilidad para interpretar de la manera más adecuada la información y funcionalidades que el sistema implementará. Para ello se recomienda conocer todos los patrones posibles, y comprender su uso, funcionamiento y finalidad de creación, algunos documentos que presentan una descripción detallada de los mismos es:

1) (Debrauwer, 2013)

2) (Gamma, 2002)

3) (Debrauwer y Evain, 2015)

5.6. Diseño de interfaz abstracta de usuario

Las interfaces graficas de usuario generan un ambiente familiar para el cliente, además de guiar de manera gráfica al usuario final a través de las funcionalidades que presenta la aplicación web. El diseño de interfaces abstractas se basa en el modelado de prototipos, especificando los elementos de los que se encuentran compuestos.

La capa de interfaz abstracta no se basa simplemente en la modelación de los posibles diseños que se deben implementar en las etapas posteriores; sino, se basa en modelar los requerimientos de usuarios, en componentes gráficos, estéticamente agradables y que brinden la información suficiente para que el usuario de manera intuitiva pueda navegar dentro de sus funcionalidades.

Nota:

Al igual que la capa de modelo conceptual y navegacional, el diseño de interfaz brinda mayor beneficio al equipo de proyecto durante la etapa de implementación, debido a que ofrece una idea general de la estructura de la aplicación. Además, permite a los programadores conocer que tipos de datos son requeridos en cada proceso y como presentar la información ante los usuarios.

El diseño puede ser realizado de diversas maneras, para agilizar el proceso de creación y sobre todo entendimiento, se hace uso de las herramientas CASE de modelado de interfaz, como balsamiq Mockups; sin embargo, existen analistas que desean realizar

los diseños de manera convencional, es decir, a lápiz y papel. Indiferentemente, del método que se emplee para el desarrollo, se debe verificar que el diseño cumpla con ciertos requisitos, los cuales son:

- Las interfaces deben cumplir los estándares de usabilidad en base a métricas de evaluación.
- Deben cumplir los requerimientos que el cliente ha especificado.
- Debe ser fácil de entender y brindar facilidad de entendimiento.
- Debe ser responsiva.
- Presenta d manera clara la información requerida por el usuario.
- Representa de forma clara la navegación, conectividad y dependencia entre interfaces.

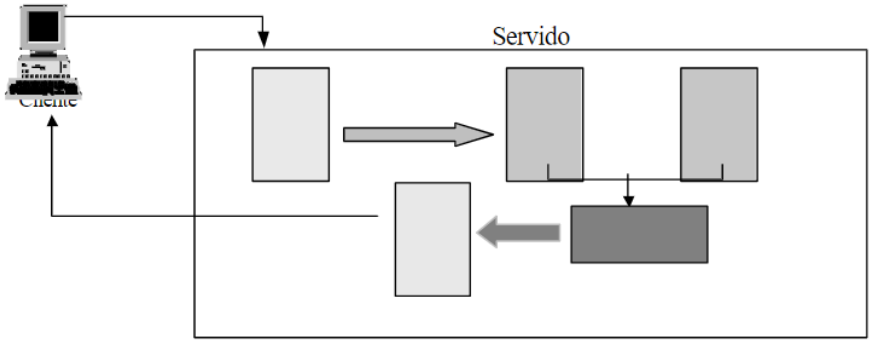

Ilustración 18. Diseño de la capa de interfaz abstracta. **Fuente:** (Silva y Mercerat, 2001)

5.6.1. Interfaces externas

El termino de interfaces externas dentro de SWIRL se considera como los elementos que todas las interfaces poseen en común, es decir, los componentes estáticos de la aplicación. Es requerido la clasificación y especificación de tres categorías de interfaces en esta etapa:

- Interfaces comunes de usuario.
- Interfaces de dependencia (Si requiere)
- Interfaces de comunicación (Opcional).

Las interfaces comunes de usuario especifican los componentes de interfaz, como botones, iconos, tipo de letra, que todas las interfaces deben contener de manera estandarizada. Es necesario que se especifique el diseño, formato y función de cada uno de los componentes comunes, ya que el usuario debe lograr familiarizarse con las interfaces, sin cambiar drásticamente el modelo de presentación de una tarea a otra.

Por ejemplo: *Una situación muy común cuando se realizan diseño de interfaz de usuario, es el uso de íconos o imágenes en los fondos de botones, pues bien, el diseño que poseen para los botones repetidos en las interfaces como "guardar, cancelar, modificar", deben ser similares y encontrarse en la misma ubicación (o similar) en cada una de las interfaces. Adicional, a esto algunos ejemplos de interfaces comunes son:*

- Presentación de mensajes informativos (bienvenida, error, proceso exitoso, etc).

- La estructura para representar la información/ datos (mediante tablas, de manera textual, separada, etc).

- La ubicación de componentes para el ingreso de datos (Derecha, superior, inferior, etc)

Es importante que se especifiquen los componentes y tipos de diseños empleados en el mismo dentro de la interfaz debido a que ayudan tanto al cliente como al desarrollador a conocer el comportamiento y forma de navegar del sistema.

Interfaces de dependencia

Las aplicaciones web, comúnmente suelen emplear navegadores o plugins internos para presentar páginas oficiales, o necesarias para que la aplicación cumple con su cometido. Las interfaces de dependencia hacen referencia a las páginas web u otras aplicaciones que se encuentran inmersas o redireccionadas dentro de la aplicación, en caso de que existan, y que no pertenecen al diseño general de aplicación web. Para especificarlas, se debe indicar los siguientes parámetros.

- El nombre de la página.
- La funcionalidad que cumple dentro de la aplicación.
- La historia de usuario (código o identificador) a la que se vincula. En caso de ser necesaria para efectuar algún requerimiento.
- La interfaz propia del programa de la cual depende.
- Los riesgos que se pueden presentar.

Para ello se realiza la siguiente plantilla.

Tabla 44. Plantilla de interfaces de dependencia.

Nombre / ID	Funcionalidad	Historia de usuario	Interfaz dependiente	Riesgos

Fuente: elaboración propia.

Nota:

Estas interfaces se deben indicar durante el desarrollo del proyecto siempre y cuando la aplicación dependa de otra(s) interfaces o páginas (se incluyen aplicaciones) para su correcto funcionamiento.

Interfaces de Comunicación

Todo proyecto debe ser implementado, codificado y testeado mediante una herramienta de programación, las interfaces de comunicación hacen referencia a las aplicaciones, o programas con los cuales el desarrollador debe interactuar para lograr desarrollar el producto final. Las interfaces que considerarse para desarrollar una aplicación web son:

- Editor de texto.
- Host para el almacenamiento de base de datos (si requiere).
- Programa desarrollador.
- Entre otros.

5.6.2. Tendencias de diseño

El proceso de implementación de la interfaz de usuario requiere un modelo específico en el cual basarse, a esto se conoce como tendencia de diseño. Las tendencias de diseño permiten al desarrollador conocer el comportamiento de los componentes de la interfaz de usuario, las características que las interfaces deben cumplir, y finalmente el estilo de presentación.

Las tendencias de diseño incrementan con el pasar del tiempo, para lograr seleccionar una tendencia de diseño adecuada el desarrollador debe conocer el ambiente en el que se empleará la aplicación y que características poseen los usuarios. Las tendencias con mayor auge y uso dentro del desarrollo web son:

- Responsive design.
- Flat design.
- Progressive web apps.
- Mediante animaciones.

5.6.3. Prototipado de interfaces de usuarios

El prototipado de las interfaces de usuario se desarrolla mediante el uso de herramientas CASE de diseño, las cuales permiten especificar el diseño global de las plantillas de las interfaces, de manera que el desarrollador conozca cómo realizar cada una de estas, ahorrando tiempo y esfuerzo.

El diseño se puede realizar también a mano, mediante el uso de papel y lápiz, sin embargo, se debe verificar que el diseño sea comprensible y sin mal interpretaciones.

Crear prototipos de interfaces beneficia a ambas partes interesadas, al equipo de trabaja beneficia al momento de minimizar los recursos de tiempo y esfuerzo, además de brindar un conocimiento claro, general y detallado de cómo quedará la aplicación final.

Por otro lado, beneficia al cliente al brindarle información de cómo funcionará el programa, el uso de imágenes es más comprensible que el uso de texto al momento de explicar el funcionamiento, además, es posible que, durante las reuniones para la explicación y validación de las interfaces, el usuario modifique o agregue condiciones a los requerimientos funcionales establecidos.

Una de las herramientas CASE con amplio uso es Balzamiq Mockups, la cual brinda una gran gama de componentes destinados para el diseño de interfaces. Los sketch o bocetos realizados de la interfaz son visualmente atractivos y fácil de entender por cualquier persona, posea o no conocimientos del proyecto. En base al diseño, los desarrolladores deben implementar el proyecto, lo más semejante posible.

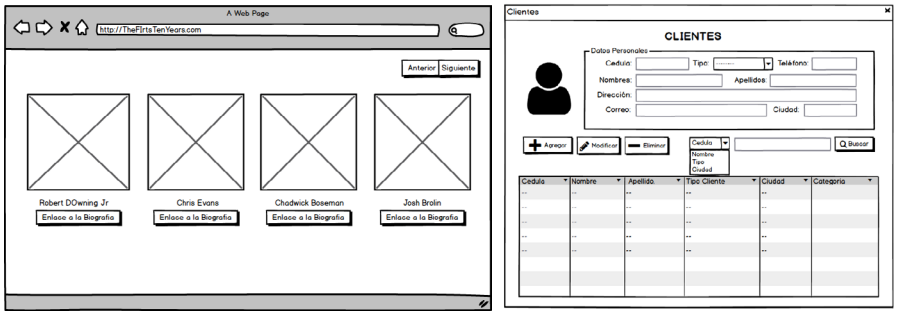

Ilustración 19. Ejemplo de prototipos de interfaz de usuario. **Fuente:** elaboración propia.

5.7. Diseño de diagramas UML

El paradigma orientado a objetos se ha empleado en gran cantidad dentro del diseño de sistemas de información, debido a la inmensa cantidad de beneficios que este ofrece al desarrollador. El modelado de diagramas UML permite aprovechar en su porcentaje total, las ventajas que ofrece el paradigma orientado a objeto.

Es importante recalcar que los diagramas UML se enfocan más en el modelado del comportamiento del sistema, más no por completo en el diseño de bases de datos. Los modelos generados brindan al equipo conocimiento sobre los componentes que posee el sistema, los actores que intervienen, las actividades y secuencia en la que estas se realizan, entre otros.

¿Cómo diseñar los diagramas UML?

El proceso de diseño de os diagramas está especificado según el tipo de diagrama a realizar, además cada uno tienen su finalidad y sus componentes respectivos. A continuación, se presentan algunas referencias, que proveen conocimiento básico para realizar un buen modelado de diagramas UML.

1) (Stevens y Pooley, 2007)

2) (Ríos, Ordóñez, y Tapia, 2017)

3) (Booch, Rumbaugh, Jacobson, García Molina, y Sáez Martínez, 2006)

A pesar de que existe una gran variedad de modelos UML, SWIRL toma en consideración los siguientes diagramas para la representación adecuada del sistema, cada uno representa un conocimiento diferente dentro del proceso de desarrollo, mediante los mismos se pueden representar entidades del mundo real, destacando el comportamiento normal de la aplicación y simulándolo en un entorno con situaciones cercanas a la vida real. Sin embargo, a pesar de todos los beneficios que brinda, el más significativo es la descomposición de procesos complejos, en módulos o procesos fáciles de comprender. SWIRL emplea obligatoriamente cuatro diagramas UML, los cuales son el diagrama de actividades, de secuencia, de colaboración, y de

componente; mientras que de modo alternativo u opcional establece dos diagramas, los cuales son diagramas de estados, y de distribución. Estos diagramas representan el comportamiento que poseerá el programa de una manera comprensible.

Diagrama de actividades

- Representa el flujo de tareas realizadas por el sistema dentro de una función en específico, características de la aplicación o componente. De esta manera el gerente y el equipo de trabajo conocer el comportamiento dinámico que se genera de manera interna en el programa al realizar una actividad determinada.

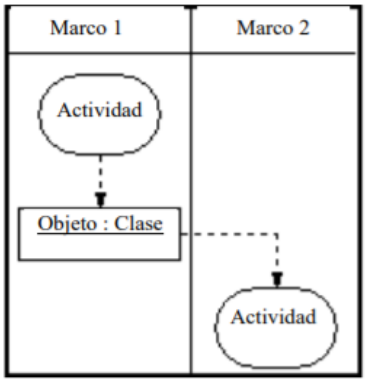

Ilustración 20. Diagrama de actividades. **Fuente:** (Hernández, s. f.)

Diagrama de secuencia

- Representa como su nombre lo indica la secuencia que seguirán las actividades internas del sistema, adicional a ello se logra conocer los objetos que interactúan dentro de estas actividades, y como realizan su comunicación.

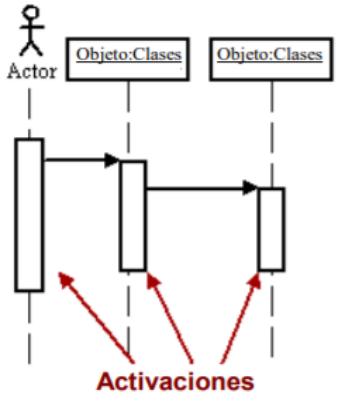

Ilustración 21. Diagrama de secuencia. **Fuente:** (Hernández, s. f.)

Diagrama de estados (opcional)

- Este diagrama es opcional, y representa el estado particular de un objeto contemplado dentro del sistema en una situación en específica. Se emplea con la finalidad de capturar comportamiento real del sistema.

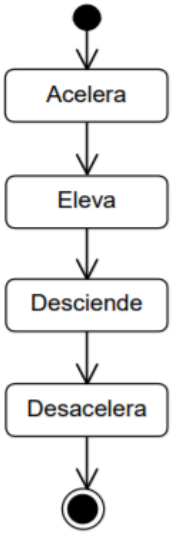

Ilustración 22. Diagrama de estados. **Fuente:** (Hernández, s. f.)

Diagrama de colaboración

- Este diagrama es de suma importancia debido a que describe de una manera específica las interacciones que se realizan entre varios componentes dentro del sistema, basándose en los diagramas de secuencia y casos de uso.

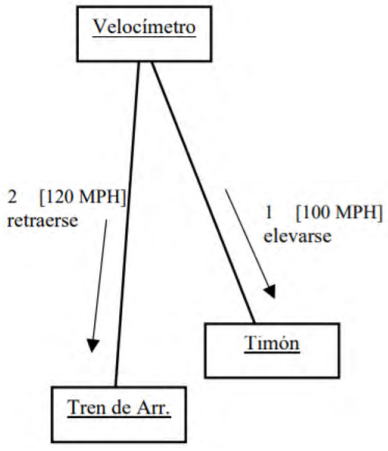

Ilustración 23. Diagrama de comportamiento. **Fuente:** (Hernández, s. f.)

Diagrama de componentes

- Mediante el diagrama de componentes se logra identificar los elementos que conforman un sistema, como se relacionan y si poseen dependencia entre los mismos.

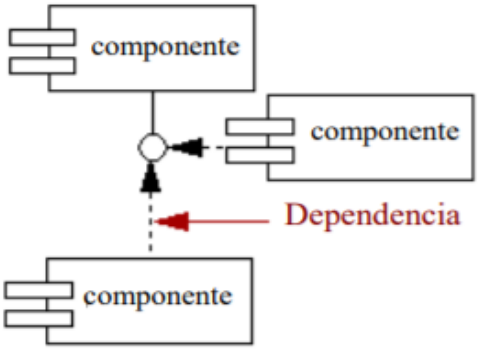

Ilustración 24. Diagrama de comportamiento. **Fuente:** (Hernández, s. f.)

Diagrama de distribución (opcional)

- Es empleado con la finalidad de presentar la estructura física de la aplicación a desarrollar, dentro del mismo se identifican los componentes del sistema, las conexiones y relaciones entre los mismos. Es importante mencionar, que un componente puede representar recursos físicos empleados para ejecutar componentes.

CAPÍTULO VI: FASE DE IMPLEMENTACIÓN

La etapa que emplea la mayor cantidad de tiempo dentro de una metodología de desarrollo web, es la fase de implementación y codificación de la aplicación web, esto se debe a que no solo se codifican las funcionalidades que se han especificado en fases previas, y programan las interfaces de usuario; también se debe llevar un control de código limpio, mediante el uso de estándares de codificación, y un control de la calidad de cada componente programado, mediante estándares de calidad.

Esta etapa se encuentra en constante iteración, en cada ciclo se debe realizar cambio, implementar nuevas funcionalidades, y en ocasiones simplemente ejecutar validaciones recientes que el cliente solicita.

6.1. Objetivos

Los objetivos de esta etapa son los más sencillo de comprender, pero los más complejos de cumplir Debido a que el proceso de codificación modular implica la participación de diversos involucrados, y la validación de cada componente creado es complejo de elaborar. Por consiguiente, esta etapa posee los siguientes objetivos:

- Implementar las funcionalidades de la aplicación web.
- Brindar un producto de calidad y fácil de entender para el usuario.
- Cumplir las expectativas del cliente al emplear la aplicación.
- Diseñar las interfaces y especificaciones del comportamiento del sistema con gran similitud a las características detalladas en fases anteriores.

6.2. Actividades en la fase de implementación

La codificación, consta de etapas y tareas que deben ser repartidas entre los diferentes miembros del equipo de desarrollo, debido a su complejidad y gran cantidad de funcionalidades. Para cumplir con los objetivos mencionados en el apartado anterior, SWIRL plantea las siguientes actividades dentro de la etapa de implementación.

- Separación de tareas entre los miembros de trabajo acorde a una categoría seleccionada (es decir, dividir el trabajo por módulos, por funcionalidades, por interfaces, entre otros).

- Diseño de las interfaces de usuario modeladas en la fase anterior.
- Codificación de las funcionalidades y componentes de cada una de ellas.
- Validación de las restricciones globales y generales delimitadas por el cliente.
- Implementación de métricas de usabilidad y calidad en las interfaces de usuario.

6.3. ¿Quiénes intervienen?

Dentro de esta etapa los únicos miembros que intervienen son el equipo de programación o desarrollo de manera completa, y de manera leve el cliente (considerado por las reuniones, y las modificaciones especificadas al prototipo de aplicación diseñado).

Nota:

El gerente interviene de manera personal con el cliente durante las reuniones de verificación del prototipo de la aplicación, sin embargo, no se considera como un actor principal dentro de esta etapa. El gerente en esta etapa se comunica directamente con el cliente para recabar modificaciones o nuevas implementaciones, y posteriormente comunicarle al equipo desarrollador las implementaciones y cambios a realizar dentro de la aplicación.

6.4. Elementos de la fase de implementación

Para lograr un buen proceso dentro de la etapa de implementación es necesario que el equipo de desarrollo considere los siguientes componentes. La comunicación dentro de esta fase es primordial, los miembros del equipo de desarrollo deben comprender las ideas y comportamiento del programa, deben conocer cuáles son los estándares y terminologías empleadas dentro del código fuente, de manera que cualquier miembro comprenda posteriormente el código. Este punto es importante ya que se debe suponer el peor de los escenarios, en el que se adhieran miembros nuevos al equipo y una modificación del código fuente de la aplicación sea necesario.

- Estándares de codificación y de calidad empleados para el conocimiento global del equipo de desarrollo.
- Definición de los componentes y detalles generales de los módulos implementados dentro del sistema.
- Diseño y codificación de las interfaces de usuario y páginas principales.

6.4.1. Estándares

Un estándar es una modelo habitual que seguir, el cual reúne las características necesarias para ser comprendido y aplicación por todos los miembros participantes de una actividad. Los estándares benefician a la comunicación entre los miembros del equipo de desarrollo, generando una referencia habitual para el proceso de desarrollo de las funcionalidades o módulos de la aplicación.

Estos estándares se dividen en dos categorías, el primero consiste en los estándares empleados durante la codificación del código fuente, mientras que el segundo se basa en estándares de evaluación de calidad, verificando que el producto final cumpla con las especificaciones y normas de calidad necesarias para brindar completa satisfacción al cliente.

6.4.1.1. Estándares a nivel de codificación

Los estándares de codificación que los desarrolladores deben adoptar dentro del proceso son la sangría, sentencias, e identificadores.

- Sangría: debe ser empleado en función de permitir la facilidad de entendimiento de código, este se conforma de máximo 4 espaciados como unidad de sangría.
- Sentencias: se especifica que cada línea de código posee una sentencia, de esta manera se mantiene segura la multi-operación de iteraciones.
- Identificadores: comúnmente los desarrolladores no especifican la manera en la cual identifican o asignan nombres a variables, pero es de suma importancia que estos se identifiquen y se conozcan por los demás miembros del equipo de desarrollo.

Las nomenclaturas se deben realizar mediante un orden determinado, de esta manera beneficiara en un futuro a las personas que deberán entender la codificación.

Tabla 45. Plantilla de estándares de codificación.

Identificador	Nomenclatura	Excepciones	Descripción

Fuente: elaboración propia.

Donde:

- Identificador: especifica el componente, tipo de variable o elemento dentro del código. Por ejemplo, un identificador puede ser, variables globales, variables estáticas, funciones, sentencias, etc.
- Nomenclatura: especifica la manera en la que se asigna algún componente, puede estar sujeto a letras o variables fijas, o asignaciones referentes a los contenidos y funcionalidades de la variable.
- Excepciones: especifica las situaciones en las cuales no se usa o se omite la nomenclatura especificada.
- Descripción: mediante el mismo se puede conocer el comportamiento del sistema además de detalles más profundos de los elementos que lo conforman.

Por ejemplo: *Algunos ejemplos de estándares para la codificación de la aplicación web son:*

- Los tags tienen que estar escritos en minúsculas. Por ejemplo <td> en vez de <TD>.

- Los objetos html deben tener id y name, y ambos deben ser iguales.

- Las funciones deben ser identificadas con la primera letra en mayúscula y en caso de tener espacios reemplazarlos por _.

6.4.1.2. Estándares a nivel de evaluación

La evaluación de la página web se realizará mediante dos formas, la primera los integrantes y desarrolladores realizarán una evaluación acorde a las normativas y características de la métrica ISO/IEC 9126; y una segunda evaluación acorde a indicadores de herramientas de evaluación de páginas web, como W3C, o seosite checkup.

Definir los indicadores que se emplearán ayuda a que los desarrolladores conozcan las bases para desarrollar la aplicación web, y brindar al finalizar el proceso de desarrollo una aplicación conforme a las características de usabilidad pertinentes para aplicaciones orientadas a la web.

El uso de métricas ayuda al equipo a conocer las características que la aplicación debe cumplir, y los estándares que se definen deben ser acorde a la gestión del gerente del proyecto. Para ello se define la siguiente plantilla, en la cual se especifica que características y en ocasiones los elementos que se evalúan dentro de cada indicador.

Tabla 46. Plantilla de estándares de evaluación.

Indicador	Característica	Actividad
Funcionalidad		
Fiabilidad		
Usabilidad		
Eficiencia		
Mantenibilidad		

Fuente: elaboración propia.

Donde:

- Característica: Hace referencia a las características que se desean considerar durante la evaluación de la aplicación correspondiente a un indicador en específico. Por ejemplo, en funcionalidad, las características que se desean considerar son idoneidad, exactitud, y seguridad.

- Actividad: En este apartado se define cual es la función, elemento o respuesta del sistema que se debe evaluar, se debe referenciar a la característica e indicador correspondiente. Por ejemplo, en funcionalidad en la característica de seguridad, se evalúa que el sistema mantenga la información y la almacene correctamente en la base de datos.

Por otro lado, se deben definir los indicadores SEO que se evalúan, estos pueden ser relacionados a los indicadores empleados por las herramientas de evaluación de páginas web.

Por ejemplo: *Algunos ejemplos de indicadores SEO a considerar para la evaluación de la aplicación web son:*

- Desempeño/ funcionalidad.

- Tamaño de la página.

- Adaptación de ventana gráfica.

- Meta descripción.

- Navegabilidad.

CAPÍTULO VII: FASE DE PRUEBAS Y REVISIÓN

Según Pérez (2011, p.72):

Las pruebas de unidad deben implementarse con un marco de trabajo que permita automatizarlas, con la finalidad de realizar pruebas de integración y validación diarias, esto proporcionará al equipo un indicador del progreso y revelarán a tiempo si existe alguna falla en el sistema.

Durante el ciclo de vida de un proyecto, cada iteración debe ser verificada y evaluada acorde a sus características, de manera que se cada una de ellas cumpla el cometido que el gerente del proyecto ha definido con anterioridad. La evaluación se realiza al final de cada ciclo o iteración, y para ello es necesario presentar el ejecutable o prototipo de la aplicación de forma funcional, de esta manera el cliente puede identificar de manera sencilla si existe alguna falla.

Las aplicaciones web a diferencia de las aplicaciones de escritorio tradicionales, requieren un proceso de evaluación más complejo. La revisión debe incluir las funcionalidades, enlaces de redireccionamiento, almacenamiento correcto de la información, estadísticas SEO para la aplicación correcta de modelos de negocios, y verificación de las páginas y enlaces que la conforman.

7.1. Objetivos

Las pruebas durante el desarrollo de un proyecto de software web se realizan con la finalidad de comprender los posibles fallos o errores que posee la aplicación en relación con el funcionamiento, validaciones, o usabilidad. Este conjunto de actividades pretende identificar las fallas del sistema o prototipo hasta el momento para posteriormente corregirlos. A más de eso, busca los siguientes objetivos:

- Identificar problemas de interfaz de usuario, entendimiento o usabilidad.
- Identificar fallas de funcionamiento y validación dentro de la aplicación web.
- Corregir los errores identificados.
- Identificar posibles modificaciones o adhesión de componentes en la interfaz de usuario.

7.2. Actividades en la fase de pruebas y revisión

Las actividades de control dentro de esta fase se basan en la ejecución del prototipo de la aplicación web, y revisión de conformidad con los requerimientos y el usuario final. La realización de diferentes tipos de pruebas permite que la aplicación final sea eficiente y brinde la mayor satisfacción al usuario final. Es importante mencionar que la evaluación debe ser considerada desde un enfoque del cliente y los usuarios finales, es decir el uso de la aplicación por parte de terceros. En base a lo mencionado anteriormente, SWIRL propone las siguientes actividades para la realización de la correcta revisión de la aplicación web.

- Control integrado de cambios.
- Realización de las pruebas de integración.
- Detección y corrección de errores.
- Realización de pruebas de sistema (calidad, tendencia y herramientas SEO).

7.3. ¿Quiénes intervienen?

El encargado de la realización de esta fase es netamente el tester o evaluador. El tester es el encargado de ejecutar la aplicación, realizar las pruebas y análisis respectivos para proporcionar una aplicación de calidad. El proceso de esta fase es sencillo, el tester realiza las pruebas de integración y la detección de errores antes de ser presentado ante el cliente y usuario final. Seguido de la realización de las pruebas básicas, el tester y el gerente se reúnen con el cliente, el cual debe visualizar, ejecutar e interactuar con el programa. Durante este proceso el tester debe realizar las pruebas del sistema calidad y tendencia, y adicionalmente identificar si se presentan errores durante el funcionamiento.

¿Cómo el tester puede realizar la evaluación al cliente?

El tester debe ser capaz de reconocer cuando al usuario se le dificulta el uso de una aplicación o se encuentra insatisfecho con la misma. Algunas pautas útiles para el reconocimiento de los mismos son las expresiones faciales que el usuario demuestra al navegar a través de la aplicación, o si genera muchas preguntas de dudas sobre cómo funciona la aplicación. Si el usuario constantemente se encuentra preguntando ¿Cómo accedo a…?, ¿Qué debo hacer después?, ¿Qué hace este botón?, ¿Dónde se encuentra la opción de…?, entonces la aplicación no es intuitiva ni fácil de aprender, por lo que es necesario mejorarla.

Cuando no se presenten errores durante la evaluación con el cliente, y el sistema se encuentre con todos sus módulos en funcionamiento, el tester debe realizar una prueba de verificación a los usuarios finales.

7.4. Control integrado de cambios

Un correcto desarrollo del proyecto se basa en la gestión y control constante de las modificaciones que se presenten durante todo el proceso. Administrar los cambios que se efectúan, integrando el antes y después de cada uno, ayuda al equipo de trabajo a poseer una idea más clara de los procesos y cualidades que el sistema debe y no debe tener.

El control integrado es un documento en el cual se identifican los cambios realizados, la persona que aborda esos cambios, la funcionalidad a la que afecta y la fecha en la que se realiza. De esta manera, ante cualquier problema o incertidumbre planteada sobre versiones anteriores del proyecto se logre identificar de manera rápida.

La gestión de cambios se lleva a cabo en base a la siguiente plantilla, que brinda al gerente un mayor conocimiento sobre el cambio efectuado y las versiones anteriores del proyecto.

Tabla 47. Plantilla de control de cambios.

No	Solicitante	Área.	Fecha	Categoría del cambio	Descripción del cambio

Fuente: elaboración propia.

Donde:

- Solicitante: es la persona que solicita la modificación y quien se designa encargado para la modificación del mismo.
- Área: representa el área donde se desempeña la persona encargada de la modificación.
- Fecha: esta debe ser la fecha actual, es decir cuando se solicita el cambio y cuando finaliza el cambio.
- Categoría de cambio: implica el ámbito en el cual se encuentra inmersa la modificación dentro del proyecto.
- Descripción del cambio: se detalla en que consiste la modificación a realizar.

A más de la plantilla de control de cambios especificada anteriormente, el gerente debe llevar un control más detallado, el cual se basa en la elaboración de un documento el cual queda como anexo, por cada cambio realizado. Este documento debe cumplir con las siguientes secciones.

1. **Datos Generales de la solicitud:**

   <<Nro:>>

   <<Nombre del solicitante:>>

   <<Nombre del encargado:>>

   <<Área del encargado:>>

   <<Fecha:>>

2. **Categoría de cambio:** especifica en qué contexto la solicitud puede afectar a la gestión del proyecto. Estos pueden ser, <<Alcance, Cronograma, Costos, Calidad, Recursos, Procedimientos, Documentación, Otros>>.

3. **Causa/ Origen del cambio:** se indica la razón por la cual se solicita el cambio dentro del proyecto. Estos pueden ser, <<Solicitud del cliente, Reparación de defecto, Acción correctiva, Acción preventiva, Actualización o modificación de documento, Otros>>.

4. **Descripción de la propuesta:** debe contemplar que se propone hacer, porque razón se efectúa la propuesta, y cómo se pretende cumplir.

5. **Justificación de la propuesta de cambio:**

6. **Impacto del cambio:** se debe indicar el impacto en función del cronograma, es decir las etapas a las que afecta, en que tiempo o actividad se presenta el inconveniente, etc.

7. **Implicaciones de recursos:** se especifica que recursos ya sean estos recursos materiales o recursos humanos, se emplean para lograr resolver el cambio.

8. **Implicaciones para los interesados:** las actividades en las que se debe involucrar el cliente.

9. **Riesgos:** que riesgos se pueden llegar a contemplar al elaborar la propuesta.

10. **Comentarios.**

11. **Aprobación:** La fecha y persona que aprueba el cambio.
12. **Firmas del comité de cambios.**
13. **Anexos:** en esta sección se agregan las modificaciones respectivas, por ejemplo, si se presentó una modificación en el diseño de las interfaces de usuario se debe indicar las interfaces iniciales, y la propuesta de las iniciales al finalizar el proceso de cambio.

7.5. Pruebas de integración

7.5.1. Finalidad de las pruebas de integración

Las pruebas de integración se centran en probar cómo se da la interacción entre dos o más unidades de un software, es decir estas pruebas verifican que los componentes de una página, como es este el caso, funcionen correctamente en conjunto.

Las pruebas de integración son realizadas por el equipo de trabajo, y consisten en la evaluación de la aplicación por componentes, elementos y la manera en que estos se interrelacionan entre sí. De esta manera se logra conocer si el proyecto se encuentra correctamente unificado, en función de los elementos, componentes y módulos.

7.5.2. Tipos de revisiones posibles

Para las pruebas de integración se suelen desarrollar evaluaciones a los componentes de la aplicación, estas suelen ser evaluaciones a los enlaces de páginas y enlaces externos; la revisión de los componentes de las interfaces de usuario; y finalmente la revisión de mantenimiento, la cual consiste en una identificación de si es factible realizar modificaciones y actualizaciones futuras a la aplicación.

7.5.2.1. Revisión de enlaces

Cuando se habla de enlaces, se hace referencia a las relaciones entre las páginas web o interfaces, y los enlaces que la aplicación posee en cuanto a otras páginas web o componentes externos de la aplicación usualmente alojados en internet.

Enlaces internos: se consideran dentro de esta clasificación las vinculaciones entre cada una de las páginas propias de la aplicación, es decir, que la navegación entre las interfaces se realice sin ningún problema y sin retardo. Mediante esta evaluación se logra conocer si el modelo navegacional diseñado y programado funciona correctamente.

Enlaces externos: se consideran dentro de esta clasificación las vinculaciones con fuentes externas a la página web, es decir, documentos de internet, otras páginas web, otras aplicaciones, blogs, etc. Las cuales no se han codificado ni diseñado dentro del presente proyecto, pero si son de utilidad para brindar un mejor funcionamiento y cumplir a cabalidad con los requerimientos que ha solicitado el usuario.

Por ejemplo: *Algunos ejemplos de enlaces externos considerados comúnmente en una aplicación web, son:*

Páginas oficiales.

Enlaces para descarga de archivos, o documentos que se encuentran en línea.

Re direccionamiento a blogs, u otras páginas de internet para brindar mayor información o información relacionada.

7.5.2.2. Revisión de componentes

Se considera componente a los módulos, elementos de interfaz y funciones dentro de las líneas de código fuente. La revisión de estos componentes del software brinda el conocimiento del funcionamiento, adecuado o inadecuado, de la aplicación web y ayuda a detectar las posibles fallas para la posterior corrección de los mismos.

Esta revisión a diferencia de la prueba de funcionalidades brinda mayor certeza del componente en el cual se encuentra la falla, y se pueda modificar de manera que se ahorra tiempo.

7.5.2.3. Revisión de mantenimiento

Estas pruebas tienen la finalidad de identificar si la estructura o codificación de la aplicación se encuentra válida para futuros cambios o mantenimientos, de manera rápida y sin uso exagerado de recurso humano. Para la ejecución de esta evaluación se considera la revisión de la documentación, manual de usuario y programador, generadas durante la etapa de implementación.

7.6. Detección y corrección de errores

Para esta prueba se debe hacer uso de herramientas externas como ***Validator*** de W3C. Los usos de estas herramientas permiten de manera sencilla la detección y corrección de errores dentro del proyecto. Esta actividad es fundamental antes de presentar el prototipo de la aplicación ante el cliente.

Una vez detectado los errores dentro de la aplicación, e inclusive dentro de las líneas de código, se debe proceder a corregir cada uno de ellos, y volver a realizar la evaluación con la misma herramienta, y verificar que no se presenten errores.

Es recomendable que se haga uso de dos herramientas de validación como mínimo, y no se realice de manera manual, debido a que la revisión es interna al programa y se pueden omitir algunos errores por desconocimiento de los mismos.

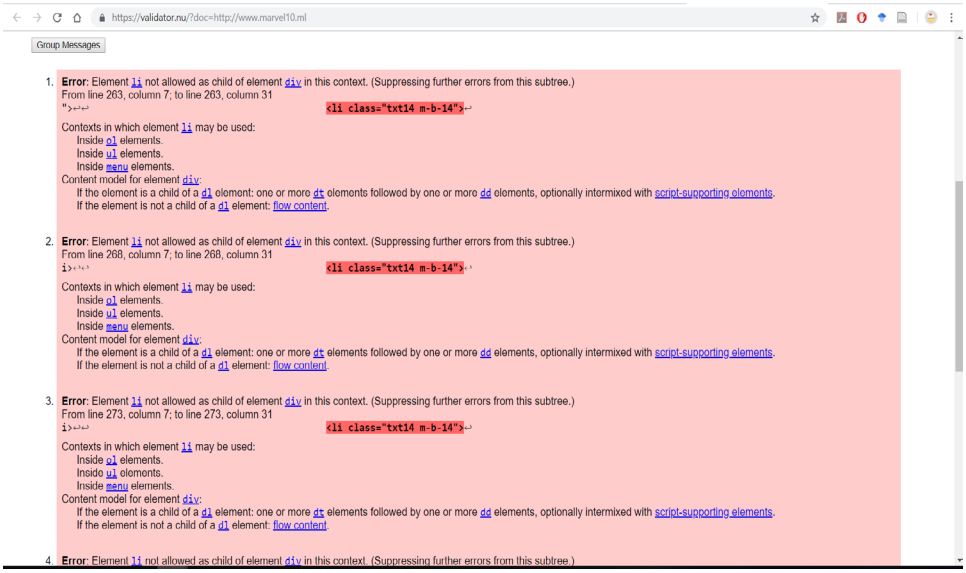

Ilustración 25. Revisión de errores mediante la herramienta W3C. **Fuente:** elaboración propia.

7.7. Pruebas del sistema

Estas pruebas deben ser realizadas específicamente por el equipo externo y diferente al equipo de desarrollo de la aplicación. Debido a que puede interferir las relaciones personales, o consideraciones subjetivas al momento de evaluar cada uno de los componentes de la aplicación web. Para lograr evaluar de manera acertada se realizan tres tipos de pruebas, la primera es la evaluación de calidad, verificando que los indicadores asignados sean cumplidos; el segundo es la evaluación de tendencia de diseño, de manera que la aplicación se logre ejecutar y cumpla con la tendencia que se especificó con anterioridad; y finalmente las pruebas de evaluación con herramientas SEO, para ello se usan herramientas externas.

7.7.1. Pruebas de calidad

La prueba de calidad se realiza mediante el uso de los indicadores que se especifican acorde a la métrica ISO/IEC 9126. Para facilitar la evaluación de estos indicadores se emplea la escala de Likert, considerando a 1 como Pésimo, y 5 como Excelente.

A través de esta evaluación se permite conocer si la funcionalidad y comportamiento del sistema ante los usuarios finales es la más adecuada para el ámbito en el que se va a emplear.

Tabla 48. Indicadores para evaluar la calidad.

Criterios de evaluación	
Valorización	**Interpretación del resultado**
5	Excelente
4	Bueno
3	Regular
2	Malo
1	Pésimo

Fuente: elaboración propia.

La plantilla empleada para evaluar correctamente el sistema acorde a la calidad según los criterios de evaluación. Se evalúa calidad externa, calidad interna y calidad de uso.

Tabla 49. Plantilla de evaluación de calidad.

Evaluación de la métrica							
Característica	***Sub características***	***Criterio***	***1***	***2***	***3***	***4***	***5***

Fuente: elaboración propia.

7.7.2. Pruebas de tendencia

Al igual que la evaluación de la calidad, es necesario que se realice una evaluación de la tendencia empleada, acorde a las características consideradas en las etapas anteriores. La evaluación se realiza en función de criterios de la escala de Likert y su valoración se realiza mediante el uso de porcentaje.

La plantilla empleada para evaluar correctamente el sistema acorde a la calidad según los criterios de evaluación. Se evalúa la tendencia en cada módulo del sistema.

Tabla 50. Plantilla de evaluación de tendencia.

Tendencia <<Nombre de tendencia>>				
Característica	***Bajo***	***Medio***	***Alto***	***%***
TOTAL				

Fuente: elaboración propia.

7.7.3. Pruebas de evaluación con herramienta SEO

El uso de las herramientas SEO son útiles para el conocimiento y análisis de la competencia existente sobre la herramienta web, a más de aportar información requerida para conocer internamente y de manera porcentual el rendimiento y correcto funcionamiento de la aplicación web desarrollada.

Para la evaluación de la aplicación en función de indicadores SEO, se debe emplear herramientas de evaluación web como, Open Site Explorer, Semruch, Google Webmaster Tools, Website Grader, Seosite Checkup, entre otras. Para elegir la mejor herramienta se debe conocer los indicadores que cada una ofrece y cuáles son los que se desean conocer, además de tener en cuenta cuales son de paga y cuales son gratuitas.

La plantilla empleada para evaluar correctamente el sistema acorde a la calidad según los criterios de evaluación. Es recomendable realizar la evaluación en 2 o más herramientas diferentes.

Tabla 51. Plantilla de evaluación de herramientas SEO.

Característica	**Herramienta 1**	**Herramienta 2**	**Herramienta 3**

Fuente: elaboración propia.

CAPÍTULO VIII: FASE DE LANZAMIENTO

La fase de lanzamiento es una de las características peculiares que posee una metodología de desarrollo web, y consiste en la publicación de la aplicación en el internet, de manera que se logre acceder al mismo desde una dirección web, o http. SWIRL enfoca esta fase desde el alojamiento hasta la publicación de la aplicación en la web, considerando adicionalmente y de manera opcional la campaña de marketing y SEO, lo cual queda a criterio del cliente.

8.1. Objetivos

Esta fase posee dos objetivos principales, el primero consiste en la publicación de la aplicación para permitir el acceso a la misma de manera online, evitando el uso de host local. Por otro lado, el segundo objetivo que persigue esta fase es la clausura del proyecto realizando la entrega correspondiente de los entregables finales al cliente, de manera que se dé por concluido el proyecto.

8.2. Actividades en la fase de lanzamiento

Las actividades dentro de esta fase son relativamente vinculadas al proceso de publicación de la aplicación en la web, para ello se debe considerar varios componentes como el host en el cual se alojará la aplicación web, el dominio que se desea asignar y otras características propias de una página web. Con base a lo anterior, en esta fase se presentan las siguientes actividades.

- Selección y preparación del dominio.
- Selección y alojamiento de la aplicación en un hosting.
- Configuración del certificado SSL.
- Campaña de marketing/ SEO (Opcional).
- Presentación de entregables.

8.3. ¿Quiénes intervienen?

Dentro de esta etapa intervienen simplemente dos miembros del equipo de trabajo, el gerente, que es el encargado de comunicarse con el cliente y de la presentación de los entregables respectivos; y el analista web que es el encargado de seleccionar el host, dominio y certificado SSL. Ambos cumplen un rol importante dentro de esta

etapa, y aunque no se documenta en muchos proyectos de desarrollo, es necesaria que se realice y estudie de manera detallada. El analista a más de identificar y alojar la aplicación en la web es el encargado de realizar el proceso de marketing SEO en caso de que el cliente desee contemplar esto dentro del proyecto.

8.4. Elementos de la fase

Los elementos primordiales que esta fase contempla son las actividades designadas para la publicación de la aplicación en internet, y las actividades de entrega y cierre del proyecto desarrollado.

8.4.1. Preparación del dominio

El dominio es un punto de gran importancia dentro de una aplicación web, el registro del mismo debe ser adecuado y acorde a la aplicación, además de realizarlo en una compañía que brinde seguridad y garantía de los servicios brindados. Un dominio debe presentar las siguientes características.

- Debe ser corto para que el usuario logre aprenderlo con facilidad.
- Debe estar vinculado con la temática de la aplicación desarrollada, de manera que sea más fácil de encontrar en las búsquedas.
- No debe estar repetido, ni tener similitud a otro dominio existente.
- El proveedor debe ser de confianza y asegurar un buen servicio del dominio.

8.4.2. Selección y alojamiento en el hosting

El hosting donde se almacenará la aplicación web debe cumplir ciertos requerimientos, los cuales debe estar detallada por el gerente del proyecto. Existen hosts de paga y gratuitos, y cada uno presenta sus restricciones, ventajas y características peculiares. Las consideraciones que se deben tener presente al momento de seleccionar el host son:

- La capacidad de alojamiento del host.
- La velocidad de respuesta ante las peticiones.
- Restricciones de uso u horarios de suspensión.
- Las vinculaciones con el lenguaje de programación y la base de datos en la que se alojan los datos.

- El pago generado por su uso.

Una vez que el analista web genera una lista con las ventajas y desventajas de cada host consultado, se debe reunir con el gerente y plantear la propuesta, mediante la cual se llegara a una solución y se seleccionara el host propicio para el proyecto desarrollado.

Una vez que se tenga seleccionado el host es necesario que se realice el procedimiento correspondiente para su alojamiento y publicación. Y al concluir se debe verificar que todo funcione de manera adecuada, y que no exista falla de comunicación con la aplicación.

Ilustración 26. Proceso de alojamiento de la aplicación. **Fuente:** elaboración propia.

8.4.3. Configuración del protocolo SSL

El protocolo SSL (Secure Sockets Layer) se emplea con la finalidad de mantener la comunicación con cualquier componente de internet seguro, mediante el uso de algoritmos de criptografía para que la conexión sea segura. Al agregar un certificado SSL la dirección de la aplicación web publicada se modifica de http// a https//, y agregando una confiabilidad y seguridad en la transición de los datos, de manera que ya no se envían en texto plano a la base de datos.

SWIRL propone esta configuración necesaria, para la asignación de un certificado se necesita un proveedor, el costo depende de donde se adquiera el servicio. Para mayor seguridad de la información que transcurre dentro de la aplicación se debe asegurar el dominio de la página.

8.4.4. Campaña de marketing/ SEO (Opcional)

La campaña de marketing SEO es opcional, esto dependerá del cliente y del uso de la aplicación en el ámbito de negocios. Esta actividad consiste únicamente en la aplicación de propaganda o asignación de un lugar en los buscadores de la aplicación web. Es decir, al realizar una búsqueda en un navegador, un usuario pueda encontrar la aplicación entre los primeros resultados, de manera que sea visible a

simple búsqueda y además fácil y rápida de acceder. Esto brinda a la empresa mayor competitividad dentro el mercado ante otras compañías que ofrecen los mismos servicios.

REFERENCIAS BIBLIOGRÁFICAS

Acuña, A. P. (2012). *La gestión de los stakeholders: análisis de los diferentes modelos*. Recuperado de: http://repositoriodigital.uns.edu.ar/handle/123456789/4441

Alarcón, V. F. (2006). *Desarrollo de sistemas de información: Una metodología basada en el modelado*. Universitat Politècnica de Catalunya.

Amaolo, M. P. (2007). Lenguajes de Modelado de Reglas de Negocio y la Web Semántica. *IX Workshop de Investigadores en Ciencias de la Computación*, 5.

Arévalo, W., y Atehortúa, A. (2012). Metodología de Software MSF en Pequeñas Empresas. *Cuaderno Activa*, (4), 83-90.

Arias, M. (2005). La ingeniería de requerimientos y su importancia en el desarrollo de proyectos de software. *InterSedes*, *6*(10). Recuperado de: https://www.researchgate.net/publication/237038693_La_ingenieria_de_requerimientos_y_su_importancia_en_el_desarrollo_de_proyectos_de_software

Ariste, C., Ponisio, J., Nahuel, L., y Giandini, R. (2015). Diseñando Transformaciones de Modelos CIM / PIM: desde un enfoque de negocio hacia un enfoque de sistema. *Simposio Argentino de Ingeniería de Software (ASSE 2015) - JAIIO 44*, 10.

Arteaga, C., Barrera, F., y Chaparro, J. (2013). Factores claves en la gestión de proyectos. *Tecnología Investigación Y Academia, 1*(2). Recuperado de: https://revistas.udistrital.edu.co/ojs/index.php/tia/article/view/4935

Báez, M. G., y Barba, S. I. (2000). Metodología DoRCU para la Ingeniería de Requerimientos, *WER*.

Baum, G., Daniele, M., Martellotto, P., y Novaira, M. (2004). *Un Modelo Genérico para el Modelo de Negocio*. Recuperado de: http://hdl.handle.net/10915/22361

Brhel, M., Meth, H., Maedche, A., y Werder, K. (2015). Exploring principles of user-centered agile software development: A literature review. *Information and Software Technology*, *61*, 163-181. doi: https://doi.org/10.1016/j.infsof.2015.01.004

Burneo-Valarezo, S., Delgado Victore, R., y Vérez, M. A. (2016). Estudio de factibilidad en el sistema de dirección por proyectos de inversión. *Ingeniería Industrial*, *37*(3), 305-312.

Cohn, M. (s. f.). Better User Stories - Discover and Deliver What Customers Really Want - Video Course by Mike Cohn. Recuperado el 10 de junio de 2019, de: https://learn.mountaingoatsoftware.com/better-user-stories/

Del Carpio Gallegos, J. (2006). Análisis del riesgo en la administración de proyectos de tecnología de información. *Industrial Data*, *9*(1). Recuperado de: http://www.redalyc.org/resumen.oa?id=81690113

Dillibabu, R., y Krishnaiah, K. (2005). Cost estimation of a software product using COCOMO II.2000 model – a case study. *International Journal of Project Management*, *23*(4), 297-307. doi: https://doi.org/10.1016/j.ijproman.2004.11.003

Domínguez-Mayo, F. J., Escalona, M. J., Mejías, M., Ross, M., y Staples, G. (2012). Quality evaluation for Model-Driven Web Engineering methodologies. *Information and Software Technology*, *54*(11), 1265-1282. doi: https://doi.org/10.1016/j.infsof.2012.06.007

Estrada, H., Martínez, A., Pastor, O., y Sánchez, J. (2002). Generación de Especificaciones de Requisitos de Software a partir de Modelos de Negocios: un enfoque basado en metas. *V Workshop de Engenharia de Requisitos*, *2*(11), 17.

Ferraro, M. A., Medina, Y., Dapozo, G., y Estayo, M. (2016). Especificación y trazabilidad de requerimientos en el desarrollo de aplicaciones web. *II Jornadas de investigacion en ingenieria del NEA y paises limítrofes*. Recuperado de: http://repositorio.unne.edu.ar/handle/123456789/1617

Ferré Grau, X., y Sánchez, M. I. (s. f.). *Desarrollo Orientado a Objetos con UML*.

Giraldo, G. L., Acevedo, J. F., y Moreno, D. A. (2011). Una ontología para la representación de conceptos de diseño de software. *Revista Avances en Sistemas e Informática*, *8*(3), 103-110.

Grigera, J., Rivero, J. M., Robles, E., Giacosa, F., y Rossi, G. (2012). From Requirements to Web Applications in an Agile Model-Driven Approach. In: *Brambilla M., Tokuda T., Tolksdorf R. (eds) Web Engineering. ICWE 2012. Lecture Notes in Computer Science, 7387*. Springer, Berlin, Heidelberg. doi: https://doi.org/10.1007/978-3-642-31753-8_15

Hernández, H. (s. f.). *Diagramas del UML*.

Huidobro, J., Heredia, B., Salmona, M., y Alvarado, L. (2009). Inclusión en la gestión de riesgos en el estudio de ofertas para licitaciones de proyectos de construcción. *Revista de la Construcción*, *8*(2). Recuperado de: http://www.redalyc.org/resumen.oa?id=127619798003

Kazemifard, M., Zaeri, A., Ghasem-Aghaee, N., Nematbakhsh, M. A., y Mardukhi, F. (2011). Fuzzy Emotional COCOMO II Software Cost Estimation (FECSCE) using Multi-Agent Systems. *Applied Soft Computing*, *11*(2), 2260-2270. doi: https://doi.org/10.1016/j.asoc.2010.08.006

Kendall, K. E., y Kendall, J. E. (2005). *Análisis y diseño de sistemas*. Recuperado de: https://books.google.com.ec/books?hl=es&lr=&id=5-rZA0FggusC&oi=fnd&pg=PA1&dq=an%C3%A1lisis+de+requerimientos&ots=1E6HEFw4B4&sig=IKx-oLlHkusE2LAthpb_BM5q3-s#v=onepage&q=an%C3%A1lisis%20de%20requerimientos&f=false

Leonardi, M. C., y Mauco, V. (2004). Uso de modelos de negocios y de requisitos en desarrollos basados en MDA. *VI Workshop de Investigadores en Ciencias de la Computación*, 19-24. Recuperado de: http://sedici.unlp.edu.ar/handle/10915/21216

Lledó, P. (2013). *El ABC para un director de proyectos exitoso_Pablo Lledó.pdf - 95,4 Sexta Edición Incluye tips de Project y Excel 1 Administración de Proyectos Datos* (3.ª ed.). Recuperado de: https://www.coursehero.com/file/40641274/El-ABC-para-un-director-de-proyectos-exitoso-Pablo-Lled%C3%B3pdf/

Lledó, P., y Rivarola, G. (2007). *Gestion de Proyectos* (1.ª ed.). Recuperado de: https://vdocuments.mx/gestion-de-proyectos-pablo-lledo-y-gustavo-rivarola.html

Martínez, J . G. (2007). *Introducción al análisis de riesgos*. Editorial Limusa.

Melendez, F. (2013). *PMBOK* (5.ª ed.). Recuperado de: https://www.academia.edu/7458781/PMBOK_5ta_Edicion_Felipe_Melendez

Molina Ríos, J. R., Honores Tapia, J. A., y Zea Ordóñez, M. P. (2015). *Nociones de Ingeniería de Software.* Recuperado de: http://repositorio.utmachala.edu.ec/handle/48000/6919

Molina Ríos, J. R., Zea Ordóñez, M. P., Contento Segarra, M. J., y García Zerda, F. G. (2018). Comparación de metodologías en aplicaciones web. *3C Tecnología: glosas de innovación aplicadas a la pyme*, *7*(1), 1-19.

Molina Ríos, J. R., Zea Ordóñez, M. P., Redrován Castillo, F. F., Loja Mora, N. M., Valarezo Pardo, M. R., y Honores Tapia, J. A. (2018). *SNAIL, Una metodología híbrida para el desarrollo de aplicaciones web*. 3Ciencias. Recuperado de: https://www.3ciencias.com/libros/libro/snail-una-metodologia-hibrida-para-el-desarrollo-de-aplicaciones-web/

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

Padilla, M. C. (2011). *Formulación y evaluación de proyectos*. Recuperado de: https://books.google.com.ec/books?id=1drDDQAAQBAJ

Penadés, M. C., y Letelier, P. O. (2006). Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP). *Técnica administrativa*, *5*(26).

Peña Abreu, M., Rodríguez Rodríguez, C. R., y Piñero Pérez, P. Y. (2017). Computación con palabras para el análisis de factibilidad de proyectos de software. *Tecnura*, 20(50), 69-84. Recuperado de: https://revistas.udistrital.edu.co/index.php/Tecnura/article/view/11562

Pérez, O. A. (2011). Cuatro enfoques metodológicos para el desarrollo de Software RUP – MSF – XP - SCRUM. *INVENTUM, 6*(10), 64-78. doi: https://doi.org/10.26620/uniminuto.inventum.6.10.2011.64-78

Pinciroli, F., y Zeligueta, L. (2017). *Modelado de Negocios Orientado a Aspectos con AOP4ST*. Repositorio institucional de la UNLP, 591-595. Recuperado de: http://hdl.handle.net/10915/62104

Ponce, H. (2005). *La matriz FODA: una alternativa para realizar diagnósticos y determinar estrategias de intervención en las organizaciones productivas y sociales*. Recuperado de: https://www.researchgate.net/publication/5016062_La_matriz_FODA_una_alternativa_para_realizar_diagnosticos_y_determinar_estrategias_de_intervencion_en_las_organizaciones_productivas_y_sociales

Rodríguez, J. R. (2011). *Gestión de proyectos informáticos: métodos, herramientas y casos*. Editorial UOC.

Romano, G., y Yacuzzi, E. (2011). *Elementos de la gestión de proyectos.* Recuperado de: https://www.econstor.eu/handle/10419/84368

Salazar Morales, T., y Rivero Ceballos, J. L. (2013). Debilidades, amenazas, fortalezas y oportunidades en el INCES penitenciario región Los Andes venezolanos 2011. *Visión gerencial,* 12(2).

Salgado, C., Peralta, M., Baigorria, L., Berón, M., Riesco, D., y Montejano, G. (2011). Modelado de Procesos de Negocio: Evaluación y Comparación de Modelos y Lenguajes de Modelado. *XIII Workshop de Investigadores en Ciencias de la Computación*, 4.

Satpathy, T. (2013). *A guide to the Scrum Body of knowledge (SBOK Guide)*. Phoenix, Arizona: SCRUMstudy, A brand of VMEdu, Inc.

Silva, D. A., y Mercerat, B. (2001). Construyendo aplicaciones web con una metodología de diseño orientada a objetos. *Revista Colombiana de Computación*, *2*(2), 1-21. Recuperado de: https://pdfs.semanticscholar.org/a6e6/e239c41ef454258d3ed615efa3343892ab76.pdf

Ingeniería y Tecnología

3ciencias