“MMS” METODOLOGÍA PARA EL DISEÑO Y DESARROLLO DE APLICACIONES MÓVILES

Jimmy Rolando Molina Ríos
Mariuxi Paola Zea Ordóñez
Fausto Fabián Redrován Castillo
Milton Rafael Valarezo Pardo
Joofre Antonio Honores Tapia
Rodrigo Fernando Morocho Román
Jorge Luis Armijos Carrión
Oscar Efrén Cárdenas Villavicencio
Bárbara Brigitte Romero Macharé

Ingeniería y Tecnología

3ciencias

“MMS”
METODOLOGÍA PARA EL DISEÑO Y DESARROLLO DE APLICACIONES MÓVILES

Jimmy Rolando Molina Ríos
Mariuxi Paola Zea Ordóñez
Fausto Fabián Redrován Castillo
Milton Rafael Valarezo Pardo
Joofre Antonio Honores Tapia
Rodrigo Fernando Morocho Román
Jorge Luis Armijos Carrión
Oscar Efrén Cárdenas Villavicencio
Bárbara Brigitte Romero Macharé

Editorial Área de Innovación y Desarrollo,S.L.

ÁREA DE INNOVACIÓN Y DESARROLLO, S.L.

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

Primera edición: **mayo 2021**

ISBN: **978-84-123661-9-8**

DOI: https://doi.org/10.17993/IngyTec.2021.77

GRUPO DE INVESTIGACIÓN DE INGENIERÍA DE SISTEMAS – G.I.I.S.

Proyecto titulado:

Construcción de un modelo referencial híbrido para la gestión de procesos de desarrollo de software móvil.

Fecha aprobado el Proyecto por Consejo Universitario de la Universidad Técnica de Machala- 19 de agosto de 2020; Resolución Res. 359/2020

Dando cumplimiento al Objetivo: Diseño del modelo referencial hibrido para la gestión de procesos de desarrollo ágil de software móvil.

Ing. Jimmy Molina Ríos, Mg.

Ing. Mariuxi Zea Ordóñez, Mg.

Ing. Joofre Honores Tapia, Mg.

Ing. Milton Valarezo Pardo, Mg.

Ing. Fausto Redrován Castillo, Mg.

Ing. Rodrigo Morocho Román, Mg.

Ing. Oscar Cárdenas Villavicencio, Mg.

Ing. Jorge Armijos Carrión, Mg.

Miembros del Proyecto

Srta. Barbará Romero Macharé

Sr. Henry Pardo León

Sr. Donis Torres Apolinarioo

Sr. Ricardo Ramón Ramón

Miembros en Formación

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE ILUSTRACIONES

CAPÍTULO I: INTRODUCCIÓN A MMS

1.1. Resumen

El presente libro describe cada una de las fases y actividades inmersas en el proceso de desarrollo de la metodología propuesta para el desarrollo móvil, MMS (Modelo Mobile Sprint). La metodología surgió de la necesidad de incluir procesos del desarrollo ágil relevantes y necesarios para cubrir las necesidades primordiales del proceso de desarrollo en aplicaciones móviles. El proceso basado en enfoque ágil beneficia tanto el proceso de desarrollo como al aplicativo móvil ya desarrollado, dentro del proceso de desarrollo brinda mayor flexibilidad y rapidez en cuanto a las modificaciones que requiera el proyecto, mediante la inclusión constante del usuario en las actividades requeridas; por otro lado, beneficia la aplicación móvil al asegurar la calidad, usabilidad y demás requerimientos especiales que posee el software móvil. La metodología propuesta incluye las necesidades comúnmente presentadas en las aplicaciones móviles mediante las actividades y artefactos empelados en las cinco fases de desarrollo.

1.2. Introducción

El surgimiento de los teléfonos inteligentes abrió paso a una nueva era para el software, las distintas actividades que se realizaban por medio de una computadora ahora pueden ser ejecutadas en pequeños dispositivos portátiles, es así como surgen las populares aplicaciones móviles.

En la actualidad, son muchos los negocios y empresas que están adoptando este tipo de software para la automatización de sus procesos debido a la portabilidad que ofrecen, permitiéndoles conocer el estado de sus actividades en cualquier lugar y al alcance de la mano.

Sin embargo, han surgido varias interrogantes acerca de la calidad que ofrecen estas aplicaciones, y es precisamente esto lo que ha generado un nuevo reto para la ingeniería de software, debido que al tratarse de productos que son utilizados en dispositivos con diferentes características, provocan diversos escenarios y con ello diversas experiencias de usuario.

Una empresa desea que el funcionamiento de su aplicación móvil sea óptimo y fácil de utilizar, la misma que puede ser para uso interno o para la prestación de servicios a sus clientes. Sin embargo, en muchos de los casos existen quejas por parte de

los usuarios, quienes mencionan ciertos problemas con el uso de las mismas, esto suele suceder debido a que algunas empresas de software no hacen uso de una metodología adecuada para su desarrollo.

Las metodologías de desarrollo de software son un conjunto de procedimientos o instrucciones agrupadas por fases que permiten elaborar un software de calidad. Han pasado varios años desde la aparición de las primeras metodologías, y con el pasar del tiempo estas han evolucionado para dar solución a nuevos problemas, en primera instancia se encontraban las metodologías tradicionales que luego fueron reemplazadas por las ágiles y estas a su vez por las híbridas (tradicional y ágil), todas ellas se han enfocado al software de escritorio y web, sin embargo también han sido adoptadas para el desarrollo de aplicaciones móviles a pesar de no abordar las problemáticas inherentes a este tipo de software. Aquí es donde surgen los inconvenientes con respecto a la calidad de las aplicaciones móviles, debido a que los dispositivos donde serán ejecutadas poseen diversas prestaciones de hardware y software lo que dificulta definir un modelo estándar, Maia *et al.* (2019) menciona en su investigación que es necesario considerar factores como el rendimiento, eficiencia, compatibilidad, y usabilidad al momento de desarrollar una aplicación móvil, adicionalmente a los parámetros que establecen las metodologías y estándares utilizados.

Esta es la razón principal por la cual se ha elaborado el Modelo Mobile Sprint, el cual toma como base algunas características de la metodología SCRUM como son los Sprint, para combinarlos con una serie de métricas identificadas por diversos autores que contribuyen al desarrollo de aplicaciones móviles de calidad, adicionalmente aborda temas relacionados intrínsecamente al diseño y despliegue de este tipo de software.

1.3. Necesidades específicas en aplicaciones móviles

La calidad en las aplicaciones móviles a diferencia del software web posee una estrecha relación con las características propias de los terminales en los cuales serán ejecutadas, originando un verdadero desafío para las organizaciones que se dedican al área de desarrollo de software.

Diversos autores coinciden que aspectos como el consumo de energía y otros recursos de los teléfonos móviles son factores claves para una buena valoración de las aplicaciones por parte de los usuarios, así como el aprovechar al máximo las

pantallas de estos dispositivos, ya que en su mayoría no sobrepasan las 7 pulgadas, razón por la cual la información a presentarse debe estar estratégicamente organizada garantizando así la usabilidad de la aplicación.

El consumo de datos o acceso a internet por parte de estas aplicaciones debe ser correctamente controlado. Por tal motivo se aconseja controlar los procesos en segundo plano que hacen uso de estos servicios de esta manera también se optimizaría el consumo de otros recursos como almacenamiento y procesamiento.

A pesar de ser varios los parámetros que los usuarios consideran para determinar si las aplicaciones que utilizan son de calidad, poco o nada se ha considerado estas características dentro de las metodologías presentes en el mercado, es por esta razón que el Modelo Mobile Sprint integra estas métricas junto con las buenas prácticas de las metodologías ágiles para brindar una guía al desarrollo de aplicaciones móviles de calidad.

1.4. MMS, Modelo Mobile Sprint

MMS por sus siglas Modelo Móvil Sprint, es una metodología híbrida creada específicamente para el desarrollo de aplicaciones móviles. Caracterizada principalmente por un enfoque ágil dentro de cada una de sus etapas. La metodología considera en sus fases características y aspectos relevantes de otras metodologías como son la Mobile-D, KANBAN, y principalmente la SCRUM, ya que emplea la idea de Sprints y revisiones programadas incluyendo las necesidades que poseen las aplicaciones desarrolladas para dispositivos móviles.

1.4.1. Objetivos

- Fortalecer el uso de sprints como modelo de éxito para el desarrollo de aplicaciones móviles.
- Brindar conjunto de buenas prácticas que integren las características inherentes de los dispositivos móviles durante el desarrollo de aplicaciones.
- Integrar opiniones de los usuarios finales para mejorar la usabilidad de las aplicaciones.

1.4.2. Características

- Modelo basado en sprints que permite la obtención temprana de

resultados.

- Facilita el desarrollo de aplicaciones simples o complejas
- Incorpora métricas específicas para el uso de aplicaciones en dispositivos móviles.
- Promueve el uso de prototipos para mejorar la usabilidad de la aplicación.
- Permite el fácil desarrollo de la aplicación por medio de módulos casi independientes.
- Facilita la gestión de las distintas dimensiones de un proyecto.

1.5. ¿Quiénes intervienen?

En esta primera fase intervienen únicamente dos actores, como son el gerente del proyecto y el cliente. La responsabilidad total de esta fase recae sobre el gerente pues es el encargado de gestionar correctamente los recursos que posee la empresa tales como recursos materiales, tecnológicos y humanos, con la finalidad de lograr los objetivos planteados y satisfacer las expectativas del cliente.

Este último también tiene un papel importante debido a que si se quiere tener una aplicación móvil de calidad se deben incluir los cambios que el cliente considere necesarios en cualquier etapa y así obtener una concordancia entre las partes interesadas. Cabe recalcar que a pesar de que en esta etapa no intervienen todo el equipo de desarrollo, el gerente se encarga de comunicar las novedades del mismo.

1.5.1. Arquitectura del modelo

El cliente forma parte importante dentro del desarrollo de este tipo de aplicaciones, es importante conocer sus necesidades a detalle, por esta razón está presente en diversas fases del desarrollo. Dentro de la fase de planificación ayudará a definir la Gestión del Alcance dónde se detallará las funcionalidades que la aplicación abarcará. Dentro de esta misma fase existen otras Gestiones como Tiempo, Interesados, Costos entre otras que serán responsabilidades directas del Director de proyecto, sin embargo, los entregables obtenidos deben ser socializado con el cliente para su aprobación.

Dentro de la fase de Diseño el cliente probará los prototipos diseñados por los desarrolladores con el objetivo de estudiar la usabilidad de la aplicación y mejorar

diversos aspectos que sean necesarios antes de pasar a la fase de codificación.

En las siguientes fases como Codificación y Pruebas el equipo de desarrollo junto con el director mantendrá reuniones constantes (DSM) y en caso de ser necesario el cliente también tendrá que intervenir.

1.5.2. Ciclo de vida

El ciclo de vida del Modelo Mobile Sprint se encuentra conformada por 6 fases como son Planificación, Diseño, Ejecución, Pruebas y Lanzamiento. Este modelo tiene como base la implementación de Sprints o ciclos cortos, los mismos que pueden durar de 2 a 4 semanas, dentro de este tiempo es muy importante mantener una continua comunicación para conocer los avances que se van obteniendo, por ello se incorporan las Daily Sprint Meeting (DSM) donde se exponen cualquier dificultad presentada o sugerencia.

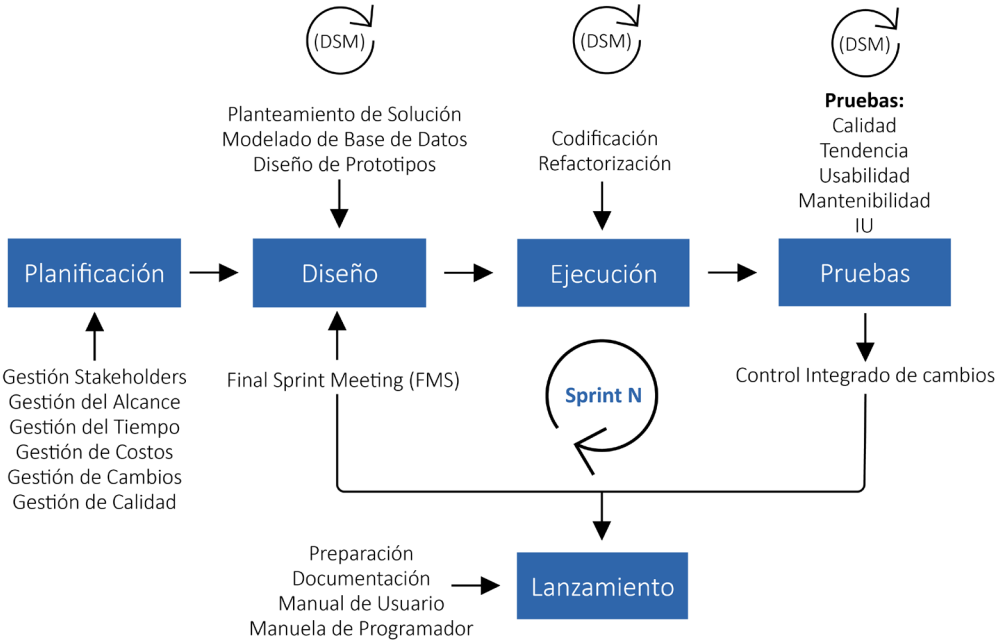


Ilustración 1. Ciclo de vida Modelo Mobile Sprint.
Fuente: elaboración propia.

El modelo propone en primer lugar realizar una correcta planificación de las principales dimensiones del proyecto, de esta manera se conoce el entorno y modelo de negocio que posee la empresa y con ello dar paso al planeamiento de soluciones.

Cuando la planificación esté lista se procede al modelado de la base de datos, así como la elaboración de prototipos que permitirán a todos los interesados tener una noción del entregable o producto final.

La ejecución o codificación es la parte más extensa del proyecto, los desarrolladores se limitan a realizar lo que el planificado para dicho sprint.

Para que una aplicación pueda ser lanzada a los usuarios finales, esta debe pasar por una serie de pruebas que permitan garantizar la calidad de la misma, donde en caso de identificar errores o no alcanzar un determinado requerimiento, se debe llevar un registro de cambios.

Cada una de las fases posee actividades organizadas con el objetivo de obtener un producto funcional como entregable al terminar cada sprint, donde además se obtendrá los cambios necesarios a realizarse lo cuáles serán discutidos en la FMS antes de empezar un nuevo sprint, esto con la finalidad de ofrecer una aplicación de calidad.

CAPÍTULO II: FASE DE PLANIFICACIÓN

En todo proyecto de desarrollo de software la fase de planificación o etapa que se encarga de gestionar eficientemente los recursos y establecer las actividades necesarias para lograrlo, es uno de los pilares fundamentales para el éxito del proyecto, debido a que una correcta planificación refleja menores costos y tiempos. La planificación permite al director del proyecto tener un control preciso de los procesos a efectuarse, identificando y preparándose para eventualidades que todo desarrollo de proyecto conlleva.

2.1. Objetivos

La fase de planificación es el punto de partida del proyecto, por esta razón el objetivo principal es estructurar los procesos a seguir para el desarrollo de una aplicación móvil de calidad, por esta razón se debe abordar objetivos adicionales como:

- Identificar los procesos y actividades de la organización para la cual se desarrollará la aplicación móvil.
- Establecer las partes interesadas y como estarán vinculadas durante el desarrollo del proyecto.
- Definir las actividades a realizar para el desarrollo del proyecto en base al tiempo y recursos existentes en la organización.
- Establecer las métricas y estándares a implementar para asegurar la calidad de la aplicación a desarrollar.

2.2. ¿Quiénes intervienen?

En esta primera fase intervienen únicamente dos actores, como son el gerente del proyecto y el cliente. La responsabilidad total de esta fase recae sobre el gerente pues es el encargado de gestionar correctamente los recursos que posee la empresa tales como recursos materiales, tecnológicos y humanos, con la finalidad de lograr los objetivos planteados y satisfacer las expectativas del cliente.

Este último también tiene un papel importante debido a que si se quiere tener una aplicación móvil de calidad se deben incluir los cambios que el cliente considere necesarios en cualquier etapa y así obtener una concordancia entre las partes interesadas. Cabe recalcar que a pesar de que en esta etapa no intervienen todo el equipo de desarrollo, el gerente se encarga de comunicar las novedades del mismo.

2.3. Actividades en la fase de planificación

La fase de planificación detalla la estructura a seguir para el desarrollo de este proyecto, esta planificación se basa en los recursos disponibles para la empresa tales como tiempo, dinero, y recurso humano, las actividades a desarrollarse en esta fase se enlistan a continuación:

a. Entorno de la empresa.
b. Modelo de Negocio.
c. Análisis de Factibilidad.
d. Resultados esperados.
e. Gestión de los stakeholders.
f. Gestión del alcance.
g. Gestión del cronograma.
h. Gestión de Costos.
i. Gestión de Cambios.
j. Gestión de Calidad.

Cada una de estas actividades generan entregables, los mismo que ayudan a que el equipo de trabajo tenga conocimiento acerca de las decisiones tomadas por el gerente. Entre los entregables a obtener se encuentran los siguientes:

- Acta de constitución.
- Matriz de responsabilidades.
- Estructura de Desglose de Trabajo por sprint (EDTS).
- Cronograma.
- Plan de costos.
- Matriz de seguimiento de cambios.
- Plan de control de calidad.

2.3.1. Entorno de la empresa

Antes de empezar con un proyecto de software, es importante conocer el entorno de la empresa, cuáles son las necesidades que se desea cubrir o procesos a optimizar

ya que esto influye directamente en el desarrollo del proyecto.

Una empresa al plantear su misión y visión da a conocer cuáles son sus aspiraciones a futuro, un paso importante para alcanzarlas es la innovación, por ello es importante conocer cuáles son los procesos o actividades que son efectuadas interna o externamente, y de esta manera poder identificar las deficiencias existes y buscar una solución óptima, esta solución muchas veces va ligada con la automatización de dichos procesos haciendo uso de software, ya sea este web o móvil.

Cabe recalcar que estos sistemas deben estar enfocados en las necesidades u objetivos planteados por las empresas, es decir determinar actividades o servicios específicos que se desean automatizar, un ejemplo de ello es la implementación de aplicaciones móviles para la prestación de servicios a sus clientes. Además, se puede mencionar otros factores que pueden influir en estos proyectos como sus políticas de seguridad, estructura organizacional y razón social.

2.3.1.1. Análisis FODA

"El análisis FODA es una herramienta de planificación estratégica, diseñada para realizar una análisis interno (Fortalezas y Debilidades) y externo (Oportunidades y Amenazas) en la empresa" (Riquelme, 2016) de esta manera identificar los factores que pueden influir en el éxito del desarrollo del proyecto.

Tabla 1. Análisis FODA.

FORTALEZAS	**OPORTUNIDADES**
Son las capacidades especiales o factores internos que la empresa puede utilizar para lograr sus objetivos.	Hace referencia a las cualidades o factores externos que permiten que la empresa alcance sus metas y destaque entre las demás.
DEBILIDADES	**AMENAZAS**
Factores que pueden poner a la empresa en una posición desfavorable y afectar el resultado del proyecto.	Situaciones que provienen del entorno y pueden afectar negativamente la estabilidad de la organización y proyecto.

Fuente: elaboración propia.

2.3.1.2. Objetivos del Proyecto

Los objetivos del proyecto se deben plantear de forma clara, concisa y a la vez alineados con los principios de la empresa, además estos constituyen el camino a seguir y en base a los cuales se pueden planificar diversas actividades a desarrollar. Unos objetivos bien definidos permiten que el equipo del proyecto tenga una meta clara, y puedan organizarse de la mejor manera para alcanzarla.

2.3.2. Modelo de Negocio

Mediante un modelo de negocio se puede ilustrar las actividades que agreguen valor a la organización, cuáles son los actores que interactúan con ellas y de qué manera son efectuadas, de esta manera constatar si estos procesos van acordes a los objetivos de la empresa (Li, Qiao y Wang, 2017).

2.3.2.1. Identificación de procesos

Antes de dar paso a la planificación del proyecto es importante comprender los procesos de negocio de la organización con el objetivo de identificar las actividades, métodos, usuarios y reglas involucradas.

Para empezar con la elaboración del modelo de negocio se debe enlistar los procesos para los cuales se va a desarrollar la aplicación móvil, estos deben ser obtenidos con el mayor detalle posible de lo contrario el software no logrará satisfacer las necesidades existentes.

Si los procesos son demasiado grandes serán divididos en módulos más pequeños respetando una jerarquía que facilite la compresión por parte del grupo de trabajo.

Además, los usuarios serán parte importante para el desarrollo de una aplicación ya que serán quienes interactúen diariamente con el software, por esta razón se debe obtener la lista de usuarios que intervienen en los procesos o actividades y de qué manera lo hacen.

Habitualmente se utilizan diagramas de flujo para represar estos procesos, los cuales muestran la dirección de la información, los roles y las actividades asignadas, así como las relaciones existentes entre cada proceso.

Para complementar estos diagramas se debe especificar cada una de sus actividades,

para esta manera evitar cometer errores durante la fase de desarrollo, se puede utilizar cartillas donde se especifiquen las actividades predecesoras a las cuales se las nombrará origen, también se debe mencionar quien realizará dicha actividad es decir el actor o usuario, así como las condiciones o reglas de negocio que deben ser respetadas antes y después de la actividad, en otras palabras las pre y post condiciones.

A continuación, se presenta una plantilla para la especificación de actividades.

Tabla 2. Plantilla para la especificación de actividades del Modelo de Negocio.

CAMPO	DESCRIPCIÓN
Actividad	Nombre de la actividad a realizarse
Origen	Actividad/es precedentes/s
Actor	Usuario encargado de la actividad
Pre-condición	Reglas o estado previo a la ejecución de la actividad
Post-condición	Reglas o estado posterior a la ejecución de la actividad
Caso de Uso	Detalla el caso de uso que corresponde a la actividad

Fuente: elaboración propia.

2.3.2.2. Segmentación de mercado / usuarios

Muchas empresas consideran desarrollar aplicaciones móviles con el objetivo de ofrecer servicios a sus clientes a través de estos canales digitales, por esta razón deben tener presente que es necesario realizar una segmentación de los mismos, debido a que la experiencia del usuario a utilizar un nuevo software puede ser distinto ya sea por factores sociales, el uso de dispositivos móviles, entre otros.

Esta sección va de la mano con el apartado anterior por esta razón se debe especificar en el modelo de negocio para que tipo de usuarios va dirigido este nuevo proyecto.

2.3.2.3. Reglas del negocio

Las reglas de negocio ayudan a definir o restringir acciones respecto a los procesos de una empresa. Su nivel de aplicación es muy alto y pueden ser utilizadas para toda actividad que se desarrolle de forma interna o externa.

Cuando se habla de actividades que se desarrollan externamente se hace referencia

a los procesos que los clientes pueden ejecutar desde las aplicaciones móviles diseñadas para ellos, entre las cuales se pueden mencionar el nivel de acceso a la información, políticas y términos de aceptación de este tipo de servicios, etc.

2.3.3. Análisis de Factibilidad

Realizar un análisis de factibilidad antes de iniciar un proyecto permite tener una orientación para continuar o abandonar el desarrollo del mismo. El efectuar este análisis permite que los interesados tengan expectativas más cercanas a la realidad con respecto al software a diseñarse.

Además, este tipo de análisis permite identificar cuáles serían las condiciones idóneas para ser llevado a cabo, así como las dificultades que se pueden presentar, de esta manera el gerente del proyecto puede establecer estrategias que permitan alcanzar el éxito. A continuación, se detallan los tipos de factibilidad a considerarse en análisis de factibilidad.

2.3.3.1. Factibilidad Operativa

La factibilidad operativa está relacionada con los recursos internos que posee la empresa para el desarrollo de proyectos, siendo el más importante el recurso humano, debido a que es el que efectuará las actividades en las diferentes fases del proyecto, por esta razón se debe cuestionar si los empleados se encuentran capacitados para el tipo de proyecto asignado.

2.3.3.2. Factibilidad Técnica

La factibilidad técnica evalúa la infraestructura de la empresa para responder eficientemente en el desarrollo del proyecto, esto hace referencia tanto a los conocimientos técnicos, como el hardware y software necesarios.

En este aspecto se puede cuestionar si el hardware o software existente son suficientes para iniciar con el proyecto o si necesario realizar nuevas adquisiciones, si esto último es necesario, debe ser indicado dentro en el análisis de factibilidad.

2.3.3.3. Factibilidad Económica

"Este tipo de factibilidad se ocupa de identificar con la mayor precisión posible el coste que implica el instalar y desarrollar el proyecto de software" (Molina, Honores y Zea, 2015), tales como recurso humanos, tecnológicos y materiales. si el resultado

obtenido refleja que los costes son superiores a los beneficios se debe descartar el desarrollo de dicho proyecto, mientras que, si los resultados son favorables, el riesgo disminuye, aunque esto no significa que dejen de existir.

Tabla 3. Plantilla para la especificación recursos del proyecto.

RECURSOS HUMANOS			
Rol/Profesión	Cantidad	Costo Individual	Costo Total
RECURSOS MATERIALES			
Material	Cantidad	Costo Individual	Costo Total
RECURSOS TECNOLÓGICOS			
HARDWARE			
Material	Cantidad	Costo Individual	Costo Total
SOFTWARE			
Material	Cantidad	Costo Individual	Costo Total

Fuente: elaboración propia.

2.3.4. Resultados esperados

Las empresas al momento de emprender un nuevo proyecto lo realizan con la expectativa de obtener beneficios, ya sean estos económicos o de optimización y reducción de costes en sus procesos. “El análisis de los beneficios del proyecto es un punto clave para conocer la eficiencia, calidad y sobre todo viabilidad del proyecto planteado.” (Molina *et al.*, 2018).

2.3.4.1. Beneficios tangibles e intangibles

Beneficios Tangibles

Cuando se habla de beneficios tangibles se hace referencia aquellos que pueden ser medidos en términos monetarios y por lo general están relacionados con:

- Reducción de recursos.
- Reducción de obsolescencia de valores.
- Mejora en la productividad de procesos y personal.

Beneficios Intangibles

No pueden ser cuantificados fácilmente, sin embargo, pueden tener un gran impacto para la organización, ejemplo de este tipo de beneficios son:

- Mejor toma de decisiones.
- Obtención de ventajas competitivas.
- Mejora en la respuesta de clientes.
- Aumento de la transparencia de la organización.

2.3.5. Gestión de Stakeholders

La gestión de interesados "permite al gerente del proyecto desarrollar diferentes formas de lograr la participación eficaz de los interesados en el proyecto" (PMBOK, s.f.) debido a que los mismos estarán presentes en las distintas fases del desarrollo del proyecto, también permite conocer a los usuarios finales a quienes está dirigida la aplicación teniendo en cuenta la segmentación correspondiente.

La importancia de una correcta gestión de interesados radica en que, si no se conoce bien los stakeholder y sus expectativas reales, se puede obtener productos con alta probabilidad de no ser percibidos como objetos de valor por parte de los usuarios finales (Bolshakova, Guerriero, y Halin, 2020).

Dentro de esta sección es necesario el planteamiento de los roles a desempeñar en relación con las actividades existentes, los mismos que se describen a continuación.

2.3.5.1. Establecimiento de roles

La asignación de roles a los stakeholder permite definir las actividades que cada uno de ellos deben llevar a cabo, por eso es importante identificar todas las personas que intervengan en el proyecto detallando el nivel de implicación, sus expectativas,

necesidades, la relación existente con otros interesados y el nivel de acceso a la información.

A lo largo del tiempo se han propuesto diversos modelos para la identificación de los interesados, de los cuales el Modelo Mobile Sprint (MMS) ha recopilado diversas caracteristicas para integrarlos en la elaboración de su plantilla de matriz de stakeholders. Los roles utilizados en el MMS se definen de forma similar a Scrum:

- Gerente de Proyecto.
- Analista de Requerimientos.
- Equipo de desarrollo.

No es necesario especificar en la matriz general el rol especifico de los miembros del equipo de desarrollo, ya que estos serán descritos en la matriz especifica por fases.

Tabla 4. Plantilla de Matriz general de stakeholders.

MATRIZ GENERAL DE STAKEHOLDERS				
Id	Interesado	Cargo	Rol	Responsabilidades
Identificador único, puede utilizarse letras y números.	Nombres y apellidos.	Titulo o profesión.	Función a desempeñar durante el desarrollo del proyecto.	Descripción general de las actividades o responsabilidades asignadas durante el proyecto.

Fuente: elaboración propia.

Esta matriz general de stakeholder se convierte en uno de los entregables dentro de la fase de planificación la misma que además viene indexada en el acta de constitución del proyecto.

2.3.5.2. Formación de equipos

El gerente del proyecto puede realizar agrupaciones de los interesados de acuerdo a las fases, actividades y responsabilidades asignadas, esto con la finalidad de mejorar el flujo de trabajo en cada una de las etapas del desarrollo del proyecto, por ello es necesario utilizar una matriz específica para este trabajo.

Tabla 5. Plantilla de matriz específica de stakeholders por fases.

MATRIZ ESPECÍFICA DE STAKEHOLDERS				
Fase				
Fecha Inicio				
Fecha Finalización				
Id	Rol	Actividades	Nivel Implicación	Acceso Información
Identificador único, puede utilizarse letras y números.	Función a desempeñar durante el desarrollo del proyecto.	Descripción de las responsabilidades o identificadores de las actividades asignadas durante el proyecto.	Puede ser medido en una escala numérica del 1 al 5.	Se detallan los documentos a los que posee acceso.

Fuente: elaboración propia.

2.3.5.3. Asignación de actividades

Las actividades a realizar cada uno de los interesados pueden ser desglosadas en una matriz individual para facilitar su control, a continuación, se puede observar una matriz de actividades, la misma que estará vinculada con la matriz especifica de stakeholders mediante su identificador.

Tabla 6. Plantilla de matriz de actividades.

MATRIZ DE ACTIVIDADES		
Identificador	Actividad	Descripción
Identificador único, puede utilizarse letras y números, este será utilizado en la matriz especifica de stakeholder.	Nombre de la actividad que se desempeñará	Descripción especifica de las tareas correspondientes a esta actividad.

Fuente: elaboración propia.

Como resultado de esta primera etapa se obtiene un acta de constitución de proyecto en el que sintetiza lo antes estudiado.

Tabla 7. Plantilla de acta de constitución.

ACTA DE CONSTITUCIÓN DEL PROYECTO		
Empresa / Organización		
Proyecto		
Fecha de preparación		
Cliente		
Patrocinador principal		
Gerente de proyecto		
Lista de Patrocinadores		
Nombre	Cargo	Departamento / División
Justificación del Proyecto		
Lista de Stakeholders		
Nombre	Cargo	Departamento / División
Premisas y restricciones		
Síntesis de análisis de factibilidad		
Presupuesto Inicial		
Requisitos de aprobación del proyecto		
Aprobaciones		
Interesados / Patrocinadores	Fecha	Firma

Fuente: elaboración propia.

2.3.6. Gestión del Alcance

Según PMBOK (s.f.) la Gestión del Alcance del Proyecto incluye los procesos necesarios para garantizar que el proyecto incluya todo el trabajo requerido y únicamente el trabajo para completar el proyecto con éxito.

En esta etapa se define un plan para la gestión del alcance, el que permitirá establecer cómo se van a definir, validar y controlar alcance del proyecto. Se pueden utilizar diversas técnicas para la obtención de los requerimientos, como son juicios de expertos, reuniones con los interesados y uso de prototipos, siendo estas dos últimas las herramientas utilizadas en este modelo, con la cual se procede a la elicitación de historias de usuarios.

2.3.6.1. Requerimientos funcionales y no funcionales

Requerimientos Funcionales

Se define como requerimientos funcionales a aquellos servicios o funciones que proveerá el software a sus usuarios, estos a la vez se convierten en limitadores sobre lo que la aplicación debe y no debe hacer.

Una correcta identificación de los requerimientos funcionales y la relación con los actores que interactuaran con el sistema es clave para el desarrollo del proyecto.

El Modelo Mobile Sprint propone las siguientes plantillas para la eficiente recolecciones y administración de los requerimientos funcionales del proyecto.

En primer lugar, una plantilla general sin tecnicismos que se utiliza para socializar estos requerimientos con el cliente.

Tabla 8. Plantilla general de requerimientos funcionales.

REQUERIMIENTOS FUNCIONALES		
Identificador	Requerimiento	Observación

Fuente: elaboración propia.

Por otro lado, tenemos una segunda plantilla destinada al equipo de desarrollo donde se describen detalles propios de cada requerimiento tales como:

- Identificador: por convención los requerimientos funcionales se codifican como RF-01.
- Pre-condición: es la acción que debe efectuarse para que este requerimiento pase a ejecución.
- Post-Condición: es la acción que se ejecutará luego de que la aplicación realice una determinada función.
- Descripción: en esta sección se debe ser muy específico en cuanto a las actividades que conlleva este requerimiento funcional.

Tabla 9. Plantilla especificación de requerimientos funcionales.

ESPECIFICACIÓN DE REQUERIMIENTOS FUNCIONALES					
Identificador		Fecha de Recepción		Versión	
Responsable					
Actor (es)					
Pre-condición					
Post-condición					
Descripción					
Observación					

Fuente: elaboración propia.

Requerimientos No Funcionales

Los requerimientos no funcionales hacen referencia a características generales, tipos de restricciones o atributos de calidad. Tomando en cuenta que este modelo se enfoca en el desarrollo de aplicaciones móviles es importante identificar este tipo de requerimientos ya que su ausencia puede afectar la experiencia de usuario. Es importante mencionar además que los requerimientos no funcionales pueden ser organizacionales o externos como en el caso de requerimientos éticos, legislativos,

entre otros.

Para la recolección de estos requerimientos se puede utilizar las plantillas propuestas para los requerimientos no funcionales, siempre respetando la convención de los identificadores.

Tabla 10. Plantilla general de requerimientos no funcionales.

REQUERIMIENTOS NO FUNCIONALES		
Identificador	Requerimiento	Observación

Fuente: elaboración propia.

Tabla 11. Plantilla especificación de requerimientos no funcionales.

ESPECIFICACIÓN DE REQUERIMIENTOS NO FUNCIONALES					
Identificador		Fecha de Recepción		Versión	
Responsable					
Actor (es)					
Pre-condición					
Post-condición					
Descripción					
Observación					

Fuente: elaboración propia.

2.3.6.2. Elicitación de historias de usuarios

La elicitación hace referencia a la recolección de los requerimientos para el desarrollo del software, el modelo planteado (MMS) hace uso de uno de una combinación de historias de usuarios junto con bosquejos de la aplicación diseñados en base lo mencionado, esto conlleva a que se organicen múltiples reuniones, pero se obtienen resultados más precisos en cuanto los requerimientos solicitados.

Las historias de usuarios se pueden definir como cartillas que poseen los pasos a seguir para efectuar una determinada actividad, estas indicaciones deben ser entendibles para los demás interesados, debido a que deben pasar por una revisión donde se valide y confirme su contenido.

Debido a las características propias de los dispositivos móviles como sus pantallas es necesario diseñar una aplicación que sea fácil de comprender y navegar para el usuario, pero a su vez tenga todo el potencial para cumplir con su función, por tal razón este modelo recomienda el uso de bosquejos digitales o prototipos tal como lo menciona (Duh *et al.*, 2016) en su investigación, de esta manera se busca complementar las historias de usuario.

Estas historias de usuario deben ser solucionadas mediante algunos de los requerimientos funcionales descritos en los apartados anteriores.

A continuación, se presenta una plantilla para la recolección de las historias de usuario, donde además se vinculan con los bosquejos diseñados en este proceso.

Tabla 12. Plantilla para la recolección de las historias de usuario.

HISTORIA DE USUARIO	
Identificador	
Actor	
Prioridad	
Sprint	
Responsable	
Requerimiento Funcional	
Observaciones	
Bosquejos – Prototipos	

Fuente: elaboración propia.

La plantilla de historias de usuarios consiste en una serie de campos que deben ser llenados correctamente para evitar confusiones, por esta razón se describe a continuación lo valores a ser asignados en cada parámetro.

- El identificador debe ser un valor único e irrepetible, pero a su vez secuencial para identificar fácilmente la continuidad de las historias. En caso de que la empresa maneje varios proyectos simultáneamente se recomienda utilizar un código adicional que haga referencia al proyecto en curso, por ejemplo: SAD-1, SAD-2…
- El actor es aquel usuario que efectuará la funcionalidad descrita en la historia.
- La prioridad se puede medir mediante los identificadores, baja, media y alta. Dichos valores estarán ligados a las reglas de negocio o a su vez a las necesidades del cliente.
- El sprint como su nombre lo indica hace referencia a la iteración del proyecto, recordemos que el Modelo Mobile Sprint tiene como base el desarrollo en ciclos cortos denominados sprints.
- El responsable por lo general siempre pertenece al grupo de desarrolladores, ya sea este un programador, diseñador o Tester.
- En requerimiento funcional se debe referenciar uno de los registrados previamente que solucionar la historia de usuario mencionada.
- La descripción debe ser escrita evitando palabras técnicas para facilitar la comprensión del cliente y gerente, detallando el ¿qué se va a realizar? y ¿cómo se lo va a realizar?
- La observación permite redactar indicaciones adicionales para el equipo de desarrollo.

Es importante mencionar que al finalizar cada sprint del proyecto se realizarán reuniones con el cliente, para analizar si es necesario la implementación de cambios.

2.3.6.3. Refinamiento de Requerimientos

Antes de pasar a la creación del sprint Backlog es necesario realizar un refinamiento de los requerimientos recolectados, esto consiste reunir a los miembros a cargo del proyecto, tales como gerente, analista y equipo de desarrollo para identificar aquellos que se encuentren incompletos o son poco factibles, debido a que los requerimientos deben ser descritos de la manera más clara y precisa.

2.3.6.4. Creación del Sprint Backlog

El sprint backlog es una planificación de los requerimientos a desarrollarse durante un sprint, se debe considerar el nivel de priorización y si estos pueden ser completados y demostrados al cliente al final de la iteración.

El Modelo Mobile Sprint propone la siguiente plantilla para la elaboración del Sprint Backlog.

Tabla 13. Plantilla para la elaboración del Sprint Backlog.

SPRINT BACKLOP														
Fecha Inicio	Fecha de culminación	Sprint	Requeri-miento	Tarea	Responsable	Semana	1						
						Dias	1	2	3	4	5	6	7
						Horas							
		Sprint A	RF-01	T-01	Responsable 1		16						
				T-02			8	2					
			RF-02	T-01			10	4	x	x			
				T-02			5	5	x	x			
				T-03			3	9	x	x			
			RF-03	T-01			6	10	x	x			
				T-02			16		x	x			
		Sprint B	RF-04	T-01	Responsable 1		16						
				T-02			8	2					
			RF-05	T-01			10	4	x	x			
				T-02			5	5	x	x			
				T-03			3	9	x	x			
			RF-06	T-01			6	10	x	x			
				T-02			16		x	x			

Fuente: elaboración propia.

Los sprint por general duran de 3 a 4 semanas, por lo que deben especificarse los días necesarios para el desarrollo de cada requerimiento, este a su vez posee una serie de tareas las mismas que pueden ser desarrolladas en un intervalo de horas, al igual que Scrum cada tarea tendrá destinado un mínimo de 4 horas y un máximo de 16, estas no deben ser muy pequeñas, pero si de un tamaño considerable de tal manera que se puede detectar los avances diarios del proyecto.

2.3.6.5 Elaboración de la EDT

La estructura de desglose de trabajo (EDT) es "una descomposición jerárquica orientada al trabajo que será ejecutado por el equipo del proyecto para lograr los objetivos del mismo y crear los entregables requeridos" (PMBOK, s.f.).

Una correcta elaboración de una EDT permite obtener una mayor precisión de la definición del alcance, así como obtener una visión más clara de las unidades o equipos de trabajo que conforman el proyecto. Además, facilita la asignación de recursos, responsabilidades, seguimiento y control, minimizando el riesgo y errores en el desarrollo de la aplicación.

Cuando las unidades de trabajo son muy grandes se pueden descomponer cada entregable en partes más pequeñas, en el caso de MMS el desglose de actividades empieza por sus fases, luego por el número de sprint, como se puede observar en la plantilla EDT a continuación:

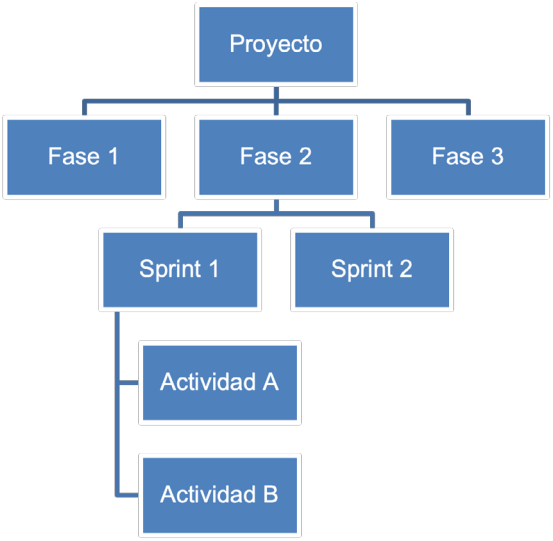


Gráfico 1. Plantilla de Estructura de Desglose de Trabajo (EDT).
Fuente: elaboración propia.

2.3.7 Gestión del Tiempo

El éxito de un proyecto de software tiene como pilar fundamental una correcta gestión del tiempo, por tal motivo, el gerente de proyecto debe saber administrar responsablemente el recurso humano que posee la organización al momento

de desarrollar las actividades planteadas. De esta manera se puede garantizar la viabilidad del proyecto y obtener entregables en el momento indicado. El Modelo Mobile Sprint ha considerado como actividades esenciales dentro de esta área la:

- Priorización de los sprint.
- Definición de la holgura.
- Elaboración del cronograma.

2.3.7.1 Priorización de Sprint

Antes de comenzar con la elaboración del cronograma de actividades es importante revisar la priorización de los sprint (Gestión del Alcance), de esta manera se detallarán los tiempos correctos para el cumplimiento de cada uno de ellos. Además, hay que recordar que pueden surgir pequeños contratiempos o cambios por lo que también es necesario definir una holgura apropiada.

2.3.7.2 Holgura del proyecto

Cada vez que se estima la duración de una actividad se debe tener presente que pueden existir contratiempos en interrumpan el flujo de trabajo, esta es la razón de asignar un tiempo adicional para la culminación de dicha tarea. La estimación del tiempo de holgura debe considerar el nivel de complejidad de la actividad, así como el recurso humano asignado, de tal forma que no todas las actividades poseerán el mismo tiempo de holgura.

2.3.7.3 Elaboración del cronograma

Una vez revisada la priorización de los sprint y estimar la holgura adecuada para las actividades correspondientes, se procede a la elaboración del documento más importante de un proyecto, como es el cronograma, recordando establecer tiempos medibles, realistas y estipulados con el resto del equipo de proyecto.

En la actualidad existen diversas herramientas que facilitan la elaboración de cronogramas, uno de los más conocidos es Microsoft Project que adicionalmente proporciona un diagrama de Gantt el mismo que permiten una mejor visualización de la planificación y secuencia de actividades.

2.3.8 Gestión de Costos

Esta gestión tiene como finalidad, estimar, presupuestar y financiar los costos de tal manera que el proyecto pueda ser realizado dentro del presupuesto aprobado con la otra parte interesada

2.3.8.1 Estimación de Costos

La estimación de los costos es una de las actividades más complejas dentro de un proyecto de software, por tal motivo es necesario implementar estrategias que permitan obtener una estimación lo más acertada posible.

Entre los modelos más utilizados para estimar costos en proyectos de desarrollo de software se encuentran:

- Modelos basados en SLOCS.
- Modelos no basados en SLOCS.
- Modelo basado en puntos de caso de uso.

Cuando se habla modelos basados en SLOCS son aquellos que utilizan como métrica de evaluación la cantidad (en miles) de líneas de código fuente utilizadas.

Dentro de este tipo de modelos el más conocido es el modelo COCOMO, el cual divide un proyecto en tres tipos dependiendo su tamaño: modo orgánico, semi-acoplado, y acoplado. Por el contrario, lo modelos no SLOCS utiliza preguntas, o transacciones de entrada (Antúnez et al., 2016)

El modelo Mobile Sprint deja a libre elección el modelo de estimación de costos, ya que la organización debe analizar cual representa un mayor beneficio para ellos.

2.3.8.2 Determinación del presupuesto

La determinación del presupuesto permite conocer si en términos económicos el desarrollo del proyecto es viable o no, considerando los costos en recursos humanos, tecnológicos, materiales, imprevistos y otros recursos generales. Los mismos que fueron obtenidos en el análisis de factibilidad.

Tabla 14. Plantilla para la determinación del presupuesto.

DETERMINACIÓN DE PRESUPUESTO	
Recurso Humano	$ --.--
Recurso Tecnológico	$ --.--
Recurso Material	$ --.--
Recursos Generales	$ --.--
Imprevistos ($)	La organización establecerá el porcentaje de imprevistos
PRESUPUESTO	$ --.--

Fuente: elaboración propia.

2.3.9 Gestión de Cambios

Los proyectos de software en su mayoría están sujetos a cambios durante su desarrollo, estos deben gestionarse eficientemente ya que podrían ser los causantes del fracaso del proyecto.

Un cambio es aquel que modifica una o varias limitaciones iniciales del proyecto, por esta razón deben ser correctamente supervisados y llevar un control exacto de los mismos. Esta modelo propone efectuar reuniones diarias (Daily Sprint Meeting) que permitan conocer los avances de proyectos y si existe las necesidades de efectuar modificaciones.

2.3.9.1 Daily Sprint Meeting (DSM)

Una forma de asegurar una mejor coordinación entre los miembros del equipo de desarrollo del proyecto es llevar a cabo las reuniones diaria donde se exponen los avances que cada integrante ha realizado, además de conocer si hay que efectuar cambios sobre el proyecto.

Estas reuniones son de carácter informal y normalmente son realizadas antes de empezar la jornada laboral.

2.3.9.2 Solicitudes de cambio

En caso de ser necesario realizar un cambio sobre el proyecto se debe presentar una solicitud de cambios al gerente del proyecto, las mismas que deben detallar las características generales y específicas a ser modificadas y además deben ser complementadas con un estudio técnico acerca del impacto que pueden generar dentro de la aplicación.

Luego estas solicitudes son revisadas por el gerente para ser aprobadas o rechazadas.

Tabla 15. Plantilla de solicitud de cambio

SOLICITUD DE CAMBIO					
N. Solicitud					
Solicitante					
Area/Departamento					
Gerente					
Categoria de cambio					
Alcance		Procedimientos		Recursos	
Cronograma		Documentación		Otros	
Costos		Calidad			
Causa del cambio					
Solicitud del cliente		Reparación			
Acción Preventiva		Otros			
Acción Correctiva					
Descripción de la propuesta de cambio					
Justificación de la propuesta de cambio					
Impacto del cambio					
Alcance					
Cronograma					
Costo					
Calidad					
Implicaciones de Recursos					

Fuente: elaboración propia

2.3.9.3 Matriz de seguimiento de cambios

Durante el desarrollo de un proyecto pueden surgir múltiples solicitudes de cambios por esta razón es importante tener un correcto control de cada uno de ellos, el Modelo Mobile Sprint propone el uso de la siguiente plantilla donde mediante el identificador se referencia al número de solicitud de cambio registrada, y una vez

que esta sea revisada se debe detallar el estado de la misma (aceptada o rechazada).

Tabla 16. Plantilla de solicitud de cambio.

GESTIÓN DE CAMBIOS					
Fecha	Identificador	Fase / Sprint	Descripción general	Solicitante	Estado

Fuente: elaboración propia.

2.3.10 Gestión de Calidad

En esta sección se describen las normativas o estándares de calidad que la organización implementará durante el desarrollo del proyecto, las mismas que pueden contemplar las responsabilidades, procedimientos o recursos necesarios para asegurar la calidad de la aplicación.

El objetivo de la implementación de estándares de calidad es obtener software de mayor confiabilidad, mantenibilidad y productividad.

En la actualidad existen muchas normativas o estándares de calidad que pueden ser incorporados a los proyectos de software, tales como las normativas ISO/IEC, SPICE, CMMI, entre otros.

Adicionalmente se deben tener presente que el desarrollo de aplicaciones móviles conlleva nuevos retos respecto a la calidad debido a los atributos propios de los dispositivos móviles, tanto de hardware y software (Gao y Yin, 2020).

2.3.10.1 Métricas de calidad para aplicaciones Móviles

La diversidad de características en los dispositivos móviles se ha convertido en uno de los nuevos retos para la ingeniería de software, asegurar la calidad de las aplicaciones en estos dispositivos va más allá de la aplicación de buenas prácticas durante su desarrollo.

Algunos investigadores han identificado diversas métricas que pueden ayudar asegurar la calidad de las aplicaciones móviles y las cuales deben ser consideradas durante la fase de desarrollo tales como:

- **Interfaz de usuario:** la misma que debe ser amigable, interactiva y facilite la accesibilidad de cualquier usuario (Davalbhakta et al., 2020).
- **Multiplataforma:** en la actualidad existes dos plataformas software muy bien posicionadas para dispositivos móviles, como son iOS y Android, ambas poseen un número significativo de usuarios, por esta razón al momento de desarrollar una aplicación móvil, esta debe ser capaz de ejecutarse en dichas plataformas sin que esto afecte su rendimiento (Biørn-Hansen et al., 2020).
- **Recursos de hardware:** la capacidad de hardware que poseen los dispositivos móviles es muy variada, por ello se debe considerar el tamaño de las pantallas, el consumo de energía, memoria, almacenamiento y red, así como el rendimiento, para brindar una aplicación de calidad (Maia, Gonçalves y da Rocha, 2019).
- **Datos:** el aseguramiento, disponibilidad y privacidad de los datos es otro de los factores que los clientes toman en cuenta al momento de utilizar una aplicación móvil (Xiang et al., 2020).

En base a lo mencionado anteriormente se puede manifestar que el desarrollo de una aplicación móvil abarca muchos atributos adicionales al desarrollo de software tradicional, está en la razón principal por la cual se desarrolló el modelo Mobile Sprint como referente para la construcción de aplicaciones móviles, el mismo que implementa el estándar de calidad ISO/IEC 25010 para el aseguramiento de la calidad en sus productos.

2.3.10.2 Estándares de Calidad

MMS utiliza el estándar ISO/IEC 25010 para la evaluación de la calidad de sus productos por ser uno de los estándares que más se ajusta al contexto de las aplicaciones móviles, a pesar de que no está definido para tal propósito.

El estándar está conformado por ocho características las cuales se mencionan a continuación:

- Adecuación funcional.
- Eficiencia de desempeño.
- Compatibilidad.

- Usabilidad.
- Fiabilidad.
- Seguridad.
- Mantenibilidad.
- Portabilidad.

En base a las métricas que proponen diversos autores junto a las características de este estándar se efectuar una evaluación que permita conocer si la aplicación de desarrollada cumple con los lineamientos de calidad.

CAPÍTULO III: FASE DE DISEÑO

El desarrollo de software está estructurado por diferentes fases orientadas a distintos aspectos encaminados a mantener la calidad del producto final de software. Por ello, durante el ciclo de vida del proyecto la fase de diseño está a cargo de transformar todas aquellas ideas en un modelo estructurado el cual define la arquitectura del sistema.

El diseño del software transforma los elementos estructurales de la arquitectura del programa. Se puede utilizar una palabra "calidad" para definir la importancia del diseño de software, que puede mejorar la calidad del proyecto. El diseño es la única forma de satisfacer con precisión las necesidades del cliente.

La fase de diseño de la metodología Modelo Móvil Sprint está definida de acuerdo a la esencia del desarrollo ágil y su enfoque sobre el uso de sprint para lograr una mejor adaptación a la naturaleza del cambio que tienen los proyectos de desarrollo de software.

3.1 Objetivos

Esta fase tiene como objetivo implementar todos los requisitos del cliente y del análisis realizado para la construcción del software aplicando técnicas, herramientas y buenas practicas enfocadas en mantener el nivel de calidad del producto final.

El resultado debe proporcionar un concepto de software completo, centrándose en los dominios de datos, funciones y comportamiento desde una perspectiva de implementación. Para evaluar la calidad de las presentaciones de diseño, se deben establecer estándares técnicos para un buen diseño, tales como:

- Producir módulos con características operativas independientes
- Utilizar un método repetible para diseñar basado en la información obtenida durante el análisis de requisitos de software.
- Proponer una organización jerárquica, que puede utilizar inteligentemente el control entre componentes de software.

3.2 ¿Quiénes intervienen?

Al ser una metodología orientada al desarrollo ágil emplea ciclos cortos para cada iteración o también llamados sprint, el cual define un ritmo de trabajo para todos los

involucrados con el proyecto. El objetivo de emplear los sprint es lograr subdividir a los proyectos extensos en proyectos más cortos con una durabilidad más corta.

En cada iteración se llevan a cabo reuniones entre el equipo de trabajo y cliente con la finalidad de integrar la opinión del usuario con el desarrollo del producto. De forma general, un sprint comprende el análisis de los requerimientos hasta la fase de retrospección que es donde se ejecuta la evaluación de errores que se detectaron durante la ejecución del sprint y a la vez cómo solucionarlos Ilustración 2.

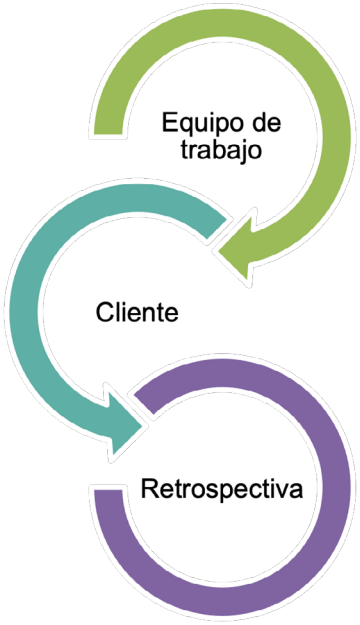


Ilustración 2. Involucrados en un Sprint.
Fuente: elaboración propia.

3.3 Actividades en la fase de diseño

Durante el ciclo de diseño se tiene la colaboración de los involucrados o interesados en el proyecto y actividades enfocadas a la extracción de requerimientos del cliente y su transformación a una estructura física del software para alcanzar a cubrir aquellas expectativas.

El proceso del diseño contempla actividades que le permiten analizar los casos de estudio y realimentar con la observación desde diferentes perspectivas para la construcción de modelados más robustos orientados a mantener la calidad en

el flujo de procesos y gestión de datos. Todo esto se lleva a cabo en la sección de análisis de requerimientos.

Por otra parte, está la construcción de las posibles soluciones orientadas a las iteraciones de ciclos cortos, simplicidad y la sostenibilidad de calidad. Esto implica la participación de las actividades de análisis de requerimientos donde se realiza el levantamiento de la información que será analizada posteriormente y como resultado se obtiene la generación de perspectivas. A partir de la generación de perspectivas se procede a la elaboración de la solución ágil.

La arquitectura de datos también es otro factor determinante en la aplicación de la metodología puesto que establece el flujo de datos que mantendrá la aplicación, el comportamiento y gestión de los mismos por parte del gestor de la base de datos. En esta sección se plantea el uso del modelo entidad-relación y el modelo relacional, con el objetivo de realizar un análisis más refinado en cuanto a las entidades y atributos que se van a emplear.

Al igual que otras metodologías Modelo Móvil Sprint, desarrolla su propio diccionario de datos pensado en las necesidades para la gestión apropiada de las características tanto de entidades u objetos de estudio, así como de sus característicos.

Cada una de estas actividades están relacionadas entre si como se muestra en la Ilustración 3, de esta manera se integran los resultados de cada sub fase dentro de la etapa de diseño.

Ilustración 3. Etapas y actividades de la fase del diseño.

Fuente: elaboración propia

3.4 Análisis de requerimientos

El análisis de requerimientos es un proceso donde se determina la información necesaria para la comprensión del problema, empleando instrumentos y técnicas

por parte del analista con el objetivo de cristalizar la información recibida en un marco conceptual determinado por medio de la sistematización de actividades a realizar y la asignación de roles y funciones de los interesados.

3.4.1 Reconocimiento y evaluación del problema

En esta sección se realiza el levantamiento de la información relacionada a las especificaciones del sistema en conjunto con el plan de proyecto del software. En este punto, es necesaria la comprensión del funcionamiento del software en el contexto de implementación, por ende, los analistas deben fomentar estrategias de intercomunicación con los involucrados del proyecto para la asimilación del problema desde la perspectiva del usuario.

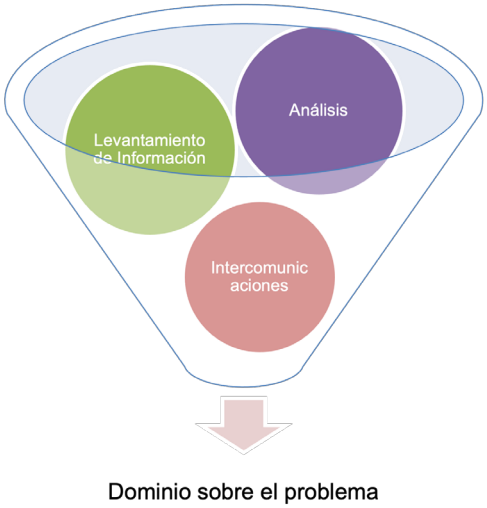


Ilustración 4. Componentes para el reconocimiento y evaluación del problema.
Fuente: elaboración propia.

3.4.2 Observaciones múltiples

En la etapa anterior se plantea el reconocimiento y evaluación del problema con el objetivo de establecer un dominio completo sobre el mismo. A partir de esto, se procede a realizar modelados de posibles soluciones desde varias perspectivas donde se consideran aspectos como: el flujo de datos, control de acciones, asignación de roles y los procesos externos e internos.

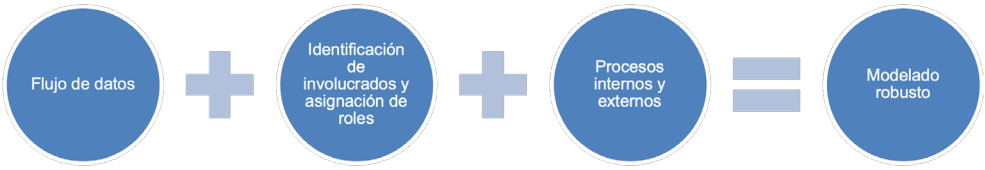


Ilustración 5. Representación del modelado basado en perspectivas.
Fuente: elaboración propia.

La generación de modelados desde diferentes perspectivas permite contemplar la mayor cantidad de aspectos lo cual facilita la realización de unificar los diagramas y generar un modelo de mayor robustez para el desarrollo de soluciones que satisfagan los requerimientos del sistema (Ilustración 4).

Para la documentación de las observaciones desde otras perspectivas se emplea una ficha con el siguiente formato (Ilustración 4), que recoge datos similares a una historia de Usuario utilizados en la metodología Scrum (Mariño, 2014).

Tabla 17. Plantilla para perspectiva de análisis.

PERSPECTIVA DE ANÁLISIS			
Código:		Usuario:	
Descripción:			
Observaciones:			
Número de iteración:		Nivel prioridad:	

Fuente: elaboración propia.

La plantilla está compuesta por los siguientes parámetros, véase también en el ejemplo dentro de la Ilustración 6:

- **Código:** Corresponde al identificador de cada ficha y debe mantener el formato N.E, donde N es el número de la ficha y E es el número de evaluación y corrección de esa ficha. Ejemplo: 1.2 (esto indica que la ficha 1 ha sido evaluada y corregida 2 veces)
- **Usuario:** Este parámetro indica al usuario que hace la observación.
- **Descripción:** El campo descripción se debe hacer uso para detallar a profundidad los aspectos que han sido considerados por parte del

usuario en el análisis de una situación o caso de estudio.

- **Observaciones:** Se describe de qué forma esta ficha puede afectar a otra o a su vez, la manera en cómo se relaciona con el resto.
- **Número de iteración:** Indica el número del sprint en el que tuvo como origen la ficha.
- **Nivel de Prioridad:** este campo sirve para colocar una jerarquía entre las fichas desarrolladas, la cual representa el orden de ser atendida. Se emplea el siguiente formato de enumeración: 1 al 5, donde 1 es de mayor importancia y 5 de menor importancia.

Tabla 18. Ejemplo de una ficha de Perspectiva de Análisis.

PERSPECTIVA DE ANÁLISIS			
Código:	1.2	Usuario:	Juan Armijos (Analista principal)
Descripción:	Para el módulo de facturación hay que añadir una opción que permita ingresar un nuevo cliente sin hacer uso del módulo de gestión de clientes, esto es en caso de que haya la necesidad de añadir un nuevo cliente durante el proceso de facturación.		
Observaciones:	Está relacionada al módulo de gestión de clientes, por lo cual se puede reutilizar el proceso de añadir un nuevo cliente.		
Número de iteración:	2	Nivel prioridad:	3

Fuente: elaboración propia.

Al finalizar este proceso de observación y análisis se pueden plantear nuevas soluciones a partir de las correcciones necesarias para obtener modelados eficientes. Este proceso se caracteriza por la constancia durante el ciclo de vida del proyecto haciéndolo flexible al cambio, la cual es una característica propia de los proyectos de desarrollo de software.

3.5 Planteamiento de solución ágil

En las metodologías de desarrollo de software que emplean o no, procesos y/o principios ágiles tienen como objetivo de manera general producir software con altos índices de calidad. Sin embargo, esto no se logra cristalizar en todos los proyectos de desarrollo. Entre las razones más comunes por la que un proyecto de software no cumple con las expectativas están las siguientes:

- Falta de participación del cliente.

- Planificación incorrecta.
- Inflexibilidad al cambio.
- Diseño inapropiado de actividades.
- Corrupción del alcance.

Modelo Móvil Sprint toma en consideración los factores que generan los errores más frecuentes durante el ciclo de vida del proyecto de software y plantea soluciones más adaptables al contexto de aplicación. Al ser una metodología con enfoque ágil da prioridad a las interacciones e iteraciones, es decir: fortalece la comunicación con las partes interesadas y lo hace de manera periódica a través de reuniones.

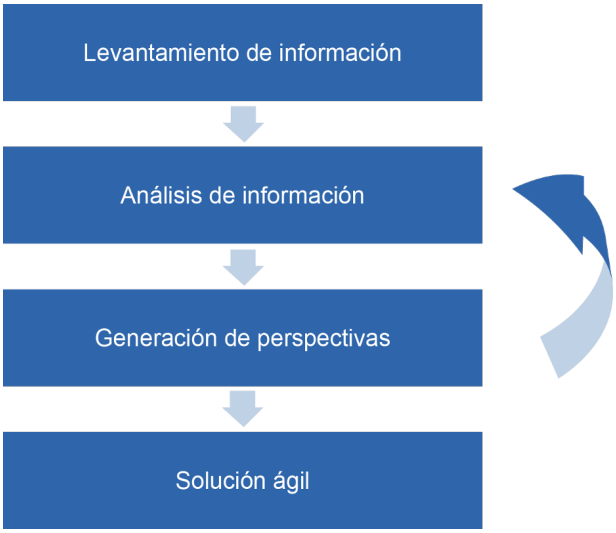


Ilustración 6. Proceso de generación de la solución ágil.
Fuente: elaboración propia.

A partir del levantamiento de información y el análisis de diferentes perspectivas en relación al desarrollo de una solución que satisfaga los requerimientos en su totalidad, se construyen modelados enfocados a los siguientes conceptos:

- ***Simplicidad***

Se ha comprobado que los diseños con características complejas consumen una mayor cantidad de tiempo en lograr su implementación. Además, existe una probabilidad muy alta de cometer errores que no son detectados a tiempo.

Por ello, esta metodología se enfoca a desarrollar diseños simples, eficientes y accesibles. Esto permite que puedan ser interpretados con facilidad, satisfagan los requerimientos y que puedan estar disponibles para los involucrados en la parte operativa del proyecto en desarrollo.

- ***Iterativo e incremental***

Es una cualidad importante de Modelo Móvil Sprint, subdivide los procesos más extensos en mini procesos. De esta manera permite centrarse en satisfacer los requerimientos de manera escalonada hasta el objetivo principal. Cada subproceso es tratado como una iteración en el cual se obtienen incrementos y al final es considerado como crecimiento del producto.

Este proceso se lleva a cabo como se muestra en la Ilustración 7. Comienza con el análisis de las perspectivas luego realiza la retroalimentación de requerimientos y termina con la aprobación y desarrollo de la posible solución.

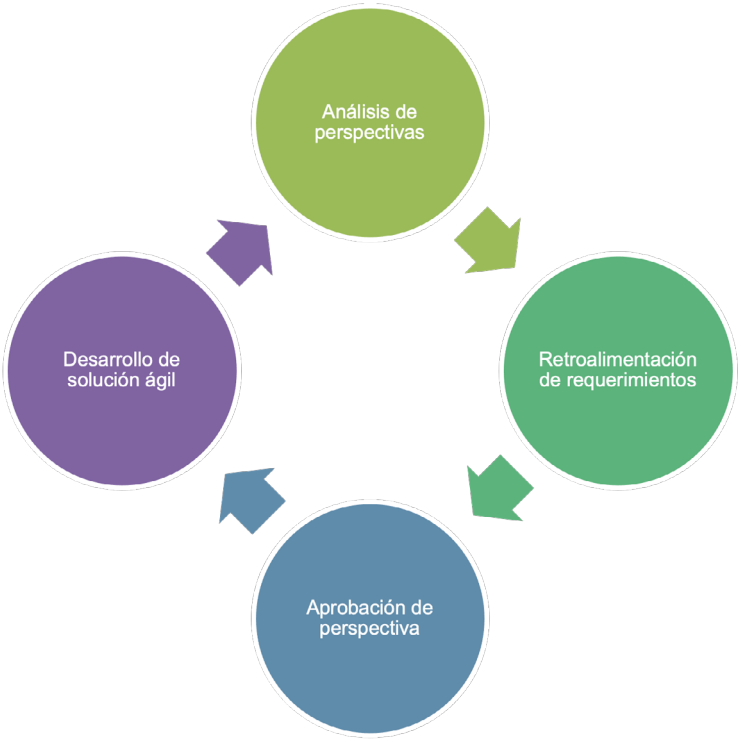


Ilustración 7. Proceso iterativo e incremental (interno).
Fuente: elaboración propia.

- ***Calidad***

En el desarrollo de proyectos informáticos la carencia de calidad en los diseños es uno de los problemas más comunes por diferentes factores como considerar a la calidad como una opción y no una prioridad, creer que la calidad se debe aplicar al final del proyecto o responsabilizar a un grupo de personas y no realizar procesos para controlar y realizar el aseguramiento de la misma.

Modelo Móvil Sprint puntualiza algunas funciones dentro del control de calidad para realizar el seguimiento a los cambios y de cómo afectan a los resultados finales sean positivos o negativos.

3.6 Modelado de la base de datos

En los proyectos de desarrollo de software se encuentra implícito el uso de medios para almacenamiento de la información. Sin embargo, es necesario definir la gestión que se va a desarrollar sobre el mismo debido a la importancia de mantener la integridad de los datos que se generan en las distintas áreas de la empresa como sucede a nivel transaccional.

De manera similar, el modelo de la base de datos también establece qué clase de procesos podrán efectuarse con los datos, además determina la forma en que se utilizan y proporciona la base para diseñar un lenguaje de consulta.

Las bases de datos son la representación lógica de los procesos o funciones abstractas que se desempeñan en las labores cotidianas que pretenden ser automatizadas con la implementación del software. La importancia de desarrollar un buen modelado de base de datos radica en analizar a profundidad todas las entidades que son partícipes y la forma en cómo se desenvuelven para realizar dicha acción.

Existen casos (Zúñiga-Vásquez, 2017) donde un mal diseño puede desencadenar una lista de errores que provocan retrasos importantes en cuanto a la planificación y su seguimiento del proyecto.

Por tal motivo, en la presente metodología se implementan dos tipos de modelados, el modelo de entidad-relación y el relacional. Cada uno cumple con funciones diferentes y complementarias para definir los flujos de datos tanto en el almacenamiento como el acceso o consumo desde la aplicación.

3.6.1 Modelo Entidad-Relación (MER)

De acuerdo con Zea (2017), el Modelo Entidad-Relación, es considerado como el paso previo a desarrollar el modelado relacional. Generalmente se complementa con una breve descripción donde se especifican los atributos y el tipo de relaciones por cada elemento, entidad u objeto dentro del análisis. En la presente metodología se utiliza este tipo de diagramas como un filtrado en el proceso de desarrollo del modelo relacional, donde se van descartando elementos o añadiendo según sea el caso.

3.6.2 Modelo Relacional (MR)

Luego del desarrollo del modelado de entidades y relaciones, se comienza a construir la estructura lógica que va a emplear el gestor de base de datos como tal. Es decir, este modelado hace uso del modelo entidad-relación y lo convierte en un diagrama que está compuesto por tablas o conjunto de tablas que mantienen una relación lógica. Este tipo de diagramas se caracteriza por ser de fácil comprensión para los usuarios inexpertos debido a que emplea conceptos como lógica de predicados.

3.6.3 Diccionario de datos

En la gestión de datos es importante manejar una bitácora que describa el comportamiento, relaciones y atributos que pueden llegar a tener los datos dentro de la gestión de una base de datos además puede identificar claramente el propósito, alcance y dominio de la aplicación, dando a los usuarios direcciones para el tipo de información que encontrarán en su contenido.

En la Tabla 19 se especifica un modelo de diccionario de datos elaborado para gestionar de manera correcta la descripción de los datos al igual que su flujo dentro de los procesos del software.

Tabla 19. Modelo del diccionario de datos.

Fecha:		Nº Actualización:		
Descripción:				
Campo	Tipo de dato	Tamaño	Descripción	Restricciones
Relación (Tabla/s):			Campos prioritatios:	

Fuente: elaboración propia.

En el modelo planteado se consideran los siguientes campos para su descripción:

- **Fecha:** indica cuándo se realizó la actualización del diccionario de datos.
- **Nº de Actualización:** identifica el numero de actualización del diccionario:
- **Descripción:** detalla a precisión las causas y que elementos están involucrados en la actualización.
- **Campo:** requiero el nombre del campo sobre el cual está tratándose, también se lo puede identificar como el nombre del atributo de la entidad.
- **Tipo de dato:** indica la clase de dato que se utiliza dentro del gestor de la base de datos.
- **Tamaño:** se utiliza para estipular el rango de almacenamiento requerido del dato dentro de la base de datos.
- **Descripción:** detalla el significado del campo, es decir especifica el valor que está representando.
- **Restricciones:** se colocan aquellas restricciones que son necesarias de acuerdo a los requerimientos del cliente y del gestor de la base de datos.
- **Relación:** detalla las tablas con las que se encuentra relacionada en el flujo de datos.
- **Campos prioritarios:** hace referencia a aquellos campos utilizados tanto para clave principal como claves foráneas.

3.7 Definición de interfaz

En el diseño de interfaz de usuario se utilizan conceptos como el DCU cuyas siglas corresponden al nombre de Diseño Centrado en el Usuario. Las interfaces de usuario son el resultado de una ardua labor que plantea definir el aspecto ergonómico, la función y utilidad que son reflejados en la apariencia de un sistema o software.

Aunque resulte relativamente fácil desarrollar una interfaz de usuario, esta debe cumplir con el objetivo maximizar la funcionalidad y usabilidad en la experiencia del usuario para garantizar que la interacción entre el usuario y el software se desarrolle de la manera más natural, simple, eficiente y objetiva en relación del propósito de la

aplicación con el cliente.

El concepto DCU, integra un conjunto de métodos y tecnologías diferentes con un objetivo común de comprender las necesidades, limitaciones, comportamientos y características de los usuarios, que involucran muchas situaciones que los incluyen. La base de esta concepción está fundamentada con las preguntas ¿Qué? ¿Cómo? Y ¿Cuándo?

- **¿Qué?:** Estas pruebas se basan en observar cómo un grupo de usuarios realiza una serie de tareas encomendadas por el evaluador y analizar los problemas de usabilidad que encuentran.

  Incluso si el diseñador tiene una amplia comprensión de la usabilidad, se recomienda evaluar el diseño con el usuario. Esto se debe a que cuanto más tiempo pasa el diseñador en el proyecto, menos conocimiento tiene, más difícil es encontrar posibles problemas. Se puede decir que gran parte de lo que sienten los diseñadores al mirar su propio trabajo es una estructura mental. Vea lo que piensa, no lo que ve el usuario.

- **¿Cómo?:** El número de participantes necesarios para detectar el total de los problemas de usabilidad del diseño es de aproximadamente 15. Sin embargo, no es recomendables utilizar ese número de participantes, sino realizar tres pruebas entre los 5 participantes de cada prueba que se encuentran dispersas en diferentes momentos del proceso de desarrollo.

  El propósito de estas pruebas es mejorar iterativamente la usabilidad de la aplicación, aun si no se detectan los problemas en su totalidad debido a que cada prueba con 5 participantes nos proporcionará información suficiente para mejorar la solución de diseño.

  Cada participante realizará la prueba por separado, y durante cada prueba, debemos registrar toda la información relevante para el posterior análisis del comportamiento del usuario. Para hacer esto, puede grabar el video del usuario grabando la operación del usuario en la interfaz y usándolo desde el bloc de notas.

- **¿Cuándo?:** Aunque los test de usuario son una prueba de evaluación, no se debe considerar que, una vez completado el proceso de diseño, desarrollo e implementación del producto, se deba aprobar. En retrospectiva el DCU es un concepto de diseño iterativo basado en la mejora incremental del producto.

En Modelo Móvil Sprint el desarrollo de la interfaz sigue un proceso de planificación para el diseño de prototipos que proceden a su evaluación para luego de ser aprobados pasen a la fase de implementación como se observa en la Ilustración.

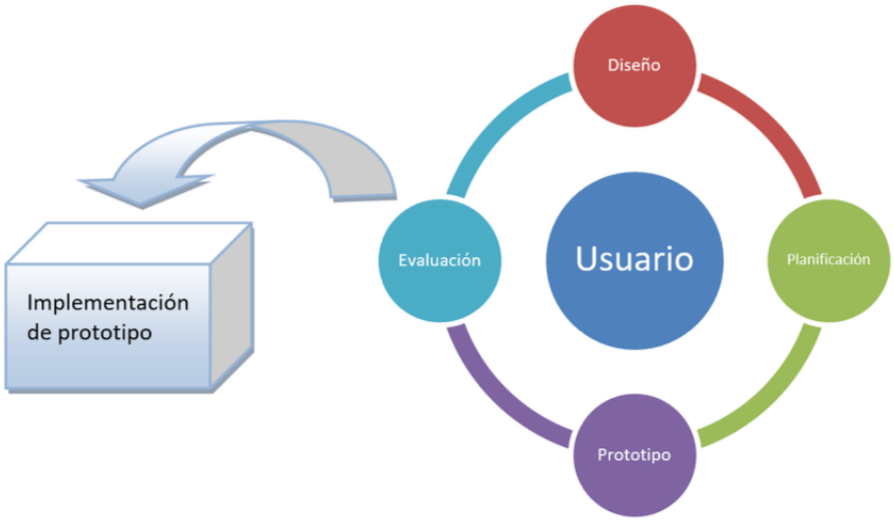


Ilustración 8. Modelo centrado en el usuario.
Fuente: elaboración propia.

3.7.1 Diseño Modular

Desarrollar el diseño en base a módulos lógicos permite observar todas aquellas relaciones existentes entre los componentes que serán parte del diseño final. Esto conlleva a clarificar el flujo de información que pueden mantener entre dichos módulos por medio de sus conectores o enlaces también llamados protocolos de comunicación.

El diseño modular estimula al funcionamiento organizado dentro de la aplicación y a la vez fomenta el trabajo ordenado para la asignación de funciones, procesos y el flujo de datos que manejarán los componentes que formen parte de dicha interfaz.

Las fases del diseño modular implementadas por parte de Modelo Móvil Sprint

Ilustración, están estructuradas de acuerdo a una jerarquía planteada que pretende clasificar los módulos considerando la prioridad de elementos y funciones al igual que el flujo de datos.

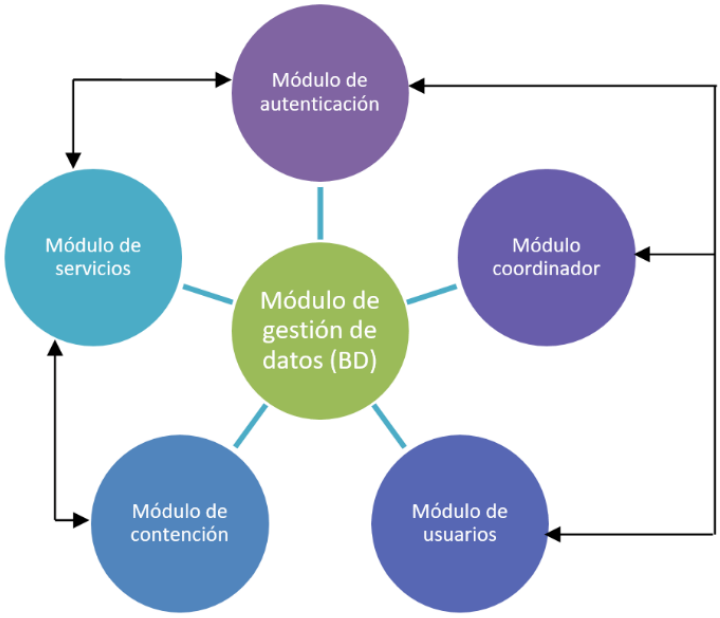


Ilustración 9. Componentes del diseño modular.
Fuente: elaboración propia.

3.7.2 ¿Qué se consideran en los módulos?

3.7.2.1 Módulo de autenticación

En el módulo de autenticación se gestiona e implementa procesos para validar el tipo de sesiones y control de acceso a la aplicación por medio de una cuenta de usuario. Cabe resaltar que este módulo se caracteriza por desarrollarse por medio de clases abstractas, es decir, no es visible para el usuario final. Sin embargo, desempeña un rol sustancial para resguardar la integridad de los datos que se manipulan dentro de la aplicación.

3.7.2.2 Módulo de gestión de datos

La gestión de datos dentro de la aplicación es responsabilidad del presente módulo, el cual tiene como objetivo principal gestionar la accesibilidad al consumo de información almacenados en la base de datos. Al igual que el módulo anterior, no es visible para el usuario final.

No obstante, se encarga de realizar los procesos o funciones requeridos por parte de los componentes que requieren la accesibilidad a la base de datos para visualizarlos al usuario.

3.7.2.3 Módulo de coordinador

Se encarga de realizar el prototipado para la interfaz de usuario empleando las conexiones con el módulo de usuarios y autenticación por medio de protocolos de verificación de sesiones y su acceso a la aplicación.

El objetivo principal de este módulo es desarrollar un diseño simple, eficaz y ergonómico para mejorar la interacción entre el usuario y el software. De esta manera la experiencia del cliente será más satisfactoria debido a la sensación de control sobre la aplicación lo cual no ocurre cuando el diseño de la aplicación es complejo a la hora de realizar tareas o acciones para la cual fue desarrollada.

3.7.2.4 Módulo de usuarios

La gestión visual para cada usuario se compone del nivel de privilegios de cada usuario y el tipo de cuenta (en caso de las aplicaciones con esas características). Este módulo distribuye los elementos acordes al nivel del usuario y contempla el uso de otros módulos como el de autenticación para validar las sesiones y el coordinador para definir el prototipo.

3.7.2.5 Módulo de contención

En este módulo se maneja la distribución de algunas funciones que puede implementar la aplicación y para ello necesita de los módulos de autenticación y servicios. Cabe señalar, para la correcta ejecución de este módulo se debe tener en consideración las siguientes recomendaciones:

- El sistema debe ser coherente para facilitar su uso.
- Las tareas que pueden realizar los usuarios deben poder aprender rápidamente.
- El diseño debe minimizar los riesgos y consecuencias de las operaciones involuntarias que puedan realizar los usuarios.
- Siempre se deben tener en cuenta los errores importantes que pueden cometer los usuarios objetivo del producto.

3.7.2.6 Módulo de servicios

A nivel de clases, pueden representarse todas aquellas funcionalidades que provienen de aplicaciones externas o terceras que intervienen en el correcto funcionamiento de la aplicación. Al igual que otros módulos, este se integra para corroborar el uso de recursos desde la aplicación por parte del usuario como por ejemplo el uso del API de tecnologías utilizadas en el desarrollo y ejecución de la aplicación.

CAPÍTULO IV: FASE DE EJECUCIÓN

En las metodologías de desarrollo de software implícitamente se incorporan procesos de integración de recursos disponibles para el proyecto a fin de llevar a cabo la planificación desarrollada. Esto se puede definir como una fase de ejecución. Modelo Móvil Sprint implementa buenas practicas a seguir para efectuar la correcta incorporación de los instrumentos, funciones y recursos utilizables.

En la fase anterior de diseño se establecieron los prototipos desarrollados en base a las consideraciones que fueron planteadas tras el análisis de los requerimientos del proyecto de software.

A continuación, se procede a realizar la implementación de aquellos prototipos en función de las sugerencias establecidas para cada parte del proceso que involucra las etapas de codificación, el diseño de pruebas y la refactorización.

4.1 Objetivos

En todo proyecto de software se considera que la etapa en donde se consumen la mayor cantidad de los recursos es en la fase de ejecución puesto que lleva a cabo toda la planificación y construcción de los análisis realizados en cuanto a los requerimientos que se desean satisfacer.

Al ser una etapa donde se consumen gran parte de los recursos se debe gestionar de manera óptima para evitar contratiempos por falta de los mismos. Por ello, el objetivo principal de esta etapa es la implementación de buenas prácticas orientadas a la simplificación del trabajo y la maximización de la producción, además de integrar herramientas que permitan obtener un resultado satisfactorio.

4.2 Herramientas involucradas

Modelo Móvil Sprint emplea tres herramientas específicas para la fase de ejecución, las cuales le permiten a los grupos de desarrollo mantener una organización limpia y fluida indistintamente si se trata de macro proyectos o micro proyectos.

Las herramientas seleccionadas para implementar durante esta fase se clasifican en:

- Control de versiones.
- Editor de código.

Para el control de versiones se aplican dos herramientas las cuales son: Github y Azure DevOps. Aunque su función es similar, cabe resaltar que en conjunto se convierten en un sistema bastante completo para el control de versiones. Esto se debe a la compatibilidad que existe entre sí, mientras Azure DevOps mantiene un enfoque sobre la gestión de las versiones, Github trabaja con repositorios de código.

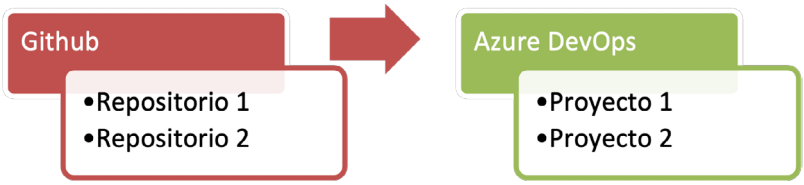


Ilustración 10. Relación entre Github y Azure DevOps.
Fuente: elaboración propia.

4.3 Actividades en la fase de ejecución

La fase de ejecución está compuesta de manera general por el proceso de codificación y refactorización, donde cada una ejemplifica las buenas prácticas que se recomiendan dentro de la presente metodología. Gran parte de los desarrolladores experimentados emplean y desarrollan distintas buenas prácticas para mantener una codificación más limpia y sobre todo realizar un mayor control sobre el desarrollo de funciones, variables, librerías, recursos, entre otras.

De acuerdo con la Ilustración 12, las buenas prácticas planteadas se ejecutan en el siguiente orden con el objetivo de preparar el entorno de trabajo antes de llevarlo a cabo.

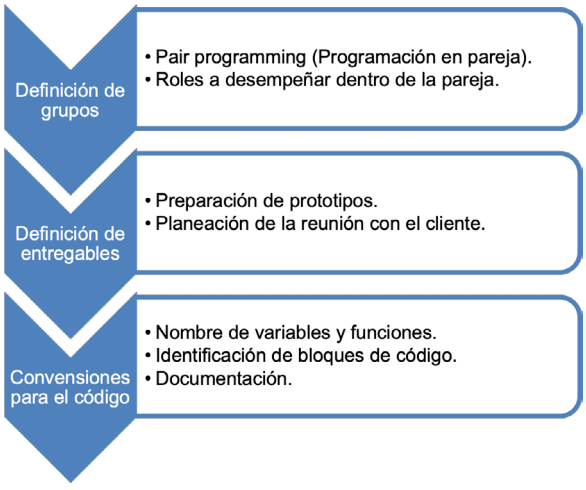


Ilustración 11. Actividades y buenas prácticas en la codificación.
Fuente: elaboración propia.

Mientras que para la refactorización se explican en la Ilustración 13, cabe resaltar que el propósito de la refactorización es simplificar y optimizar los bloques de código para obtener una mayor legibilidad del código, así como un mejor entendimiento. Por ello se enfatiza a plantear las siguientes buenas prácticas:

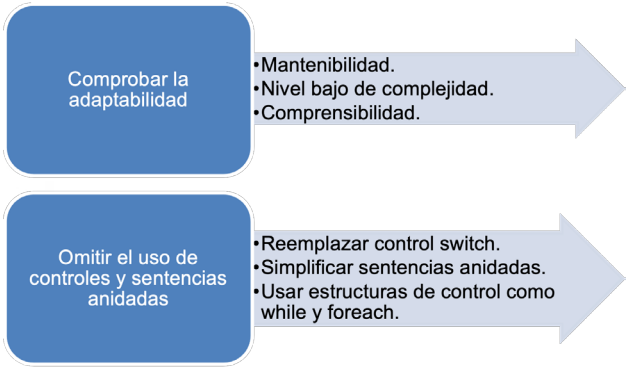


Ilustración 12. Buenas prácticas en el proceso de refactorización.
Fuente: elaboración propia.

4.4 Codificación

La codificación del software según Palomo (2020), lo define como el proceso de llevar los requerimientos funcionales del software al desarrollo del código fuente en un determinado lenguaje de programación. Generalmente esta disciplina suele ser desempeñada de forma desordenada e ineficiente por la carencia de herramientas y/o buenas prácticas que garanticen la compresión del código por desarrolladores externos.

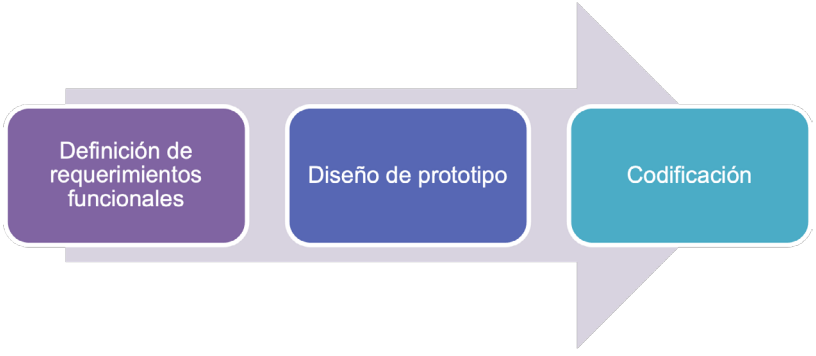


Ilustración 13. Proceso del desarrollo del prototipo.
Fuente: elaboración propia.

La presente metodología se enfoca en cuatro aspectos importantes como la fomentación del trabajo en equipo, el desarrollo de muestras o entregables, el uso de estándares para la codificación y la documentación.

4.4.1Definición de grupos

La idea de conformar grupos de desarrolladores surgió por la necesidad de optimizar recursos como el tiempo puesto que antes de su llegada, las metodologías de desarrollo tradicionales ocupaban más tiempo en llevar a cabo la elaboración del código y revisión del mismo.

Por lo general en las metodologías de desarrollo con enfoque ágil tienen preestablecido la conformación de parejas para llevar a cabo el despliegue de los módulos a construirse.

Modelo Móvil Sprint implementa el concepto de Pair Programming, que necesita de dos programadores se integren al trabajo de desarrollo en un lugar de trabajo. Los integrantes realizan operaciones distintas a lo largo del desarrollo, por ejemplo, cuando una unidad prueba el código, otro miembro considera las clases que cumplen con los requisitos de la prueba. El integrante que codifica o desarrolla recibe el nombre de controlador y el integrante al que se dirige se llama navegador. De forma general, es recomendable que los roles se intercambien al menos cada media hora o después de completar el testeo del módulo desarrollado.

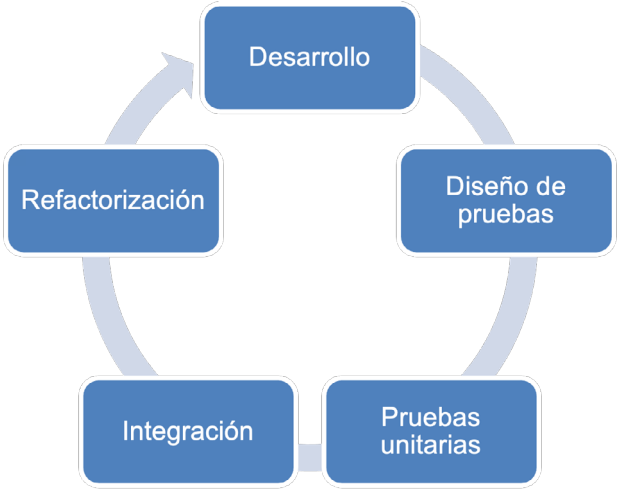


Ilustración 14. Actividades rotativas de la programación en pareja.
Fuente: elaboración propia.

4.4.2 Definición de entregables

Al final de cada interacción de la metodología se realiza la entrega de resultados provistos de los prototipos desarrollados en la fase anterior. Estos resultados que reciben el nombre de entregables deben ser evaluados antes de previamente a la reunión con el cliente.

Es recomendable seleccionar aquellos productos que se encuentren completos o lo más cercanos a satisfacer los requerimientos que fueron utilizados para desarrollar el prototipo.

4.4.3 Convenciones para el código

Los desarrolladores de software generalmente no hacen uso de algún tipo de sistematización a la hora de escribir códigos, por lo que eso puede ser un proceso altamente peligroso porque se expone a problemas como la dificultad de realizar actualizaciones debido a la espesa cantidad de bloques de código, tanto como el nombre de variables, funciones, procesos, librerías entre otras.

En esta metodología se plantea el uso de buenas prácticas a la hora de estructurar los bloques de código, el uso de nombres adecuados para las variables y funciones, la identación o tipo de sangría.

4.4.3.1 Nombre de variables y funciones

Los nombres de variables deben caracterizarse por ser determinantes, legibles y diferentes a los que usan las librerías con las que se encuentra trabajando. De esa manera fácilmente se evitan colisiones con otros nombres que son propios de frameworks y/o librerías.

Para determinar el nombre de las variables se plantea la siguiente sistematización:

Tabla 20. Buenas prácticas para variables y funciones.

	RECOMENDACIÓN	EJEMPLO
Numero de caracteres máximo	20	*String nombre;*
Signo de agrupación de nombre en caso de variables	“_”	*String nombre_cliente;*

Agrupación de nombres para funciones	A partir de la segunda palabra, la primera letra debe ser en mayúscula	*guardarCliente();*
Variables recomendadas para las iteraciones en las estructuras iterativas	"i", "j", "k" y "n"	*for(i=0;i<5;++i){}*
Para las funciones usadas constantemente	Se debe escribir todo en mayúsculas, en caso de usar más de una palabra en su descripción se debe usar el signo "_" para la separación.	*REFRESCAR_PAGINA();*

Fuente: elaboración propia.

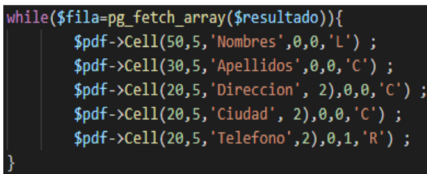
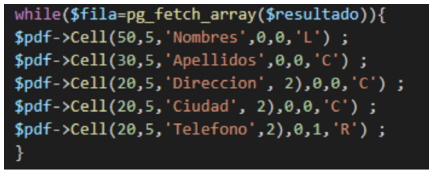
4.4.3.2 Identación

Los lenguajes de programación como java, C, C++, entre otros, utilizan signos como el ";" para indicar al depurador hasta dónde llega una línea de código. Sin embargo, actualmente con el auge del desarrollo de aplicaciones móviles y web se emplean lenguajes de programación como Python que tiene un repertorio muy extenso de librerías y funciones para procesar datos y realizar operaciones complejas. En este caso, Python es un lenguaje de programación que no emplea el ";" para indicar el final de una línea de código. En este caso utiliza lo que se conoce como identación.

Usar correctamente la sangría posibilita que los bloques de código extensos puedan ser legibles estructuralmente sin importar el lenguaje que se encuentre utilizando. Existen muchos beneficios al usar pestañas de 8 espacios para la sangría. La identación está claramente marcada para facilitar la lectura del código. Al obligar a la función a dividirse en bloques más modulares y bien definidos, también ayuda a mantener limpio el código; si la sangría excede el margen correcto, significa que la función está mal diseñada y debe dividirse para hacerla más modular u otro.

Las pestañas con sangría de 8 espacios también ayudan a que las funciones de diseño quepan en la pantalla, lo que significa que las personas pueden comprender el código sin tener que desplazarse hacia adelante y hacia atrás para comprender el código. A continuación, se plantean unos ejemplos de recomendaciones sobre la identación.

Tabla 21. Recomendaciones para la identación de código.

RECOMENDADO	NO RECOMENDADO
	
	

```
while($fila=pg_fetch_array($resultado)){
        $pdf->Cell(50,5,'Nombres',0,0,'L') ;
        $pdf->Cell(30,5,'Apellidos',0,0,'C') ;
        $pdf->Cell(20,5,'Direccion', 2),0,0,'C') ;
        $pdf->Cell(20,5,'Ciudad', 2),0,0,'C') ;
        $pdf->Cell(20,5,'Telefono',2),0,1,'R') ;
}
```

```
while($fila=pg_fetch_array($resultado)){
$pdf->Cell(50,5,'Nombres',0,0,'L') ;
$pdf->Cell(30,5,'Apellidos',0,0,'C') ;
$pdf->Cell(20,5,'Direccion', 2),0,0,'C') ;
$pdf->Cell(20,5,'Ciudad', 2),0,0,'C') ;
$pdf->Cell(20,5,'Telefono',2),0,1,'R') ;
}
```

```
while($fila=pg_fetch_array($resultado)){
        if($existe==0){
                $this->Cell(140,5) ;
                $this->Cell(120,5) ;
                $this->Cell(20,5) ;
                $this->Cell(120,5) ;
                $this->Cell(20,5) ;
                $this->Cell(120,5) ;
                $this->Cell(20,5) ;

                $existe=1;
        }else{
            $existe=1;
        }
```

```
while($fila=pg_fetch_array($resultado)){
if($existe==0){
$this->Cell(140,5) ;
$this->Cell(120,5) ;
$this->Cell(20,5) ;
$this->Cell(120,5) ;
$this->Cell(20,5) ;
$this->Cell(120,5) ;
$this->Cell(20,5) ;

$existe=1;
}else{
$existe=1;
}
```

Fuente: elaboración propia.

4.4.4 Documentación

Para la documentación se pretende hacer una serie de sugerencias al momento de escribir el programa, el uso de estas consideraciones puede ayudar a la claridad del código desarrollado. Cada programador tiene sus propios estándares para los mejores puntos para colocar la documentación en el código fuente y la mejor manera de implementar la documentación.

El estilo correcto del documento depende del lenguaje de programación y las prácticas de la entidad para la que se va a desarrollar el software y quién lo mantiene. Para realizar la documentación del código se consideran lo atributos descritos en la siguiente tabla.

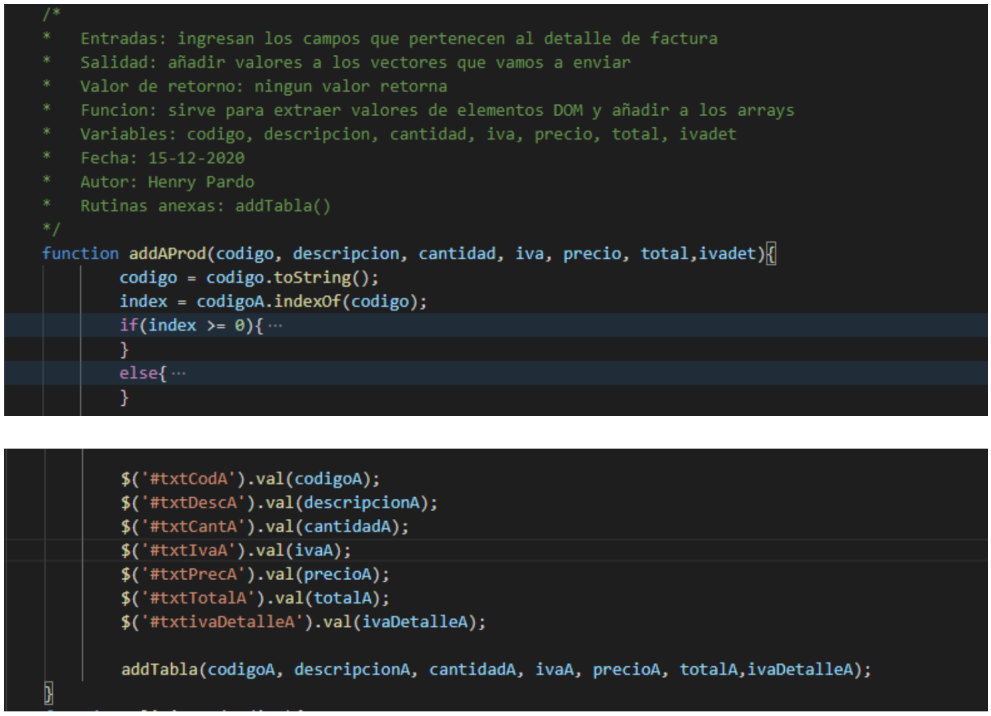
Tabla 22. Parámetros para describir en la documentación.

PARÁMETRO	DESCRIPCIÓN
Entradas	Listado de variables y su definición dentro de la función.
Salidas	Resultado que se obtiene de funciones.

Valor de retorno	Define los valores que son retornados por la función y que significan.
Función	Explique para que sirve dicha función.
Variables	Define el tipo y nombre de variables que son utilizadas en distintos contextos.
Fecha	Indica la fecha en la que se realiza la observación del código
Autor	Indica al responsable del comentario
Rutinas anexas	Describe su relación con otras rutinas (en caso de ser un caso de funciones anidadas)

Fuente: elaboración propia.

A continuación, se realiza un ejemplo en donde se identifica cada uno de los parámetros:



```
/*
*   Entradas: ingresan los campos que pertenecen al detalle de factura
*   Salidad: añadir valores a los vectores que vamos a enviar
*   Valor de retorno: ningun valor retorna
*   Funcion: sirve para extraer valores de elementos DOM y añadir a los arrays
*   Variables: codigo, descripcion, cantidad, iva, precio, total, ivadet
*   Fecha: 15-12-2020
*   Autor: Henry Pardo
*   Rutinas anexas: addTabla()
*/
function addAProd(codigo, descripcion, cantidad, iva, precio, total,ivadet){
        codigo = codigo.toString();
        index = codigoA.indexOf(codigo);
        if(index >= 0){···
        }
        else{···
        }
```

```
        $('#txtCodA').val(codigoA);
        $('#txtDescA').val(descripcionA);
        $('#txtCantA').val(cantidadA);
        $('#txtIvaA').val(ivaA);
        $('#txtPrecA').val(precioA);
        $('#txtTotalA').val(totalA);
        $('#txtivaDetalleA').val(ivaDetalleA);

        addTabla(codigoA, descripcionA, cantidadA, ivaA, precioA, totalA,ivaDetalleA);
}
```

Ilustración 15. Ejemplo de documentación.

Fuente: elaboración propia.

4.5 Refactorización

La producción de software está ligada a un proceso estructurado de desarrollo donde se analizan diferentes aspectos para la elaboración de productos con altos índices

de calidad. Los resultados de ejecutar protocolos durante el ciclo de desarrollo de software se ven cristalizados cuando el programa o sistema elaborado es puesto a prueba en la etapa de testeo y satisface a los requerimientos funcionales.

La refactorización consiste en optimizar los bloques de código a partir de su estructura sin que se comprometa su funcionamiento. Modelo Móvil Sprint se enfoca en evitar errores comunes durante la etapa de producción del código. Entre los problemas más comunes esta la duplicidad del código, métodos extensos, el exceso de parámetros que necesitan ciertas funciones, etc. Por ello se plantean dos buenas prácticas para la implementación de la refactorización:

4.5.1 Comprobar la adaptabilidad

Si la estructura de la aplicación no es suficiente para introducir los cambios necesarios, primero refactorice el código para que los cambios sean fáciles de introducir y luego agregue nuevas funciones.

Esta consideración pretende prever el surgimiento de errores durante la etapa de mejoramiento del código, puesto que empezar a modificar sin antes evaluar su adaptabilidad podría generar retrasos en el desarrollo de los entregables.

Sin embargo, al final de aplicar este consejo se obtendrán resultados favorables en aspectos como:

- Mantenibilidad: Cuando se desee realizar un mantenimiento sobre el sistema, este será sencillo de realizarlo por la simplicidad de un código limpio y navegable.
- Nivel bajo de complejidad: Los niveles de complejidad para resolver el problema o necesidad es relativamente sencillo y fácil de realizar optimizaciones.
- Comprensibilidad: El código fuente es más fácil de entender.

4.5.2 Omitir el uso de controles y sentencias anidadas

Evitar la implementación de sentencias anidadas y controles que generan la extensión innecesaria del código. Se debe buscar alternativas que permitan reducir las líneas de código como es el caso de la sentencia de control switch. Utilizar este tipo de sentencias obliga a plantear un menú extenso y difícil de reducir, para ello se puede reemplazar por otras estructuras o funciones que desempeñen las mismas actividades.

```
function getPokemon(type) {
    let pokemon;
    switch (type) {
        case 'Water':
        pokemon = 'Squirtle';
        break;
        case 'Fire':
        pokemon = 'Charmander
        break;
        case 'Plant':
        pokemon = 'Bulbasur';
        break;
        case 'Electric':
        pokemon = 'Pikachu';
        break;
        default:
        pokemon = 'Mew';
    }
    return pokemon;
}
```

Ilustración 16. Ejemplo de función switch.

Fuente: elaboración propia.

```
const pokemon = new Map([
    ['Water', 'Squirtle'],
    ['Fire', 'Charmander'],
    ['Plant', 'Bulbasur'],
    ['Electric', 'Pikachu']
]);

function getPokemon(type) {
return pokemon.get(type) || 'Mew';
}

console.log(getPokemon('Fire'));
console.log(getPokemon('unknown'));
```

Ilustración 17. Optimización de estructura de control.

Fuente: elaboración propia.

CAPITULO V: FASE DE PRUEBAS

La fase de pruebas corresponde al análisis de objetivos cumplidos y detección de errores durante el desarrollo de una aplicación móvil bajo la metodología MD-Sprint.

Las pruebas a realizar buscan identificar el cumplimiento de las funciones de la aplicación y estándares de calidad.

5.1 Definición

Las pruebas se realizan después del cumplimiento de cada módulo, y se iteran en base a errores y al incumplimiento del objetivo de cada módulo. El tester a cargo de las pruebas registra los puntos y funciones que la sección evaluada cumple y establece comentarios sobre las evidencias encontradas en las pruebas realizadas.

Cada prueba realizada supone la documentación pertinente, además de repetirse tras cada sprint para validar el funcionamiento de cada módulo.

5.2 Objetivos

El desarrollo de pruebas sobre el software móvil busca cumplir los siguientes objetivos:

- Detección de bugs y errores.
- Identificación de fallas de funcionamiento.
- Corrección de errores identificados.
- Identificar posibles mejoras sobre módulos funcionales.

5.3 Características

Las pruebas sobre el software móvil desarrollado se caracterizan por:

- Identificación de errores actuales y potenciales.
- Verificación de funcionalidad optima en módulos desarrollados.
- Documentación detallada de pruebas ejecutadas.

5.4 Actividades

Las actividades de la fase de pruebas se basan en la realización de pruebas sobre los módulos de la aplicación móvil y la ejecución del control de cambios en base a las evidencias obtenidas al realizar las pruebas; tanto para corregir errores detectados o modificar elementos que optimicen el funcionamiento de la aplicación.

Las pruebas realizadas se registran en una sola bitácora, enumeradas en el orden de ejecución e identificando que tipo de prueba se realizó y la fecha de ejecución:

Tabla 23. Plantilla de bitácora de pruebas.

BITÁCORA DE PRUEBAS			
N°	Fecha de ejecución	Responsable	Tipo de prueba

Fuente: elaboración propia.

Donde:

- N°: Registra el número de pruebas ejecutadas, la numeración empieza en 1.
- Fecha de ejecución: fecha en formato dd/mm/aaaa donde se realizó la prueba.
- Responsable: se identifica el nombre del tester a cargo de la ejecución de la prueba.
- Tipo de prueba: Describe que prueba se ejecutó en el periodo anteriormente especificado.

Las actividades que se registran en la bitácora son las diferentes pruebas de sistema y control de cambios aplicados sobre el proyecto.

5.5 Pruebas de sistema

Las pruebas de sistema se realizan con el fin de comprobar el óptimo funcionamiento de la aplicación móvil, estas se realizan por un personal o tester ajeno al proceso de desarrollo.

Con la finalidad de abarcar la mayor cantidad de aspectos posibles, las pruebas de sistema se dividen en:

- Pruebas de calidad.
- Pruebas de tendencia.
- Pruebas de usabilidad.
- Pruebas de accesibilidad.
- Pruebas de interfaz de usuario.

Para la evaluación de sistema se emplea una matriz de indicadores que se utilizaran en las diferentes pruebas.

Tabla 24. Indicadores de evaluación para pruebas de sistema.

INDICADORES DE EVALUACIÓN	
Valorización	Interpretación del resultado
1	Excelente
2	Bueno
3	Regular
4	Malo
5	Pésimo

Fuente: elaboración propia.

5.5.1 Pruebas de calidad

"La calidad del producto software se puede interpretar como el grado en que dicho producto satisface los requisitos de sus usuarios aportando de esta manera un valor." (ISO 25000, 2021).

La evaluación se realiza en base a los puntos más importantes obtenido de la normativa ISO 25010 sobre calidad de productos de software.

Tabla 25. Matriz de evaluación de calidad.

EVALUACIÓN DE CALIDAD						
Características	Sub características	1	2	3	4	5
Funcionalidad						
Eficiencia						
Compatibilidad						
Fiabilidad						
Seguridad						
Portabilidad						

Fuente: elaboración propia.

Donde:

- Funcionalidad: Indica la capacidad del producto de software para proporcionar funciones que cumplen con los requisitos especificados e implícitos cuando el producto de software se utiliza en condiciones específicas.
- Eficiencia: Esta característica representa el rendimiento en relación con la cantidad de recursos utilizados en determinadas condiciones.
- Compatibilidad: Cuando dos o más sistemas o componentes comparten el mismo entorno de hardware o software, su capacidad para intercambiar información y / o realizar sus funciones requeridas.
- Fiabilidad: La capacidad del sistema o componente para realizar funciones específicas cuando se usa bajo ciertas condiciones y períodos de tiempo.
- Seguridad: La capacidad de proteger la información y los datos para que personas o sistemas no autorizados no puedan leerlos o modificarlos.
- Portabilidad: La capacidad de un producto o componente para transferir eficientemente de un hardware, software, entorno operativo o de uso a otro.

Las sub características de cada criterio dependen del alcance del proyecto de software.

5.5.2 Pruebas de tendencia

Un estudio por parte de Dwivedi (2016) reveló que en el 2016 las tendencias de desarrollo móvil apuntaban al uso de tecnologías portátiles que unifican y aprovechan servicios de realidad virtual, realidad aumentada, etc.

La aplicación de tendencias tecnológicas depende del enfoque que se quiera dar a la aplicación, por lo que no se puede apuntar al uso de cualquier tendencia,

La evaluación de tendencias ayuda en el desarrollo a elegir cual tendencia es más compatible con las funciones de la aplicación móvil en desarrollo, o elegir entre las más optimas en caso de existir más de una.

Tabla 26. Matriz de evaluación de tendencia.

EVALUACIÓN DE TENDENCIAS					
Tendencia	1	2	3	4	5

Fuente: elaboración propia.

En la matriz se especifica las tendencias posibles o aplicables al modelo de software que se está desarrollando y se evalúa con la matriz de indicadores de evaluación respecto a la satisfacción que propone cada tendencia sobre la solución que presenta la aplicación móvil.

5.5.3 Pruebas de usabilidad

Según lo define ISO 25000 (2021) la usabilidad es la capacidad del producto software para ser entendido, aprendido, usado y resultar atractivo para el usuario, cuando se usa bajo determinadas condiciones.

Esta prueba se realiza para medir la capacidad del software para ser entendido, aprendido, usado y resultar atractivo para el usuario, cuando se usa bajo determinadas condiciones.

Tabla 27. Matriz de evaluación de usabilidad

EVALUACIÓN DE USABILIDAD						
Características	Sub características	1	2	3	4	5
Inteligibilidad						
Aprendizaje						
Operabilidad						
Protección						
Estética						
Accesibilidad						

Fuente: elaboración propia.

Donde:

- Inteligibilidad: Las características del producto permiten a los usuarios comprender si el software es adecuado para sus necesidades.
- Aprendizaje: Permita que los usuarios conozcan las características del producto de sus aplicaciones.
- Operabilidad: Las características del producto permiten a los usuarios operarlo, utilizarlo o controlarlo fácilmente.
- Protección: La capacidad del sistema para proteger a los usuarios de cometer errores.
- Estética: La interfaz de usuario agrada y satisface la capacidad de interactuar con el usuario.
- Accesibilidad: Capacidad del producto, que permite su uso a usuarios con características y defectos específicos.

Las sub características de cada criterio dependen del alcance del proyecto de software.

5.5.4 Pruebas de mantenibilidad

La mantenibilidad según ISO 25000 (2021) representa la capacidad del producto software para ser modificado efectiva y eficientemente, debido a necesidades evolutivas, correctivas o perfectivas.

Tabla 28. Matriz de evaluación de mantenibilidad.

EVALUACIÓN DE MANTENIBILIDAD						
Características	Sub características	1	2	3	4	5
Modularidad						
Reusabilidad						
Analizabilidad						
Capacidad de ser modificado						
Capacidad de ser probado						

Fuente: elaboración propia.

Donde:

- Modularidad: Permite la realización de cambios en módulos específicos si afectar el funcionamiento de un módulo diferente.
- Reusabilidad: La capacidad del activo permite que se utilice en la construcción de múltiples sistemas de software u otros activos.
- Analizabilidad: Capacidad de analizar el impacto que produce cada cambio en el funcionamiento de la aplicación; diagnosticar deficiencias o identificar elementos modificables.
- Capacidad de ser modificado: Capacidad de modificar sin causar defectos o degradar el desempeño del módulo.
- Capacidad de ser probado: Capacidad para establecer criterios de evaluación y pruebas con las que se pueden determinar las funciones de cada módulo.

Las sub características de cada criterio dependen del alcance del proyecto de software.

5.5.5 Pruebas de interfaz de usuario

La interfaz de usuario es el principal medio de interacción entre las funciones de la aplicación móvil y el usuario final.

Las pruebas de interfaz permiten determinar la efectividad de su funcionamiento y

presentación, con el fin de proporcionar la mejor calidad a los usuarios finales.

Se evalúa mediante la escala proporcionada en los indicadores de evaluación previamente descritos con respecto a las características principales de las interfaces de usuario.

Tabla 29. Matriz de evaluación de interfaz de usuario.

EVALUACIÓN DE INTERFAZ DE USUARIO						
Características	Sub características	1	2	3	4	5
Adaptabilidad						
Legibilidad						
Concisión						
Coherencia						
Flexibilidad						
Atractivo visual						

Fuente: elaboración propia.

Donde:

- Adaptabilidad: Es la capacidad de la interfaz para ser adaptada al tamaño de pantalla o dispositivo sin distorsionar sus componentes o afectar su funcionamiento.
- Legibilidad: Capacidad de usar un lenguaje sencillo para ayudar a los usuarios a leer rápidamente.
- Concisión: Solo proporciona a los usuarios la información que necesitan y solicitan.
- Coherencia: Ayuda a los usuarios a desarrollar patrones de uso y, poco a poco, aprender las funciones de varios botones, iconos y otros elementos de la interfaz.
- Flexibilidad: Permite deshacer las operaciones incorrectas del usuario sin causar problemas importantes.
- Atractivo visual: Capacidad de la interfaz de ser agradable a la vista del usuario.

Las sub características de cada criterio dependen del alcance del proyecto de software.

5.6 Control integrado de cambios

El control de cambios se basa en la detección de errores. Una vez identificado un error en las pruebas realizadas se documentan para hacer efectivo el cambio necesario que permita solucionar el error detectado u optimizar el funcionamiento de un módulo en particular.

Cada vez que se solicite un cambio se registra en la siguiente matriz:

Tabla 30. Plantilla de control de cambios.

CONTROL DE CAMBIOS				
N°	Fecha	Solicitante	Módulo	Descripción del cambio

Fuente: elaboración propia.

Donde:

- N°: Registra el número de cambios solicitados, la numeración empieza en 1.
- Fecha: fecha en formato dd/mm/aaaa donde se solicita el cambio
- Solicitante: Nombre de la persona que solicita el cambio.
- Modulo: Modulo en el que se descubrió el error o la optimización requerida.
- Descripción del cambio: De talle de lo que se solicita modificar o corregir.

Además de la plantilla, cada vez que se realiza una solicitud de cambio, esta debe ser estipulada por escrito mediante un oficio que indique la solicitud bajo el siguiente formato:

<N°> (Número de solicitudes de cambio emitidas)

<Motivo> (Razón de la emisión de la solicitud de cambio)

<Solicitante> (Nombre y cargo del personal que solicita el cambio)

<Módulo> (Módulos de la aplicación donde se detectó el activo a modificar)

<Tipo de cambio> (Solución de error, actualización o eliminación o reinicio)

Descripción del fallo: Se describe cuáles son los efectos actuales que ejerce el error sobre la aplicación o el módulo en caso de una actualización, así como el origen del mismo.

Descripción del cambio: Se detalla que es lo que se busca modificar y bajo que métodos.

Justificación: Justifica la importancia de realizar el cambio o modificación a la brevedad.

Impacto sobre otros módulos: Se detalla cómo se verían afectados los otros módulos durante la ejecución de los cambios y después de haberse implementado.

Tiempo: Tiempo estimado en días de la duración del proceso de modificación o cambio.

Aprobación: Nombre, fecha y firma de la persona que aprueba la ejecución de los cambios.

Anexos: Evidencias adicionales que sustenten el desarrollo de cambios, así como descripción de propuestas de los cambios a realizar.

CAPÍTULO VI: FASE DE LANZAMIENTO

6.1 Definición

La fase del lanzamiento y producción está destinada al cumplimiento de actividades que permiten la preparación de la aplican móvil para su despliegue en una plataforma o sistema operativo específico.

En esta fase es importante la comprobación del funcionamiento de cada módulo de la aplicación aun después de su lanzamiento.

6.2 Objetivos

El lanzamiento y producción de una aplicación móvil busca cumplir los siguientes objetivos:

- Lanzamiento de la aplicación en un entorno no local.
- Pruebas de funcionamiento reales en la aplicación desplegada.
- Entrega de manuales y documentación correspondiente.

6.3 Características

La fase de lanzamiento y producción se caracteriza por:

- Identificación los elementos que se deben preparar previo al lanzamiento.
- Verificación de funcionamiento de funciones posterior al despliegue de la aplicación.
- Documentación detallada de manuales.

6.4 Actividades

Las actividades de la fase de lanzamiento y producción se basan en la preparación de los componentes y el entorno necesarios para el despliegue de la aplicación móvil. Dentro de las actividades a realizar esta la preparación del host, identificación del nombre de la aplicación y análisis de cumplimiento de requerimientos; documentación de requerimientos y los respectivos manuales de la aplicación, tanto para el usuario como a nivel de programación.

6.4.1 Preparación

La preparación involucra los aspectos que el equipo de desarrollo debe tomar en cuenta antes de la publicación o lanzamiento de la aplicación terminada.

Dentro de la preparación se consideran los siguientes aspectos:

- Host.
- Nombre de la aplicación.
- Análisis del cumplimiento de requerimientos.

6.4.1.1 Host

La palabra hosting significa alojamiento, según lo define Baeza (2019) es el contrato, en virtud del cual una empresa proveedora de servicios de internet, aloja o alberga la página web del cliente, destinando un espacio en un servidor a cambio de una remuneración.

Las aplicaciones móviles se alojan en diferentes stores de aplicaciones móviles dependiendo del lenguaje para el cual fueron desarrollados. Otra opción es alojarlos en un hosting web que permita la descarga independiente de la aplicación, sin necesidad de alojarla en un store de aplicaciones.

Existen hostings de paga y gratuitos, estos poseen características diferentes dependiendo del tipo. Las principales características que se debe tomar en cuenta cuando se comparan diferentes servicios de hosting antes de una decisión definitiva son:

- Capacidad de almacenamiento.
- Restricciones de horario.
- Certificado SSL.
- Velocidad de trasferencia de datos.
- Vinculación con el lenguaje de programación y base de datos.
- Valor mensual/anual.
- Soporte técnico.

Tras analizar todas las ventajas de las que dispone el servicio de hosting se elige el

más óptimo para las operaciones de la aplicación, además de realizar pruebas de funcionamiento después de su alojamiento para la comprobación optima de todas sus funciones en un entorno real y no solo de producción.

6.4.1.2 Nombre de la aplicación

A través del nombre de la aplicación, los usuarios identifican la marca con mayor facilidad; siendo así la primera impresión que presenta al usuario.

Para elegir un nombre apropiado en relación a las funciones de la aplicación se toman en cuenta las siguientes consideraciones

- Nombre corto: Un nombre breve, de no más de 25 a 30 caracteres ayuda a que el usuario lo memorice.
- Único y memorable: el nombre de la aplicación debe perdurar en la mente del usuario, apuntando a la función principal de la misma para que sea identificable entre la competencia.
- Palabras clave: Utilizar palabras que indique la función principal de la aplicación sobre alguna función adicional.
- Idioma: las plataformas para aplicaciones móviles presentan opciones para nombres regionales, por lo que se puede llegar a un mayor alcance con el idioma adecuado aplicado.

6.4.2 Documentación de requerimientos

En la documentación de requerimientos de software se va a especificar la descripción, características y requisitos de la aplicación móvil

Al tratarse de una aplicación móvil desarrollada bajo el pedido de un cliente especifico, en esta documentación se detallan todas las características y requisitos que el cliente desea.

Para el desarrollo de la documentación de requerimientos se sugiere la siguiente plantilla:

Historial de versiones:

Tabla 31. Matriz de historial de versiones.

CONTROL DE CAMBIOS			
Versión	Fecha	Autor	Descripción

Fuente: elaboración propia.

Este historial se va modificando después de las reuniones con el cliente, en caso de que este solicite un cambio.

Funcionalidades del proyecto:

Tipos de usuarios: En esta sección se detalla que tipo de usuarios son los objetivos a utilizar la aplicación. Esta información permite identificar el público objetivo de la aplicación y dejarlo por escrito.

Entorno: Describe para que plataforma o sistema operativo se va a desplegar la aplicación móvil.

Requerimientos funcionales: Dentro de esta sección se listan los requerimientos funcionales de la aplicación y se describe cada uno bajo la siguiente lista:

- Descripción.
- Prioridad.
- Comportamiento esperado.

Requerimiento de interfaces: ¿En este punto se lista las diferentes pantallas que va a tener la aplicación, y en cada una se detalla los siguientes elementos:

- Descripción.
- Comportamiento esperado.
- Bocetos.

Requerimientos no funcionales: Se listan cuáles son los requerimientos no

funcionales que se espera cumpla la aplicación una vez se despliegue.

6.4.2.1 Análisis del cumplimiento de requerimientos

En el análisis de cumplimiento de requerimientos se establece el nivel de satisfacción que presenta la aplicación con respecto a sus requerimientos funcionales y n funcionales.

6.4.3 Manual de usuario

El manual de usuario corresponde a una guía que especifica el funcionamiento y alcance de cada módulo y operación que ejecuta la operación. Debe estar escrito en términos no técnicos que puedan ser entendibles por los usuarios en general y detallar cada elemento de la interfaz y su funcionamiento.

No se define un formato específico para la elaboración del manual de usuario, siempre y cuando cubra el funcionamiento de todos los módulos que presenta y haga muestra del alcance y operaciones que puede realizar.

6.4.3.1 Portada y titulo

Son una representación gráfica y llamativa del contendió del manual donde se indica el nombre del manual el producto los autores y la editorial.

6.4.3.2 Derechos de autor

En esta página se describe la tutoría de los derechos de autor a la que está sujeto el manual; puede tratarse de una persona natural o una persona jurídica; en todo caso debe dejarse claro cuál es el autor intelectual y el tutor de los derechos de autor, puesto que no necesariamente debe tratarse de la misma persona.

La forma en la que se declara la tutoría de derechos de autor depende de las regulaciones de cada país.

6.4.3.3 Prefacio

En esta sección se describen los detalles de los documentos relacionados y la información sobre cómo navegar por el manual de usuario. Incluye el índice de contenidos, tablas e imágenes.

6.4.3.4 Introducción

Aquí se incluye una breve descripción del producto de software, como funciona y su objetivo. Si se trata de una versión actualizada se incluye las novedades de la nueva versión y/o las correcciones que se hayan aplicado.

6.4.3.5 Requerimientos

En el análisis de cumplimiento de requerimientos se establece el nivel de satisfacción que presenta la aplicación con respecto a sus requerimientos funcionales y n funcionales.

Se trata d ellos requisitos previos para el manejo de la aplicación, aquí se define:

- Conocimiento mínimo que debe tener el usuario.
- Capacidades mínimas del equipo donde se verá instalado.
- Softwares asociados requeridos (en caso de ser necesario).

6.4.3.6 Contenido

El contenido del manual de usuario se basa completamente en la presentación detallada de las funciones de la aplicación móvil. En esta sección se sugiere dividir en los siguientes puntos:

- Instalación y configuración.
- Descripción de las funciones de cada módulo (Se procura describir el objetivo de cada módulo y las funciones que este desempeña).
- Solucionador de problemas (Se plantean algunos d ellos posibles problemas que el usuario puede encontrar cuando manipula la aplicación y se detalla como corregirlos).
- Preguntas Frecuentes (Incluye las principales dudas de los usuarios, información de contacto y ayuda).
- Glosario de términos (A pesar de que se busca escribir el manual de la manera menos técnica posible, se da el caso de manejar términos que pueden ser confusos y/o ambiguos, por lo que en la sección de glosario se describe el significado de esas palabras en el contexto de la aplicación).

6.4.4 Manual de programador

El manual de programador establece la documentación del código desarrollado por el equipo de desarrollo de la aplicación para su uso en futuras actualizaciones o revisiones por parte de un equipo externo o un equipo de reemplazo al de desarrollo original.

Para un desarrollo guiado del manual de programador se sugiere cumplir los siguientes elementos:

6.4.4.1 Portada y titulo

Son una representación gráfica y llamativa del contendió del manual donde se indica el nombre del manual el producto los autores y la editorial.

6.4.4.2 Prefacio

En esta sección se describen los detalles de los documentos relacionados y la información sobre cómo navegar por el manual de usuario. Incluye el índice de contenidos, tablas e imágenes.

6.4.4.3 Introducción

Aquí se incluye una breve descripción del producto de software, como funciona y su objetivo. Si se trata de una versión actualizada se incluye las novedades de la nueva versión y/o las correcciones que se hayan aplicado.

6.4.4.4 Requerimientos

En la sección de requerimientos, a diferencia de los requerimientos del manual de usuario, aquí se detalla información técnica acerca del desarrollo, como para que sistema fue desarrollado, lenguaje de progresión utilizado, peso, etc.

6.4.4.5 Documentación de código

Para cada módulo y función se describe su propósito, funcionamiento y la interacción que hace con otros módulos o funciones de la aplicación.

6.4.4.6 Bocetos y diseños

Se describe el bocetado de cada módulo y pantalla, haciendo énfasis en las funciones que cumple cada botón o panel y con qué código interactúan.

REFERENCIAS BIBLIOGRÁFICAS

Antúnez, T. A., Valdovinos, R. M., Marcial, J. R., Ramos, M. A., y Herrera, E. (2016). Estimación de costos de desarrollo, caso de estudio: Sistema de Gestión de Calidad del Reactor TRIGA Mark III. *Revista Cubana de Ciencias Informáticas, 10*(1), 215-228. http://scielo.sld.cu/scielo.php?script=sci_abstract&pid=S2227-18992016000100018&lng=es&nrm=iso&tlng=es

Baeza, F. (2019). *El contrato de hosting*. Facultad de Derecho Universidad de La Laguna. https://riull.ull.es/xmlui/bitstream/handle/915/16402/El%20contrato%20de%20hosting.pdf?sequence=1

Bolshakova, V., Guerriero , A., y Halin, G. (2020, 01 de marzo). Identifying stakeholders' roles and relevant project documents for 4D-based collaborative decision making. *Frontiers of Engineering Management*, 7(1), 104-118. https://doi.org 10.1007/s42524-019-0041-4

Davalbhakta, S., Advani, S., Kumar, S., y Agarwal, V. (2020). A Systematic Review of Smartphone Applications Available for Corona Virus Disease 2019 (COVID19) and the Assessment of their Quality Using the Mobile Application Rating Scale (MARS). *Journal of Medical Systems, 44*(9), 164. https://doi.org/10.1007/s10916-020-01633-3

Duh, E., Guna, J., Pogačnik, M., y Sodnik, J. (2016). Applications of Paper and Interactive Prototypes in Designing Telecare Services for Older Adults. *Journal of Medical Systems, 40*(4), 92. https://doi.org/10.1007/s10916-016-0463-z

Dwivedi, A. (2016). Top 5 mobile development trends that take centre stage in 2016. *Mobile Information Systems.*

Gao, H., y Yin, Y. (2020, 01 de abril). Editorial: ACM/Springer Mobile Networks & Applications - Special Issue on Mobile Computing and Software Engineering. *Mobile Networks and Applications, 25*(2), 672-673. https://doi.org/10.1007/s11036-019-01451-z

ISO 25000. (2021, 18 de enero). *ISO/IEC 25010*. https://iso25000.com/index.php/normas-iso-25000/iso-25010

ISO 25000. (2021, 18 de enero). *Mantenibilidad.* https://iso25000.com/index.php/normas-iso-25000/iso-25010/26-mantenibilidad

ISO 25000. (2021, 18 de enero). *Usabilidad.* https://iso25000.com/index.php/

normas-iso-25000/iso-25010/23-usabilidad

Li, X., Qiao, H., y Wang, S. (2017, 01 de mayo). Exploring evolution and emerging trends in business model study: a co-citation analysis. *Scientometrics, 111*(2), 869-887. https://doi.org/10.1007/s11192-017-2266-5

Maia, V., Gonçalves, G., y da Rocha. (2019). Quality Characteristics of Mobile Applications: A Survey in Brazilian Context. *ACM Digital Library,* 109–118. https://doi.org/10.1145/3364641.3364654

Mariño, S. I., y Alfonzo, P. L. (2014). Implementación de SCRUM en el diseño del proyecto del Trabajo Final de Aplicación. *Scientia Et Technica, 19*(4), 412-418. https://www.redalyc.org/pdf/849/84933912009.pdf

Molina, J., Honores, J., y Zea, M. (2015). *Nociones de Ingeniería de Software.* UTMACH. http://repositorio.utmachala.edu.ec/handle/48000/6919

Molina, J., Zea, M., Redrován, F., Loja, N., y Honores, J. (2018). *SNAIL, Una metodología híbrida para el desarrollo de aplicaciones web.* Editorial Científica 3Ciencias. https://www.3ciencias.com/wp-content/uploads/2018/05/Metodolog%C3%ADa-Hibrida-SNAIL.pdf

Palomo, R. G. (2020). *Aproximación a la ingeniería del software.* Editorial Centro de Estudios Ramon Areces.

PMBOK. (s.f.). *PMBOK® Guide.* https://www.pmi.org/pmbok-guide-standards/foundational/pmbok

Riquelme, M. (2016). FODA: Matriz o Análisis FODA – Una herramienta esencial para el estudio de la empresa. http://148.202.167.116:8080/jspui/handle/123456789/3206

Zea, M. P., Molina, J. R., y Redrován, F. F. (2017). *Administración de bases de datos con POSTGRESQL*. Editorial Científica 3Ciencias. https://www.3ciencias.com/wp-content/uploads/2017/04/Administraci%C3%B3n-bases-de-datos.pdf

Zúñiga-Vásquez, J. M. (2017). Spatial modeling of forest fires in Mexico: an integration of two data sources. *Bosque (Valdivia),* 563-574. https://scielo.conicyt.cl/scielo.php?script=sci_abstract&pid=S0717-92002017000300014&lng=e&nrm=iso

Ingeniería y Tecnología

3ciencias