UNIVERSIDAD MAYOR DE SAN ANDRÉS

FACULTAD DE CIENCIAS PURAS Y NATURALES

CARRERA DE INFORMÁTICA

PROYECTO DE GRADO

"PLATAFORMA DE PAGOS DE SERVICIOS A TRAVÉS DE IMÁGENES QR"

CASO: "CONSULTORIA Y DESARROLLO TECNOLOGICO MC4 S.R.L."

PARA OPTAR AL TÍTULO DE LICENCIATURA EN INFORMÁTICA

MENCIÓN: INGENIERÍA DE SISTEMAS INFORMÁTICOS

POSTULANTE: JONATHAN VALDIVIA RAMOS

TUTOR METODOLÓGICO: M.Sc. GROVER ALEX RODRIGUEZ RAMIREZ

ASESOR: M.Sc. ALDO RAMIRO VALDEZ ALVARADO

LA PAZ – BOLIVIA

2020

UNIVERSIDAD MAYOR DE SAN ANDRÉS
FACULTAD DE CIENCIAS PURAS Y NATURALES
CARRERA DE INFORMÁTICA

LA CARRERA DE INFORMÁTICA DE LA FACULTAD DE CIENCIAS PURAS Y NATURALES PERTENECIENTE A LA UNIVERSIDAD MAYOR DE SAN ANDRÉS AUTORIZA EL USO DE LA INFORMACIÓN CONTENIDA EN ESTE DOCUMENTO SI LOS PROPÓSITOS SON ESTRICTAMENTE ACADÉMICOS.

LICENCIA DE USO

El usuario está autorizado a:

a) visualizar el documento mediante el uso de un ordenador o dispositivo móvil.
b) copiar, almacenar o imprimir si ha de ser de uso exclusivamente personal y privado.
c) copiar textualmente parte(s) de su contenido mencionando la fuente y/o haciendo la referencia correspondiente respetando normas de redacción e investigación.

El usuario no puede publicar, distribuir o realizar emisión o exhibición alguna de este material, sin la autorización correspondiente.

DEDICATORIA

A Dios,

a mi madre, que espero esto le saque esa sonrisa que vale un millón,

a mi padre, “desearía que estés aquí”

y al amor de mi vida, Melany, siempre fuiste tú.

AGRADECIMIENTO

A Dios por guiar mi camino, por acompañarme en todo momento, brindarme salud y permitirme llegar hasta este punto de la culminación de una de mis metas.

A mi tutor metodológico M.Sc. Grover Alex Rodríguez Ramírez, quien me orientó y sugirió ideas, gracias por confiar en mi trabajo, dedicarme el tiempo necesario para la culminación del proyecto.

A mi asesor M.Sc. Aldo Ramiro Valdez Alvarado, quien me guio a lo largo de todo este tiempo y me transmitió mucho conocimiento y valores que no olvidaré, gracias por dedicarme tiempo para la culminación del proyecto.

A la empresa MC4 S.R.L, por toda la confianza y apoyo.

INDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

El presente documento contiene el proceso de desarrollo de la “Plataforma de pagos de servicios a través de imágenes QR” denominada SIP, dicha plataforma se desarrolló bajo los requerimientos solicitados por la empresa Consultoría y Desarrollo Tecnológico MC4 S.R.L.

La Plataforma de pagos de servicios a través de imágenes QR constituye la integración de dos de los sistemas más importantes desarrollados por la empresa MC4, la plataforma de pagos UNICO y el sistema de transferencias interbancarias CLIENTE ACH EXPRESS, para de esta forma crear un nuevo producto propio que tiene como aliado a las entidades financieras y como cliente a pequeñas, medianas y grandes empresas o instituciones.

En el desarrollo del sistema web se hace una combinación entre la metodología ágil Scrum y la metodología UWE, esta combinación es un aporte para futuros proyectos.

Scrum es un proceso en el que se aplican de manera regular un conjunto de buenas prácticas para trabajar colaborativamente, en equipo, y obtener el mejor resultado posible de un proyecto. Estas prácticas se apoyan unas a otras y su selección tiene origen en un estudio de la manera de trabajar de equipos altamente productivos.

UWE detalla el proceso de las aplicaciones, cuenta con cinco modelos que son el análisis de requerimientos, modelo de contenidos, modelo de navegación, modelo de presentación y modelo de procesos.

Para la calidad del sistema web se utiliza la norma ISO 9126 y para la seguridad del sistema se realiza por niveles, tanto del lado del cliente como del lado del servidor.

Palabras clave: Metodología ágil Scrum, UWE, ISO 9126, Sistema Web, COCOMO II

ABSTRACT

This document contains the development process of the "Platform for payment of services through QR images" called SIP, this platform was developed under the requirements requested by the company Consultoría y Desarrollo Tecnológico MC4 S.R.L.

The Platform for payment of services through QR images constitutes the integration of two of the most important systems developed by that company MC4, the UNICO payment platform and the CLIENTE ACH EXPRESS interbank transfer system, in order to create a new own product that has a financial institutions as an ally and a small, medium and large companies or institution as a client.

In the development of the web system, a combination is made between the agile Scrum methodology and the UWE methodology, this combination is a contribution for future projects.

Scrum is a process in which a set of good practices are applied regularly to work collaboratively, as a team, and obtain the best possible result from a project. These practices support each other and their selection originates from a study of the way of working of highly productive teams.

UWE details the application process; it has five models that are the requirements analysis, content model, navigational model, presentation model and process model.

For the quality of the web system, the ISO 9126 standard is used and for the security of the system, it is carried out by levels, both on the client side and on the server side.

Keywords: Scrum agile methodology, UWE, ISO 9126, Web System, COCOMO II

CAPÍTULO I

MARCO REFERENCIAL

1.1. INTRODUCCIÓN

La evolución de las nuevas tecnologías crece a pasos agigantados lo que ocasiona que las empresas tengan que renovar constantemente sus herramientas si quieren seguir creciendo y que sus servicios no se queden obsoletos, ésta revolución digital también ha modificado los hábitos de pago y consumo de los ciudadanos, la forma en que se relacionan con las empresas buscando métodos de pago rápidos, cómodos y seguros. (Saunders, 2020)

Los pagos electrónicos mediante una banca electrónica es una nueva herramienta comunicativa y transaccional que permite al consumidor financiero realizar acciones de manera virtual y electrónica desde un lugar que no necesariamente es un punto de atención físico. La tecnología permite ofrecer a los clientes servicios y productos demandados en el contexto y canal adecuados, llamado la personalización de servicios. (Banco Central de Bolivia, 2019)

De los antecedentes antes mencionados, este proyecto surge ante la necesidad de ofrecer un canal de pagos que permita integrar el crecimiento de la banca en línea con un sistema de pagos de servicios.

La Empresa Consultoría y Desarrollo Tecnológico MC4 S.R.L. es una compañía dedicada al desarrollo de software empresarial, que cuenta con productos propios (UNICO, Cliente Multicámara ACH[1], Cliente ACH Express, Sistema de Facturación CLIC, entre otros) que operan en las principales entidades financieras y no financieras del país, a su vez se conforma como una fábrica de Software. (Consultoría y Desarrollo Tecnológico MC4 S.R.L., 2017)

El presente proyecto de grado tiene como propósito el desarrollo de la plataforma de pago de servicios mediante imágenes QR denominada SIP, la cual es una plataforma de pago empresarial

[1] La ACH es un servicio de Compensación Automático de Transferencias Electrónicas de Fondos, mediante Órdenes de Pago y Órdenes de Cargo, al que están interconectadas las Entidades Financieras autorizadas por la Autoridad de Supervisión del Sistema Financiero (ASFI) y el Banco Central de Bolivia (BCB).

que integra dos de los productos propios de la empresa MC4 como ser la solución de pagos QR[2] Simple (Cliente ACH EXPRESS) con la plataforma UNICO de cobranza, para que de esta forma se puedan generar imágenes QR Simple de deudas específicas para pago desde cualquier aplicación móvil de los 14 bancos que están integrados con la cámara de compensación ACCL (Administradora de Cámaras de Compensación y Liquidación S.A.).

1.2. PROBLEMA

1.2.1. ANTECEDENTES

1.2.1.1. ANTECEDENTES INSTITUCIONALES

La empresa MC4 S.R.L. se caracteriza por brindar un servicio personalizado y de calidad, atendiendo todas las inquietudes, comprendiendo y comprometiéndose con las necesidades de sus clientes, creando e innovando soluciones tecnológicas siempre haciendo el uso de su lema "Digitalizando con Humanidad". (Consultoría y Desarrollo Tecnológico MC4 S.R.L., 2017)

Su Misión es construir software simple y confiable, comprendiendo y comprometiéndose con las necesidades de las personas y empresas, creando e innovando junto a su equipo. (Consultoría y Desarrollo Tecnológico MC4 S.R.L., 2017)

Su Visión es ser una empresa de software líder digitalizando con humanidad. (Consultoría y Desarrollo Tecnológico MC4 S.R.L., 2017)

La empresa Consultoría y Desarrollo Tecnológico MC4 S.R.L., con fecha de registro en Fundempresa del 27 de julio del año 2011, número de matrícula 00180751 e inscrita en el servicio nacional de impuestos (SIN) con número de identificación tributaria (NIT) 184138022 con domicilio fiscal en la ciudad de La Paz ubicado en Av. Ecuador, Nro. 2523, zona Sopocachi y la ciudad de Santa Cruz en la Av. Banzer entre 2do y 3er Anillo, Barrio Marabol, Calle Medardo Chávez No. 135 entre calle Gustavo Parada y Calle Medardo Chávez (paralela a Av. Banzer detrás del Surtidor Rivero). (Consultoría y Desarrollo Tecnológico MC4 S.R.L., 2017)

[2] Un código QR es una matriz en dos dimensiones formada por una serie de cuadrados negros sobre un fondo blanco formando una matriz la cual es leída por un lector específico (Lector de QR), el término proviene del inglés "*Quick Response*" debido a la respuesta inmediata que nos ofrecen a través de su lectura.

En la actualidad la empresa Consultoría y Desarrollo Tecnológico MC4 S.R.L. cuenta con una gran cantidad de clientes a nivel nacional, para los cuales ha implementado sistemas a medida satisfaciendo de esta forma sus necesidades.

Entre los productos más importantes se encuentra el sistema UNICO como una plataforma para la gestión de pagos a través de canales electrónicos con más de 40 empresas integradas; mediante el cual entidades financieras y no financieras pueden interconectarse a este y permitir el pago de servicios a través de los canales como ser banca por internet, banca móvil, cajeros automáticos, ventanillas y quioscos, facilitando de esta forma la comunicación entre las empresas de servicio y las entidades financieras. (Consultoría y Desarrollo Tecnológico MC4 S.R.L., 2017)

Otro sistema que la empresa Consultoría y Desarrollo Tecnológico MC4 S.R.L. ofrece es el Cliente ACH EXPRESS el cual es un sistema de mensajería de transacciones interbancarias que permite realizar el pago y cobro a través de imágenes QR. Con este sistema la entidad financiera puede conectarse y permitir a sus clientes generar QRs de cobro, para ser compartidos por redes sociales o correo electrónico; esta acción permite a otro cliente (pagador) realizar la lectura del QR con la aplicación móvil de su entidad financiera (puede ser diferente a la del cobrador) y efectivizar el pago sin tener que llenar el formulario de la cuenta destino, nombre del destinatario, monto, moneda y banco destino, ya que estos datos se encuentran en el QR leído, llevando de esta forma la seguridad de las entidades financieras en un lenguaje estándar de comunicación para realizar transferencias entre personas naturales son la facilidad de la lectura de imágenes y la comodidad de la Banca Móvil. (Consultoría y Desarrollo Tecnológico MC4 S.R.L., 2017)

Ambos sistemas UNICO y ACH EXPRESS actualmente se encuentran productivos a lo largo del territorio nacional posicionando de esta forma a la empresa a la empresa Consultoría y Desarrollo Tecnológico MC4 S.R.L. entre una de las empresas de desarrollo de software con mayor experiencia en soluciones para la banca. (Consultoría y Desarrollo Tecnológico MC4 S.R.L., 2017)

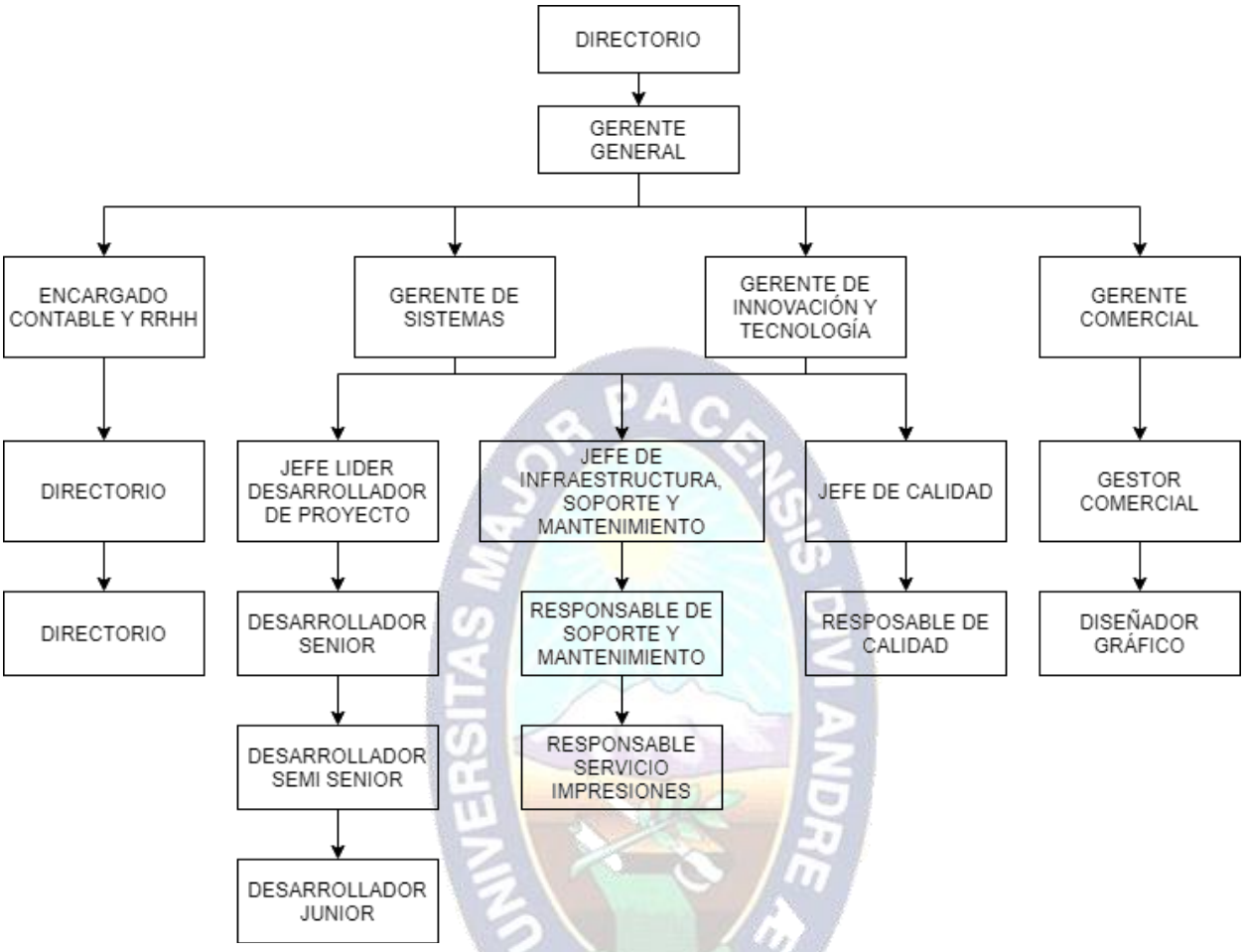


Figura 1.1 Organigrama MC4
Fuente: Consultoría y Desarrollo Tecnológico MC4 S.R.L., 2017

1.2.1.2. ANTECEDENTES DE CONFIDENCIALIDAD (LEGALES)

El presente proyecto, se encuentra limitado en la divulgación de información esto debido a acuerdos de confidencialidad suscritos entre la empresa Consultoría y Desarrollo Tecnológico MC4 S.R.L. y sus clientes y a su vez, entre MC4 S.R.L. y mi persona (Jonathan Valdivia Ramos), que me comprometen y obligan a guardar la más absoluta confidencialidad y reserva sobre toda la información con motivo de su relación con la empresa Consultoría y Desarrollo tecnológico MC4 S.R.L. y que prohíben revelar, entregar, divulgar, proporcionar y comunicar a terceros, parcial o totalmente bajo ninguna forma o modalidad, la información y documentos propiedad de MC4 S.R.L., sin embargo para objeto de estudio del presente documento se cuenta con la excepción y el permiso de Gerencia General y Gerencia de Desarrollo de MC4 S.R.L.,

para mostrar información sobre flujos de trabajo, requerimientos y solicitudes, para así demostrar la solución obtenida y que se detalla en el presente documento.

1.2.1.3. ANTECEDENTES DE TRABAJOS SIMILARES

En la actualidad existen sistemas que se encargan del pago de servicios, como ser:

- **PAGOSNET** Síntesis S.A. 2019 de Síntesis S.A.

  Es una plataforma que permite a empresas pequeñas, medianas y de comercio electrónico, gestionar sus ventas y recaudaciones a través de múltiples medios de pago. La propuesta de valor se centra en la transparencia, seguridad en el manejo de la información y los servicios de valor agregado, que facilitan y reducen los tiempos operativos de cualquier empresa. (Sintesis S.A., 2019).

- **KHIPU** Khipu 2019 de Khipu

  Khipu, mediante su aplicación móvil, puede generar un Código QR para que el cliente lo lea y realice el pago en su propio dispositivo móvil. El pago con Código QR puede ser utilizado por restaurantes, comercios en ferias, tiendas que no poseen un POS; haciendo más fácil el cobro presencial, pero con la seguridad de que el mismo se realiza en el dispositivo del cliente. (Khipu, 2019).

- **UNICO** Sinchy Diaz 2017 de Consultoría y Desarrollo Tecnológico MC4 S.R.L.

  Plataforma para la gestión de pagos a través de canales electrónicos con más de 40 empresas integradas; mediante el cual entidades financieras y no financieras pueden interconectarse a este y permitir el pago de servicios a través de los canales como ser banca por internet, banca móvil, cajeros automáticos, ventanillas y quioscos. (Consultoría y Desarrollo Tecnológico MC4 S.R.L., 2017).

1.2.2. FORMULACIÓN DEL PROBLEMA

La banca electrónica es considerada en el panorama actual como una nueva unidad estratégica de negocio dentro de la banca tradicional, por lo tanto, ofrecer un amplio rango de productos y servicios permite crear lealtad y alimentar relaciones más profundas con el cliente.

Así, la banca vive una revolución tecnológica. Los cambios de hábitos de los consumidores, los medios electrónicos, la penetración de Internet y la expansión de las nuevas tecnologías han empujado a las entidades a afrontar su digitalización para no quedarse atrás en un mercado cada vez más competitivo.

Sin embargo, esta competencia entre las entidades ha conseguido que se dividan el acceso a pagos de las diferentes empresas de servicio, tal como se muestra en los antecedentes de la sección 1.2, por lo cual para pagar un determinado servicio básico es necesario hacerlo mediante un banco específico.

1.2.2.1. PROBLEMA CENTRAL

Tomando en cuenta lo anteriormente descrito podemos plantear la siguiente pregunta:

¿Cómo integrar el pago de servicios desarrollando un producto de la empresa MC4 SRL que sea de acceso público y que permita un proceso intuitivo y disponible para la mayor cantidad de entidades financieras?

1.2.2.2. PROBLEMAS SECUNDARIOS

- No existe un portal que abarque todas las empresas de servicio para realizar el pago, por lo cual el usuario debe ingresar a algún portal de banca por internet que contiene una cantidad limitada de empresas de servicio para su pago.
- No se cuenta con la posibilidad de generar imágenes QR para el pago de servicios, lo que no permite al cliente realizar pagos de servicios de una manera sencilla y rápida.
- Cada entidad financiera tiene una forma distinta de integración con las empresas de servicio, lo que ocasiona un trabajo adicional de desarrollo.
- El pago de servicios en ventanilla actualmente es un proceso moroso que genera pérdida de tiempo del cliente y costo de personal a la entidad financiera.

1.3. OBJETIVOS

1.3.1. OBJETIVO GENERAL

Desarrollar una plataforma de pagos para la empresa MC4 S.R.L. que integre el sistema de transferencias por medio de un código QR y el sistema de cobranza UNICO para generar un canal de pagos de acceso público, e intuitivo.

1.3.2. OBJETIVOS ESPECÍFICOS

- Implementar un portal único, en la que el pago de servicios no dependa de una cuenta en una entidad financiera específica.
- Generar imágenes QR para el cobro de servicios mediante la plataforma SIP.
- Permitir la integración de cualquier empresa de servicio mediante la plataforma SIP de una manera estándar.
- Exponer códigos QR de cobro de servicios para que el cliente pueda realizar el pago a través de una transferencia interbancaria a partir de cualquier aplicación móvil de las 14 entidades financieras que operan con transacciones de ACCL Simple.

1.4. JUSTIFICACIÓN

Actualmente un cliente puede realizar el pago de sus servicios ingresando a su portal de banca por internet o banca móvil y escoger el servicio que desea pagar, consultar su deuda, seleccionar la cuenta desde la cual realizará el pago y efectivizar dicho pago, si su entidad no cuenta con el acuerdo del cobro de servicio de la empresa el cliente se ve obligado a acceder a otro portal de banca por internet que cuente con el servicio que desea pagar y en el peor de los casos, se vería obligado a ir presencialmente al banco para realizar el pago que desea, la creación de la plataforma SIP permitiría englobar a todas las empresas de servicio y procesar pagos con imágenes QR sin necesidad de que la entidad financiera de su preferencia cuente con el acuerdo del cobro de la empresa de servicios.

1.4.1. JUSTIFICACIÓN ECONÓMICA

El proyecto se justifica económicamente porque tanto las empresas de servicio que se integren a la plataforma SIP tendrán una ventaja competitiva y sumarán un canal más de pago a sus procesos actuales de cobro, por lo cual las entidades financieras que se integren a la plataforma SIP tendrán una ventaja económica ya que serán los recaudadores de todos los pagos que se generen a través de este canal.

1.4.2. JUSTIFICACIÓN SOCIAL

Esta tecnología generará mayor inclusión financiera, ya que permitirá la ampliación del uso de este medio de pago hacia segmentos de la población desatendidos, en áreas urbanas y rurales que tengan acceso a los servicios de telefonía móvil e internet (cada vez más extendidos).

Eliminará la fricción en las transacciones financieras y personales y, por lo tanto, eliminará la complejidad de realizar pagos a través de dispositivos móviles e incentiva mayor confianza en el usuario final.

Su uso incentiva la formalización de la economía, a la vez que facilitará el desarrollo de emprendimientos económicos familiares, artesanales, pues permitirá concretar sus ventas y servicios a través del uso de este medio de pagos.

1.4.3. JUSTIFICACIÓN TÉCNICA

La interoperabilidad con la tecnología de códigos QR es posible porque está interconectada al sistema de transferencias electrónicas de fondos, que funciona con la participación de todas las entidades de intermediación financiera.

El código QR generado cumplen con normas internacionales de seguridad y son encriptados y firmados digitalmente para asegurar la autenticidad, integridad y confiabilidad de los mismos.

1.5. ALCANCES Y LÍMITES

Los alcances definen los módulos que comprende el sistema y límites denota los aspectos que el proyecto no va a considerar, de tal manera que se hace un detalle de ambos criterios a continuación:

1.5.1. ALCANCES

Los alcances están enmarcados en la obtención de resultados de los objetivos específicos planteados anteriormente, el sistema a desarrollar está orientado a:

- Módulo de Administración de usuarios, roles y permisos para el control de los accesos a la información.
- Módulo de Parametrización para la debida configuración de la información inicial que requiera el sistema para su funcionamiento.
- Módulo de integración de los servicios REST expuestos por el Core de la Entidad Financiera, estos servicios permiten realizar los procesos de generación de imágenes QR.
- Módulo de Jobs para la conciliación de las transferencias pendientes o con errores.
- API estándar para la integración de las empresas de servicio con la plataforma SIP

1.5.2. LÍMITES

El desarrollo del sistema web está funcionalmente limitado por los siguientes puntos:

- El desarrollo de la plataforma SIP no incluye la aplicación móvil que servirá a la entidad financiera para generar el pago del código QR.
- La plataforma SIP no contiene la solución de registro de almacenes y productos.
- La plataforma SIP no se encarga de la mensajería estándar para el envío de transacciones ACH Express.
- La plataforma SIP no permite el registro de clientes externos.

1.6. DISEÑO METODOLÓGICO

La metodología es una guía que nos va indicando qué hacer y cómo actuar cuando se quiere obtener una investigación. Es posible decir que la metodología es aquel enfoque que permite observar un problema de una forma total, sistemática y disciplinada. (Yin, 2014, p. 13)

Para la parte de obtención y desarrollo del sistema de información, se aplicará una de las metodologías ágiles como lo es SCRUM que se adecua mejor por su flexibilidad en el desarrollo del producto. En cuanto a la herramienta de diseño se utilizará una metodología para aplicaciones orientadas a la web como lo es UWE, la cual trabaja con diagramas UML.

Método de investigación que realizaremos en este trabajo es el método científico, el cual es un proceso destinado a explicar fenómenos físicos del mundo y permitan obtener, con estos conocimientos, aplicaciones útiles al hombre.

El tipo de investigación en principio será de tipo exploratorio, porque, aunque el tema de estudio es conocido, se modifican varios procesos. Después será una investigación descriptiva, ya que se conocerá las situaciones y actitudes predominantes a través de la descripción exacta de las actividades, objetos, procesos y personas.

CAPITULO II

MARCO TEÓRICO

2.1. INTRODUCCIÓN

En este capítulo describiremos e introduciremos los principios y conceptos básicos para la realización del proyecto, sin embargo, no se puede dar una teoría completa acerca de las metodologías, técnicas y herramientas que se utilizará por el contrario se trata de presentar una base para la fácil comprensión de la misma.

2.2. INGENIERÍA DE SOFTWARE

Según la definición de Sommerville (2005), la ingeniería de software es una disciplina de ingeniería que se interesa por todos los aspectos de la producción de software, desde las primeras etapas de la especificación del sistema hasta el mantenimiento del sistema después de que se pone en operación. En esta definición se presentan dos frases clave:

- **Disciplina de ingeniería** Los ingenieros hacen que las cosas funcionen. Aplican teorías, métodos y herramientas donde es adecuado. Sin embargo, los usan de manera selectiva y siempre tratan de encontrar soluciones a problemas, incluso cuando no hay teorías ni métodos aplicables. Los ingenieros también reconocen que se deben trabajar ante restricciones organizacionales y financieras, de modo que buscan soluciones dentro de tales limitaciones.
- **Todos los aspectos de la producción del software** La ingeniería de software no solo se interesa por los procesos técnicos del desarrollo del software, sino también incluye actividades como la administración del proyecto software y el desarrollo de herramientas, así como métodos y teorías para apoyar la producción de software.

La ingeniería busca obtener resultado de la calidad requerida dentro de la fecha y del presupuesto, A menudo esto requiere contraer compromisos: los ingenieros no deben ser perfeccionistas. Sin embargo, las personas que diseñan programas para sí mismas podrían pasar tanto tiempo como deseen en el desarrollo del programa.

En general los ingenieros de software adoptan en su trabajo un enfoque sistemático y organizado, pues usualmente esta es la forma más efectiva de producir software de alta calidad. No obstante, la ingeniería busca seleccionar el método más adecuado para un conjunto de circunstancias y, de esta manera, un acercamiento al desarrollo más creativo y menos formal sería efectivo en ciertas situaciones. El desarrollo menos formal es particularmente adecuado para la creación de sistemas basados en la Web, que requieren una mezcla de habilidades de software y diseño gráfico.

La ingeniería de software es importante por dos razones:

- Cada vez con mayor frecuencia, los individuos y la sociedad se apoyan en los avanzados sistemas de software. Por ende, se requiere producir económica y rápidamente sistemas confiables.
- A menudo resulta más barato a largo plazo usar métodos y técnicas de ingeniería de software para los sistemas de software, que sólo diseñar los programas como si fuera un proyecto de programación personal. Para muchos tipos de sistemas, la mayoría de los costos consisten en cambiar el software después de ponerlo en operación.

El enfoque sistemático que se usa en la ingeniería de software se conoce en ocasiones como proceso de software. Un proceso de software es una secuencia de actividades que conducen a la elaboración de un producto de software. Existen cuatro actividades fundamentales que son comunes a todos los procesos de software, y estas son:

- Especificación del software, donde clientes e ingenieros definen el software que se producirá y las restricciones en su operación.
- Desarrollo del software, donde se diseña y programa el software.
- Validación del software, donde se verifica el software para asegurar que sea lo que el cliente requiere.
- Evolución de software, donde se modifica el software para reflejar los requerimientos cambiantes del cliente y del mercado.

Diferentes tipos de sistemas necesitan distintos procesos de desarrollo. Por ejemplo, el software en tiempo real en una aeronave debe especificarse por completo antes de comenzar el desarrollo. En los sistemas de comercio electrónico, la especificación y el programa por lo general se

desarrollan en conjunto. En consecuencia, tales actividades genéricas pueden organizarse en diferentes formas y describirse en distintos niveles de detalle, dependiendo del tipo de software que se vaya a desarrollar.

La ingeniería de software se relaciona con las ciencias de la computación y la ingeniería de sistemas:

- Las ciencias de la computación se interesan por las teorías y los métodos que subyacen en las computadoras y los sistemas de software, en tanto que la ingeniería de software se preocupa por los asuntos prácticos de la producción del software. Cierto conocimiento de ciencias de la computación es esencial para los ingenieros de software, del mismo modo que cierto conocimiento de física lo es para los ingenieros electricistas. Sin embargo, con frecuencia la teoría de las ciencias de la computación es más aplicable a programas relativamente pequeños. Las teorías de las ciencias de la computación no siempre pueden aplicarse a grandes problemas complejos que requieren una solución de software.
- La ingeniería de sistemas se interesa por todos los aspectos del desarrollo y la evolución de complejos sistemas, donde el software tiene un papel principal. Por lo tanto, la ingeniería de sistemas se preocupa por el desarrollo de hardware, el diseño de políticas y procesos, la implementación del sistema, así como por la ingeniería de software. Los ingenieros de sistemas intervienen en la especificación del sistema, definiendo su arquitectura global y, luego integrando las diferentes partes para crear el sistema terminado. Están menos preocupados por la ingeniería de los componentes del sistema (hardware, software, y otros).

Como se expone en la siguiente sección, hay muchos tipos diferentes de software. No existe un método o una técnica universales en la ingeniería de software que sea aplicable para todos estos. No obstante, tres problemas generales afectan a muy diversos tipos de software:

1. **Heterogeneidad** Cada vez con mayor frecuencia se requieren sistemas que operen como distribuidos a través de redes que incluyan diferentes tipos de computadoras y dispositivos móviles. Es posible que el software se ejecute tanto en computadoras de propósito general como en teléfonos móviles. Se tendrá que integrar con frecuencia el nuevo software con sistemas legados más viejos, escritos en diferentes lenguajes de

programación. El reto aquí es desarrollar técnicas para construir software confiable que sea suficientemente flexible para enfrentar esa heterogeneidad.

2. **Cambio empresarial y social** Los negocios y la sociedad cambian de manera increíblemente rápida, conforme se desarrollan las economías emergentes y nuevas tecnologías están a la disposición. Ambos necesitan tener la posibilidad de cambiar su software existente y desarrollar rápidamente uno nuevo. Muchas técnicas tradicionales de la ingeniería de software consumen tiempo, y generalmente la entrega de los nuevos sistemas tarda más de lo planeado. Requieren evolucionar de modo se reduzca el tiempo necesario para que el software dé valor a sus clientes.
3. **Seguridad y confianza** Dado que el software está vinculado con todos los aspectos de la vida, es esencial confiar en dicho software. Esto es especialmente cierto para los sistemas de software remoto a los que se accede a través de una página Web o una interfaz de servicio Web. Es necesario asegurarse de que usuarios malintencionados no puedan atacar el software y que se conserve la seguridad de la información.

Desde luego, éstos no son problemas independientes. Por ejemplo, quizá sea necesario realizar cambios rápidos a un sistema legado con la finalidad de dotarlo con una interfaz de servicio Web. Para enfrentar dichos retos se necesitarán nuevas herramientas y técnicas, así como formas innovadoras de combinar y usar los métodos existentes de ingeniería de software.

2.3. MÉTODOS ÁGILES

En la década de los 1990 el descontento con estos enfoques engorrosos de la ingeniería de software condujo a alguno desarrolladores de software a proponer nuevos “métodos ágiles”, los cuales permitieron que el equipo de desarrollo se enfocará en el software en lugar del diseño y la documentación. Los métodos ágiles se apoyan universalmente en el enfoque incremental para la especificación, el desarrollo y la entrega del software. Son más adecuados para el diseño de aplicaciones en que los requerimientos del sistema cambian, por lo general, rápidamente durante el proceso de desarrollo. Tienen la intención de entregar con prontitud el software operativo a los clientes, quienes entonces propondrán requerimientos nuevos y variados para incluir en posteriores iteraciones del sistema. Se dirigen a simplificar el proceso burocrático al evitar trabajo con valor dudoso a largo plazo, y a eliminar documentación que quizá nunca se emplee.

Aunque los métodos de desarrollo ágil se basan en el desarrollo y la entrega de producto incremental, estos proponen diferentes procesos para lograrlo. Sin embargo, comparten una serie de principios.

Principio	**Descripción**
Participación del Cliente	Los clientes deben intervenir estrechamente durante el proceso de desarrollo. Su función consiste en ofrecer y priorizar nuevos requerimientos del sistema y evaluar las iteraciones del mismo.
Entrega Incremental	El software se desarrolla en incrementos y el cliente especifica los requerimientos que se van a incluir en cada incremento.
Personas, no procesos	Tienen que reconocerse y aprovecharse las habilidades del equipo de desarrollo. Debe permitirse a los miembros del equipo desarrollar sus propias formas de trabajar sin procesos establecidos.
Adoptar el cambio	Esperar a que cambien los requerimientos del sistema y, de este modo, diseñar el sistema para adaptar dichos cambios.
Mantener Simplicidad	Enfocarse en la simplicidad tanto en el software a desarrollar como en el proceso de desarrollo, Siempre que sea posible, trabajar de manera activa para eliminar la complejidad del sistema.

Según Sommerville (2005) en la tabla 2.1 se detallan los principios de los métodos ágiles.

Tabla 2.1. Principios de los métodos ágiles
Fuente: Somerville, 2011

Los métodos ágiles han tenido mucho éxito para ciertos tipos de desarrollo de sistemas:

1. Desarrollo del producto, donde una compañía de software elabora un producto pequeño o mediano para su venta.

2. Diseño de sistemas a la medida dentro de una organización, donde hay un claro compromiso del cliente por intervenir en el proceso de desarrollo, y donde no existen muchas reglas ni regulaciones externas que afecten el software.

Cualquier proceso del software ágil se caracteriza por la forma en la que aborda cierto número de suposiciones clave acerca de la mayoría de proyectos de software:

1. Es difícil predecir qué requerimientos de software persistirán y cuáles cambiarán. También es difícil pronosticar cómo cambiarán las prioridades del cliente a medida que avanza el proyecto.

2. Para muchos tipos de software, el diseño y la construcción están imbricados. Es decir, ambas actividades deben ejecutarse en forma simultánea, de modo que los modelos de diseño se prueben a medida que se crean. Es difícil predecir cuánto diseño se necesita antes de que se use la construcción para probar el diseño.

3. El análisis, el diseño, la construcción y las pruebas no son tan predecibles como nos gustaría (desde un punto de vista de planeación).

4. Dadas estas tres suposiciones, surge una pregunta importante: ¿cómo crear un proceso que pueda manejar lo impredecible? La respuesta, como ya se dijo, está en la adaptabilidad del proceso (al cambio rápido del proyecto y a las condiciones técnicas), por tanto, un proceso ágil debe ser adaptable.

Pero la adaptación continua logra muy poco si no hay avance. Entonces, un proceso de software ágil debe adaptarse incrementalmente. Para lograr la adaptación incremental, un equipo ágil requiere retroalimentación con el cliente (de modo que sea posible hacer las adaptaciones apropiadas). Un catalizador eficaz para la retroalimentación con el cliente es un prototipo operativo o una porción de un sistema operativo. Así, debe instituirse una estrategia de desarrollo incremental. Deben entregarse incrementos de software (prototipos ejecutables o porciones de un

sistema operativo) en periodos cortos de tiempo, de modo que la adaptación vaya a ritmo con el cambio (impredecible). Este enfoque iterativo permite que el cliente evalúe en forma regular el incremento de software, dé la retroalimentación necesaria al equipo de software e influya en las adaptaciones del proceso que se realicen para aprovechar la retroalimentación.

Existen 12 principios de agilidad para aquellos que la quieran alcanzar:

1. La prioridad más alta es satisfacer al cliente a través de la entrega pronta y continua de software valioso.

2. Son bienvenidos los requerimientos cambiantes, aun en una etapa avanzada del desarrollo. Los procesos ágiles dominan el cambio para provecho de la ventaja competitiva del cliente.

3. Entregar con frecuencia software que funcione, de dos semanas a un par de meses, de preferencia lo más pronto que se pueda.

4. Las personas de negocios y los desarrolladores deben trabajar juntos, a diario y durante todo el proyecto.

5. Hay que desarrollar los proyectos con individuos motivados. Debe darse a éstos el ambiente y el apoyo que necesitan, y confiar en que harán el trabajo.

6. El método más eficiente y eficaz para transmitir información a los integrantes de un equipo de desarrollo, y entre éstos, es la conversación cara a cara.

7. La medida principal de avance es el software que funciona.

8. Los procesos ágiles promueven el desarrollo sostenible. Los patrocinadores, desarrolladores y usuarios deben poder mantener un ritmo constante en forma indefinida.

9. La atención continua a la excelencia técnica y el buen diseño mejora la agilidad.

10. Es esencial la simplicidad: el arte de maximizar la cantidad de trabajo no realizado.

11. Las mejores arquitecturas, requerimientos y diseños surgen de los equipos con organización propia.

12. El equipo reflexiona a intervalos regulares sobre cómo ser más eficaz, para después afinar y ajustar su comportamiento en consecuencia.

2.4. METODOLOGÍA DE DESARROLLO

En este capítulo describiremos e introduciremos los principios y conceptos básicos para la realización del proyecto, sin embargo, no se puede dar una teoría completa acerca de las metodologías, técnicas y herramientas que se utilizará por el contrario se trata de presentar una base para la fácil comprensión de la misma.

2.4.1. SCRUM

SCRUM es un *framework*[3] que es usado para implementar desarrollo ágil, un *framework* dentro del cual se puede emplear varios proceso y técnicas que permitirán mejorar el producto, el equipo y el ambiente de trabajo continuamente. SCRUM emplea un enfoque iterativo e incremental para optimizar la predicción y control de riesgos. (Scrum.org, 2020)

Según el portal "*Scrum.org The Home of Scrum*" (2020) esta metodología ágil se define en los siguientes pasos.

2.4.1.1. PROCESO

La figura 2.1 muestra el proceso que sigue la metodología SCRUM, donde se aplican las actividades que serán detalladas en el punto 2.2.1.3 por los roles que se detallan en el punto 2.2.1.2 haciendo un resumen del proceso completo de la metodología SCRUM.

[3] Los frameworks, también llamados entornos o marcos de trabajo, son conjuntos de componentes de software que se utilizan para crear la estructura de un software o una aplicación. Un framework puede verse como una caja de herramientas en la que el desarrollador busca los componentes necesarios. Este marco proporciona una estructura general para facilitar la labor de desarrollo. (School, 2019)

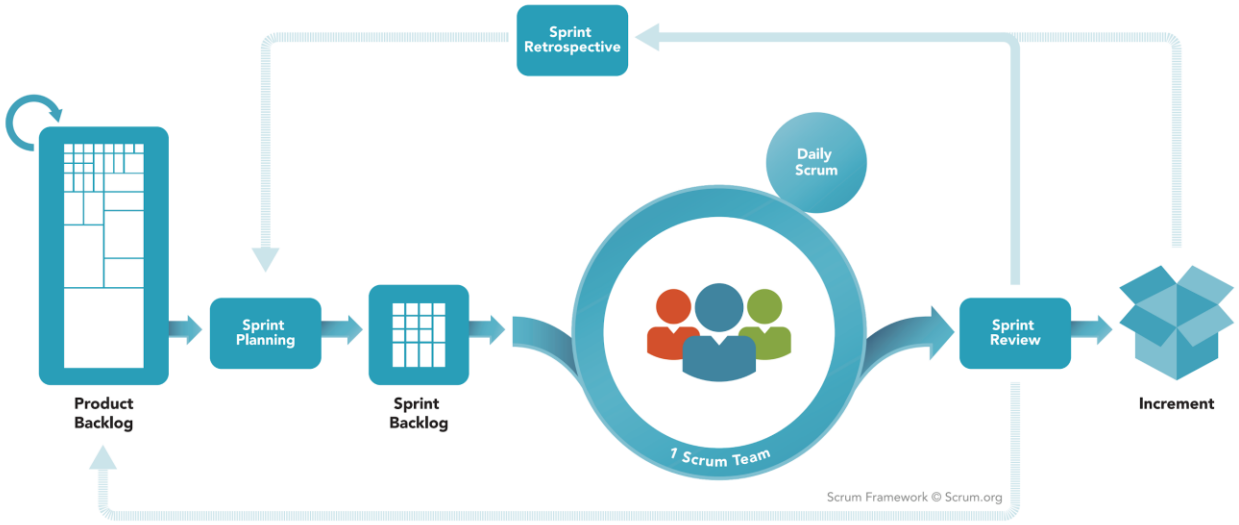


Figura 2.1 Proceso SCRUM
Fuente: Scrum.org, 2020

Scrum es un *framework* cuyo proceso se describe de la siguiente manera:

El proceso empieza por un *Product Backlog* que contiene la lista de requerimientos del producto. El *Product Owner*, que representa al negocio, se encarga de mantener la lista actualizada y priorizada.

Luego el equipo de desarrollo (*Development Team*) decide qué tareas puede completar en un tiempo que dura un *Sprint*, durante todo el tiempo de desarrollo el equipo se reúne diariamente (*Daily scrum*) para analizar la lista de tareas y actualizar el trabajo que realizan a diario, también comparten dudas, problemas o impedimentos y juntos deciden cómo pueden resolverlos.

Una vez que los requerimientos del *Sprint* estén terminados realizan una revisión del incremento (*Sprint Review*) con los *Stakeholders*, y en base a las observaciones actualizan el *Product Backlog*.

Después del *Sprint Review*, el *Scrum Master* organiza una sesión de Retrospectiva para todo el equipo y analizan la forma de trabajar, plantean soluciones a los problemas o mejoras que deben realizar.

La retrospectiva marca el final del *Sprint* y comienza un nuevo *Sprint* siguiendo el mismo proceso.

2.4.1.2. ROLES

Según el artículo “Scrum: roles y responsabilidades” Palacios (2020) los roles de SCRUM son:

- **Dueño del producto (***Product Owner***)**

  Es la persona responsable de administrar el *Product Backlog*, expresando claramente los elementos del backlog, ordenándolos de manera que sean entendibles, visibles y claros.

  El dueño del puede hacer el trabajo de administración del *Product Backlog* o puede hacer que el equipo de desarrollo lo haga, pero el Dueño del producto sigue siendo responsable.

  El Dueño del producto es una sola persona que puede representar a un comité; las decisiones del Dueño del producto deben ser respetadas y expresadas en los elementos del *Product Backlog*.

- **Equipo de desarrollo (***Development team***)**

  El equipo de desarrollo es aquel que realiza el desarrollo del producto para entregarlo al final de cada *sprint*. Es un equipo multifuncional en el que todos los integrantes trabajan de forma responsable, solidaria, de manera cohesionada y auto organizada.

- **Maestro SCRUM (***Scrum Master***)**

  El maestro SCRUM es el encargado de promover SCRUM, ayudando a entender la teoría, prácticas, reglas y valores de SCRUM.

  a) El servicio del maestro SCRUM al Dueño del producto

     - Se asegura que los objetivos, alcance y el dominio del producto son entendidos por cada uno de los integrantes del equipo SCRUM.
     - Encontrar técnicas para una administración efectiva del *Product Backlog*.
     - Ayuda al equipo Scrum entender la necesidad de elementos claros y concisos del *Product Backlog*.
     - Se asegura que el Dueño del producto sepa como ordenar el Product Backlog para maximizar el valor.

- Entender las prácticas ágiles.
- Facilitar los demás eventos Scrum a medida que sean requeridos.

b) El servicio del maestro SCRUM al Equipo de desarrollo

- Guiar al equipo de desarrollo en auto organización.
- Ayudar a crear productos de alto valor.
- Quitar impedimentos que afecten al progreso del desarrollo.
- Facilitar eventos SCRUM cuando se necesiten o se soliciten.
- Guiar al equipo de desarrollo en organizaciones donde todavía no se haya adoptado o entendido SCRUM.

c) El servicio del maestro SCRUM a la Organización

- Liderar y guiar a la organización en la adopción de SCRUM.
- Planificar implementaciones SCRUM dentro de la organización.
- Ayudar empleados y *stakeholders* a entender, aplicar SCRUM y el desarrollo empírico de productos.
- Provocar cambios que incrementen la productividad del equipo SCRUM.
- Trabajar con otros SCRUM masters para incrementar la efectividad de la aplicación de SCRUM en la reunión

2.4.1.3. ACTIVIDADES O EVENTOS (SCRUM EVENTS)

Las actividades o eventos se utilizan en SCRUM para crear regularidad y minimizar la necesidad de reuniones no definidas en SCRUM.

Todos los eventos tienen una definición de tiempo máximo.

Un sprint es un evento contenedor de otros eventos, y estos eventos son una oportunidad para inspeccionar y adaptar algo de manera transparente.

Según “La Guía de Scrum: Las Reglas del Juego” (Schwaber & Sutherland, 2013) se definen los siguientes eventos:

- **Sprint**

  El corazón de SCRUM es un sprint que tiene una duración de máximo un mes, donde un incremento del producto hecho, usable y potencialmente liberable es creado. La duración de los *sprints* tiene un tiempo consistente con el esfuerzo requerido. Inmediatamente un sprint es acabado, otro comienza.

  Durante un sprint no existen cambios que comprometan el objetivo del *sprint*, los objetivos de calidad no rebajan y el alcance puede ser aclarado y renegociado entre el dueño del producto y el equipo de desarrollo mientras más se aprende.

  Un *sprint* puede ser cancelado antes de que el tiempo se cumpla. Solamente el dueño del producto tiene la autoridad para realizar esta acción, aunque puede verse influenciado por los demás involucrados, el equipo de desarrollo o el *Scrum Master*.

- **Planificación de un sprint (*Sprint planning*)**

  El trabajo que se hará en un sprint es planificado en esta actividad, se define el objetivo del sprint y se seleccionan los elementos del *Product Backlog* y se define el *sprint backlog*. Esta actividad es llevada a cabo con el equipo Scrum. La planificación del sprint tiene una duración máxima de ocho horas para un sprint de un mes, para un sprint de menor duración la planificación es más corta. El maestro Scrum se asegura que este evento se lleve a cabo y que los asistentes entiendan el propósito del mismo y que se lleve a cabo en el tiempo programado.

- **Scrum diario (*Daily Scrum*)**

  El *daily scrum* es un evento que dura 15 minutos para el equipo de desarrollo, y es llevado a cabo todos los días del *sprint*. En este evento el equipo planifica las próximas 24 horas de trabajo, se hace la revisión de las actividades llevadas a cabo el día anterior y se expone los problemas o impedimentos que se tengan para realizar el desarrollo.

- **Revision del sprint (*Sprint review*)**

  Esta es una reunión informal y no de informe de estado y máximo debe durar cuatro horas para sprints de un mes de duración. La revisión del sprint se lleva a cabo al final de cada sprint para revisar el incremento del producto y adaptar el *Product Backlog* si fuera necesario.

  La revisión incluye los siguientes elementos:

  - ➢ El dueño del producto explica los elementos "hechos" y "no hechos" del *product backlog*.
  - ➢ El equipo de desarrollo explica que es lo que se hizo durante el *sprint*, que problemas atravesaron y como resolvieron esos problemas.
  - ➢ El equipo de desarrollo demuestra el trabajo que está "terminado"
  - ➢ El grupo entero colabora en la definición de lo que se debe hacer luego que servirá como entrada para la planificación del siguiente *sprint*.

- **Retrospectiva del sprint (*Sprint retrospective*)**

  Esta actividad se lleva a cabo después del *sprint review* y previo a la planificación del siguiente *sprint*. Este evento debe durar máximo tres horas para un sprint de un mes.

  El propósito de la retrospectiva es:

  - ➢ Examinar cómo marchó el último *sprint* con respecto a las personas, relaciones, procesos y herramientas.
  - ➢ Identificar y ordenar los elementos que fueron bien y potenciales mejoras.
  - ➢ Crear un plan para implementar mejoras al modo en el que el equipo Scrum realiza su trabajo.

2.4.1.4. ARTEFACTOS

Los artefactos de Scrum representan trabajo o valor en diversas formas que son útiles para proporcionar transparencia y oportunidades para la inspección y adaptación. Los artefactos definidos por Scrum están diseñados específicamente para maximizar la transparencia de la información clave, que es necesaria para asegurar que todos tengan el mismo entendimiento del artefacto. (Schwaber & Sutherland, 2013)

Según "La Guía de Scrum: Las Reglas del Juego" (Schwaber & Sutherland, 2013) se definen los siguientes artefactos de Scrum:

- **Pila de producto (*Product Backlog*)**

  El *Product Backlog* es una lista ordenada de las cosas que se sabe que son necesarias en el producto y este representa la única fuente de requerimientos para cualquier cambio que se deba realizar al producto. El dueño del producto es responsable del contenido, disponibilidad y orden.

  El *Product Backlog* es dinámico y constantemente cambia para definir las necesidades apropiadas y útiles del producto. Esta lista contiene los características, funciones, requisitos, mejoras y arreglos que constituyen los cambios a realizarse en futuros lanzamientos del producto.

  Los elementos del *Product Backlog* tienen atributos de descripción, orden, estimación, valor y la definición de "terminado" que prueba que un requisito es cumplido.

- **Pila de sprint (*Sprint Backlog*)**

  El *Sprint Backlog* es el conjunto de elementos seleccionados del *Product Backlog* que pertenecerán al *Sprint*, más un plan para liberar el incremento del producto y cumplir el objetivo del *Sprint*.

  El equipo de desarrollo modifica el *Sprint Backlog* a medida que se está ejecutando el *Sprint* ya que el equipo aprende más acerca del trabajo necesario para cumplir con el objetivo del *Sprint*.

El equipo de desarrollo es el único que puede modificar el *Sprint Backlog*, ya que puede adicionar más tareas que se requieran o quitar las que son innecesarias.

El seguimiento al trabajo realizado lo hace el equipo de desarrollo cada reunión diaria (*Daily Scrum*).

- **Incremento**

  El incremento es la suma de todos los elementos del *Product Backlog* que fueron completados durante el *Sprint* y que están en condición de "terminado". El incremento debe están en condición usable aun cuando el dueño del producto decida liberarlo o no.

  El incremento es funcionalidad adicionada al incremento anterior y que está correctamente probado asegurando que todos los incrementos trabajan bien juntos.

- **Definición de terminado (*Definition of "Done"*)**

  Cuando un elemento del *Product Backlog* o un incremento es descrito como "terminado", todos deben entender lo que "terminado" significa. El propósito de cada *Sprint* es liberar incrementos de funcionalidad potencialmente liberable, así el dueño del producto puede escogerlo para liberarlo inmediatamente.

2.5. ESPECIFICACIÓN DE REQUERIMIENTOS

La especificación de requerimientos es parte resultante de la ingeniería de requerimientos, esta debe ser consistente, no ambigua y servirá como base para acuerdos entre las partes involucradas, donde se describe las funciones que realizará el sistema.

La ingeniería de requerimientos proporciona el mecanismo para entender lo que desea el cliente, analizar las necesidades, evaluar la factibilidad, negociar una solución razonable, especificar la solución sin ambigüedades, validar la especificación y administrar los requerimientos a medida que se transforman en un sistema funcional (Pressman, 2010).

Según Pressman (2010) la ingeniería de requerimientos incluye siete tareas diferentes: Concepción, indagación, elaboración, negociación, especificación, validación y administración,

todas estas se adaptan a las necesidades del proyecto y algunas de estas tareas ocurren en paralelo.

A. **Concepción:** En esta se establece el entendimiento básico del proyecto, las personas que quieren una solución y una mejora, así como la eficacia de la comunicación y colaboración preliminares entre los participantes y el equipo de desarrollo de software. En esta parte, los ingenieros de software realizan preguntas "libres de contexto" (generales), para establecer un entendimiento básico del proyecto, identificar a los usuarios, negociar la mejor solución, definir el rol de los colaboradores y la manera de comunicaciones que existe entre los usuarios y el equipo de desarrollo.

B. **Indagación:** Se refiere a definir formalmente los requerimientos del proyecto, consultar al cliente, a los usuarios y bibliografía par tener los objetivos del sistema, lo que va a lograr, cómo se ajusta el sistema a las necesidades del negocio y, finalmente cómo va a usarse el sistema en las operaciones cotidianas. Sin embargo, esta tarea es difícil de realizar, puesto se puede presentar los siguientes problemas:

    - Problemas de alcance: la frontera del sistema está mal definida o los clientes o usuarios finales especifican detalles técnicos innecesarios, que confunden.
    - Problemas de entendimiento: Los clientes o usuarios no están completamente seguros de lo que se necesita.
    - Problemas de volatilidad: Los requerimientos cambian con el tiempo.

C. **Elaboración:** Se centra en desarrollar un modelo refinado de los requerimientos que identifique distintos aspectos de la función del software, su comportamiento e información. La elaboración está motivada por la creación y mejora de escenarios de usuario que describen cómo interactúa todos los usuarios finales con el sistema. Cada escenario de usuario se enuncia con sintaxis apropiada para extraer clases de análisis, que son entidades del dominio del negocio visibles para el usuario final. Se definen los atributos de cada clase de análisis y se identifican los servicios que requiere cada una de ellas. Se identifican las relaciones y colaboración entre clases, y se producen varios diagramas adicionales.

D. **Negociación:** Cuando el cliente o usuario propone requerimientos conflictivos con argumentos válidos. Estos requerimientos conflictivos deben reconciliarse por medio de un proceso de negociación. Se debe reunir a los participantes del proyecto para ordenar, priorizar y analizar el conflicto a través de un enfoque iterativo, además evaluar su coste y riesgo. Algunos requerimientos se deberán eliminar, combinar o modificar de modo que cada parte logre cierto grado de satisfacción.

E. **Especificación:** Una especificación puede ser: un documento escrito, un conjunto de modelos gráficos, un modelo matemático formal, un conjunto de escenarios de uso, un prototipo o combinación de éstos, según se requiera Pressman sugiere utilizar una "plantilla estándar", para obtener requerimientos consistentes y comprensibles, sin embargo, en ocasiones es necesario ser flexible cuando se desarrolla una especificación para sistemas grandes, el mejor enfoque sería un documento escrito que combine descripciones en un lenguaje natural con modelos gráficos y para sistemas pequeños con ambientes entendidos, será suficiente quizá escenarios de uso.

F. **Validación:** se validará la calidad de los requerimientos, a fin de garantizar que los requerimientos han sido enunciados sin ambigüedades; que se detectaron, corrigieron las inconsistencias, las omisiones y los errores, y que la especificación se presenta conforme a los estándares establecidos para el proyecto según los participantes.

G. **Administración:** Es el conjunto de actividades que ayudan al equipo de desarrollo del proyecto a identificar, controlar y dar seguimiento a los requerimientos y a sus cambios en cada fase del proyecto. Esta tarea se practica para proyectos grandes y para proyectos pequeños, esta actividad tiene considerablemente menos formalidad.

2.6. INGENIERÍA WEB

En 1998, con la moderación de Roger Pressman se llevó a cabo un debate virtual con representantes del desarrollo de software basado exclusivamente en Internet y representantes de la ingeniería de software tradicional. El debate en cuestión consistió en discutir si valía la pena aplicar un proceso de ingeniería para el desarrollo de una aplicación web (Webapp). La conclusión final fue que nunca estaba demás aplicar un proceso de ingeniería pero que este debía

adaptarse a los requerimientos de cambio continuo y rapidez que forman una parte importante del desarrollo de aplicaciones web. De este tipo de iniciativas y otras más como congresos nace el concepto de Ingeniería Web.

La ingeniería web es una versión adaptada del enfoque de ingeniería de software que se presenta en todo este libro. Propone una estructura ágil, pero disciplinada, para construir sistemas y aplicaciones basados en web con calidad industrial. (Pressman, 2010, p. 317)

Pressman además afirma que el diseño de webapps (Aplicaciones web) incluye actividades técnicas y no técnicas, que incluyen lo siguiente: establecer la vista y sensación de la webapp, creando la distribución estética de la interfaz de usuario, definiendo la estructura arquitectónica general, desarrollando el contenido y la funcionalidad que residen en la arquitectura y planeando la navegación que ocurre dentro de la webapp.

El diseño de una webapp incluye seis etapas principales que son orientadas por la información obtenida durante la modelación de los requerimientos.

1. **El diseño del contenido** utiliza el contenido del modelo (desarrollado durante el análisis) como la base para establecer el diseño de los objetos del contenido.

2. **El diseño estético** (también llamado diseño gráfico) establece la vista y sensación que el usuario final percibe.

3. **El diseño arquitectónico** se centra en la estructura general de hipermedios de todos los objetos y funciones del contenido.

4. **El diseño de la interfaz** establece la distribución y mecanismos de distribución que definen a la interfaz de usuario.

5. **El diseño de la navegación** define la forma en la que el usuario final navega a través de la estructura de hipermedios.

6. **El diseño de los componentes** representa la estructura interna detallada de los elementos funcionales de la webapp.

2.6.1. METODOLOGÍAS DE MODELADO UWE

El Método de Ingeniería Web basado en el Lenguaje de Modelado Unificado (UWE), fue propuesto por Nora Koch, del Instituto de Informática de la Universidad de Múnich de Alemania, y consiste de un método que permite especificar de mejor manera una aplicación Web en base a un proceso que utiliza el Proceso Unificado de Desarrollo de Software, por lo cual el proceso es iterativo e incremental, y además mantiene una notación estándar basada en el uso del Lenguaje de Modelado Unificado (Nieves Guerrero, Ucán Pech, & Méndez Domínguez, 2014). Este método a su vez tiene como fundamento el Desarrollo Dirigido por Modelos (MDD1), y consta de cinco fases o modelos, los cuales se ilustran en la Figura 2.2, y son:

1. El análisis de requerimientos.
2. Modelo de contenido.
3. Modelo de navegación
4. Modelo de presentación.
5. Modelo de proceso.

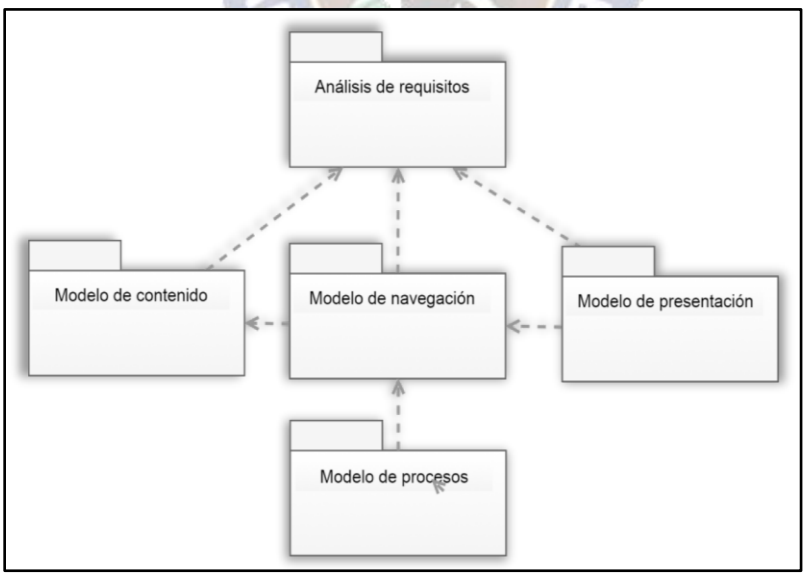


Figura 2.2 Meta Modelo de UWE
Fuente: Rossi, Schawabe, & Olsina, 2008

Basado en el análisis de requerimientos y el modelo de contenido es posible obtener el modelo de navegación. Luego, en base al modelo de navegación y los aspectos de la interfaz de usuario se obtiene el modelo de presentación. Finalmente, el modelo de navegación puede ser extendido

mediante el modelo de proceso que representa el aspecto que tienen las acciones de las clases de proceso.

2.6.1.1. ANÁLISIS DE REQUISITOS

El análisis de requisitos, consiste en la especificación de los casos de uso del sistema, y plasmar los requisitos funcionales de una aplicación Web (Nieves Guerrero, Ucán Pech, & Méndez Domínguez, 2014).

El caso de uso identifica y describe las funciones del sistema desde la perspectiva de los usuarios externos de una forma a una terminología que ellos entienden. El alcanzar exacta y minuciosamente esto requiere un alto nivel de participación por parte del usuario, así como de un experto en la materia que sea versado en el proceso de negocios. Los casos de uso es una técnica excelente para entender y documentar mejor los requerimientos del sistema.

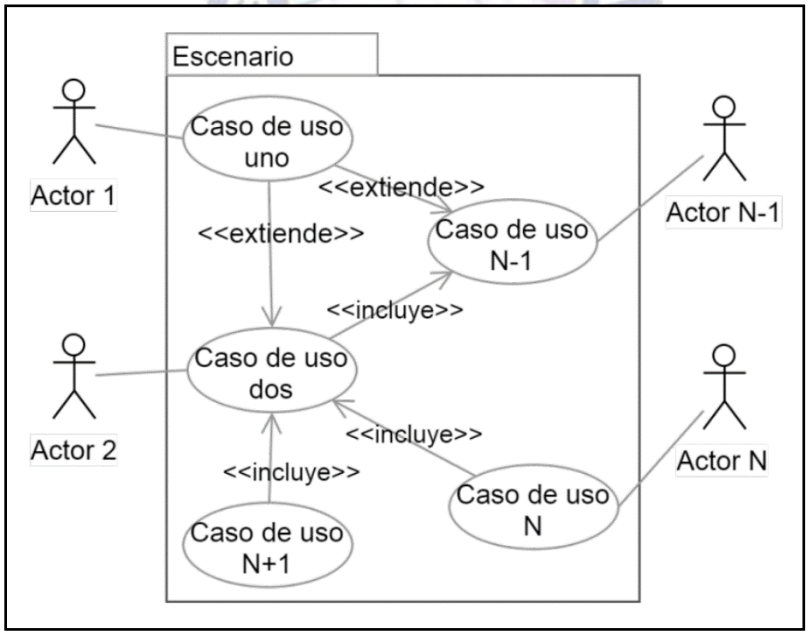


Figura 2.3 La representación gráfica del caso de uso.
Fuente: Pressman, 2010, p. 137

2.6.1.2. MODELO DE CONTENIDO

Modelo de contenido, o diseño conceptual, es la etapa donde se representa el dominio del problema con un diagrama de clases de UML, definiendo a detalle los conceptos involucrados en la aplicación (Nieves Guerrero, Ucán Pech, & Méndez Domínguez, 2014).

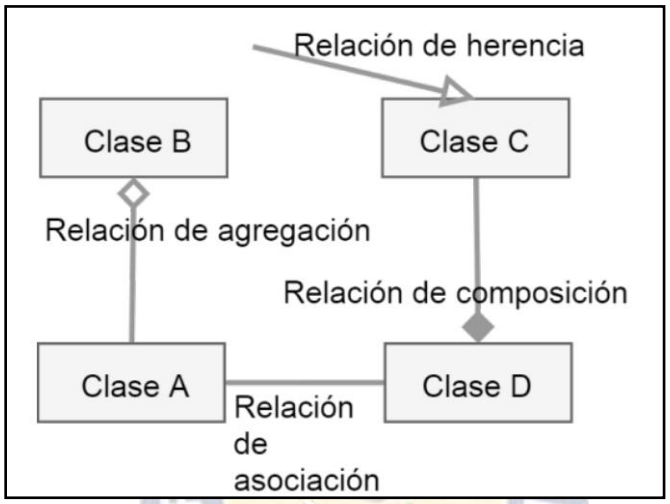


Figura 2.4 Diagrama de clases simplificada.
Fuente: Sommerville, 2005, p. 166

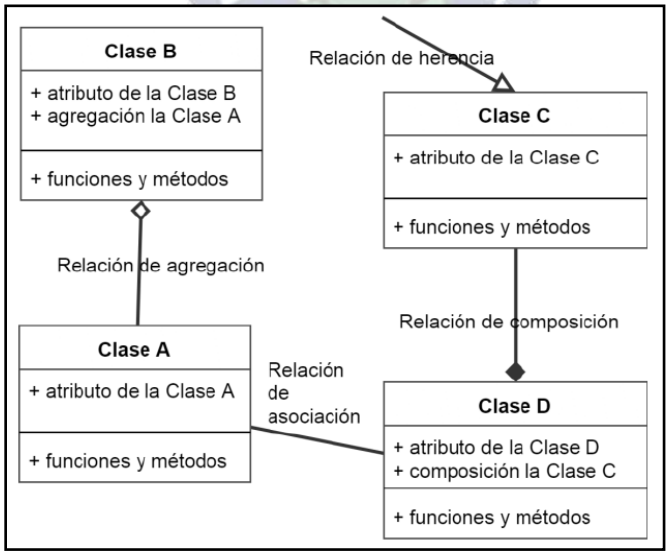


Figura 2.5 Diagrama de clases completo.
Fuente: Sommerville, 2005, p. 167

2.6.1.3. MODELOS DE NAVEGACIÓN

Modelo de navegación, representa la navegación de los objetos dentro de la aplicación y conjunto de estructuras como son: índices, menús y consultas, cuya notación se ilustra en la Figura 2.6. Además, comprende dos etapas, 1) la definición del espacio de navegación, y 2) el diseño de las estructuras de navegación. (Nieves Guerrero, Ucán Pech, & Méndez Domínguez, 2014)

La definición del espacio de navegación se trata de una vista del diagrama conceptual, y se define mediante un diagrama de clases en UML.

El diseño de las estructuras de navegación, establece las estructuras de acceso que permiten visitar los objetos del espacio navegación, y están constituidas por: menús, índices, visitas guiadas y formularios.

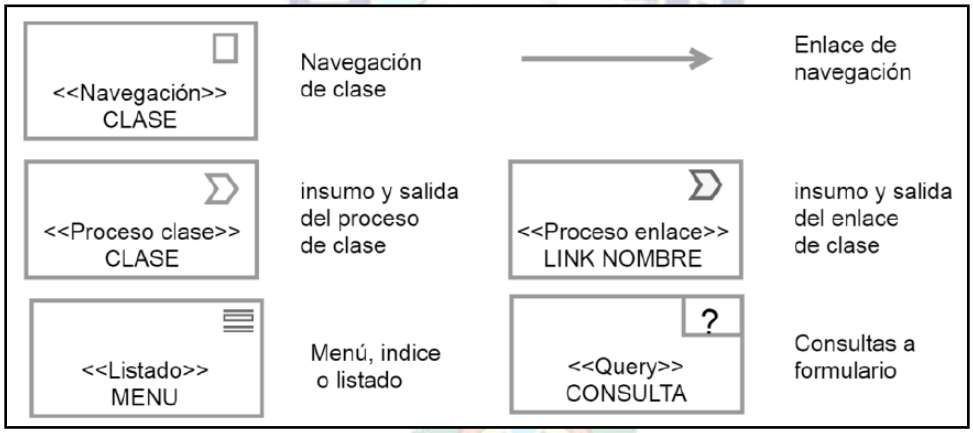


Figura 2.6 Notación de navegación UWE
Fuente: Koch, 2002

2.6.1.4. MODELOS DE PRESENTACIÓN

Modelo de Presentación, representa las interfaces de usuario por medio de vistas abstractas o clases de presentación, y está relacionado con los elementos de las interfaces definidas en HTML, además están definidos como estereotipos UML y cada clase del modelo navegacional tiene asociados un conjunto de clase del modelo de presentación. Los elementos del modelo de presentación son: anclas, entradas de texto, imágenes, audio y botones. (Nieves Guerrero, Ucán Pech, & Méndez Domínguez, 2014)

2.6.1.5. MODELO DE PROCESOS

Modelo de Proceso, representa el aspecto que tienen las actividades que se conectan con cada clase de proceso, y contempla dos tipos de modelos, que son: modelo de estructura de proceso que describe las relaciones entre las distintas clases de proceso, y el modelo del flujo de proceso que especifica las actividades conectadas con cada clase de proceso. (Nieves Guerrero, Ucán Pech, & Méndez Domínguez, 2014)

2.7. TECNOLOGÍAS DE SOFTWARE

Las tecnologías en software que se usarán para el desarrollo del proyecto de grado son:

2.7.1. JAVA 8

Java es un lenguaje de programación y una plataforma informática comercializada por primera vez en 1995 por Sun Microsystems. Hay muchas aplicaciones y sitios web que no funcionarán a menos que tenga Java instalado y cada día se crean más. Java es rápido, seguro y fiable. Desde portátiles hasta centros de datos, desde consolas para juegos hasta súper computadoras, desde teléfonos móviles hasta Internet, Java está en todas partes. (Oracle, 2020)

Java 8 es la versión más reciente de Java que incluye nuevas características, mejoras y correcciones de bugs para mejorar la eficacia en el desarrollo y la ejecución de programas Java. La nueva versión de Java primero se pone a disposición de los desarrolladores para dar tiempo suficiente para realizar las pruebas y certificaciones antes de que esté disponible para su descarga en el sitio web java.com. (Oracle, 2020)

2.7.1.1. PRINCIPALES CARACTERÍSTICAS

Java es un lenguaje de programación de propósito general orientado a objetos desarrollado por Sun Microsystems. También se puede decir que Java es una tecnología que no sólo se reduce al lenguaje, sino que además provee de una máquina virtual Java que permite ejecutar código compilado Java, sea cual sea la plataforma que exista por debajo; plataforma tanto hardware, como software (el sistema operativo que soporte ese hardware). El apoyo a esta tecnología viene dado por la gran cantidad de fabricantes que apoyan esta especificación de máquina virtual.

Aprender el lenguaje de programación Java requiere tiempo y esfuerzo, pero en este curso trataremos de sentar las bases para el conocimiento general del lenguaje. El lenguaje se inspira en otros lenguajes:

- Sentencias comunes de C y C++ (sintaxis parecida a dichos lenguajes)
- Concurrencia parecida a la de Mesa (un lenguaje de investigación de Xerox)
- Interrupciones parecidas a las de Modula-3
- Tratamiento de enlace dinámico de código nuevo parecido al de Lisp
- Definiciones de interfaces parecidas a las de Objective C
- Gestión de almacenamiento automático parecida a la de Lisp

Sun describe al lenguaje Java de la siguiente manera:

- Simple
- Orientado a Objetos
- Tipado estáticamente
- Distribuido
- Interpretado
- Robusto
- Seguro
- de Arquitectura Neutral
- Multihilo
- con Recolector de basura (Garbage Collector)
- Portable
- de Alto Rendimiento: sobre todo con la aparición de hardware especializado y mejor software
- Dinámico

Sun admite que lo dicho anteriormente son un montón de halagos por su parte, pero el hecho es que todas esas características pueden servir para describir el lenguaje. Todas ellas son importantes, sin embargo, cabe destacar tres, que son las que han proporcionado tanto interés por el lenguaje: la portabilidad, el hecho de que sea de arquitectura neutral y su simplicidad. Java

ofrece toda la funcionalidad de los lenguajes potentes, pero sin las características menos usadas y más confusas de éstos.

Java elimina muchas de las características de otros lenguajes como C++, para mantener reducidas especificaciones del lenguaje y añadir características muy útiles como el recolector de basura. No es necesario preocuparse de liberar memoria, el recolector se encarga de eliminar la memoria asignada. Gracias al recolector, sólo te tienes que preocupar de crear los objetos relevantes de tu sistema ya que él se encarga de destruirlos en caso de no ser reutilizados.

Java reduce en un 50% los errores más comunes de programación con lenguajes como C y C++. Entre las características más "indeseables" de C++ que se han evitado en el diseño de Java destacan: ficheros de cabecera, aritmética de punteros, sobrecarga de operadores, estructuras, uniones, conversión implícita de tipos, clases base virtuales, pre-procesador, etc.

2.7.2. ANGULAR 7

Angular es una plataforma y un framework para construir aplicaciones de una sola página en HTML y TypeScript. Angular está escrito en TypeScript e implementa la funcionalidad básica y opcional como un conjunto de librerías TypeScript que se importa a sus aplicaciones. (Google, 2020)

La arquitectura de una aplicación Angular se basa en ciertos conceptos fundamentales:

- Los bloques de construcción básico son NgModules, los cuales proporcionan un contexto de compilación para componentes. NgModules recopila código relacionado en conjuntos funcionales; una aplicación Angular se define mediante un conjunto de NgModules. Una aplicación siempre tiene al menos un módulo raíz que permite el arranque y, por lo general, tiene muchos más módulos de características.

- Los componentes definen vistas, que son conjuntos de elementos de pantalla que Angular puede elegir y modificar de acuerdo con la lógica y los datos de su programa.

- Los componentes usan servicios, que proporcionan una funcionalidad específica que no está directamente relacionada con las vistas. Los proveedores de servicios pueden

inyectarse en los componentes como dependencias, haciendo que su código sea modular, reutilizable y eficiente.

Tanto los componentes como los servicios son simplemente clases, con decoradores que marcan su tipo y proporcionan metadatos que le indican a Angular cómo usarlos.

- Los metadatos para una clase de componente lo asocian con una plantilla que define una vista. Una plantilla combina HTML ordinario con directivas angulares y marcas de enlace que permiten a Angular modificar el HTML antes de mostrarlo.
- Los metadatos para una clase de servicio proporcionan la información que Angular necesita para ponerla a disposición de los componentes a través de la inyección de dependencia (DI).

Los componentes de una aplicación suelen definir muchas vistas, ordenadas jerárquicamente. Angular proporciona el servicio de enrutador para ayudarlo a definir rutas de navegación entre vistas. El enrutador proporciona capacidades de navegación sofisticadas en el navegador.

Las principales características de Angular son:

2.7.2.1. PLATAFORMA CRUZADA

- **APLICACIONES WEB PROGRESIVAS**

  Utiliza capacidades modernas de la plataforma web para ofrecer experiencias similares a las aplicaciones. Instalación de alto rendimiento, fuera de línea sin la necesidad de instalación. (Google, 2020)

- **APLICACIONES NATIVAS**

  Permite la construcción de aplicaciones móviles nativas con estrategias desde Cordoba, Ionic, o Script Nativo. (Google, 2020)

- **APLICACIONES DE ESCRITORIO**

  Permite creación de aplicaciones en los diferentes sistemas operativos, IOS, Windows y Linux usando los mismos métodos para la web, además de contar con las capacidades de acceder a las APIs del sistema operativo nativo. (Google, 2020)

2.7.2.2. VELOCIDAD Y RENDIMIENTO

- **GENERACION DE CODIGO**

  Angular convierte sus plantillas en código altamente optimizado para las máquinas virtuales actuales de JavaScript, brindando todos los beneficios de código escrito a mano con la productividad de un framework. (Google, 2020)

- **UNIVERSALIDAD**

  Compatibilidad de vista para aplicaciones en NodeJS, .NET, PHP, y otros servidores para la representación casi instantánea en solo HTML y CSS. Adaptándose también para sitios que optimizan para SE. (Google, 2020)

2.7.3. POSTGRESQL

PostgreSQL es un servidor de base de datos objeto relacional libre, ya que incluye características de la orientación a objetos, como puede ser la herencia, tipos de datos, funciones, restricciones, disparadores, reglas e integridad transaccional, liberado bajo la licencia BSD. Como muchos otros proyectos open source, el desarrollo de PostgreSQL no es manejado por una sola compañía, sino que es dirigido por una comunidad de desarrolladores y organizaciones comerciales las cuales trabajan en su desarrollo, dicha comunidad es denominada el PGDG (PostgreSQL Global Development Group). (The PostgreSQL Global Development Group, 2020)

2.7.3.1. CARACTERISTICAS

a) DBMS Objeto-Relacional

PostgreSQL aproxima los datos a un modelo objeto relacional, y es capaz de manejar complejas rutinas y reglas. Ejemplos de su avanzada funcionalidad son consultas SQL

declarativas, control de concurrencia multi-versión, soporte multi-usuario, transaccionalidad, optimización de consultas, herencia, y arreglos.

b) Herencia

Las tablas puedes ser configuradas para heredar características de una tabla padre. Los datos son compartidos entre las tablas padre e hija(s). Las tuplas insertadas o eliminadas en la tabla hija serán insertadas o eliminadas en la tabla padre respectivamente.

c) Altamente Extensible

PostgreSQL soporta operadores, funcionales métodos de acceso y tipos de datos definidos por el usuario.

d) Soporte SQL Comprensivo

PostgreSQL soporta la especificación SQL99 e incluye características avanzadas tales como las uniones (*joins*) SQL92.

e) Integridad Referencial

PostgreSQL soporta integridad referencial, la cual es utilizada para garantizar la validez de los datos de la base de datos.

f) API Flexible

La flexibilidad del API de PostgreSQL ha permitido a los vendedores proporcionar soporte al desarrollo fácilmente para el RDBMS PostgreSQL. Estas interfaces incluyen Object Pascal, Python, Perl, PHP, ODBC, Java/JDBC, Ruby, TCL, C/C++, y Pike.

g) Lenguajes Procedurales

PostgreSQL tiene soporte para lenguajes procedurales internos, incluyendo un lenguaje nativo denominado PL/pgSQL. Este lenguaje es comparable al lenguaje procedural de

Oracle, PL/SQL. Otra ventaja de PostgreSQL es su habilidad para usar Perl, Python, o TCL como lenguaje procedural embebido.

h) MVCC

MVCC, o Control de Concurrencia Multi-Versión (Multi-Version Concurrency Control), es la tecnología que PostgreSQL usa para evitar bloqueos innecesarios. Si alguna vez ha usado algún DBMS con capacidades SQL, tal como MySQL o Access, probablemente habrá notado que hay ocasiones en las que una lectura tiene que esperar para acceder a información de la base de datos. La espera está provocada por usuarios que están escribiendo en la base de datos. Resumiendo, el lector está bloqueado por los escritores que están actualizando registros. Mediante el uso de MVCC, PostgreSQL evita este problema por completo. MVCC está considerado mejor que el bloqueo a nivel de fila porque un lector nunca es bloqueado por un escritor. En su lugar, PostgreSQL mantiene una ruta a todas las transacciones realizadas por los usuarios de la base de datos. PostgreSQL es capaz entonces de manejar los registros sin necesidad de que los usuarios tengan que esperar a que los registros estén disponibles.

i) Cliente / Servidor

PostgreSQL usa una arquitectura proceso-por-usuario cliente/servidor. Esta es similar al método del Apache 1.3.x para manejar procesos. Hay un proceso maestro que se ramifica para proporcionar conexiones adicionales para cada cliente que intente conectar a PostgreSQL.

j) Write Ahead Logging (WAL)

La característica de PostgreSQL conocida como *Write Ahead Logging* incrementa la dependencia de la base de datos al registro de cambios antes de que estos sean escritos en la base de datos. Esto garantiza que en el hipotético caso de que la base de datos se caiga, existirá un registro de las transacciones a partir del cual podremos restaurar la base de datos. Esto puede ser enormemente beneficioso en el caso de caída, ya que cualesquiera cambios que no fueron escritos en la base de datos pueden ser recuperados usando el dato

que fue previamente registrado. Una vez el sistema ha quedado restaurado, un usuario puede continuar trabajando desde el punto en que lo dejó cuando cayó la base de datos.

2.7.4. DOCKER

Docker es una plataforma abierta para desarrollar, enviar y ejecutar aplicaciones distribuidas, ya sea en computadoras portátiles, máquinas virtuales de centros de datos o en la nube. Docker ofrece la capacidad de empaquetar y ejecutar una aplicación en un entorno aislado llamado contenedor. La plataforma Docker consta de múltiples productos / herramientas, incluyendo el motor Docker, imágenes, contenedores y *Hub*, entre otros. (Nevada Media, 2018)

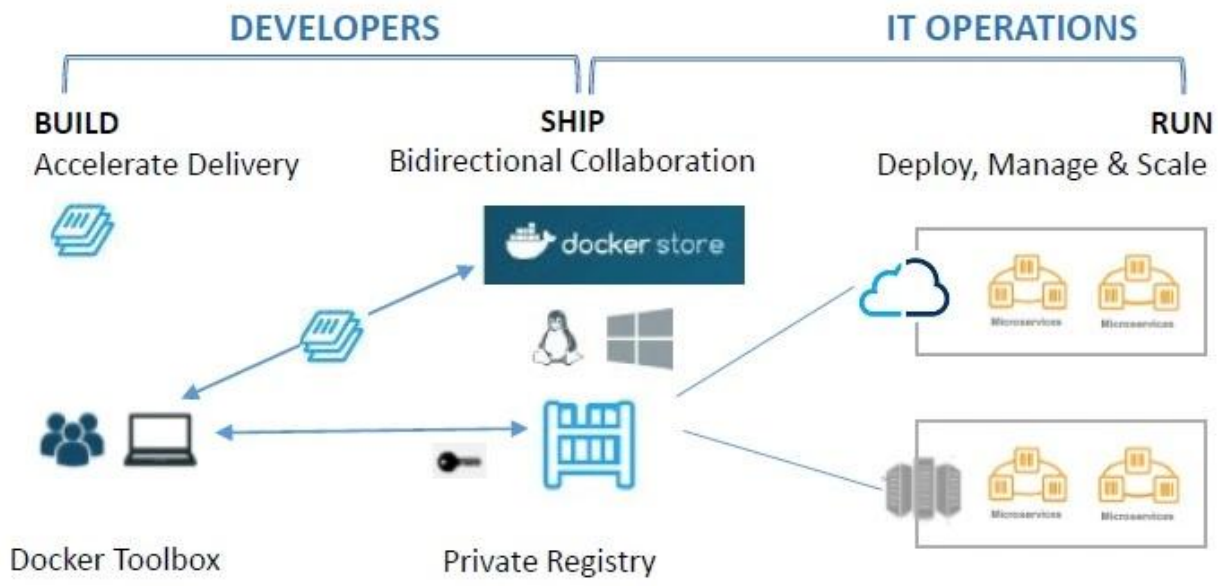


Figura 2.7 Arquitectura Docker
Fuente: Nevada Media, 2018

Para comprender un poco mejor el concepto de contenedor que Docker maneja, haciendo un poco de historia se puede recordar el problema de traslado de carga de todo tipo que existía anteriormente. Trasladar coches, pianos, barriles o cualquier cosa representaba un problema de organización para los buques de la época, se debían establecer reglas complicadas para establecer cierto orden que con tantos artículos era algo difícil de manejar. Para solucionar este problema se crearon los contenedores estandarizados, los cuales son transportados por buques portacontenedores. Cada contenedor es independiente del resto, y dentro de él se establecen ciertas reglas de organización, las cuales permiten el objetivo final, en este caso, transportar de

manera segura ciertos artículos. De igual manera la elaboración de varios de proyectos de software maneja siempre un mismo ambiente de trabajo, de ahí la necesidad de crear contenedores estandarizados independientes y funcionales.

Las ventajas de un entorno virtualizado para el desarrollo de software son muchas, entre ellas:

- Separación del entorno de desarrollo de la configuración de la máquina anfitrión.
- Rápida puesta en marcha de entornos con diferentes configuraciones y aplicaciones de manera rápida y segura.
- Sería posible desarrollar con garantías en un escenario con diferentes desarrolladores con diferentes sistemas operativos.
- Facilita trabajar con proyectos con distintas dependencias.
- Reduce o puede eliminar las inconsistencias entre máquinas de desarrollo y producción. (Nevada Media, 2018)

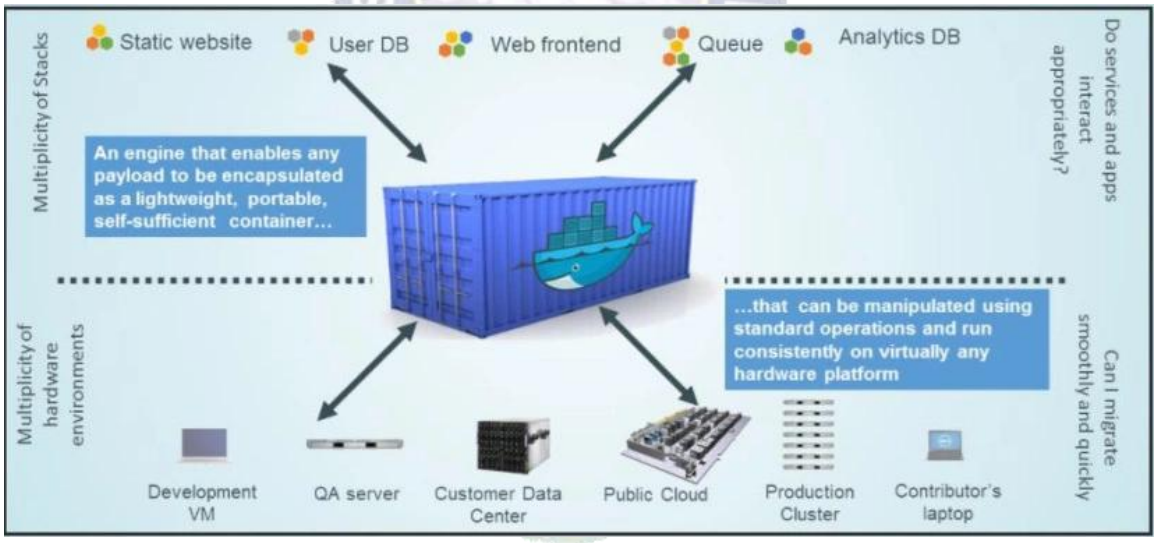


Figura 2.8 Características de un contenedor Docker
Fuente: Nevada Media, 2018

2.7.4.1. CARACTERÍSTICAS PRINCIPALES

a) Portabilidad

Un contenedor Docker es desplegable en cualquier plataforma que soporte esta tecnología, con lo que se ahorra el proceso de configurar un mismo entorno de trabajo en varias estaciones.

b) Ligereza

El peso de este sistema no tiene comparación con cualquier otro sistema de virtualización. Por poner un ejemplo, una de las herramientas de virtualización más extendida es VirtualBox, y cualquier imagen de Ubuntu pesará aproximadamente 1Gb si contamos únicamente con la instalación limpia del sistema. Con Docker, un Ubuntu con Apache y una aplicación web, pesa alrededor de 180Mb, lo que demuestra un significativo ahorro a la hora de almacenar diversos contenedores que podrían ser desplegados en el futuro.

c) Autosuficiencia

Un contenedor Docker no contiene un sistema completo, sino únicamente aquellas librerías, archivos y configuraciones necesarias para desplegar las funcionalidades que contenga. De igual manera Docker se encarga de la gestión del contenedor y de las aplicaciones que dispone.

Para obtener esta fluidez Docker extiende LXC (LinuX Containers), un sistema de virtualización ligero que permite crear múltiples sistemas totalmente aislados entre sí sobre la misma máquina o sistema anfitrión. Y todo dado que no se emula un sistema operativo completo, sólo las librerías y sistemas de archivos necesarios para la utilización de las aplicaciones que están instaladas en cada contenedor.

2.7.5. SERVICIO WEB AMAZON (AMAZON WEB SERVICE)

Amazon Web Services, también conocida como AWS, es un conjunto de herramientas y servicios de Cloud Computing de Amazon. Este servicio se lanzó oficialmente en 2006 y para junio de 2007 AWS ya contaba con una base de usuarios de aproximadamente 180 mil personas. Entre las empresas que la utilizan se encuentran algunas como Reddit, Foursquare, Pinterest, Netflix, la NASA o la CIA, y algunas españolas como Mapfre, el FC Barcelona o Interflora. Esto se debe principalmente a la madurez del servicio frente a otros similares y las posibilidades que ofrece el amplio abanico de herramientas disponibles. En la Guía de Cloud Computing podrá encontrar una comparativa de todas las herramientas de Amazon Web Services con las de otras plataformas similares. (EKCIT, 2019)

2.7.5.1. HERRAMIENTAS DE AMAZON WEB SERVICES

La tendencia general para las plataformas en la nube es la de ofrecer la mayor cantidad posible de herramientas y servicios, para que así se pueda crear todo un entorno de computación en una misma nube. Al igual que otras plataformas como Microsoft Azure, Amazon dispone de una gran cantidad de herramientas para la gestión de diferentes elementos dentro de la empresa. Los servicios de AWS están preparados tanto para autónomos, como pequeñas y medianas empresas o grandes corporaciones, ya que existen posibilidades para escalar las instancias o el almacenamiento según su empresa vaya también creciendo.

Amazon Web Services ofrece herramientas en las siguientes categorías:

- ***Cloud computing*:** todo lo necesario para la creación de instancias y el mantenimiento o el escalado de las mismas. Amazon EC2 es el rey indiscutible dentro de los servicios de computación en la nube de Amazon.

- **Bases de datos:** distintos tipos de bases de datos pueden permanecer en la nube mediante el servicio Amazon RDS, que incluye distintos tipos a elegir como MySQL, PosgreSQL, Oracle, SQL Server y Amazon Aurora, o Amazon DynamoDB para NoSQL.

- **Creación de redes virtuales:** permite la creación de redes privadas virtuales a través de la nube, gracias principalmente al servicio Amazon VPC.

- **Aplicaciones empresariales:** Amazon WorkMail es el servicio de correo empresarial que ofrece Amazon, al que pueden unirse otros servicios como Amazon WorkDocs y Amazon WorkSpaces.

- **Almacenamiento y gestores de contenido:** tipos de almacenamiento diferentes, tanto para archivos con acceso regular, poco frecuente o incluso como archivo. Amazon S3 es el servicio principal, aunque complementan la oferta otros como Amazon Glacier o Amazon EBS.

- **Inteligencia de negocios o *Business Intelligence* (BI):** sistemas para análisis de datos empresariales a gran escala y otros servicios para la gestión de flujos de datos.

- **Gestión de aplicaciones móviles:** herramientas como Amazon Mobile Hub permiten la gestión, creación, testeo y mantenimiento de aplicaciones móviles a través de la nube.

- **Internet de las cosas *(Internet of Things*):** para establecer conexiones y análisis de todos los dispositivos conectados a internet y los datos recogidos por los mismos.

- **Herramientas para desarrolladores:** para almacenar código, implementarlo automáticamente o incluso publicar software mediante un sistema de entrega continua.

- **Seguridad y control de acceso:** se pueden establecer autenticaciones en varios pasos para poder proteger el acceso a sus sistemas internos, ya están en la nube o instalados de forma local en sus instalaciones.

2.8. CALIDAD DE SOFTWARE

2.8.1. MÉTRICAS DE CALIDAD

Cada métrica debe validarse empíricamente en una amplia variedad de contextos antes de publicarse o aplicarse la toma de decisiones. Una métrica debe medir el factor de interés, independientemente de otros factores. Debe crecer para aplicarse a sistemas grandes y funcionar en diversos lenguajes de programación y dominios de sistemas. Aunque la formulación, caracterización y validación son críticas, la recopilación y el análisis son las actividades que dirigen el proceso de medición, el siguiente esquema atiende a las cuatro más importantes en el desarrollo de software.

a) **Métricas para el modelo de análisis.** Estas métricas atienden varios aspectos de la etapa de análisis en donde se incluyen:
    - Funcionalidad entregada. Proporciona una medida indirecta de la funcionalidad que se empaqueta con el software.
    - Tamaño del sistema. Mide el tamaño general del sistema, definido desde el punto de vista de la información disponible como parte del modelo de análisis.
    - Calidad de la especificación. Proporciona un indicador específico o el grado en que se ha completado la especificación de los requisitos.

b) **Métricas para el modelo de diseño**. Estas métricas cuantifican los atributos del diseño de manera tal que le permiten al ingeniero de software evaluar la calidad del diseño, la métrica incluye:
    - Métricas arquitectónicas. Proporcionan un indicio de la calidad del diseño arquitectónico.
    - Métricas al nivel de componente. Mide la complejidad de los componentes del software y otras características que impactan la calidad.
    - Métricas de diseño de la interfaz. Se concentran principalmente en la facilidad de uso.
    - Métricas especializadas en diseño orientado a objetos. Miden características de clases, además de las correspondientes a comunicación y colaboración.
c) **Métricas para el código fuente.** Estas métricas miden el código fuente y se usan para evaluar su complejidad, además de la facilidad con que se mantiene y prueba entre otras características como:
    - Métricas de complejidad. Miden la complejidad lógica del código fuente.
    - Métricas de longitud. Proporcionan un indicio del tamaño del software.
d) **Métricas para pruebas.** Estas métricas ayudan a diseñar casos de prueba efectivos y evaluar la eficacia de las pruebas en donde se incluyen:
    - Métricas de cobertura de instrucciones y ramas. Lleva al diseño de casos de prueba que proporcionan cobertura del programa.
    - Métricas relacionadas con los defectos. Se concentran en encontrar defectos y no en las propias pruebas.
    - Efectividad de la prueba. Proporciona un indicio en tiempo real de la efectividad y de las pruebas aplicadas.
    - Métricas en el proceso. Métrica relacionada con el proceso de las pruebas.

2.8.2. ISO 9126

Para determinar la calidad del producto, el estándar internacional para la evaluación de software ISO 9126, esta norma internacional fue publicada en 1992, la cual es usada para la evaluación de la calidad de software.

La normativa define características de la aplicación, estas seis características se dividen en un número de sub – características, los cuales representan un modelo detallado para la evaluación de cualquier sistema informático.

El modelo establece diez características, seis que son comunes a las vistas interna y externa, y los restantes cuatro que son propias de la vista en uso. A continuación, se presenta las características y sub – características propias de este estándar que se encuentran dentro de las vistas interna y externas, que fueron usadas para evaluar el software. (Figueroa, 2000)

a) Funcionalidad

En este grupo se conjunta una serie de atributos que permiten calificar si un producto de software maneja en forma adecuada el conjunto de funciones que satisfagan las necesidades para las cuales fue diseñado. Para este propósito se establecen las siguientes características:

- Adecuación: Se enfoca a evaluar si el software cuenta con un conjunto de funciones apropiadas para efectuar las tareas que fueron especificadas en su definición.

- Exactitud. -: Este atributo evalúa el resultado final, observa si el software presenta resultados o efectos acordes a las necesidades para las cuales fue creado.

- Interoperabilidad. - Consiste en revisar si el sistema puede interactuar con otro sistema independiente.

- Seguridad. - Se refiere a la habilidad de prevenir el acceso no autorizado, ya sea accidental o predeterminado, a los programas y datos.

b) Fiabilidad

En esta parte se refiere a la capacidad del software de mantener las prestaciones requeridas del sistema, durante un tiempo establecido y bajo un conjunto de condiciones definidas. Las sub características que el estándar sugiere son:

- Madurez: Permite medir la frecuencia de falla por errores de software. Se debe verificar si estas han sido eliminadas durante el tiempo de pruebas o uso del sistema.

- Recuperabilidad: Se refiere a la capacidad de restablecer el nivel de operación y recobrar los datos que hayan sido afectados directamente por una falla.

- Tolerancia a fallos: Se refiere a la habilidad de mantener un nivel específico de funcionamiento en caso de fallas del software.

c) Usabilidad

Consiste de un conjunto de atributos que permite evaluar el esfuerzo necesario que deberá invertir el usuario para utilizar el sistema.

- Comprensibilidad: Se refiere al esfuerzo requerido por los usuarios para reconocer la estructura lógica del sistema y los conceptos relativos a la aplicación del software.

- Facilidad de Aprender: Establece atributos del software relativos al esfuerzo que los usuarios deben hacer para aprender a usar la aplicación.

- Operabilidad: Agrupa los conceptos que evalúan la operación y el control de sistema.

d) Eficiencia

Esta característica permite evaluar la relación entre el nivel de funcionamiento del software y la cantidad de recursos usados. Los aspectos a evaluar son:

- Comportamiento con respecto al tiempo: Atributos del software relativo a los tiempos de respuesta y de procesamiento de los datos.

- Comportamiento con respecto a recursos: Atributos del software relativos a la cantidad de recursos usados en la realización de sus funciones.

e) Mantenibilidad

Se refiere a los atributos que permiten medir el esfuerzo necesario para adaptarse a las nuevas especificaciones y requisitos del software.

- Estabilidad. - Permite evaluar los riesgos de efectos inesperados debidos a las modificaciones realizadas al software.
- Capacidad de análisis. - Relativo al esfuerzo necesario para diagnosticar las deficiencias o causas de fallas, o para identificar las partes que deberán ser modificadas.
- Capacidad de modificación. - Mide el esfuerzo necesario para modificar aspectos del software, remover fallas o adaptar el software para que funciones en un ambiente diferente.
- Facilidad de prueba. - Se refiere al esfuerzo necesario para validar el software una vez que fue modificado.

e) Portabilidad

En este caso, se refiere a la habilidad del software de ser transferido de un ambiente a otro, y considera los siguientes aspectos:

- Adaptabilidad. - Evalúa la oportunidad para adaptar el software a diferentes ambientes.
- Facilidad de instalación. - Es el esfuerzo necesario para instalar el software en un ambiente determinado.
- Conformidad. - Permite evaluar si el software se adhiere a estándares o convenciones relativas a portabilidad.
- Capacidad de reemplazo. - Se refiere a la oportunidad y el esfuerzo usado en sustituir el software por otro producto con funciones similares.

2.8.3 MÉTRICAS Y MEDICIÓN

La medición es un elemento clave en cualquier proceso de ingeniería. Las medidas se emplean para comprender mejor los atributos de los modelos que se crean y evaluar la calidad de los productos de la ingeniería. Por las características inherentes al software, sus medidas y métricas son indirectas y, por lo tanto, expuestas al debate. (Pressman, 2010, p. 545)

Una métrica contiene la definición de un método de medición o un método de cálculo y la escala asociada. El método de medición es la secuencia lógica particular de operaciones y posibles heurísticas, especificada para permitir la realización de la descripción de una métrica por una actividad de medición. Por otro lado, la escala se define como un conjunto de valores con propiedades definidas. La propiedad más importante de una escala es su tipo, considerando que puede ser Categórica o Numérica. A su vez, dependiendo de la naturaleza de la relación entre los componentes de la escala, pueden clasificarse en: nominal, ordinal, intervalo, proporción o absoluta. El tipo de escala de los valores medidos define las transformaciones admisibles y afecta las operaciones matemáticas y estadísticas que pueden ser aplicadas. Las métricas pueden ser directas, sobre las que puede aplicarse un método de medición (objetivo o subjetivo); o indirectas, que son aquellas definidas en función de otras métricas y se calculan en base al método de cálculo asociado, es decir en base a una fórmula.

a) Calidad en aplicaciones Web

Los avances en Internet han conducido a un desarrollo impactante de sistemas y aplicaciones basadas en la Web, suceso que se presenta como el más significativo en la historia de la Computación Muchas de las nuevas tecnologías y estándares de la Web han surgido en los últimos años para mejorar el apoyo a nuevas aplicaciones Web: XML, servicios Web, Web semántica, técnicas de personalización de la Web, minería Web, inteligencia, *context aware* y móviles y servicios Web. Las aplicaciones web son diferentes de otras categorías de software; son eminentemente de red, las gobiernan los datos y se encuentran en evolución continua. La inmediatez dirige su desarrollo, la seguridad es un requisito prioritario y la demanda de estética, así como la entrega de contenido funcional, son factores diferenciales adicionales. (Pressman, 2010)

El estudio de la calidad de productos y procesos de desarrollo para la Web es muy reciente y todavía no se dispone de métodos de evaluación ampliamente difundidos para este tipo de entorno, por lo tanto, existe la necesidad de metodologías efectivas para la obtención de aplicaciones Web de calidad. La Ingeniería Web surge debido a la necesidad de lograr enfoques disciplinados y nuevos métodos y herramientas para desarrollar, desplegar y evaluar los sistemas y aplicaciones basados en la Web. Estos enfoques y técnicas deben considerar las particularidades del nuevo medio, el contexto y los escenarios operativos y, principalmente, la diversidad de perfiles de usuarios que constituyen desafíos adicionales al desarrollo de aplicaciones Web.

¿Cómo se mide la calidad del software para la web? En general, con los mismos modelos que para el software tradicional. Sin embargo, hay características que son más relevantes en este contexto, como, por ejemplo, la facilidad de uso, funcionalidad, confiabilidad, eficiencia y facilidad de mantenimiento. Olsina (1999) define un "árbol de requisitos de calidad" para aplicaciones Web y Offut (2002) agrega otros atributos como Seguridad, Disponibilidad, Escalabilidad, Tiempo en el Mercado.

b) Calidad en Programación Orientada a objetos

El desarrollo de programas orientados a objetos (POO) es cada vez mayor, sin embargo, no ha evolucionado al mismo ritmo el uso de métricas para este paradigma. Las métricas orientadas a objetos, al igual que las del software convencional, buscan poder entender mejor la calidad del producto, evaluar la efectividad del proceso y mejorar la calidad del trabajo llevado a cabo al nivel del proyecto. Sin embargo, la POO difiere en importante medida del desarrollado utilizando enfoques tradicionales. Por esta razón las métricas deben ajustarse a las características que lo distinguen, como ser encapsulamiento, ocultamiento de información, herencia y técnicas de abstracción de objetos que hagan única a esa clase. Entre las métricas encontradas en la literatura que han tenido relevancia en la orientación a objetos sobresalen las definidas por Abreu, Chidamber, Kemerer, Lorenz y Kidd, que abordan todos los posibles niveles de granularidad y características en sistemas OO, como ser:

- Métricas a nivel de sistema
- Métricas a nivel de acoplamiento

- Métricas a nivel de herencia
- Métricas a nivel de clases
- Métricas a nivel de métodos (Rodríguez & Harrison, 2005)

2.9. COSTO BENEFICIO DEL SISTEMA

2.9.1. MÉTODO DE ESTIMACIÓN COCOMO II

El modelo original COCOMO (*Constructive Cost Model*) se publicó por primera vez en 1981 por Barry Boehm y reflejaba las prácticas en desarrollo de software de aquel momento. En la década y media siguiente las técnicas de desarrollo software cambiaron drásticamente. Estos cambios incluyen el gasto de tanto esfuerzo en diseñar y gestionar el proceso de desarrollo software como en la creación del producto software, un giro total desde los mainframes que trabajan con procesos *batch* nocturnos hacia los sistemas en tiempo real y un énfasis creciente en la reutilización de software ya existente y en la construcción de nuevos sistemas que utilizan componentes software a medida. (Boehm, 2012)

Estos y otros cambios hicieron que la aplicación del modelo COCOMO original empezara a resultar problemática. La solución al problema era reinventar el modelo para aplicarlo a los 90. Después de muchos años de esfuerzo combinado entre USC-CSE1, IRUS y UC Irvine22 y las Organizaciones Afiliadas al Proyecto COCOMO II, el resultado es COCOMO II, un modelo de estimación de coste que refleja los cambios en la práctica de desarrollo de software profesional que ha surgido a partir de los años 70. Este nuevo y mejorado COCOMO resultará de gran ayuda para los estimadores profesionales de coste software. Por tanto, COCOMO II es un modelo que permite estimar el coste, esfuerzo y tiempo cuando se planifica una nueva actividad de desarrollo software. El principal cálculo en el modelo COCOMO es el uso de la ecuación del esfuerzo para estimar el número de personas o de meses necesarios para desarrollar el proyecto. El resto de resultados del modelo se derivan de esta medida.

Por un lado, COCOMO define tres modos de desarrollo o tipos de proyectos:

• Orgánico: proyectos relativamente sencillos, menores de 50 KDLC líneas de código, en los cuales se tiene experiencia de proyectos similares y se encuentran en entornos estables.

- Semi-acoplado: proyectos intermedios en complejidad y tamaño (menores de 300 KDLC), donde la experiencia en este tipo de proyectos es variable, y las restricciones intermedias.

- Empotrado: proyectos bastante complejos, en los que apenas se tiene experiencia y se engloban en un entorno de gran innovación técnica. Además, se trabaja con unos requisitos muy restrictivos y de gran volatilidad.

Y por otro lado existen diferentes modelos que define COCOMO:

- Modelo básico: Se basa exclusivamente en el tamaño expresado en LDC.

- Modelo intermedio: Además del tamaño del programa incluye un conjunto de medidas subjetivas llamadas conductores de costes.

- Modelo avanzado: Incluye todo lo del modelo intermedio además del impacto de cada conductor de coste en las distintas fases de desarrollo. La función básica que utilizan los tres modelos es:

$$E = a(Kl)b * m(X)$$

Dónde:

- a y b son constantes con valores definidos en cada submodelo.

- Kl es la cantidad de líneas de código, en miles.

- m(X) Es un multiplicador que depende de 15 atributos.

El resultado se da en unidades salario/mes y horas-hombre.

Modelo básico. Se utiliza para obtener una primera aproximación rápida del esfuerzo, y hace uso de la siguiente tabla de constantes para calcular distintos aspectos de costes:

MODO	a	b	c	d
Orgánico	2.40	1.05	2.50	0.38
Semi-libre	3.00	1.12	2.50	0.35
Rígido	3.60	1.20	2.50	0.32

Tabla 2.2 Estimación de Esfuerzo
Fuente: Grupo de Investigación de costos (Beltran, 2008)

Estos valores son para las fórmulas:

- Personas necesarias por mes para llevar adelante el proyecto (MM) = a*(Klb)
- Tiempo de desarrollo del proyecto (TDEV) = c*(MMd)
- Personas necesarias para realizar el proyecto (CosteH) = MM/TDEV
- Costo total del proyecto (CosteM) = CosteH * Salario medio entre los programadores y analistas.

Se puede observar que a medida que aumenta la complejidad del proyecto (modo), las constantes aumentan de 2.4 a 3.6, que corresponde a un incremento del esfuerzo del personal. Hay que utilizar con mucho cuidado el modelo básico puesto que se obvian muchas características del entorno.

Atributos. Cada atributo se cuantifica para un entorno de proyecto. La escala es muy baja - baja - nominal - alta- muy alta - extremadamente alto. Dependiendo de la calificación de cada atributo, se asigna un valor para usar de multiplicador en la fórmula (por ejemplo, si para un proyecto el atributo DATA es calificado como muy alto, el resultado de la fórmula debe ser multiplicado por 1000).

El significado de los atributos es el siguiente, según su tipo:

Software

- RELY: garantía de funcionamiento requerida al software. Indica las posibles consecuencias para el usuario en el caso que existan defectos en el producto. Va desde la sola inconveniencia de corregir un fallo (muy bajo) hasta la posible pérdida de vidas humanas (extremadamente alto, software de alta criticidad).

- DATA: tamaño de la base de datos en relación con el tamaño del programa. El valor del modificador se define por la relación: D / K, donde D corresponde al tamaño de la base de datos en bytes y K es el tamaño del programa en cantidad de líneas de código.

- CPLX: representa la complejidad del producto.

Hardware

- TIME: limitaciones en el porcentaje del uso de la CPU.

- STOR: limitaciones en el porcentaje del uso de la memoria.

- VIRT: volatilidad de la máquina virtual.

- TURN: tiempo de respuesta requerido.

Personal

- ACAP: calificación de los analistas.

- AEXP: experiencia del personal en aplicaciones similares.

- PCAP: calificación de los programadores.

- VEXP: experiencia del personal en la máquina virtual.

- LEXP: experiencia en el lenguaje de programación a usar.

Proyecto

- MODP: uso de prácticas modernas de programación.
- TOOL: uso de herramientas de desarrollo de software.
- SCED: limitaciones en el cumplimiento de la planificación.

El valor de cada atributo, de acuerdo a su calificación, se muestra en la siguiente tabla:

Atributos	Valor					
	Muy bajo	Bajo	Nominal	Alto	Muy alto	Extra alto
Atributos de software						
Fiabilidad	0,75	0,88	1,00	1,15	1,40	
Tamaño de Base de datos		0,94	1,00	1,08	1,16	
Complejidad	0,70	0,85	1,00	1,15	1,30	1,65
Atributos de hardware						
Restricciones de tiempo de ejecución			1,00	1,11	1,30	1,66
Restricciones de memoria virtual			1,00	1,06	1,21	1,56
Volatilidad de la máquina virtual		0,87	1,00	1,15	1,30	
Tiempo de respuesta		0,87	1,00	1,07	1,15	
Atributos de personal						
Capacidad de análisis	1,46	1,19	1,00	0,86	0,71	
Experiencia en la aplicación	1,29	1,13	1,00	0,91	0,82	
Calidad de los programadores	1,42	1,17	1,00	0,86	0,70	
Experiencia en la máquina virtual	1,21	1,10	1,00	0,90		
Experiencia en el lenguaje	1,14	1,07	1,00	0,95		
Atributos del proyecto						

Técnicas actualizadas de programación	1,24	1,10	1,00	0,91	0,82	
Utilización de herramientas de software	1,24	1,10	1,00	0,91	0,83	
Restricciones de tiempo de desarrollo	1,23	1,08	1,00	1,04	1,10	

Tabla 2.3 Valores de los factores de escala
Fuente: Modelos de estimación (Beltran, 2008)

2.10. SEGURIDAD

Los problemas de seguridad en sistemas web, pueden venir de las herramientas que se utilizan en el momento del desarrollo o producto de un diseño lógico que no se contempló de las posibles amenazas que pueda surgir como ser (Meucci, 2008):

- Entradas no válidas.
- Control de accesos rotos.
- Sesiones y Autenticaciones no controladas.
- Ataques Cross Site Scripting.
- Inyección de códigos.
- Manejo Inadecuado de Errores

Para las medidas de seguridad para el sistema desarrollado se contemplan dos aspectos importantes y vulnerables que están en el lado del cliente y lado del servidor.

2.10.1. SEGURIDAD DEL LADO DEL CLIENTE

Uno de los mecanismos de seguridad que se implementan son las validaciones por el lado del cliente. Existen mecanismos de validación provistos por las herramientas que utilizamos para hacer la aplicación, HTML cuenta con atributos para validar datos requeridos, numéricos, formato de correos, etc. estas validaciones son realizadas antes de que la información introducida llegue al servidor, esto evita que se envíen datos incorrectos al servidor, además se ahorra

tiempo, ya que si la información es incorrecta simplemente no se envía al servidor. (Meucci, 2008)

Las medidas que se implementó en el lado del servidor del sistema es la autenticación de usuarios, los únicos que tienen acceso al sistema son el personal de la empresa, estos están registrados como usuarios y con su contraseña respectiva.

2.10.2. SEGURIDAD DEL LADO DEL SERVIDOR

El desarrollo de una aplicación web requiere de una serie de herramientas: servidores web, servidores de base de datos, servidores de aplicaciones, lenguajes de programación del lado del servidor, etc. Las vulnerabilidades mencionadas que pueden comprometer la seguridad de un sistema web son las siguientes (Meucci, 2008):

- Versiones no actualizadas.
- Configuraciones por defecto inadecuadas.
- Cuentas por defecto no modificadas.

CAPITULO III

MARCO APLICATIVO

3.1. INTRODUCCIÓN

En este capítulo se describe la especificación de requerimientos, los roles del sistema, el diseño del software Web como ser: el modelo de casos de uso, el modelo de contenido, el modelo de navegación, el modelo de presentación y el modelo de procesos.

Además, en el desarrollo de este capítulo se analiza la combinación de la metodología ágil Kanban con la metodología UWE, y la forma en la que se utilizó dicha combinación en la planificación y desarrollo de la plataforma SIP para el pago de servicios mediante códigos QR para la empresa MC4 S.R.L.

3.2 ANÁLISIS Y DISEÑO DE SOFTWARE

A esta fase del desarrollo del proyecto se denomina Pre-game, en esta se describirán los requerimientos, los roles y su función, el diseño de software a través de diagramas UML para la Web.

Para el control de avance del desarrollo se utilizó la herramienta Trello, la cual es ideal para la coordinación de equipos de trabajo y de revisión de avance para la metodología Scrum.

El proyecto se dividió en 3 sprints:

1.- Modulo de Administración y Seguridad

2.- Módulo de Integración y Transacciones

3.- Pruebas Integrales

En la Figura 3.1 se puede apreciar la definición de las tareas definidas para el desarrollo de esta aplicación con el detalle interior llamado *Checklist*, dichas tareas fueron agregadas a medida que se desarrolla el Sprint inicial.

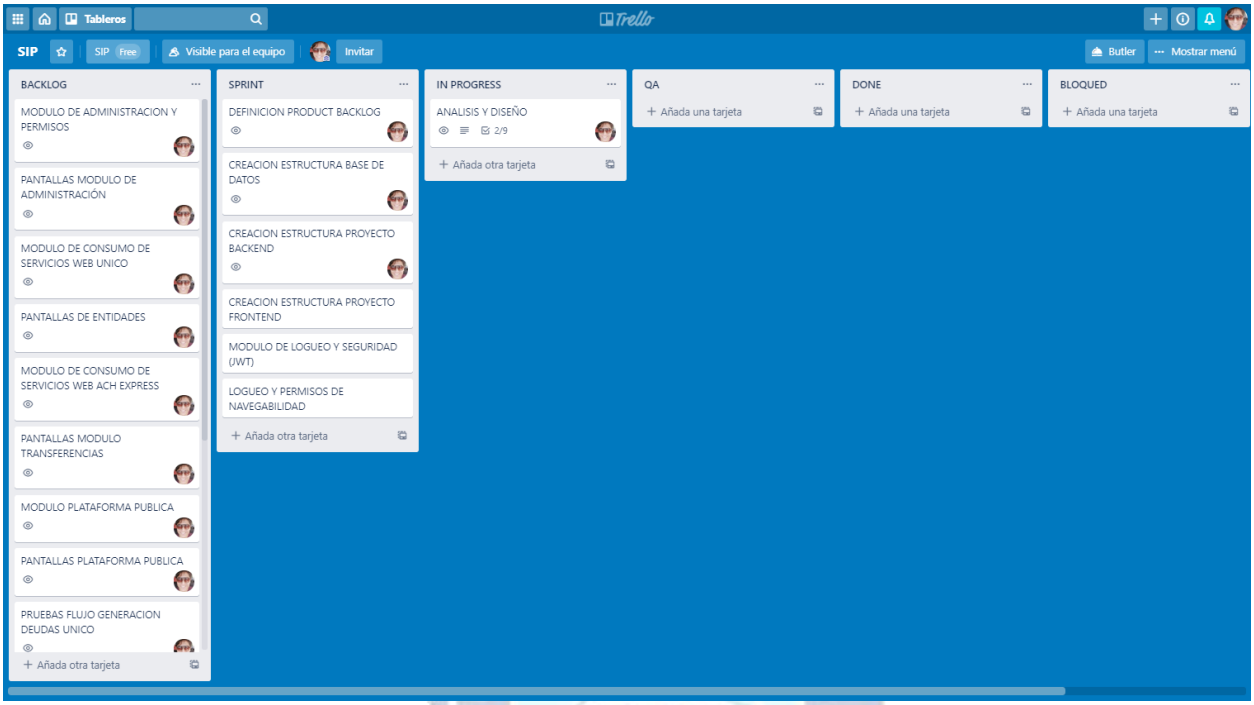


Figura 3.1: Herramienta Trello de control de avance
Fuente: Elaboración propia

3.3 MÓDULO DE ADMINISTRACIÓN (SPRINT 1)

En el Sprint 1 se definieron las siguientes tareas:

1. Análisis y Diseño.
2. Creación Estructura Base de Datos.
3. Creación Estructura Proyecto Back-end
4. Creación Estructura Proyecto Front-end
5. Generación de token con JWT
6. Logueo y Permisos de Navegabilidad
7. Módulo de Administración y Permisos
8. Pantallas Modulo de Administración

En la figura 3.2 se visualiza las tareas definidas para el 1er sprint

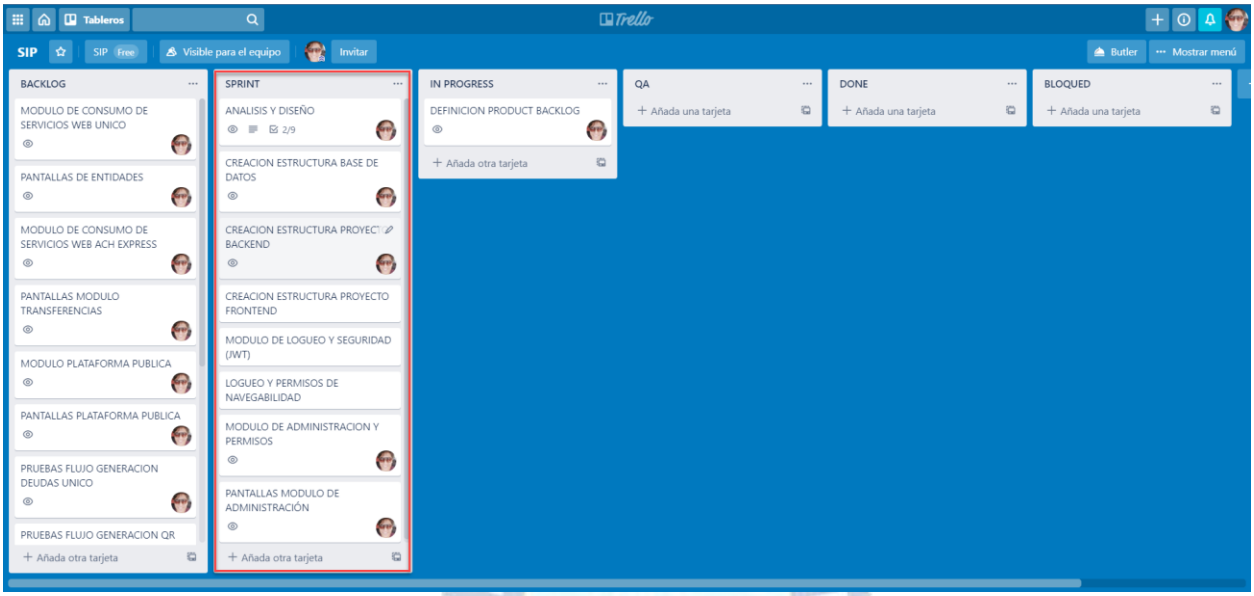


Figura 3.2. Herramienta Trello Sprint 1
Fuente: Elaboración Propia

3.3.1 FASE DE ANÁLISIS

3.3.1.1 ANÁLISIS DE REQUERIMIENTOS

La especificación de requerimientos constituye una base sólida para el diseño y la construcción del software (Pressman, 2010), sin esta el software resultante tiene alta probabilidad de no satisfacer las necesidades de los clientes.

a) Requerimientos Funcionales

Especifica los servicios que proveerá el sistema (funciones específicas del sistema). En algunos casos, también declaran explícitamente lo que el sistema no debe hacer.

RF.01 El Logueo al sistema debe realizarse utilizando token con JWS.

RF.02 El gestor de Base de datos debe ser Postgresql y ser desplegado en un contenedor de Docker en AWS.

RF.03 El Back-end debe desarrollarse en Java utilizando el framework Spring Boot.

RF.04 El Front-end debe desarrollarse en Angular 7 usando componentes de Material UI

RF.05 El administrador del sistema crea, edita, habilita, inhabilita y consulta datos de los usuarios.

RF.06 El administrador del sistema crea, edita, elimina, asigna roles y consulta datos de los roles de usuario.

RF.07 El administrador del sistema crea, edita, elimina, y consulta datos de empresas.

RF.08 El administrador del sistema crea, edita, habilita, inhabilita y consulta datos de empresa.

RF.09 El administrador del sistema consulta datos de las transacciones.

RF.10 El administrador del sistema crea, edita, habilita, inhabilita y consulta datos de los usuarios.

3.3.1.2 CASOS DE USO

La figura 3.3 muestra el caso de uso para administración usuario:

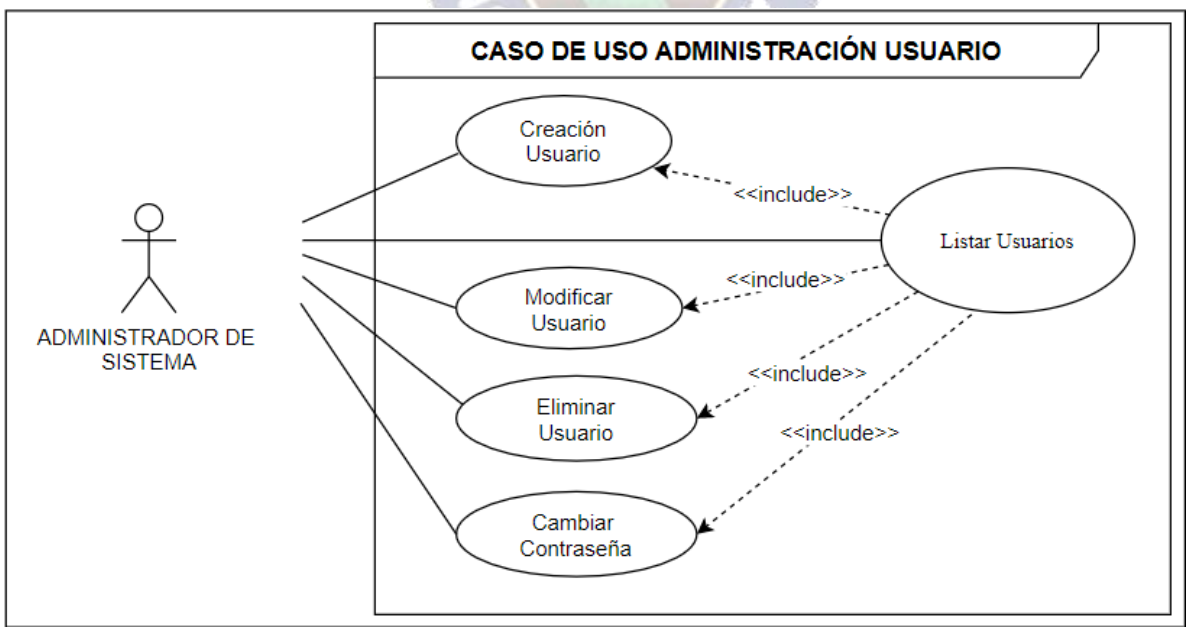


Figura 3.3. Caso de uso administración usuario.
Fuente: Elaboración propia

La tabla 3.1 detalla las especificaciones del caso de uso administración de usuario:

Caso de uso	Administración de usuario
Actores	Administrador de sistema
Descripción	Permite al administrador de sistema crear, modificar, eliminar, cambiar contraseña y ver el listado de usuarios.
Precondiciones	El actor debe ser un usuario registrado y con el rol asignado de administrador del sistema.
Flujo Normal	1. El actor ingresa a la opción de usuarios 2. El sistema despliega el listado de usuarios registrados, cada uno con las opciones de crear, modificar, eliminar y cambiar contraseña. 3. El actor escoge una de las acciones.
Postcondiciones	El sistema ejecuta la opción elegida por el actor y muestra los cambios realizados.

Tabla 3.1 Especificación del caso de uso administración usuario
Fuente Elaboración propia.

La figura 3.4 muestra el caso de uso para administración rol:

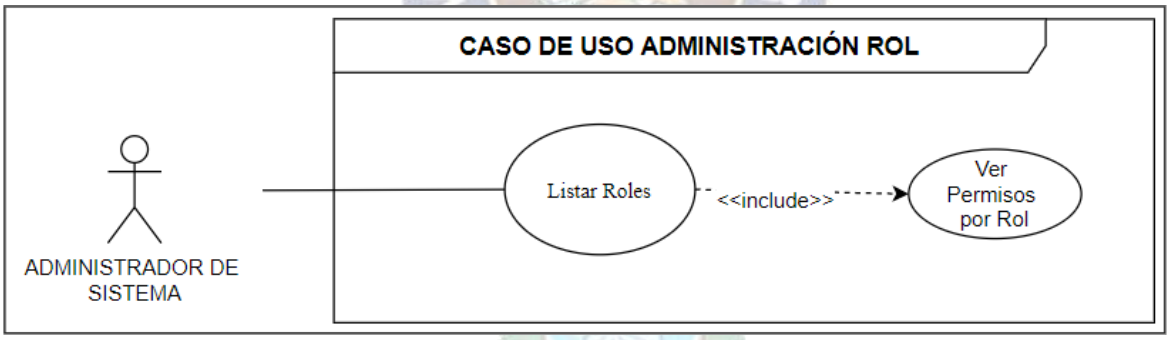


Figura 3.4 Caso de uso administración rol.
Fuente: Elaboración propia

La tabla 3.2. detalla las especificaciones del caso de uso administración rol:

Caso de uso	Administración de rol
Actores	Administrador de sistema
Descripción	Permite al administrador de sistema ver el listado de roles y los permisos correspondientes a cada rol.
Precondiciones	El actor debe ser un usuario registrado y con el rol asignado de administrador del sistema.

Flujo Normal	1. El actor ingresa a la opción de roles. 2. El sistema despliega el listado de roles registrados, cada uno con la opción de ver permisos. 3. El actor escoge una de las acciones.
Postcondiciones	El sistema ejecuta la opción elegida por el actor.

Tabla 3.2 Especificación del caso de uso Administración Rol
Fuente Elaboración propia.

La figura 3.5 muestra el caso de uso para administración parámetros:

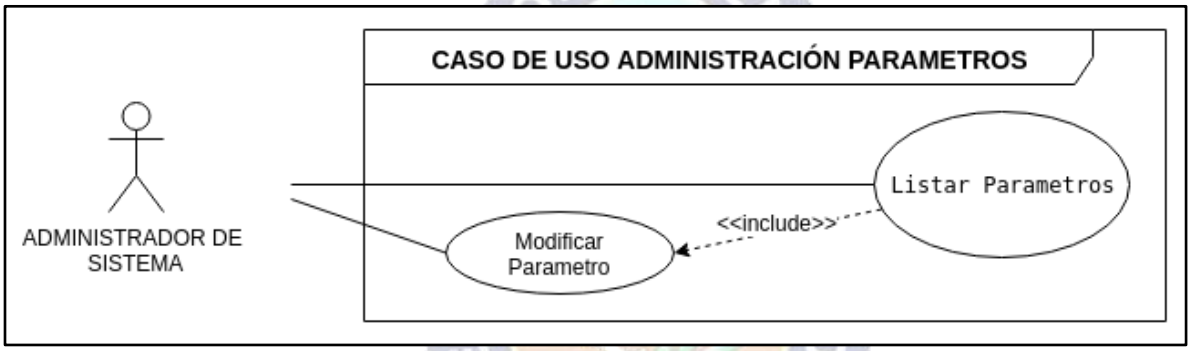


Figura 3.5 Caso de uso administración parámetros.
Fuente: Elaboración propia

La tabla 3.3 detalla las especificaciones del caso de uso administración parámetros:

Caso de uso	Administración de parámetros
Actores	Administrador de sistema
Descripción	Permite al administrador de sistema modificar y ver el listado de parámetros.
Precondiciones	El actor debe ser un usuario registrado y con el rol asignado de administrador del sistema.
Flujo Normal	1. El actor ingresa a la opción de parámetros. 2. El sistema despliega el listado de parámetros registrados, cada uno con las opciones modificar. 3. El actor escoge una de las acciones.
Postcondiciones	El sistema ejecuta la opción elegida y muestra los cambios realizados.

Tabla 3.3 Especificación del caso de uso administración parámetros

Fuente Elaboración propia.

La figura 3.6 muestra el caso de uso para administración de categorías:

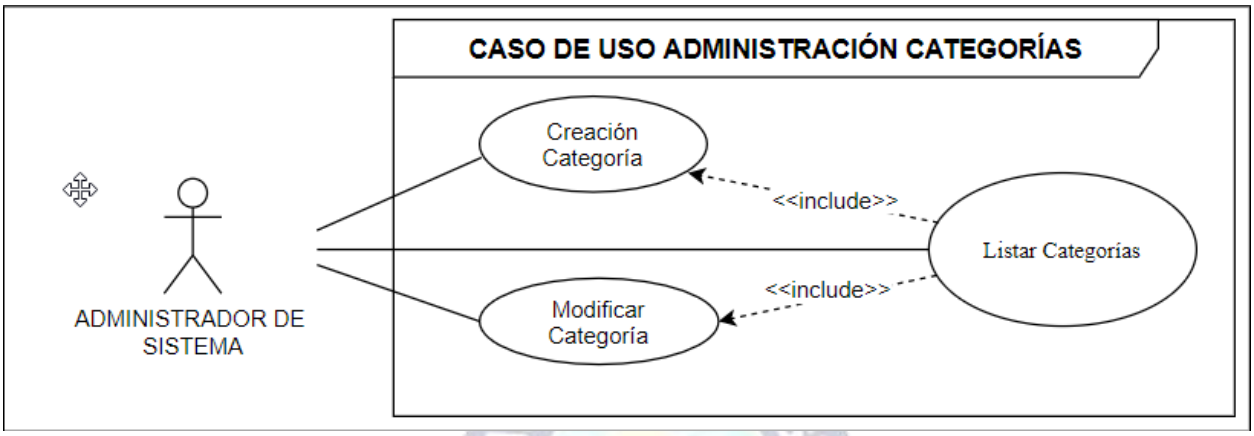


Figura 3.6 Caso de uso administración categorías.
Fuente: Elaboración propia

La tabla 3.4 detalla las especificaciones del caso de uso administración de categorías:

Caso de uso	Administración de categorías
Actores	Administrador de sistema
Descripción	Permite al administrador de sistema crear, modificar, y ver el listado de categorías.
Precondiciones	El actor debe ser un usuario registrado y con el rol asignado de administrador del sistema.
Flujo Normal	1. El actor ingresa a la opción de usuarios 2. El sistema despliega el listado de categorías registradas, cada una con las opciones de crear y modificar. 3. El actor escoge una de las acciones.
Postcondiciones	El sistema ejecuta la opción elegida por el actor y muestra los cambios realizados.

Tabla 3.4 Especificación del caso de uso administración categorías
Fuente Elaboración propia.

La figura 3.7 muestra el caso de uso para transacciones:

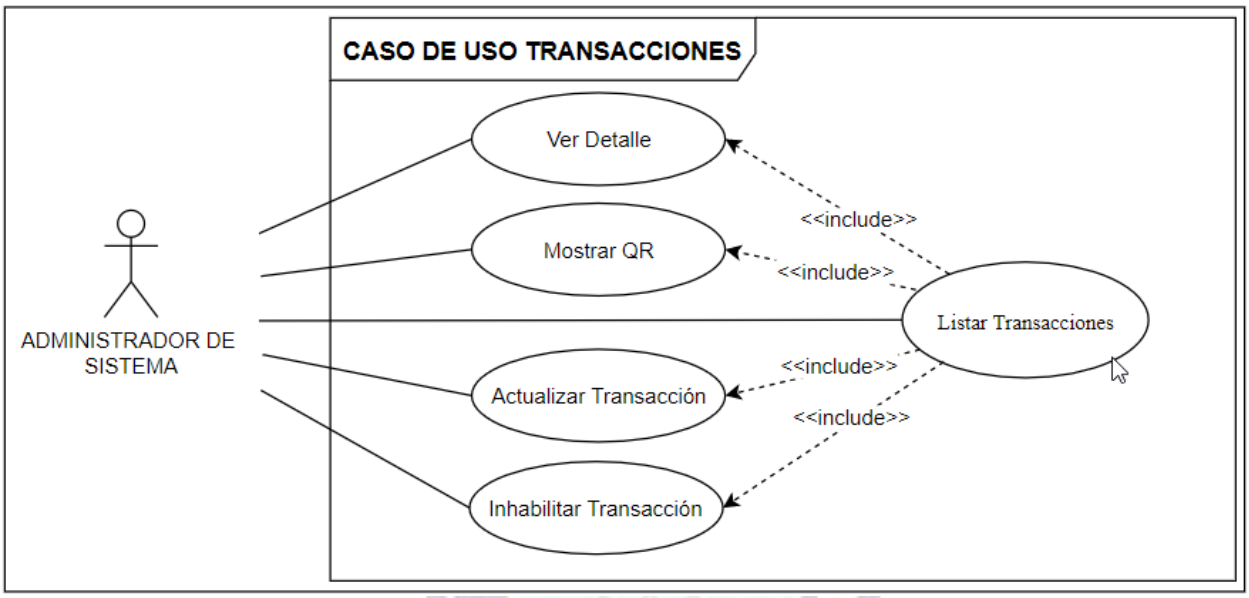


Figura 3.7: Caso de uso administración transacciones.
Fuente: Elaboración propia

La tabla 3.5 detalla las especificaciones del caso de uso de transacciones:

Caso de uso	Transacciones
Actores	Administrador de sistema
Descripción	Permite al administrador de sistema visualizar las transacciones y ver el detalle, mostrar el QR, actualizara e inhabilitar cada una de las transacciones.
Precondiciones	El actor debe ser un usuario registrado y con el rol asignado de administrador del sistema.
Flujo Normal	1. El actor ingresa a la opción de Transacciones. 2. El sistema despliega el listado de las transacciones registradas, cada uno con las opciones de ver detalle, mostrar QR, actualizar transacción e inhabilitar transacción. 3. El actor escoge una de las acciones.
Postcondiciones	El sistema ejecuta la opción elegida por el actor y muestra los cambios realizados.

Tabla 3.5 Especificación del caso de uso de transacciones
Fuente Elaboración propia.

3.3.1.3 MODELOS DE CONTENIDO

El modelo de contenidos específica las clases que se ha realizado en la etapa de desarrollo del módulo de administración, teniendo en cuenta clase Usuario que contiene datos necesarios, la clase Roles que se asignan a los usuarios registrados y la clase Permiso que controla a los roles de cada usuario, la figura 3.8 muestra el diagrama de contenidos de la administración del sistema.

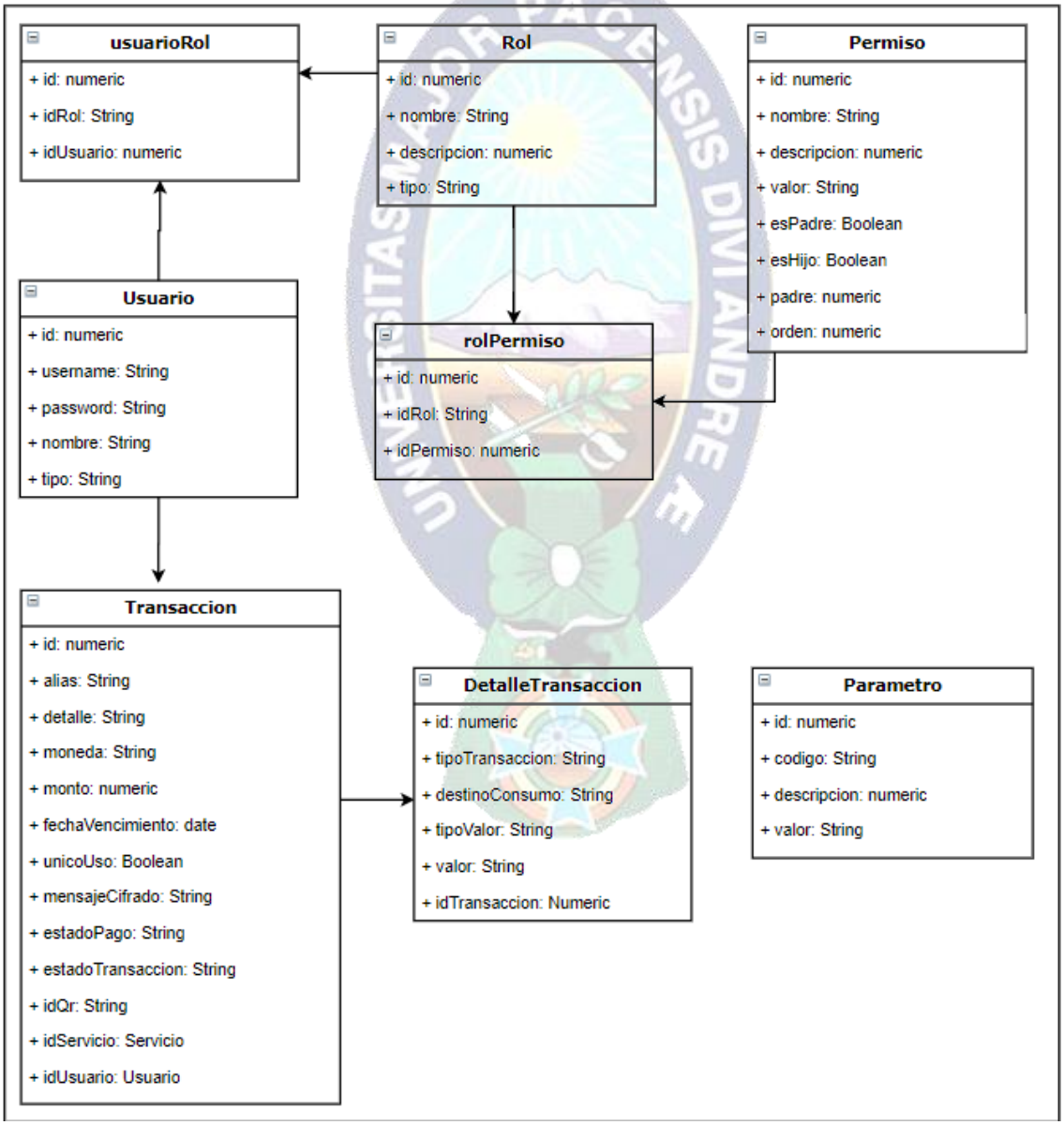


Figura 3.8: Diagrama de contenido módulo administración
Fuente: Elaboración propia

3.3.2 FASE DE DISEÑO

3.3.2.1 MODELO NAVEGACIONAL

El modelo navegacional determina las páginas del Módulo Administración, utilizando los nodos (nodes) y enlaces (links), en la Figura 3.8 muestra el comportamiento del usuario en el momento de ingresar a la página Administración, donde puede elegir entre ver las opciones de transacciones, reportes, usuarios, roles, parámetros, categorías, bancos y empresas.

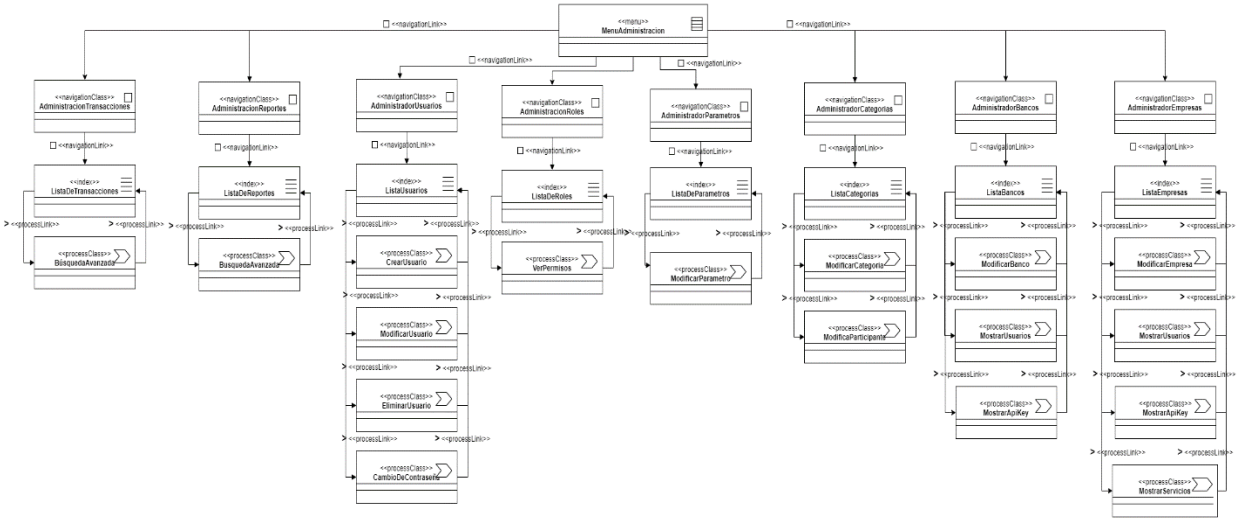


Figura 3.9: Diagrama de Navegación Módulo Administración
Fuente: Elaboración Propia

3.3.2.2 MODELO DE PRESENTACIÓN

Para el modelo de presentación se realizó el diseño de interfaz de las vistas del módulo de administración, consta del diseño de las ventanas de transacciones, reportes, usuarios, roles, parámetros, categorías, bancos y empresas.

En la Figura 3.10 se muestra el diseño de la vista de administración de transacciones

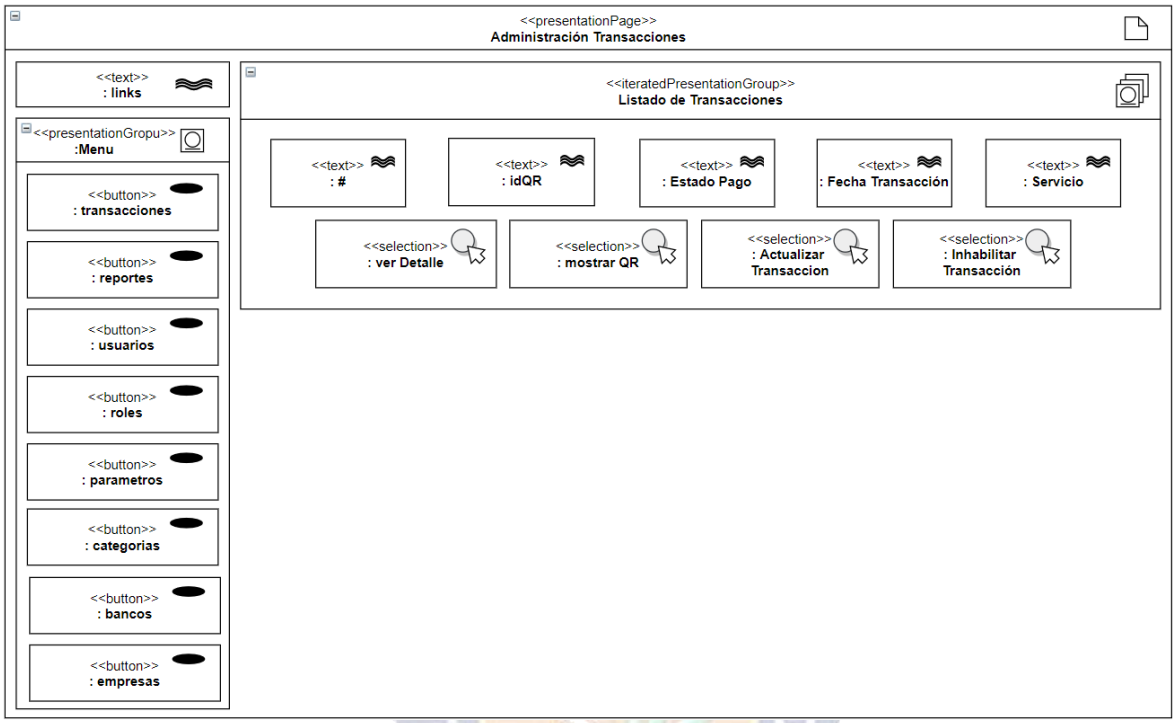


Figura 3.10 Diseño de la vista administración Transacciones
Fuente: Elaboración propia

En la Figura 3.11 se muestra el diseño de la vista de administración de Reportes

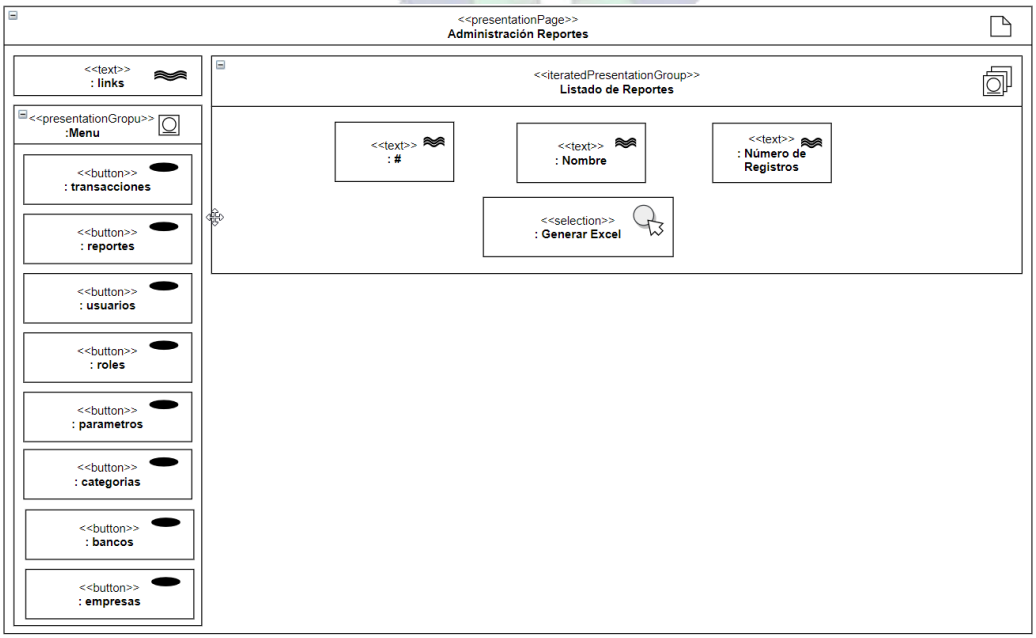


Figura 3.11 Diseño de la vista administración Reportes
Fuente: Elaboración propia

En la Figura 3.12 se muestra el diseño de la vista de administración de Usuarios

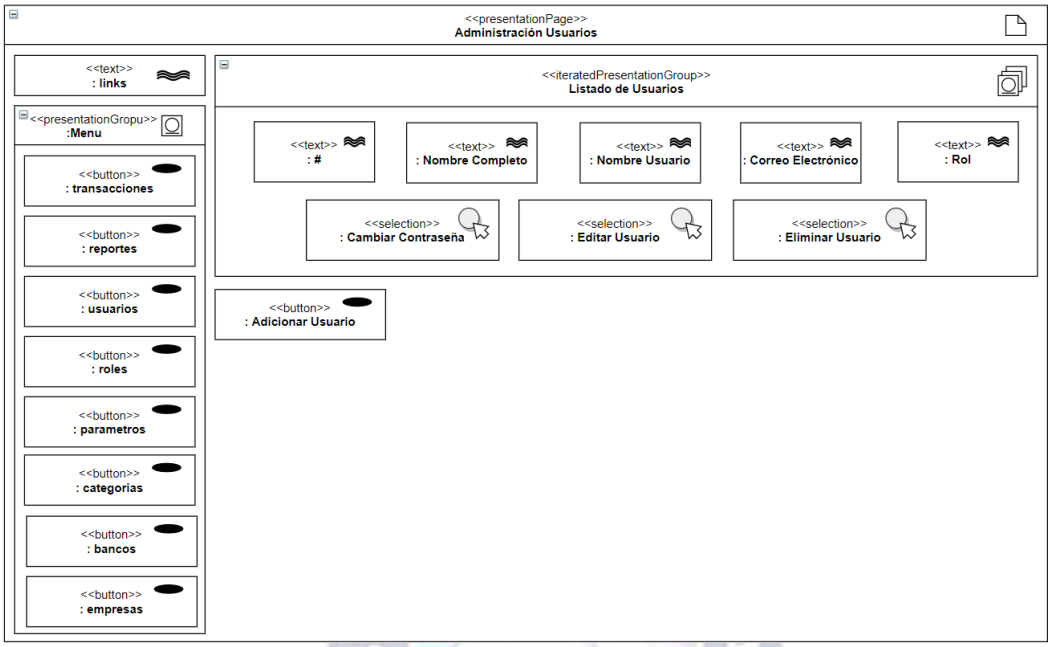


Figura 3.12 Diseño de la vista administración Usuarios
Fuente: Elaboración propia

En la Figura 3.13 se muestra el diseño de la vista de administración de Roles

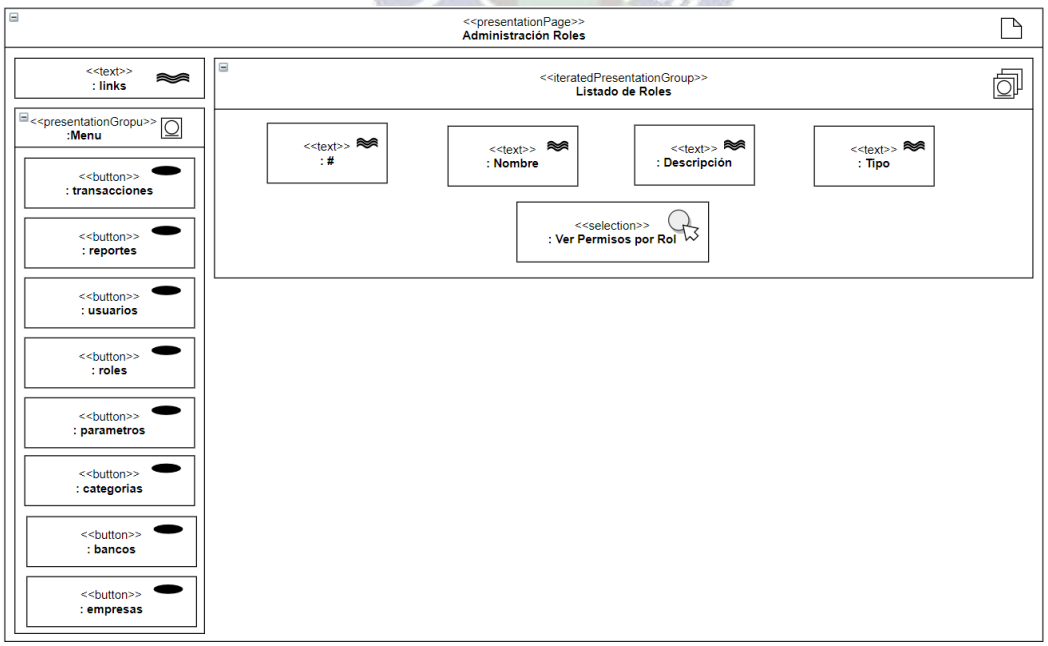


Figura 3.13 Diseño de la vista administración Roles
Fuente: Elaboración propia

En la Figura 3.14 se muestra el diseño de la vista de administración de Parámetros

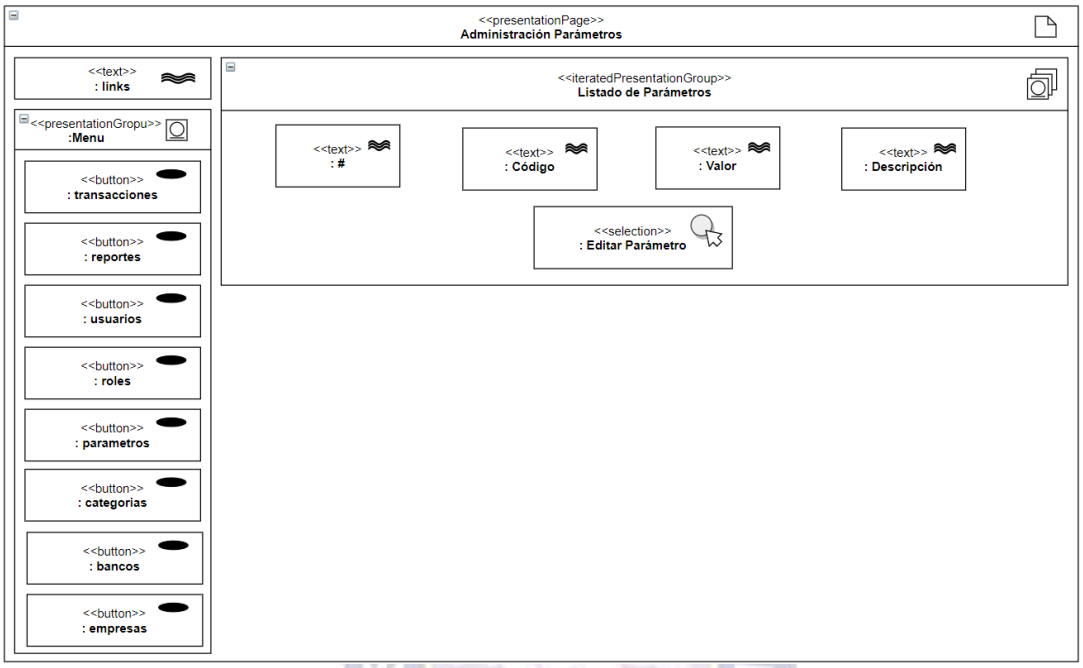


Figura 3.14 Diseño de la vista administración Parámetros
Fuente: Elaboración propia

En la Figura 3.15 se muestra el diseño de la vista de administración de Categorías

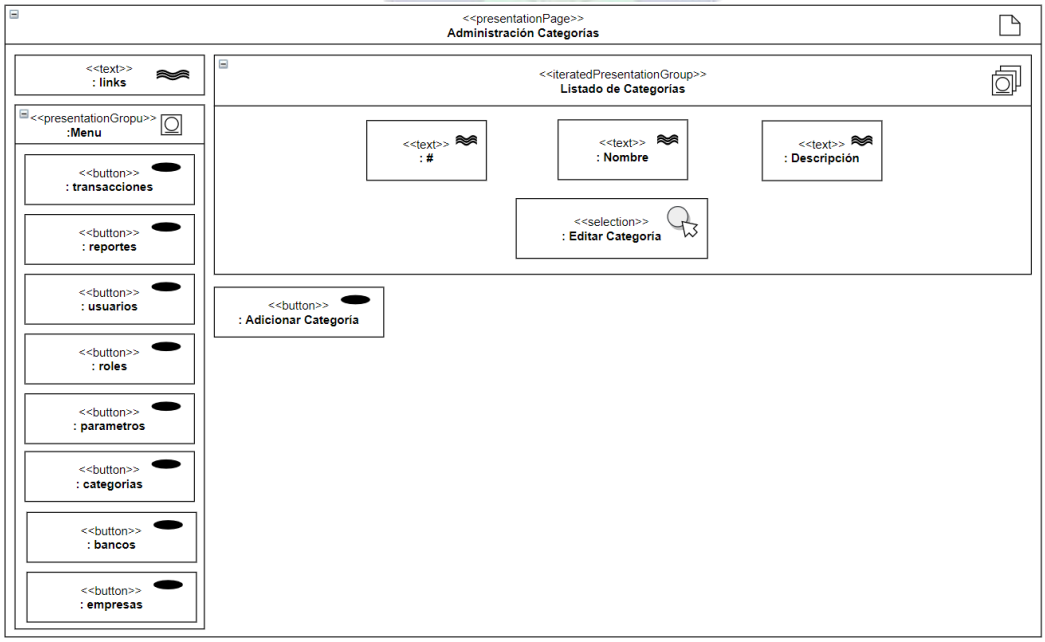


Figura 3.15 Diseño de la vista administración Categorías
Fuente: Elaboración propia

En la Figura 3.16 se muestra el diseño de la vista de administración de Bancos

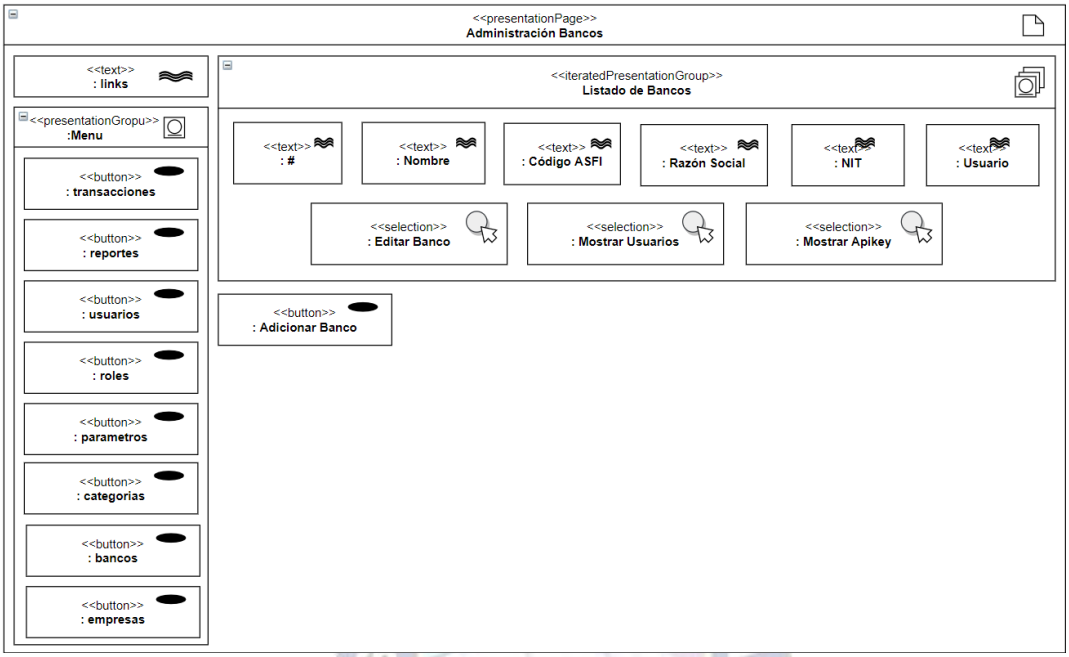


Figura 3.16 Diseño de la vista administración Bancos
Fuente: Elaboración propia

En la Figura 3.17 se muestra el diseño de la vista de administración de Empresas

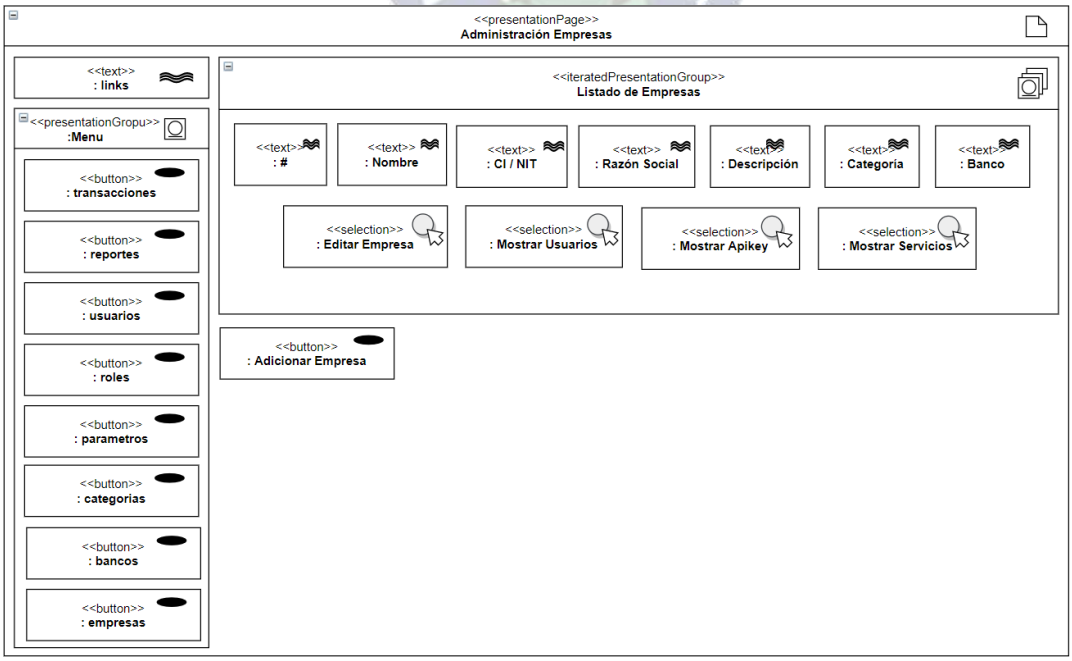


Figura 3.17 Diseño de la vista administración Empresas
Fuente: Elaboración propia

3.3.3 FASE DE DESARROLLO

3.3.3.1 MODELO DE FLUJO DE PROCESO

En este modelo se encuentra el flujo de proceso del comportamiento de administración de usuarios como muestra la figura 3.18.

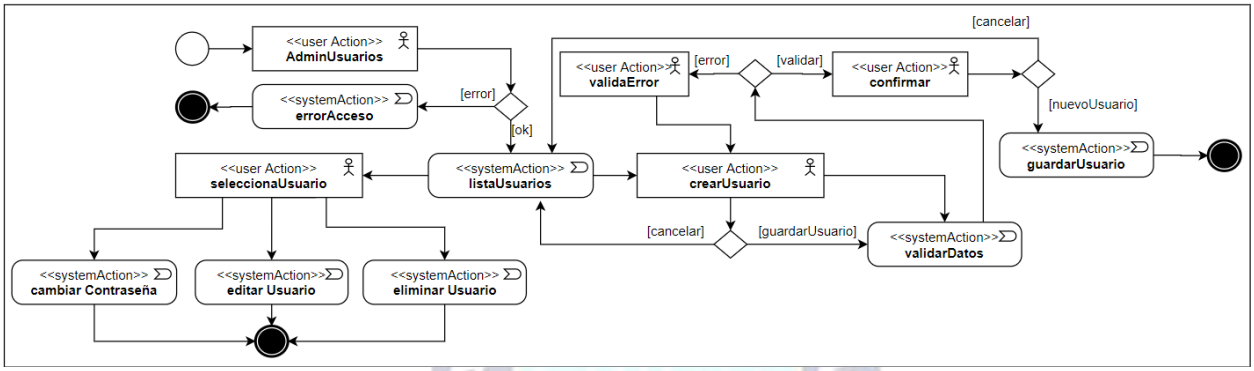


Figura 3.18 Diseño del modelo de flujo de proceso para administración de usuarios.
Fuente: Elaboración propia

En este modelo se encuentra el flujo de proceso del comportamiento de administración de roles como muestra la figura 3.19.

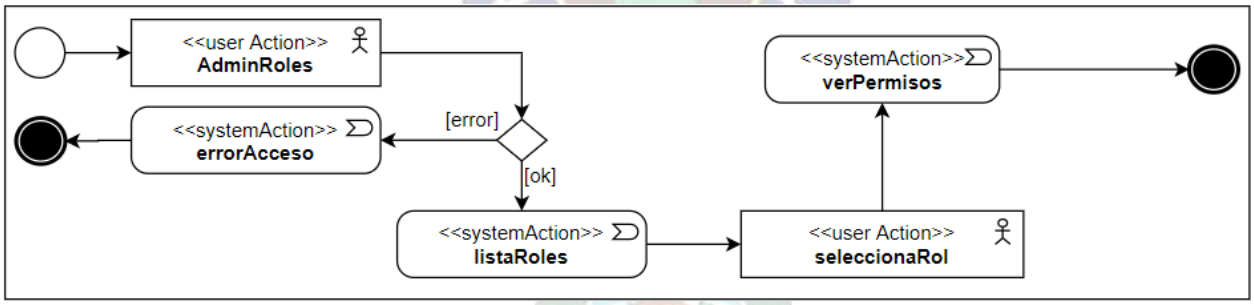


Figura 3.19 Diseño del modelo de flujo de proceso para administración de roles.
Fuente: Elaboración propia

En este modelo se encuentra el flujo de proceso del comportamiento de administración de parámetros como muestra la figura 3.20

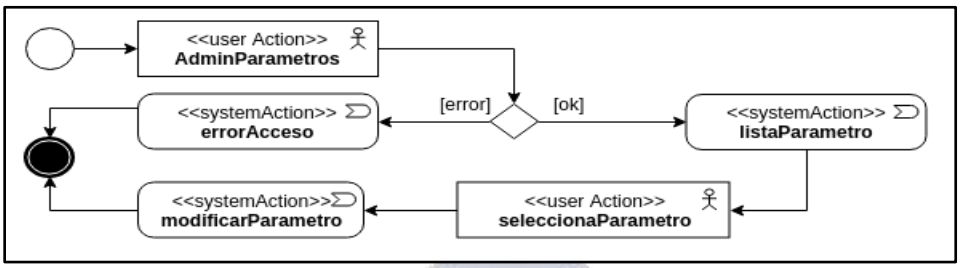


Figura 3.20 Diseño del modelo de flujo de proceso para administración de parámetros.
Fuente: Elaboración propia

En este modelo se encuentra el flujo de proceso del comportamiento de administración de categorías como muestra la figura 3.21

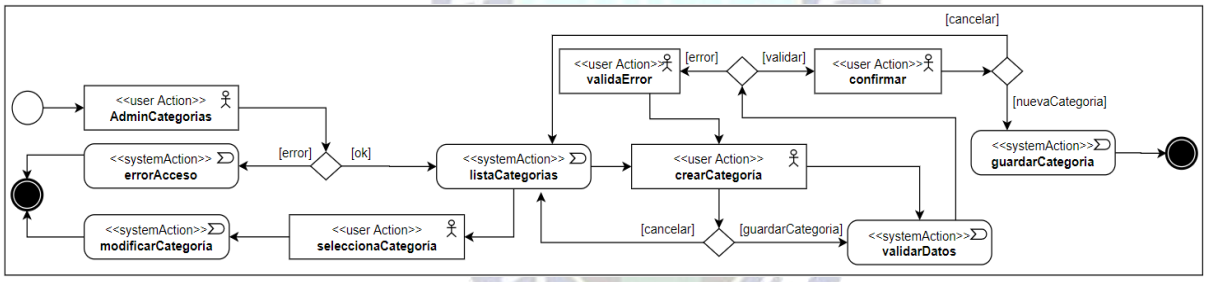


Figura 3.21 Diseño del modelo de flujo de proceso para administración de categorías.
Fuente: Elaboración propia

En este modelo se encuentra el flujo de proceso del comportamiento de administración de empresas como muestra la figura 3.22

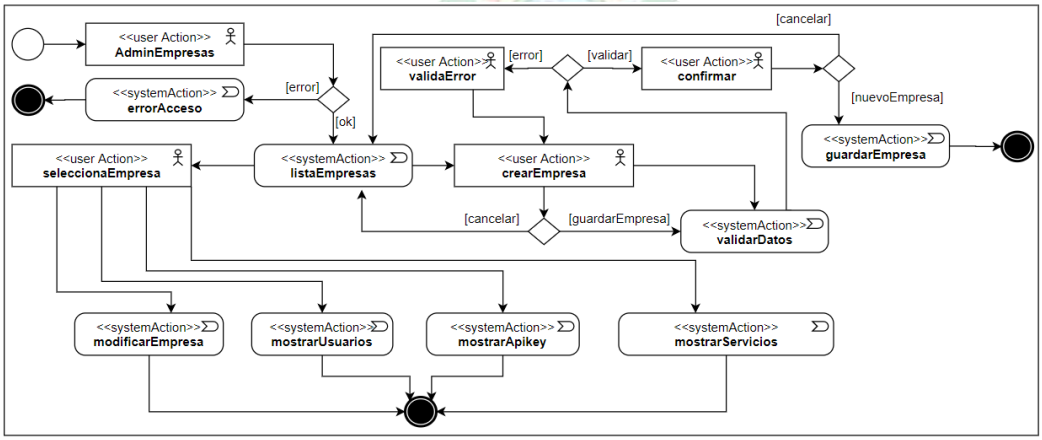


Figura 3.22 Diseño del modelo de flujo de proceso para administración de empresas.
Fuente: Elaboración propia

3.3.4 FASE DE IMPLEMENTACIÓN

Para esta parte se desarrolló el prototipo de la página de Administración de transacciones como se puede observar la figura 3.23 que muestra una captura de pantalla

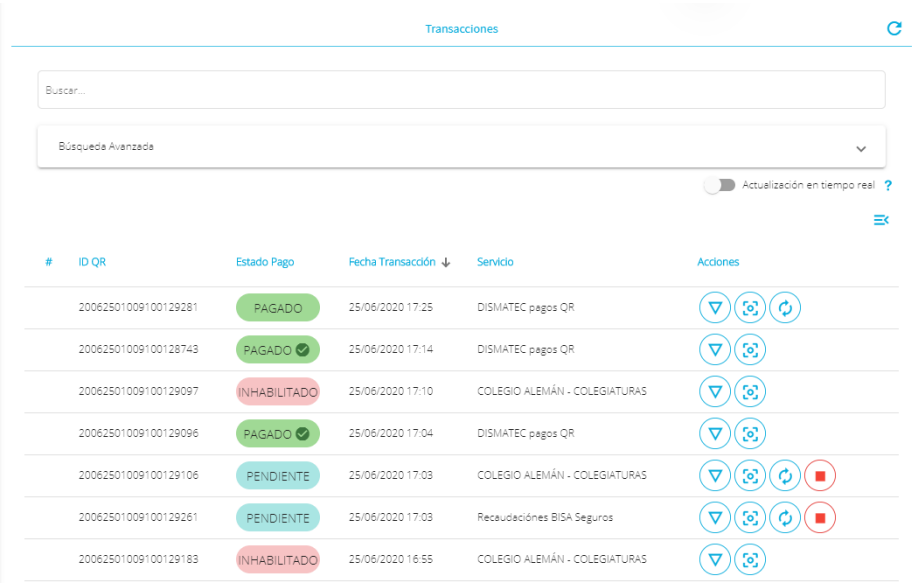


Figura 3.23 Captura de pantalla del prototipo de Administración de transacciones.
Fuente: Elaboración propia

Para esta parte se desarrolló el prototipo de la página de Detalle de transacciones como se puede observar la figura 3.24 que muestra una captura de pantalla

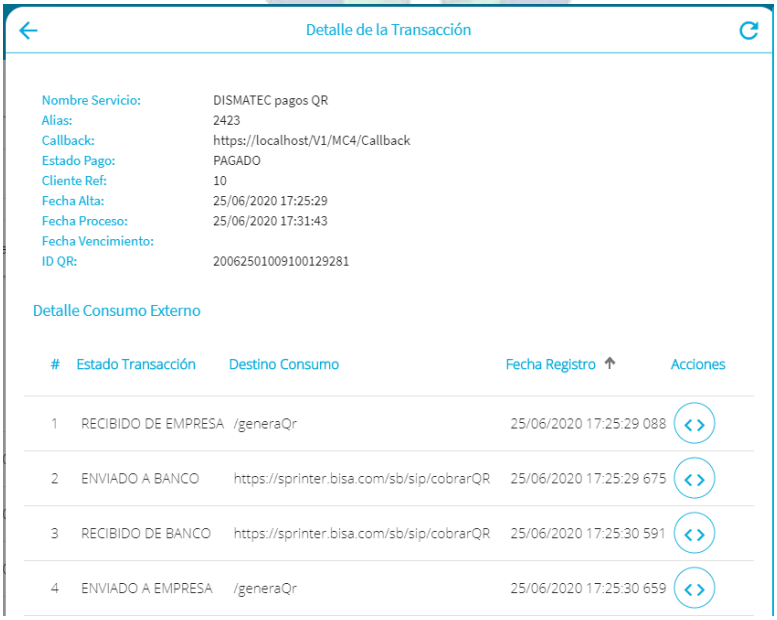


Figura 3.24 Captura de pantalla del prototipo de Detalle de transacciones
Fuente: Elaboración propia

Para esta parte se desarrolló el prototipo de la página de Mostrar Código QR de la transacción como se puede observar la figura 3.25 que muestra una captura de pantalla

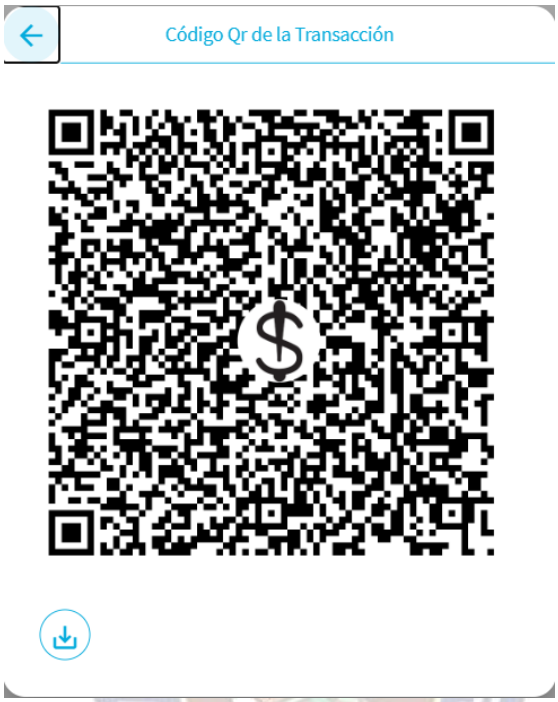


Figura 3.25 Captura de pantalla del prototipo de mostrar código QR de la transacción.
Fuente: Elaboración propia

Para esta parte se desarrolló el prototipo del diálogo de Inhabilitación de código QR como se puede observar la figura 2.26 que muestra una captura de pantalla

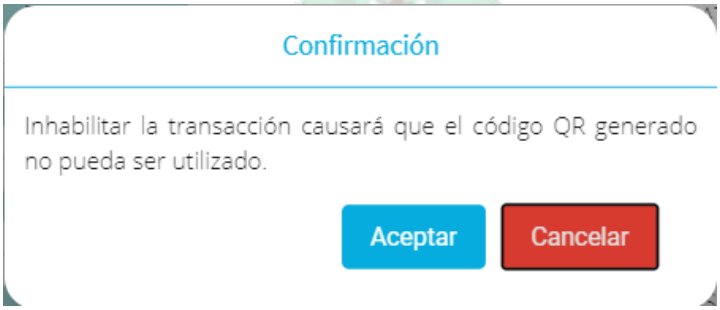


Figura 3.26 Captura de pantalla del prototipo de Inhabilitación de código QR.
Fuente: Elaboración propia

Para esta parte se desarrolló el prototipo de la página de Administración de usuarios como se puede observar la figura 3.27 que muestra una captura de pantalla

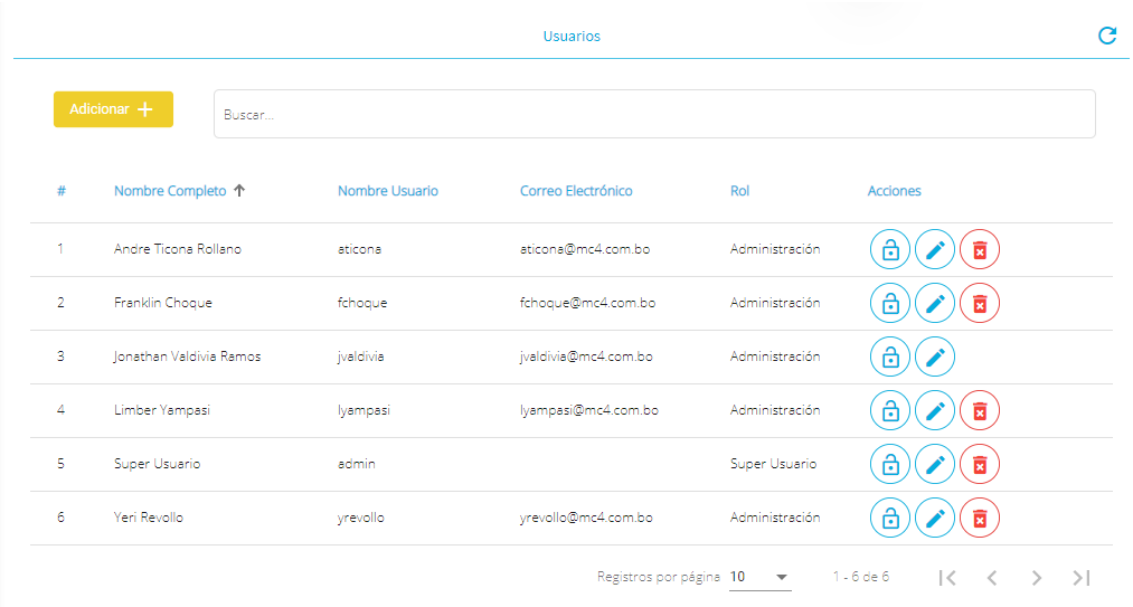


Figura 3.27 Captura de pantalla del prototipo de Administración de usuarios.
Fuente: Elaboración propia

Para esta parte se desarrolló el prototipo de la página de creación de usuarios como se puede observar la figura 3.28 que muestra una captura de pantalla.

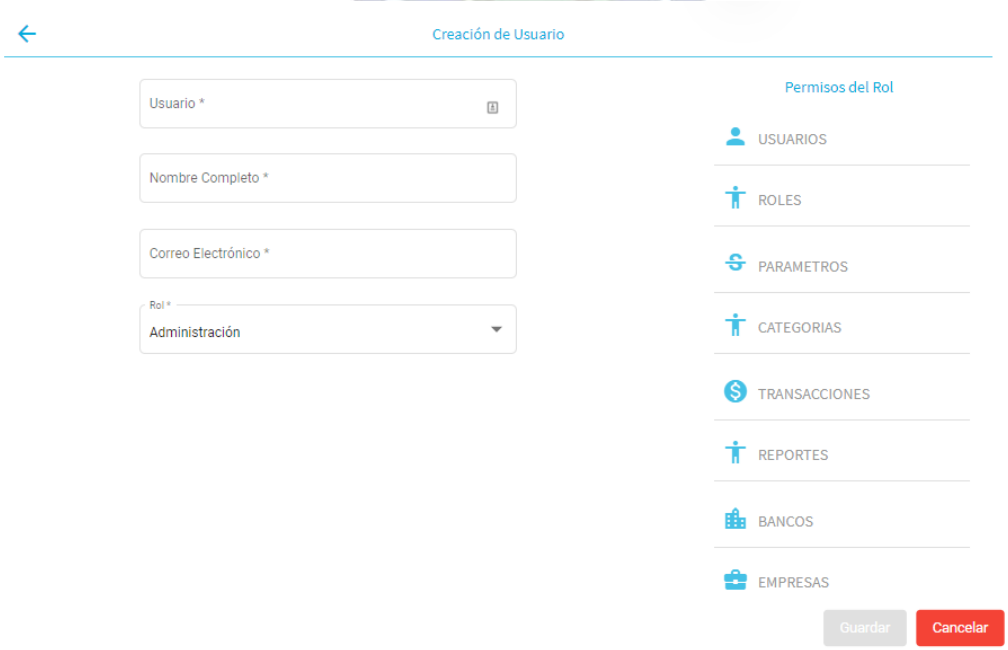


Figura 3.28 Captura de pantalla del prototipo de creación de usuarios.
Fuente: Elaboración propia

Para esta parte se desarrolló el prototipo de la página de modificación de usuarios como se puede observar la figura 3.29 que muestra una captura de pantalla.

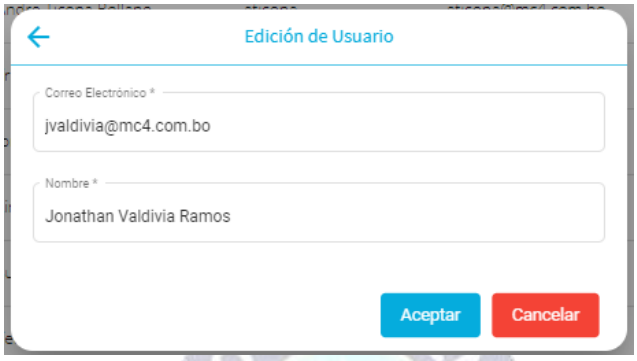


Figura 3.29 Captura de pantalla del prototipo de modificación de usuarios
Fuente: Elaboración propia

Para esta parte se desarrolló el prototipo de la página de cambio de contraseña como se puede observar la figura 3.30 que muestra una captura de pantalla.

Figura 3.30 Captura de pantalla del prototipo de cambio de contraseña
Fuente: Elaboración propia

Para esta parte se desarrolló el prototipo de la página de eliminación de usuario cómo se puede observar la figura 3.31 que muestra una captura de pantalla.

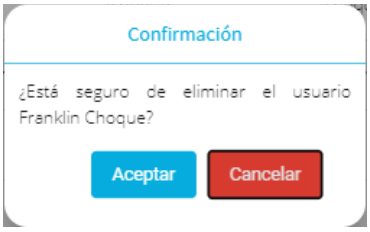


Figura 3.31 Captura de pantalla del prototipo de eliminación de usuarios.
Fuente: Elaboración propia

Para esta parte se desarrolló el prototipo de la pantalla de Administración de roles como se puede observar la figura 3.32 que muestra una captura de pantalla.

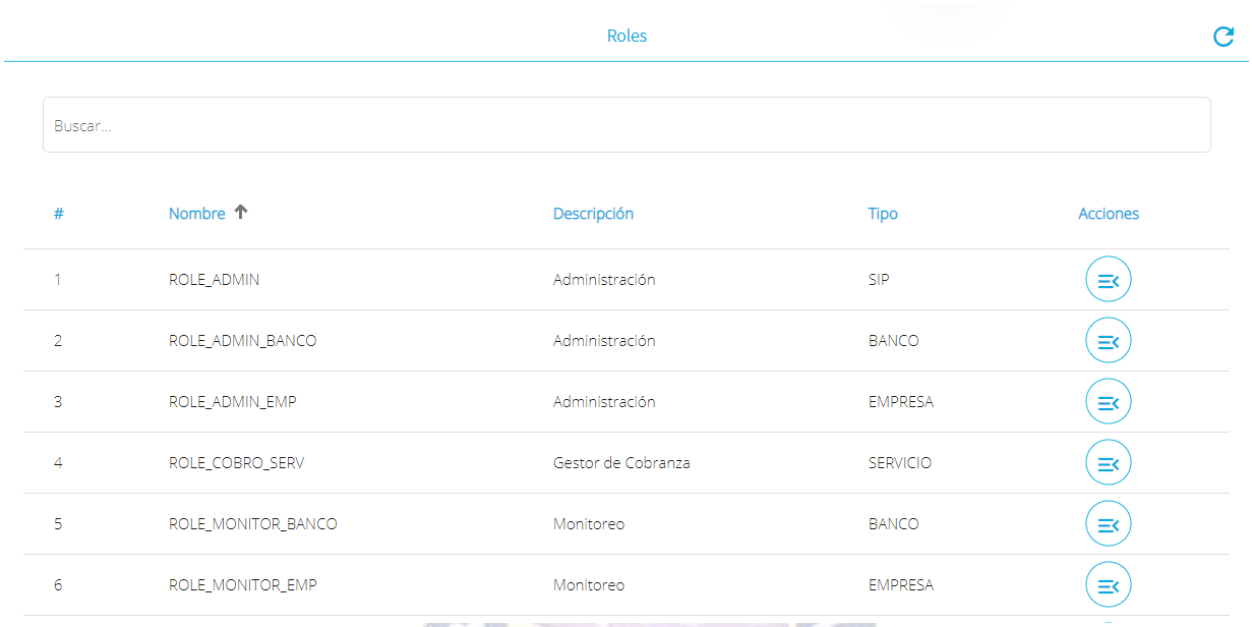


Figura 3.32 Captura de pantalla del prototipo de Administración de roles.
Fuente: Elaboración propia

Para esta parte se desarrolló el prototipo de la opción de Permisos por Rol como se puede observar la figura 3.33 que muestra una captura de pantalla

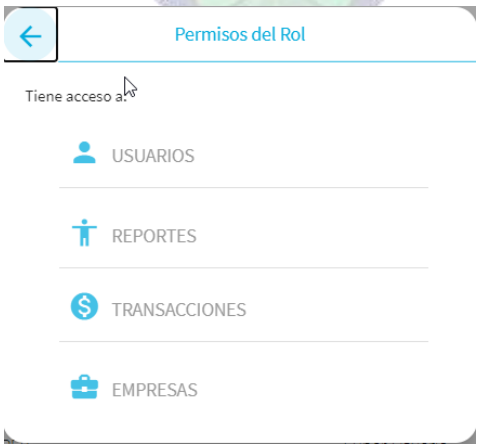


Figura 3.33 Captura de pantalla del prototipo de Permisos por Rol.
Fuente: Elaboración propia

Para esta parte se desarrolló el prototipo de la página de Administración de parámetros como se puede observar la figura 3.34 que muestra una captura de pantalla

Figura 3.34 Captura de pantalla del prototipo de Administración de Parámetros.
Fuente: Elaboración propia

Para esta parte se desarrolló el prototipo de la página de edición de parámetros como se puede observar la figura 3.35 que muestra una captura de pantalla

Figura 3.35 Captura de pantalla del prototipo de Edición de parámetros.
Fuente: Elaboración propia

Para esta parte se desarrolló el prototipo de la página de Administración de categorías como se puede observar la figura 3.36 que muestra una captura de pantalla

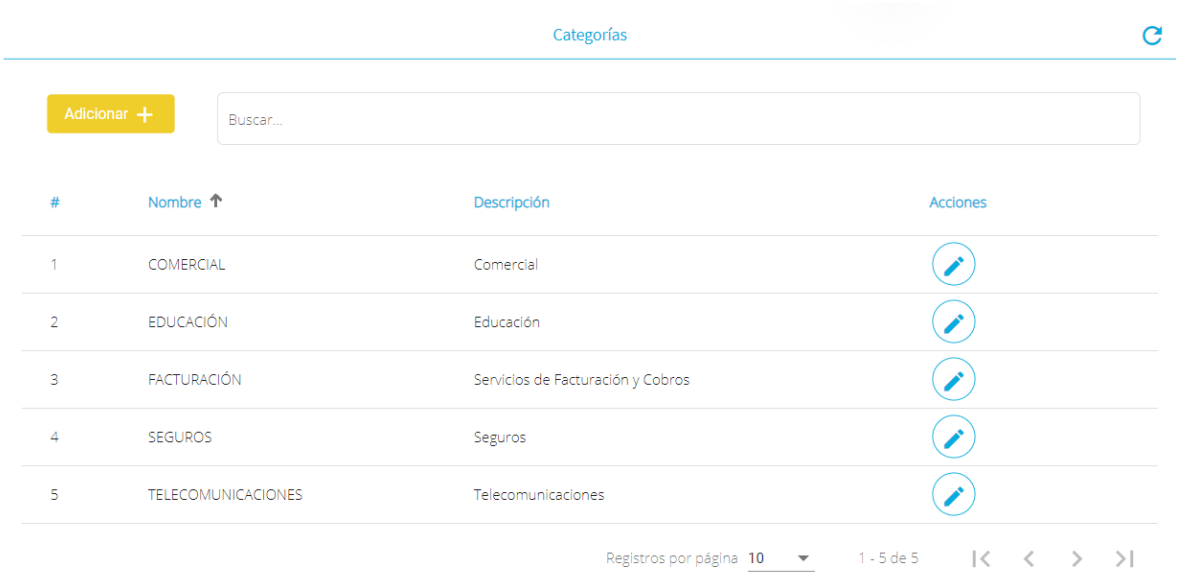


Figura 3.36 Captura de pantalla del prototipo de Administración de categorías.
Fuente: Elaboración propia

Para esta parte se desarrolló el prototipo de la página de creación de categorías como se puede observar la figura 3.37 que muestra una captura de pantalla

Figura 3.37 Captura de pantalla del prototipo de creación de categoría.
Fuente: Elaboración propia

Para esta parte se desarrolló el prototipo de la página de edición de categorías como se puede observar la figura 3.38 que muestra una captura de pantalla

Figura 3.38 Captura de pantalla del prototipo de edición de categorías.
Fuente: Elaboración propia

Para esta parte se desarrolló el prototipo de la página de Administración de empresas como se puede observar la figura 3.39 que muestra una captura de pantalla

Figura 3.39 Captura de pantalla del prototipo de Administración de empresas.
Fuente: Elaboración propia

Para esta parte se desarrolló el prototipo de la página de creación de empresas como se puede observar la figura 3.40 que muestra una captura de pantalla

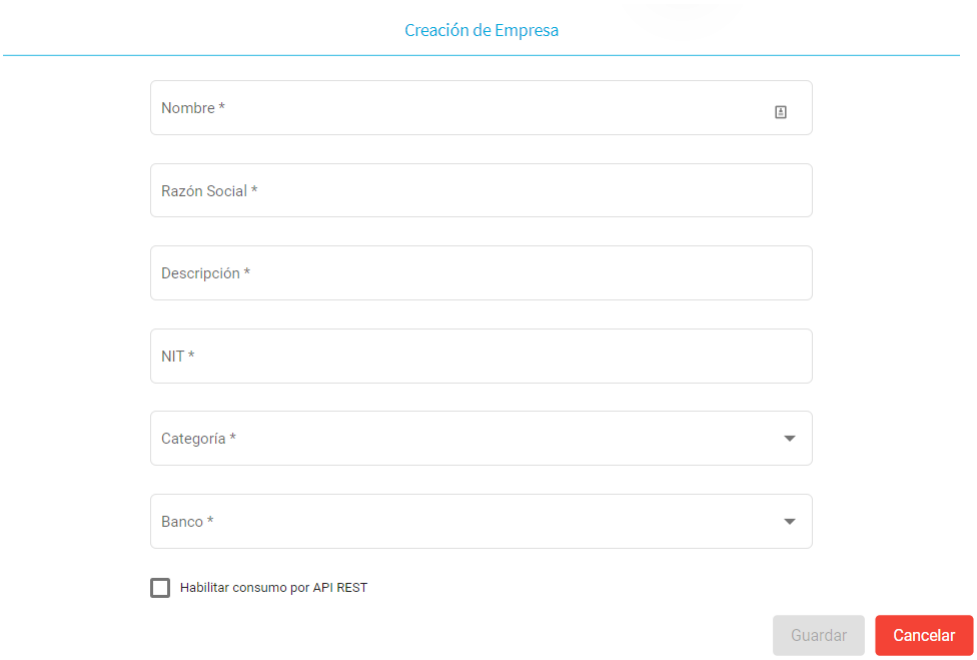


Figura 3.40 Captura de pantalla del prototipo de Creación de Empresa.
Fuente: Elaboración propia

Para esta parte se desarrolló el prototipo de la página de Edición de empresa como se puede observar la figura 3.41 que muestra una captura de pantalla

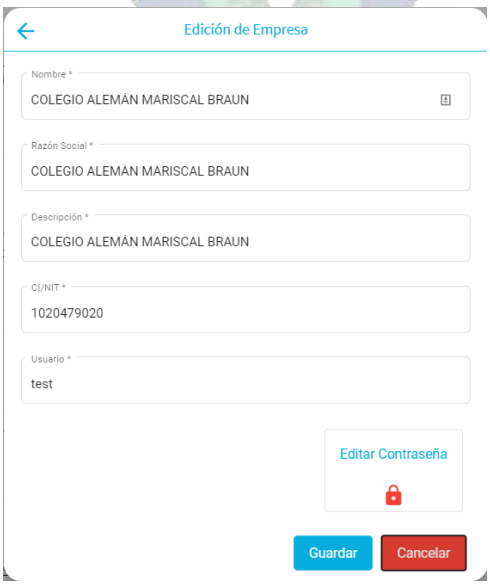


Figura 3.41 Captura de pantalla del prototipo de Edición de Empresa.
Fuente: Elaboración propia

Para esta parte se desarrolló el prototipo de la página de Usuarios de Empresa como se puede observar la figura 3.42 que muestra una captura de pantalla

Figura 3.42 Captura de pantalla del prototipo de Usuario de Empresa.
Fuente: Elaboración propia

Para esta parte se desarrolló el prototipo de la página de la opción mostrar Apikey como se puede observar la figura 3.43 que muestra una captura de pantalla

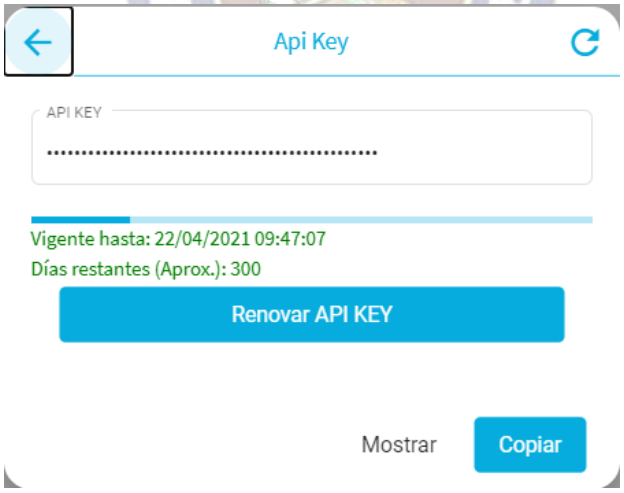


Figura 3.43 Captura de pantalla del prototipo de mostrar Apikey.
Fuente: Elaboración propia

Para esta parte se desarrolló el prototipo de la página de la opción mostrar Servicios por Empresa como se puede observar la figura 3.44 que muestra una captura de pantalla

Figura 3.44 Captura de pantalla del prototipo de mostrar Servicios por Empresa.
Fuente: Elaboración propia

3.4 MÓDULO DE INTEGRACIÓN UNICO Y ACH EXPRESS (Sprint 2)

En el Sprint 2 se definieron las siguientes tareas:

1. Módulo de consumo de servicios Web UNICO.
2. Módulo de consumo de servicios Web ACH EXPRESS.
3. Pantallas Módulo de Transferencias
4. Módulo plataforma pública
5. Módulo Orquestador de Transferencias
6. Pantallas Plataforma Pública

En la figura 3.45 se visualiza las tareas definidas para el 2do sprint

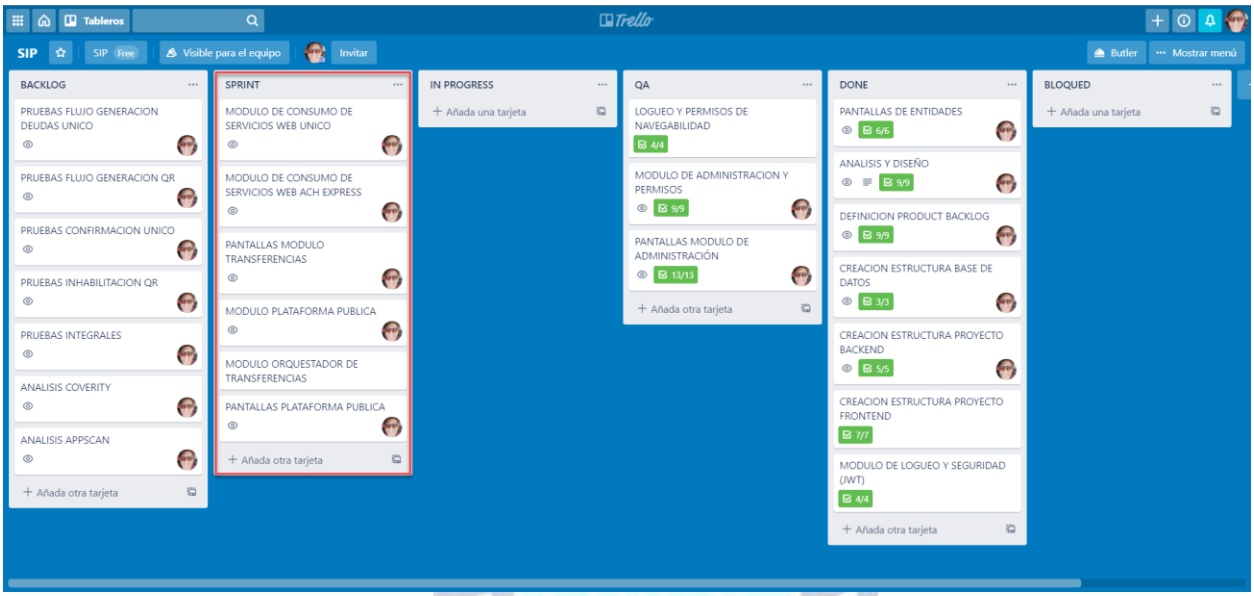


Figura 3.45 Herramienta Trello Sprint 2
Fuente: Elaboración Propia

3.4.1 FASE DE ANÁLISIS

3.4.1.1 ANÁLISIS DE REQUERIMIENTOS

La especificación de requerimientos constituye una base sólida para el diseño y la construcción de software (Pressman, 2010), sin ésta el software resultante tiene alta probabilidad de no satisfacer las necesidades de los clientes.

En este apartado se detalla la especificación de todos los requerimientos funcionales relevados para desarrollar la Integración con la plataforma de pagos UNICO y el sistema ACH EXPRESS.

a) Requerimientos Funcionales

Especifica los servicios que proveerá el sistema (funciones específicas del sistema). En algunos casos, también declaran explícitamente lo que el sistema no debe hacer.

RF.01 La plataforma SIP debe poder consumir los servicios web expuestos por la plataforma de pagos UNICO.

RF.02 La plataforma SIP debe poder consumir los servicios web expuestos por el sistema ACH EXPRESS.

RF.03 Exigir autenticación mediante usuario y contraseña para cualquier consumo de los servicios de la plataforma UNICO y el sistema ACH EXPRESS.

RF.04 La plataforma SIP debe consumir el servicio web “Consulta de Cuentas” de la plataforma de pagos UNICO.

RF.05 La plataforma SIP debe consumir el servicio web “Consulta de Deudas” de la plataforma de pagos UNICO.

RF.06 La plataforma SIP debe consumir el servicio web “Pagos” de la plataforma de pagos UNICO.

RF.07 La plataforma SIP debe consumir el servicio web “Genera Token” del sistema ACH EXPRESS.

RF.08 La plataforma SIP debe consumir el servicio web “cobrarQR” del sistema ACH EXPRESS.

RF.09 La plataforma SIP debe consumir el servicio web “inhabilitarQR” del sistema ACH EXPRESS.

RF.10 La plataforma SIP debe validar toda la información requerida para el consumo de servicios web de la plataforma de pagos UNICO de acuerdo a la especificación de servicios de Cobranza provista por la empresa MC4.

RF.11 La plataforma SIP debe validar toda la información requerida para el consumo de servicios web del sistema ACH EXPRESS de acuerdo a la especificación de servicios Cliente ACH Express provista por la empresa MC4.

RF.12 La plataforma SIP debe exponer el servicio web “generaToken” de acuerdo a la especificación de servicios SIP provista por la empresa MC4.

RF.13 La plataforma SIP debe exponer el servicio web “confirmaPago” de acuerdo a la especificación de servicios SIP provista por la empresa MC4

b) Requerimientos No Funcionales

Son aquellos requerimientos que no se refieren directamente a las funciones específicas que entrega el sistema, sino a las propiedades emergentes de éste como la fiabilidad, la respuesta en el tiempo y la capacidad de almacenamiento.

RNF.01 La plataforma SIP debe proveer la información necesaria para la facturación mediante la plataforma de pagos UNICO.

RNF.02 La plataforma SIP debe tener un proceso de confirmación parametrizable y re reintentos para los pagos de UNICO.

3.4.1.2 MODELO DE FLUJO DE NEGOCIO

En este modelo se encuentra el flujo de negocio del comportamiento de integración a la plataforma de pagos UNICO y al sistema ACH EXPRESS mediante servicios web, como se ve en la figura 3.46

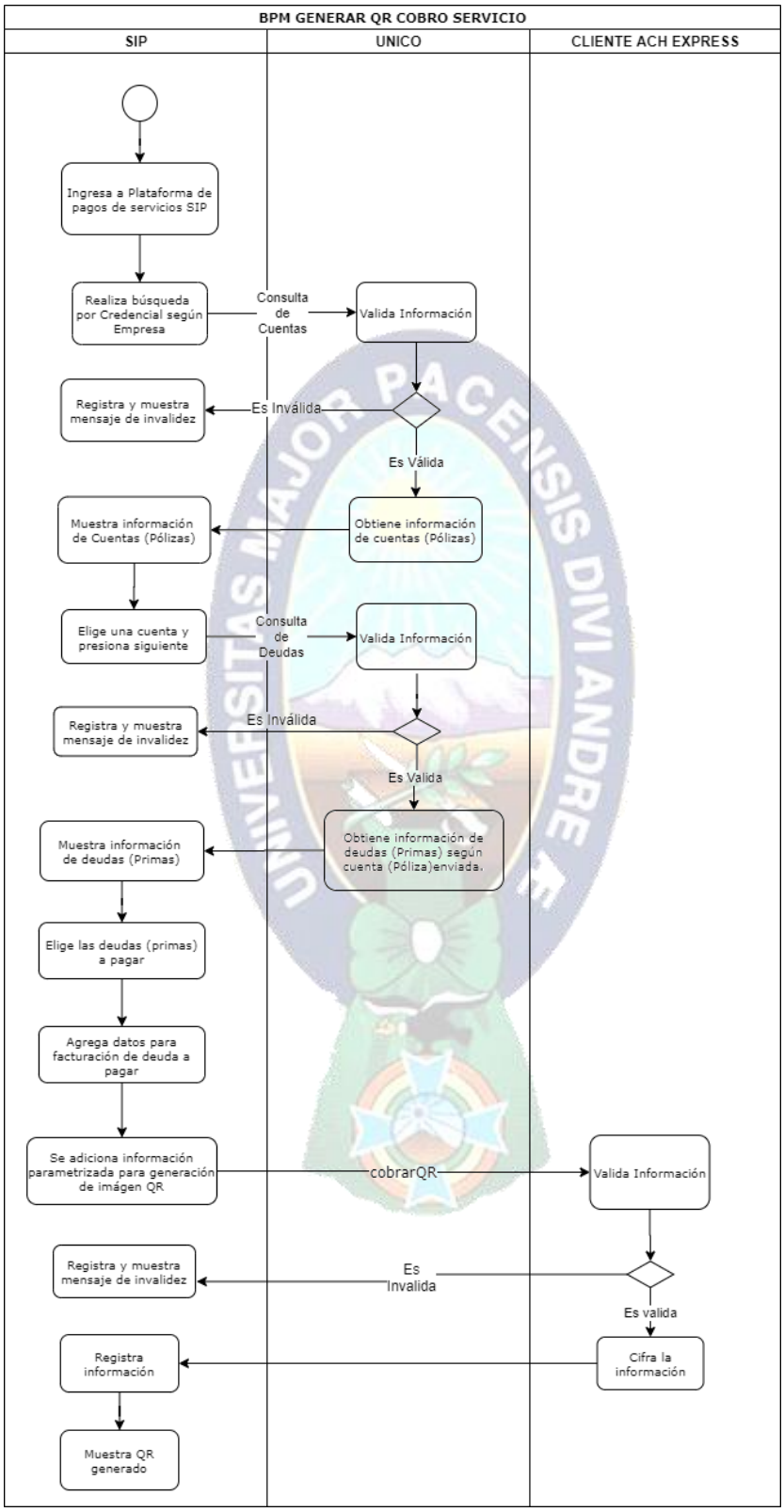


Figura 3.46: Diseño del modelo de flujo de negocio para Generar Imagen QR de Cobro
Fuente: Elaboración Propia

En este modelo se encuentra el flujo de negocio del comportamiento de Pago de imagen QR por el Sistema ACH EXPRESS y configuración a plataforma UNICO, como se muestra en la figura 3.47

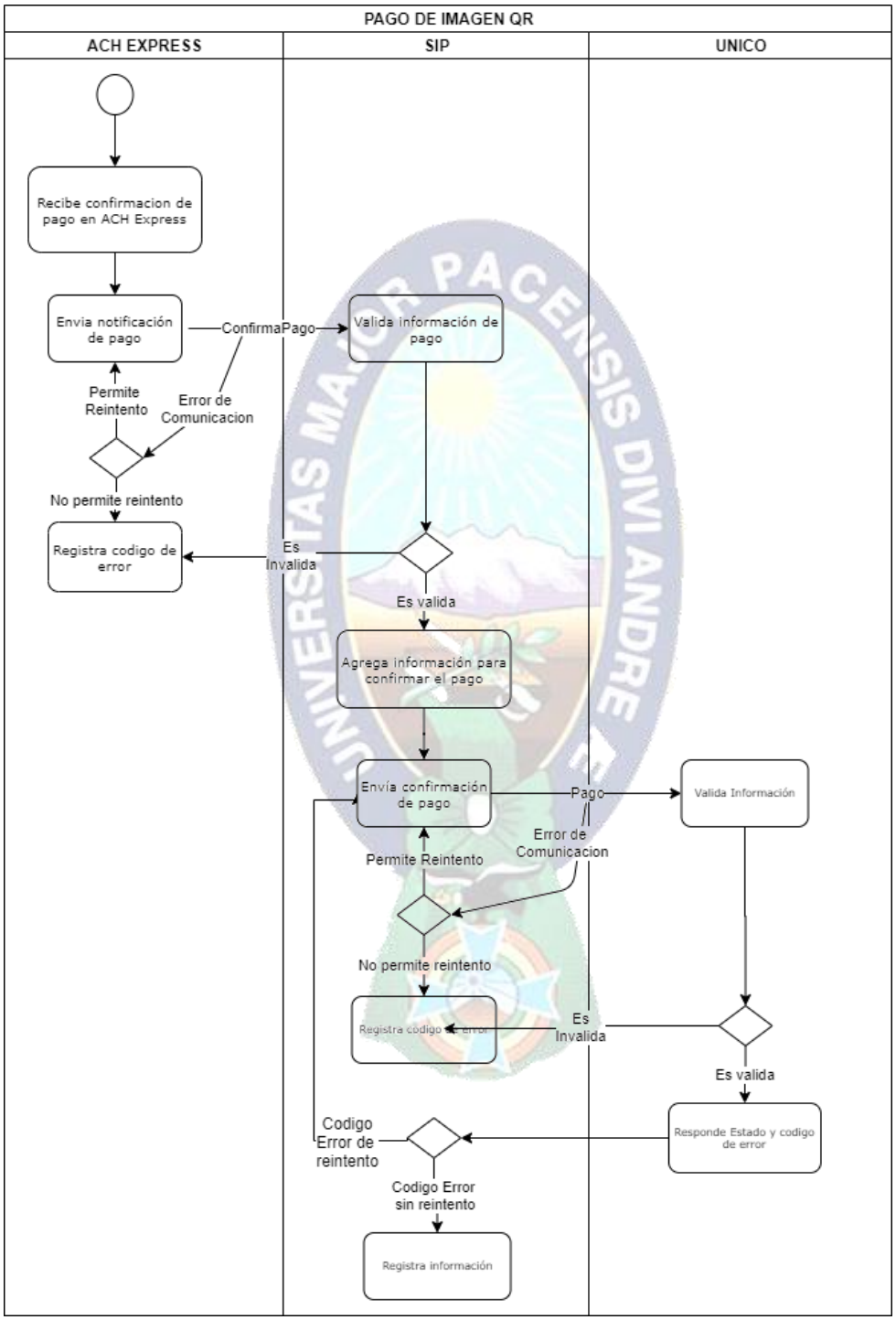


Figura 3.47 Diseño del modelo de flujo de negocio para el Pago de Imagen QR y Confirmación
Fuente: Elaboración Propia

3.4.2 FASE DE IMPLEMENTACIÓN

Para esta parte se desarrolló el prototipo del flujo de pago de servicios mediante la plataforma SIP como se puede ver en las siguientes figuras.

En la figura 3.48 se muestra el Ingreso a la plataforma SIP el cual está publicado por internet.

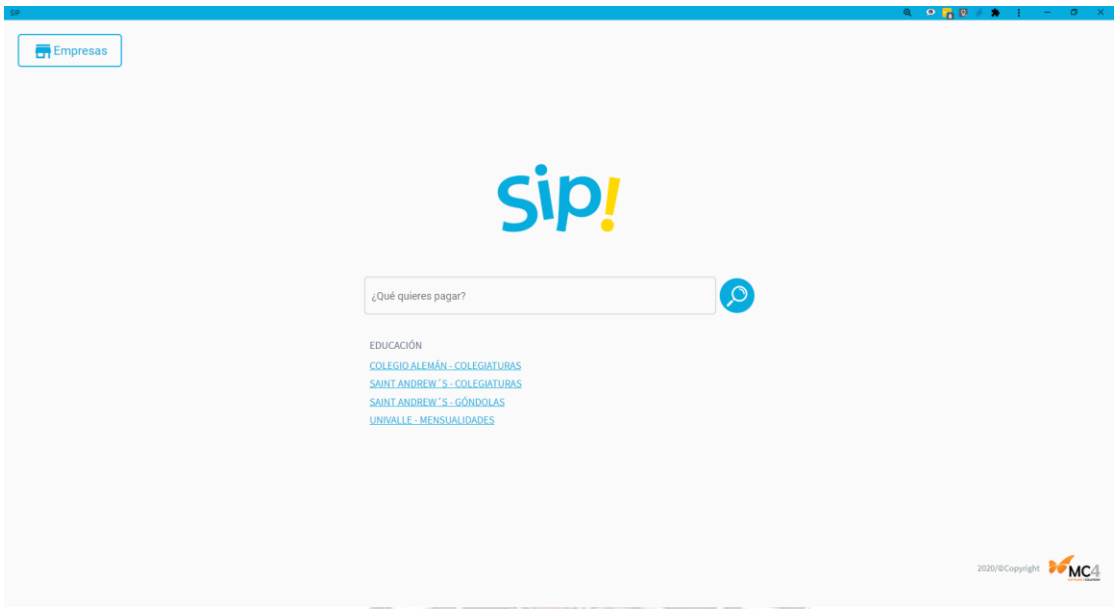


Figura 3.48 Captura de pantalla del prototipo de plataforma SIP
Fuente: Elaboración Propia

Para esta parte se desarrolló el prototipo de flujo de pago con el ejemplo de una institución educativa y el pago de sus pensiones, como se ve en la Figura 3.49

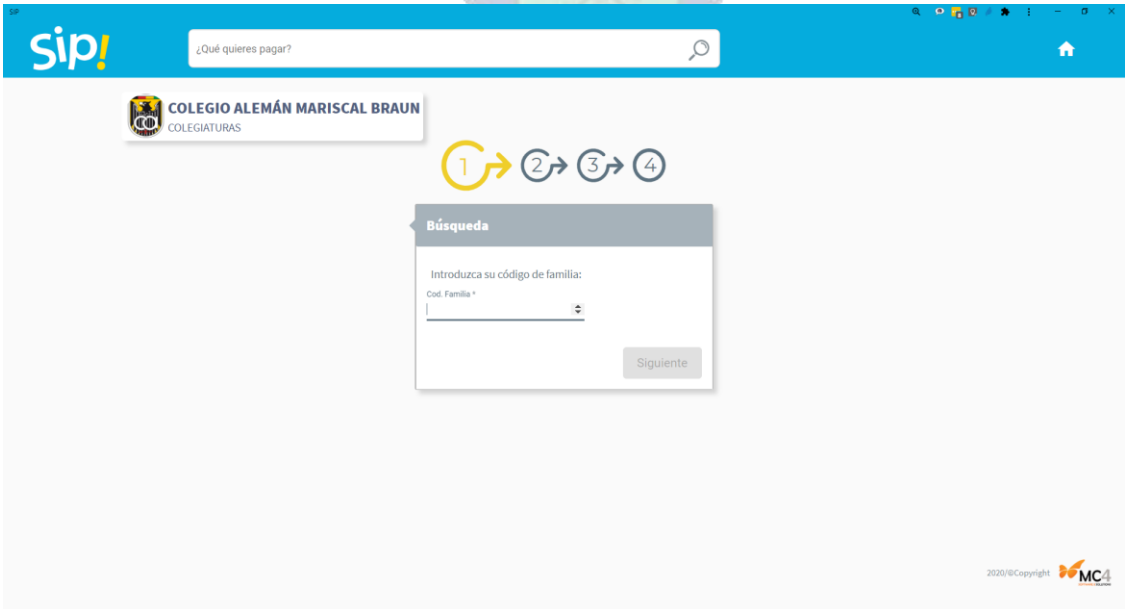


Figura 3.49 Captura de pantalla de prototipo de Ingreso de datos de Búsqueda
Fuente: Elaboración Propia

Para esta parte se desarrolló el prototipo de flujo de pago consumiendo el servicio de Cuentas de la plataforma UNICO para obtener los datos de las cuentas como se ve en la Figura 3.50

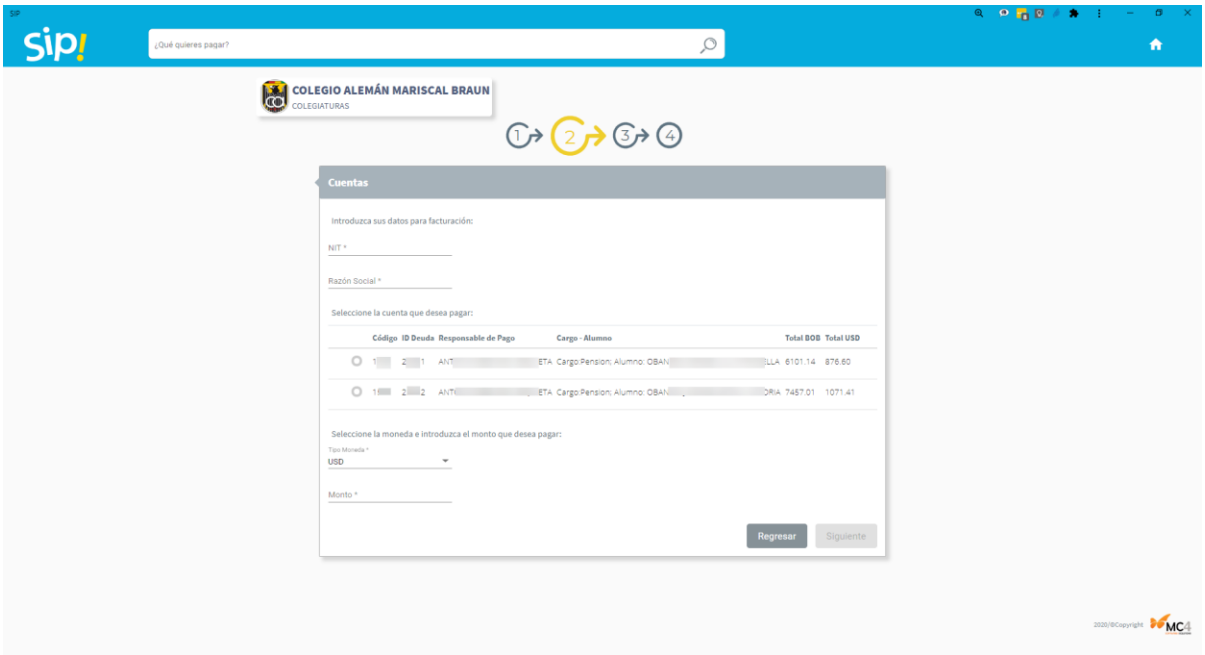


Figura 3.50 Captura de pantalla de prototipo de Consumo de Cuentas de UNICO
Fuente: Elaboración Propia

Para esta parte se desarrolló el prototipo de flujo de pago consumiendo el servicio de Deudas de la plataforma UNICO para obtener los datos de las deudas como se ve en la Figura 3.51

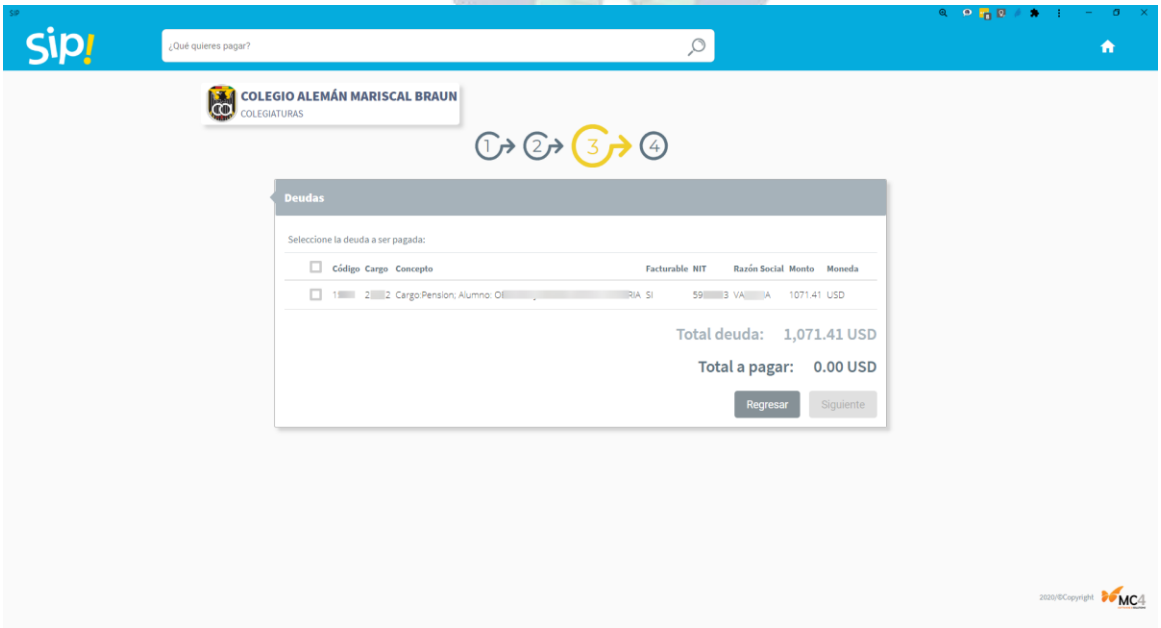


Figura 3.51 Captura de pantalla de prototipo de Consumo de Deudas de UNICO
Fuente: Elaboración Propia

Para esta parte se desarrolló el prototipo de flujo de pago consumiendo el servicio de cobrar QR del sistema ACH EXPRESS para generar la imagen QR la cual contiene toda la información de cobro del servicio elegido como se ve en la Figura 3.52

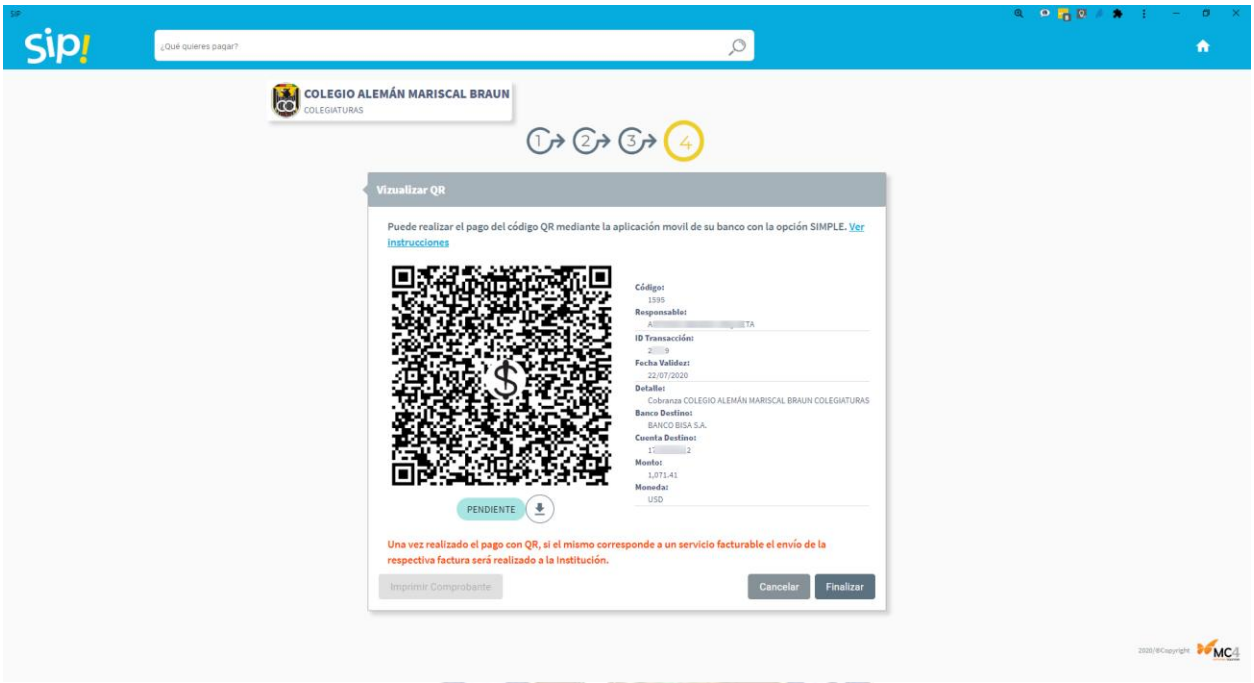


Figura 3.52 Captura de pantalla de prototipo de Generación de Imagen QR por ACH EXPRESS
Fuente: Elaboración Propia

3.5 FASE DE PRODUCCIÓN

Durante esta última etapa se realizó la prueba de software del producto cumpliendo las tareas trazadas de cada fase.

3.5.1 PRUEBA DE ESTRÉS

Teniendo el sistema en modo de producción y desplegado en un ambiente de producción se realizó la prueba de esfuerzo o prueba de estrés con la ayuda de la herramienta de software Apache JMETER v3.1.

Como se puede observar en la tabla 3.6 se muestra el resultado, la media total fue de 1147 ms. Esto quiere decir que el sistema en promedio tardó en responder 1,7 segundos, el cual es un tiempo bastante bueno, considerando que son 80 usuarios conectados al mismo tiempo.

URL	#Muestras	Media	Mediana	Min	Max	$Error	Rendimiento	Kb/Sec
Logeo	80	335	121,45	290	613	7.5%	7,7/sec	140.8
Registrar Usuarios	80	1372	200	676	2500	5%	7,3/sec	59.5
Modificación de Usuario	80	1266	371	727	1986	2%	7,3/sec	48.8
Buscar Usuario	80	1372	404	721	2000	0%	7,2/sec	50.8
Generar QR cifrado	80	1338	401	475	2043	1%	7,2/sec	60.1
Realizar Transferencia	80	1201	334	870	1993	0%	7,1/sec	31.0
Total	480	1147	305.2	626	1855	2.5 %	7.3/sec	65

Tabla 3.6 Datos agregados para la prueba de estrés, 80 usuarios
Fuente Elaboración propia.

Para poder concluir que el Sistema Web soporta la cantidad de usuarios requeridos, se hizo la prueba de estrés teniendo en cuenta que no todos los usuarios harán uso del servicio al mismo tiempo.

CAPÍTULO IV

CALIDAD Y SEGURIDAD

4.1 INTRODUCCIÓN

En el desarrollo de este capítulo se determina la calidad del sistema web en base a los parámetros de medición de la norma ISO 9126, donde hallaremos el punto función que nos servirá para el capítulo de costo beneficio, y las medidas de seguridad que se deben adoptar del lado del cliente y lado del servidor.

4.2 CALIDAD DE SOFTWARE

La ingeniería de software se diferencia de otras áreas, al no estar basada en leyes cuantitativas básicas, en su lugar se realiza un conjunto de medidas conocidas como métricas, las cuales proporcionan una referencia de la calidad de algún producto de software (Pressman, 2002).

Para valorar la calidad de los productos de software o sistemas que se desarrollan se proporciona información adecuada sobre los datos referentes de la misma a la calidad del producto, permitiendo una visión más profunda sobre el cumplimiento de los objetivos del proyecto (Pressman, 2002).

Medir la calidad de un software determina una de las tareas más complicadas que se presenta en el desarrollo de un sistema. Pero gracias a esta necesidad se fueron creando diferentes formas de medición de las mismas. Para el presente proyecto implementado utilizaremos el modelo de calidad de la ISO 9126, donde se medirá aspectos como la funcionalidad, fiabilidad, usabilidad y mantenibilidad. (Pressman, 2010)

4.3 NORMA ISO 9126

La norma ISO 9126 (International Standard Organization – Organización Internacional de Normalización) es un estándar internacional para la evaluación de software, que nos ayudará a medir la calidad del sistema siguiendo los siguientes criterios.

4.3.1 FUNCIONALIDAD

Métrica para obtener una valoración mediante el cálculo del punto función en base a la evaluación de un conjunto de características y capacidades que debe cumplir el sistema:

- **Número de entradas de usuario:** Se cuenta cada entrada de usuario que proporciona al software diferentes datos orientados a la aplicación. La tabla 4.1 muestra la lista de entradas de usuario que tiene el sistema.

Nro.	Entrada de Usuario
1	Ingreso al sistema
2	Registro de usuarios
3	Registro de roles
4	Registro de parámetros
5	Registro de categorías
6	Registro de Empresas
7	Registro de transacciones

Tabla 4.1 Lista de entradas de usuario del sistema.
Fuente Elaboración propia.

- **Número de salidas de usuario:** Se refiere a información elaborada por el sistema para ser mostrada al usuario. En la tabla 4.2 se muestra las salidas de usuarios que tiene el sistema.

Nro.	Salida de Usuario
1	Listado de usuarios
2	Listado de roles y permisos
3	Listado de empresas
4	Listado de transacciones

Tabla 4.2 Lista de salidas de usuario del sistema
Fuente Elaboración propia.

- **Número de peticiones de usuario:** Son entradas interactivas entre el usuario y el sistema, donde la salida es inmediata. La tabla 4.3 muestra las peticiones de usuario al sistema.

Nro.	Peticiones de Usuario
1	Autenticación de usuario
2	Listar empresas
3	Listar Roles y Permisos
4	Listar transacciones
5	Mostrar detalle transacción
6	Generar código QR cifrado
7	Enviar transferencia

Tabla 4.3 Lista de peticiones del usuario al sistema
Fuente Elaboración propia.

- **Número de archivos:** Se cuenta cada archivo maestro lógico. En otras palabras, las tablas existentes en la base de datos. La tabla 4.4 muestra los archivos lógicos del sistema.

Nro.	Archivos Lógicos
1	Usuario
2	Roles
3	Permisos

4	Empresa
5	Transacción
6	Detalle transacción
7	Categoría
8	Banco

Tabla 4.4 Lista de archivos lógicos del sistema
Fuente Elaboración propia.

- **Número de interfaces externas:** Se cuenta todas las interfaces legibles por el ordenador que son utilizados para transmitir la información. En este caso es solo internet.

La tabla 4.5 muestra los factores de ponderación con las listas obtenidas anteriormente, para el caso se utilizó el factor medio.

Parámetros de medición	**Cuenta**	**Factores de ponderación**			**Valor Obtenido**
		Simple	**Media**	**Complejo**	
Número de entradas de usuario	7	-	4	-	28
Número de salidas de usuario	4	-	5	-	20
Número de peticiones de usuario	7	-	4	-	28
Número de archivos	8	-	6	-	48
Número de interfaces externas	1	-	2	-	2
Cuenta total					126

Tabla 4.5 Factor de ponderación
Fuente Elaboración propia.

La tabla 4.6 muestra el factor de ajuste de complejidad en base a las respuestas de las siguientes preguntas evaluadas entre 0 y 5.

Factores de complejidad	**Sin influencia**	**Incidental**	**Moderado**	**Medio**	**Significativo**	**Esencial**	**Fi**
	0	**1**	**2**	**3**	**4**	**5**	
¿Requiere el sistema copias de seguridad y de recuperación fiables?				X			3
¿Se requiere comunicación de datos?						X	5
¿Existen funciones de procesamiento distribuido?					X		4
¿Es crítico el rendimiento?					X		4
¿Se ejecutará el sistema con un entorno operativo existente y fuertemente utilizado?					X		4
¿Requiere el sistema entrada de datos interactiva?						X	5
Facilidad Operativa					X		4
¿Se actualiza los archivos maestros de forma interactiva?					X		4
¿Son complejas las entradas, las salidas, los archivos o las peticiones?						X	5
Procesamiento interno complejo				X			3
Diseño de código reutilizable						X	5

Facilidad de Instalación					X		4
¿Soporta múltiples instalaciones en diferentes sitios?						X	5
Facilidad de cambios						X	5
Factor de ajuste de complejidad							60

Tabla 4.6 Factor de ajuste de complejidad
Fuente Elaboración propia.

La funcionalidad es medida a través del punto función (PF), que proporciona una medida objetiva, cuantitativa y auditable del tamaño de la aplicación, basada en la visión del usuario de la aplicación. (Pressman, 2010)

Para calcular el punto función se utiliza la siguiente relación:

$$PF = \text{Cuenta Total} * (X + \text{Min}(Y) * \sum Fi)$$

Dónde:

PF: Medida de funcionalidad

Cuenta Total: Es la suma de los siguientes datos (número de entradas, número de salidas, número de peticiones, número de archivos, número de interfaces externas).

X: Confiabilidad del proyecto, varía entre 1 a 100%.

Min (Y): Error mínimo aceptable al de la complejidad.

$\sum$ **Fi:** Son los valores de ajuste de complejidad, donde $(1 \leq i \leq 14)$.

Para calcular el PF se usa la siguiente ecuación:

$$PF = \text{cuenta total} * (X + \text{Min}(Y) * \sum Fi)$$

$$PF = cuenta\ total * [0.65 + (0.01 * \sum Fi)]$$

Reemplazando los valores obtenidos en las tablas 4.5 y 4.6 se obtiene el siguiente resultado:

$$PF = 126 * [0.65 + (0.01 * 60)]$$

$$PF = 157.5$$

A continuación, calculamos el PF ideal:

$$PFideal = 126 * [0.65 + (0.01 * 70)]$$

$$PFideal = 170.1$$

Entonces la funcionalidad del sistema es:

$$Funcionalidad = (PF / PFidea) * 100$$

$$Funcionalidad = (157.5 / 170.1) * 100$$

$$Funcionalidad = 92.59\%$$

Con el resultado obtenido se puede interpretar que 9 de cada 10 personas, consideran que el sistema responde de manera óptima a las funcionalidades requeridas por la empresa.

4.3.2 CONFIABILIDAD

Para determinar la confiabilidad de un sistema, se toma en cuenta las fallas que puedan ocurrir en el sistema en un tiempo determinado. En el desarrollo de software las fallas son más que todo por diseño e implementación. Para medir el tiempo medio entre fallos (TMEF) se usará la siguiente fórmula:

$$TMEF = TMDF + TMDR$$

Donde:

TMDF: Tiempo medio de fallo.

TMDR: Tiempo medio de reparación

Se estima que un fallo puede ocurrir cada 20 días hábiles y su reparación en promedio pueda tomar 1 hora después de haber entregado una nueva funcionalidad del sistema, entonces:

$$TMEF = (20*8) + 1$$

$$TMEF = 161 \text{ horas}$$

Por lo que la disponibilidad es un buen indicador de fiabilidad, en base de la siguiente fórmula se tiene:

$$Disponibilidad = (TMDF / TMEF) * 100$$

$$Disponibilidad = (160 / 161) * 100 = 99{,}4\%$$

Con lo que se llega a la conclusión de que el sistema tiene un **99.4% de confiabilidad**.

4.3.3 USABILIDAD

La usabilidad representa facilidad de uso que el usuario final recibirá del sistema. Esta métrica nos muestra el esfuerzo necesario para aprender a manipular el sistema. (Pressman, 2010)

La tabla 4.7 muestra los resultados obtenidos en base a preguntas propuestas a los usuarios del sistema.

Nro.	Pregunta	Valor 0 - 100
1	¿Es entendible?	91%
2	¿Las pantallas son agradables a la vista del usuario?	95%
3	¿Es fácil de aprender?	92%
4	¿Contiene información necesaria?	90%
5	¿Facilita su trabajo?	86%
6	¿La navegabilidad es fluida?	94%
Promedio		91.3%

Tabla 4.7 Preguntas para obtener el grado de usabilidad
Fuente Elaboración propia.

Entonces la **usabilidad del sistema sería del 91.3%**, lo que indica que 9 de cada 10 usuarios pueden utilizar el sistema con facilidad.

4.3.4 MANTENIBILIDAD

Para la evaluación de la mantenibilidad, se desarrolló algunas preguntas, estas preguntas son valoradas en porcentaje por el desarrollador del sistema al momento de la culminación del proyecto.

Este valor tiene consideración por la experiencia y la forma de trabajo de cada programador, el mismo puede ser relativo respecto a otros desarrolladores (Largo y Marin, 2005).

La tabla 4.8 muestra las preguntas y los resultados obtenidos en la evaluación de mantenibilidad, estas preguntas se las hizo a todo el equipo de desarrollo del sistema.

Factor de ajuste	Valor
¿Se puede modificar el sistema?	97%
¿Deja identificar las partes que deben ser modificadas?	95%
¿Permite implementar una modificación específica?	93%
¿Presenta efectos inesperados como posibles errores?	98%
Total	95.75%

Tabla 4.8 Evaluación de mantenibilidad
Fuente Elaboración propia.

El resultado de la **mantenibilidad es de 95.75%**, lo que significa que el esfuerzo necesario para realizar mantenimiento al sistema es mínimo.

4.3.5 PORTABILIDAD

La portabilidad es la capacidad con que un software puede ser llevado de un entorno a otro, considera la facilidad de instalación, ajuste y adaptación al cambio. Para medir la portabilidad del sistema usaremos la siguiente relación:

$$GP = 1 - (ET / ER)$$

Donde:

GP: Grado de portabilidad

ET: Recursos necesarios para llevar el sistema a otro entorno.

ER: Recursos necesarios para crear el sistema en el entorno residente.

Si:

GP > 0, la portabilidad es más rentable que el re-desarrollo.

GP < 0, el re-desarrollo es más rentable que la portabilidad.

GP = 0, la portabilidad es perfecta

Por lo que los valores obtenidos son: ET = 9 y ER = 10

$$GP = 1 - (9 / 10)$$

$$GP = 0.90$$

Con el resultado obtenido sabemos que el **grado de portabilidad es del 90%**, entonces la portabilidad del sistema es más rentable que su re-desarrollo.

4.3.6 CALIDAD GLOBAL

Para poder obtener la calidad global del sistema, se saca la media de todas las medidas expresadas en porcentaje hasta el momento, funcionalidad, confiabilidad, usabilidad, mantenibilidad y portabilidad. (Pressman, 2010)

La tabla 4.9 muestra la calidad global del sistema expresado en porcentajes.

Criterios	**Resultado**
Funcionalidad	91.1%
Confiabilidad	99.4%
Usabilidad	91.3%
Mantenibilidad	95.75%
Portabilidad	90%
Calidad Global	93.51%

Tabla 4.9 Calidad Global
Fuente Elaboración propia.

El resultado de la **calidad global es de 93.51%**, con este resultado concluimos que 9 de cada 10 usuarios consideran al sistema web de calidad.

4.4 SEGURIDAD

Los problemas de seguridad en sistemas web, pueden venir de las herramientas que se utilizan en el momento del desarrollo o producto de un diseño lógico que no se contempló de las posibles amenazas que pueda surgir como ser (Meucci, 2008):

- Entradas no válidas.
- Control de accesos rotos.
- Sesiones y Autenticaciones no controladas.
- Ataques *Cross Site Scripting*.
- Inyección de códigos.
- Manejo Inadecuado de Errores

Para las medidas de seguridad para el sistema desarrollado se contemplan dos aspectos importantes y vulnerables que están en el lado del cliente y lado del servidor.

4.4.1 SEGURIDAD DEL LADO DEL CLIENTE

Uno de los mecanismos de seguridad que se implementan son las validaciones por el lado del cliente. Existen mecanismos de validación provistos por las herramientas que utilizamos para hacer la aplicación, HTML cuenta con atributos para validar datos requeridos, numéricos, formato de correos, etc. estas validaciones son realizadas antes de que la información introducida llegue al servidor, esto evita que se envíen datos incorrectos al servidor, además se ahorra tiempo, ya que si la información es incorrecta simplemente no se envía al servidor. (Meucci, 2008)

Las medidas que se implementó en el lado del servidor del sistema es la autenticación de usuarios, los únicos que tienen acceso al sistema son el personal de la empresa, estos están registrados como usuarios y con su contraseña respectiva.

4.4.2 SEGURIDAD DEL LADO DEL SERVIDOR

El desarrollo de una aplicación web requiere de una serie de herramientas: servidores web, servidores de base de datos, servidores de aplicaciones, lenguajes de programación del lado del servidor, etc. Las vulnerabilidades mencionadas que pueden comprometer la seguridad de un sistema web son las siguientes (Meucci, 2008):

- Versiones no actualizadas.
- Configuraciones por defecto inadecuadas.
- Cuentas por defecto no modificadas.

CAPÍTULO V

COSTO Y BENEFICIO

5.1 INTRODUCCIÓN

En el desarrollo de este capítulo, se dará a conocer a la empresa Consultoría y Desarrollo de Software MC4 S.R.L. que en la implementación y utilización del sistema se obtendrán muchos beneficios.

Para tal efecto se realiza el análisis de costo del sistema usando el método COCOMO II y en base al costo del sistema y otros gastos se determina la rentabilidad del sistema con el cálculo del valor neto actual y la tasa interna de retorno.

Después de realizar los cálculos necesarios para la obtención de los resultados esperados estaremos en la capacidad de afirmar si el proyecto es viable, rentable y comprobar que es buena opción invertir en el proyecto.

5.2 COCOMO II

El Modelo Constructivo de Costes (COCOMO) es un modelo matemático de base empírica, utilizando para la estimación de costes de software. Incluye tres submodelos, cada uno ofrece un nivel de detalle y aproximación, cada vez mayor, a medida que avanza el proceso de desarrollo del software: básico, intermedio y detallado.

En su libro clásico acerca de economía de la ingeniería de software; Barry Boehm introdujo una jerarquía de modelos de estimación de software que llevan el nombre de COCOMO. El modelo original se convirtió en uno de los modelos de estimación de costo más ampliamente utilizados y estudiados en la industria. Evolucionó hacia un modelo de estimación más exhaustivo llamado COCOMO II. (Pressman, 2010)

COCOMO II consta con tres modelos de estimación, los mismos se representan en 3 ecuaciones:

$$E = a(KLDC)^b, \frac{Personas}{mes}$$

$$D = c(E)^2, mes$$

$$P = \frac{E}{D}, personas$$

Donde:

E: Esfuerzo requerido por el proyecto expresado en persona-mes.

D: Tiempo requerido por el proyecto expresado en meses.

P: Número de personas requeridas para el proyecto.

a, b, c y d: Constantes con valores definidos según cada sub-modelo.

KLDC: Cantidad de líneas de código, en miles.

A la vez cada modelo se subdivide en modos, los mismos son:

- **Modo orgánico:** Es un pequeño grupo de programadores experimentados desarrollando proyectos de software en un entorno familiar. El tamaño del software varía desde unas pocas líneas (tamaño pequeño) a miles (medio).
- **Modo semi – libre o semi – acoplado:** Corresponde a un esquema intermedio entre el modo orgánico y el rígido, el grupo de desarrollo puede incluir una mezcla de personas experimentadas y no experimentadas.
- **Modo rígido o empotrado:** El proyecto tiene fuertes restricciones, que pueden estar relacionadas con la funcionalidad y/o pueden ser técnicas. El problema a resolver es único, siendo difícil basarse en la experiencia puesto que puede no haberla.

En la siguiente tabla 5.1 se muestra los coeficientes del producto de software de acuerdo a los tres modos expuestos anteriormente

Proyecto de Software	a	b	c	d
Orgánico	2.4	1.05	2.5	0.38
Semilibre	3.0	1.12	2.5	0.35
Rígido	3.6	1.20	2.5	0.32

Tabla 5.1 Constante a, b, c, d COCOMO II
Fuente: Coeficientes del producto de software (Pressman, 2010)

5.3 COSTO DEL SISTEMA

El costo del sistema se lo plantea en dos partes:

5.3.1 COSTO DE DESARROLLO DEL SISTEMA

Para el cálculo del desarrollo del software se tendrá como partida el punto función no ajustada que se encontró en el capítulo anterior, cuyo valor encontrado es:

$$PF = 157{,}5$$

La tabla 5.2 muestra la relación para convertir el valor de PF a KLDC (Kilos de líneas de código)

Lenguaje	Nivel	Factor LDC / PF
C	2.5	128
Ansi Basic	5	64
Java	6	53
PL / I	4	80
Ansi Cobol 74	3	107
Visual Basic	7.00	46
ASP	9.00	36
PHP	11.00	29
Visual C++	9.50	34

Tabla 5.2 Conversión de PF a KLDC
Fuente Elaboración propia

Entonces, realizando los cálculos y escogiendo el valor del lenguaje de programación JAVA de la tabla 19, tenemos:

$$LDC = PF * Factor\ LDC/PF$$

$$LDC = 157{,}5 * 53$$

$$LDC = 8347{,}5$$

$$KLDC = 8347{,}5 / 1000$$

$$KLDC = 8{,}3475$$

Ahora, para hallar el esfuerzo reemplazamos los valores obtenidos hasta ahora:

$$E = 2.4 * (8{,}3475)^{1.05} = 22{,}28$$

$$D = 2.5 * (22{,}28)^{0.38} = 8{,}13$$

Para el cálculo del número de programadores para el desarrollo:

$$P = (22{,}28 / 8{,}13) = 2{,}74 \cong 3$$

Estimando que el salario medio de un programador es de Bs. 2500, esta cifra será tomada en cuenta para la siguiente estimación:

Costo del software desarrollado = Numero de programadores * Salario de un programador (Pressman, 2002).

Costo del Software Desarrollado por Persona = 3 * 2.500 = 7500 Bs.

Costo total del Software Desarrollado = 7500 * 5 = 37.500,00 Bs.

Lo que significa que el costo del sistema desarrollado es de Bs. 37.500,00 por los 5 meses.

5.3.2 COSTO DE ELABORACIÓN DEL PROYECTO

La tabla 5.3 muestra los costos de inversión de los recursos que se usaron para la elaboración del sistema.

Recursos	**Costo (Bs)**
Material de escritorio	250
Investigación del proyecto	300
Internet	600
Otros	600
Total	1750

Tabla 5.3 Costo de recursos empleados para la elaboración del sistema
Fuente Elaboración propia.

Por tanto, el costo de la elaboración del proyecto es de Bs. 1750.

5.3.3 COSTO TOTAL DEL SISTEMA

El costo total del sistema se obtiene de la sumatoria del costo de desarrollo y el costo de elaboración del proyecto, en la tabla 5.4 se puede observar los resultados, todos los costos están expresados en bolivianos.

Detalle	**Costo (Bs)**
Costo de desarrollo	37.500,00
Costo de elaboración del proyecto	1.750,00
Total	39.250,00

Tabla 5.4 Costos totales del sistema
Fuente Elaboración propia.

Entonces, el costo total del sistema es de Bs. 39.250,00

5.4 VALOR ACTUAL NETO

El valor actual neto (VAN), es un procedimiento que permite calcular el valor presente de un determinado número de flujos de caja futuros, originados por una inversión. La metodología consiste en descontar al momento actual (es decir, actualizar mediante una tasa) todos los flujos de caja futuros del proyecto. A este valor se le resta la inversión inicial, de tal modo que el valor obtenido es el valor actual neto del proyecto (Bu, 2009). La fórmula que utilizaremos para hallar el valor actual neto será:

$$\text{VAN} = \frac{\text{ganancias}}{(1+\text{k})^{\text{n}}} - \frac{\text{costos}}{(1+\text{k})^{\text{n}}}$$

Dónde:

VAN: Valor actual neto.

ganancias: ingreso de flujo anual.

costos: salidas de flujo anual.

n: número de periodo.

k: tasa de descuento o tasa de interés de préstamo.

Además, tenemos que, si:

VAN > 0, el proyecto es rentable

VAN = 0, no hay pérdidas ni ganancias

VAN < 0, el proyecto no es rentable

En la tabla 5.5 mostramos los gastos y las ganancias que se estiman en un lapso de 5 años, tomando en cuenta la tasa de descuento del 10%.

Año	Costo (Bs)	Ganancias	costos/(1+k)n	ganancias/(1+k)n
1	39.250,00	0	35.681,81	0,00
2	20.000,00	10.000,00	16.528,00	8.264,40
3	10.000,00	20.000,00	7.513,10	15.026,00

4	5.000,00	30.000,00	3.415,00	20.490,00
5	0	39.250,00	0,00	24.371,00
Total			63.137,91	68.151,40

Tabla 5.5 Estimación de gastos y ganancias en 5 años
Fuente Elaboración propia.

$$VAN = 5.013,49$$

Con este resultado concluimos que nuestro proyecto es rentable ya que 5.013,49 es mayor a cero.

5.5 COSTO BENEFICIO

Para encontrar la relación costo/beneficio de un proyecto se aplica la siguiente ecuación:

$$costo/beneficio = \sum ganacias \,/\, \sum costos$$

Entonces, reemplazando los valores obtenidos en la tabla 5.5 en la ecuación anterior:

$$costo/beneficio = 99.250,00 \,/\, 74.250,00$$

$$costo/beneficio = 1,34$$

Entonces de aquí por cada boliviano invertido en el sistema, la empresa genera una ganancia de Bs. 0,34.

5.6 TASA INTERNA DE RETORNO

La tasa interna de retorno (TIR) es una tasa porcentual que indica la rentabilidad promedio anual, es decir es la máxima tasa de descuento que puede tener un proyecto para que sea rentable. La fórmula para el cálculo del TIR es:

$$TIR = \sum_{t=1}^{n} \frac{B^t}{(1+i)^t} - Io = 0$$

Donde de esta ecuación se tiene que despejar k y ese valor sería nuestra TIR:

Si:

TIR > k, el proyecto es rentable

TIR = k, no hay pérdidas ni ganancias

TIR < k, el proyecto no es rentable

Se debe aclarar que el valor k que se compara con TIR, es el k usado en la ecuación de VAN.

Entonces, para para hallar nuestra TIR:

$$TIR = 0.27$$

$$TIR = 27\%$$

Como 0,27 > 0,10 concluimos que el proyecto es rentable.

CAPÍTULO VI

SEGURIDAD DEL SISTEMA

6.1. INTRODUCCIÓN

Hoy en día todas las organizaciones y personas utilizan dispositivos inteligentes, computadoras, redes inalámbricas, etc. Y están expuestas a diferentes amenazas cibernéticas derivadas de la utilización de páginas web, apps, documentos, correos electrónicos, servicios de chat, redes sociales, etc. (Castro, 2016)

La mayoría de estas amenazas están siendo creadas para extraer información personal o corporativa y con esto realizar ataques dañinos que vulneran nuestra capacidad para realizar transacciones, acceso a documentos, sistemas internos, etc. (Castro, 2016)

Mientras que por un lado hoy tenemos a la disposición cientos de servicios de interconexión entre personas y organizaciones, por el otro estamos teniendo mucha mayor exposición de nuestra información personal y corporativa hacia personas no autorizadas que utilizan diferentes métodos para atacar y estos están siendo cada vez más complejos, más difíciles de prevenir y sobre todo más dañinos. Esto ha llevado a las organizaciones a poner mucho más énfasis en la ciberseguridad y los aspectos preventivos y correctivos ante un ataque. (Castro, 2016)

Dentro de una correcta planeación de protección preventiva y correctiva se debe de considerar el análisis de vulnerabilidades como una actividad clave para asegurar que estamos al día ante la creciente ola de amenazas que día a día va creciendo de manera exponencial. (Castro, 2016)

6.1. ANALISIS DE VULNERABILIDADES

Por definición una vulnerabilidad informática se puede considerar como una debilidad de cualquier tipo que afecta o compromete la seguridad de un componente informático.
Las vulnerabilidades informáticas las podemos agrupar en función de:

- Diseño de la seguridad perimetral
- Debilidad en el diseño de protocolos utilizados en las redes.
- Políticas de seguridad deficientes e inexistentes.
- Implementación

- Errores de programación.
- Existencia de “puertas traseras” en los sistemas informáticos.
- Descuido de los fabricantes.
- Uso
- Configuración inadecuada de los sistemas informáticos.
- Desconocimiento y falta de sensibilización de los usuarios y de los responsables de informática.
- Disponibilidad de herramientas que facilitan los ataques.
- Limitación gubernamental de tecnologías de seguridad.
- Vulnerabilidad del día cero

6.1.1. OWASP

Según Ciberseguridad (2019) el proyecto OWASP *Application Security Verification Standard* (ASVS) proporciona una base para probar los controles técnicos de seguridad de las aplicaciones web y también proporciona a los desarrolladores una lista de requisitos para un desarrollo seguro.

El objetivo principal del proyecto OWASP *Application Security Verification Standard* (ASVS) es normalizar el rango de cobertura y nivel de rigor disponible en el mercado a la hora de realizar la verificación de seguridad de aplicaciones web utilizando un estándar abierto comercialmente viable.

El estándar proporciona una base para probar los controles técnicos de seguridad de la aplicación, así como cualquier control técnico de seguridad en el entorno, en el que se confía para proteger contra vulnerabilidades como *Cross-Site Scripting* (XSS) e inyección SQL. Este estándar se puede utilizar para establecer un nivel de confianza en la seguridad de las aplicaciones web.

Los requisitos se desarrollaron con los siguientes objetivos en mente:

- Uso como métrica: proporciona a los desarrolladores y propietarios de aplicaciones un criterio con el que evaluar el grado de confianza que se puede depositar en sus aplicaciones web,
- Utilización como guía: brinda orientación a los desarrolladores de control de seguridad sobre qué incorporar en los controles de seguridad para satisfacer los requisitos de seguridad de la aplicación,
- Uso durante la adquisición: proporciona una base para especificar los requisitos de verificación de seguridad de la aplicación en los contratos.

El Proyecto de Guía de Pruebas de Seguridad Web produce el principal recurso de pruebas de seguridad cibernética para desarrolladores de aplicaciones web y profesionales de seguridad.

Es una guía completa para probar la seguridad de las aplicaciones web y los servicios web. Creado por los esfuerzos de colaboración de profesionales de ciberseguridad y voluntarios dedicados, proporciona un marco de las mejores prácticas utilizadas por los probadores de penetración y las organizaciones de todo el mundo.

6.1.1.1 PRINCIPALES VULNERABILIDADES

OWASP establece y explica las diez vulnerabilidades más importantes que pueden aparecer en un sitio web.

Los atacantes pueden usar diferentes rutas a través de la aplicación de un negocio para causar importantes daños al mismo.

El riesgo total para una empresa viene dado por la unión de:

- La probabilidad asociada con cada agente de amenaza
- Vector de ataque
- Debilidad de seguridad
- Estimación del impacto técnico
- Estimación del impacto para el negocio

Cada aplicación y cada empresa son diferentes por lo que habrá que evaluar el riesgo en cada caso enfocándonos en:

- Agentes amenazantes
- Controles de seguridad
- Impacto de negocio

Las diez vulnerabilidades más importantes y comunes de las aplicaciones web son:

- Inyección SQL
- Secuencia de comandos en sitios cruzados (XSS)
- Pérdida de autenticación y gestión de sesiones
- Referencia directa insegura a objetos
- Falsificación de peticiones en sitios cruzados
- Configuración de seguridad defectuosa
- Almacenamiento criptográfico inseguro
- Fallo de restricción de acceso a URL
- Protección insuficiente en la capa de transporte
- Redirecciones y reenvíos no validados

6.1.1.2. OWASP ZAP

OWASP ZAP (Zed Attack Proxy). esta plataforma está diseñada especialmente para monitorizar la seguridad de las aplicaciones web de las compañías, siendo una de las aplicaciones del proyecto más activas en cuanto a auditorías de seguridad. (Velasco, 2015)

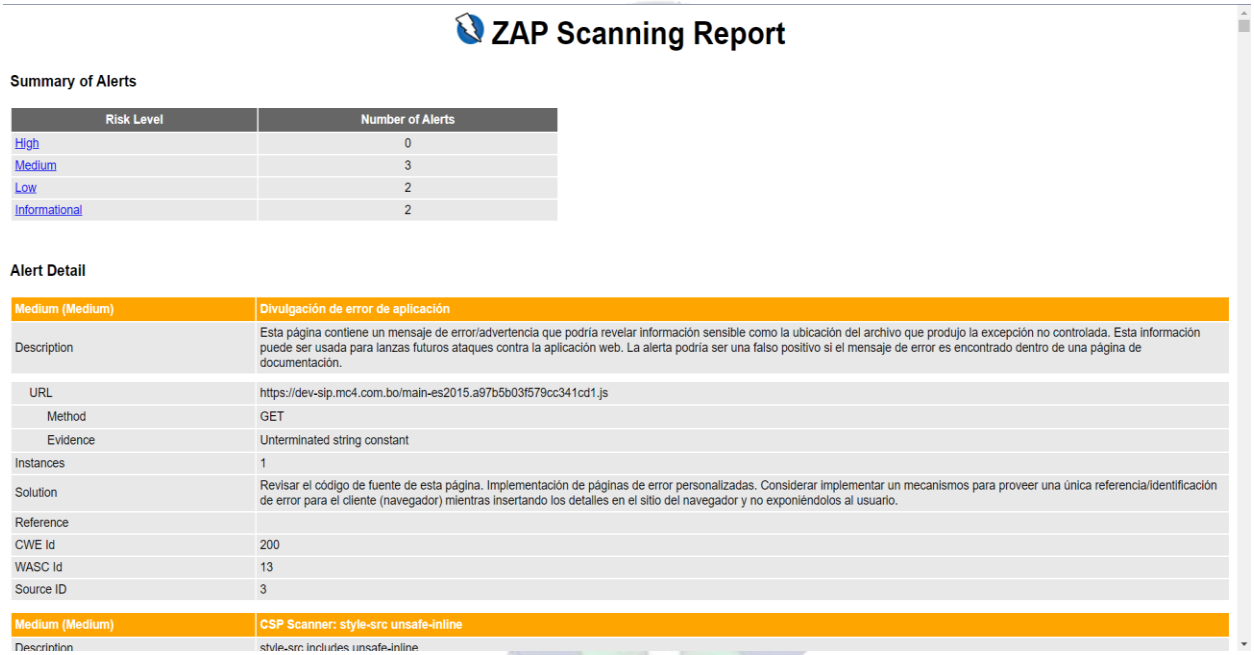
Velaszo (2015) define las principales características de OWASP ZAP:

- Posibilidad de comprobar todas las peticiones y respuestas entre cliente y servidor.
- Posibilidad de localizar recursos en un servidor.
- Análisis automáticos.
- Análisis pasivos.
- Posibilidad de lanzar varios ataques a la vez.
- Capacidad para utilizar certificados SSL dinámicos.
- Soporte para utilizar tarjetas inteligentes (DNI-e, por ejemplo) y certificados personales.
- Análisis de sistemas de autenticación.
- Posibilidad de actualizar la herramienta automáticamente.
- Dispone de una tienda de extensiones (*plugins*) con las que añadir más funcionalidades a la herramienta

6.1.1.3 ESCANEO DE SEGURIDAD CON OWASP ZAP

Para el análisis de vulnerabilidades del proyecto Plataforma de Pagos de Servicios a través de imágenes QR se realizó el escaneo de seguridad utilizando la herramienta OWASP ZAP.

En la figura 6.1 se puede evidenciar los resultados de dicho escaneo:



ZAP Scanning Report

Summary of Alerts

Risk Level	Number of Alerts
High	0
Medium	3
Low	2
Informational	2

Alert Detail

Medium (Medium)	Divulgación de error de aplicación
Description	Esta página contiene un mensaje de error/advertencia que podría revelar información sensible como la ubicación del archivo que produjo la excepción no controlada. Esta información puede ser usada para lanzas futuros ataques contra la aplicación web. La alerta podría ser una falso positivo si el mensaje de error es encontrado dentro de una página de documentación.
URL	https://dev-sip.mc4.com.bo/main-es2015.a97b5b03f579cc341cd1.js
Method	GET
Evidence	Unterminated string constant
Instances	1
Solution	Revisar el código de fuente de esta página. Implementación de páginas de error personalizadas. Considerar implementar un mecanismos para proveer una única referencia/identificación de error para el cliente (navegador) mientras insertando los detalles en el sitio del navegador y no exponiéndolos al usuario.
Reference	
CWE Id	200
WASC Id	13
Source ID	3

Medium (Medium)	CSP Scanner: style-src unsafe-inline
Description	style-src includes unsafe-inline

Figura 6.1 Escaneo de Vulnerabilidades con la Herramienta OWASP ZAP
Fuente: Elaboración Propia

En el escaneo de vulnerabilidades se pudo evidenciar que no existen riesgos elevados, sin embargo, se debido para la entrega del producto el área de QA requiere que se subsane todas las alertas o caso contrario se presente el informe de justificación en el caso de ser un falso positivo

6.1.2. SONARQUBE

La documentación de SonarQube (2020) afirma que es una herramienta de revisión automática de código para detectar errores, vulnerabilidades y malas prácticas en el código. Puede integrarse al flujo de trabajo para permitir la inspección continua de código en las ramas del proyecto y en las solicitudes de extracción de código.

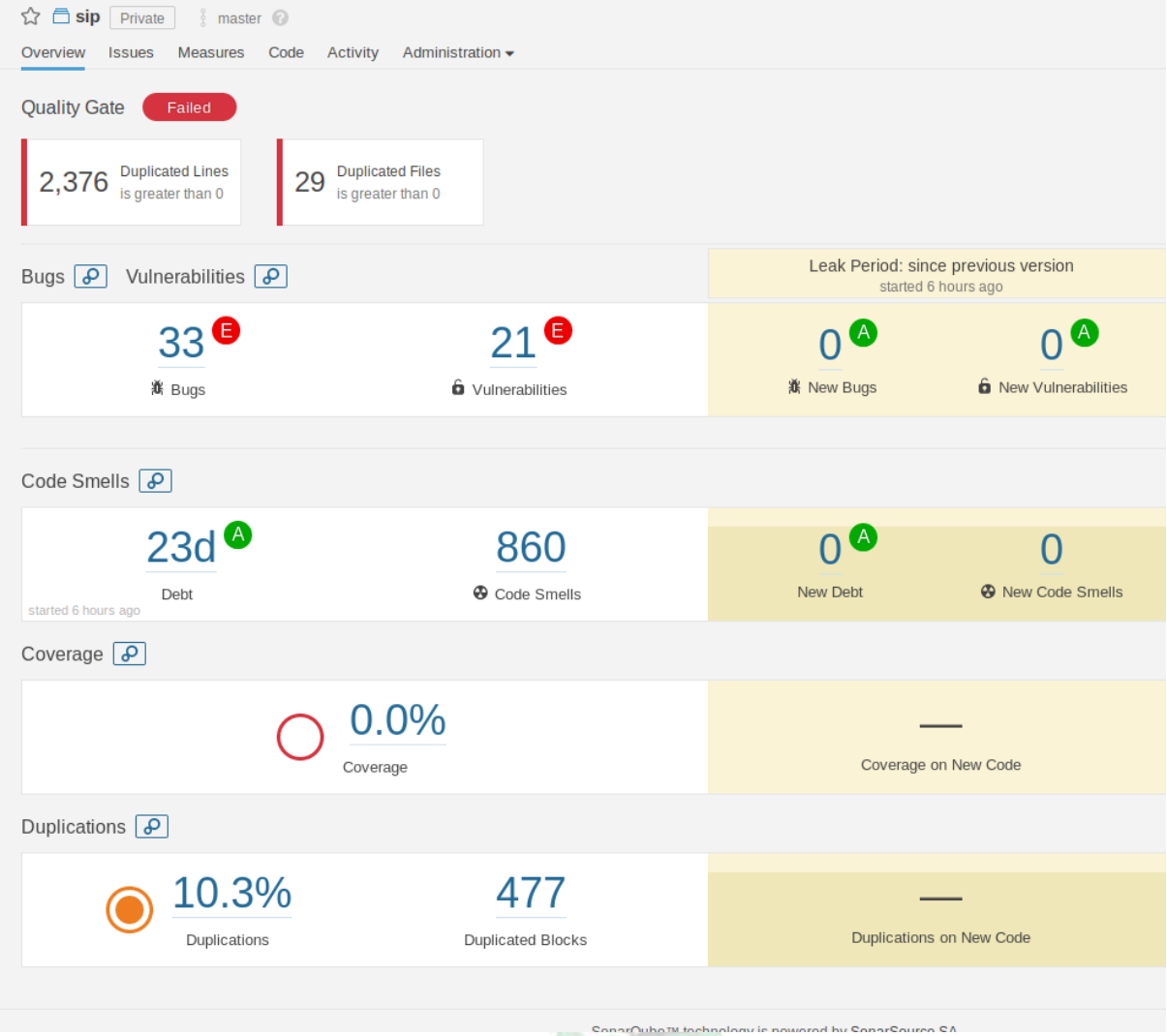


Figura 6.2: Escaneo de Vulnerabilidades con la herramienta Sonarqube
Fuente: Elaboración Propia

CAPÍTULO VII

CONCLUSIONES Y RECOMENDACIONES

7.1. CONCLUSIONES

Una vez de haber terminado con el desarrollo y la implementación del sistema para la empresa Consultoría y Desarrollo de Software MC4 S.R.L., aplicando todas las metodologías de análisis y diseño de software en base a los objetivos propuestos, se puede afirmar que se han cumplido con todos los objetivos. Entonces, se concluye que:

- Se implementó la plataforma de pagos de servicios SIP cumpliendo con los estándares de seguridad de la empresa MC4 integrando las deudas de servicios que se obtiene de la integración con la plataforma UNICO y el pago mediante QR del sistema ACH EXPRESS.
- Se implementó la plataforma que agrupa los pagos de servicio con un solo modo de pago disponible para las 14 entidades financieras más grandes del país.
- Se implementó la generación de QRs con la integración al sistema ACH EXPRESS facilitando de esta forma la complejidad de realizar el pago de un servicio.
- Se implementó la plataforma con acceso público y la experiencia de usuario necesaria para no crear fricción con el usuario final.
- Se implementó un API estándar de integración con la plataforma UNICO lo cual permite disponibilizar cualquier empresa registrada en UNICO sin realizar un desarrollo adicional
- Se implementó los Servicio Web para Generación de QR según ISO 18004:2015.
- Se implementó la Integración mediante servicios web con la plataforma UNICO.
- Se implementó la Integración mediante servicios web con el sistema ACH EXPRESS.
- Se implementó el uso de *Apikey* de seguridad para la integración mediante APIs.
- Se implementó la parametrización de Empresas y servicios para configuración de cuentas recaudadoras.
- Se implementó parametrización de información para el funcionamiento correcto de las transacciones.
- Se implementó una plataforma como aplicación progresiva y responsiva lo cual permite la navegación por cualquier dispositivo sin decrementar la experiencia de usuario

7.2. RECOMENDACIONES

Las recomendaciones para mejorar la plataforma web son las siguientes:

- Se recomienda implementar el módulo de conciliación automática, para la generación del reporte de todas las transacciones realizadas hasta finalizar el día, mes o periodo parametrizable, para un mejor control en los ingresos de la entidad financiera y la empresa de servicios.
- Se recomienda la parametrización de 5 reintentos al momento de la confirmación del pago y en caso de error enviar correos de notificación con la información que debe procesarse
- Se recomienda dar de baja a los usuarios que por algún motivo ya no pertenecen a la empresa.

FUENTES DE INFORMACIÓN

Banco Central de Bolivia. (1 de Junio de 2019). *Evolución al acceso y uso de la banca electrónica en Bolivia.* Obtenido de http://eeb.bcb.gob.bo/revisores/uploads/10309.PDF

Beltran, M. (2008). *Modelos de Estimación.* Valencia: Tesis.

Boehm. (2012). COCOMO II Model Definition Manual. *COCOMO II Model Definition Manual*.

Castro, I. (26 de Julio de 2016). *Cero Uno Software Coorporativo*. Obtenido de https://cerounosoftware.com.mx/2016/07/26/que-es-un-analisis-de-vulnerabilidades-inform%C3%A1ticas/

Ciberseguridad. (2019). *Ciberseguridad.com.* Obtenido de https://ciberseguridad.com/guias/desarrollo-seguro/owasp/

Consultoría y Desarrollo Tecnológico MC4 S.R.L. (2017). *MC4 Software & Solutions*. Obtenido de https://mc4.com.bo/

EKCIT. (2019). *tic.PORTAL*. Obtenido de https://www.ticportal.es/temas/cloud-computing/amazon-web-services

Figueroa, M. A. (2000). *Normas ISO-9126.* Obtenido de http://www.javier8a.com/

Google. (2020). *Angular*. Obtenido de https://angular.io/

Khipu. (2019). *khipu*. Obtenido de https://bo.khipu.com/

Koch, N. (2002). *The expresive Power of UML - Based Web Engineering.* Munich: Universität München Deutschland.

Meucci, M. (2008). *Guía de Pruebas OWASP.* Owasp Foundation. Obtenido de https://owasp.org/www-pdf-archive/Gu%C3%ADa_de_pruebas_de_OWASP_ver_3.0.pdf

Nevada Media. (03 de 2018). *DZone*. Obtenido de https://dzone.com/articles/introduction-to-docker-1

Nieves Guerrero, C. G., Ucán Pech, J. P., & Méndez Domínguez, V. H. (Junio de 2014). *ResearchGate.* Obtenido de https://www.researchgate.net/publication/280580830_UWE_en_Sistema_de_Recomendacion_de_Objetos_de_Aprendizaje_Aplicando_Ingenieria_Web_Un_Metodo_en_Caso_de_Estudio

Offutt, J. (2002). Quality Attributes of Web Software Applications. *IEEE Software*, 25-32. Obtenido de https://www.researchgate.net/

Olsina. (1999). Specifying Quality Characteristics and attributes for web sites. *ACM*, 40-48.

Oracle. (2020). *Java*. Obtenido de https://www.java.com

Palacios, J. (2020). *Deloitte.* Obtenido de Scrum: roles y responsabilidades: https://www2.deloitte.com/es/es/pages/technology/articles/roles-y-responsabilidades-scrum.html

Pressman, R. S. (2010). *Ingeniería del Software, Un enfoque práctico 7ma. Edición.* México: Mc Graw Hill.

Rodríguez, D., & Harrison, R. (2005). *Departamento Cientas Computación.* Obtenido de http://www.cc.uah.es/

Rossi, G. P., Schawabe, O., & Olsina, L. (2008). *Web Engineering - Modelling and Implementing Web Applications.* Londres: Springer.

Saunders, A. (1 de Enero de 2020). *OpenMid BBVA.* Obtenido de El impacto de la tecnología en el crecimiento y el empleo: https://www.bbvaopenmind.com/articulos/el-impacto-de-la-tecnologia-en-el-crecimiento-y-el-empleo/

School, W. C. (13 de Junio de 2019). *Wild Code School*. Obtenido de https://www.wildcodeschool.com/es-ES/blog/que-es-un-framework

Schwaber, K., & Sutherland, J. (Julio de 2013). *Scrumguides.* Obtenido de La Guía de Scrum: https://www.scrumguides.org/docs/scrumguide/v1/scrum-guide-es.pdf

Scrum.org. (2020). *Scrum.org The home od Scrum*. Obtenido de scrum.org

Sintesis S.A. (2019). *pagosnet*. Obtenido de http://www.pagosnet.com.bo/

Sommerville, I. (2005). *Ingeniería del Software 7ma. Edición.* Madrid: Pearson Addison Wesley.

SonarQube. (2020). *SonarQube.org*. Obtenido de https://www.sonarqube.org/features/enhance-your-workflow/

The PostgreSQL Global Development Group. (2020). *PostgreSQL*. Obtenido de https://www.postgresql.org/

Velasco, R. (25 de Abril de 2015). *RZ Redes Zone.* Obtenido de https://www.redeszone.net/2015/04/25/seguridad-web-owasp-zap/

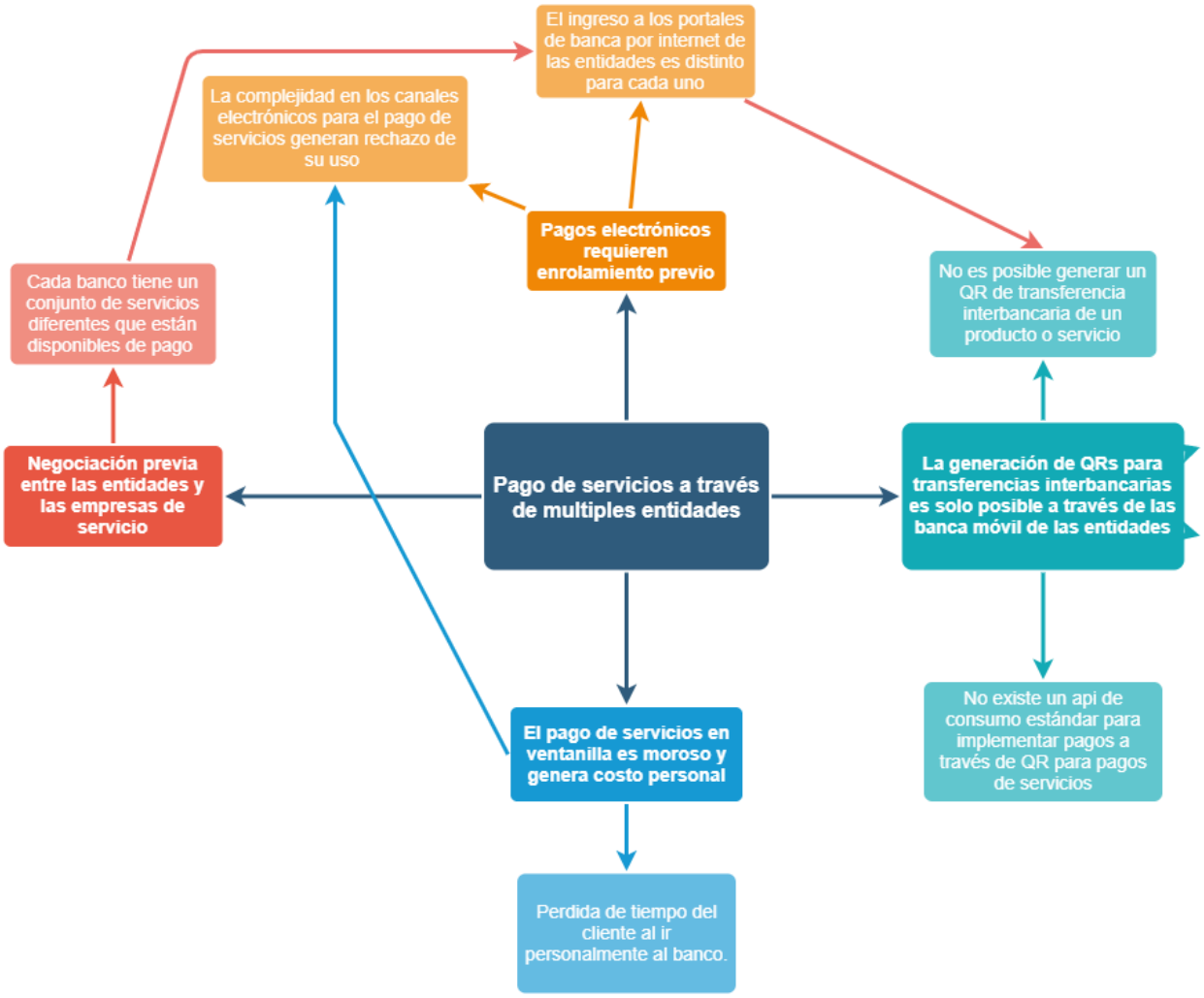
Yin, R. K. (2014). *Case Study Research Design and Methods.* California: Thousand Oaks.

ANEXOS

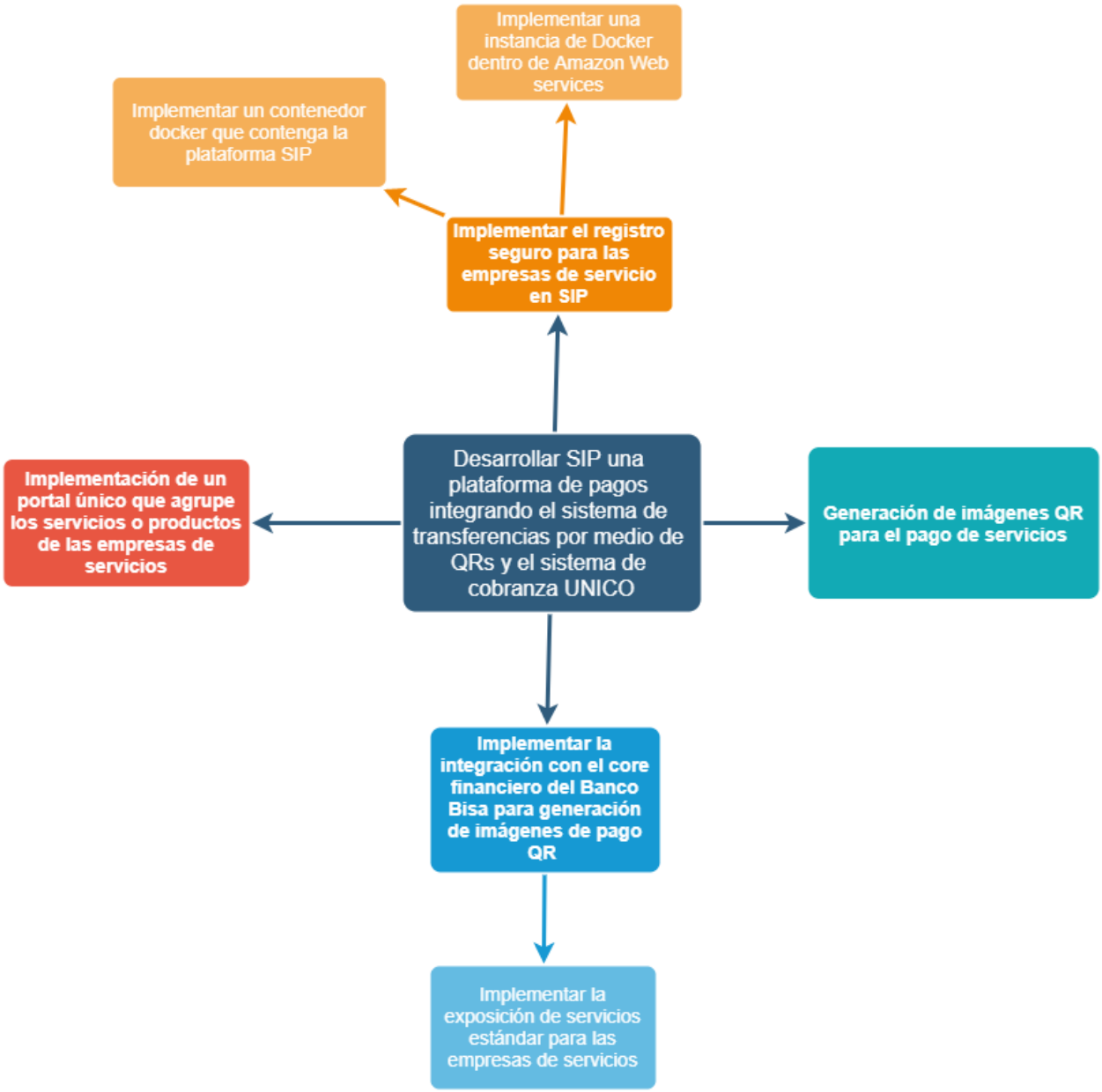
CRONOGRAMA

Actividades	Duración en días	Del 1 de Julio a 25 de Enero																											
		Julio				Agosto				Septiembre				Octubre				Noviembre				Diciembre				Enero			
		1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Análisis y Diseño	3																												
Definición Product Backlog	3																												
Planificación de Sprint 1	1																												
Sprint 1																													
Creación estructura Base de datos	2																												
Creación estructura de proyecto Backend	4																												
Creación estructura Frontend	4																												
Mapeo de Objetos y Servicios Modulo de Administración	10																												
Planificación de Sprint 2	1																												
Sprint 2																													
Modulo de Logueo y seguridad	5																												
Logueo y permisos de navegabilidad	2																												
Modulo de Administración y permisos	8																												
Pantallas Módulo de Administración	5																												
Planificación de Sprint 3	1																												
Sprint 3																													
Modulo de servicios web Entidades Banco y Empresa y Servicios	10																												
Pantallas Modulo Entidades	10																												
Planificación de Sprint 4	1																												
Sprint 4																													
Modulo de Servicios Web Transferencias	10																												
Pantallas Modulo de Transferencias	10																												
Planificación de Sprint 5	1																												
Sprint 5																													
Integración Core Entidad Financiera	10																												
Integración Core Empresa de Servicios	10																												
Planificación de Sprint 6	1																												
Sprint 6																													
Pruebas de Flujo Generación QR	5																												
Pruebas de Flujo Confirmar QR	5																												
Pruebas de Flujo Inhabilitación QR	5																												
Pruebas de Flujo Obtiene Transacción	5																												
Planificación de Sprint 7	1																												
Sprint 7																													
Pruebas Integrales	10																												
Coverity	5																												

ÁRBOL DE PROBLEMAS



ÁRBOL DE OBJETIVOS



MATRIZ DE MARCO LÓGICO

Resume de objetivos y Actividades	Indicadores	Medios de verificación	Supuestos
Fin: Permitir a los clientes el pago de servicios a través de un portal único y de una manera fácil desde cualquier aplicación móvil.	% de pagos de servicios realizados a través de SIP	Estadísticas de los registros en el sistema SIP	Clientes con aplicación móvil con servicio de pagos ACH Express
Propósito: Implementar una solución de pago de servicios a través de imágenes QR	- % de requerimientos cumplidos - Tiempo de entrega	- Matriz de validación - Cronograma planteado vs ejecución real	Sistema adoptado por las empresas de servicio Sistema implementado con al menos una entidad financiera.
Componentes: Sistema de pago de servicios integrado a las plataformas UniCo y Cliente ACH Express	- Tiempo ejecutado para la integración entre ambas soluciones	- Cronograma planteado vs ejecución real	Sistema uniCo y ACH Express implementados en la entidad financiera

Actividades: Diseño y Análisis Definición Product Backog Planificación de Sprints Desarrollo Pruebas Pruebas de calidad de código	- Nro de requerimientos cumplidos - Nro de requerimientos negociados - Nro de defectos de código encontrados	- Documentación del proyecto.	

DOCUMENTACIÓN