UNIVERSIDAD PÚBLICA DE EL ALTO

CARRERA DE INGENIERÍA DE SISTEMAS

PROYECTO DE GRADO

SISTEMA DE INFORMACIÓN PARA EL SEGUIMIENTO Y CONTROL DE EJECUCIÓN DE PROYECTOS MUNICIPALES CASO: GOBIERNO AUTÓNOMO MUNICIPAL DE CATACORA

Para optar al título de Licenciatura en Ingeniería de Sistemas

MENCIÓN: INFORMÁTICA Y COMUNICACIONES

Postulante: Eddy Heberson Choque Tito
Tutor Metodológico: Ing. Marisol Arguedas Balladares
Tutor Especialista: Ing. Ivan Fernando Mujica Mamani
Tutor Revisor: Ing. Sergio Ramiro Rojas Saire

EL ALTO - BOLIVIA

2020

DEDICATORIA

A Dios, a mi esposa e hijos, a mi padres y hermanos(as), quienes han sido la guía y el camino para poder llegar a este punto de mi carrera, que con sus palabras de aliento nunca bajaron los brazos para que yo tampoco lo haga aun cuando todo se complicaba los amo.

Mi amada Familia

AGRADECIMIENTOS

A Dios por darme salud, y por guiarme día tras día a lo largo de mi vida en alcanzar la culminación del presente proyecto de grado.

A mi esposa Eylin Eulogia Mamani Chuca, por estar siempre a mi lado apoyándome incondicionalmente.

A mis hijos Kheddy Russell y Sharon Alexia, por ser la luz, el orgullo y la fuente de mis alegrías.

A mis papás Abraham Choque Alanoca y Benedicta Tito Yujra, por el apoyo constante y la confianza brindada en la etapa de mi formación.

A mis hermanos(as) Evana, Roxana y Alvaro por el apoyo en esta etapa de mi vida.

Un agradecimiento profundo a mis tutores Ing. Ivan Fernando Mujica Mamani, Ing. Sergio Ramiro Rojas Saire e Ing. Marisol Arguedas Balladares por su apoyo, colaboración, paciencia para concluir el presente trabajo.

No fue fácil el camino para llegar hasta donde estoy, pero gracias a su apoyo, su enorme amabilidad y acompañamiento, lo difícil se hizo más fácil y llevar a feliz término este proyecto se hizo una realidad. Les agradezco, y hago eco de mi enorme aprecio hacia ustedes.

MIL GRACIAS A TODOS
ESTARÉ SIEMPRE AGRADECIDO.

RESUMEN

El desarrollo de este trabajo se estructura a partir del CAPITULO I: Marco Preliminar, en el, se da a conocer los antecedentes de la institución y la problemática existente en el mismo, que llevaron a la necesidad de desarrollar un sistema de información para el seguimiento y control de ejecución de proyectos municipales. Al mismo tiempo, se indican los objetivos, las diferentes justificaciones, entre otros aspectos.

El capítulo II: Marco Teórico, hace referencia a las bases principales para sustentar la investigación, para ello se seleccionaron los aspectos más importantes de la metodología y herramientas que se utilizaran. Los aspectos más relevantes de este acápite son los Sistemas de Información, la metodología de desarrollo SCRUM, con el lenguaje UML, herramientas de desarrollo de software, etc.

En capítulo III, Marco Aplicativo, se da énfasis a todo lo que significa el proceso de desarrollo de software desde los requerimientos del usuario hasta la prueba final del sistema. Cada una de las actividades que son necesarias para la transformación de los requisitos del usuario en un sistema software, son detalladas de acuerdo a la metodología utilizada (SCRUM).

Antes de finalizar el trabajo se pone a prueba el software, se determina la calidad del mismo y se presenta un análisis de costos.

Finalmente, se presentan las conclusiones del proyecto y las recomendaciones para los trabajos futuros; incluyendo además la bibliografía y los diferentes anexos.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO I

1. MARCO PRELIMINAR

1. MARCO PRELIMINAR

1.1. INTRODUCCIÓN

Hoy en día en la actualidad, los municipios cumplen un rol fundamental en todo el proceso de cambio del país, fundamentalmente en la gestión y administración de proyectos, por consiguiente, a medida que crece una institución, las técnicas tradicionales utilizadas en las operaciones administrativas tienden a ser inadecuadas. Es por eso que al pasar de los años se hace inminente la importancia de adecuarse a los cambios tecnológicos sobre el manejo de la información, esto con el objetivo de realizar una mejor toma de decisiones. Para esto es necesario el uso de las tecnologías de la información y las comunicaciones (TIC[1]), que está causando un gran impacto sobre la estructura y el ciclo vital de las instituciones públicas y privadas.

Las TIC hacen viable la administración de información de pequeñas empresas, lo que da lugar a una competencia adicional; por otra parte, dan la oportunidad de que la entrega de productos y la prestación de servicios se realice con nuevo medio corporativo que aumenta la calidad y la rentabilidad. Estas nuevas tecnologías están afectando a casi todos los aspectos de la vida económica y sobre todo a la organización y dirección de las instituciones.

Desde hace algunos años se viene desarrollando el nuevo concepto denominado Gobierno Electrónico (e-gov[2]), que es el medio para modernizar la gestión pública, en busca de mejores prácticas, mayor control y transparencia. En definitiva, una mejor prestación de servicios al gobierno en su conjunto. Este el caso del “Gobierno Autónomo Municipal de Catacora”, quien tiene como propósito contribuir a los ciudadanos de dicho municipio a la oportunidad de acceder y participar de forma más flexible y sin tener que acudir a dependencias del gobierno a toda la información acerca de los proyectos, de forma más puntual y a cualquier hora.

[1] TIC “Tecnologías de Información y Comunicación”.

[2] e-gov “Gobierno Electrónico”.

Este proyecto de grado hace referencia a la aplicación del concepto gobierno electrónico para el Gobierno Autónomo Municipal de Catacora, a través del desarrollo de un Sistema de información que interactúe entre el gobierno y el ciudadano, así mismo facilite toda la información que se genere a cerca del registro de proyectos, tiempo de ejecución de un proyecto, cumplimiento de cronogramas de acuerdo a la planificación de las poas y demás.

1.2. ANTECEDENTES

1.2.1. Antecedentes de la institución

El municipio Catacora, Segunda Sección de la provincia Gral. José Manuel Pando, está situado al Sur del Departamento de La Paz; la capital de la Sección se sitúa a 147 km. de la sede de Gobierno. Los límites territoriales del municipio de Catacora son: al Norte con la hermana República del Perú, al Sur con la hermana República del Perú y la Primera Sección Municipal Santiago de Machaca de la Provincia Gral. José Manuel Pando; al Oeste con la hermana República del Perú y al Este con la Primera Sección Municipal Santiago de Machaca de la Provincia Gral. José Manuel Pando.

El Municipio de Catacora cuenta con una superficie aproximada de 610 Km2, distribuidos en cuatro cantones, esta superficie constituye el 32,7% de la superficie provincial y el 0.06 % del total de la superficie nacional. Este Municipio políticamente cuenta con cuatro capitales de Cantón que a su vez agrupa a 58 Comunidades o estancias reconocidas. En el área urbana de Catacora, capital de Sección y de cantón, se encuentra el poder público municipal.

Actualmente, el Municipio no está dividido por distritos, por lo que no cuenta con subalcaldes; el Municipio se divide en la actualidad en cuatro cantones que son: Catacora, Pairumani Grande, Tolacollo y Parachi y una Sub-Central Pajchiri las cuales están organizadas en sindicatos y centrales agrarias.

1.2.2. Antecedentes de trabajos realizados

- (Ibarra G., 2007)**, "SISTEMA DE INFORMACIÓN PARA EL CONTROL Y SEGUIMIENTO DE PROYECTOS VÍA WEB"**, donde el objetivo es desarrollar un Sistema de Información vía Web para realizar el control y seguimiento de proyectos del Municipio de Mecapaca, que garantice su correcto monitoreo, tomando en cuenta las actividades, recursos humanos y presupuesto en el desarrollo de un proyecto, facilitando información precisa y confiable que coadyuve en la toma de decisiones.

  La metodología aplicada para el desarrollo del Sistema de Información vía Web para realizar el control y seguimiento de proyectos del Municipio de Mecapaca, fue la metodología ágil "ASD" (Desarrollo de Software Adaptable), el cual permitió realizar iteraciones de pruebas veloces en el transcurso de desarrollo del sistema, con las que logro agilizar el trabajo y conseguir la adaptabilidad del sistema en la institución. Este proyecto de grado, fue realizado en La Paz – Bolivia, en la casa superior de Estudios "Universidad Mayor de San Andrés", Carrera Informática, para optar el título de Licenciatura en Informática, mención: Ingeniería de Sistemas Informáticos.

- (Perez Q., 2011)**, "SISTEMA DE SEGUIMIENTO Y CONTROL DE EJECUCIÓN DE OBRAS DE PROYECTOS MUNICIPALES APLICANDO EL MÉTODO BALANCED SCORECARD"**, donde el objetivo es desarrollar un Sistema de Información, aplicando la metodología Balanced Scorecard en los procesos de seguimiento y control de los proyectos que se encuentran en la etapa de ejecución del Gobierno Municipal de Chacarilla, tomando en cuenta la ejecución financiera y ejecución física, dotando a las Autoridades del Gobierno Municipal, instancias técnicas responsables y organizaciones de base de información rápida y veraz para la toma de decisiones.

Para desarrollar el Sistema se utiliza la metodología (RUP), que es el proceso unificado de desarrollo de software, es un método incremental que comprende de cuatro fases: inicio, elaboración, construcción y transición. Este proyecto de grado, fue realizado en La Paz – Bolivia, en la casa superior de Estudios "Universidad Mayor de San Andrés", Carrera Informática, para optar el título de Licenciatura en Informática, mención: Ingeniería de Sistemas Informáticos.

- (Blanco B. & Hernández Z., 2016), **"SISTEMA DE INFORMACIÓN PARA LA GESTIÓN DE PROYECTOS PARA LA FUNDACIÓN UNIVERSITARIA LOS LIBERTADORES",** donde el objetivo es diseñar y desarrollar un sistema de información para la gestión de proyectos, que permita llevar el control de forma sistemática y ordenada, de los diferentes proyectos e investigaciones de grado propuestos por los alumnos de la fundación universitaria los libertadores, en la facultad de ingenierías, para el programa de ingeniería de sistemas.

  La metodología seleccionada para el desarrollo del presente proyecto es SCRUM la cual hace parte de las metodologías ágiles y se tendrá en cuenta la arquitectura de N capas, con cual se tendrá en cuenta el modelamiento, el diseño y la programación. Este proyecto de grado, fue realizado en Bogotá – Ecuador, en la casa superior de Estudios "Fundación Universitaria los Libertadores", Carrera Ingeniería de Sistemas, para optar el título de Ingeniero de Sistemas.

- (Chávez G., 2010), **"SISTEMA DE INFORMACIÓN PARA EL CONTROL, SEGUIMIENTO Y MANTENIMIENTO DEL EQUIPAMIENTO HOSPITALARIO"**, donde el objetivo es analizar, diseñar, desarrollar e implementar un Sistema de Información para el Control, Seguimiento y Mantenimiento del Equipamiento Hospitalario en el Hospital Central de la Fuerza Aérea del Perú.

  El presente proyecto de tesis usa la metodología RUP (Rational Unified Process), que está dirigido por casos de uso, ello facilita las tareas de diseño y

programación, además que reduce riesgos ya que su ciclo de vida es iterativo. Este proyecto de tesis, fue realizado en Lima – Perú, en la casa superior de Estudios “Universidad Ricardo Palma”, Facultad de ingeniería, para optar el título profesional de Ingeniero Informático.

1.3. PLANTEAMIENTO DEL PROBLEMA

El Gobierno Autónomo Municipal de Catacora, administra una gran cantidad de datos acerca de los proyectos que se manejan, y solo se trabaja de forma tradicional, el cual no hace posible tener información disponible de forma ágil y rápida, puesto que no existe un mecanismo o proceso automático que se encargue de esta labor.

En la actualidad, el procesamiento de los datos es realizado por programas de Ofimática como: Procesador de texto, hojas de cálculo, archivos físicos (folders), y no se cuenta un visualizador de información que interactúen entre sí para obtener la información requerida. Este esquema de procesamiento no favorece la visualización de reportes por lo que es necesario un sistema informático que permita la automatización de la información del Gobierno Autónomo Municipal de Catacora.

En síntesis, el Gobierno Autónomo Municipal de Catacora tiene deficiencias en el registro de proyectos y en el control de avance de ejecución de los mismos, no cuenta con información actualizada acerca de las empresas que ejecutan un determinado proyecto, no cuenta con información a la mano de las comunidades beneficiadas con la ejecución de las obras y las poas que le corresponde. Y al no existir un sistema informático que se encargue de gestionar toda esta información, se vuelve desordenada causando que no se cuente con información disponible.

Por todo lo expuesto en el planteamiento general, surge la siguiente interrogante suscrita en el problema principal.

1.3.1. Problema Principal

¿Cómo mejorar el manejo de la información para el seguimiento y control de ejecución de proyectos en el Gobierno Autónomo Municipal de Catacora a través del desarrollo del Sistema de Información altamente disponible?

1.3.2. Problemas Secundarios

- Existe excesiva documentación física en el área de proyectos de la institución causando cuellos de botella en el acceso a la información de los proyectos concluidos e inconclusos.
- No se tiene un catálogo de todas las empresas que están relacionadas con el proyecto lo que genera desconocimiento de los mismos.
- No se cuenta con información tangible acerca de las comunidades beneficiadas con los proyectos ocasionando que se obtenga información imprecisa.
- No se cuenta con una adecuada gestión de poas de los diferentes sectores donde la institución presta servicio ocasionando retraso en la planificación de proyectos en una determinada gestión.
- No se cuenta con un cronograma actualizado en línea el cual dificulta el seguimiento eficiente a las obras.

1.4. OBJETIVOS

1.4.1. Objetivo General

Desarrollar un Sistema de Información para realizar el Seguimiento y Control de Ejecución de Proyectos del Gobierno Autónomo Municipal de Catacora, que garantice un correcto monitoreo a la gestión de proyectos, facilitando información actualizada, precisa y confiable que coadyuve a la toma decisiones.

1.4.2. Objetivos Específicos

- Digitalizar documentos físicos que ingresan a la institución concerniente a los proyectos que están en ejecución.

- Desarrollar un módulo para gestión de todas las empresas relacionadas con los proyectos, evitando desconocimiento de los mismos.
- Obtener información de todas las comunidades pertenecientes al municipio, y registrar en la base de datos para tener datos precisos de las unidades beneficiadas con los proyectos.
- Elaborar un módulo para la gestión de poas de los diferentes sectores concernientes a los proyectos para realizar una buena planificación en una determinada gestión.
- Desarrollar un módulo para la gestión de cronogramas, que coadyuve a realizar un mejor seguimiento y control de las obras.
- Implementar un módulo de reportes para así tener facilidades en la elaboración del mismo y evitar la pérdida de tiempo e información de los proyectos.

1.5. JUSTIFICACIÓN

1.5.1. Justificación Técnica

El municipio, donde se implementará el Sistema de Información para el Seguimiento y Control de Ejecución de Proyectos Municipales, cuenta con los requerimientos de software y hardware necesarios para su instalación.

Por consiguiente, el proyecto se justifica técnicamente, además por el uso de alta tecnología de comunicación: Una red de computadoras que manejan los métodos de acceso, manipulación de datos. El sistema será de fácil uso para los diferentes usuarios, permitiendo una eficiente administración de los proyectos.

1.5.2. Justificación Económica

El sistema a desarrollarse incrementará el grado de eficiencia y el desempeño de las funciones que cumplen los encargados del seguimiento y control de los proyectos, logrando optimizar el uso de los recursos humanos y tiempo.

Por otro lado, la LEY Nº 164, Ley General de Telecomunicaciones, tecnologías de Información y Comunicación, en su artículo 77 menciona en sus parágrafos “I. Los Órganos Ejecutivo, Legislativo, Judicial y Electoral en todos sus niveles, promoverán y priorizarán la utilización del software libre y estándares abiertos, en el marco de la soberanía y seguridad nacional” y “II. El Órgano Ejecutivo del nivel central del Estado, elaborará el plan de implementación de software libre y estándares abiertos en coordinación con los demás órganos del Estado y entidades de la administración pública.”

En ese sentido el proyecto es factible económicamente para el Gobierno Autónomo Municipal de Catacora, ya que el desarrollo del Sistema de Información estará basado en plataformas de programación Open Source (Código Abierto), con licencia GPL (Licencia Pública General) y no tendrá ningún costo en desarrollo por el software que se utilizará.

1.5.3. Justificación Social

La necesidad de un control social sobre los proyectos que se ejecutan en el Gobierno Autónomo Municipal de Catacora es evidente, debido al alto compromiso que se tienen con los vecinos de las diferentes estancias, es en este sentido que el Sistema de Información a desarrollar beneficiara tanto a la institución como también a la población en general.

Con la implementación del Sistema de Información los usuarios finales; el Alcalde del municipio, responsables de los proyectos, realizarán de manera más fácil, adecuada, ordenada y responsable el seguimiento y control de los diferentes proyectos.

1.6. METODOLOGÍA

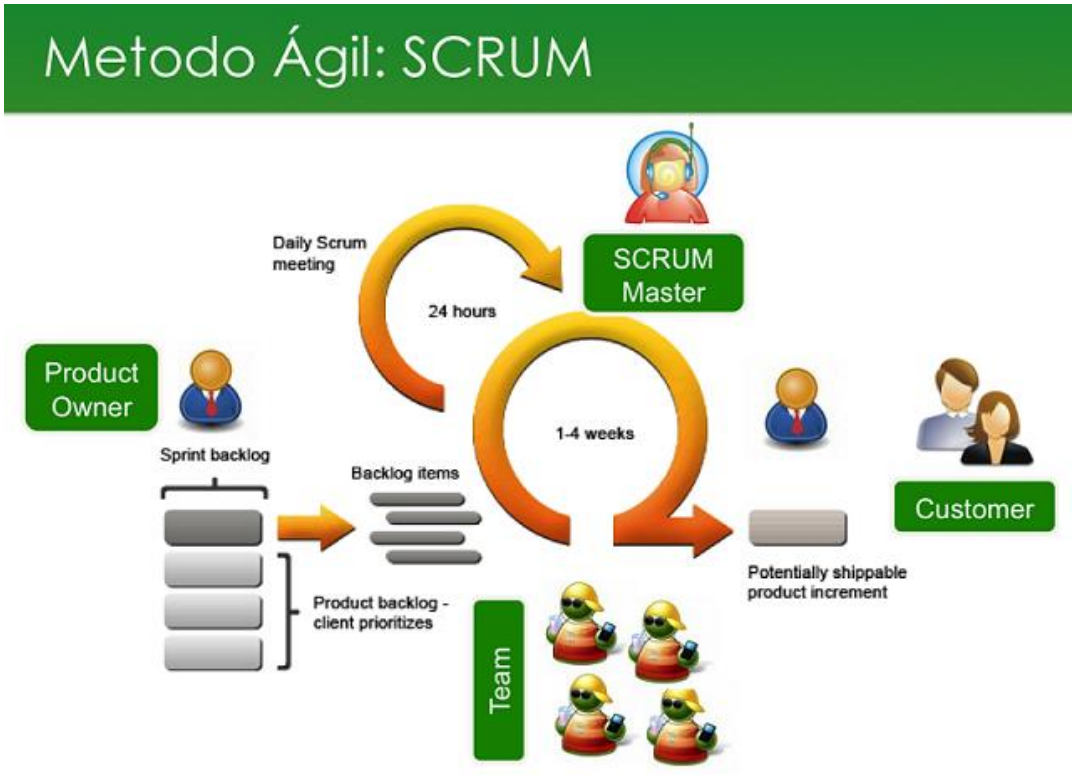
La metodología ágil, que se usará para el desarrollo del presente proyecto será **SCRUM,** que es un proceso en el que se aplican de manera regular un conjunto

de buenas prácticas para trabajar colaborativamente, en equipo, y obtener el mejor resultado posible de un proyecto.

1.6.1. Metodología SCRUM

SCRUM permite abordar proyectos complejos desarrollados en entornos dinámicos y cambiantes de un modo flexible. Está basado en entregas parciales y regulares del producto final en base al valor que ofrecen a los clientes. Es una opción de gestión ideal para acometer proyectos desarrollados en entornos complejos que exigen rapidez en los resultados y en los que la flexibilidad es un requisito imprescindible, Scrum ofrece agilidad y el valor agregado es el resultado de su implementación. Los procesos de la metodología Scrum se muestran a continuación. (Gerrero, 2016).

Figura 1: Proceso de la metodología Scrum



Fuente: (Gerrero, 2016)

1.7. HERRAMIENTAS

Para realizar el Sistema de Información se utilizará la siguiente plataforma de desarrollo:

- **Linux**, como sistema operativo, para desarrollar y utilizar como servidor del software.
- **JavaScript**, (abreviado comúnmente JS) es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico. Todos los navegadores modernos interpretan el código JavaScript integrado en las páginas web. (ICTEA, 2020).
- **PostgreSQL,** que es un sistema de gestión de bases de datos relacional orientado a objetos y de código abierto, publicado bajo la licencia PostgreSQL, para almacenar todos los datos. (POSTGRESQL, 2020).
- **Nodejs,** es una forma de ejecutar JavaScript en el servidor, además de mucho más. **Node**.**js** es un entorno Javascript del lado del servidor, basado en eventos. **Node** ejecuta javascript utilizando el motor V8, desarrollado por Google para uso de su navegador Chrome. (Morales A. , 2012).
- **VueJs**, es un framework para javascript, es decir, es un conjunto de herramientas y funciones que permiten desarrollar páginas web de una manera más cómoda. Vue nace con la necesidad de no tener que escribir tanto código javascript y sobre todo con la idea de ahorrar tiempo al programador. (By Equipo Geek, 2020).
- **ExpressJs**, es un framework para Node.js que sirve para ayudarnos a crear aplicaciones web en menos tiempo ya que nos proporciona funcionalidades como el enrutamiento, opciones para gestionar sesiones y cookies. (By Equipo Geek, 2020).

Y como herramientas de desarrollo se utilizará los siguientes:

- **Visual Studio Code**, que es un editor de código fuente desarrollado por Microsoft para Windows, Linux y macOS. Incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código. También es personalizable, por lo que los usuarios pueden cambiar el tema del editor, los atajos de teclado y las preferencias. Es gratuito y de código abierto. (EcuRed contributors, 2020).
- **PgAdmin**, es una herramienta de código abierto para la administración de bases de datos PostgreSQL y derivados (EnterpriseDB Postgres Plus Advanced Server y Greenplum Database). PgAdmin se diseña para responder a las necesidades de la mayoría de los usuarios, desde escribir simples consultas SQL hasta desarrollar bases de datos complejas. (EcuRed contributors, 2020).
- **UML**, (Lenguaje de Modelado Unificado) es “un lenguaje estándar para escribir diseños de software. El UML puede usarse para visualizar, especificar, construir y documentar los artefactos de un sistema de software intensivo”. (Pressman, 2010).

1.8. LÍMITES Y ALCANCES

1.8.1. Límites

El sistema está específicamente orientado a todos los procesos concernientes para el seguimiento y control de proyectos del Gobierno Autónomo Municipal de Catacora bajo los siguientes módulos:

- Módulo de Administración del Sistema.
- Módulo de Gestión de Proyectos.
- Módulo para Catálogo de Empresas.
- Módulo de Gestión de Poas.
- Módulo de Reportes.

El Sistema a desarrollarse no contemplará otros módulos como ser para la administración financiera o administración de inventarios.

1.8.2. Alcances

En cuanto al alcance del proyecto, será el diseño y la implementación de un sistema de seguimiento y control físico de ejecución de proyectos, brindando información actualizada de cada proyecto, emisión de reportes y la capacitación del personal del municipio en el manejo del sistema automatizado.

1.9. APORTES

El principal aporte a la institución es el Sistema de Información para el Seguimiento y Control de Ejecución de Proyectos Municipales, que apoyara al Gobierno Autónomo Municipal de Catacora, automatizando los procesos y reportes, manejando la información de manera oportuna y segura, esto con el fin de mejorar el tiempo de respuesta y servicio a la población de municipio en general.

Se entregará la documentación del proyecto con el manual de usuario al Gobierno Autónomo Municipal de Catacora, para las actualizaciones o desarrollo de nuevos proyectos.

CAPÍTULO II

2. MARCO TEÓRICO

2. MARCO TEÓRICO

2.1. INTRODUCCIÓN

En todo proceso de elaboración e investigación, un elemento que direcciona el camino a seguir en todo trabajo científico es el marco teórico, en este capítulo describiremos 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 utilizara más al contrario se trata de presentar una base para la fácil comprensión de la misma.

2.2. MARCO INSTITUCIONAL

El Gobierno Autónomo Municipal de Catacora es la segunda sección municipal de la provincia José Manuel Pando del departamento de La Paz. Al norte y oeste limita con la República del Perú, y al noreste, este y sur con el municipio Santiago de Machaca. Tiene cuatro cantones: Catacora, Thola Khollu, Pojo Pajchiri y Pairumani Grande. Su acceso vial es a través de la ruta La Paz - San Andrés de Machaca - Catacora, transitable casi todo el año.

La región presenta una topografía ondulada en extensas planicies. Su clima es frío y seco, con una temperatura promedio de 7°C, y una precipitación pluvial de 320 mm. Su principal río es el Cussicussini (permanente). El origen de la población es aymara, siendo este junto al castellano los principales idiomas.

Desde algún tiempo el gobierno municipal en se encuentra abocado a proyectos de vinculación caminera, trabajos efectuados en forma conjunta con la Prefectura del Departamento, además del mejoramiento de la infraestructura productiva en el rubro agrícola y el desarrollo pecuario, en especial de camélidos de las razas kara, tacupullo, en alpacas suri y huacaya.

Figura 2: Mapa de Ubicación

Fuente: (Contributors, 2019)

2.2.1. Misión

La Misión Institucional es la razón de ser del municipio expresados en los objetivos que determinaron su creación, por tanto, la misión del Gobierno Autónomo Municipal de Catacora, es apoyar al desarrollo de las actividades productivas agropecuarias y artesanales para hacer de este un municipio productivo y eficiente en el manejo de sus recursos.

La misión más primordial de la Gobierno Autónomo Municipal de Catacora es el bienestar de la población en la región donde se desarrolla, los habitantes de esta región han buscado el vivir bien basado en la protección del medio ambiente, como municipio la entidad, ha desarrollado políticas basados en la Constitución Política del Estado, el trabajo específico de la Municipio es el desarrollo basado en el bienestar de la población urbana que habita en esta región especifica.

Promover el desarrollo ocupacional, higiene, producción, y ambiental, por política estratégica para el desarrollo de la región de Catacora. Así mismo la generación de proyectos sociales, la solución a las problemáticas sociales y económicas de la coyuntura y la reproducción y difusión de la cultura nacional.

2.2.2. Visión

La visión estratégica de desarrollo representa la imagen del municipio al que se pretende alcanzar en los próximos años, esta visión involucra a los diferentes actores del proceso de planificación. La Visión Institucional del municipio está orientada a ser un municipio productivo y eficiente en el uso de sus recursos.

Por tanto, en lo económico, el municipio de Catacora pretende ser un municipio productivo y competitivo dedicado a la explotación de ganado camélido y ovino, esta visión se complementa con la implantación de infraestructura productiva que apoya a la producción ganadera con capacitación de productores pecuarios y agrícolas de cada organización económica.

El gobierno autónomo Municipal de Catacora, tiene como visión el desarrollo de la sociedad de Catacora, en sus diferentes aspectos: Cultural, Productivo Ambiental, Educacional, Agropecuario y la Salud en las diferentes etapas mencionamos algunas por su importancia.

En lo Productivo el municipio tiene muchas fortalezas a nivel de producción por ser una región productora de hortalizas y crianza de animales en diferentes grupos.

En lo Educativo, el municipio está en proceso de búsqueda de políticas específicas que mejoren la preparación de los jóvenes, para diferentes tareas del sector ocupacional y laboral.

2.2.3. Principios y Objetivos del Gobierno Autónomo Municipal de Catacora

El objetivo principal del Gobierno Municipal es el de contribuir a la satisfacción de las necesidades de la población se centran básicamente en las siguientes materias:

- ✓ Desarrollo Humano
- ✓ Infraestructura
- ✓ Administrativa Financiera
- ✓ Defensa del Consumidor
- ✓ Servicios.

Las competencias antes señaladas deben ser ejercidas en función de las políticas nacionales como departamentales, expresadas en el Plan de Desarrollo Municipal, para ello la Gestión Municipal se constituye en el medio que va a permitir alcanzar dichas finalidades.

2.2.4. Interacción con la Sociedad

1º. El Municipio contribuye el ente más factible para transmitir a la sociedad civil la defensa, protección y fomento de los bienes culturales y patrimoniales en el país y de la región, así como a la conservación y uso racional de los recursos naturales y humanos dentro de una perspectiva de integración nacional, así como también a la defensa de los derechos y libertades fundamentales de nuestro pueblo.

2º. El Municipio se integra a la sociedad a través de planes y programas de carácter educativo, cultural, productivo y la salud y se compromete con el desarrollo socio-económico del país, al mejoramiento de las condiciones de vida de sus habitantes; y contribuir al desarrollo regional con sentido de integración nacional.

3º. El Municipio contribuye mediante la economía comunitaria a fortalecer la relación con las áreas urbanas de la ciudad y asuman la necesidad histórica de una transformación revolucionaria de las estructuras sociales y económicas que configuran una realidad de dependencia que impide el desarrollo autodeterminado de nuestro pueblo.

4º. El Municipio debe orientar la actividad productiva, artesanal y cultural en relación con las necesidades de la población y dentro de la perspectiva de integrar el movimiento comunitario con las políticas del vivir bien.

2.2.5. Estructura Organizacional

El Gobierno Autónomo Municipal de Catacora está conformado por un Concejo Municipal, que es la máxima autoridad del Gobierno Municipal y cuya naturaleza y atribuciones se describen con detalle en la Ley de Municipalidades Artículo 12°. Actualmente esta es la estructura del Concejo Municipal del Gobierno Autónomo Municipal de Catacora:

Concejo Municipal:

Presidente del Concejo Municipal	: Sr. Fidel Cantuta Mamani
Secretario del Concejo Municipal	: Sra. Esperanza Poma Usnayo
Concejal Vocal	: Sr. Rolando Vasquez Mamani

El Ejecutivo Municipal o Máxima Autoridad ejecutiva del Gobierno Municipal es el Alcalde Municipal, las atribuciones de este son: planificar, organizar, administrar, dirigir y supervisar las labores del Órgano Ejecutivo y las descritas en el artículo 44° de la Ley Nº 2028 de Municipalidades.

En la Actualidad la estructura del Órgano Ejecutivo (Incluyendo a las Oficialías Mayores por área del Gobierno Autónomo Municipal de Catacora es la siguiente:

Órgano Ejecutivo Municipal:

Alcalde Municipal	: Sr. Absalon Conurana Surco
Oficial Mayor Administrativo	: Sr. Lucio Huanca Huanca
Oficial Mayor Financiero	: Sr. José Ururi Castro

2.3. SISTEMA DE INFORMACIÓN

2.3.1. Definiciones

"Un sistema de información, es un sistema hombre/máquina integrado que provee información para el apoyo de las funciones de operación, gerencia y toma de decisiones en una organización". (Montilva, 1999).

Por otra parte, “podemos plantear la definición técnica de un sistema de información como un conjunto de componentes interrelacionados que recolectan (o recuperan), procesan, almacenan y distribuyen información para apoyar los procesos de toma de decisiones y de control en una organización. Además de apoyar la toma de decisiones, la coordinación y el control, los sistemas de información también pueden ayudar a los gerentes y trabajadores del conocimiento a analizar problemas, visualizar temas complejos y crear nuevos productos”. (Laudon & Laudon, 2012).

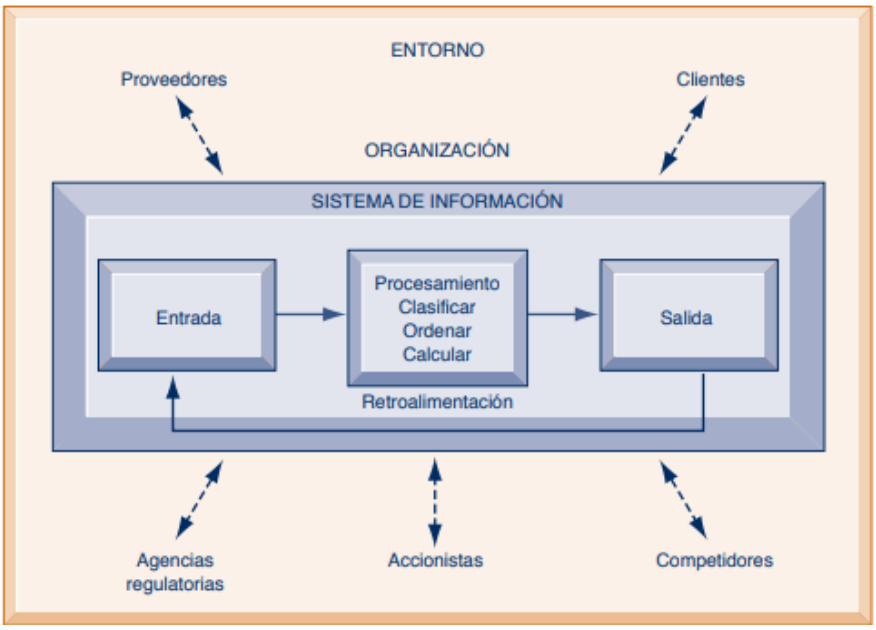
“Un sistema de información es un conjunto de componentes que interactúan entre sí, buscando un objetivo en común. En nuestro caso: satisfacer las necesidades de Gobierno Autónomo de Catacora en la Unidad de Proyectos” (Elaboración propia).

2.3.2. Funciones de un Sistema de Información

Hay tres actividades en un sistema de información que producen los datos necesarios para que las organizaciones tomen decisiones, controlen las operaciones, analicen problemas y creen nuevos productos o servicios. Estas

actividades son: entrada, procesamiento y salida (vea la figura 4). La entrada captura o recolecta los datos en crudo desde el interior de la organización o a través de su entorno externo. El procesamiento convierte esta entrada en bruto en un formato significativo. La salida transfiere la información procesada a las personas que harán uso de ella, o a las actividades para las que se utilizará. Los sistemas de información también requieren retroalimentación: la salida que se devuelve a los miembros apropiados de la organización para ayudarles a evaluar o corregir la etapa de entrada. (Laudon & Laudon, 2012).

Figura 3: Funciones de un Sistema de Información



Fuente: (Laudon & Laudon, 2012)

2.3.3. Componentes de un Sistema de Información

Para estudiar los componentes de un Sistema de Información es necesario es necesario clasificarlos previamente, de acuerdo a su naturaleza, en dos tipos: a) componentes físicos, representados por las entidades que forman el sistema de información, y b) componentes funcionales, que agrupan una o más entidades en torno a una función básica del sistema. (Montilva, 1999, pág. 35).

a) Componentes Físicos: Un sistema de información se puede dividir en varios subsistemas físicos en la forma que se muestra a continuación:

El subsistema del computador está formado por el equipo de computación (procesador, unidades de entrada y de salida de datos), y para los programas de apoyo para ese equipo (sistema operativo, sistema de manejo de base de datos, editores, compiladores, utilitarios entre otros).

El subsistema de personal lo constituyen los usuarios del sistema, el administrador de base de datos, los operadores, el personal de entrada de datos, ingenieros de sistema, analistas y programadores). El número de requisitos de los miembros de este subsistema depende del tamaño y complejidad del sistema de información.

El subsistema programado consiste en los programas de aplicación que ejecuta el procesamiento en el computador y los procedimientos para realizar las aplicaciones operativas, esto es, formas, manuales de operación, usuario, sistema, instrucciones para la entrada de datos, formularios, plantillas, entre otros. Las características de este subsistema dependen del propósito y tipo de sistema de información.

El subsistema de datos está constituido por los elementos de almacenamiento de datos. Estos elementos pueden ser de dos tipos: 1) archivos convencionales o, 2) base de datos. El tipo de elemento que se utilice determina tanto los programas de aplicación como los programas de apoyo del equipo. Tal es el caso de las bases de datos que requieren el uso de los sistemas de Manejo de Base de Datos (SMBD[3]).

b) Componentes Funcionales: Considerando las dos funciones básicas de un sistema de información: Procesamiento de transacciones e información y

[3] SMBD “Sistema de Manejo de Base de Datos”.

adicionando la función implícita de almacenamiento de datos, según (Senn, 1990), divide un sistema de información en tres componentes funcionales. Estos son:

Subsistema de procesamiento de transacciones. Su propósito es capturar, clasificar, ordenar, calcular y resumir los datos originados por las transacciones relevantes que acontecen en la organización.

Subsistema de administración de datos. Los datos originados por las transacciones deben almacenarse en medios apropiados tales como base de datos o archivos. Se requiere entonces, de un subsistema encargado del mantenimiento y actualización de esos medios de almacenamiento de forma, tal que facilite el almacenamiento y transformación de datos de información.

Subsistema de procesamiento de información. Se encarga de producir y distribuir la información requerida por los usuarios del sistema. Este subsistema toma los datos de la base de datos o archivo, los procesa de acuerdo a patrones predefinidos (algoritmos, modelos, entre otros) y los distribuye presentando la información en forma de reporte (informes, gráficos, entre otros).

2.3.4. Clasificación de un Sistema de Información

Los diferentes tipos de sistemas de información que existen hoy en día, están destinados a procesar datos por razones diferentes, los autores lo definen de la siguiente manera:

Según (Senn, 1990), los sistemas de información se clasifican en:

a) Sistema de Procesamiento de Transacciones

Procesa datos referentes a las transacciones o actividades de las empresas. Las cinco razones para el procesamiento de las transacciones son:

- La clasificación implica agrupar datos según características comunes.
- Las operaciones de cálculo consisten en efectuar alguna operación sobre los datos para generar resultados útiles.
- La distribución u ordenación de datos, consiste en disponerlos según una consecuencia o sucesión para facilitar el procesamiento y tomar los menos engorrosos.
- La síntesis o resumen, reduce gran cantidad de datos de transacciones a una forma más breve y concisa.
- El almacenamiento, consiste en mantener los registros de los sucesos o los eventos que afectan sus operaciones.

b) Sistemas de Información Gerencial

Llamados también sistemas de reportes de gerencia, se enfocan al apoyo de la toma de decisiones cuando los requerimientos de la información pueden ser identificados de antemano. Las decisiones respaldadas por este sistema frecuentemente se repiten.

c) Sistema de Apoyo para la Decisión

Ayuda a los gerentes en la toma de decisiones únicas y no reiteradas que relativamente no están estructuradas. Parte del proceso de la decisión consiste en determinar los factores a considerar cuál es la información necesaria.

Existen a fin de responder a condiciones inesperadas y propias de la información. Estos sistemas son particularmente necesarios para los altos niveles de dirección que deben tratar constantemente problemas cambiantes y tomar decisiones en caso de que surjan imprevistos.

Por otra parte (Kendall & Kendall, 2011), los clasifica de la siguiente manera:

a) Sistemas de Procesamiento de Transacciones

Son sistemas de información computarizados desarrollados para procesar gran cantidad de datos para transacciones rutinarias de los negocios, tales como nómina o inventarios. Eliminan el tedio de las transacciones operacionales necesarias y reduce el tiempo que alguna vez se requirió para ejecutarlas manualmente, aun cuando los usuarios todavía deben ingresar los datos a los sistemas computarizados.

b) Sistemas de Automatización de Oficina y Sistemas de Manejo de Conocimiento

A nivel conocimiento de la organización hay dos clases de sistemas. Los sistemas de automatización de oficina brindan soporte a los trabajadores de datos, quienes no crean un nuevo conocimiento, sino que usan la información para analizarla, transformarla, manipularla, compartirla o diseminarse por la organización y algunas veces más allá de ellas.

Los sistemas de manejo de conocimiento brindan soporte a los trabajadores profesionales y ayudan a crear nuevos conocimientos que contribuyen a la organización.

c) Sistemas de Información Gerencial

Son sistemas de información computarizados que trabajan debido a la interacción existente entre usuarios y computadoras, requieren que los usuarios, hardware y software trabajen de forma conjunta. Por otra parte, brindan soporte al análisis y toma de decisiones.

d) Sistemas Expertos e Inteligencia Artificial

La inteligencia artificial se puede considerar la meta de los sistemas expertos. El fundamento principal de ellos ha sido desarrollar máquinas que se comporten de forma inteligente. Su investigación se enfoca en la comprensión del lenguaje natural y el análisis de la habilidad para razonar un problema y llegar a conclusiones lógicas.

Los sistemas expertos utilizan los enfoques de razonamiento de los sistemas de inteligencia artificial para resolver los problemas que los plantean, capturan en forma efectiva y usan el conocimiento de un experto para resolver un problema particular experimentado en una organización. De igual forma selecciona la mejor solución a un problema. Los componentes básicos de un sistema experto son la base de conocimiento y una máquina que conecte al usuario con el sistema.

2.3.5. Ciclo de Vida de un Sistema de Información

Según (Senn, 1990), el método de ciclo de vida para el desarrollo de sistemas es el conjunto de actividades que los analistas, diseñadores y usuarios realizan para desarrollar e implementar un sistema de información.

a) Determinar Requerimientos del Sistema

El aspecto fundamental del análisis de sistemas es comprender todas las facetas importantes de la parte de la empresa que se encuentra bajo estudio. Los analistas, al trabajar con los empleados y administradores, deben estudiar los procesos de una empresa para dar respuesta a las siguientes preguntas clave:

- ¿Qué es lo que hace?

- ¿Cómo se hace?
- ¿Con qué frecuencia se presenta?
- ¿Qué tan grande es el volumen de transacciones o decisiones?
- ¿Cuál es el grado de eficiencia con el que se efectúan las tareas?
- ¿Existe algún problema?
- Si existe algún problema, ¿Cuál es la causa que lo origina?

b) Diseño del Sistema

El diseño de un sistema de información produce los detalles que establecen la forma en la que el sistema cumplirá con los requerimientos identificados durante la fase de análisis. Los especialistas en sistemas se refieren, con frecuencia a esta etapa como diseño lógico en contraste con la del desarrollo de software, a la que denominan diseño físico.

c) Desarrollo de Software

Los encargados de desarrollar el software pueden instalar software comprobando a terceros o escribir programas diseñados a la medida del solicitante. La elección depende del costo de cada alternativa, del tiempo disponible para escribir el código, y de la disponibilidad de los programadores. Por lo general, los programadores que trabajan en las grandes organizaciones pertenecen a un grupo permanente de profesionales.

d) Prueba de Sistemas

Durante la prueba de sistemas, el sistema se emplea de manera experimental para asegurarse de que el software no tenga fallas, es decir, que funcione de acuerdo a las especificaciones y en la forma en que los usuarios esperan que los haga. Se alimentan como entradas, a un conjunto de datos de prueba para su procesamiento y después se examina los resultados.

e) Implantación y Evaluación

La implantación es proceso de verificar e instalar nuevo equipo, entrenar a los usuarios, instalar la aplicación y construir todos los archivos de datos necesarios para utilizarlas.

Una vez instaladas, las aplicaciones se emplean durante muchos años. Sin embargo, las organizaciones y los usuarios cambian con el paso del tiempo, incluso el ambiente es diferente con el paso de las semanas y los meses. Por consiguiente, es indudable que debe darse mantenimiento a las aplicaciones.

La evaluación de un sistema se lleva a cabo para identificar puntos débiles y fuertes. La evaluación ocurre a lo largo de cualquiera de las siguientes dimensiones:

- Evaluación operacional: Valoración de la forma en que funciona el sistema, incluyendo su facilidad de uso, tiempo de respuesta, lo adecuado de los formatos de información, confiabilidad global y nivel de utilización.
- Impacto organizacional: Identificación y medición de los beneficios para la organización en áreas como la fianza, eficiencia operacional e impacto competitivo. También se incluye el impacto sobre el flujo de información externo e interno.
- Opinión de los administradores: Evaluación de las actividades de los directivos y administradores dentro de la organización, así como de los usuarios finales.
- Desempeño del desarrollo: La evaluación del proceso de desarrollo de acuerdo al criterio tales como el tiempo y esfuerzo de desarrollo, concuerdan con presupuesto y estándares, y otros criterios de administración de proyectos. También se incluye la valoración de los métodos y herramientas utilizados durante el desarrollo.

2.4. GESTIÓN DE PROYECTOS

Todo proyecto nace de una necesidad imperiosa. Se orienta pues, a la consecución de un resultado dentro de un plazo de tiempo limitado, con un principio y un final que determinan el alcance y los recursos. Para ello se estructura en función de actividades, que discurren de forma secuencial o paralela en los distintos tipos de proyectos.

2.4.1. Definición de la Gestión de Proyectos

La gestión de proyectos también conocida como gerencia o administración de proyectos es la disciplina que guía e integra los procesos de planificar, captar, dinamizar, organizar talentos y administrar recursos, con el fin de culminar todo el trabajo requerido para desarrollar un proyecto y cumplir con el alcance, dentro de límites de tiempo, y costo definidos: sin estrés y con buen clima interpersonal. Todo lo cual requiere liderar los talentos, evaluar y regular continuamente las acciones necesarias y suficientes. (Wallace, 2014).

2.4.2. Definición de Proyecto

Es la búsqueda de una solución inteligente al planteamiento de un problema pendiente a resolver, entre muchas una necesidad humana. (Sapag & Sapag, 2003).

2.4.3. Tipos de Proyecto

Existen proyectos de todo tipo y enumerarlos todos es casi imposible. El área, el presupuesto, la localización, la finalidad, el objetivo, los medios, entre otras, son muchas las variables que pueden servir para determinar en qué categoría se engloba un proyecto: grande o pequeño, público o privado, de nueva creación o de mejora, de construcción o de montaje, nacional o internacional. Sin embargo, a la hora de, por ejemplo, seleccionar las herramientas de gestión más adecuadas hay que saber discernir entre proyectos simples y complejos. Está claro lo que es un proyecto sencillo: aquél exento de dificultad aparente, de corta duración, objetivos

alcanzables, recursos suficientes y niveles de incertidumbre y riesgo muy reducidos.

Existen muchos tipos de proyectos, pero los más comunes según (Wallace, 2014) son:

a) Según la Procedencia o Capital

- ✓ Proyectos públicos.
- ✓ Proyectos privados.
- ✓ Proyectos mixtos.

b) Según el Ámbito

- ✓ Proyectos de ingeniería.
- ✓ Proyectos económicos.
- ✓ Proyectos fiscales.
- ✓ Proyectos legales.
- ✓ Proyectos médicos.
- ✓ Proyectos matemáticos.
- ✓ Proyectos artísticos.
- ✓ Proyectos literarios.
- ✓ Proyectos tecnológicos.
- ✓ Proyectos informáticos.

c) Según su Orientación

- ✓ Proyectos productivos.
- ✓ Proyectos educativos.
- ✓ Proyectos sociales.
- ✓ Proyectos comunitarios:
- ✓ Proyectos de investigación.

2.4.4. Características de los Proyectos

Según (Wallace, 2014), todos los tipos de proyectos tienen en común una serie de características:

- Cuentan con un propósito.
- Se resumen en objetivos y metas.
- Se han de ajustar a un plazo de tiempo limitado.
- Cuentan con, al menos, una fase de planificación, una de ejecución y una de entrega.
- Se orientan a la consecución de un resultado.
- Involucran a personas, que actúan en base a distintos roles y responsabilidades.
- Se ven afectados por la incertidumbre.
- Han de sujetarse a un seguimiento y monitorización para garantizar que el resultado es el esperado.
- Cada uno es diferente, incluso los de similares características.

2.4.5. Gestión del Ciclo de Proyectos

Figura 4: Gestión del Ciclo de Proyectos

Fuente: (Comisión Europea, 2001)

La gestión del ciclo de proyectos es un nuevo método para la planificación de proyectos, este cuenta con varias etapas, como se muestra en la anterior figura.

La definición de las seis fases del ciclo del proyecto según la (Comisión Europea, 2001) son las siguientes:

Programación: Es donde se programa las estrategias de cómo se va a enfrentar el proyecto y dar soluciones a los problemas que se puedan presentar, definiendo así algunas políticas de elaboración y coordinación, designando tareas del proyecto a cada encargado.

Identificación: Es la primera aproximación al proyecto, que incluye el análisis de la participación de los problemas, de los objetivos y de las alternativas, así como la elaboración consensuada entre todas las partes afectadas, de una lógica de intervención expresadas globalmente en objetivos, resultados y actividades.

Formulación: Determina todos los aspectos detallados de un proyecto, que además de la información recogida en la identificación, debe incluir al menos, los indicadores del logro de los objetivos y resultados, las fuentes de verificación de esos indicadores, los factores externos al proyecto que le puedan afectar, los estudios de viabilidad económica, social, medio ambiental y demás, los cronogramas y los presupuestos.

Financiación: La financiación se refiere al monto económico que es destinado a un proyecto cuando este ya está formulado y aprobado, en el cual se tiene una fiscalización de todos los recursos económicos asignados al proyecto desde su inicio hasta su culminación.

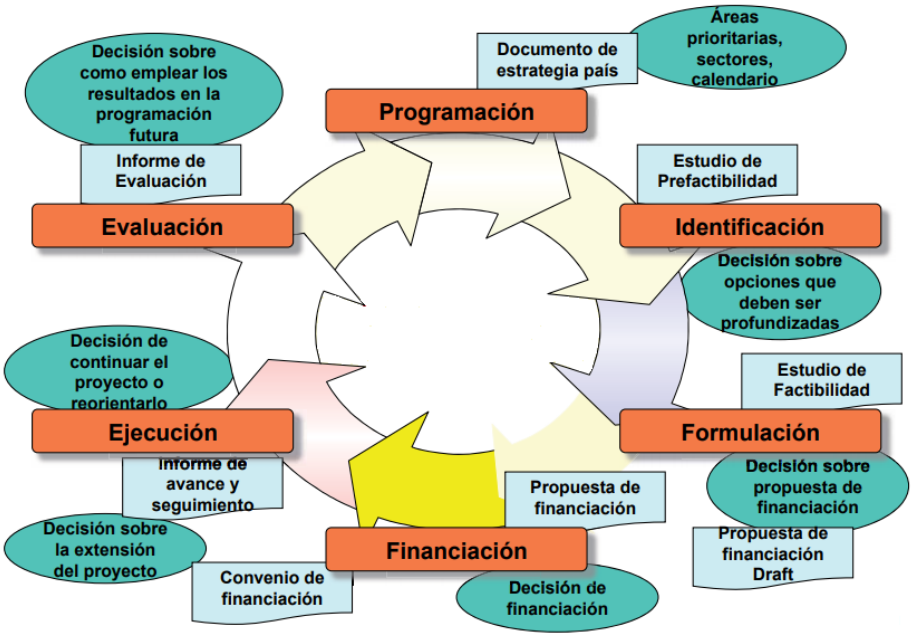
Ejecución – Seguimiento: La ejecución es la realización del proyecto con el fin de alcanzar paulatinamente los resultados específicos en el documento de formulación y con ello el objetivo esperado. Paralelamente a la ejecución se lleva a cabo el seguimiento, que es el estudio y la valoración del proyecto que compara el trabajo realizado frente al planificado, y en el caso de que haya diferencias

importantes aplica medidas correctivas, bien en el procedimiento de ejecución o bien en la formulación del proyecto.

Control: El control de proyecto tiene como objetivo principal el mantener el proyecto alineado con sus objetivos. Analizar tendencias futuras que permitan estimar los costes y plazos de finalización del proyecto.

Evaluación: La evaluación consiste en hacer una apreciación sobre un proyecto en curso o acabado. Se trata de determinar la pertinencia de los objetivos y su grado de realización, la eficacia en cuanto a su desarrollo, la eficacia, el impacto y la viabilidad. Una evaluación debe proporcionar informes creíbles y útiles, que permitan mejorar de forma progresiva la gestión de los proyectos. Por eso se evalúa en todas las fases del ciclo del proyecto los mismos elementos clave, que permanece constantemente en el tiempo.

Figura 5: El Ciclo del Proyecto: Documentos Principales y Decisiones Claves



Fuente: (Comisión Europea, 2001).

2.5. INGENIERÍA DE SOFTWARE

2.5.1. Definición

La ingeniería de software es la aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento de software; es decir, la aplicación de la ingeniería al software. (Pressman, 2010).

La ingeniería de software es una tecnología con varias capas. Como se aprecia en la figura 7, cualquier enfoque de ingeniería (incluso la de software) debe basarse en un compromiso organizacional con la calidad. La administración total de la calidad, Six Sigma y otras filosofías similares alimentan la cultura de mejora continua, y es esta cultura la que lleva en última instancia al desarrollo de enfoques cada vez más eficaces de la ingeniería de software. El fundamento en el que se apoya la ingeniería de software es el compromiso con la calidad. (Pressman, 2010).

Figura 6: Capas de Ingeniería de Software

Fuente: (Pressman, 2010)

El fundamento para la ingeniería de software es la capa proceso. El proceso de ingeniería de software es el aglutinante que une las capas de la tecnología y permite el desarrollo racional y oportuno del software de cómputo. El proceso define una estructura que debe establecerse para la obtención eficaz de tecnología de ingeniería de software. El proceso de software forma la base para el control de la administración de proyectos de software, y establece el contexto en el que se aplican métodos técnicos, se generan productos del trabajo (modelos,

documentos, datos, reportes, formatos, y demás.), se establecen puntos de referencia, se asegura la calidad y se administra el cambio de manera apropiada. (Pressman, 2010).

Los métodos de la ingeniería de software proporcionan la experiencia técnica para elaborar software. Incluyen un conjunto amplio de tareas, como comunicación, análisis de los requerimientos, modelación del diseño, construcción del programa, pruebas y apoyo. Los métodos de la ingeniería de software se basan en un conjunto de principios fundamentales que gobiernan cada área de la tecnología e incluyen actividades de modelación y otras técnicas descriptivas. (Pressman, 2010).

Las herramientas de la ingeniería de software proporcionan un apoyo automatizado o semiautomatizado para el proceso y los métodos. Cuando se integran las herramientas de modo que la información creada por una pueda ser utilizada por otra, queda establecido un sistema llamado ingeniería de software asistido por computadora que apoya el desarrollo de software. (Pressman, 2010).

2.5.2. El Proceso de Desarrollo de Software

Un proceso es un conjunto de actividades, acciones y tareas que se ejecutan cuando va a crearse algún producto del trabajo. Una actividad busca lograr un objetivo amplio (por ejemplo, comunicación con los participantes) y se desarrolla sin importar el dominio de la aplicación, tamaño del proyecto, complejidad del esfuerzo o grado de rigor con el que se usará la ingeniería de software. Una acción (diseño de la arquitectura) es un conjunto de tareas que producen un producto importante del trabajo (por ejemplo, un modelo del diseño de la arquitectura). Una tarea se centra en un objetivo pequeño, pero bien definido (por ejemplo, realizar una prueba unitaria) que produce un resultado tangible. (Pressman, 2010).

La estructura del proceso establece el fundamento para el proceso completo de la ingeniería de software por medio de la identificación de un número pequeño de actividades estructurales que sean aplicables a todos los proyectos de software, sin importar su tamaño o complejidad. Además, la estructura del proceso incluye un conjunto de actividades sombrilla que son aplicables a través de todo el proceso del software. (Pressman, 2010).

Una estructura de proceso general para la ingeniería de software, según (Pressman, 2010), consta de cinco actividades:

Comunicación: Antes de que comience cualquier trabajo técnico, tiene importancia crítica comunicarse y colaborar con el cliente (y con otros participantes). Se busca entender los objetivos de los participantes respecto del proyecto, y reunir los requerimientos que ayuden a definir las características y funciones del software.

Planeación: Cualquier viaje complicado se simplifica si existe un mapa. Un proyecto de software es un viaje difícil, y la actividad de planeación crea un "mapa" que guía al equipo mientras viaja. El mapa, llamado plan del proyecto de software, define el trabajo de ingeniería de software al describir las tareas técnicas por realizar, los riesgos probables, los recursos que se requieren, los productos del trabajo que se obtendrán y una programación de las actividades.

Modelado: Ya sea usted diseñador de paisaje, constructor de puentes, ingeniero aeronáutico, carpintero o arquitecto, a diario trabaja con modelos. Crea un "bosquejo" del objeto por hacer a fin de entender el panorama general, cómo se verá arquitectónicamente, cómo ajustan entre sí las partes constituyentes y muchas características más. Si se requiere, refina el bosquejo con más y más detalles en un esfuerzo por comprender mejor el problema y cómo resolverlo. Un ingeniero de software hace lo mismo al crear modelos a fin de entender mejor los requerimientos del software y el diseño que los satisfará.

Construcción: Esta actividad combina la generación de código (ya sea manual o automatizada) y las pruebas que se requieren para descubrir errores en este.

Despliegue: El software (como entidad completa o como un incremento parcialmente terminado) se entrega al consumidor que lo evalúa y que le da retroalimentación, misma que se basa en dicha evaluación.

Estas cinco actividades estructurales genéricas se usan durante el desarrollo de programas pequeños y sencillos, en la creación de aplicaciones web grandes y en la ingeniería de sistemas enormes y complejos basados en computadoras. Los detalles del proceso de software serán distintos en cada caso, pero las actividades estructurales son las mismas. (Pressman, 2010).

Para muchos proyectos de software, las actividades estructurales se aplican en forma iterativa a medida que avanza el proyecto. Es decir, la comunicación, la planeación, el modelado, la construcción y el despliegue se ejecutan a través de cierto número de repeticiones del proyecto. Cada iteración produce un incremento del software que da a los participantes un subconjunto de características y funcionalidad generales del software. Conforme se produce cada incremento, el software se hace más y más completo. (Pressman, 2010).

2.5.3. Modelos Genéricos de Desarrollo de Software

Modelo de la Cascada

El modelo de la cascada, a veces llamado ciclo de vida clásico, sugiere un enfoque sistemático y secuencial para el desarrollo del software, que comienza con la especificación de los requerimientos por parte del cliente y avanza a través de planeación, modelado, construcción y despliegue, para concluir con el apoyo del software terminado. (Pressman, 2010).

Modelos de Proceso Incremental

El modelo incremental combina elementos de los flujos de proceso lineal y paralelo estudiados. El modelo incremental aplica secuencias lineales en forma escalonada a medida que avanza el calendario de actividades. Cada secuencia lineal produce “incrementos” de software susceptibles de entregarse de manera parecida a los incrementos producidos en un flujo de proceso evolutivo. (Pressman, 2010).

El Modelo Espiral

El modelo de desarrollo espiral es un generador de modelo de proceso impulsado por el riesgo, que se usa para guiar la ingeniería concurrente con participantes múltiples de sistemas intensivos en software. Tiene dos características distintivas principales. La primera es el enfoque cíclico para el crecimiento incremental del grado de definición de un sistema y su implementación, mientras que disminuye su grado de riesgo. La otra es un conjunto de puntos de referencia de anclaje puntual para asegurar el compromiso del participante con soluciones factibles y mutuamente satisfactorias. (Pressman, 2010).

Con el empleo del modelo espiral, el software se desarrolla en una serie de entregas evolutivas. Durante las primeras iteraciones, lo que se entrega puede ser un modelo o prototipo. En las iteraciones posteriores se producen versiones cada vez más completas del sistema cuya ingeniería se está haciendo. (Pressman, 2010).

2.6. METODOLOGÍAS PARA EL DESARROLLO DE SOFTWARE

En la ingeniería de software es vital definir los caminos a seguir para conseguir el éxito. Y para esto existen diferentes metodologías para aplicar de acuerdo con el proyecto que se desarrolle.

2.6.1. Metodologías Ágiles vs Metodologías Tradicionales

Metodologías Ágiles

Las metodologías ágiles son flexibles, pueden ser modificadas para que se ajusten a la realidad de cada equipo y proyecto. (Navarro, 2013)

Los proyectos ágiles se subdividen en proyectos más pequeños mediante una lista ordenada de características. Cada proyecto es tratado de manera independiente y desarrolla un subconjunto de características durante un periodo de tiempo corto, de entre dos y seis semanas. La comunicación con el cliente es constante al punto de requerir un representante de él durante el desarrollo. Los proyectos son altamente colaborativos y se adaptan mejor a los cambios; de hecho, el cambio en los requerimientos es una característica esperada y deseada, al igual que las entregas constantes al cliente y la retroalimentación por parte de él. Tanto el producto como el proceso son mejorados frecuentemente. (Navarro, 2013).

Metodologías Tradicionales

Las metodologías tradicionales de desarrollo de software son orientadas por planeación. Inician el desarrollo de un proyecto con un riguroso proceso de licitación de requerimientos, previo a etapas de análisis y diseño. Con esto tratan de asegurar resultados con alta calidad circunscritos a un calendario. (Navarro, 2013).

En las metodologías tradicionales se concibe un solo proyecto, de grandes dimensiones y estructura definida; se sigue un proceso secuencial en una sola dirección y sin marcha atrás; el proceso es rígido y no cambia; los requerimientos son acordados de una vez y para todo el proyecto, demandando grandes plazos de planeación previa y poca comunicación con el cliente una vez ha terminado ésta. (Navarro, 2013).

Comparación entre Metodologías

La siguiente tabla, muestra aspectos relevantes de las metodologías de desarrollo tradicional contrastándolas con los aspectos relevantes de las metodologías de

desarrollo ágil.

Tabla 1 : Metodologías Tradicionales vs Metodologías Ágiles

Metodologías Tradicionales	Metodologías Ágiles
Predictivos	Adaptativos
Orientadas a procesos	Orientadas a personas
Proceso rígido	Proceso flexible
Se concibe como un proyecto	Un proyecto es subdividido en varios proyectos más pequeños
Poca comunicación con el cliente	Comunicación constante con el cliente
Entrega de software al finalizar el desarrollo	Entregas constantes de software
Documentación extensa	Poca documentación

Fuente: (Navarro, 2013)

2.6.2. Metodología SCRUM

2.6.2.1. Definición de la metodología Scrum

La metodología Scrum es un proceso ágil para desarrollar software que fue aplicado por primera vez por Ken Schwaber y Jeff Sutherland., quienes lo documentaron en detalle en el libro Agile Software Development with Scrum. Esta metodología centra su atención en las actividades de Gerencia y no especifica prácticas de Ingeniería. Fomenta el surgimiento de equipos auto dirigidos cooperativos y aplica inspecciones frecuentes como mecanismo de control. (Peralta, 2003).

Scrum parte de la base de que los procesos definidos funcionan bien sólo si las entradas están perfectamente definidas y el ruido, ambigüedad o cambio es muy pequeño. Por lo tanto, resulta ideal para proyectos con requerimientos inestables, ya que fomenta el surgimiento de los mismos. El ciclo de vida definido por Scrum es incremental iterativo y se caracteriza por ser muy adaptable. (Peralta, 2003).

Principales Características de la Metodología

- ✓ Equipos autodirigidos.
- ✓ Utiliza reglas para crear un entorno ágil de administración de proyectos.
- ✓ No prescribe prácticas específicas de ingeniería.
- ✓ Los requerimientos se capturan como ítems de la lista Product Backlog.
- ✓ El producto se construye en una serie de Sprints de un mes de duración.

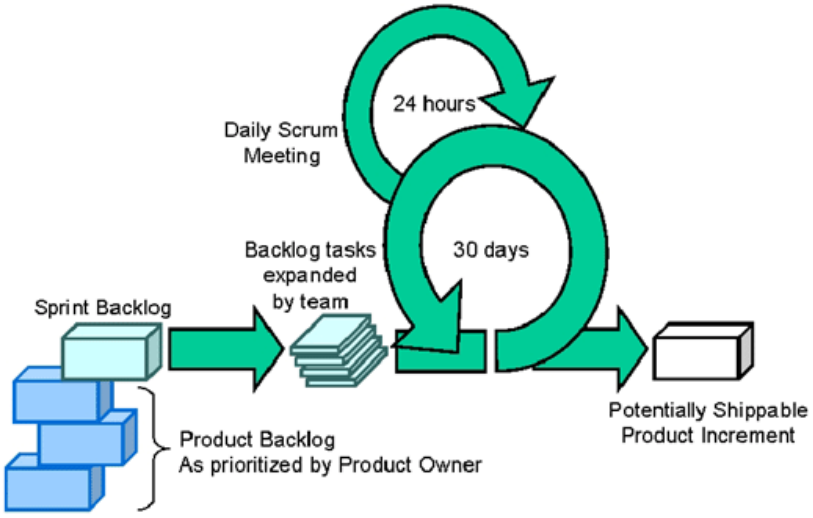
Principales Elementos de la Metodología

- **Herramientas**
  - ✓ Product Backlog.
  - ✓ Sprint Backlog.
- **Prácticas**
  - ✓ Sprints.
  - ✓ Sprint Planning Meeting.
  - ✓ Daily Meetings.
  - ✓ Sprint Review Meeting.
  - ✓ Design Review Meeting.
  - ✓ Stabilization Sprints.
  - ✓ Meta Scrums.

- **Roles y Responsabilidades**
  - ✓ Scrum Master.
  - ✓ Product Owner.
  - ✓ Scrum Team.
  - ✓ Customer.
  - ✓ Management.

Esquema General

Figura 7 : Proceso de Desarrollo Scrum



Fuente: (Peralta, 2003)

El trabajo a ser realizado en un proyecto Scrum es listado en el Product Backlog, que es una lista de todos los cambios requeridos sobre un producto.

Los proyectos se realizan durante una serie de iteraciones de un mes de duración llamadas Sprints. Al comienzo de cada Sprint tiene lugar una Sprint Planning Meeting durante la cual el Product Owner prioriza el Product Backlog y el Scrum Team selecciona las tareas que serán completadas durante el Sprint que va a comenzar. Esas tareas son removidas del Product Backlog para ser llevadas al Sprint Backlog.

Durante el Sprint el equipo se mantiene en contacto a través de los Daily Meetings. Y al final del Sprint debe mostrar la funcionalidad completa en la Sprint Review Meeting.

2.6.2.2. Herramientas y Prácticas de Scrum

Scrum no requiere ni provee prácticas de Ingeniería. En lugar de eso, específica prácticas y herramientas de gerencia que se aplican en sus distintas fases para evitar el caos originado por la complejidad e imposibilidad de realizar predicciones.

Product Backlog List

Es una lista priorizada que define el trabajo que se va a realizar en el proyecto. Cuando un proyecto comienza es muy difícil tener claro todos los requerimientos sobre el producto. Sin embargo, suelen surgir los más importantes que casi siempre son más que suficientes para un Sprint.

La Product Backlog List puede crecer y modificarse a medida que se obtiene más conocimiento acerca del producto y del cliente. Con la restricción de que solo puede cambiarse entre Sprints. El objetivo es asegurar que el producto definido al terminar la lista es el más correcto, útil y competitivo posible y para esto la lista debe acompañar los cambios en el entorno y el producto.

Existe un rol asociado con esta lista y es el de Product Owner. Si alguien quiere realizar cualquier modificación sobre la lista, por ejemplo: agregar o incrementar la prioridad de sus elementos tiene que convencer al Product Owner.

Sprints

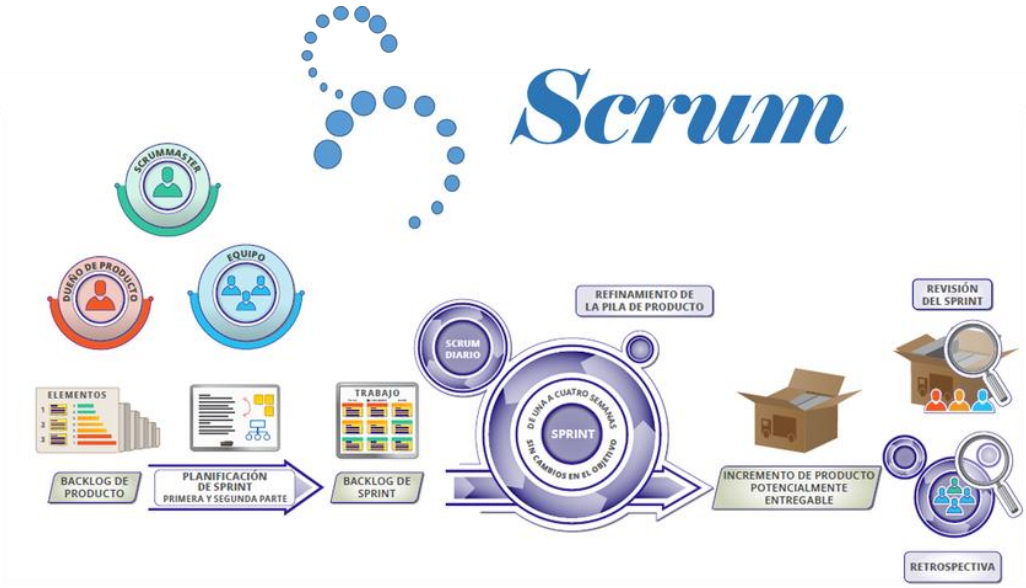
Un Sprint es el procedimiento de adaptación de las cambiantes variables del entorno (requerimientos, tiempo, recursos, conocimiento, tecnología). Son ciclos iterativos en los cuales se desarrolla o mejora una funcionalidad para producir nuevos incrementos. Durante un Sprint el producto es diseñado, codificado y probado. Y su arquitectura y diseño evolucionan durante el desarrollo.

El objetivo de un Sprint debe ser expresado en pocas palabras para que sea fácil de recordar y esté siempre presente en el equipo. Es posible definir una serie de restricciones que el equipo deba aplicar durante un Sprint.

Un Sprint tiene una duración planificada de entre una semana y un mes. No es posible introducir cambios durante el Sprint, por lo tanto, para planificar su duración hay que pensar en cuanto tiempo puedo comprometerme a mantener los cambios fuera del Sprint. Dependiendo del tamaño del sistema, la construcción de un reléase puede llevar entre 3 y 8 Sprints. Por otra parte, podrían formarse equipos para desarrollar en forma paralela distintos grupos de funcionalidad.

Las actividades que se desarrollan durante el Sprint son: Sprint Planning Meeting, Sprint Backlog, Daily Scrum Meetings y Sprint Review Meeting. En la figura 8 se pueden ver las prácticas involucradas en un Sprint.

Figura 8: Actividades del Sprint



Fuente: (Peralta, 2003)

Burn down Chart

En Scrum se planifica y mide el esfuerzo restante necesario para desarrollar el producto. Esta gráfica suele utilizarse en lugar de un diagrama de PERT debido a que el camino crítico en un desarrollo ágil cambia diariamente. Esto haría obsoleto el diagrama de PERT cada día. Es por esto que no es útil una herramienta que

modele el camino crítico a partir de actividades. La solución es utilizar una técnica que permita medir la velocidad de desarrollo. Para esto se utiliza el criterio del equipo a partir del cual se calcula diariamente el camino crítico. Esto permite recalcular el plan y la velocidad en que se realiza el trabajo. En función de esto el equipo puede trabajar para acelerar o desacelerar el trabajo para cumplir con la fecha de entrega.

Sprint Backlog

Es el punto de entrada de cada Sprint. Es una lista que tiene los ítems de la Product Backlog List que van a ser implementados en el siguiente Sprint.

Los ítems son seleccionados por el Scrum Team, el Scrum Master y el Product Owner en la Sprint Planning Meeting a partir de la priorización de los ítems y los objetivos que se marcaron para ese Sprint. A partir de los objetivos a cumplir durante el Sprint el Scrum Team determina que tareas debe desempeñar para cumplir el objetivo. De esto surge el Sprint Backlog. Es importante destacar que es el equipo quien se organiza para alcanzar el objetivo. El Manager no asigna tareas a los individuos y tampoco toma decisiones por el equipo. El equipo puede agregar nuevas tareas o remover tareas innecesarias en cualquier momento si lo considera necesario para cumplir el objetivo. Pero el Sprint Backlog solo puede ser modificado por el equipo. Las estimaciones se actualizan cada vez que aparece nueva información.

Stabilization Sprints

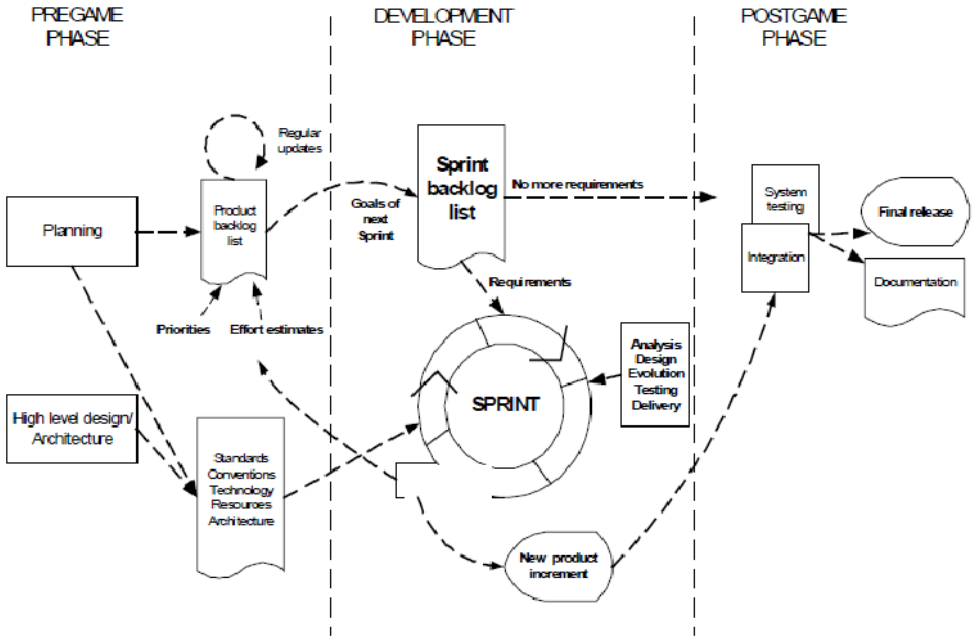
En estos Sprints el equipo se concentra en encontrar defectos, no en agregar funcionalidad. Suelen aplicarse cuando se prepara un producto para el release. Son útiles cuando se están realizando pruebas beta, se está introduciendo a un equipo en la metodología de Scrum o cuando la calidad de un producto no alcanza los límites esperados.

No fueron definidos por Scrum pero han sido recomendados por su utilidad al aplicar esta metodología.

2.6.2.3. El Proceso de Scrum

Scrum consta de tres fases: Pregame, Development y Postgame.

Figura 9: Fases Pregame, Development y Postgame



Fuente: (Peralta, 2003)

La fase de **Pregame** incluye dos subfases: Planning y Architecture.

- **Planning**

  Consiste en la definición del sistema que será construido. Para esto se crea la lista Product Backlog a partir del conocimiento que actualmente se tiene del sistema. En ella se expresan los requerimientos priorizados y a partir de ella se estima el esfuerzo requerido. La Product Backlog List es actualizada constantemente con ítems nuevos y más detallados, con estimaciones más precisas y cambios en la prioridad de los ítems.

- **Architecture / High level Design**

  El diseño de alto nivel del sistema se planifica a partir de los elementos existentes en la Product Backlog List. En caso de que el producto a construir sea una mejora a un sistema ya existente, se identifican los cambios necesarios para implementar los elementos que aparecen en la lista Product Backlog y el impacto que pueden tener estos cambios. Se sostiene una Design Review Meeting para examinar los objetivos de la implementación y tomar decisiones a partir de la revisión. Se preparan planes preliminares sobre el contenido de cada release.

La fase de **Development** también llamada Game Phase es la parte ágil de Scrum:

En esta fase se espera que ocurran cosas impredecibles. Para evitar el caos Scrum define prácticas para observar y controlar las variables técnicas y del entorno, así también como la metodología de desarrollo que hayan sido identificadas y puedan cambiar. Este control se realiza durante los Sprints. Dentro de variables de entorno encontramos: tiempo, calidad, requerimientos, recursos, tecnologías y herramientas de implementación. En lugar de tenerlas en consideración al comienzo del desarrollo, Scrum propone controlarlas constantemente para poder adaptarse a los cambios en forma flexible.

La fase de **PostGame**:

Contiene el cierre del reléase. Para ingresar a esta fase se debe llegar a un acuerdo respecto a las variables del entorno por ejemplo que los requerimientos fueron completados. El sistema está listo para ser liberado y es en esta etapa en la que se realiza integración, pruebas del sistema y documentación.

Pregame Phase

Entrada: La concepción inicial del producto que tienen los accionistas o interesados.

Tareas: Las tareas y asignación de responsabilidades de la fase de pregame son descritas en la siguiente tabla:

Tabla 2: Tareas de la fase Pregame

Nombre de la tarea	Descripción	Responsables
Crear la Product Backlog List y controlar su consistencia.	Posibles elementos de esta lista son requerimientos técnicos y del negocio, funciones, errores a reparar, defectos, mejoras y actualizaciones tecnológicas requeridas. Es importante controlar la consistencia de la lista. Para esto se agregan, modifican, eliminan, especifican y priorizan sus elementos.	Product Owner
Priorizar la Product Backlog List.	Esta actividad se basa en considerar que elementos tienen más o menos influencia en el éxito del proyecto en un momento dado; considerando que los elementos con mayor prioridad se realizan primero.	Product Owner
Effort Estimation	Es un proceso iterativo que reúne toda la información que haya acerca un elemento para tener un mayor nivel de precisión en la estimación. Siempre se mide el esfuerzo que falta para cumplir con el / los objetivos tanto a nivel de la lista Product Backlog como para el Sprint Backlog (lo que resta).	Product Owner Scrum Team
Design Review Meeting	En esta instancia se comunica el diseño a los interesados para revisar el cumplimiento de los ítems especificados en el Product Backlog.	

Fuente: (Peralta, 2003)

Verificación: Deben estar realizadas todas las tareas requeridas.

Salida: Product Backlog List, Architectura.

Development Phase

Entrada: Product Backlog List

Tareas:

Tabla 3: Tareas de la Fase de Desarrollo

Nombre de la tarea	Descripción	Responsables
Sprint Planning Meeting	Es una reunión organizada por el Scrum Master, que se realiza en dos fases. La primera fase tiene como objetivo establecer que ítems de la Product Backlog List van a ser realizados durante el Sprint. En la segunda fase se decide cómo se van a alcanzar los objetivos del Sprint. En esta fase se crea la Sprint Backlog, indicando qué tareas debe desempeñar el equipo para cumplir con dichos objetivos.	Scrum Master Customer, User Management Product Owner Scrum Team Scrum Team Scrum Master Product Owner
Daily Scrum Meeting	Las reuniones se realizan en el mismo lugar y a la misma hora cada día. Idealmente en la mañana para definir el trabajo para el día. Tienen una duración de 15 minutos y los participantes se quedan parados. Estas reuniones no se utilizan para resolver problemas. En ellas se realizan tres preguntas: ¿Qué hiciste ayer?, ¿Qué harás hoy? y ¿Qué obstáculos ves en tu camino? Los participantes son clasificados según el compromiso que tengan con las actividades del proyecto en dos categorías: gallinas y chanchos Los chanchos son los que están más comprometidos y por lo tanto son los que pueden hablar y brindar opiniones. Esto ayuda a evitar reuniones innecesarias.	Scrum Team
Sprint Review Meeting	Es una reunión informal que tiene como regla que su preparación no puede tomar más de 2 horas. En ella el equipo presenta lo que ha logrado durante el Sprint. Generalmente toma la forma de una demo de las nuevas características o la arquitectura.	Customers Management Product Owner Otros interesados

Fuente: (Peralta, 2003)

Verificación: Durante un sprint se puede acortar funcionalidad, pero la fecha de entrega debe ser respetada.
Salida: Incremento del producto.

2.6.2.4. Roles y Responsabilidades de Scrum

Scrum Master

Es un rol de administración que debe asegurar que el proyecto se está llevando a cabo de acuerdo con las prácticas, valores y reglas de Scrum y que todo funciona según lo planeado. Su principal trabajo es remover impedimentos y reducir riesgos del producto. Este rol suele ser desempeñado por un Gerente de Proyecto o Líder de equipo.

Product Owner

Es el responsable del proyecto, administra, controla y comunica la Backlog List. Es el responsable de encontrar la visión del producto y reflejarla en la Backlog List. Generalmente esta persona puede ser el Product Manager, Marketing, Internal Customer, etc.

Scrum Team

Es el equipo del proyecto que tiene la autoridad para decidir cómo organizarse para cumplir con los objetivos de un Sprint. Sus tareas son: Effort Estimation (Estimar Esfuerzo), crear el Sprint Backlog, revisar la Product Backlog List y sugerir obstáculos que deban ser removidos para cumplir con los ítems que aparecen.

Típicamente es un equipo de entre 5 y 10 personas cada una especializada en algún elemento que conforma los objetivos a cumplir, por ejemplo: Programadores, Diseñadores de Interfaz de usuario, etc. La dedicación de los

miembros del equipo debería ser full-time con algunas excepciones. La membresía solo puede cambiar entre sprints (no durante).

Customer

El cliente participa en las tareas que involucran la lista Product Backlog.

Management

Es el responsable de tomar las decisiones finales, acerca de estándares y convenciones a seguir durante el proyecto. Participa en la selección de objetivos y requerimientos y en la selección del Scrum Owner. Tiene la responsabilidad de controlar el progreso y trabaja junto con el Scrum Master en la reducción de la Product Backlog List.

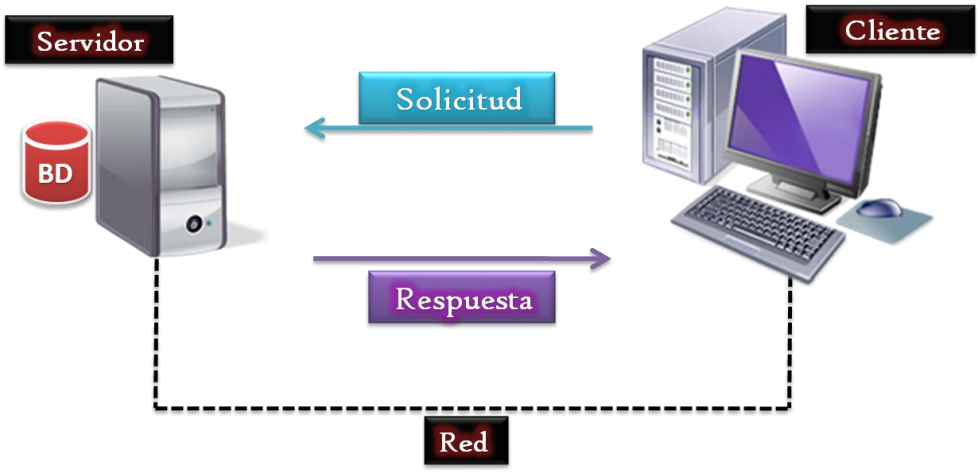
2.7. ARQUITECTURA CLIENTE/SERVIDOR

El modelo Cliente/Servidor es un modelo de aplicación distribuida en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes. Las aplicaciones Clientes realizan peticiones a una o varias aplicaciones Servidores, que deben encontrarse en ejecución para atender dichas demandas. (Marini, 2012).

El modelo Cliente/Servidor permite diversificar el trabajo que realiza cada aplicación, de forma que los Clientes no se sobrecarguen, cosa que ocurriría si ellos mismos desempeñan las funciones que le son proporcionadas de forma directa y transparente. En esta arquitectura la capacidad de proceso está repartida entre los clientes y los servidores, aunque son más importantes las ventajas de tipo organizativo debidas a la centralización de la gestión de la información y la separación de responsabilidades, lo que facilita y clarifica el diseño del sistema. Tanto el Cliente como el Servidor son entidades abstractas que pueden residir en la misma máquina o en máquinas diferentes. (Marini, 2012).

Una disposición muy común son los sistemas multicapa en los que el servidor se descompone en diferentes programas que pueden ser ejecutados por diferentes computadoras aumentando así el grado de distribución del sistema.

Figura 10: Modelo Cliente/Servidor



Fuente: (Marini, 2012)

Una aplicación cliente/servidor, es un servidor de base de datos al que varios usuarios realizan consultas simultáneamente. El proceso cliente realiza una consulta, el proceso servidor le envía las tablas resultantes de la consulta y el proceso cliente las interpreta y muestra el resultado en pantalla. Los sistemas distribuidos pueden consistir en diversos servidores que alojan datos, de forma que el cliente no tiene por qué conocer exactamente donde se encuentran, simplemente hace una petición de servicio, y es el sistema servidor encargado de localizarlos y proporcionar el resultado de la consulta al usuario que hizo la petición. (Marini, 2012).

2.8. ARQUITECTURA DDD (Domain Driven Design)

Definición de Diseño guiado por el dominio DDD

En el desarrollo de software, el enfoque del diseño guiado por el dominio se utiliza para necesidades muy complejas, para conectar la implementación a un modelo en evolución de la idea principal del modelo de negocio. Pone el foco en el problema relevante y básicamente ayuda a identificar la arquitectura e informar sobre los mecanismos que el software necesita replicar. (Administrador, 2019).

Definición de Dominio

Área temática o campo a la que un usuario aplica un software.

Definición de Modelo de Dominio

Representa la terminología y los conceptos clave del dominio del problema. Identifica las relaciones entre las entidades incluidas dentro del ámbito del dominio del problema, identifica sus atributos y proporciona una visión estructural del dominio. Domain Driven Design es un enfoque para el desarrollo de software definido por (Evans, 2015) en su libro Domain-driven design: Tackling Complexity in the Heart of Software, que se centra en un modelo rico, expresivo y en constante evolución para resolver problemas del dominio de una forma semántica. (Administrador, 2019).

2.8.1. Conceptos Clave del Domain Driven Design

Entidades (entities)

Las entidades son objetos del modelo que se caracterizan por tener identidad en el sistema, los atributos que contienen no son su principal característica. Representan conceptos con una identidad que se mantienen en el tiempo, y que con frecuencia también se mantienen bajo distintas representaciones de la entidad. Deben poder ser distinguidas de otros objetos, aunque tengan los mismos atributos. Tienen que poder ser consideradas iguales a otros objetos aun cuando sus atributos difieren. (Administrador, 2019).

Objetos de Valor (Value objects)

Al contrario que las entidades los value objects representan conceptos que no tienen identidad. Simplemente describen características. Por lo tanto, solo nos interesan sus atributos. (Administrador, 2019).

Los value object representan elementos del modelo que se describen por el qué son, y no por quién o cuál son.

Servicios (Services)

Los servicios representan operaciones, acciones o actividades que no pertenecen conceptualmente a ningún objeto de dominio concreto. Los servicios no tienen ni estado propio ni un significado más allá que la acción que los definen. (Administrador, 2019).

Al contrario que las entidades y los value objects, los servicios son definidos en términos de lo que pueden hacer por un cliente, y por tanto tienden a ser nombrados como verbos. Los verbos utilizados para nombrar a los servicios deben pertenecer al ubiquitous language, o ser introducidos en el caso de que aún no lo sean. A la hora de implementarlos tanto sus parámetros como resultados deben ser objetos pertenecientes al dominio. (Administrador, 2019).

Un servicio debe de cumplir tres características principales:

- La operación que lo define está relacionada con un concepto de dominio, pero no es natural modelarlo como una entidad o un value object.
- Su interfaz se especifica usando otros elementos del modelo de dominio.
- La operación no tiene estado, por lo que cualquier cliente podría usar cualquier instancia del servicio sin tener en cuenta las operaciones que se han realizado con anterioridad en esa instancia.

Podemos dividir los servicios en tres tipos diferentes según su relación con el núcleo del dominio.

Domain services

Son responsables del comportamiento más específico del dominio, es decir, realizan acciones que no dependen de la aplicación concreta que estemos desarrollando, sino que pertenecen a la parte más interna del dominio y que podrían tener sentido en otras aplicaciones pertenecientes al mismo dominio. Por ejemplo, crear un usuario, actualizar los detalles de un cliente, y demás. (Administrador, 2019).

Application services

Son responsables del flujo principal de la aplicación, es decir, son los casos de uso de nuestra aplicación. Son la parte visible al exterior del dominio de nuestro sistema, por lo que son el punto de entrada-salida para interactuar con la funcionalidad interna del dominio. Su función es coordinar entidades, value objects, domain services e infrastructure services para llevar a cabo una acción. Por ejemplo, realizar un pago, añadir un producto al carrito de la compra, realizar una transferencia a otra cuenta, y demás. (Administrador, 2019).

Infrastructure services

Declaran comportamiento que no pertenece realmente al dominio de la aplicación pero que debemos ser capaces de realizar como parte de este. Por ejemplo, enviar un email de confirmación tras realizar un pago, loguear transacciones, y demás. (Administrador, 2019).

Layered architecture

Un sistema software está compuesto por muchas partes, de las cuales, la parte que resuelve problemas para el dominio es una porción pequeña, aunque su importancia es desproporcionada a su tamaño. (Administrador, 2019).

Para ser capaces de trabajar con el dominio sin perdernos en otros detalles presentes en el software necesitamos desacoplar los objetos de dominio de otras funciones del sistema. Tendremos que aislar nuestro dominio del resto del sistema para así evitar confundir conceptos pertenecientes al dominio con conceptos que sólo están relacionados con la tecnología utilizada. (Administrador, 2019).

Podemos utilizar cualquiera de las muchas arquitecturas que existen para aislar las distintas partes del sistema, pero la opción que elijamos debería dividir nuestro sistema en al menos cuatro capas: presentación, aplicación, dominio e infraestructura. (Administrador, 2019).

Presentation

Capa responsable de mostrar la información al usuario e interpretar los eventos de entrada del usuario. Cabe destacar que el usuario puede ser un ser humano u otro sistema que se comunica con el nuestro. (Administrador, 2019).

Application

Capa que declara las funcionalidades que el software tiene que llevar a cabo y orquesta los objetos de dominio para resolver los distintos problemas. Esta capa no contiene reglas de negocio o conocimiento, solamente coordina y delega el trabajo a la colaboración de los objetos de dominio que se encuentran en la siguiente capa. (Administrador, 2019).

Domain

Capa donde se encuentran los conceptos del dominio y las reglas de negocio. Es la capa más importante del sistema y es la que realmente aporta valor y resuelve los problemas para los que un determinado software es creado. (Administrador, 2019).

Infrastructure

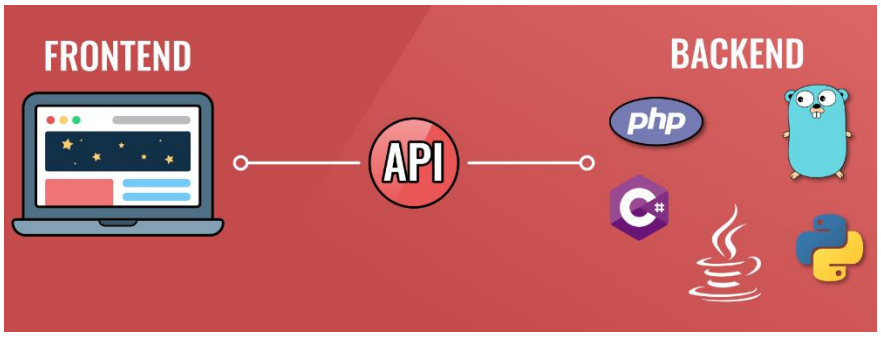
Capa que provee las implementaciones que apoyan a las capas definidas anteriormente. Aquí se encapsulan la mayoría de las decisiones tecnológicas adoptadas para un sistema, por ejemplo: el envío del email de confirmación tras un pago, persistencia para el dominio, comunicación con otros sistemas, y demás. (Administrador, 2019).

2.9. DIFERENCIAS ENTRE FRONTEND Y BACKEND

En el desarrollo web actual existen conceptos claves a tener en cuenta: Frontend y Backend. Decimos que son conceptos "actuales" debido a que anteriormente todo se hacía con PHP, increíble ¿cierto? pero todo empezó así, la estructura de las webs y la comunicación con el servidor se hacía con PHP, no era necesario utilizar JavaScript lo cual hoy en día es casi impensable para el desarrollo de una Web. (Díaz, 2020).

2.9.1. Desarrollo Web Moderno

Figura 11: Desarrollo Web Moderno



Fuente: (Díaz, 2020).

El desarrollo web moderno surge alrededor del 2008 cuando empieza el auge de HTML5[4], es en este momento dónde empezamos a hablar de Frontend y Backend, debido a que la tecnología web crece enormemente gracias a las APIs

[4] HTML5: Acrónimo de “HyperText Markup Language”

que vinieron con HTML5, estas APIs[5] hacen posible una comunicación entre Backend y Frontend, de esta manera cualquier Frontend se puede comunicar con cualquier Backend. (Díaz, 2020).

2.9.2. Definición de FrontEnd

Frontend es la parte de un programa o dispositivo a la que un usuario puede acceder directamente. Son todas las tecnologías de diseño y desarrollo web que corren en el navegador y que se encargan de la interactividad con los usuarios.

Un programador Frontend debe saber de códigos (HTML5, CSS[6] y JavaScript) para poder usar algunos frameworks o librerías que expanden sus capacidades para crear cualquier tipo de interfaces de usuarios. React, Redux, Angular, Bootstrap y Foundation son algunos de ellos. (Díaz, 2020).

2.9.3. Definición de BackEnd

Backend es la capa de acceso a datos de un software o cualquier dispositivo, que no es directamente accesible por los usuarios, además contiene la lógica de la aplicación que maneja dichos datos. El Backend también accede al servidor, que es una aplicación especializada que entiende la forma como el navegador solicita cosas. (Díaz, 2020).

Algunos de los lenguajes de programación de Backend son NodeJs, Python, PHP, Ruby y Java, y así como en Frontend, cada uno de los anteriores tiene diferentes frameworks que te permiten trabajar mejor según el proyecto que estás desarrollando. (Díaz, 2020).

[5] APIs: Acrónimo de “Application Programming Interface”

[6] CSS: Acrónimo de “Cascading StyleSheets” o “Hojas de Estilo en Cascada”

2.10. STACK DE DESARROLLO DE SOFTWARE

2.10.1. Sistema Operativo Linux

Para entender qué es Linux debemos saber antes qué es un sistema operativo (a veces se utilizan simplemente las siglas S.O.). Podemos dar una definición sencilla de este concepto: “Un sistema operativo es un programa que permite al usuario interactuar con el ordenador y sus componentes (monitor, disco duro, impresora, y demás) y que facilita la realización de tareas básicas como copiar o mover ficheros de un sitio a otro, editar archivos de texto, establecer una conexión a internet o hacer copias de seguridad”.

El sistema operativo es el primer programa que se ejecuta al encender el ordenador. A un nivel superior tenemos los programas que permiten al usuario realizar tareas específicas. Estos programas se denominan aplicaciones de usuario, o simplemente aplicaciones. Podemos encontrar muchos ejemplos en el trabajo diario con el ordenador: programas de gestión contable como ContaPlus, procesadores de texto como OpenOffice.org, MS Word, programas de retoque fotográfico como The Gimp o Photoshop. (Sánchez González , 2009).

Según (Sánchez González , 2009), GNU/Linux (comúnmente Linux a secas) es uno más de los muchos Sistemas Operativos que existen en la actualidad que tiene una serie de características que lo hacen especial:

- **Libre:** Se puede descargar de internet, se puede copiar y distribuir sin que por ello se incurra en ningún tipo de delito. La licencia que establece los términos de uso, copia y distribución se denomina Licencia GNU (www.gnu.org).
- **Hecho por voluntarios:** Linux no se creó para obtener beneficios con él sino para satisfacer una serie de necesidades a la hora de trabajar con el ordenador. Hoy día sigue funcionando así. Cuando alguien necesita un determinado programa, simplemente lo crea y lo pone al servicio de la comunidad para que lo use y para que cada cual lo mejore y lo adapte a sus propias necesidades.

- **Multiusuario:** Varios usuarios pueden conectarse y usar el mismo ordenador a la vez.
- **Multitarea:** Pueden funcionar varios programas al mismo tiempo en la misma máquina.
- **Multiplataforma:** Hay versiones de Linux para gran cantidad de plataformas: todos los PCs basados en procesadores Intel o AMD, ordenadores Digital/Compaq con procesadores Alpha, ordenadores Apple, ultraportátiles como el Asus Eee e incluso dispositivos móviles como el Sharp Zaurus.
- **Estable:** Linux es un sistema operativo muy maduro, probado durante mucho tiempo. Hay muchos servidores que llevan funcionando bajo Linux de forma ininterrumpida muchos años sin un solo cuelgue.
- **Eficiente:** Linux aprovecha bien los recursos hardware. Incluso los viejos Pentium pueden funcionar bien con Linux y servir para alguna tarea.
- **Hay miles de programas libres:** Hay una gran cantidad de programas, desde procesadores de texto hasta programas de dibujo pasando por todo tipo de servidores, totalmente libres y gratuitos que se pueden descargar e instalar desde el propio entorno de Linux.

2.10.2. Lenguaje de Programación JavaScript

JavaScript es un lenguaje de programación que permite el script de eventos, clases y acciones para el desarrollo de aplicaciones Internet entre el cliente y el usuario. JavaScript permite con nuevos elementos dinámicos ir más allá de clicar y esperar en una página Web. Los usuarios no leerán únicamente las páginas, sino que además las páginas ahora adquieren un carácter interactivo. Esta interacción permite cambiar las páginas dentro de una aplicación: poner botones, cuadros de texto, código para hacer una calculadora, un editor de texto, un juego, o cualquier otra cosa que pueda imaginarse. (ICTEA, 2020).

Según (Alvarez, Peña, & Fernandez, 2017), las propiedades más importantes del Lenguaje JavaScript son las siguientes:

- Se interpreta por el ordenador que recibe el programa, no se compila.

- Tiene una programación orientada a objetos. El código de los objetos está predefinido y es expandible. No usa clases ni herencia.
- El código está integrado (incluido) en los documentos HTML.
- Trabaja con los elementos del HTML.
- No se declaran los tipos de variables.
- Ejecución dinámica: los programas y funciones no se chequean hasta que se ejecutan.
- Los programas de JavaScript se ejecutan cuando sucede algo, a ese algo se le llama evento.

2.10.3. Nodejs

NodeJS, conocido habitualmente también con la palabra "node" a secas, surge en 2009 como respuesta a algunas necesidades encontradas a la hora de desarrollar sitios web, específicamente el caso de la concurrencia y la velocidad.

NodeJS es una plataforma super-rápida, especialmente diseñada para realizar operaciones de entrada / salida (Input / Output o simplemente I/O en inglés) en redes informáticas por medio de distintos protocolos, apegada a la filosofía UNIX. Es además uno de los actores que ha provocado, junto con HTML5, que Javascript gane gran relevancia en los últimos tiempos, pues ha conseguido llevar al lenguaje a nuevas fronteras como es el trabajo del lado del servidor. (Morales A. , 2012).

¿Qué es NodeJS?

"Node Yei es", tal como se pronuncia NodeJS en inglés, es básicamente un framework para implementar operaciones de entrada y salida, como decíamos anteriormente. Está basado en eventos, streams y construido encima del motor de Javascript V8, que es con el que funciona el Javascript de Google Chrome.

Si queremos entender esta plataforma, lo primero que debemos de hacer es desprendernos de varias ideas que los desarrolladores de Javascript hemos cristalizado a lo largo de los años que llevamos usando ese lenguaje. Para empezar, NodeJS se programa del lado del servidor, lo que indica que los procesos para el desarrollo de software en "Node" se realizan de una manera muy diferente que los de Javascript del lado del cliente. (Morales A. , 2012).

De entre alguno de los conceptos que cambian al estar Node.JS del lado del servidor, está el asunto del "Cross Browser", que indica la necesidad en el lado del cliente de hacer código que se interprete bien en todos los navegadores. Cuando trabajamos con Node solamente necesitamos preocuparnos de que el código que escribas se ejecute correctamente en tu servidor. El problema mayor que quizás podamos encontrarnos a la hora de escribir código es hacerlo de calidad, pues con Javascript existe el habitual problema de producir lo que se llama "código espagueti", o código de mala calidad que luego es muy difícil de entender a simple vista y de mantener en el futuro. (Morales A. , 2012).

Otras de las cosas que deberías tener en cuenta cuando trabajas con NodeJS, que veremos con detalle más adelante, son la programación asíncrona y la programación orientada a eventos, con la particularidad que los eventos en esta plataforma son orientados a cosas que suceden del lado del servidor y no del lado del cliente como los que conocemos anteriormente en Javascript "común". (Morales A. , 2012).

Además, NodeJS implementa los protocolos de comunicaciones en redes más habituales, de los usados en Internet, como puede ser el HTTP, DNS, TLS, SSL, etc. Mención especial al protocolo SPDY, fácilmente implementado en Node, que ha sido desarrollado mayoritariamente por Google y que pretende modernizar el protocolo HTTP, creando un sistema de comunicaciones que es sensiblemente más rápido que el antiguo HTTP (apuntan un rendimiento 64% superior). (Morales A. , 2012).

2.10.4. Framework VueJS

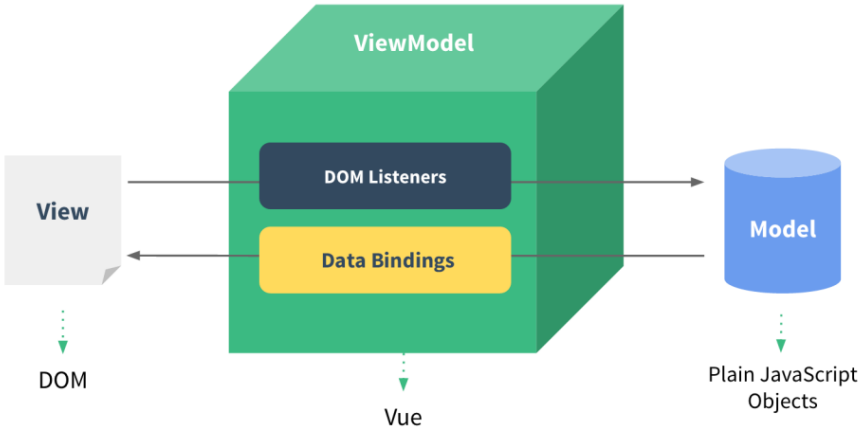
Vue fue creado por Evan You, ex-trabajador de Google, que decidió crear su propio framework en el año 2014. Desde entonces, Vue ha subido muchísimo de popularidad, gracias a su sencillez y a todo lo que puede ofrecer, que lo vamos a ver a continuación.

Según (By Equipo Geek, 2020), las características de VueJs son las siguientes:

- **Un framework progresivo**: Podemos incluir las partes que necesitamos además de la librería core y luego incluir otras librerías como si fueran módulos separados. Esto permite añadir funcionalidad a medida que lo necesitemos. El core principal de VueJS está formado por una librería encargada de renderizar vistas en el navegador.
- **Intuitivo, moderno y fácil de usar**: VueJS no ha reinventado la rueda, pero desde sus comienzos su premisa es que sea tan fácil y simple de usar que la comunidad pueda usarlo para sus proyectos sin apenas curva de aprendizaje.
- **Un ecosistema muy variado que cubre todo lo necesario**: VueJS tiene a su alrededor una serie de herramientas que ayudan a conseguir que el desarrollador sepa en todo momento qué está haciendo y cómo lo está haciendo. Desde una interface de línea de comandos (CLI) a una plantilla base que podemos invocar mediante la herramienta vue-cli disponible en los repositorios npm, pasando por un largo etcétera de módulos y complementos.
- **Una comunidad muy activa**: Revisiones constantes, buena documentación, comunidad dispuesta a echar una mano cuando estás atascado. Un lujo.
- **Todo el código de un componente se encuentra en un único fichero**: Los componentes guardan todo lo necesario en ficheros con extensión .vue. Estos ficheros contienen de manera ordenada todo el HTML, el CSS y el JavaScript necesario.

El siguiente diagrama muestra cómo se relacionan estas piezas básicas.

Figura 12: Arquitectura VueJs



Fuente: (By Equipo Geek, 2020).

2.10.5. Framework ExpressJS

Express es el framework web más popular de Node, y es la librería subyacente para un gran número de otros frameworks web de Node populares. Proporciona mecanismos para:

- Escritura de manejadores de peticiones con diferentes verbos HTTP en diferentes caminos URL (rutas).
- Integración con motores de renderización de "vistas" para generar respuestas mediante la introducción de datos en plantillas.
- Establecer ajustes de aplicaciones web como qué puerto usar para conectar, y la localización de las plantillas que se utilizan para renderizar la respuesta.
- Añadir procesamiento de peticiones "middleware" adicional en cualquier punto dentro de la tubería de manejo de la petición.

A pesar de que Express es en sí mismo bastante minimalista, los desarrolladores han creado paquetes de middleware compatibles para abordar casi cualquier problema de desarrollo web. Hay librerías para trabajar con cookies, sesiones, inicios de sesión de usuario, parámetros URL, datos POST, cabeceras de

seguridad y muchos más. Puedes encontrar una lista de paquetes middleware mantenida por el equipo de Express en Express Middleware (junto con una lista de algunos de los paquetes más populares de terceros). (Morales A. , 2015).

2.10.6. Gestor de Base de Datos PostgreSQL

“PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente. Es el sistema de gestión de bases de datos de código abierto más potente del mercado”. (Zea, Molina, & Redrován, 2017).

“PostgreSQL utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando”. (Zea, Molina, & Redrován, 2017).

Según el sitio oficial de PostgreSQL son varias las características de este software, las cuales se detallan a continuación:

- Es una base de datos 100% ACID
- Integridad referencial
- Tablespaces
- Nested transactions (savepoints)
- Replicación asincrónica/sincrónica / Streaming replication - Hot Standby
- Two-phase commit
- PITR - point in time recovery
- Copias de seguridad en caliente (Online/hot backups)
- Unicode
- Juegos de caracteres internacionales
- Regionalización por columna
- Multi-Version Concurrency Control (MVCC)
- Múltiples métodos de autentificación

- Acceso encriptado vía SSL
- Actualización in-situ integrada (pg_upgrade)
- SE-postgres
- Completa documentación
- Licencia BSD
- Disponible para Linux y UNIX en todas sus variantes (AIX, BSD, HP-UX, SGI IRIX, Mac OSX, Solaris, Tru64) y Windows 32/64bit.

Otra definición sobre PostgreSQL indica que es “Un sistema de base de datos relacionales es un sistema que permite la manipulación de acuerdo con las reglas del álgebra relacional. Los datos se almacenan en tablas de columnas y renglones. Con el uso de llaves, esas tablas se pueden relacionar unas con otras.” (Zea, Molina, & Redrován, 2017).

2.11. HERRAMIENTAS DE DESARROLLO DE SOFTWARE

2.11.1. Visual Studio Code

Visual Studio Code, es un editor de código fuente desarrollado por Microsoft para Windows, Linux y macOS. Incluye soporte para depuración, control de Git integrado, resaltado de sintaxis, finalización de código inteligente, fragmentos de código y refactorización de código. También es personalizable, de modo que los usuarios pueden cambiar el tema del editor, los métodos abreviados de teclado y las preferencias. Es gratuito y de código abierto. (EcuRed contributors, 2020).

2.11.1.1. Características de Visual Studio Code

El código combina la interfaz de usuario optimizada de un editor moderno con asistencia y navegación de código enriquecido y una experiencia de depuración integrada, sin la necesidad de un IDE completo. Visual Studio Code, cuenta con herramientas de Debug hasta opciones para actualización en tiempo real de nuestro código en la vista del navegador y compilación en vivo de los lenguajes que lo requieran. Además de las extensiones, tendremos la posibilidad de optar

por otros themes o bien configurarlo a nuestro gusto. Para modificar el esquema de colores y los iconos. (EcuRed contributors, 2020).

2.11.1.2. Ventajas al usar Visual Studio Code

- Se puede utilizar como lenguajes de programación.
- Visual Studio Code es una herramienta que tiene soporte nativo para gran variedad de lenguajes, entre ellos podemos destacar los principales del desarrollo Web: HTML, CSS, y JavaScript, entre otros.
- Posibilidad de configurar la interfaz a nuestro gusto. De esta forma, podremos tener más de un código visible al mismo tiempo, las carpetas de nuestro proyecto y también acceso a la terminal o un detalle de problemas, entre otras posibilidades.
- Existencia de una amplísima gama de temas o estilos visuales para Visual Studio Code, que hacen el trabajo con el software más agradable a la vista.
- Goza de un soporte técnico formidable pues debido a su frecuente uso por la comunidad de desarrolladores, se puede encontrar fácilmente documentación y ayuda en foros y sitios relacionados.

2.11.2. PgAdmin 4

pgAdmin es una herramienta indispensable para gestionar y administrar PostgreSQL, la base de datos de código abierto más avanzada del mundo. Por lo tanto, pgAdmin es la herramienta para gestionar nuestras bases de datos espaciales PostGIS. (Morales A. , 2018).

El software tiene la apariencia de una aplicación de escritorio sea cual sea el entorno de tiempo de ejecución (escritorio o web), y mejora enormemente respecto a pgAdmin III con elementos de interfaz de usuario actualizados, opciones de despliegue multiusuario / web, paneles y un diseño más moderno. (Morales A. , 2018).

2.11.3. UML (Lenguaje Unificado de Modelado)

(UML, por sus siglas en inglés, Unified Modeling Language) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; está respaldado por el Object Management Group (OMG). (Pressman, 2010).

Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos, funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y compuestos reciclados. (Pressman, 2010).

UML proporciona 13 diferentes diagramas para su uso en modelado de software. En este proyecto se utilizará solamente diagramas de clase, implementación, caso de uso, secuencia, actividad y estado que a continuación se detalla:

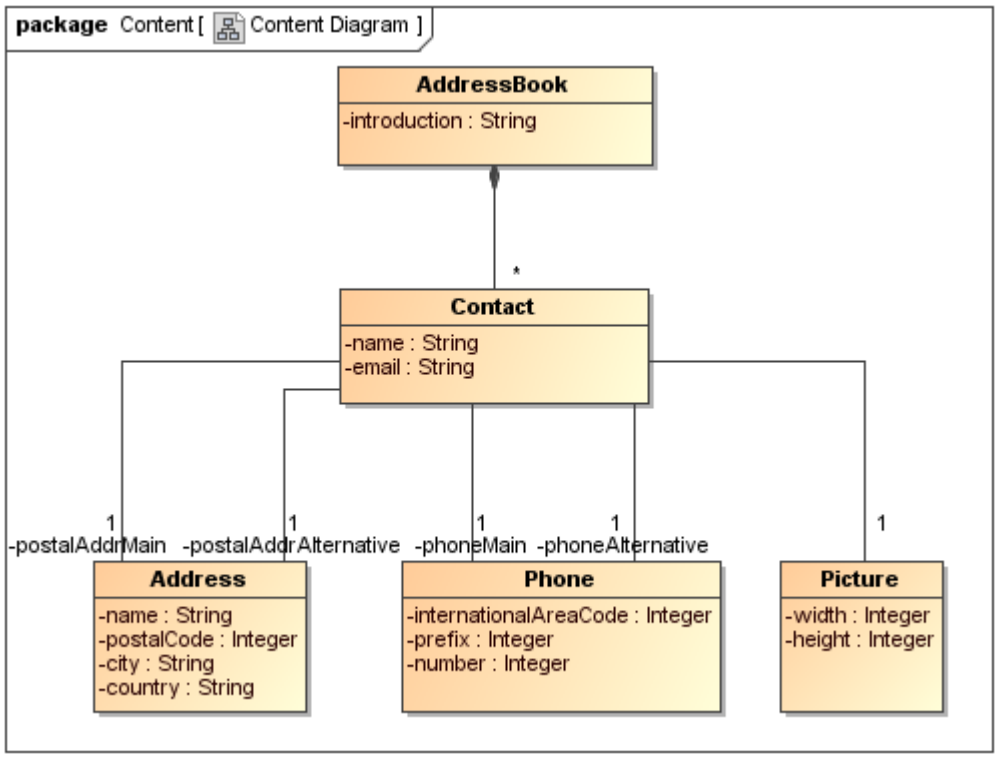
2.11.3.1. Diagrama de Clases

Para modelar clases, incluidos sus atributos, operaciones, relaciones y asociaciones con otras clases, el UML proporciona un diagrama de clase, que aporta una visión estática o de estructura de un sistema, sin mostrar la naturaleza dinámica de las comunicaciones entre los objetos de las clases. (Pressman, 2010).

Los elementos principales de un diagrama de clase son cajas, que son los íconos utilizados para representar clases e interfaces. Cada caja se divide en partes horizontales. La parte superior contiene el nombre de la clase. La sección media menciona sus atributos. Un atributo es algo que un objeto de dicha clase conoce o puede proporcionar todo el tiempo. Por lo general, los atributos se implementan como campos de la clase, pero no necesitan serlo. Podrían ser valores que la clase puede calcular a partir de sus variables o valores instancia y que puede obtener de otros objetos de los cuales está compuesto. Por ejemplo, un objeto puede conocer siempre la hora actual y regresarla siempre que se le solicite. Por

tanto, sería adecuado mencionar la hora actual como un atributo de dicha clase de objetos. Sin embargo, el objeto muy probablemente no tendría dicha hora almacenada en una de sus variables instancia, porque necesitaría actualizar de manera continua ese campo. En vez de ello, el objeto probablemente calcularía la hora actual (por ejemplo, a través de consulta con objetos de otras clases) en el momento en el que se le solicite la hora. La tercera sección del diagrama de clase contiene las operaciones o comportamientos de la clase. Una operación es lo que pueden hacer los objetos de la clase. Por lo general, se implementa como un método de la clase. (Pressman, 2010).

Figura 13: Diagrama de Clases



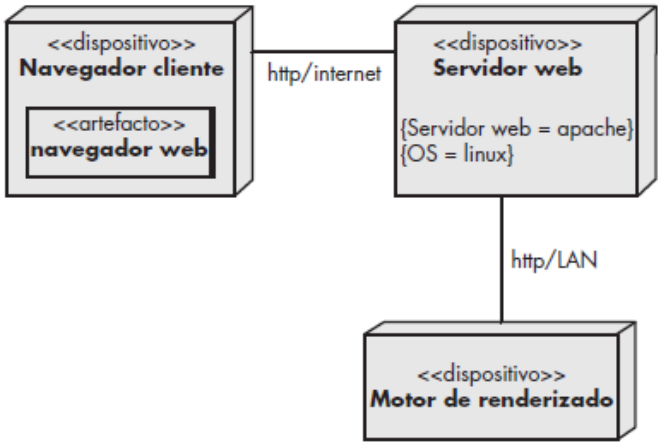
Fuente: (Nieves G., Ucan P., & Medez D., 2014)

2.11.3.2. Diagrama de Implementación

Un diagrama de implementación UML se enfoca en la estructura de un sistema de software y es útil para mostrar la distribución física de un sistema de software entre plataformas de hardware y entornos de ejecución. Suponga, por ejemplo,

que desarrolla un paquete de renderizado de gráficos basado en web. Los usuarios de su paquete usarán su navegador web para ir a su sitio web e ingresar la información que se va a renderizar. Su sitio web renderizaría una imagen gráfica de acuerdo con la especificación del usuario y la enviaría de vuelta al usuario. Puesto que las gráficas renderizadas pueden ser computacionalmente costosas, usted decide mover el renderizado afuera del servidor web y hacia una plataforma separada. Por tanto, habrá tres dispositivos de hardware involucrados en su sistema: el cliente web (la computadora que corre un navegador del usuario), la computadora que alberga el servidor web y la computadora que alberga el motor de renderizado. (Pressman, 2010).

Figura 14 : Diagrama de Implementación

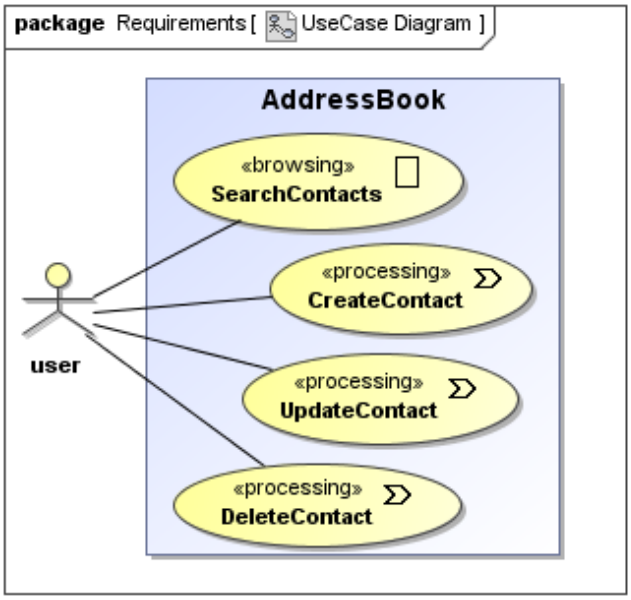


Fuente: (Pressman, 2010).

2.11.3.3. Diagrama de Casos de Uso

Un diagrama de caso de uso es un modelo de las funciones previstas del sistema y su entorno y sirve como un contrato entre el cliente y los desarrolladores. Los casos de uso sirven como hebra de unión a lo largo del desarrollo del sistema. El mismo modelo de caso de uso es el resultado de la disciplina de requisitos. (Nieves G., Ucan P., & Medez D., 2014).

Figura 15: Diagrama de Casos de Uso



Fuente: (Engineering, 2016).

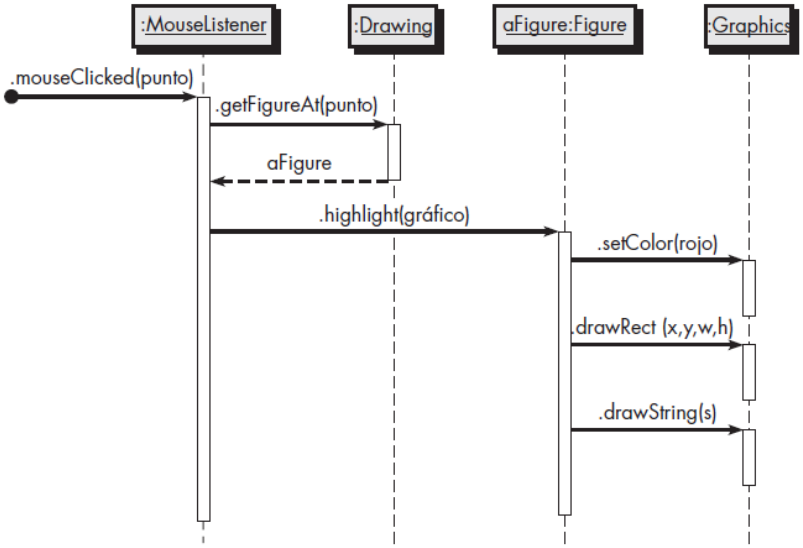
Como se muestra en la figura anterior, un diagrama de casos de uso consta de los siguientes elementos:

- Actor.
- Casos de Uso.
- Relaciones de Uso, Herencia y Comunicación.

2.11.3.4. Diagrama de Secuencia

En contraste con los diagramas de clase y con los diagramas de implementación, que muestran la estructura estática de un componente de software, un diagrama de secuencia se usa para mostrar las comunicaciones dinámicas entre objetos durante la ejecución de una tarea. Este tipo de diagrama muestra el orden temporal en el que los mensajes se envían entre los objetos para lograr dicha tarea. Puede usarse un diagrama de secuencia para mostrar las interacciones en un caso de uso o en un escenario de un sistema de software. (Pressman, 2010).

Figura 16: Diagrama de Secuencia



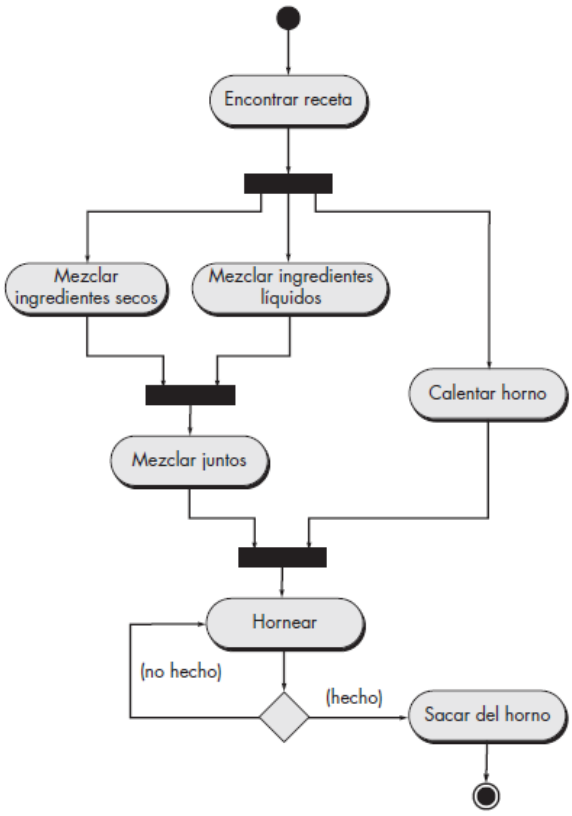
Fuente: (Pressman, 2010).

2.11.3.5. Diagrama de Actividad

Un diagrama de actividad UML muestra el comportamiento dinámico de un sistema o de parte de un sistema a través del flujo de control entre acciones que realiza el sistema. Es similar a un diagrama de flujo, excepto porque un diagrama de actividad puede mostrar flujos concurrentes. (Pressman, 2010).

El componente principal de un diagrama de actividad es un nodo acción, representado mediante un rectángulo redondeado, que corresponde a una tarea realizada por el sistema de software. Las flechas desde un nodo acción hasta otro indican el flujo de control; es decir, una flecha entre dos nodos acción significa que, después de completar la primera acción, comienza la segunda acción. Un punto negro sólido forma el nodo inicial que indica el punto de inicio de la actividad. Un punto negro rodeado por un círculo negro es el nodo final que indica el fin de la actividad. (Pressman, 2010).

Figura 17: Diagrama de Actividad



Fuente: (Pressman, 2010).

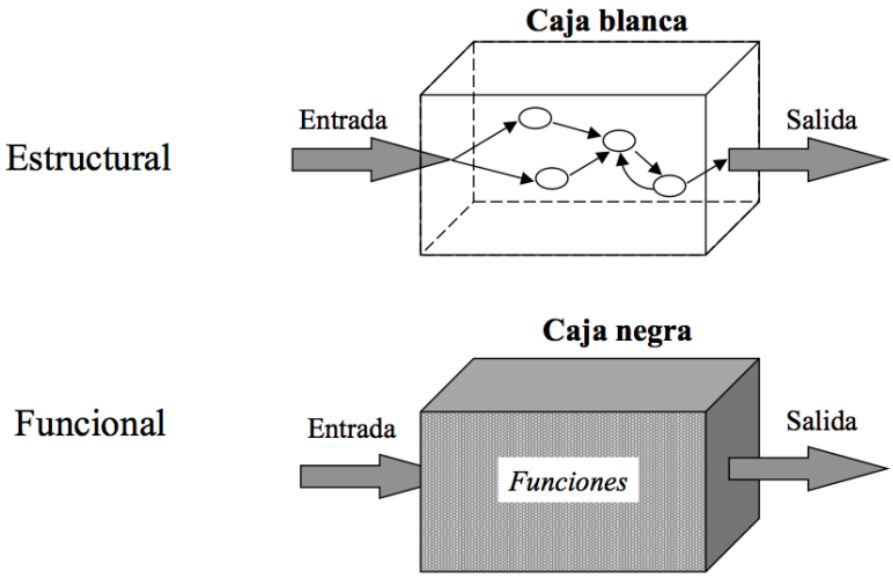
2.12. PRUEBAS DE SOFTWARE

El objetivo de la prueba es descubrir errores en el software. Es necesario crear buenos casos de prueba (aquel que tiene una alta probabilidad de mostrar errores aún no descubiertos).

Una prueba tiene éxito si descubre un error no detectado hasta entonces.
Cualquier producto de software se puede probar de dos formas:

- Pruebas de caja negra
- Pruebas de caja blanca

Figura 18: Técnicas de Prueba



Fuente: (Pressman, 2010).

2.12.1. Prueba de Caja Negra

Las pruebas de caja negra, también llamadas pruebas de comportamiento, se enfocan en los requerimientos funcionales del software; es decir, las técnicas de prueba de caja negra le permiten derivar conjuntos de condiciones de entrada que revisarán por completo todos los requerimientos funcionales para un programa. Las pruebas de caja negra no son una alternativa para las técnicas de caja blanca. En vez de ello, es un enfoque complementario que es probable que descubra una clase de errores diferente que los métodos de caja blanca. (Pressman, 2010).

Según (Pressman, 2010), las pruebas de caja negra intenta encontrar errores en las categorías siguientes:

- Funciones incorrectas o faltantes
- Errores de interfaz
- Errores en las estructuras de datos o en el acceso a bases de datos externas
- Errores de comportamiento o rendimiento y

- Errores de inicialización y terminación.

2.12.2. Prueba de Caja Blanca

Según (Pressman, 2010), la prueba de caja blanca, en ocasiones llamada prueba de caja de vidrio, es una filosofía de diseño de casos de prueba que usa la estructura de control descrita como parte del diseño a nivel de componentes para derivar casos de prueba. Al usar los métodos de prueba de caja blanca, puede derivar casos de prueba que:

- Garanticen que todas las rutas independientes dentro de un módulo se revisaron al menos una vez
- Revisen todas las decisiones lógicas en sus lados verdadero y falso
- Ejecuten todos los bucles en sus fronteras y dentro de sus fronteras operativas y
- Revisen estructuras de datos internas para garantizar su validez.

2.13. CALIDAD DE SOFTWARE

Incluso los desarrolladores de software más experimentados estarán de acuerdo en que obtener software de alta calidad es una meta importante. Pero, ¿cómo se define la calidad del software? En el sentido más general se define como: Proceso eficaz de software que se aplica de manera que crea un producto útil que proporciona valor medible a quienes lo producen y a quienes lo utilizan. (Pressman, 2010).

Según (Pressman, 2010), existen tres puntos importantes en la calidad de software:

- Un *proceso eficaz de software* establece la infraestructura que da apoyo a cualquier esfuerzo de elaboración de un producto de software de alta calidad. Los aspectos de administración del proceso generan las verificaciones y equilibrios que ayudan a evitar que el proyecto caiga en el

caos, contribuyente clave de la mala calidad. Las prácticas de ingeniería de software permiten al desarrollador analizar el problema y diseñar una solución sólida, ambas actividades críticas de la construcción de software de alta calidad. Por último, las actividades sombrilla, tales como administración del cambio y revisiones técnicas, tienen tanto que ver con la calidad como cualquier otra parte de la práctica de la ingeniería de software.

- Un *producto útil* entrega contenido, funciones y características que el usuario final desea; sin embargo, de igual importancia es que entrega estos activos en forma confiable y libre de errores. Un producto útil siempre satisface los requerimientos establecidos en forma explícita por los participantes. Además, satisface el conjunto de requerimientos (por ejemplo, la facilidad de uso) con los que se espera que cuente el software de alta calidad.
- Al *agregar valor para el productor y para el usuario* de un producto, el software de alta calidad proporciona beneficios a la organización que lo produce y a la comunidad de usuarios finales. La organización que elabora el software obtiene valor agregado porque el software de alta calidad requiere un menor esfuerzo de mantenimiento, menos errores que corregir y poca asistencia al cliente. Esto permite que los ingenieros de software dediquen más tiempo a crear nuevas aplicaciones y menos a repetir trabajos mal hechos. La comunidad de usuarios obtiene valor agregado porque la aplicación provee una capacidad útil en forma tal que agiliza algún proceso de negocios. El resultado final es 1) mayores utilidades por el producto de software, 2) más rentabilidad cuando una aplicación apoya un proceso de negocios y 3) mejor disponibilidad de información, que es crucial para el negocio.

2.13.1. Factores de Calidad ISO/IEC 9126

La norma ISO 9126 se define como un estándar internacional, se publicó en 1992, y define los propósitos para la evaluación de la calidad de software, como la

adquisición, requerimientos, desarrollo, uso, evaluación, soporte, mantenimiento, aseguramiento de la calidad y auditoría de software. (Calderón M., 2016).

Este estándar está fraccionado en cuatro partes, las cuales se encargarán de dirigir, las métricas externas, las métricas internas, la calidad en las métricas de uso y expendido.

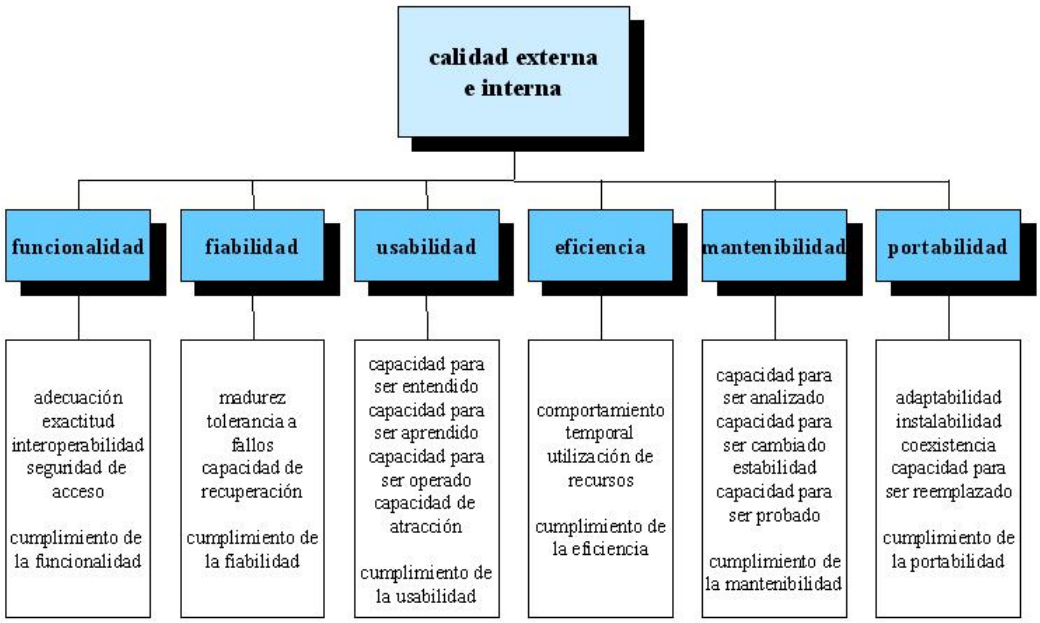
Los modelos de calidad para el software se describen así:

- **Calidad interna y externa**: Pormenoriza los detalles mediante la aplicación de 6 características para calidad interna y externa. Cuando el software se utiliza como una parte de un sistema informático global, estas divisiones se aprecian externamente, y se presentan como el resultado de atributos internos de software.
- **Calidad en uso**: es el resultado final que contempla el cliente, después de aplicar las 6 características de la calidad interna y externa del software. También se especifican 4 características para la calidad en uso.

Al agrupar calidad interna, calidad externa y la calidad en uso se conseguirá un modelo para la evaluación mucho más robusto y eficaz.

Según (Calderón M., 2016), el estándar describe 6 características generales, las cuales se detallarán a continuación, y en la siguiente figura:

Figura 19: Características de la Norma ISO/IEC 9126



Fuente: (Informática, 2017).

- **Funcionalidad:** Es la capacidad que tiene el software para cumplir y dotar de las funciones necesarias para satisfacer las necesidades explícitas e implícitas cuando es utilizado en condiciones específicas. Incluye las siguientes subcaracterísticas:
    - **Adecuación:** Propiedades que tiene el software y están relacionadas con la presencia y aptitud de un conjunto de funciones para tareas especificadas.
    - **Exactitud:** Atributos del software relacionados con la disposición de resultados o correctos.
    - **Interoperabilidad:** Atributos del software que se relacionan con su pericia para la interacción con sistemas especificados.
    - **Seguridad:** Propiedades del software relacionadas con su capacidad para advertir y evitar acceso no autorizados, deliberados o fortuitos, a aplicaciones o bases de datos.
    - **Cumplimiento funcional:** La capacidad del software de cumplir los estándares referentes a la funcionalidad.

- **Fiabilidad:** Es el conjunto de atributos que tienen relación con la capacidad del software para mantener su nivel de prestación durante un período establecido. Sus subcaracterísticas son:
    - **Madurez:** Capacidad que presentará el sistema y que conseguirá disminuir la probabilidad de sufrir fallos.
    - **Recuperabilidad:** Atributos relacionados con la magnitud de restablecer el nivel de continuar trabajando y recuperación del estado y de los datos afectados en caso de avería.
    - **Tolerancia a fallos:** Capacidad de mantener una cota de garantía y estabilidad adecuada en el caso de sufrir inconvenientes no esperados.
    - **Cumplimiento de Fiabilidad:** Posibilidad de aglutinar normativas y legislación que tengan como finalidad garantizar la fiabilidad del sistema.

- **Usabilidad:** Es el conjunto de propiedades de un software, que facilita a todos los roles de usuario final, poder trabajar y gestionarlo con la suficiente destreza e intuición. Sus subcaracterísticas son:
    - **Aprendizaje:** Cuantifica la cantidad de tiempo necesaria que necesita un usuario para poder conocer con cierta soltura potencial de las funciones y recursos que ofrece el software.
    - **Comprensión:** Mide el nivel de rapidez y facilidad con el que los usuarios pueden conocer el potencial de las funciones del software.
    - **Operatividad:** Facilidad con la que el usuario puede realizar las operaciones que desea en cada momento para ejecutar una acción.
    - **Atractividad:** Valora la presentación gráfica del software, así como la muestra de las acciones.

- **Eficiencia:** Se refiere a la valoración y cuantificación entre la cantidad de recursos que ofrece la aplicación, y que puede ser en relación con el tiempo

de resolución de las operaciones, como por el número de recursos que ofrece. Existen dos alternativas:

- **Comportamiento en el tiempo:** Cantidad de tiempo de respuesta y procesamiento necesarios para ejecutar una acción con el software.
- **Comportamiento de recursos:** Cantidad y número de tipos de recursos necesarios para la aplicación para resolver determinadas tareas.

- **Mantenibilidad:** Magnitud que cuantifica la facilidad que presenta la aplicación para ser ampliada, alterada o depurar posibles fallos que se localicen. Sus subcaracterísticas serían:
  - **Estabilidad:** Fortaleza con la que el sistema responde ante posibles riesgos o situaciones inesperadas.
  - **Facilidad de análisis:** Rapidez del sistema para diagnosticar y mostrar los resultados de posibles fallos o circunstancias excepcionales.
  - **Facilidad de cambio:** Versatilidad con la que el software puede ser modificado por cuestiones de actualizaciones o imprevistos sufridos.
  - **Facilidad de pruebas:** Medida con la que se valora la simplicidad con la que la aplicación ejecuta distintas funciones

- **Portabilidad:** Es la capacidad y potencia de la que dispone una aplicación o sistema software para poder ser trasladado de unas plataformas a otras.
  - **Capacidad de instalación:** Esfuerzo que se necesita realizar para proceder a la instalación de una aplicación a un entorno concreto.
  - **Capacidad de reemplazamiento:** Esfuerzo necesario para trabajar con una aplicación diferente a otro software previo en un entorno concreto.

- **Calidad de uso:** Son el conjunto de propiedades que envuelven a la seguridad del sistema y la aprobación por el cliente del software creado.

- **Eficacia:** Capacidad del software para resolver los procesos que el cliente le encarga.
- **Productividad:** Cuantificación del rendimiento con el que el sistema resuelve las labores que le encomienda el usuario.
- **Seguridad:** Capacidad de evitar riesgos e intrusiones, voluntarias o involuntarias, de acceso no autorizado al sistema.
- **Satisfacción:** Grado de conformidad que presenta el cliente del software encargado en origen y del producto obtenido.

El atributo “Conformidad” no se detalla anteriormente porque se especificó en todas las características.

2.14. ESTIMACIÓN DE COSTOS DEL PROYECTO

Una parte importante de la toma de decisiones al comenzar un nuevo proyecto de desarrollo de software está dada por el costo que éste tendrá. La estimación de estos costos ha preocupado a analistas de sistema, gerentes de proyecto e ingenieros de software durante décadas. El primer obstáculo es clarificar el alcance del proyecto. ¿Qué debería ser capaz de hacer el sistema?, la captura de los requisitos funcionales en casos de uso ha ayudado considerablemente a comunicar los requisitos de una forma que sea comprensible a los usuarios y a otros expertos en el campo. Al comienzo del proyecto debe hacerse un modelo de caso de uso que contenga una lista de todos los actores (usuarios o sistemas externos) y casos de uso del sistema, así como sus nombres y una breve descripción de los mismos. Esta información hace más fácil alcanzar un acuerdo sobre el tamaño del sistema al comienzo del proyecto. (Dekker, 2009).

2.14.1. Modelo Cocomo II

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 COCOMO, por COnstructive COst MOdel: modelo constructivo de costos.

El modelo COCOMO 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).

Según (Calero, 2010), el modelo COCOMO II consta de 3 ecuaciones las que mostraremos a continuación:

Tabla 4 : Ecuaciones por tipo de modelo COCOMO II: Básico e Intermedio

Ecuación	Submodelo básico	Submodelo intermedio
Esfuerzo (E)	$(\mathrm{E}) = \mathrm{a}(\mathrm{KLDC})^{\mathrm{b}}$	$(\mathrm{E}) = \mathrm{a}(\mathrm{KLDC})^{\mathrm{b}} * \mathrm{ME}$
Tiempo (T)	$(\mathrm{T}) = \mathrm{c}(\mathrm{E})^{\mathrm{d}}$	$(\mathrm{T}) = \mathrm{c}(\mathrm{E})^{\mathrm{d}}$
Personal (P)	$(\mathrm{P}) = \mathrm{E}/\mathrm{T}$	$(\mathrm{P}) = \mathrm{E}/\mathrm{T}$

Fuente: (Calero, 2010).

Dónde:

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

T: 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 distribuidas 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 unos pocos miles de líneas (tamaño pequeño) a unas docenas de miles (tamaño medio).

- **Modo semi – libre o semi – acoplado:** Corresponde a un esquema intermedio entre 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.

La tabla 5 muestra los coeficientes del proyecto de software de acuerdo a los tres modos expuestos anteriormente.

Tabla 5: Constante de Coste Modelo Básico

Proyecto de Software	A	B	C	D
Orgánico	2.4	1.05	1.05	0.38
Semi-Acoplado	3.0	1.12	2.5	0.35
Empotrado	3.6	1.20	2.5	0.32

Fuente: (Calero, 2010).

2.15. SEGURIDAD

La información es como el aparato circulatorio para las organizaciones y requiere que se proteja ante cualquier amenaza que pueda poner en peligro las empresas tanto públicas como privadas, pues en otro caso podría dañarse la salud empresarial. La información, como uno de los principales activos de las organizaciones, debe protegerse a través de la implantación, mantenimiento y mejora de las medidas de seguridad para que cualquier empresa logre sus objetivos de negocio, garantice el cumplimiento legal, de prestigio y de imagen de la compañía. (Fernández, 2012).

2.15.1. Norma ISO/IEC 27001

La ISO 27001 es una norma internacional emitida por la Organización Internacional de Normalización (ISO) dicha norma describe cómo gestionar la seguridad de la información en una empresa y/o institución. ISO 27001 puede ser implementada en cualquier tipo de organización, con o sin fines de lucro, privada o pública, pequeña o grande. (Zapata Ch., 2018).

Según (Zapata Ch., 2018), para garantizar que el Sistema de Gestión de la Seguridad de la Información (SGSI) funcione de manera correcta se debe identificar el ciclo de vida de los aspectos más relevantes que asegura la ISO como:

- **Confidencialidad:** La seguridad de la información no se deja a disposición ni de individuos, organizaciones o procesos que no estén autorizados.
- **Integridad:** Hace referencia al mantenimiento de la exactitud y complejidad de la información y los métodos de proceso.
- **Disponibilidad:** El acceso y utilización de la información y los sistemas de tratamiento por parte de los individuos, entidades o procesos autorizados cuando lo requieran. De igual forma se deben desarrollar e implementar planes para mantener o recuperar las operaciones y asegurar la disponibilidad de la información en el grado y la escala de tiempos requeridos, posterior a la falla de los procesos críticos para el proyecto.

2.15.2. JWT (Json Web Token)

En esencia JWT es un bloque de información en formato JSON que está firmado. Gracias a esta firma se puede verificar su autenticidad, y al estar en un formato JSON, el tamaño es pequeño y familiar. JWT son siglas de Json Web Token, que es una las herramientas más importantes en la parte de seguridad y autenticación que hace a todas las empresas que ofrezcan servicios por medios que requieran el uso de la web. Es una herramienta sencilla, y lo suficientemente liviana, y esto es lo que hace tan factible su uso. (Olivera, 2014).

CAPÍTULO III

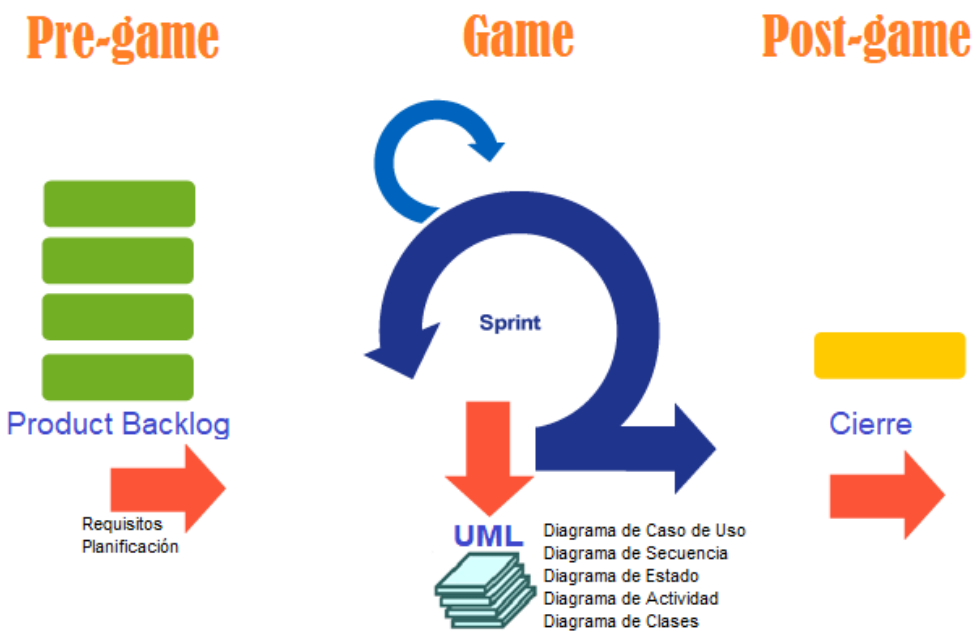
3. MARCO APLICATIVO

3. MARCO APLICATIVO

3.1. INTRODUCCIÓN

En el presente capítulo aplicaremos todo lo referente a los fundamentos técnicos para el análisis, diseño correspondiente al sistema, la metodología de desarrollo de software a usar en el siguiente proyecto es **Scrum** y todos los pasos que implican llevar adelante el desglose de la metodología ya antes mencionada, a la par se dará uso de la herramienta (UML – Lenguaje Unificado de Modelado), el cual nos permitirá modelar los módulos de la aplicación web.

Figura 20: Modelo de Implementación del Proyecto



Al iniciar cada sprint (iteración) se determina que partes se van a construir, tomando como criterios la prioridad para el negocio y los alcances que tendrá cada sprint. El presente proyecto es un sistema de información mediante la web, por lo cual cada sprint, contendrá los procesos necesarios que indica SCRUM, de acuerdo a las necesidades encontradas.

3.1.1. Identificación de Roles Scrum

El grupo de trabajo para el presente proyecto estará conformado de la siguiente manera, ver tabla 6.

Tabla 6: Roles Scrum

Rol		*Nombre*
Product Owner		Dr. Absalon Conurana Surco
Scrum Master		Eddy Heberson Choque Tito
Scrum Team	Analista Diseñador Desarrollador Testeador	Eddy Heberson Choque Tito
Stakeholders		Gobierno Autónomo Municipal de Catacora

3.2. PRE-GAME (ANTES DEL DESARROLLO)

3.2.1. Diagnóstico de la Situación Actual

Actualmente el Gobierno Autónomo Municipal de Catacora no está dividido por distritos, por lo que no cuenta con subalcaldes. El municipio se divide en cuatro cantones que son: Catacora, Pairumani Grande, Tolacollo, Parachi y una Sub-central Pajchiri, las cuales están organizadas por sindicatos y centrales agrarias.

Estas organizaciones tienen como función primordial la de gestionar y supervisar la ejecución de los proyectos inscritos en el Plan Operativo Anual (POA), dicha tarea es realizada posteriormente a la aprobación del mismo por el Honorable Consejo Municipal de Catacora, hasta su cumplimiento.

A continuación, se proporciona los pasos más importantes por la que atraviesa un proyecto:

Paso 1: Inicio del Proceso

- **Inscripción del proyecto:** Las unidades solicitantes (comunidades) por intermedio de sus autoridades inscriben sus proyectos a la Unidad Técnica del Gobierno Autónomo Municipal de Catacora.
- **Proyecto consolidado:** Es aquel proyecto que se encuentra inscrito en el POA, y este es aprobado por el Honorable Consejo Municipal de Catacora, dichos proyectos se ejecutan inmediatamente.
- **Elaboración del proyecto:** Intervienen la Unidad Técnica del Gobierno Autónomo Municipal de Catacora, juntamente con las Unidades solicitantes que en este caso son las comunidades representadas por sus autoridades. Ambas unidades coordinan para aprobar un determinado proyecto.

Paso 2: Seguimiento y Control de Proyectos

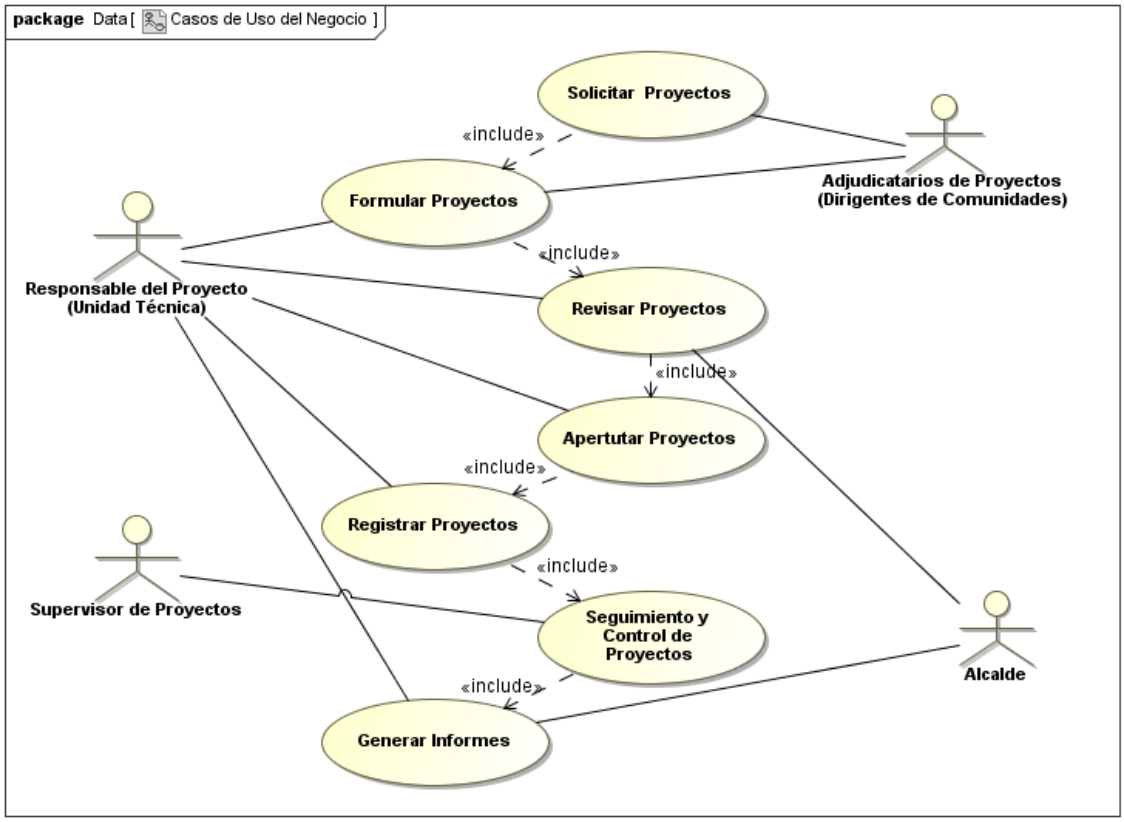
- El paso siguiente corresponde al seguimiento, control y ejecución de proyectos de parte de la alcaldía municipal, por medio de un supervisor de obras, el cual mediante los cronogramas realizará informes del estado del proyecto.
- La unidad de supervisión de obras, registra la empresa adjudicada, fecha de adjudicación, fecha de inicio de obra y demás.

3.2.2. Modelado del Negocio

El modelado del negocio es una técnica para comprender los procesos de una organización en este caso el Gobierno Autónomo de Catacora. Dichos procesos son descritos en términos de casos de usos del negocio actores del contexto que se corresponden con los procesos del sistema y los usuarios.

El modelo de negocio, identifica los procesos más importantes del contexto del sistema, describiendo procesos relacionados con actores y casos de usos encontrados en el Gobierno Autónomo de Catacora, y que se muestra a continuación en la siguiente figura.

Figura 21: Diagrama de Caso de Usos del Negocio



Los actores involucrados en el modelo de negocio del Gobierno Autónomo Municipal de Catacora son:

- **Alcalde:** Es la persona que da el visto bueno para la ejecución de los proyectos, el alcalde tiene la posibilidad de ingresar al sistema y poder monitorear el avance según la búsqueda, clasificación y generación de reportes de los proyectos.
- **Responsable del Proyecto (Unidad Técnica):** Es la persona que ocupa el cargo de Director en la Unidad Técnica del municipio, se encarga de realizar el respectivo registro, seguimiento, control y clasificación de los proyectos. Otra de las funciones es realizar la revisión de la documentación y dar curso a los proyectos.
- **Supervisor de Proyectos:** Es la persona encargada de realizar el seguimiento y control de ejecución de proyectos, dentro de sus funciones está el adjuntar informes sobre el avance de acuerdo al cronograma del proyecto.

- **Adjudicatarios de Proyectos:** Son autoridades de las Comunidades beneficiadas que se atribuyen el proyecto presentado en POA.

3.2.3. Creación del Product Backlog (Pila de Producto)

El presente proyecto se realizó un análisis de requerimiento, mediante reuniones, y entrevistas en el Gobierno Autónomo Municipal de Catacora para genera la pila de producto.

A continuación, se presentará el Product Backlog del proyecto, que tiene los requerimientos y características finales del sistema.

Tabla 7: Product Backlog (Pila de Producto)

ID	DESCRIPCIÓN	PRIORIDAD	MODULO	ESTADO
R1	Base de datos para Sistema Información para el Seguimiento y Control de Ejecución de Proyecto Municipales	Alta	Plataforma	Terminado
R2	Registrar usuarios para el ingreso al sistema. Y determinar un respectivo rol. Adicionar, modificar y eliminar	Alta	Contenido	Terminado
R3	Registrar datos de todas la Comunidades pertenecientes al municipio.	Media	Contenido	
R4	Registrar datos de todas las categorías de los Proyectos.	Media	Contenido	
R5	Registra datos de la Empresas que ejecutan un determinado Proyecto. Adicionar, modificar y eliminar	Media	Contenido	Terminado
R6	Registrar Poas para una determinada Comunidad. Adicionar, modificar y eliminar	Media	Contenido	Terminado
R7	Registrar datos del Proyecto Adicionar, modificar y eliminar	Alta	Contenido	Terminado
R8	Registrar el seguimiento de un Proyecto mediante un	Media	Contenido	Terminado

	cronograma. Adicionar, modificar y eliminar			
R9	Registrar el avance de un Proyecto Adicionar, modificar y eliminar	Media	Contenido	Terminado
R10	Realizar Reportes	Media	Contenido	Terminado

3.2.4. Análisis de Riesgo

Un riesgo es la probabilidad de que ocurra algo adverso durante el desarrollo del Proyecto, existen 3 tipos de riesgos:

- **Riesgo del Proyecto**: Es el que afecta a la calendarización o recursos del proyecto.
- **Riesgo del Producto**: Afectan a la calidad o rendimiento del software que se está desarrollando.
- **Riesgo de Negocio**: Afecta a la organización que desarrolla o suministra el software.

En la siguiente tabla se muestran los riesgos que se identificaron para el desarrollo del siguiente proyecto, las cuales describimos a continuación:

Tabla 8: Análisis de Riesgo

RIESGO	PROBABILIDAD	EFECTO	ESTRATEGIA
No se cumplan las fechas establecidas en el cronograma.	Alta	Tolerable	Realizar un segundo cronograma que sea más flexible.
Riesgo de que haya cambios de requerimientos de la institución que no hayan sido considerados.	Moderado	Tolerable	Realizar una revisión constante a los requerimientos Programar reuniones constantes con el dueño del producto (Product Owner).

No cumplan los plazos de entrega de producto.	Moderada	Serio	Agilizar los procesos de desarrollo del producto. Realizar una correcta planificación considerando el tiempo y los alcances del proyecto.
La institución no cuenta con los recursos, y existe el riesgo de que no tenga el equipamiento necesario.	Alto	Tolerable	Solicitar con anticipación el equipamiento y las condiciones necesarias para la implementación del sistema.
No se construya con algunos de los requerimientos.	Alto	Tolerable	Realizar un análisis para el desarrollo del módulo faltante

3.3. GAME (DURANTE EL DESARROLLO)

Durante esta etapa del proyecto se desarrollaron 3 iteraciones, cada una de ellas corresponde a un elemento. A continuación, se desglosan las actividades realizadas en cada una de estas etapas.

La estrategia que se utilizó para el desarrollo de cada iteración fue construir en un principio los modelos propios de la metodología UML y posteriormente implementarlos utilizando como elemento central la Base de Datos. La solución por la que se optó para la persistencia de objetos fue el hacer corresponder cada clase del modelo conceptual con una tabla de la base de datos, en donde las filas representan las instancias de los objetos y las columnas a los atributos de la clase. Finalmente se desarrolló el Sistema de Información para el Seguimiento y Control de Proyectos Municipales, en base a los todos los modelos que se diseñaron para su respectiva elaboración.

3.3.1. Primera Iteración

Durante la primera iteración se desarrollaron los elementos pertenecientes a Registro, Control y Seguimiento de los procesos. Las actividades realizadas

durante esta iteración se observan en la siguiente tabla que constituye el backlog del Sprint.

Tabla 9: Backlog del Primer Sprint

		Sprint	Inicio	Duración
		1	05-08-2020	31 días
ID	**Tarea**	**Tipo**	**Días de trabajo**	**Estado**
1.1	Realizar la planificación de la iteración.	Planificación	2	Terminado
1.2	Analizar los requerimientos de Product Backlog del proyecto.	Planificación	1	Terminado
1.3	Identificar los procesos más importantes en el municipio con el modelado del negocio.	Desarrollo	1	Terminado
1.4	Analizar los requerimientos de iteración con los modelos de casos de uso del sistema.	Desarrollo	3	Terminado
1.5	Construir diagramas de estado para el mejor entendimiento de los casos de uso.	Desarrollo	2	Terminado
1.6	Construir un modelo conceptual.	Desarrollo	2	Terminado
1.7	Diseñar y construir la base de datos del sistema.	Desarrollo	3	Terminado
1.8	Construir el modelo de implementación.	Desarrollo	2	Terminado
1.9	Construir diagramas de paquetes.	Desarrollo	2	Terminado

1.10	Construir diagramas de componentes.	Desarrollo	2	Terminado
1.11	Construir diagramas de despliegue.	Desarrollo	2	Terminado
1.12	Desarrollar la página de ingreso al sistema	Desarrollo	4	Terminado
1.13	Realizar Módulo de Administración del Sistema	Desarrollo	5	Terminado

Las funcionalidades correspondientes al incremento de la iteración son:

- Construcción de la Base de Datos independiente del Sistema.
- Página de ingreso con control de acceso a los usuarios.
- Módulo de Administración del Sistema.

3.3.2. Segunda Iteración

En la segunda iteración se realizó un proceso de análisis y complementos a la anterior iteración, las actividades realizadas se las puede observar en la tabla del Backlog del Sprint.

Tabla 10: Backlog del Segundo Sprint

		Sprint	Inicio	Duración
		1	05-09-2020	31 días
ID	**Tarea**	**Tipo**	**Días de trabajo**	**Estado**
1.1	Realizar la planificación de la iteración.	Planificación	2	Terminado
1.2	Analizar los requerimientos de Product Backlog del proyecto.	Planificación	1	Terminado
1.3	Analizar los requerimientos de	Desarrollo	3	Terminado

	iteración con los modelos de casos de uso del sistema.			
1.4	Complementar diagramas de estado para el mejor entendimiento de los casos de uso.	Desarrollo	3	Terminado
1.5	Complementar la base de datos del sistema.	Desarrollo	3	Terminado
1.6	Complementar el modelo de implementación.	Desarrollo	2	Terminado
1.7	Complementar diagramas de paquetes.	Desarrollo	2	Terminado
1.8	Complementar diagramas de componentes.	Desarrollo	2	Terminado
1.9	Complementar diagramas de despliegue.	Desarrollo	2	Terminado
1.10	Realizar Módulo de Gestión de Proyectos	Desarrollo	6	Terminado

En la segunda iteración se desarrollaron las siguientes funcionalidades para el sistema:

- Módulo de Gestión de Proyectos, donde se incluyen catálogo de empresas, gestión de poas y gestión de cronogramas.
- Acción para el seguimiento y Control de Proyectos mediante cronograma de actividades.

3.3.3. Tercera Iteración

Finalmente, durante la tercera iteración se desarrollaron los requerimientos finales del software, como la generación de reportes, pruebas del sistema entre otros.

Las actividades realizadas en esta iteración se pueden observar en la siguiente tabla, que constituye el backlog del tercer Sprint.

Tabla 11: Backlog del Tercer Sprint

		Sprint	Inicio	Duración
		1	05-10-2020	19 días
ID	**Tarea**	**Tipo**	**Días de trabajo**	**Estado**
1.1	Realizar la planificación de la iteración, que contemplara el modulo reportes y pruebas del sistema.	Planificación	2	Terminado
1.2	Analizar los requerimientos de Product Backlog del proyecto.	Planificación	1	Terminado
1.3	Analizar los requerimientos de iteración con los modelos de casos de uso del sistema.	Desarrollo	3	Terminado
1.4	Complementar diagramas de estado para el mejor entendimiento de los casos de uso.	Desarrollo	3	Terminado
1.5	Complementar diagramas de paquetes.	Desarrollo	2	Terminado
1.6	Complementar diagramas de componentes.	Desarrollo	2	Terminado
1.7	Complementar diagramas de despliegue.	Desarrollo	2	Terminado
1.8	Generación de reportes	Desarrollo	4	Terminado
1.9	Pruebas del Sistema	Desarrollo	2	Terminado

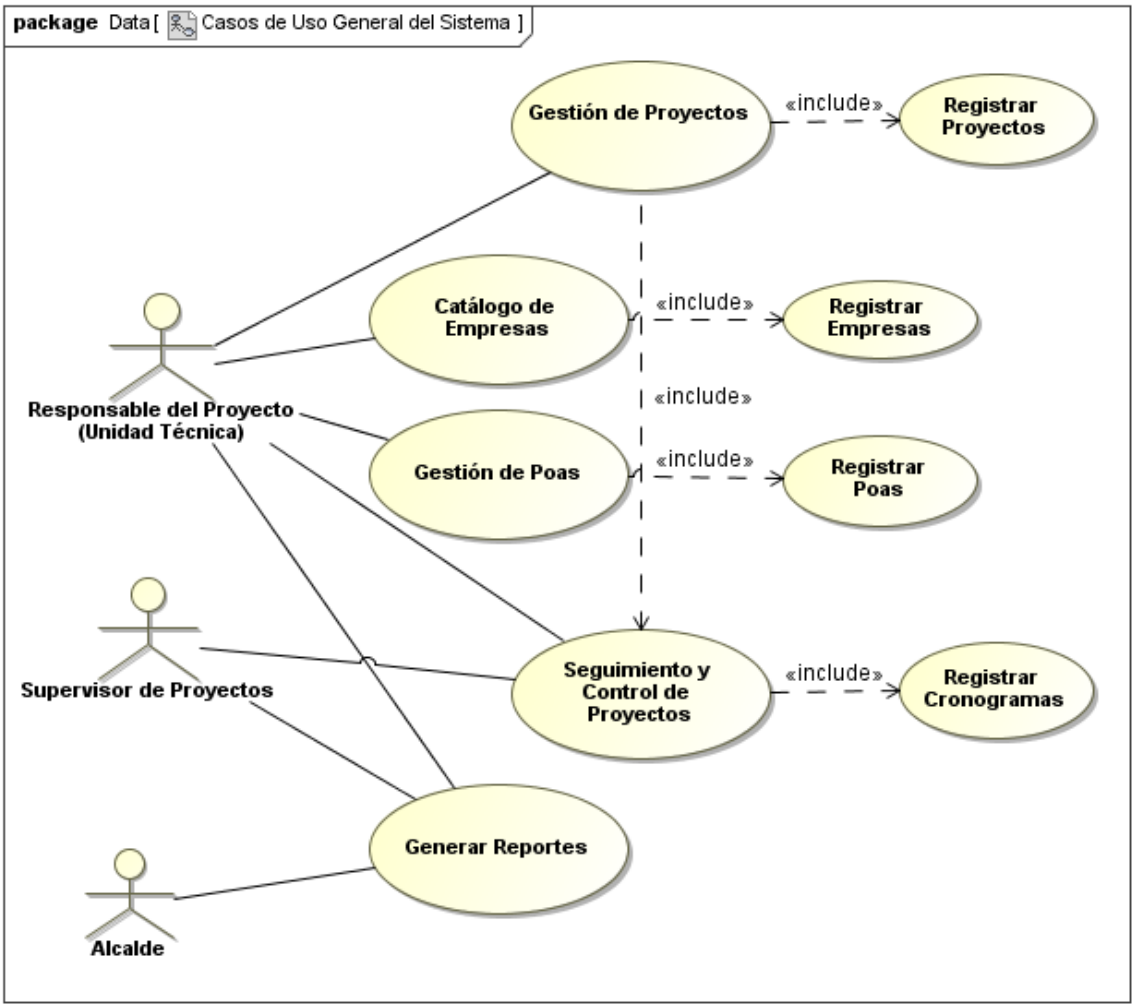
Durante la tercera iteración se desarrollaron las siguientes funcionalidades para el sistema:

- Módulo de Reportes.
- Pruebas del sistema.

3.3.4. Modelo de Casos de Uso General del Sistema

Los diagramas de casos de uso general del sistema sirven para especificar la comunicación y el comportamiento de un sistema mediante su interacción con los usuarios.

Figura 22: Diagrama de Caso de Uso General del Sistema

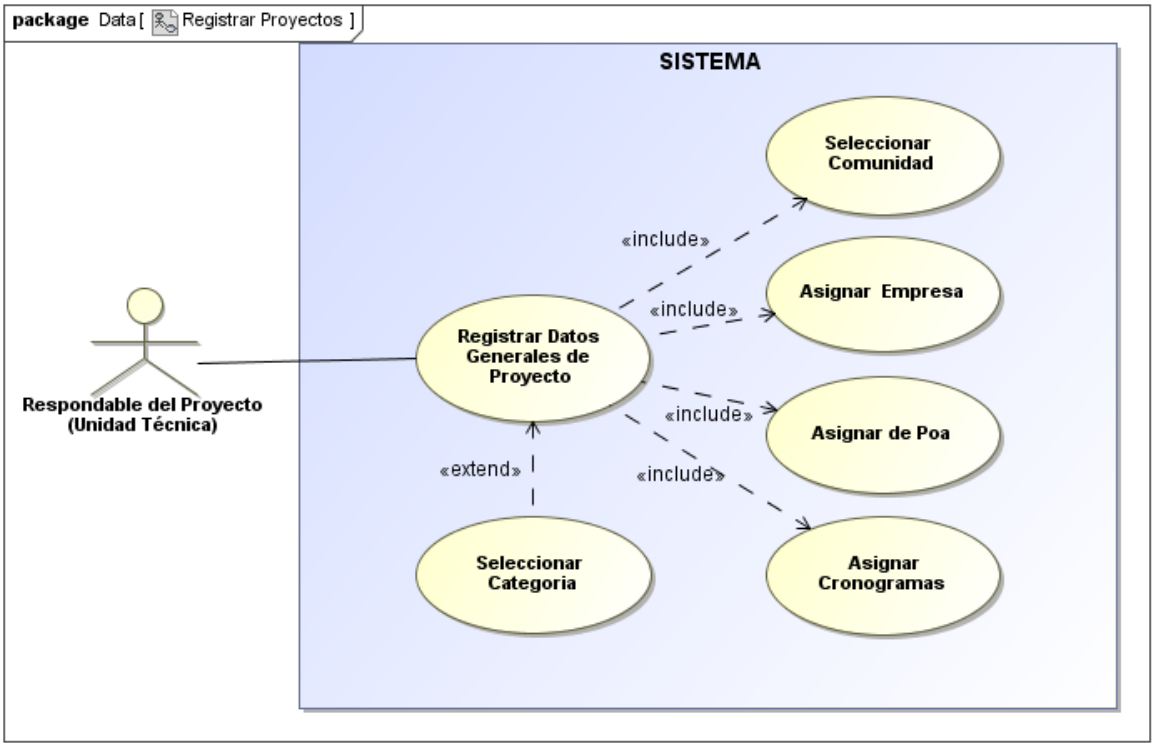


3.3.5. Modelo de Casos de Uso del Sistema

A continuación, se muestran los casos de uso del sistema Gestión de Proyectos agrupado por procesos, de las cuales solo mostraremos los más importantes.

- **Proceso - Registrar Proyectos**

Figura 23: Diagrama de Caso de Uso - Registrar Proyectos



La siguiente tabla es un resumen de las especificaciones del caso de uso Registro de Proyectos.

Tabla 12: Especificación de Caso de Uso - Registrar Proyecto

CASO DE USO	Registrar Proyectos
ACTORES	Responsable del Proyecto (Unidad Técnica).
DESCRIPCIÓN	El Gobierno Autónomo de Catacora, las Unidades Adjudicatarias y el Responsable de Proyectos

	(Unidad Técnica), después de haber formulado el proyecto, se procede al registro de los datos principales del proyecto a ejecutarse.
	FLUJO DE EVENTOS
FLUJO PRINCIPAL	1. El personal autorizado posee una cuenta de ingreso al sistema. 2. La pantalla muestra un login, donde el personal ingresa su usuario y su contraseña. 3. El sistema muestra el menú con diferentes opciones, dentro las cuales está gestión de proyectos. 4. El responsable del proyecto, antes de registrar un proyecto, necesita tener registrado la empresa que ejecutará la obra y la poa que le corresponde. 5. Si el responsable elige la opción proyecto, se despliega un formulario con los datos de todos los proyectos registrados. 6. El responsable tiene tres opciones en el formulario, agregar nuevo proyecto, editar y eliminar. 7. Si el responsable elige agregar un nuevo proyecto. En la nueva instancia emergentes puede llenar los datos generales del proyecto. 8. Guardar nuevo proyecto y actualizar la interfaz. 9. Si el responsable elige modificar un proyecto, en la instancia emergente el responsable puede cambiar los datos del proyecto. 10. Confirmar los cambios. 11. Si el responsable elige Eliminar, en la instancia emergente pedirá confirmación para eliminar.

	12. Cambiar a estado “INACTIVO”, y actualizar la interfaz gráfica.
ALTERNATIVAS	Si el Responsable de Proyectos desiste del nuevo registro, puede cancelar cerrar la ventana de nuevo, sin efectuar cambios en la base de datos.
PRE-CONDICIONES	El Responsable de Proyectos, debe ingresar sesión (login). Registrar la Empresa que ejecutará el proyecto. Registrar la Poa correspondiente al proyecto.
POST – CONDICIONES	Se actualiza la interfaz gráfica del Responsable de Proyectos.

Para el mejor entendimiento del proceso registro de proyectos, los diagramas de secuencia, estado y actividades que a continuación se muestran, explicaran las transiciones en forma gráfica.

Figura 24: Diagrama de Secuencia - Registrar Proyectos

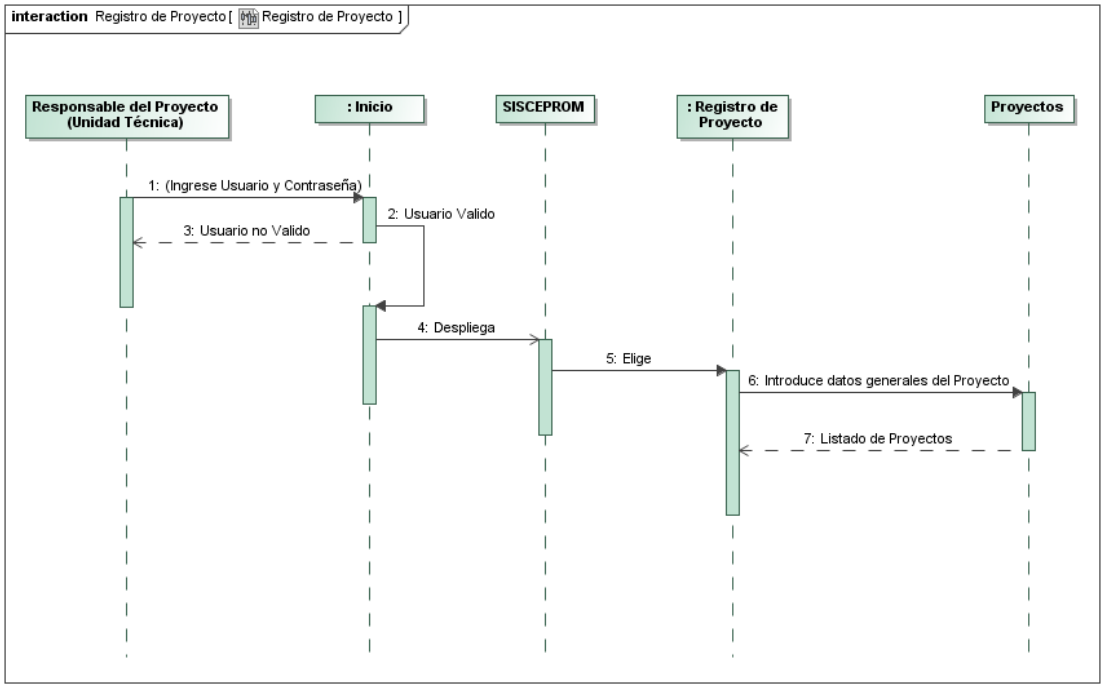


Figura 25: Diagrama de Estado - Registrar Proyectos

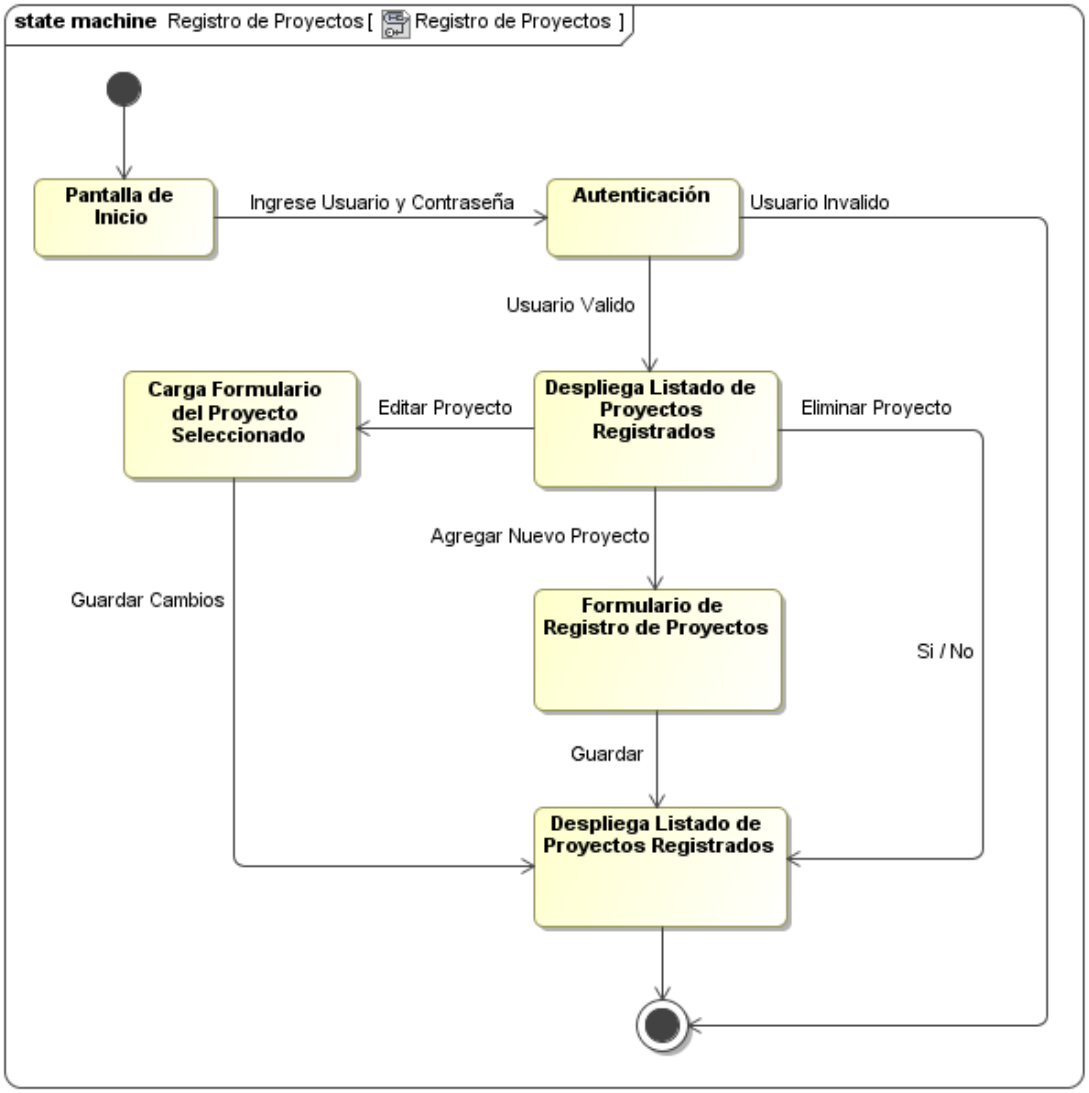
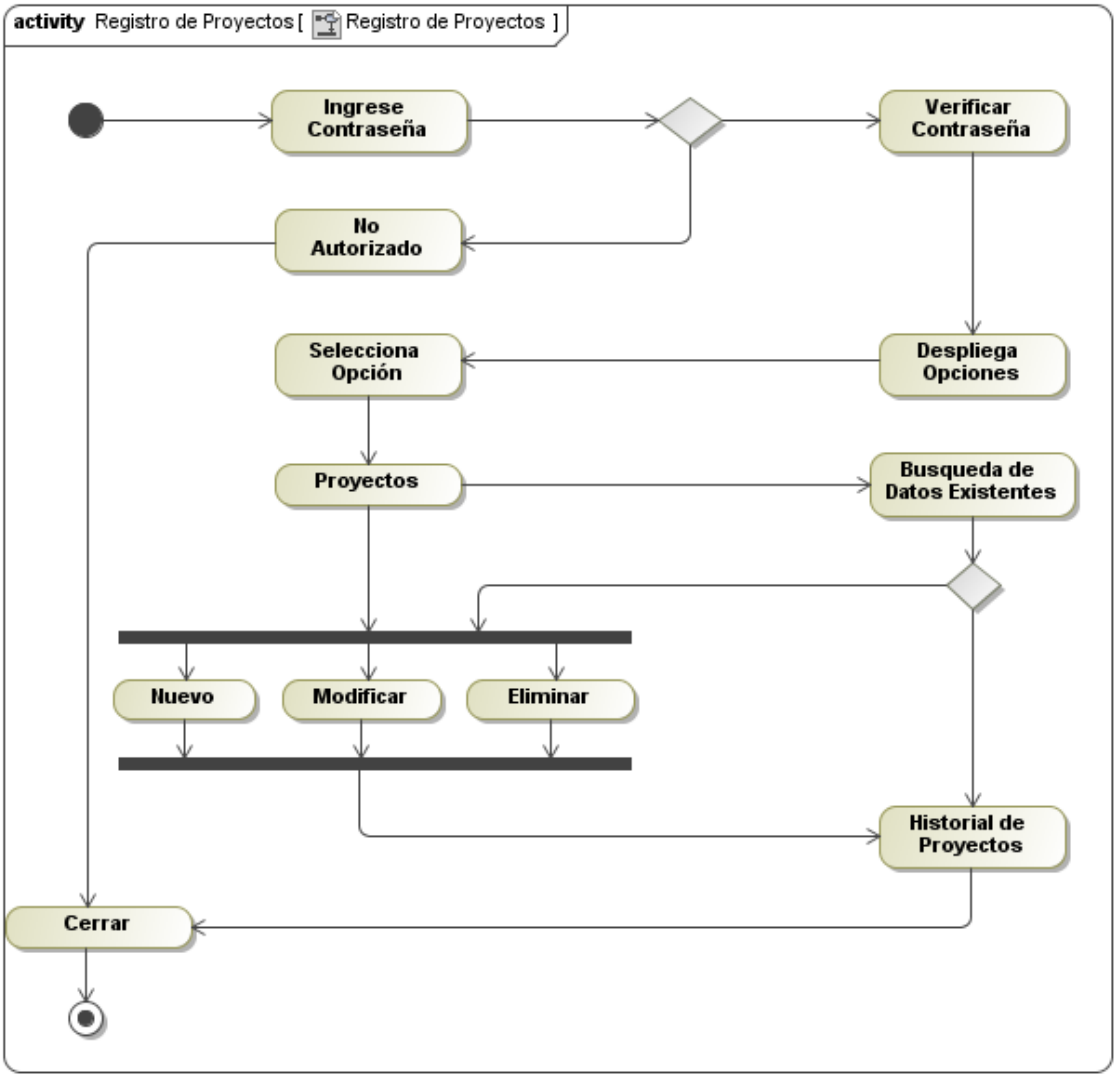
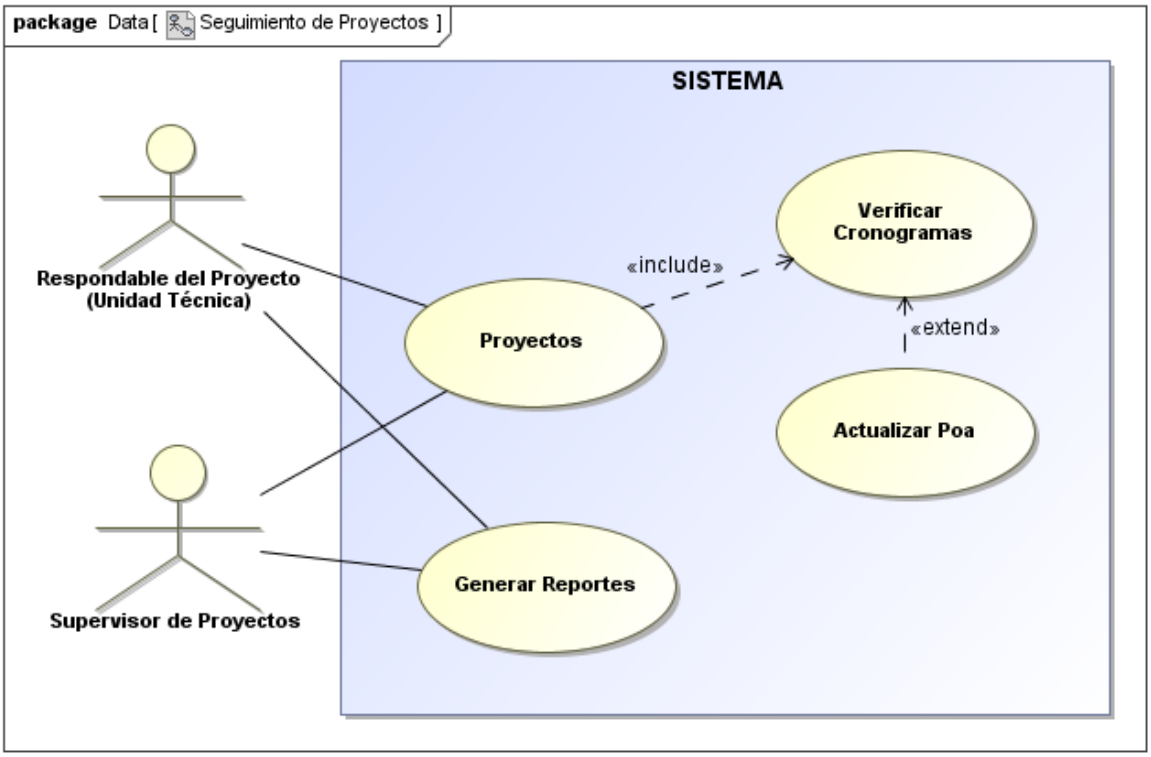


Figura 26: Diagrama de Actividad - Registrar Proyectos



- **Proceso – Seguimiento de Proyectos**

Figura 27: Diagrama de Caso de Uso - Seguimiento de Proyectos



La siguiente tabla es un resumen de las especificaciones del caso de uso Seguimiento de Proyectos.

Tabla 13: Especificación Caso de Uso - Seguimiento de Proyectos

CASO DE USO	Seguimiento de Proyectos
ACTORES	Responsable del Proyecto (Unidad Técnica) Supervisor de Proyectos
DESCRIPCIÓN	Permite al Responsable y Supervisor de Proyectos identificar los objetivos alcanzados y valorar los posibles problemas mediante el cronograma de actividades. Este proceso se realiza registrando los datos según el avance del proyecto.

FLUJO DE EVENTOS	
FLUJO PRINCIPAL	1. El personal autorizado posee una cuenta de ingreso al sistema. 2. La pantalla muestra un login, donde el personal ingresa su usuario y su contraseña. 3. El sistema muestra el menú con diferentes opciones, dentro las cuales está gestión de proyectos. 4. En pantalla emergente de proyectos existe una opción mostrar cronograma, el cual despliega opciones de agregar, editar y eliminar. 5. Si el responsable/supervisor elige la opción Agregar, se muestra una ventana emergente con el cronograma correspondiente al proyecto. 6. El responsable/supervisor, puede registrar los datos generales del cronograma. 7. Guardar y Actualizar. 8. Si el responsable/supervisor elige la opción editar, se muestra una ventana emergente cargando el cronograma del proyecto. 9. El responsable/supervisor, puede modificar según al informe del seguimiento que se da un determinado proyecto. 10. Guardar cambios y actualizar. 11. Si el responsable/supervisor elige la opción eliminar, se muestra una ventana emergente preguntando si desea eliminar o cancelar. 12. Actualizar.
ALTERNATIVAS	Si el Responsable de Proyectos desiste agregar, modificar o eliminar, puede cancelar y cerrar la ventana emergente, sin efectuar cambios en la base

	de datos.
PRE-CONDICIONES	El Responsable de Proyectos, debe ingresar sesión (login). Registrar los datos del Cronograma.
POST – CONDICIONES	Se actualiza la interfaz gráfica del Responsable/Supervisor de Proyectos.

Para el mejor entendimiento del proceso seguimiento de proyectos, los diagramas de secuencia, estado y actividades que a continuación se muestran, explicaran las transiciones en forma gráfica.

Figura 28: Diagrama de Secuencia - Seguimiento de Proyectos

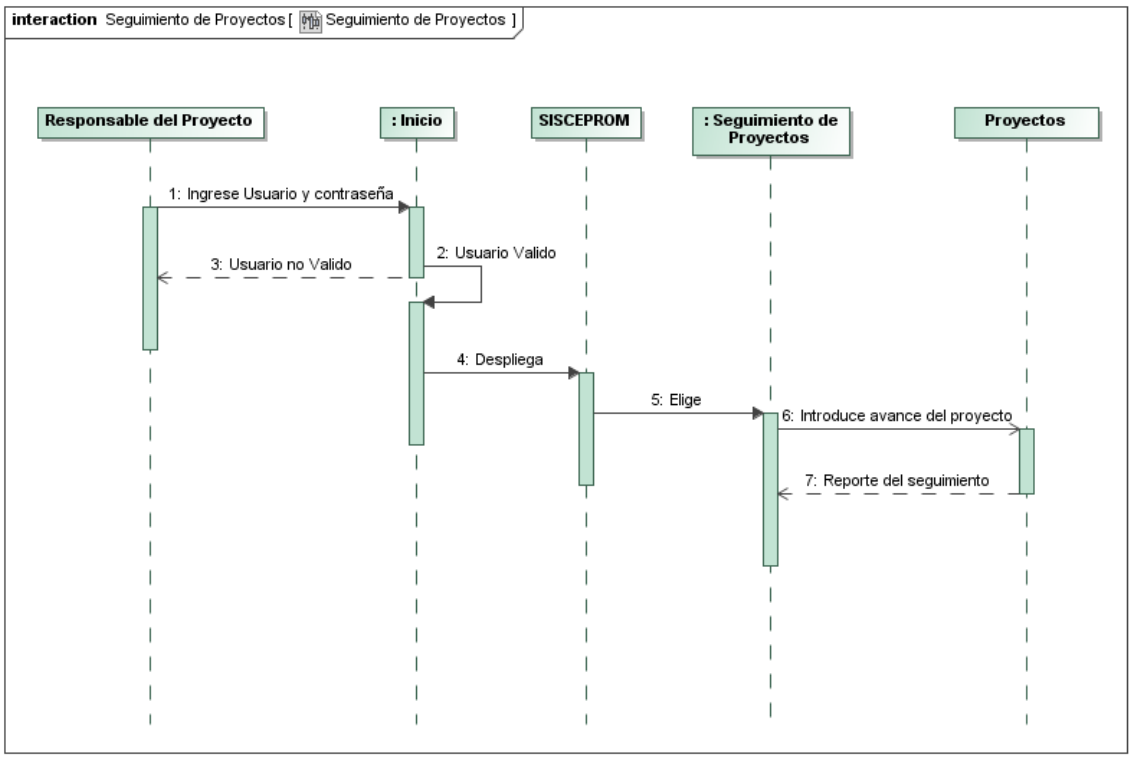


Figura 29: Diagrama de Estado - Seguimiento de Proyectos

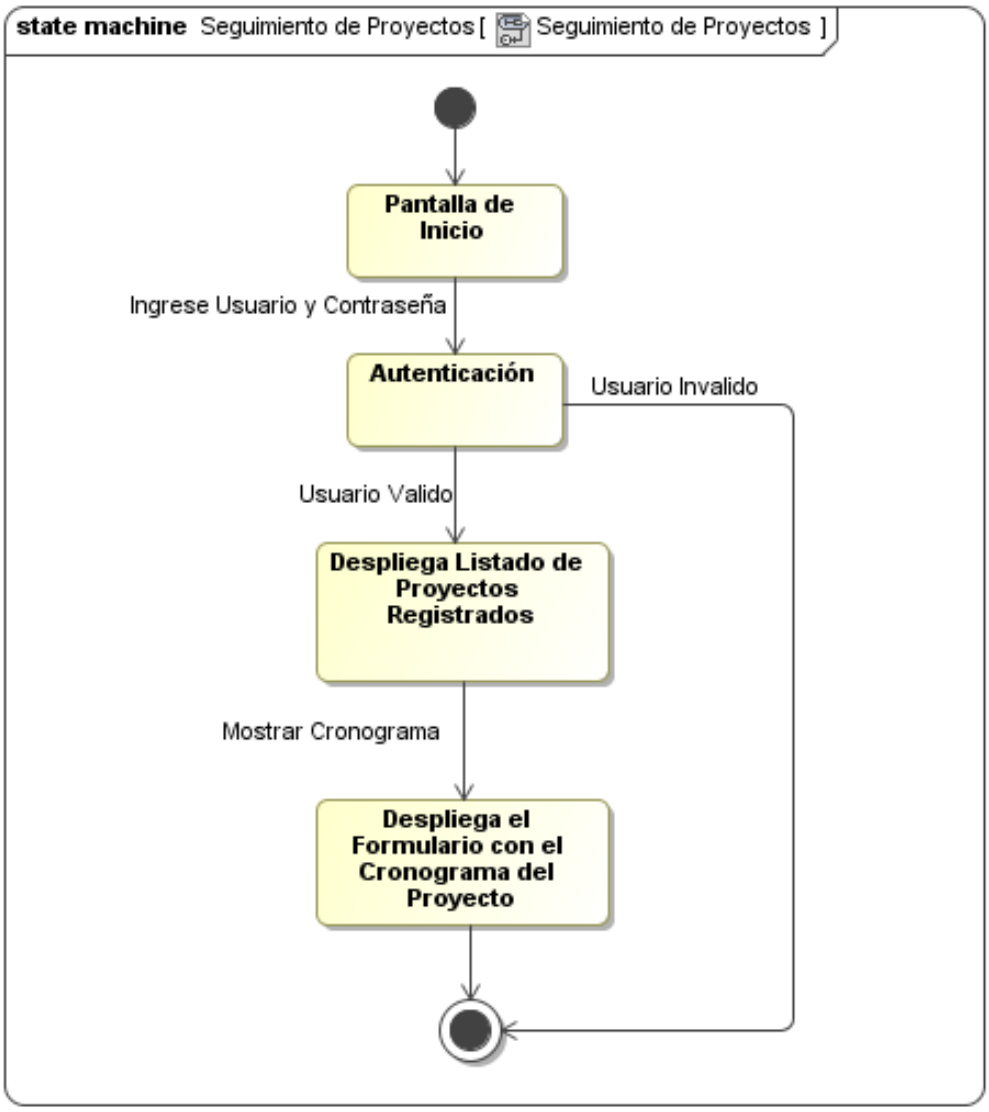
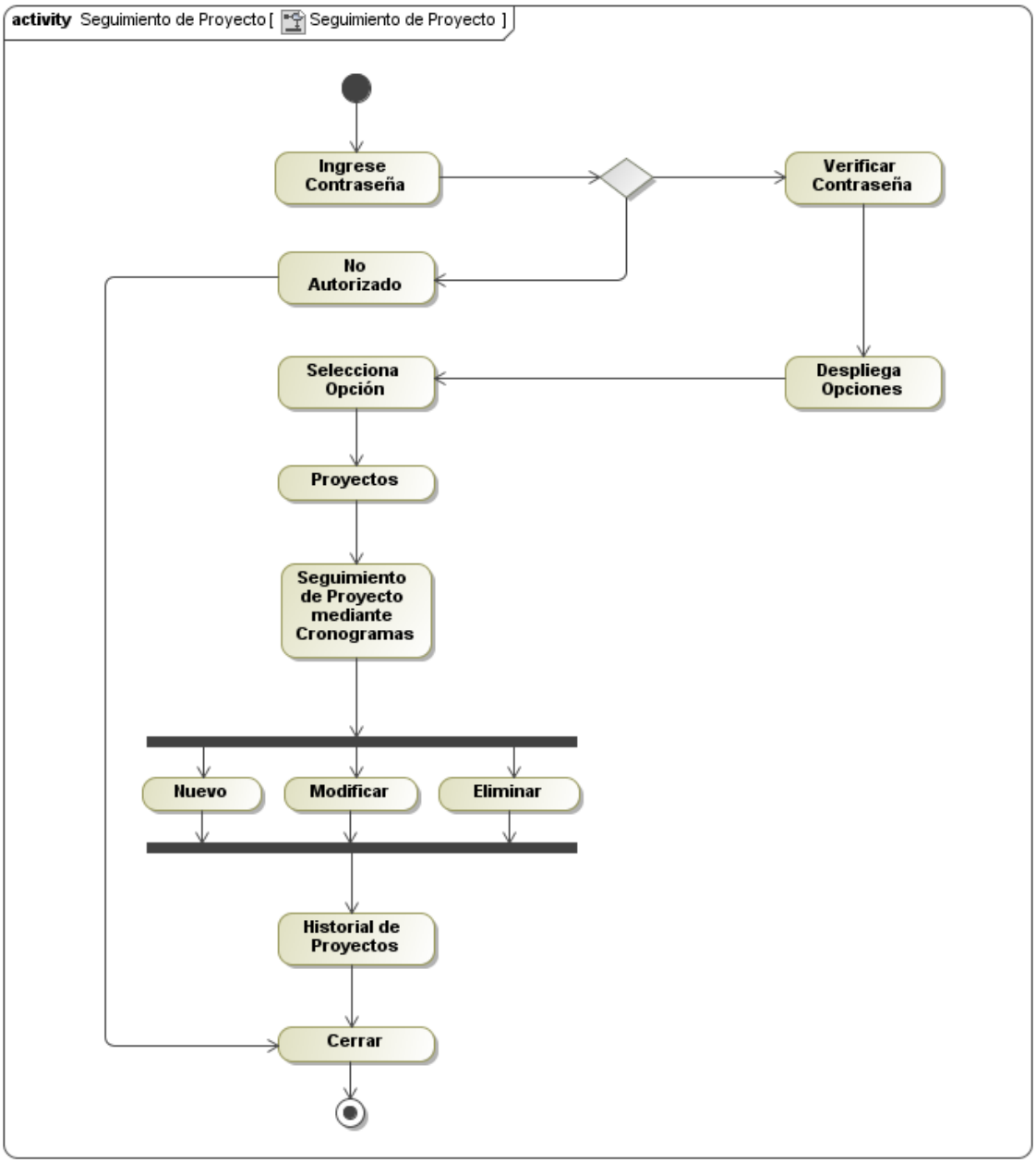
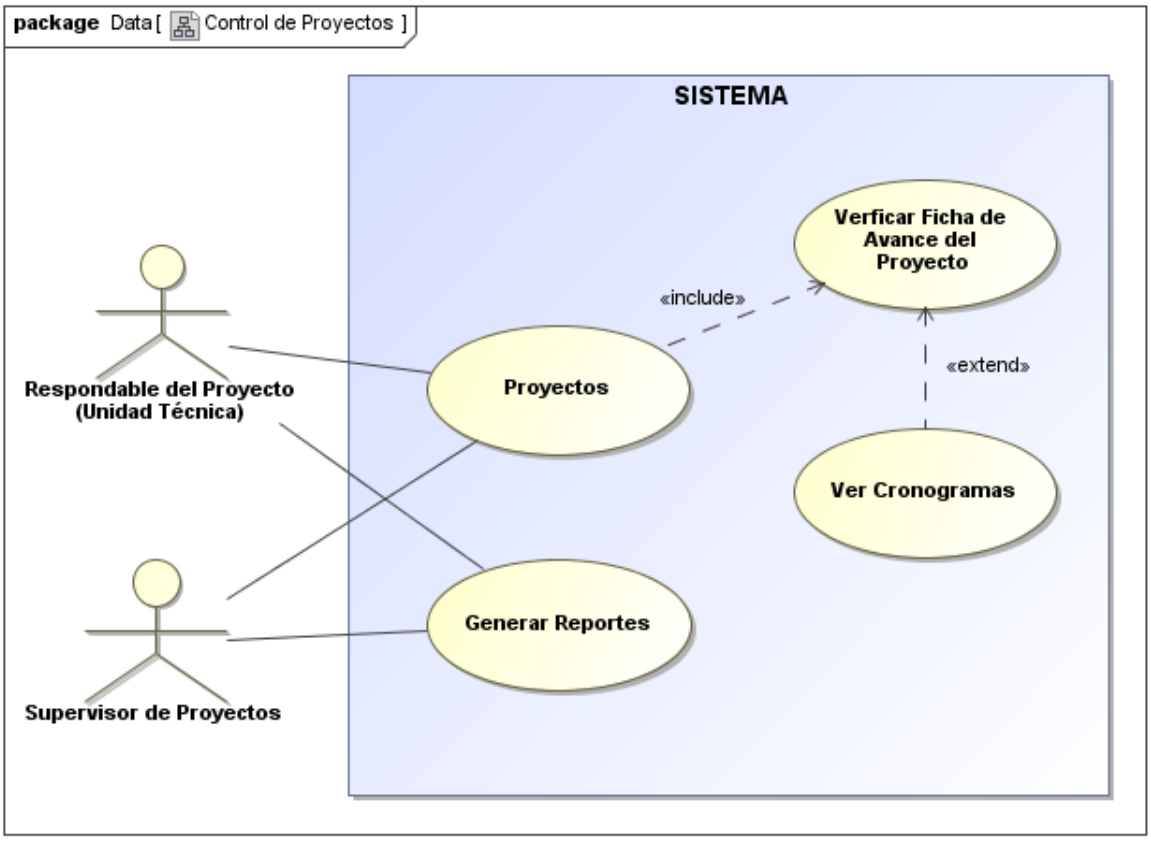


Figura 30: Diagrama de Actividad - Seguimiento de Proyectos



- **Proceso – Control de Proyectos**

Figura 31: Caso de Uso - Control de Proyectos



La siguiente tabla es un resumen de las especificaciones del caso de uso Control de Proyectos.

Tabla 14: Especificación de Caso de Uso - Control de Proyectos

CASO DE USO	Control de Proyectos
ACTORES	Responsable del Proyecto (Unidad Técnica) Supervisor de Proyectos
DESCRIPCIÓN	Permite al Responsable y Supervisor de Proyectos identificar los desfases de tiempo de cada proyecto que sufre algún retraso en la ejecución. Esta operación se realiza en función del tiempo y la

	comprobación de la información registrada en la ejecución física del proyecto.
FLUJO DE EVENTOS	
FLUJO PRINCIPAL	1. El personal autorizado posee una cuenta de ingreso al sistema. 2. La pantalla muestra un login, donde el personal ingresa su usuario y su contraseña. 3. El sistema muestra el menú con diferentes opciones, dentro las cuales está gestión de proyectos. 4. En la ventana emergente de listado de proyecto, donde se muestra toda la información. 5. El Responsable/Supervisor, realiza una comprobación de acuerdo al cronograma y la poa presionando la opción “ficha del proyecto”, para ver si todo está yendo acorde a las actividades programadas. 6. Cerrar ventana emergente. 7. Si el Responsable/Supervisor, ve algún desfase en el tiempo de ejecución en el proyecto, elige la opción editar cronograma o poa. 8. Registra los datos de desfase. 9. Guardar y actualizar.
ALTERNATIVAS	Si el Responsable/Supervisor de Proyectos desiste modificar, puede cancelar y cerrar la ventana emergente, sin efectuar cambios en la base de datos.
PRE-CONDICIONES	El Responsable/Supervisor de Proyectos, debe ingresar sesión (login).
POST – CONDICIONES	Se actualiza la interfaz gráfica del Responsable/Supervisor de Proyectos.

Para el mejor entendimiento del proceso control de proyectos, los diagramas de secuencia, estado y actividades que a continuación se muestran, explicaran las transiciones en forma gráfica.

Figura 32: Diagrama de Secuencia - Control de Proyectos

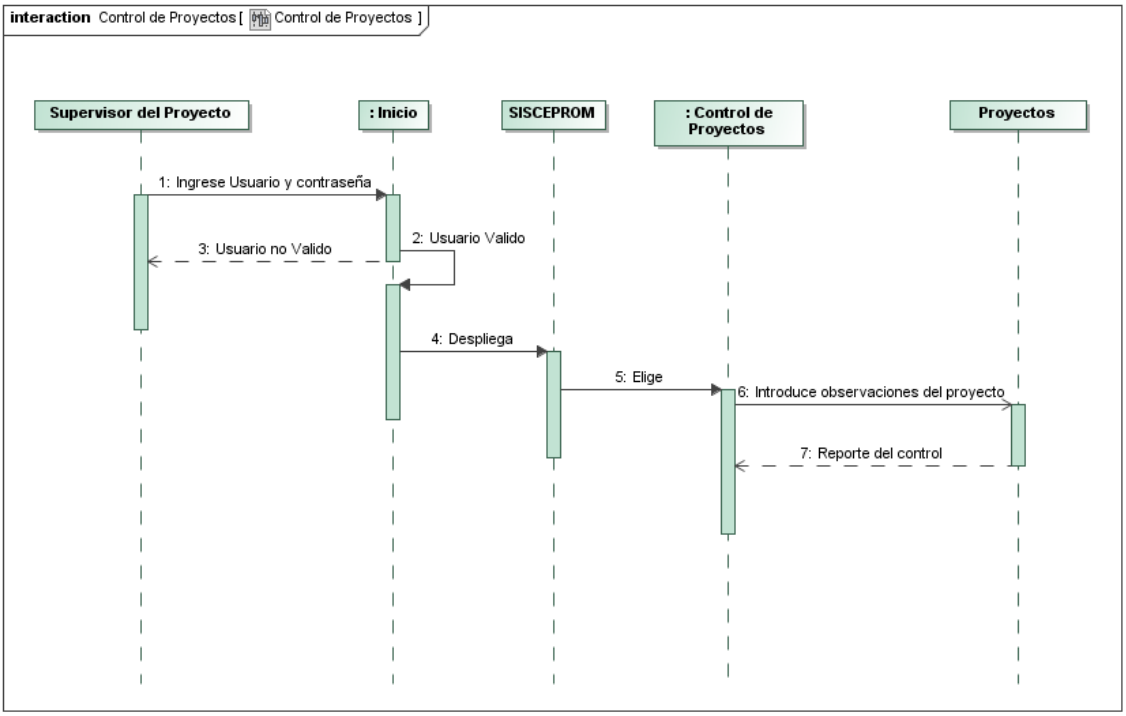


Figura 33: Diagrama de Estado - Control de Proyectos

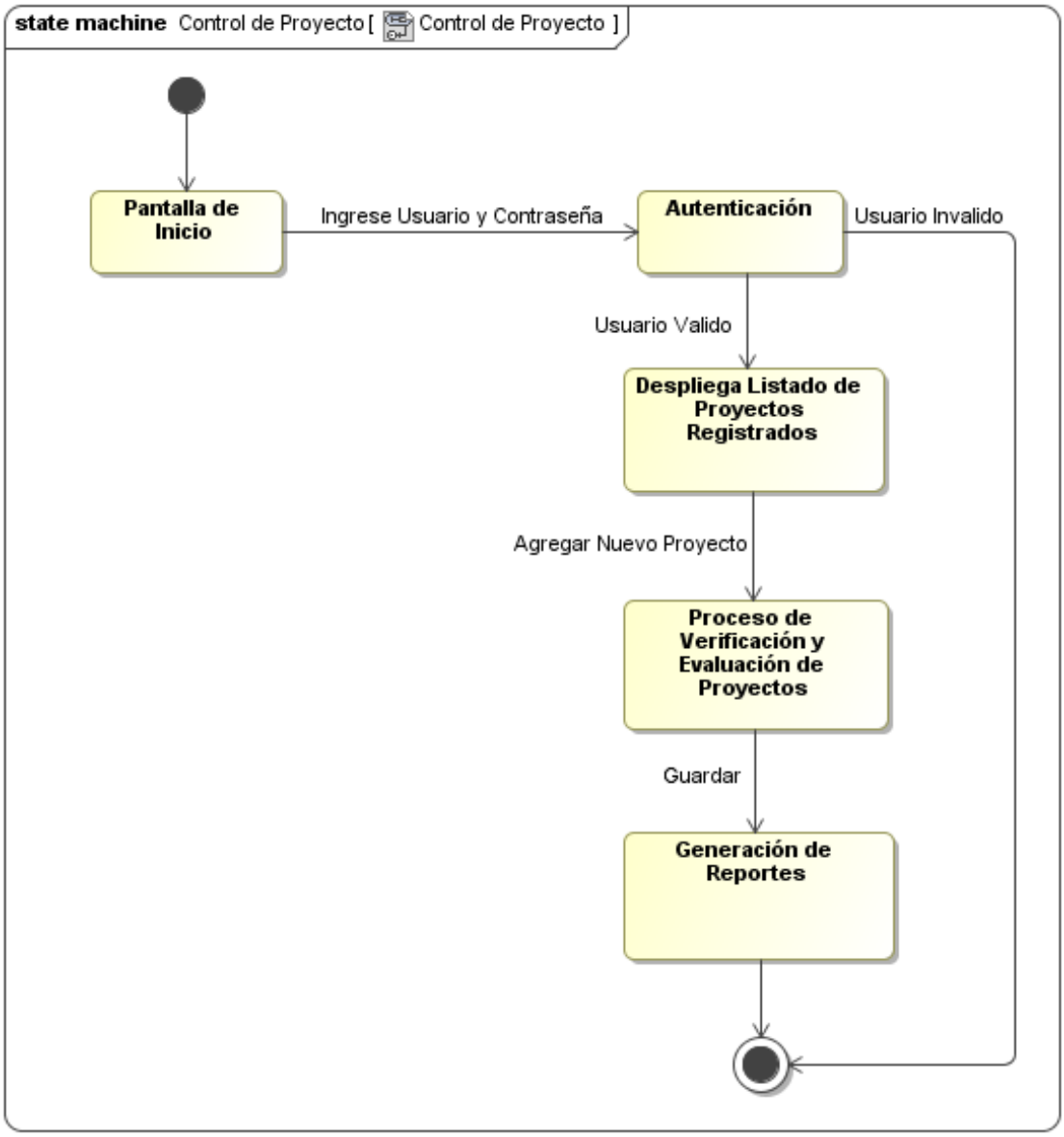
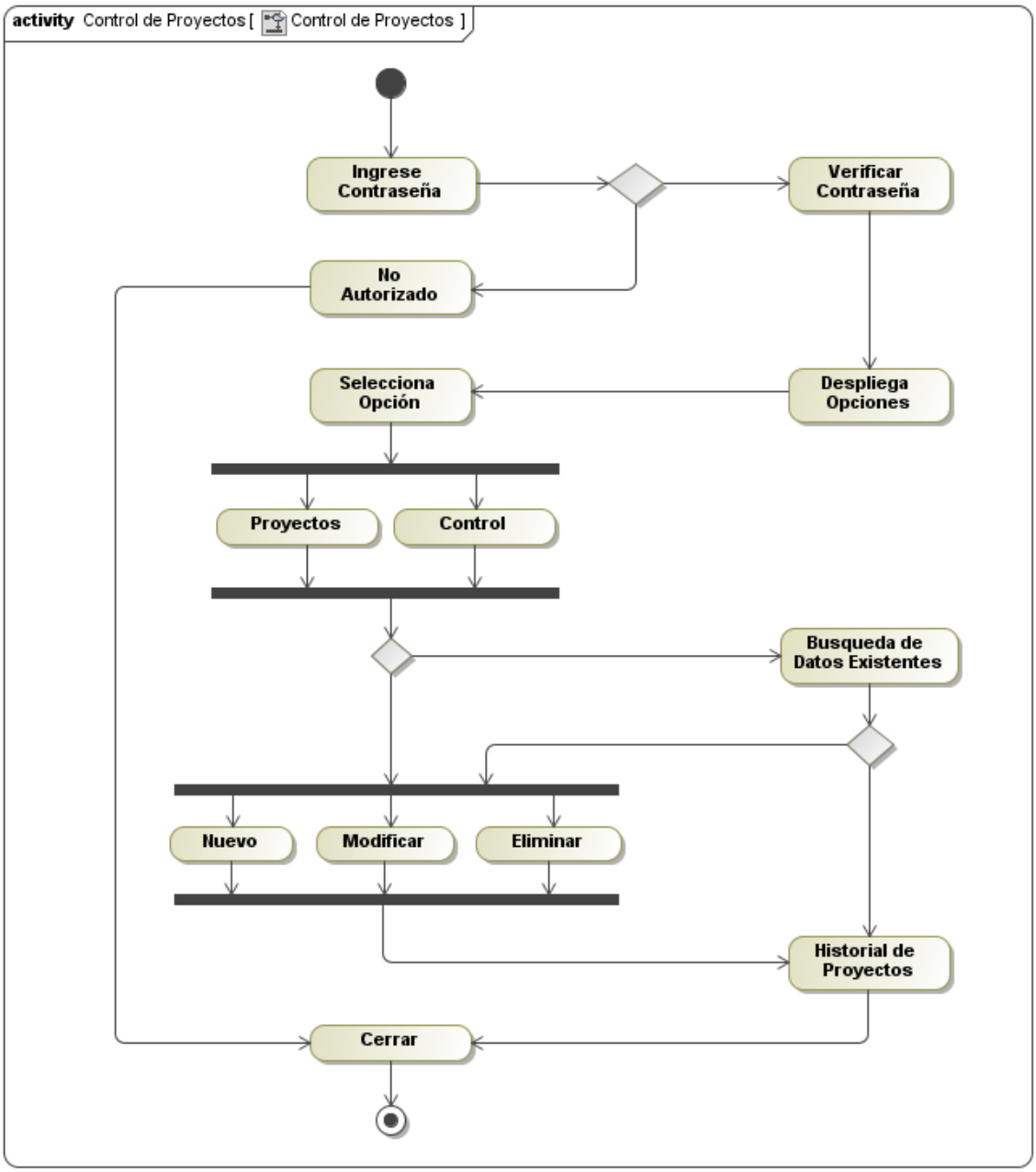
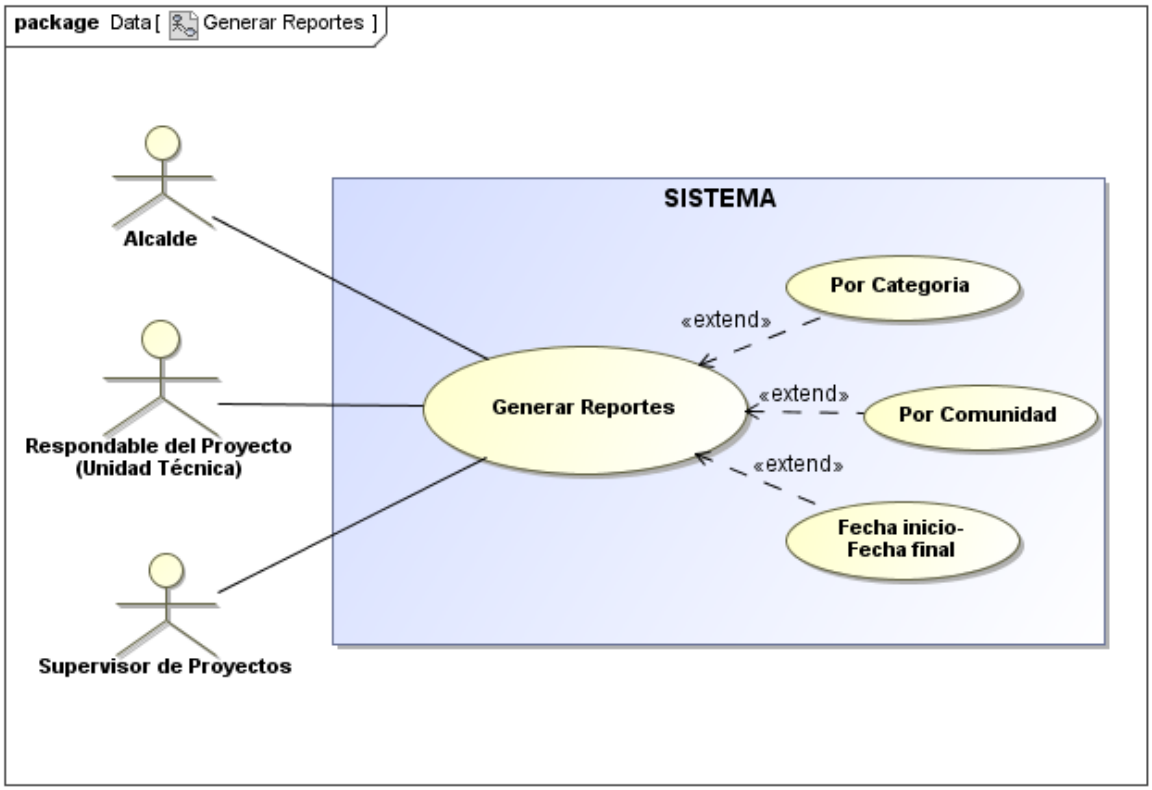


Figura 34: Diagrama de Actividad - Control de Proyectos



- **Proceso – Generar Reportes**

Figura 35: Diagrama de Caso de Uso - Generar de Reportes



La siguiente tabla es un resumen de las especificaciones del caso de uso Generación de Reportes.

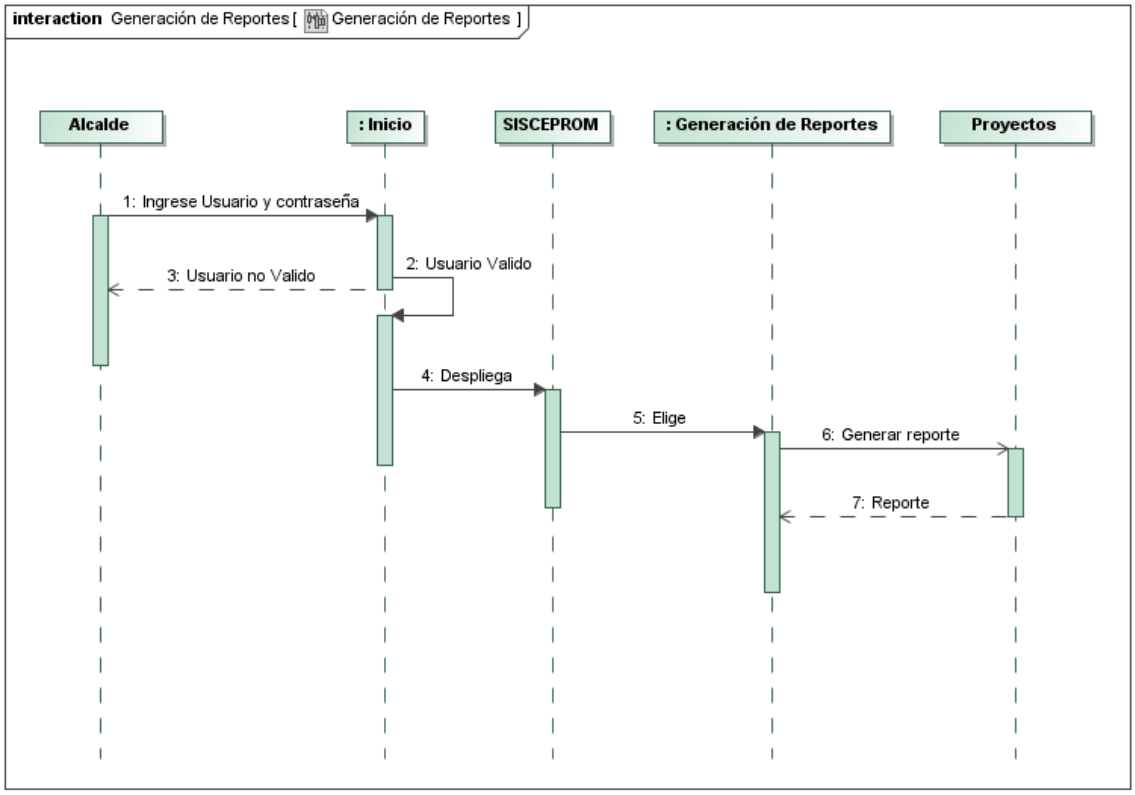
Tabla 15: Especificación Caso de Uso – Generación de Reportes

CASO DE USO	Generación de Reportes
ACTORES	Alcalde Responsable del Proyecto (Unidad Técnica) Supervisor de Proyectos
DESCRIPCIÓN	El Alcalde, el Responsable/Supervisor de proyectos, realizan esta labor a los proyectos que se están ejecutando o que ya han llegado a ser culminados y

	cerrados, para tal efecto todos obtienen dicha información para su posterior archivo físico.
FLUJO DE EVENTOS	
FLUJO PRINCIPAL	1. El personal autorizado posee una cuenta de ingreso al sistema. 2. La pantalla muestra un login, donde el personal ingresa su usuario y su contraseña. 3. El sistema muestra el menú con diferentes opciones, dentro las cuales está generar reportes. 4. El personal elige la opción que le ofrece el módulo reportes. 5. Generar reportes. 6. Cerrar y Actualizar la ventana emergente.
ALTERNATIVAS	Si el Alcalde, Responsable/Supervisor de Proyectos desiste realizar un reporte, puede cancelar y cerrar la ventana emergente.
PRE-CONDICIONES	El Alcalde, Responsable/Supervisor de Proyectos, debe ingresar sesión (login).
POST – CONDICIONES	Se actualiza la interfaz gráfica del Alcalde, Responsable/Supervisor de Proyectos.

Para el mejor entendimiento del proceso de generación de reportes, el diagrama de secuencia que a continuación se muestran, explicara las transiciones en forma gráfica.

Figura 36: Diagrama de Secuencia - Generar de Reportes



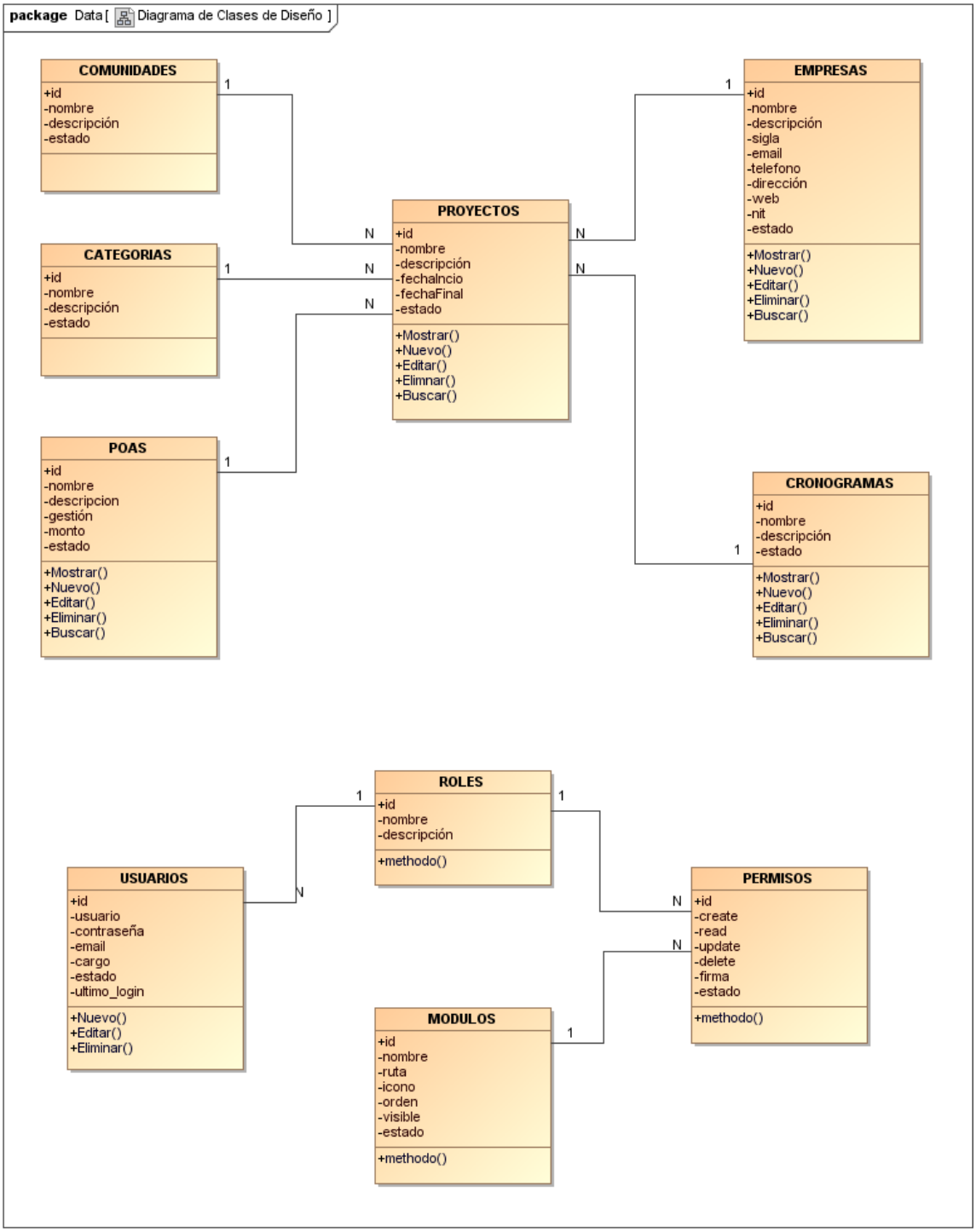
3.3.6. Análisis y Diseño de la Base de Datos

Durante el diseño modelamos el sistema y su arquitectura para que soporte a los requerimientos funcionales. El artefacto que se genera en esta actividad es el modelo de diseño, compuesto por clases de diseño principalmente que muestren un conjunto de objetos con atributos operacionales, métodos y relaciones.

3.3.6. 1. Modelo de Diseño

En el siguiente diagrama de Clases. Es el diagrama principal para el análisis y diseño estático. Un diagrama de clases presenta las clases y objetos del sistema con sus relaciones estructurales y de herencia. La definición de clase u objeto incluye definiciones para atributos y métodos.

Figura 37: Diagrama de Clases de Diseño



3.3.6. 2. Modelo Relacional

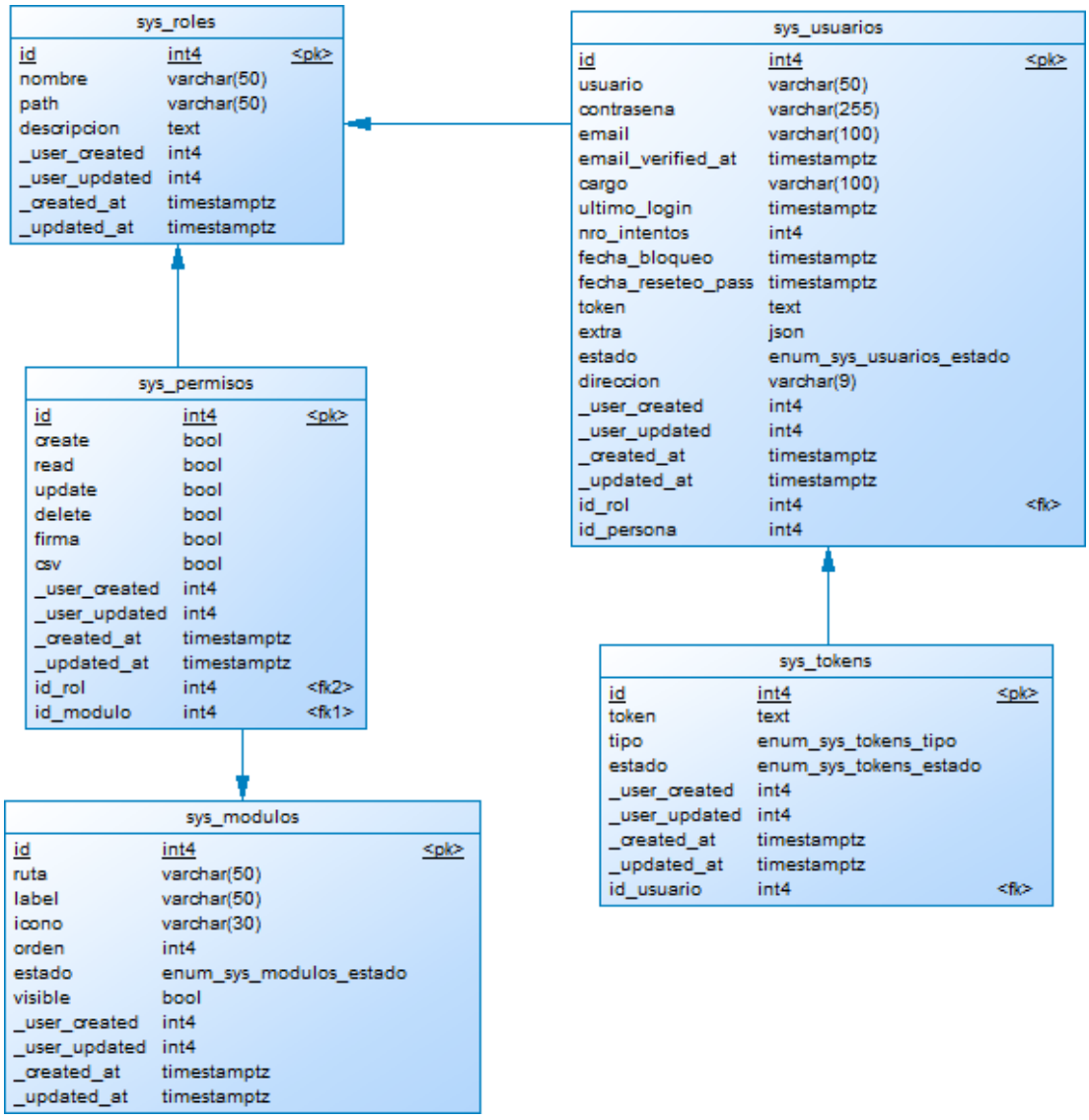
Se basa en una abstracción del mundo real, el diseño de la base de datos es la parte más importante durante la actividad de diseño, ya que este es el que debe soportar las actividades previas a los requisitos, análisis y diseño.

El modelo relacional para la gestión de una base de datos es un modelo de datos basado en la lógica de predicados y en la teoría de conjuntos. Es el modelo más utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente. La ventaja del modelo relacional es que los datos se almacenan, al menos conceptualmente, de un modo en que los usuarios entienden con mayor facilidad. Asimismo, mantiene información sobre las propias características de la base de datos.

Para esta actividad hay varias herramientas que te permiten hacer ingeniería inversa, entre ellas está PowerDesigner una de las más utilizadas, entre sus principales características tenemos la propiedad de generar un modelo relacional de una base de datos determinada a partir de la ingeniería inversa.

En la figura siguiente se muestra el modelo relaciona físico de la base de datos del sistema.

Figura 38: Modelo Relacional Físico de la Base de Datos del Sistema


sys_roles
id int4 <pk>
nombre varchar(50)
path varchar(50)
descripcion text
_user_created int4
_user_updated int4
_created_at timestamptz
_updated_at timestamptz
sys_usuarios
id int4 <pk>
usuario varchar(50)
contrasena varchar(255)
email varchar(100)
email_verified_at timestamptz
cargo varchar(100)
ultimo_login timestamptz
nro_intentos int4
fecha_bloqueo timestamptz
fecha_reseteo_pass timestamptz
token text
extra json
estado enum_sys_usuarios_estado
direccion varchar(9)
_user_created int4
_user_updated int4
_created_at timestamptz
_updated_at timestamptz
id_rol int4 <fk>
id_persona int4
sys_permisos
id int4 <pk>
create bool
read bool
update bool
delete bool
firma bool
csv bool
_user_created int4
_user_updated int4
_created_at timestamptz
_updated_at timestamptz
id_rol int4 <fk2>
id_modulo int4 <fk1>
sys_tokens
id int4 <pk>
token text
tipo enum_sys_tokens_tipo
estado enum_sys_tokens_estado
_user_created int4
_user_updated int4
_created_at timestamptz
_updated_at timestamptz
id_usuario int4 <fk>
sys_modulos
id int4 <pk>
ruta varchar(50)
label varchar(50)
icono varchar(30)
orden int4
estado enum_sys_modulos_estado
visible bool
_user_created int4
_user_updated int4
_created_at timestamptz
_updated_at timestamptz

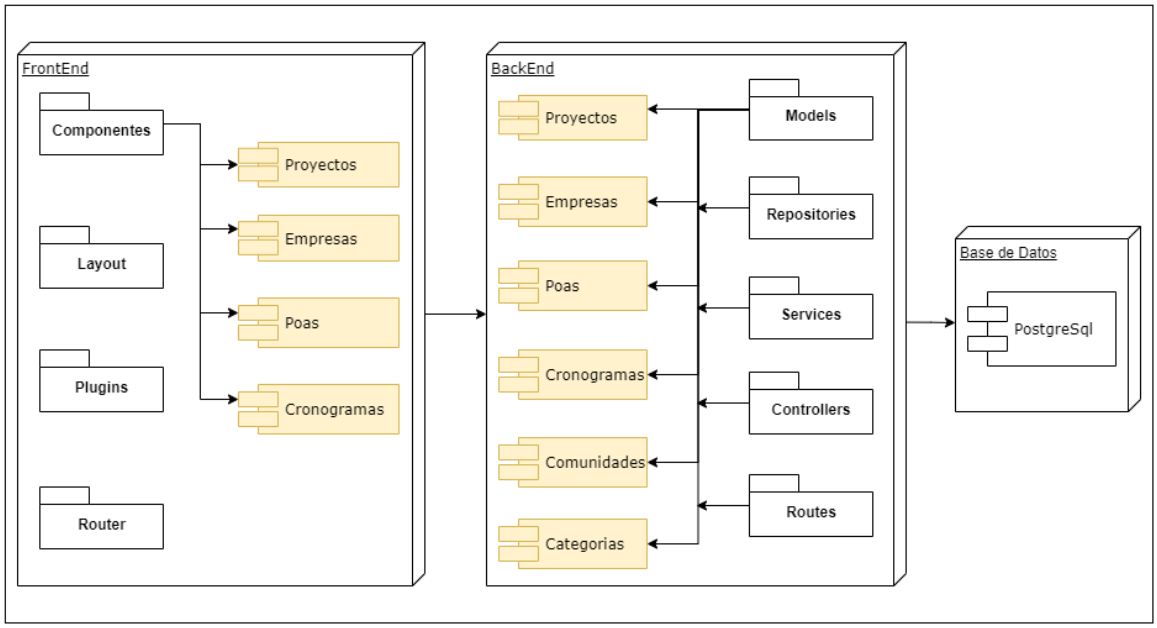
3.3.7. Implementación

3.3.7.1. Modelo de Implementación

El Modelo de Implementación es comprendido por un conjunto de componentes y módulos que constituyen la composición física de la implementación del sistema.

El siguiente diagrama implementación, muestra las dependencias entre el Frontend, Backend y la Base de Datos del sistema.

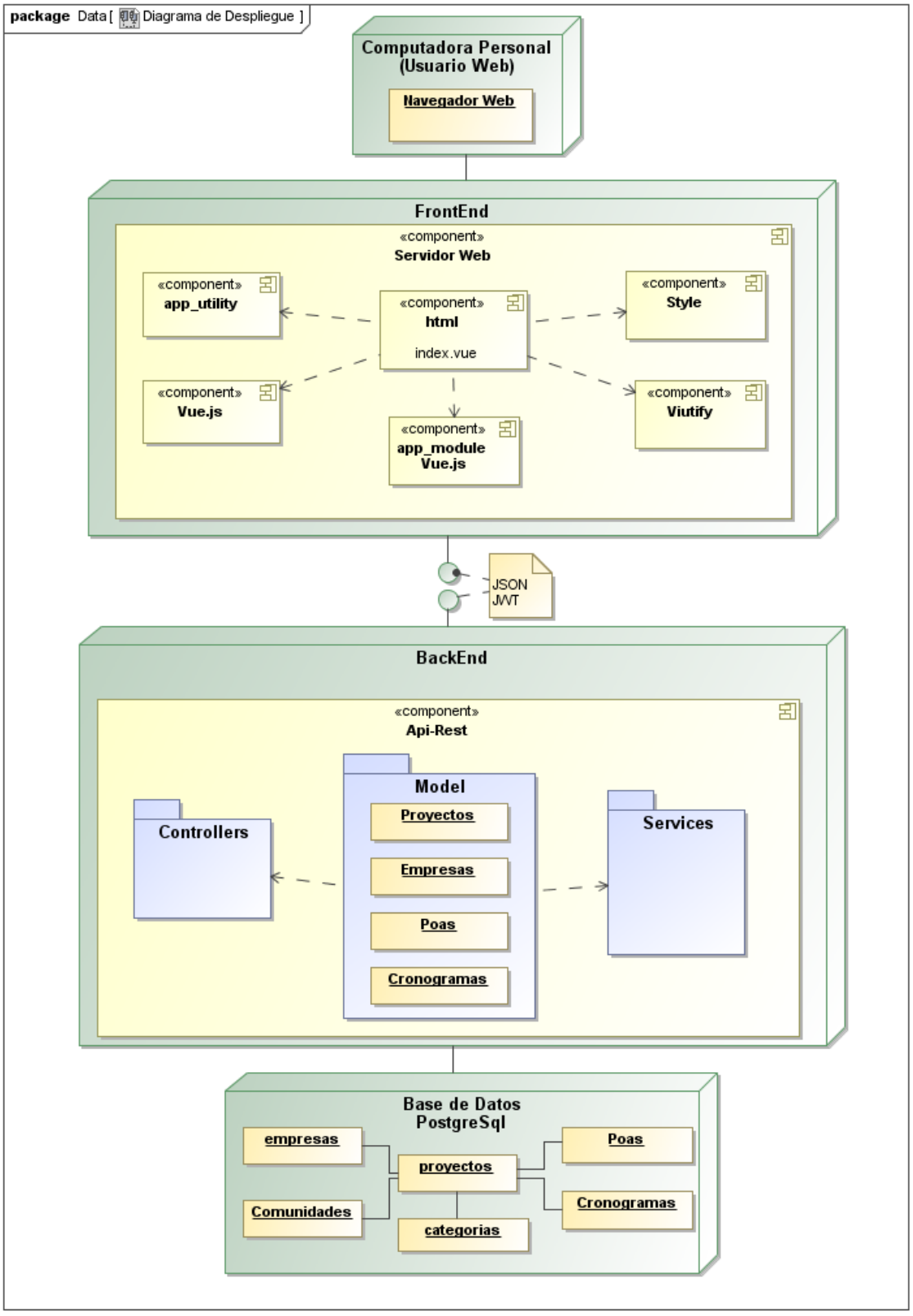
Figura 39: Diagrama de Paquetes del Sistema



3.3.7.2. Modelo de Despliegue

El siguiente modelo de despliegue nos permitirá modelar la disposición física del sistema, también nos mostrará las conexiones entre el hardware y las relaciones entre componentes.

Figura 40: Diagrama de Despliegue del Sistema



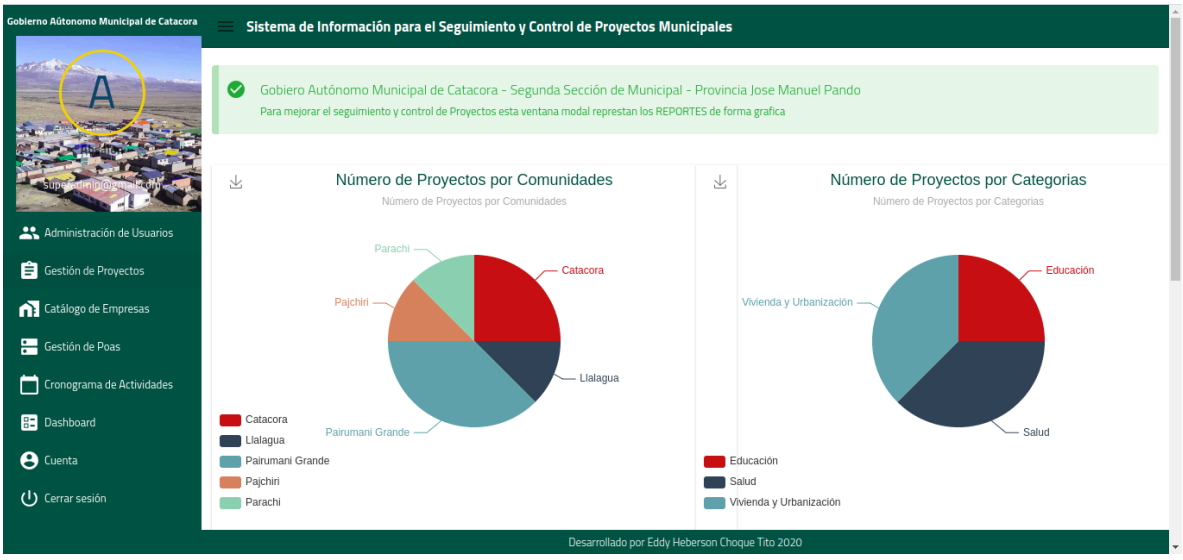
3.3.7.3. Interfaces Gráficas del Sistema

Pantalla de interfaz de autenticación o inicio de sesión de los usuarios registrados al sistema, que verifica el usuario y contraseña respectivamente.

Figura 41: Inicio de Sesión o Autenticación del Sistema

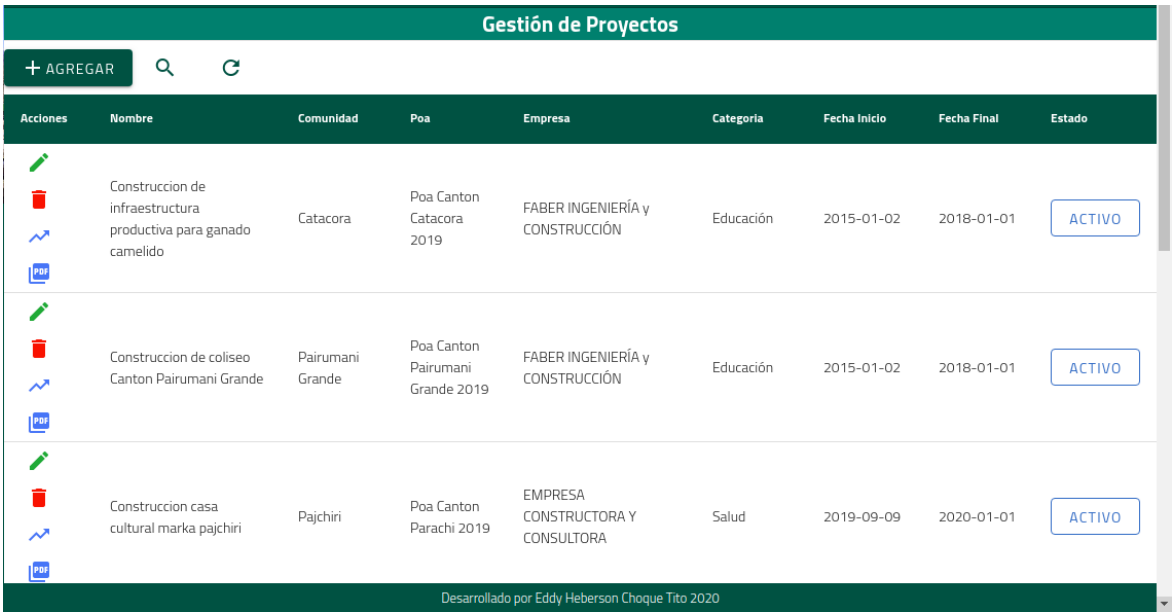
Una vez que inicie sesión, se mostrará una ventana emergente con los menús correspondientes al sistema, donde se visualiza los módulos de Administración de Usuarios, Gestión de Proyectos, Catálogo de Empresas, Gestión de Poas Cronogramas de Actividades.

Figura 42: Venta Principal del Sistema



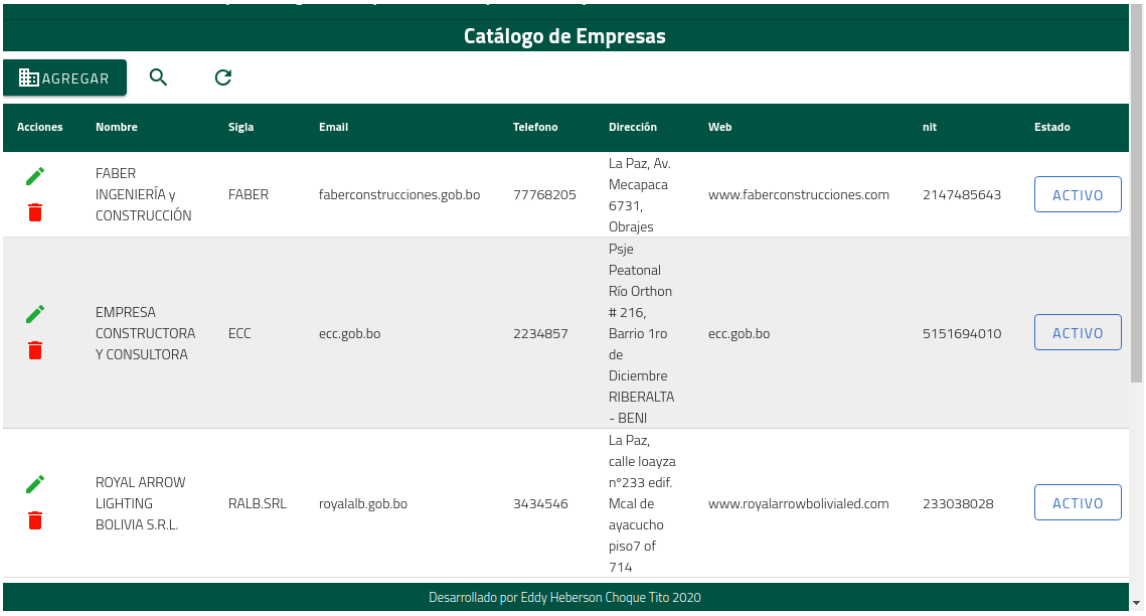
El módulo proyectos, visualiza el listado de todos los proyectos, también visualiza un botón que nos permite agregar un nuevo proyecto, además en cada proyecto listado nos da la opción de editar, eliminar, realizar seguimiento al proyecto y ver ficha del proyecto.

Figura 43: Módulo Gestión de Proyectos



El menú empresas, visualiza primero el listado de todas las empresas registradas, segundo también visualiza un botón que nos permite agregar una nueva empresa, además cada empresa listada, nos da la opción de editar y eliminar.

Figura 44: Módulo Catálogo de Empresas

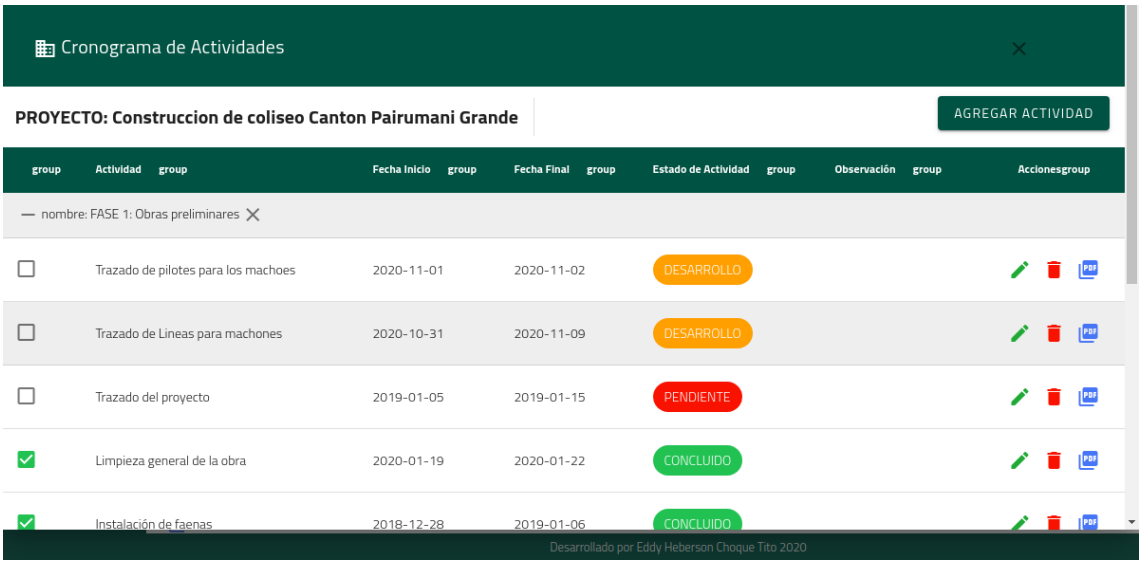


El menú poas, visualiza el listado de todas las poas registradas, también visualiza un botón que nos permite agregar una nueva poa, además cada poa listada, nos da la opción de editar y eliminar.

Figura 45: Módulo de Gestión de Poas

El menú cronogramas, visualiza el listado de todos los cronogramas registrados, también visualiza un botón que nos permite agregar un nuevo cronograma, además cada cronograma listada, nos da la opción de editar y eliminar.

Figura 46: Seguimiento y Control por Cronogramas



3.4. POST - GAME (DESPUÉS DEL DESARROLLO)

Durante esta etapa post-game, se realizará las actividades de prueba a los módulos desarrollados en sistema de información.

3.4.1. Pruebas del Sistema

3.4.1.1. Pruebas de Caja Blanca

Con este tipo de prueba, se garantiza que se ejerciten por lo menos una vez todos los caminos independientes de cada módulo, además de todas las decisiones lógicas para asegurar su validez.

Para la aplicación de la siguiente prueba se toma en cuenta un solo módulo, el cual describe la funcionalidad de los otros.

Figura 47: Diagrama de Flujo – Registro de Proyectos

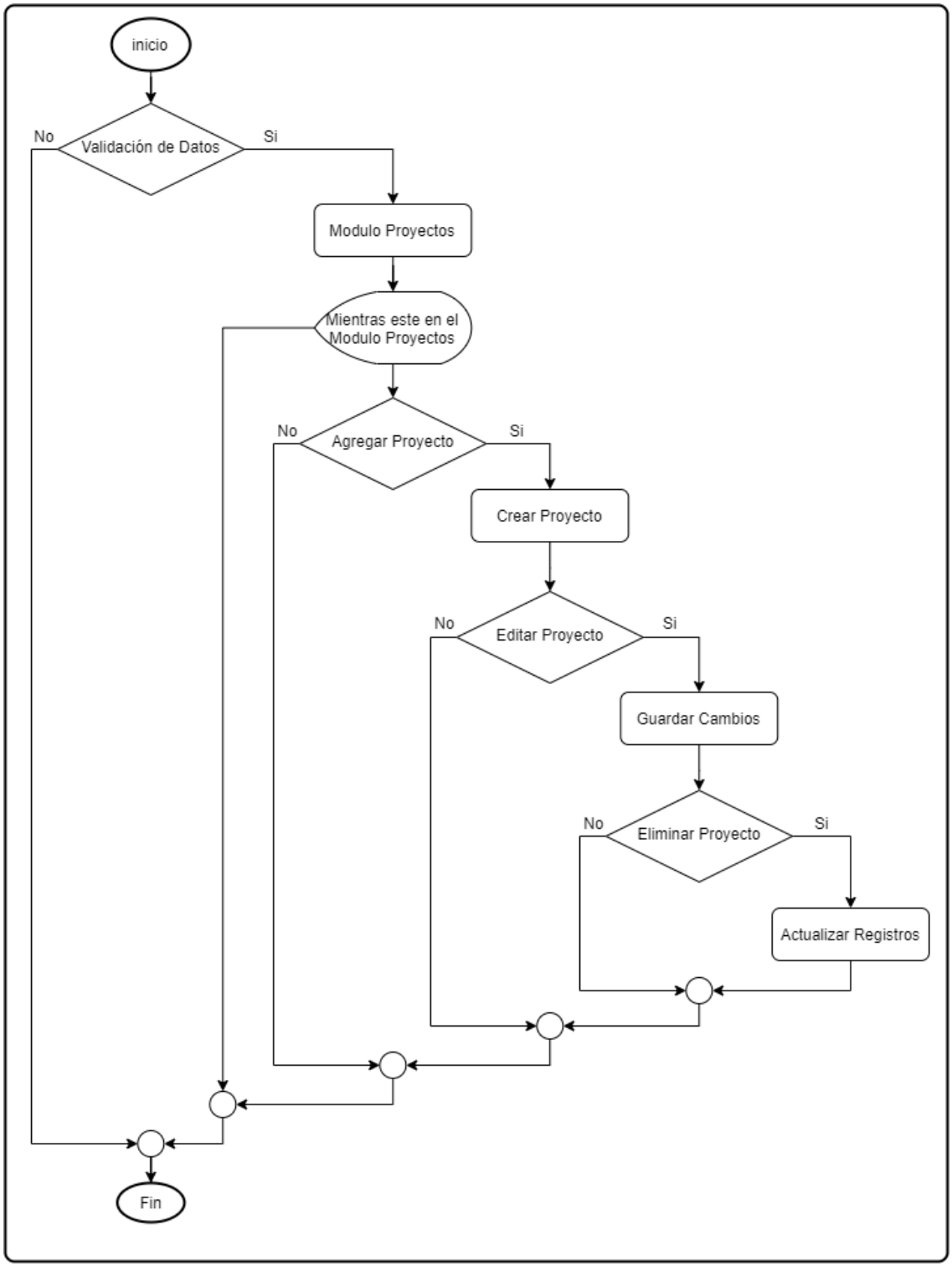
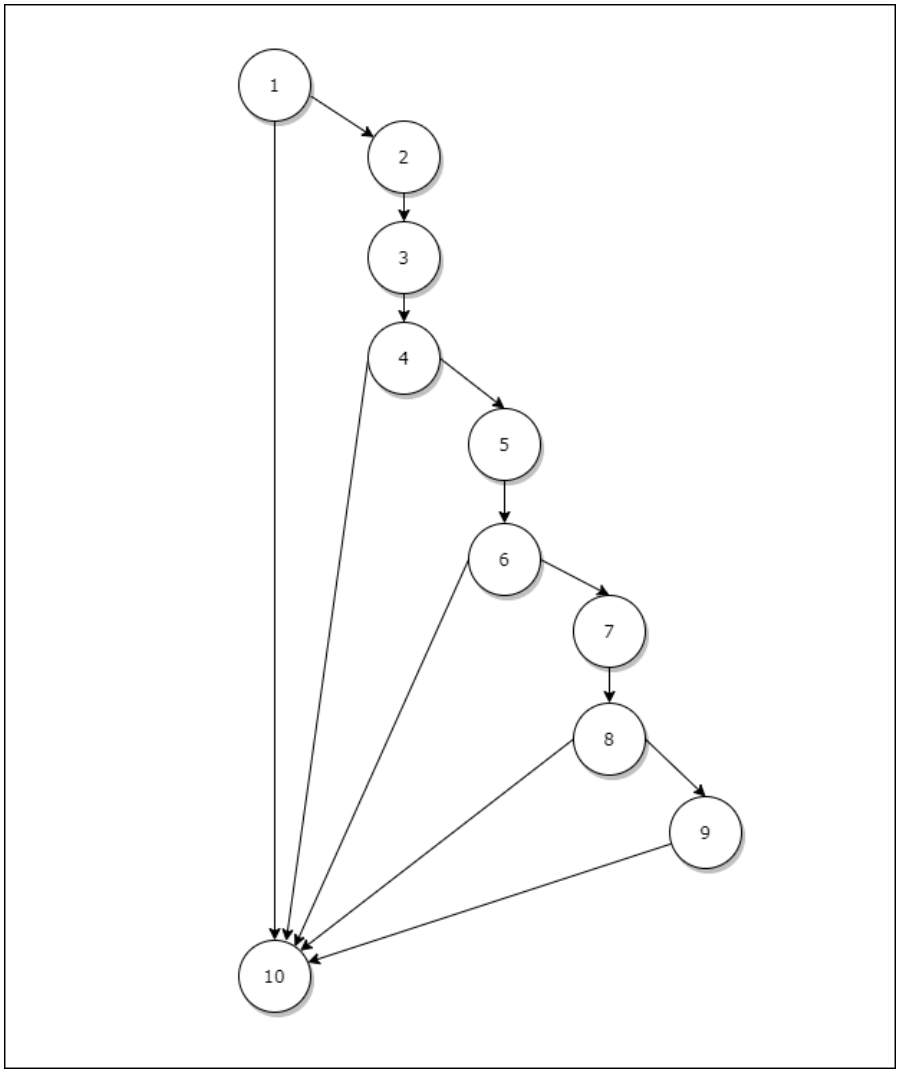


Figura 48: Grafo de Métrica - Registro de Proyectos



Según la figura 46 tenemos:

Número de nodos = 10

Número de aristas = 13

El grafo que se genera a través del diagrama de flujo, nos ayuda a determinar la complejidad ciclomática, para ello utilizaremos la siguiente fórmula:

$$V(G) = \text{A} - \text{N} + 2$$

Dónde:

A: Es el número de aristas (13 Aristas)
N: Es el número de nodos (10 Nodos)

Por tanto, reemplazando valores tenemos:

$$V(G) = 13 - 10 + 2$$

$$\boldsymbol{V(G) = 5}$$

Con el resultado de la complejidad V(G) = 5, nos indica que son 5 los casos de prueba que se deben de ejecutar, estos caminos independientes son los siguientes:

Camino 1: 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 10
Camino 2: 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 10
Camino 3: 1 – 2 – 3 – 4 – 5 – 6 – 10
Camino 4: 1 – 2 – 3 – 4 – 10
Camino 5: 1 – 10

3.4.1.2. Pruebas de Caja Negra

La prueba de caja negra consiste en probar cada una de las funciones del sistema que fue desarrollado.

Con este tipo de prueba se debe buscar que las funciones sean operativas, además se debe agotar al sistema de tal manera buscar la mayor cantidad de errores.

Son pruebas sobre la interfaz del software. A continuación, se muestran algunas pruebas relevantes.

- **Prueba de autentificación**

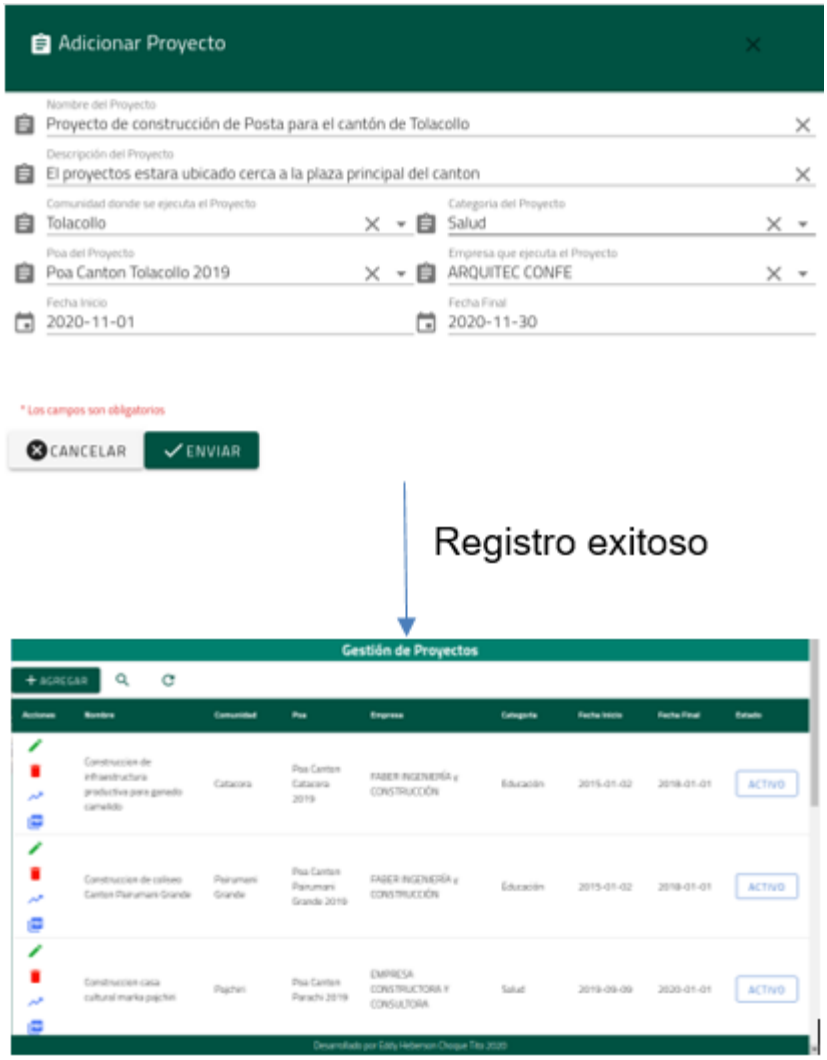
Si el usuario que desea ingresar al sistema es el correcto, desplegará la interfaz gráfica de ingreso, mostrando todos los menús correspondientes al rol, caso contrario se mostrará un mensaje de acceso denegado.

Figura 49: Prueba de Autenticación - Login

- **Validación de un Formulario**

En la siguiente figura se muestra la validación del formulario correspondiente módulo proyectos. A la culminación de un buen registro se despliega en el formulario el proyecto adicionado con todos los datos, caso contrario se mostrarán mensajes de alerta describiendo el error de la información correspondiente.

Figura 50: Validación de Formulario Adicionar Proyecto



CAPÍTULO IV

4. CALIDAD Y SEGURIDAD DE SOFTWARE

4. CALIDAD Y SEGURIDAD DEL SOFTWARE

4.1. INTRODUCCIÓN

La calidad y seguridad del software es una de las etapas fundamentales para la implementación del mismo, ya que en este proceso se evalúa el funcionamiento adecuado, de acuerdo a lo planificado y establecido, siguiendo procedimientos para la seguridad del mismo.

4.2. CALIDAD DEL SOFTWARE

La calidad del sistema web será definida en base a un conjunto de propiedades inherentes a un producto, que le confieren capacidad para satisfacer necesidades implícitas o explícitas.

Existen diversos estándares y modelos, el presente proyecto usará la técnica ISO/IEC 9126 que a continuación se mostrará.

4.2.1. Técnica ISO/IEC 9126

El objetivo principal de esta técnica es alcanzar la calidad necesaria para satisfacer las necesidades del cliente. Se evalúan dos ámbitos: el producto final y los procesos.

La calidad según esta norma ISO 9126 puede ser pedida de acuerdo a los siguientes factores:

- Funcionalidad
- Confiabilidad
- Mantenibilidad
- Portabilidad
- Usabilidad

4.2.1.1. Funcionalidad

El Punto Función es una métrica orientada a la función del software y del proceso por el cual se desarrolla. Se centra en la funcionalidad o utilidad del programa, los puntos de función se calculan realizando una serie de actividades, comenzando por determinar los siguientes números:

- **Números de entrada de usuario:** Se cuenta cada entrada del usuario que proporcione al software diferentes datos orientados a la aplicación.
- **Número de salida del usuario:** En este contexto las salidas se refieren a informes, pantalla, mensajes de error, etc.
- **Números de peticiones al usuario:** Una petición está definida como una entrada interactiva que resulta de la generación de algún tipo de respuesta en forma de salida interactiva.
- **Número de archivos:** se cuenta cada archivo maestro lógico,
- **Número de interfaces externas:** se cuentan todas las interfaces legibles por la máquina, por ejemplo: archivos de datos, en cinta o discos que son utilizados para transmitir información a otro sistema.

De acuerdo a lo mencionado tenemos los siguientes resultados:

Tabla 16: Entrada de datos para el Cálculo de Funcionalidad

Entradas de usuario	25
Salidas de usuario	15
Consultas de usuario	20
Número de archivos	17
Interfaces externas	2

Los puntos de función se calculan rellenando la tabla 17 con los datos obtenidos en la tabla 16, considerando un factor de ponderación medio.

Tabla 17: Entradas para el Cálculo de Funcionalidad

Parámetros de medición	Cuenta		Factor de ponderación medio		Total
Número de entradas de usuario.	25	*	4	=	100
Número de salidas de usuario.	15	*	5	=	75
Número de consultas de usuario.	20	*	4	=	80
Número de archivos	17	*	10	=	170
Número de interfaces	2	*	7	=	14
CUENTA TOTAL					439

La relación que permite calcular los puntos de función es la siguiente:

$$\mathrm{PF} = \text{Cuenta Total} * (\text{Grado de confiabilidad} + \text{tasa de error} * \sum \mathrm{F}i)$$

Donde:

- **PF** = Medida de Funcionalidad.
- **Cuenta Total** = es la suma de valor de las entradas, salidas, peticiones, interfaces externas y archivos.
- **Grado de Confiabilidad** = Es la confiabilidad estimada del sistema.
- **Tasa de error** = Probabilidad subjetiva estimada del dominio de la información, este error estimado es de 1%.
- **Fi** = Son valores de ajuste de complejidad que toman los valores de la tabla 18 y que dan respuesta a las preguntas de la tabla 18.

Tabla 18: Ajustes de Complejidad del Punto Función

ESCALA	Sin importancia	Incidental	Moderado	Medio	Significativo	Esencial
Factor	0	1	2	3	4	5
1. ¿Requiere el sistema copias de seguridad y recuperación fiable?						X
2. ¿Se requiere comunicación de datos?			X			
3. ¿Existe funciones de procesamiento distribuido?						X
4. ¿Es crítico el rendimiento?				X		
5. ¿Será ejecutado el sistema en un entorno operativo existente y frecuentemente utilizado?						X
6. ¿Requiere el sistema entrada de datos interactivos?					X	
7. ¿Requiere la entrada de datos interactivos que las transiciones de entrada se lleven a cabo sobre múltiples o variadas operaciones?			X			
8. ¿Se actualiza los archivos de forma interactiva?				X		
9. ¿Son complejas las entradas, las salidas, los archivos o peticiones?			X			

10. ¿Es complejo el procesamiento interno?			X			
11. ¿Se ha diseñado el código para ser reutilizable?						X
12. ¿Están incluidos en el diseño la conversión y la instalación?						X
13. ¿Se ha diseñado el sistema para soportar múltiples instalaciones en diferentes plataformas?			X			
14. ¿Se ha diseñado la aplicación para facilitar los cambios y para ser fácilmente utilizados por el usuario?						X
TOTAL	$\sum(Fi) = 50$					

A continuación, reemplazamos valores y calculamos el valor de PF:

$$\mathrm{PF} = 439 * (0.65 + 0.01 * 50)$$
$$\mathrm{PF} = 504.85$$

Si consideramos el máximo valor de ajuste de complejidad como $\sum(Fi) = 65$ se tiene:

$$\mathrm{PF} = 439 * (0.65 + 0.01 * 65)$$
$$\mathrm{PF} = 570.7$$

Entonces si $\sum(Fi)$ es considerada como el 100%, la relación obtenida entre los puntos será:

$$\text{FUNCIONALIDAD } = \frac{504.85}{570.7} \times 100$$

$$\text{FUNCIONALIDAD } = 88.46\ \%$$

Por lo tanto, la funcionalidad o utilidad del sistema es del 88% lo que significa que el software se desenvuelve satisfactoriamente.

4.2.1.2. Confiabilidad

La confiabilidad es la capacidad del software de mantener su nivel de performance o rendimiento, bajo las condiciones establecidas por un periodo de tiempo.

La confiabilidad del Software se la mide de la siguiente manera:

$$R(\text{t}) \ = \ (\text{Funcionalidad}) \ * \ e^{-\lambda t}$$

Dónde:
R(t)= Confiabilidad del Sistema
Funcionalidad = 0.88
λ = 0.01 (es decir 1 error en cada 6 ejecuciones)
t = 12 meses

Hallamos la confiabilidad del sistema:

$$R(12) \ = \ 0.88 \ * \ e^{-\frac{1}{6}*12}$$

$$R(12) \ = \ 0.12$$

Reemplazando en la fórmula de no hallar una falla se tiene:

$$P(T > t) = \ 1 - R(t)$$

$$P(T > t) = 1 - 0.119$$
$$P(T > t) = 0.881$$
$$P(T > t) = 0.881 * 100$$
$$P(T > t) = 88.1\ \%$$
$$\text{CONFIABILIDAD} = 88.1\ \%$$

Por lo tanto, la confiabilidad del sistema es del 88.1%, lo que significa que en términos de 12 meses el sistema mantendrá un rendimiento óptimo.

4.2.1.3. Mantenibilidad

Para el cálculo de esta mantenibilidad del sistema, es decir índice de madurez del software (IMS), se establecerá los cambios que ocurrieron con cada versión del producto. Para los cual el IMS se determina con la siguiente ecuación:

$$\text{IMS} = \frac{\text{Mt} - (\text{Fc} + \text{Fa} + \text{Fd})}{\text{Mt}}$$

Donde:

Mt = Número de módulos en la versión actual.

Fc = Número de módulos en la versión actual que se ha cambiado.

Fa = Número de módulos en la versión actual que se han añadido.

Fd = Número de módulos en la versión actual que se han eliminado.

En el sistema se obtuvieron los siguientes valores como muestra la tabla 19 para la información requerida para el IMS.

Tabla 19: Información Requerida para el IMS

Información	Valores obtenidos
Mt	6
Fc	1
Fa	0
Fd	0

Ahora calculamos el índice de madurez del software sustituyendo los valores:

$$\text{IMS} = \frac{6 - (1 + 0 + 0)}{6}$$

$$\text{IMS} = 0.83$$

Para la mejor interpretación del resultado multiplicamos por 100 para sacar el porcentaje, de la siguiente forma:

$$\text{MANTENIBILIDAD} = 0.83 * 100$$

$$\text{MANTENIBILIDAD} = 83\%$$

De acuerdo a los datos obtenidos se concluye que el sistema, tiene una Mantenibilidad de 83%, este se encuentra en un rango satisfactorio.

4.2.1.4. Portabilidad

La portabilidad se refiere al esfuerzo necesario para transferir el programa de un entorno ya sea de hardware y/o software a otro, es una característica deseable de todo software.

La portabilidad tiene la siguiente fórmula:

$$P = \left[1 - \left(\frac{EP}{EI}\right)\right]$$

Dónde:

P = Portabilidad

EP = Esfuerzo para portar

EI = Esfuerzo para implementar

Consideremos que el esfuerzo para portar y para implementar es de 10 y 50 respectivamente de un rango entre el 1-100.

Entonces reemplazando valores tenemos:

$$P = \left[1 - \left(\frac{10}{50}\right)\right]$$

$$P = 0.8$$

$$P = 0.8 * 100$$

$$\text{PORTABILIDAD} = 80\%$$

Lo que significa que el sistema es portable.

4.2.1.5. Usabilidad

La usabilidad o facilidad de uso (FU), se calcula de la siguiente con la siguiente ecuación:

$$FU = \frac{\left[\frac{\sum x_i}{n} * 100\right]}{n}$$

En la tabla se calcula x_i y $\sum x_i$ utilizando la escala de evaluaciones:

Tabla 20: Evaluación de preguntas para Calcular la Usabilidad

Nº	Preguntas	Evaluación (xi)
1	¿El sistema satisface los requerimientos de manejo de información?	4
2	¿Las salidas del sistema están de acuerdo a sus requerimientos?	4
3	¿Cómo considera el ingreso de datos del sistema?	4
4	¿Cómo considera los formularios que elabora el sistema?	4

5	¿El sistema facilita el trabajo que realiza?	5
	TOTAL $\sum x_i$	21

Calculando FU:

$$FU = \frac{\left[\frac{21}{5} * 100\right]}{5}$$

$$FU = 84$$

$$\text{USABILIDAD} = 84\%$$

Por lo tanto, la facilidad de uso es del 84%.

4.2.2. Resultados

De acuerdo a los resultados obtenidos, se puede establecer la calidad total del sistema en base a los parámetros medidos anteriormente.

CARACTERÍSTICAS	RESULTADO
Funcionalidad	88%
Confiabilidad	88%
Mantenibilidad	83%
Portabilidad	80%
Usabilidad	84%
Evaluación de Calidad Total	84.6 %

El nivel de aceptación satisfactorio, indica que los valores de preferencia se encuentran en el rango de 60% a 100%, en nuestro caso la evaluación final nos indica que tenemos una calidad de 84.6%, que es aceptable.

4.3. SEGURIDAD DEL SOFTWARE

La ISO 27001 brinda diferentes herramientas que contribuyen a las buenas prácticas para asegurar, integrar y tener de manera confidencial toda la información y de los sistemas que la almacenan para evitar algún tipo de ciberataques, haciendo más competitiva y cuidando la reputación de la compañía.

Con el fin de prevenir que agentes externos no autorizados puedan tener acceso a estos datos, fue diseñado dela siguiente forma:

Cifrado de datos: Cuando el mensaje es enviado por el emisor lo que hace es ocultar la información hasta que esta llegue al receptor mediante un JWT (Json Web Token).

Lógica: Debe contar con un orden en donde primero van los datos del mensaje, el significado y en qué momento este se va a enviar.

Autenticación: Esta técnica se utiliza para saber que la información está siendo manipulada por un ente autorizado y no está sufriendo algún tipo de intervención por agentes externos.

4.3.1. Seguridad a Nivel de Sistema Operativo

Se tiene un servidor SONY VAIO con el S.O. Ubuntu 20.4 y sus configuraciones necesarias de red segura, como cortafuego activado, ssh y seguridad en IP.

4.3.2. Seguridad a nivel Base de Datos

A nivel de base de datos se aplica la seguridad de copias de seguridad, acceso y permiso a cuentas de usuario, mediante una contraseña.

4.3.3. Seguridad a nivel de Software

Al ser un sistema web privado, se tiene un acceso de usuario (Login) mediante un Json Web Token. Y un código de seguridad para evitar acceso a personas no autorizadas al sistema.

CAPÍTULO V

5. COSTO DEL SISTEMA

5. COSTO DEL SISTEMA

5.1. INTRODUCCIÓN

La técnica de Análisis de Costo y Beneficio, tiene como objetivo fundamental proporcionar una medida de la rentabilidad de un proyecto, mediante la comparación de los costos previstos con los beneficios esperados en la realización del mismo. Esta técnica se debe utilizar al comparar proyectos para la toma de decisiones.

Para calcular el costo del proyecto se lo realizará haciendo uso del modelo COCOMO II. El modelo COCOMO tiene una jerarquía de modelos como ser: básico, intermedio y avanzado, la cual se aplica a tres diferentes tipos de software.

5.2. COCOMO II

Para el sistema se utilizará el modelo COCOMO en su nivel básico semi-acoplado.

Aplicando de las fórmulas básicas de esfuerzo, tiempo calendario y personal requerido.

Las 3 ecuaciones de COCOMO básico tiene la siguiente forma:

$$\mathrm{E} = \mathrm{a}(\mathrm{KLDC})^{\mathrm{b}}, \text{en personas /mes}$$

$$\mathrm{T} = \mathrm{c}(\mathrm{E})^{\mathrm{d}}, \text{en meses}$$

$$\mathrm{P} = \frac{\mathrm{E}}{T}, \text{en personas}$$

Donde:

E: Esfuerzo requerido por el proyecto, en meses.

D: Tiempo requerido por el proyecto, en meses.

P: Número de personas requeridas por el proyecto.

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

KLDC: Cantidad de líneas de código, en miles.

Para las constantes utilizaremos la tabla 21 que se muestra a continuación:

Tabla 21: Coeficiente a y c y los exponentes b y d

Proyecto de Software	A	B	C	D
Orgánico	2.4	1.05	2.2	0.38
Semi-Acoplado	3.0	1.12	2.5	0.35
Empotrado	3.6	1.20	2.5	0.32

5.2.1. Costo del Desarrollo del Software

Para el cálculo del desarrollo del software se tendrá como partida el punto función no ajustado valor ya encontrado en el capítulo anterior:

$$\mathrm{PF} = 504.85$$

Este resultado se debe convertir a KLDC (Kilos Líneas de Código), para ello se utiliza la siguiente tabla.

Tabla 22: Factor LCD/PF de Lenguajes de Programación

LENGUAJE	FACTOR LDC/PF
Java	53
JavaScript	**31**
Visual Basic	46
ASP	36
PHP	12
Ensamblador	320
C	150

Fuente: (QSM, 2020).

Calculando las líneas de código del sistema utilizando la siguiente ecuación tenemos:

$$\mathrm{LDC} = \mathrm{PF} * \mathrm{FACTOR\ LDC/PF}$$
$$\mathrm{LDC} = 504.85 * 31$$
$$\mathbf{LDC = 15650.35}$$

Para convertirlo a Kilo Líneas de Código (KLDC), dividimos LDC entre 1000. Calculando KLCD tenemos:

$$\mathrm{KLDC} = \frac{\mathrm{LDC}}{1000}$$
$$\mathrm{KLDC} = \frac{15650.35}{1000}$$
$$\mathbf{KLDC = 15.65}$$

A continuación, realizaremos el cálculo del esfuerzo necesario para la programación del sistema. La ecuación que nos ayudará a hallar el esfuerzo es la siguiente:

$$\mathrm{E} = \mathrm{a} * (\mathrm{KLDC})^{\mathrm{b}}, \text{en personas/mes}$$

Reemplazando Valores tenemos:

$$\mathrm{E} = 2.4 * (15.65)^{1.05}$$
$$\mathbf{E = 43.1\ Personas/Mes}$$

Calculando Tiempo requerido D con c = 2.2 y d = 0.38 tenemos:

$$\mathrm{T} = \mathrm{c}(\mathrm{E})^{\mathrm{d}}, \text{en meses}$$
$$T = 2.2 * (43.1)^{0.38}$$
$$\boldsymbol{T = 9.2 \cong 9\ meses}$$

El proyecto aproximadamente tiene un tiempo de duración para su elaboración de 9 meses.

La siguiente ecuación es para calcular el personal requerido. En este caso el número de programadores para el desarrollo es:

$$\mathrm{P} = \frac{\mathrm{E}}{T}, \text{en personas}$$

$$\mathrm{P} = \frac{43.1}{9.2}, \text{en personas}$$

$$\boldsymbol{P = 4.68 \cong 5 \text{ programadores}}$$

El salario de un programador puede oscilar entre 500 $, cifra que será tomada en cuenta para la estimación siguiente:

$$\text{Costo del software por persona} = \mathrm{P} * \text{salario de un programador}$$

$$\text{Costo del software por persona} = 5 * 500\ \$$$

$$\textbf{Costo del software por persona} = 2500\ \$$$

Convirtiendo en bolivianos tenemos:

$$\text{Costo total de desarrollo} = \text{Costo del software desarrollado} * 6.97$$

$$\text{Costo total de desarrollo} = 2500 * 6.97$$

$$\textbf{Costo total de desarrollo} = \textbf{17425 Bs}.$$

5.2.2. Costo de Elaboración del Proyecto

Los costos de elaboración del proyecto se refieren a los costos del estudio del sistema, en la etapa de análisis y recopilación principalmente, estos costos se representan en la siguiente tabla.

Tabla 23: Costo de Elaboración del Proyecto

DETALLE	IMPORTE (Bs)
Análisis y diseño de sistema	1000
Bibliografía	500
Material de escritorio	1000

Internet	500
Otros	100
TOTAL	**3100**

5.2.3. Costo Total de Proyecto

El costo es la sumatoria del costo del software desarrollado, costo de elaboración del proyecto, detallados en la siguiente tabla.

Tabla 24: Costo Total del Proyecto

DETALLE	IMPORTE (Bs)
Costo del Software Desarrollado	17425
Costo de Elaboración del Proyecto	3100
TOTAL	**20525**

Por lo tanto, el costo total del proyecto es 20525 Bs. o su equivalencia en dólares americanos a una tasa de cambio de 6.97 es de 2945 $us.

CAPÍTULO VI

6. CONCLUSIONES Y RECOMENDACIONES

6. CONCLUSIONES Y RECOMENDACIONES

6.1. INTRODUCCIÓN

En este capítulo después de haber llevado a cabo el proceso de desarrollo del sistema y de haber realizado el análisis de los resultados, para concluir, se procede a presentar las conclusiones del estudio y posteriormente las recomendaciones que se hacen al municipio.

6.2. CONCLUSIONES

El objetivo general de este proyecto de grado, era desarrollar un “Sistema de Información para realizar el Seguimiento y Control de Ejecución de Proyectos”, con la finalidad de apoyar el fortalecimiento del Gobierno Autónomo Municipal de Catacora, con la elaboración del presente proyecto se llegó a su conclusión satisfactoria logrando alcanzar los objetivos específicos propuestos y cumpliendo los diferentes requerimientos del municipio:

- Se optimizó la gestión de información acerca de los proyectos que están en ejecución y conclusión, evitando perdida de archivos y manteniendo un control de los mismos.
- Se optimizó la información acerca de las empresas que ejecutan los proyectos en el municipio.
- Se optimizó la información acerca de las comunidades beneficiadas con los proyectos, mejorando en tiempo la búsqueda de una determinada comunidad.
- Se optimizó la información acerca de las poas que pertenecen a una determinada comunidad por ende a un determinado proyecto, evitando confusión de los mismos.
- Se optimizó el seguimiento y control de proyectos, mediante los cronogramas de actividades, cumpliendo fechas de inicio de conclusión.
- Se logró optimizar la generación de reportes de los proyectos en ejecución y los que ya fueron concluidos a través del módulo de reportes.

6.3. RECOMENDACIONES

Una vez concluido el proyecto de grado se recomienda como nuevos trabajos de investigación lo siguiente:

- Implementar módulos que hagan el seguimiento financiero a las direcciones de finanzas, licitaciones, contrataciones.
- Implementar el módulo de personal, el cual pueda adaptarse y contribuir al presente Proyecto de Grado.
- Realizar el mantenimiento del sistema en un determinado tiempo de (3 meses) para ver el buen funcionamiento, en el almacenamiento de información y procesos concurrentes como consultas, registro y demás.
- Se recomienda realizar backups de seguridad de la base de datos periódicamente, para llevar una copia de respaldo.

BIBLIOGRAFÍA

Administrador. (03 de 02 de 2019). *DDD Diseño Guiado por Dominios*. Obtenido de APIService: https://apiservice.cl/ddd-diseno-guiado-por-dominios-conceptos-clave/

Alvarez, M., Peña, J., & Fernandez, E. (06 de 11 de 2017). *Manual de TypeScript.* Obtenido de DesarrolloWeb: http://83.46.77.126/files/1556647787_manual-typescript-86.pdf

Blanco B., P. A., & Hernández Z., M. (2016). *SISTEMA DE INFORMACIÓN PARA LA GESTIÓN DE PROYECTOS PARA LA FUNDACIÓN UNIVERSITARIA LOS LIBERTADORES.* Bogota.

By Equipo Geek. (2 de abril de 2020). *VueJS, el framework JavaScript que te hace la vida más fácil*. Obtenido de IfGeekTeam: https://ifgeekthen.everis.com/es/vuejs-el-framework-javascript-que-te-hace-la-vida-mas-facil

Calero, W. (7 de octubre de 2010). *Ingenieria de Software.* Obtenido de ingenieraupoliana: http://ingenieraupoliana.blogspot.com/2010/10/cocomo.html

Cano, J. (22 de mayo de 2018). *Angular: Mucho más que un framework*. Obtenido de sg.com: https://sg.com.mx/revista/56/angular

Chávez G., V. G. (2010). *SISTEMA DE INFORMACION PARA EL CONTROL, SEGUIMIENTO Y MANTENIMIENTO DEL EQUIPAMIENTO HOSPITALARIO.* Lima.

Chelo, M. (01 de 04 de 2016). *Ingenieria Web*. Obtenido de http://marich.blogspot.es/1459530930/ingenieria-web/

Comisión Europea. (2001). *Manual de Gestión del Ciclo del Proyecto.* Madrid: EuropeAid.

Contributors, E. (19 de 11 de 2019). *Provincia José Manuel Pando (Bolivia)*. Obtenido de https://www.ecured.cu/index.php?title=Provincia_Jos%C3%A9_Manuel_Pando_(Bolivia)&oldid=3471162

Díaz, J. (16 de enero de 2020). *¿Qué es BACKEND y FRONTEND?* Obtenido de EDteam: https://ed.team/blog/que-es-backend-y-frontend-guia-completa

EcuRed contributors. (18 de julio de 2020). *Visual Studio Code*. Obtenido de EcuRed: https://www.ecured.cu/index.php?title=Visual_Studio_Code&oldid=3632049

Engineering, I. (07 de septiembre de 2016). *UWE - UML - BASED WEB ENGINEERING*. Obtenido de UWE: http://uwe.pst.ifi.lmu.de/

Evans, E. (12 de 06 de 2015). *Domain Driven Design*. Obtenido de domainlanguage.com: https://domainlanguage.com/wp-content/uploads/2016/05/DDD_Reference_2015-03.pdf

Gerrero, N. (28 de 6 de 2016). *Programa en Linea*. Obtenido de http://programaenlinea.net/conoces-en-que-consiste-la-arquitectura-de-software-en-scrum/

HENNICKER, R. &. (01 de 10 de 2001). *UML-based Methodology for Hypermedia Desing.* Recuperado el 20 de 04 de 2019, de UML: HENNICKER, R., & KOCH, N. (01 de 10 de 2001). A UML-based Methodology for http://www.pst.informatik.uni-muenchen.de/~kochn/Uml2000.pdf

Ibarra G., J. W. (2007). *SISTEMA DE INFORMACIÓN PARA EL CONTROL Y SEGUIMIENTO DE PROYECTOS VÍA WEB.* La Paz.

ICTEA. (5 de 6 de 2020). *¿Qué es el lenguaje de programación JAVASCRIPT?* Obtenido de ictea.com: https://www.ictea.com/cs/index.php?rp=%2Fknowledgebase%2F8801%2FiQue-es-el-lenguaje-de-programacion-JAVASCRIPT.html

Informática. (23 de mayo de 2017). *ISO 9126*. Obtenido de sites: https://sites.google.com/site/informaticamcprats/iso-9126

Kendall, J. E., & Kendall, K. E. (2011). *Análisis y Diseño de Sistemas* (Octava Edición ed.). (L. M. Cruz Castillo, Ed., & A. V. Romero Elizondo, Trad.) México: Pearson Education.

KOCH, N. (2000). *Software Engineering for Adaptive Hypermedia Systems. Referece Modeling Techniques and Development Process.* Germany:

Software Engineering for Adaptive Hypermedia Systems. ReferLudeing.Maximilians-Universität München. .

Laudon, K., & Laudon, J. (2012). *Sistemas de Información Gerencial.* Mexico: Pearson.

Maps, g. (s.f de s.f de s.f). *google Maps*. Recuperado el 20 de 04 de 2019, de Viacha: https://www.google.com.bo/maps/@-16.6540661,-68.2981548,15z

Marini, E. (15 de Octubre de 2012). *El Modelo Cliente/Servidor.* Obtenido de Manuales y descargas: https://www.linuxito.com/docs/el-modelo-cliente-servidor.pdf

Marques, J. H. (2012). *Ingenieria del Software II (Calidad del Software).* Barselona: UOQ.

Molina, T. (28 de septiembre de 2018). *Angular: Arquitectura del Framework*. Obtenido de medium: https://medium.com/angular-chile/angular-arquitectura-del-framework-e46204f38fef

Montilva, J. (1999). *Desarrollo de Sistemas de Información.* Burroughs de Venezuela: Ula.

Morales, A. (13 de 12 de 2012). *Introducción a NodeJS*. Obtenido de DesarrolloWeb: https://desarrolloweb.com/articulos/intro-nodejs.html

Morales, A. (12 de enero de 2015). *HomeMEANIntroducción al Stack MEAN y sus componentes: MongoDB, Express, AngularJS y NodeJS*. Obtenido de Funny Frontend : https://funnyfrontend.com/introduccion-stack-mean-parte-1/

Morales, A. (28 de agosto de 2018). *Descubre el nuevo pgAdmin 4 para trabajar con PostGIS*. Obtenido de MappingGIS: https://mappinggis.com/2017/11/descubre-el-nuevo-pgadmin-4-para-trabajar-con-postgis/

Morales, D. (20 de 05 de 2011). *Seguridad Informatica*. Recuperado el 27 de 08 de 2019, de Seguridad del sistema: https://www.significados.com/seguridad-informatica/

Navarro, A. (2 de 11 de 2013). *Revisión de metodologías ágiles para el desarrollo de software*. Recuperado el 20 de 6 de 2020, de redalyc.org: https://www.redalyc.org/articulo.oa?id=4962/496250736004

Olivera, J. (15 de 11 de 2014). *JWT: Json Web Token.* Obtenido de Ingeniería Informática Teoria y Aplicación de la Informática 2: http://jeuazarru.com/wp-content/uploads/2017/11/JWT.pdf

Peralta, A. (15 de 6 de 2003). *Metodología SCRUM.* Recuperado el 20 de 6 de 2020, de Universidad ORT Uruguay: https://fi.ort.edu.uy/innovaportal/file/2021/1/scrum.pdf

Pressman, R. (2010). *Ingeniería de Software. Un Enfoque Práctico.* (Septima Edición (SEPTIMA EDICIÓN ed.). ed.). Mexico: MC. GRAW HILL.

QSM. (15 de 6 de 2020). *Function Point Languages Table*. Obtenido de QSM Resources: https://www.qsm.com/resources/function-point-languages-table

Teran, A. R. (25 de 05 de 2016). *Aspectos basicos de la seguridad en aplicaciones Web.* Recuperado el 20 de mayo de 2019, de Seguridad en Aplicaciones Web: https://www.seguridad.unam.mx/historico/documento/index.html-id=17

Vásquez, G. (28 de junio de 2017). *¿Qué es Angular? Cómo aprender Angular desde Cero*. Obtenido de codigoonclick: https://codigoonclick.com/que-es-angular/

Zapata Ch., K. B. (5 de 5 de 2018). *Seguridad de la información basado en Normas ISO/IEC 27001.* Obtenido de bitstream: http://192.188.46.193/bitstream/123456789/73152/1/ZAPATA%20CHASIGUASIN%20KEVIN%20BRYAN%20-%202019.pdf

Zea, M. P., Molina, J. R., & Redrován, F. F. (2017). *ADMINISTRACIÓN DE BASES DE DATOS CON POSTGRESQL* (Primera Edición ed.). C/ Els Alzamora: Área de Innovación y Desarrollo, S.L.

ANEXOS

ANEXO A: AVALES DE CONFORMIDAD

El Alto, 23 de noviembre de 2020

Señor
Ing. David Carlos Mamani Quispe
DIRECTOR
CARRERA INGENIERÍA DE SISTEMAS – UPEA

Presente. -

REF.: AVAL DE CONFORMIDAD

Distinguido ingeniero:

Tengo a bien dirigirme a su persona para comunicarle mi conformidad del trabajo final del Proyecto de Grado, titulado **"SISTEMA DE INFORMACIÓN PARA EL SEGUIMIENTO Y CONTROL DE EJECUCIÓN DE PROYECTOS MUNICIPALES CASO: GOBIERNO AUTÓNOMO MUNICIPAL DE CATACORA"**, elaborado por el universitario: **CHOQUE TITO EDDY HEBERSON**, con cedula de identidad: **6050358 L.P.** y registro universitario: **102,** para su defensa pública y evaluación correspondiente a la materia de Taller de Licenciatura II, de acuerdo al reglamento vigente de la Carrera de Ingeniería de Sistemas de la Universidad Pública de El Alto.

Sin otro particular, me despido de usted.
Atentamente.

Ing. Sergio Ramiro Rojas Saire
TUTOR REVISOR

El Alto, 23 de noviembre de 2020

Señor
Ing. David Carlos Mamani Quispe
DIRECTOR
CARRERA INGENIERÍA DE SISTEMAS – UPEA

Presente. -

REF.: AVAL DE CONFORMIDAD

Distinguido ingeniero:

Tengo a bien dirigirme a su persona para comunicarle mi conformidad del trabajo final del Proyecto de Grado, titulado **"SISTEMA DE INFORMACIÓN PARA EL SEGUIMIENTO Y CONTROL DE EJECUCIÓN DE PROYECTOS MUNICIPALES CASO: GOBIERNO AUTÓNOMO MUNICIPAL DE CATACORA"**, elaborado por el universitario: **CHOQUE TITO EDDY HEBERSON**, con cedula de identidad: **6050358 L.P.** y registro universitario: **102,** para su defensa pública y evaluación correspondiente a la materia de Taller de Licenciatura II, de acuerdo al reglamento vigente de la Carrera de Ingeniería de Sistemas de la Universidad Pública de El Alto.

Sin otro particular, me despido de usted.
Atentamente.

Ing. Ivan Fernando Mujica Mamani
TUTOR ESPECIALISTA

El Alto, 23 de noviembre de 2020

Señor
Ing. David Carlos Mamani Quispe
DIRECTOR
CARRERA INGENIERÍA DE SISTEMAS – UPEA

Presente. -

REF.: AVAL DE CONFORMIDAD

Distinguido ingeniero:

Tengo a bien dirigirme a su persona para comunicarle mi conformidad del trabajo final del Proyecto de Grado, titulado **"SISTEMA DE INFORMACIÓN PARA EL SEGUIMIENTO Y CONTROL DE EJECUCIÓN DE PROYECTOS MUNICIPALES CASO: GOBIERNO AUTÓNOMO MUNICIPAL DE CATACORA"**, elaborado por el universitario: **CHOQUE TITO EDDY HEBERSON**, con cedula de identidad: **6050358 L.P.** y registro universitario: **102,** para su defensa pública y evaluación correspondiente a la materia de Taller de Licenciatura II, de acuerdo al reglamento vigente de la Carrera de Ingeniería de Sistemas de la Universidad Pública de El Alto.

Sin otro particular, me despido de usted.
Atentamente.

Ing. Marisol Arguedas Balladares
TUTOR METODOLÓGICO

MARKAS:

Catacora

Tolacollo

Parachi

Pairumani Grande

Pajchiri

El Alto, La Paz, noviembre de 2020

Señores:

CARRERA INGENIERÍA DE SISTEMAS

UNIVERSIDAD PÚBLICA DE EL ALTO

Presente. –

REF.: CONFORMIDAD CON EL DESARROLLO DEL "SISTEMA DE INFORMACIÓN PARA EL SEGUIMIENTO Y CONTROL DE EJECUCIÓN DE PROYECTOS MUNICIPALES"

De mi consideración:

Mediante la presente tengo a bien dirigirme a ustedes a tiempo de poner en su conocimiento las siguientes aclaraciones sobre el proyecto de grado titulado "Sistema de Información para el Seguimiento y Control de Ejecución de Proyectos Municipales Caso: Gobierno Autónomo Municipal de Catacora", desarrollado por el Sr. EDDY HEBERSON CHOQUE TITO con C.I. 6050358 LP:

- Que cumple con las especificaciones técnicas acordadas en la Unidad Técnica de Proyectos.
- Ha concluido satisfactoriamente el proyecto, cumpliendo tal cual indican los objetivos planteados.

En este sentido, corresponde emitir el Aval de conformidad del mismo.

GOBIERNO AUTÓNOMO MUNICIPAL CATACORA Prov. J.M. Pando La Paz - Bolivia

ABSALON CONURANA SURCO
ALCALDE GAM. CATACORA

ANEXO B: Cuestionario para Calcular Punto de Fusión

Nombres y apellidos……

Cargo: ……

Marcar que con una X la puntuación

Tu sinceridad es muy importante en cuanto es tu facilidad de uso del sistema

ESCALA	**Sin importancia**	**Incidental**	**Moderado**	**Medio**	**Significativo**	**Esencial**
Factor	0	1	2	3	4	5
1. ¿Requiere el sistema copias de seguridad y recuperación fiable?						
2. ¿Se requiere comunicación de datos?						
3. ¿Existe funciones de procesamiento distribuido?						
4. ¿Es crítico el rendimiento?						
5. ¿Será ejecutado el sistema en un entorno operativo existente y frecuentemente utilizado?						
6. ¿Requiere el sistema entrada de datos interactivos?						
7. ¿Requiere la entrada de datos interactivos que las transiciones de entrada se						

lleven a cabo sobre múltiples o variadas operaciones?						
8. ¿Se actualiza los archivos de forma interactiva?						
9. ¿Son complejas las entradas, las salidas, los archivos o peticiones?						
10. ¿Es complejo el procesamiento interno?						
11. ¿Se ha diseñado el código para ser reutilizable?						
12. ¿Están incluidos en el diseño la conversión y la instalación?						
13. ¿Se ha diseñado el sistema para soportar múltiples instalaciones en diferentes plataformas?						
14. ¿Se ha diseñado la aplicación para facilitar los cambios y para ser fácilmente utilizados por el usuario?						
TOTAL	$\sum(Fi) = 50$					

ANEXO C: MANUAL TÉCNICO

MANUAL TÉCNICO

INTRODUCCIÓN

Este manual describe los pasos necesarios para cualquier persona que tenga ciertas bases de sistemas pueda realizar la instalación del sistema creado para el Seguimiento y Control de Ejecución de Proyecto Municipales del Gobierno Autónomo Municipal de Catacora.

REQUERIMIENTOS DEL SISTEMA

Requerimientos mínimos de Hardware

- Un ordenador (computadora PC), con las siguientes características: Procesador Core i5 de 2.8Gz, memoria RAM de 4GB, disco duro de 250GB y tarjeta de video incorporado.
- Conexión a Internet.

Requerimientos mínimos de Software

- Sistema Operativo Linux Ubuntu v 20.04.
- El **Frontend** de SISCEPROM fue desarrollado con el framework Vue.js, y las tecnologías empleadas fueron:
  - ✓ Vue-router para el manejo de rutas. [Documentación oficial](https://router.vuejs.org/)
  - ✓ Vuex para el manejo de estados del sistema. [Documentación oficial](https://vuex.vuejs.org)
  - ✓ Vuetify en su última versión como framework CSS y componentes. [Sitio oficial](https://vuetifyjs.com/)
  - ✓ Axios para el manejo de peticiones AJAX y demás. [Documentación](https://github.com/mzabriskie/axios)
- El **Backend** de SISCEPROM fue desarrollado con las siguientes tecnologías:

- ✓ Nodejs 10.13.0 como motor de ejecución de JavaScript [Sitio oficial] (https:// https://nodejs.org//)
- ✓ Postgresql 9 Como gestor de Base de Datos [Sitio oficial] (https:// https://postgresql.org //)
- ✓ ExpressJs 4 Como framework de Nodejs [Sitio oficial] (https:// https:// https://expressjs.com///)

INSTALACIÓN

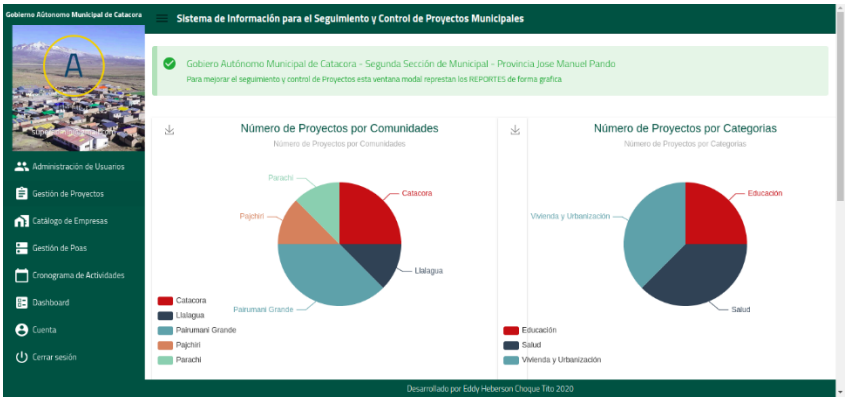
- Instale todas las tecnologías anteriormente citadas, la instalación de todas las tecnologías usadas tanto para el Backend y Frontend, se encuentra en la propia documentación indicada.
- Cree una carpeta llamada SISCEPROM en la unidad C, a continuación, copie las carpetas llamadas etc-proyeco-frontend y etc-proyecto-backend en la carpeta ya antes creada.
- Una vez copiada las carpetas, en la terminal del ordenador primero ubíquese dentro la carpeta etc-proyectos-backend y ejecute el siguiente comando **npm run startdev** y el backen entrara en funcionamiento, segundo ubíquese dentro la carpeta etc-proyectos-frontend y ejecute **npm run start** y el frontend de igual forma entrara en ejecución y de forma automática nos mostrara el login del usuario.

ANEXO D: MANUAL DE USUARIO

Manual de Usuario

SISTEMA DE INFORMACIÓN PARA EL SEGUIMIENTO Y CONTROL DE EJECUCIÓN DE PROYECTO MUNICIPALES

CASO: GOBIERNO AUTÓNOMO MUNICIPAL DE CATACORA



1. INTRODUCCIÓN

Actualmente con el avance de la tecnología es necesario contar con un sistema informático para gestionar todo tipo de información, en el Gobierno Autónomo Municipal de Catacora, es imprescindible contar con este tipo de software para el área técnica de proyectos el cual pueda gestionar proyectos municipales que están en ejecución, por tanto:

De acuerdo a lo mencionado, se desarrolló el SISTEMA DE INFORMACIÓN PARA EL SEGUIMIENTO Y CONTROL DE EJECUCIÓN DE PROYECTOS MUNICIPALES que automatiza el registro de proyectos, contiene un módulo de catálogo de empresas, un módulo para la gestión de poas, cronograma de actividades y un módulo de Administración de Usuarios, las cuales nos brindan información segura, rápida y confiable.

2. OBJETIVO

Guiar al usuario final mediante este manual para el uso del sistema en forma correcta.

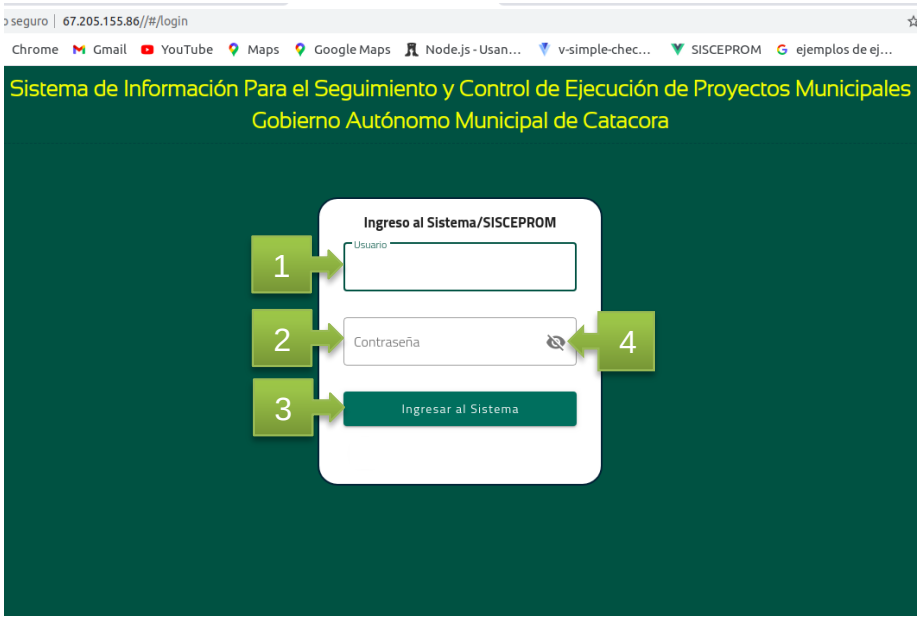
3. TIPOS DE USUARIO

El sistema contiene tres tipos de usuario:

- Usuario **superadmin**, que tiene permisos a todos los módulos del sistema.
- Usuario **admin**, que tiene permiso a los módulos de Administración de Usuarios, y el Dashboard.
- Usuario **funcionario**, que tiene permisos a los módulos de Gestión de Proyectos, Catálogo de Empresas, Gestión de Poas, Cronograma de Actividades y el Dashboard.

4. INGRESO AL SISTEMA

4.1. Interfaz de Inicio de Sesión



1. Ingrese usuario del sistema.
2. Ingrese la contraseña.
3. Presione en el botón **Ingresar al sistema**, y el sistema nos mostrara los módulos correspondientes al usuario.
4. Nos permite ver la contraseña.

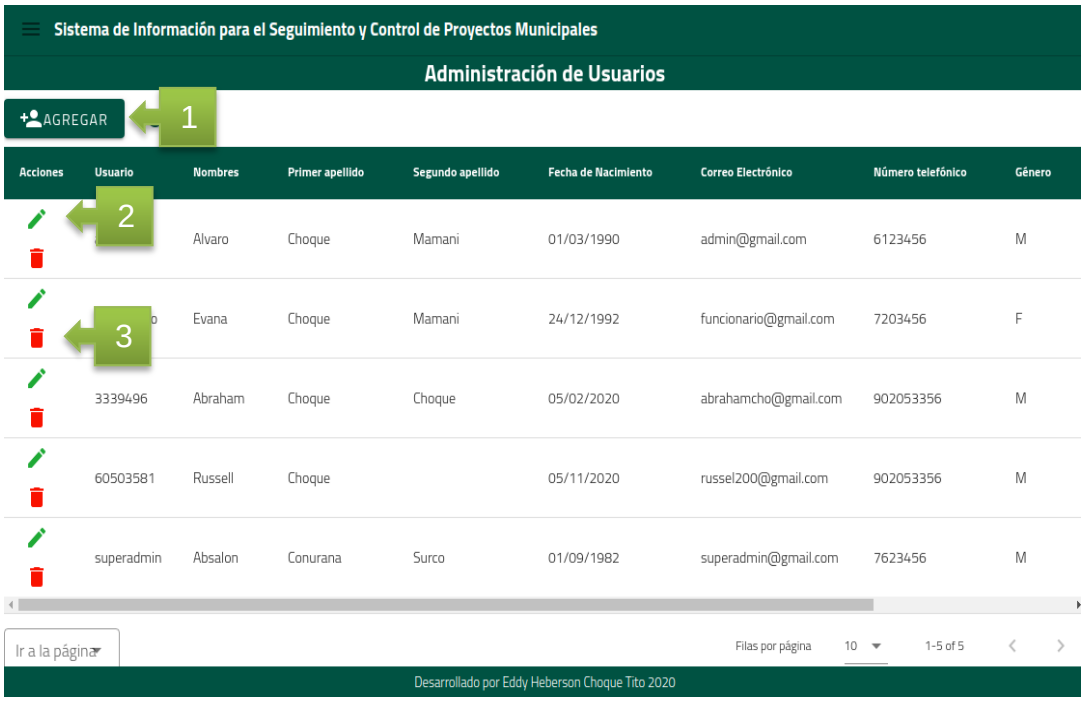
5. Módulos que integran el sistema

El sistema cuenta con los siguientes módulos:

1. Módulo de Administración de Usuarios.
2. Módulo de Gestión de Proyectos.
3. Módulo Catálogo de Empresas
4. Módulo de Gestión de Poas
5. Módulos de Cronograma de Actividades
6. Dashboard
7. Menú para personalizar la cuenta de un usuario.
8. Opción que permite Cerrar Sesión del Usuario.

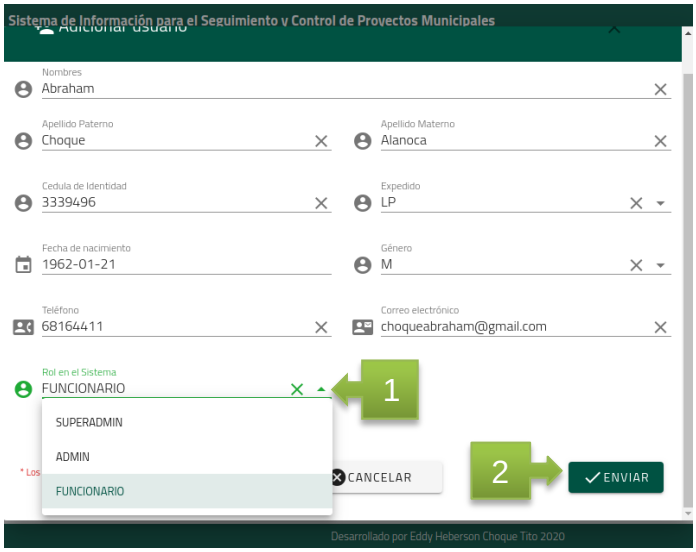
5.1. Administración de Usuarios

Este módulo permite gestionar usuarios (Agregar, Editar y Eliminar), cabe recalcar que, al momento de crear un usuario, se debe asignar un rol.



1. Acción que nos permite agregar un nuevo usuario.
2. Acción que nos permite editar un nuevo usuario.
3. Acción que nos permite eliminar un nuevo usuario.

Agregar un nuevo usuario



Se debe ingresar todos los campos obligatorios (*), para agregar un nuevo usuario.

Para editar un nuevo usuario.

1. Es muy importante seleccionar el Rol del usuario (superadmin, admin y funcionario).
2. Presione en el botón enviar para agregar y guardar un usuario.

Nota: El nombre del usuario con el cual crea el sistema es: **usuario**: cedula de identidad, **password**: cedula de identidad, que luego el usuario puede personalizar.

Editar un Usuario

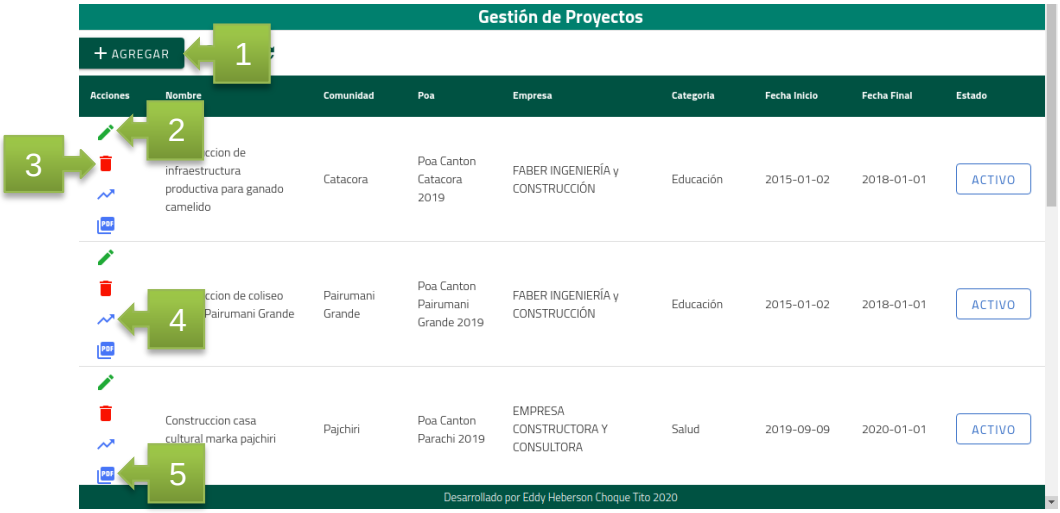
Al presionar en el botón editar usuario, se desplegara un modal donde se puede cambiar o actualizar los datos de un usuario.

Borrar un Usuario

La presionar el botón borrar usuario, se desplegará el siguiente mensaje de alerta donde puede aceptar o cancelar.

5.2. Gestión de Proyectos

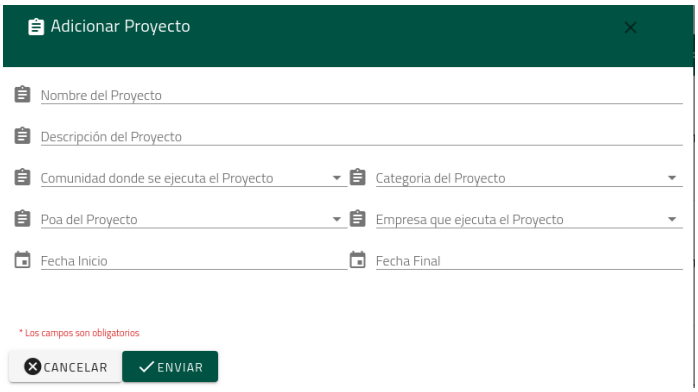
Este módulo visualiza todos los proyectos del municipio, este modal realiza el seguimiento y control mediante cronograma de actividades, también genera reportes de los datos técnicos del proyecto y del cronograma de actividades.



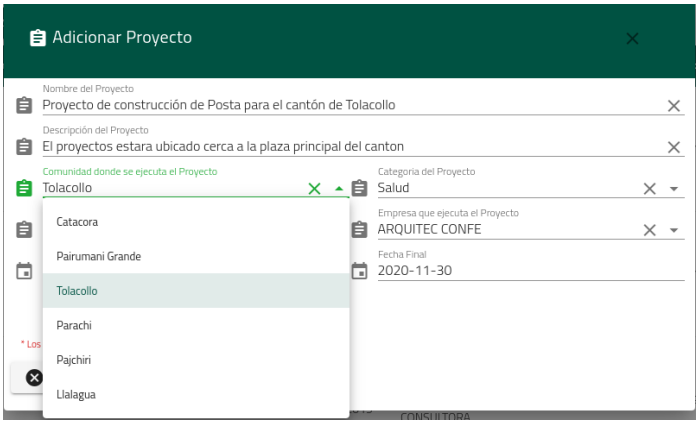
1. Acción que permite agregar un nuevo proyecto.
2. Acción que permite editar un proyecto.
3. Acción que permite eliminar un proyecto.
4. Acción que permite realizar seguimiento y control de proyecto por cronograma de actividades.
5. Acción para ver la ficha del proyecto, donde se brinda los datos técnicos del mismo.

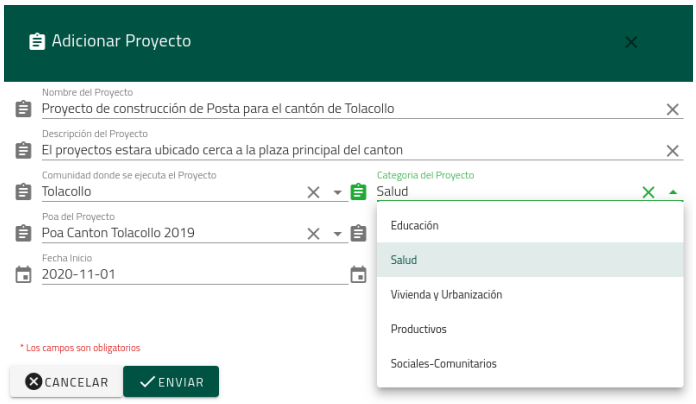
Agregar un nuevo proyecto

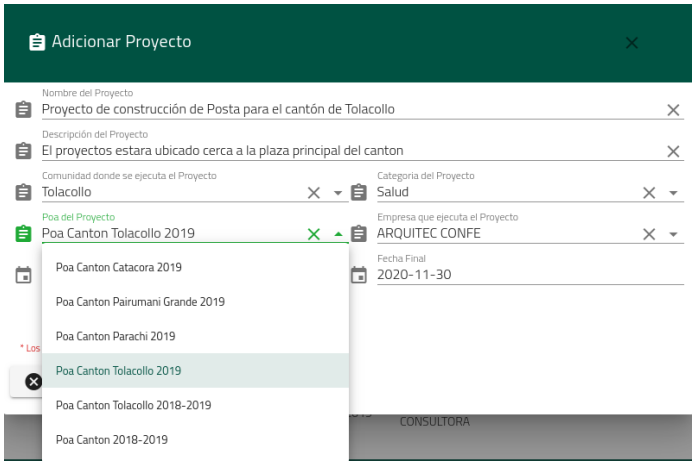
Para agregar un proyecto, es necesario primero tener registrado la empresa y la poa con la que se ejecutara el proyecto (Ver 5.3 y 5.4).

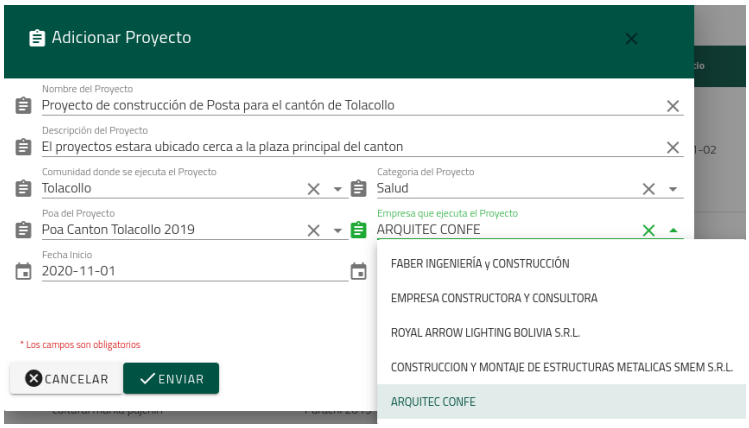


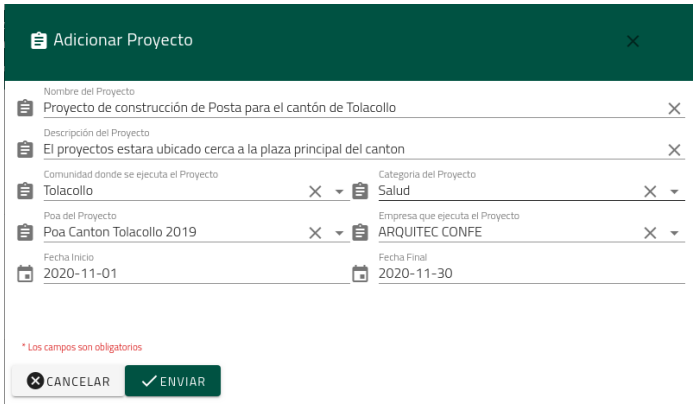
Una vez que tenemos agregado la empresa y la poa procedemos a llenar los datos del proyecto.











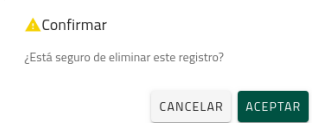
Una vez llenado los datos presione en el botón ENVIAR, para guardar los datos del proyectos.

Editar un proyecto

Al presionar en el botón editar, se desplegara un modal donde se puede cambiar o actualizar los datos del proyecto.

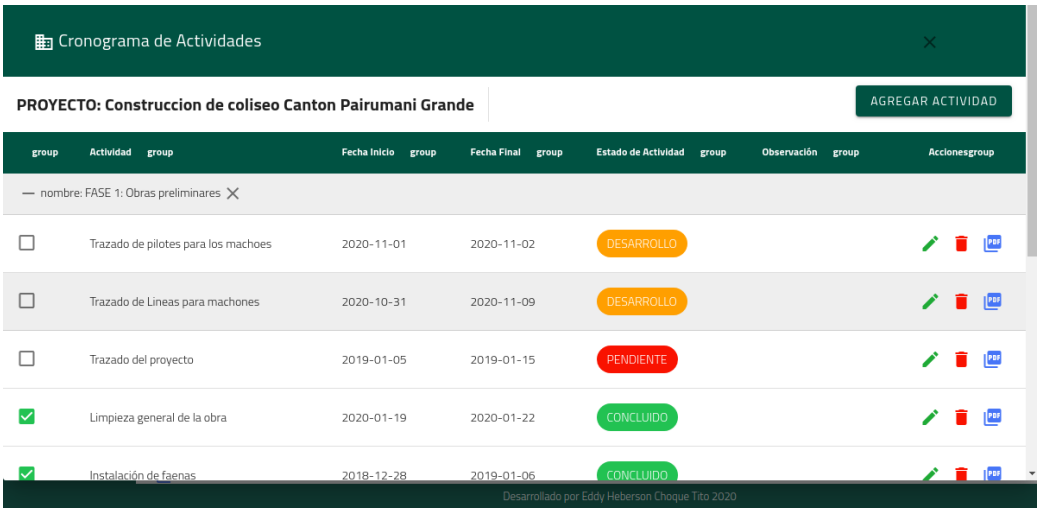
Borrar un Usuario

La presionar el botón borrar, se desplegará el siguiente mensaje de alerta donde puede aceptar o cancelar.



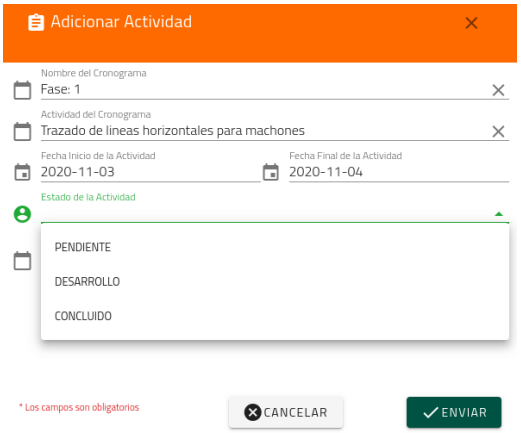
ver seguimiento

Esta acción permite ver el seguimiento y control según los cronogramas de actividades de cada proyecto.



Adicionar una Actividad

Para adicionar una nueva actividad presione en el botón AGREGAR ACTIVIDAD, y luego se desplegara un modal donde llenara los datos del formulario.



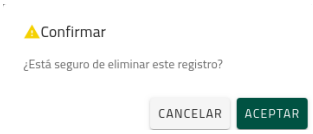
Una vez llenado los datos la actividad, es necesario seleccionar es estado de la actividad CONCLUIDO , DESARROLLO , PENDIENTE que por defecto enpieza en pendiente.

Editar una Actividad

Al presionar en el botón editar, se desplegara un modal donde se puede cambiar o actualizar los datos de una actividad.

Borrar una Actividad

La presionar el botón borrar, se desplegará el siguiente mensaje de alerta donde puede aceptar o cancelar.



Cronograma del Proyecto en PDF

Al presionar en la siguiente acción , nos mostrara el cronograma del proyecto en formato pdf como se muestra a continuación el cual se puede descargar en el formato ya antes mecionado.

La presionar en el botón Descargar, empezara a descargar un el reporte de las actividades del Proyecto en formato pdf.

Ver ficha del proyecto

Para ver la ficha del proyecto donde nos muestra los datos técnicos del proyecto presionar en la acción Ver ficha, del modal de proyectos, y se desplegara un ventana que mostrara la información que luego se puede exportar en formato pdf.

Filtrado por nombre en el modal de proyectos

Para realizar una búsqueda por nombre de proyectos, presiones en el siguiente botón , y se desplegara la opción para ingresar el nombre del proyecto y automáticamente filtrara el Proyecto.

5.3. Catálogo de Empresas

1. Acción que permite agregar una nueva empresa.
2. Acción que permite editar una empresa.
3. Acción que permite eliminar una empresa.

Agregar nueva Empresa

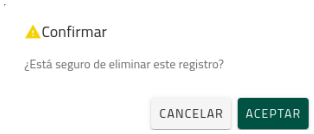
Para agregar una nueva empresa presione en el botón AGREGAR, se abrirá una ventana modal donde debemos de llenar los datos y presionar en ENVIAR para guardar los datos de la empresa.

Editar una Empresa

Al presionar en el botón editar, se desplegara un modal donde se puede cambiar o actualizar los datos de una empresa.

Borrar una Empresa

La presionar el botón borrar, se desplegará el siguiente mensaje de alerta donde puede aceptar o cancelar.



5.4. Gestión de POAs

Esta modulo nos permite gestionar las poas del municipio.

1. Acción que permite agregar una nueva poa.
2. Acción que permite editar una poa.
3. Acción que permite eliminar una poa.

5.5. Cronograma de Actividades

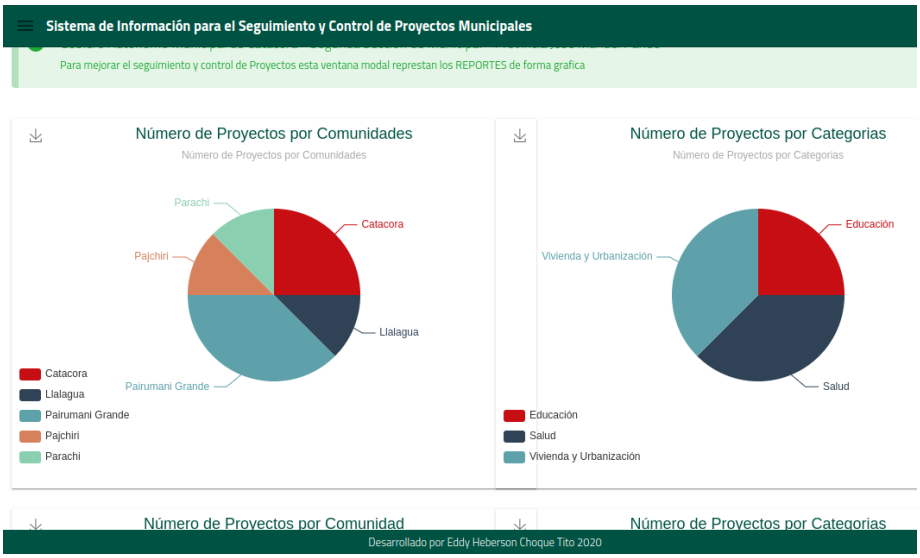
Este módulo visualiza todas las actividades de todos los proyectos, ordenando por el estado de la actividad.

1. Acción que permite agregar una nueva actividad, es muy importante seleccionar primero el proyecto al cual pertenecerá la actividad como se muestra en la siguiente ventana.

2. Acción que permite editar una actividad.
3. Acción que permite eliminar una actividad.

5.6. Dashboard

Este módulo muestra los reportes estadísticos o indicadores claves sobre la gestión de proyectos.



5.7. Cuenta

Esta opción nos permite personalizar la cuenta del usuario.

5.8. Cerrar Sesión

Esta opción permite cerrar sesión del usuario.