UNIVERSIDAD CATÓLICA BOLIVIANA “SAN PABLO”

SEDE TARIJA

DEPARTAMENTO DE CIENCIAS DE LA TECNOLOGÍA E INNOVACIÓN

CARRERA INGENIERÍA DE SISTEMAS

TALLER DE GRADO I

Sistema para el control y gestión de agua potable

ESTUDIANTE: Luis Fernando Garcia Barja

TUTOR ACADÉMICO: Ing. Lucy Faviola Lopez Choque

TARIJA-BOLIVIA

2024

Contenido

1.Introducción

1.1. Antecedentes

Antecedente de la institución: El Comité de Agua Potable Rancho Norte (CAPRN) es una organización sin fines de lucro que se basa en la participación voluntaria de sus miembros, su objetivo principal es gestionar, operar y mantener el servicio de agua potable en la localidad donde opera.

El CAPRN se rige por la Ley de Juntas de Vecinos y demás Organizaciones Comunitarias, y posee personalidad jurídica, esto significa que tiene la capacidad de actuar como una persona individual ante la ley, para su correcto funcionamiento, el CAPRN cuenta con un Estatuto y un Reglamento, estos documentos establecen las normas y procedimientos que rigen la organización y el funcionamiento del Comité, el CAPRN depende legal y administrativamente del Ministerio del Interior, a través de la Municipalidad de San Lorenzo, esto significa que la Municipalidad tiene la responsabilidad de supervisar y controlar las actividades del Comité, el Directorio del CAPRN es el órgano responsable de la dirección y administración del Comité y del Servicio de Agua Potable, la Asamblea General de Socios elige cinco miembros titulares y cinco suplentes.

EPSA Morros Blancos: En el barrio de Morros Blancos, se ha implementado un sistema de información personalizado para gestionar el suministro de agua en las viviendas de los barrios Morros Blancos, Nuevo Amanecer y Artesanal, este sistema, desarrollado como una aplicación web local, tiene la capacidad de administrar usuarios, registrar datos de las viviendas, monitorear el consumo de agua, gestionar los cobros por el servicio y generar recibos de manera automatizada, este sistema fue diseñado específicamente para las necesidades de la comunidad y el cual tuvo un costo de implementación de 12,000 bolivianos.

1.2. Delimitación del Trabajo

1.2.1. Limite Temático

El sistema abarcara la gestión de clientes, viviendas, catastros, emisión de recibos, cobranzas y control de consumo de agua, no abarcara análisis

financieros profundos, facturación, pagos por internet ni accesos abiertos a los clientes.

1.2.2. Límite Temporal

El sistema tendrá una adopción inmediata desde el momento de la entrega hasta un tiempo indefinido.

1.2.3. Limite Geográfico

El Comité de Agua Potable de Rancho Norte y su lugar de trabajo está en Rancho Norte, San Lorenzo Tarija.

1.3. Planteamiento del Problema.

En los últimos años, el Comité de Agua Potable de Rancho Norte ha enfrentado una serie de desafíos que han impactado significativamente en la gestión eficiente de sus operaciones, uno de los principales problemas radica en el hecho de que gran parte de las actividades son gestionadas manualmente, lo que ha generado una serie de inconvenientes operativos, esta dependencia de procesos manuales ha dado lugar a la pérdida ocasional de información crítica, dificultades para acceder rápidamente a datos relevantes y una notable ineficiencia en el manejo de los recursos disponibles.

Además, el almacenamiento de documentos físicos ha demostrado ser insuficiente, lo que dificulta aún más la recuperación rápida y precisa de la información cuando es necesaria, esta falta de un sistema de gestión de documentos eficiente ha llevado a situaciones donde la información importante se extravía o se vuelve difícil de rastrear, lo que impacta negativamente en la capacidad del comité para tomar decisiones informadas y oportunas.

Otro desafío importante que enfrenta el Comité de Agua Potable de Rancho Norte es la presencia de redundancias en los procesos. Esto significa que ciertas tareas se realizan varias veces, consumiendo recursos valiosos y generando confusión en el flujo de trabajo, estas redundancias no solo ralentizan la velocidad de las operaciones, sino que también aumentan el riesgo de errores y malentendidos, lo que puede tener consecuencias graves en la calidad del servicio de agua potable proporcionado a la comunidad.

1.4. Propuesta de Solución.

¿Cómo un sistema de información puede mejorar la eficiencia en la gestión del Comité de Agua Potable de Rancho Norte?

La implementación de un sistema de información integral representa una solución crucial para abordar las dificultades que enfrenta el Comité de Agua Potable de Rancho Norte en la gestión de sus operaciones, en contraste con el método manual actualmente empleado, un sistema de información ofrece una serie de ventajas significativas que pueden transformar radicalmente la eficiencia y efectividad del comité.

La automatización de procesos a través de un sistema de información eliminará la necesidad de realizar tareas repetitivas de forma manual, mientras que el método manual está sujeto a errores humanos y a la inconsistencia en la ejecución de tareas, un sistema automatizado garantiza la precisión y coherencia en cada etapa del proceso operativo y administrativo, además, la centralización de la información en un sistema de información proporciona acceso instantáneo y seguro a datos relevantes en cualquier momento y desde cualquier lugar.

En contraposición, el método manual implica la dependencia de documentos físicos dispersos, lo que dificulta la recuperación rápida y precisa de información crítica.

Con un sistema de información, el Comité de Agua Potable de Rancho Norte puede agilizar la toma de decisiones al tener a su disposición datos actualizados y completos en tiempo real.

Otro aspecto importante para considerar es la gestión financiera de los cobros, un sistema de información permite un seguimiento detallado y transparente de los recursos financieros en este caso el cobro mensual a los usuarios, facilitando la identificación de áreas desviaciones presupuestarias, el método manual carece de la capacidad de generar informes oportunos, lo que dificulta la rendición de cuentas por el agua distribuida.

1.5. Objetivos.

Objetivo General: Desarrollar e implementar un Sistema de Información para el Comité de Agua Potable de Rancho Norte que automatice y centralice sus operaciones, mejorando la eficiencia en la gestión administrativa y operativa.

Objetivos Específicos:

- **Análisis y Diseño:** En esta etapa, se llevará a cabo un análisis exhaustivo de los procesos, necesidades y herramientas del Comité de Agua Potable de Rancho Norte, con el objetivo de comprender en profundidad sus operaciones, este análisis servirá como base para diseñar la arquitectura del Sistema de Información, definiendo la estructura de la base de datos, la lógica de negocio y la interfaz de usuario, se elaborará la documentación necesaria para guiar el desarrollo e implementación del sistema.
- **Desarrollo:** Durante esta fase, se procederá a implementar las funcionalidades definidas en la etapa de análisis y diseño, se utilizarán herramientas de desarrollo adecuadas para escribir el código necesario y garantizar que el Sistema de Información cumpla con los requisitos identificados, el objetivo es desarrollar un sistema eficiente y de alta calidad que optimice los procesos del comité.
- **Implementación del sistema:** Una vez completado el desarrollo del Sistema de Información, se procederá a su instalación, configuración y migración de datos, esto implicará la instalación del software en la infraestructura del comité, así como la configuración de los parámetros necesarios para su correcto funcionamiento, se llevará a cabo la migración de los datos existentes al nuevo sistema, asegurando la integridad y consistencia de la información.
- **Capacitación y Evaluación: Finalmente:** se proporcionará formación adecuada a los usuarios del Sistema de Información para asegurar su uso efectivo, se elaborarán manuales de usuario detallados y se organizarán sesiones de capacitación para familiarizar a los

usuarios con el sistema, se realizará una evaluación continua del desempeño y satisfacción de los usuarios, recopilando retroalimentación y realizando ajustes según sea necesario para optimizar la experiencia de uso del sistema.

1.6. Justificación.

La falta de sistemas comerciales diseñados específicamente para el tipo de instituciones como las cooperativas de agua plantea un desafío significativo, ya que sus procesos de trabajo y requisitos operativos pueden diferir considerablemente de los de otras organizaciones, la adaptación de sistemas existentes a estas necesidades únicas puede resultar difícil debido a la forma especializada en la que operan estas cooperativas, esta situación subraya la importancia de buscar soluciones personalizadas que se ajusten de manera más precisa a sus requerimientos, garantizando así la eficiencia y efectividad en la gestión de sus operaciones.

1.6.1. Justificación Social:

La implementación de un sistema de información para la gestión del agua potable en la cooperativa de Rancho Norte tiene una trascendencia social significativa, los principales beneficiarios de este proyecto son los miembros de la comunidad que dependen del suministro de agua potable para sus actividades diarias, al mejorar la eficiencia y la gestión de los recursos hídricos, se garantiza un acceso más equitativo y sostenible al agua, lo que contribuye a mejorar la calidad de vida y el bienestar de todos los habitantes de la zona, además, al optimizar los procesos administrativos y operativos de la cooperativa, se fomenta una mayor transparencia y participación ciudadana en la gestión de los servicios básicos.

1.6.2. Justificación Económica:

Desde una perspectiva económica, la implementación de un sistema de información para la gestión del agua potable en la cooperativa de Rancho Norte representa una inversión con potencial para generar beneficios tanto

a corto como a largo plazo, en primer lugar, se espera que la automatización y centralización de las operaciones conduzcan a una reducción de costos operativos, derivada de la eliminación de procesos manuales y la optimización de recursos, al mejorar la eficiencia en la gestión del agua, se pueden evitar pérdidas económicas asociadas a fugas y mal uso de recursos, a largo plazo una mejor gestión del agua potable puede contribuir al desarrollo económico sostenible de la comunidad, al garantizar un suministro estable y de calidad que favorezca la actividad económica local y la atracción de inversión.

1.6.3. Justificación Personal

La decisión de emprender este proyecto de investigación y desarrollo de un sistema de información para gestionar el agua potable en el comité de Rancho Norte responde a una motivación arraigada en el compromiso con el bienestar y desarrollo de la comunidad, como estudiante de un área tecnológica, entiendo el valor de la información y así puede manejarse de forma sencilla o eficaz, por lo que buscar soluciones innovadoras y eficaces para mejorar la calidad y eficiencia de los servicios de agua potable.

2. Marco Teórico.

2.1. Marco Referencial.

EPSA Morros Blancos: La implementación de un sistema de información personalizado para la gestión del suministro de agua en los barrios Morros Blancos, Nuevo Amanecer y Artesanal ofrece un caso relevante para este estudio, aunque no se abordan todos los aspectos operativos del CAPRN, este caso proporciona información útil sobre la efectividad y los beneficios de la automatización en la gestión de servicios de agua potable a nivel comunitario.

2.2. Marco Reglamentario.

Reglamento Interno de CAPRN:

Dentro del marco legal que rige las operaciones de CAPRN, destaca su reglamento interno elaborado por los funcionarios de la organización, este reglamento meticulosamente desarrollado, aborda aspectos cruciales para el funcionamiento de CAPRN, tales como los requisitos para acceder al servicio de agua y las sanciones por consumo excesivo, al ser creado por los propios funcionarios de CAPRN, este reglamento refleja un profundo entendimiento de las necesidades y realidades locales, permitiendo una adaptación ágil a las dinámicas cambiantes de la comunidad, esta autonomía regulatoria no solo promueve una gestión más eficiente, sino que también fortalece el vínculo entre CAPRN y la comunidad que sirve, al garantizar que las políticas y normativas estén alineadas con sus intereses y necesidades específicas ((CAPRN), 2015).

Regulación de Precios por Metro Cúbico de Agua por la AAPS:

Un aspecto crucial dentro del marco legal es la regulación de los precios por metro cúbico de agua el cual esta detallado en el reglamento interno de CAPRN, no obstante, el precio no es determinado por CAPRN, sino por la Autoridad de Fiscalización y Control Social de Agua Potable y Saneamiento Básico (AAPS), esta separación de funciones garantiza una supervisión imparcial y transparente de los precios, lo que contribuye a la equidad en el acceso al agua potable, esta regulación externa proporciona un marco sólido para la fijación de precios que

tiene en cuenta tanto los costos operativos como las necesidades de la comunidad, promoviendo así la sostenibilidad a largo plazo del servicio de agua ((CAPRN), 2015).

Procesos manuales CAPRN:

En CAPRN se hacen todos los procesos de forma manual, como ser el registro de nuevos usuarios mediante del llenado de formularios de catastros, otro punto muy tardado que se transforma en un procedimiento mecánico por parte del trabajo es el cálculo del cobro del precio de agua y la búsqueda del código del usuario que quiere realizar un pago para verificar si no tiene deudas a pagar, el cobro de multas debido y las actualizaciones de información en el momento de un cambio de vivienda de algún usuario ((CAPRN), 2015).

Certificación del Agua Recabada por los Pozos Subterráneos de CAPRN por Cosaalt:

La certificación del agua recabada por los pozos subterráneos de CAPRN por parte de COOPERATIVA DE SERVICIOS PUBLICOS DE AGUA POTABLE Y ALCANTARILLADO SANITARIO TARIJA RL (Cosaalt) es un elemento vital para garantizar la calidad y seguridad del suministro de agua potable, esta asociación refleja un compromiso con los más altos estándares de calidad y seguridad, ya que Cosaalt, como entidad certificadora reconocida, realiza evaluaciones exhaustivas para asegurar que el agua cumpla con las normativas sanitarias y de calidad, esta certificación proporciona confianza a los usuarios de CAPRN, garantizando que el agua que consumen es segura y apta para su uso, lo que contribuye a la salud y bienestar de la comunidad en general.

2.3. Ingeniería del Software

La ingeniería de software es una disciplina multidisciplinaria que aborda el desarrollo de software desde un enfoque sistemático y riguroso, aplicando principios de ingeniería para gestionar de manera efectiva los procesos involucrados en todas las etapas del ciclo de vida del software, este enfoque implica no solo la creación del software en sí mismo, sino también la

consideración de aspectos como la especificación de requisitos, el diseño, la implementación, la verificación, la validación, el despliegue, el mantenimiento y la evolución del software a lo largo del tiempo, la ingeniería de software se basa en la aplicación de metodologías, modelos, estándares y prácticas de gestión que permiten organizar y dirigir el desarrollo de software de manera eficiente y controlada, minimizando los riesgos y maximizando la calidad del producto final, esto implica la adopción de procesos iterativos e incrementales, la utilización de herramientas y técnicas para el modelado y la visualización, la gestión eficaz de los recursos humanos y técnicos, y el aseguramiento de la calidad en todas las etapas del proyecto, además, la ingeniería de software no solo se centra en los aspectos técnicos del desarrollo de software, sino que también considera los aspectos humanos, organizativos, económicos y sociales relacionados con la producción y el uso del software en la sociedad, esto incluye la comunicación efectiva con los stakeholders, la gestión de los cambios en los requisitos, la evaluación del impacto económico del software, y la consideración de aspectos éticos y legales en el diseño y la implementación del software (Gómez Palomo, 2020).

2.4. Metodologías Agiles

Las metodologías ágiles son enfoques de desarrollo de software iterativos e incrementales que priorizan la entrega temprana de valor al cliente y la capacidad de respuesta al cambio, estas metodologías se basan en principios como la colaboración cercana entre equipos multidisciplinarios y stakeholders, la entrega continua de software funcional, la adaptabilidad a los requisitos cambiantes y la mejora continua del proceso, las metodologías ágiles promueven la flexibilidad y la capacidad de adaptación al fomentar la iteración rápida y la retroalimentación constante, se centran en la entrega de funcionalidades de alto valor de manera regular, permitiendo a los equipos responder de manera ágil a los cambios en los requisitos del cliente y las condiciones del mercado, además las metodologías ágiles enfatizan la autoorganización y la responsabilidad compartida dentro de los equipos, fomentando un ambiente de trabajo colaborativo y empoderado, los principios ágiles se aplican no solo al desarrollo de software, sino también a la

gestión de proyectos y la interacción con los stakeholders, promoviendo la transparencia, la comunicación abierta y la toma de decisiones basada en datos (Porras, 2023).

2.4.1. Programación Extrema (XP)

La Programación Extrema (XP) es una metodología ágil de desarrollo de software que se caracteriza por su enfoque en la simplicidad, la comunicación, la retroalimentación continua y la flexibilidad frente al cambio, XP se basa en una serie de principios y prácticas que buscan mejorar la calidad del software, la productividad del equipo y la satisfacción del cliente a través de un proceso iterativo e incremental, uno de los principios fundamentales de XP es la retroalimentación constante, que se logra a través de la realización de pruebas automáticas continuas y la integración continua del código, esto permite a los equipos detectar y corregir errores de manera rápida y eficiente, garantizando la calidad del software en todo momento, otro aspecto importante de XP es la programación en parejas, donde dos desarrolladores trabajan juntos en el mismo código, revisando y mejorando constantemente el trabajo del otro, esto no solo mejora la calidad del código, sino que también fomenta el intercambio de conocimientos y la colaboración dentro del equipo, XP también enfatiza la importancia de la simplicidad en el diseño y la implementación del software, los equipos de XP buscan minimizar la complejidad innecesaria y mantener el código lo más simple y claro posible, lo que facilita su comprensión, mantenimiento y evolución a lo largo del tiempo, además, XP aborda la incertidumbre y el cambio en los requisitos del cliente mediante la adopción de prácticas como la planificación de liberaciones cortas y frecuentes, que permiten a los equipos adaptarse rápidamente a las necesidades cambiantes del negocio (Rashina Hoda, 2020).

2.4.2. Valores XP

- Comunicación: Fomentar una comunicación abierta y efectiva entre todos los miembros del equipo de desarrollo, así como con los clientes y stakeholders, la comunicación constante y clara ayuda a evitar malentendidos y a mantener a todos en la misma página respecto a los objetivos del proyecto y los requisitos del cliente.
- Simplicidad: Buscar la solución más simple y directa para cada problema, evitando la complejidad innecesaria en el diseño y la implementación del software, la simplicidad facilita la comprensión, el mantenimiento y la evolución del código a lo largo del tiempo.
- Retroalimentación: Obtener retroalimentación continua sobre el trabajo realizado, tanto de los clientes como de los compañeros de equipo, la retroalimentación ayuda a identificar y corregir errores rápidamente, así como a adaptarse a los cambios en los requisitos y las condiciones del mercado.
- Valentía: Estar dispuesto a asumir riesgos y enfrentar los desafíos con determinación y confianza, la valentía implica tomar decisiones difíciles, experimentar con nuevas ideas y estar abierto a aprender de los errores.
- Respeto: Mostrar respeto y consideración hacia todos los miembros del equipo, reconociendo sus habilidades, experiencias y contribuciones, el respeto fomenta un ambiente de trabajo colaborativo y empoderado, donde cada persona se siente valorada y motivada a dar lo mejor de sí misma (Rashina Hoda, 2020).

2.4.3. Fases de la Metodología XP

- Planeación: En esta etapa, se definen los objetivos del proyecto, se identifican los requisitos del cliente y se establece un plan para el desarrollo del software, esto puede incluir la creación de un

cronograma, la asignación de recursos y la identificación de posibles riesgos.

- Diseño: Durante esta fase, se elabora la arquitectura del software y se diseñan las interfaces de usuario, se definen las estructuras de datos, los algoritmos y se crean los diagramas que representan la lógica del sistema, el objetivo es tener una comprensión clara de cómo se verá y funcionará el software antes de pasar a la implementación.
- Codificación: En esta etapa, los desarrolladores escriben el código del software según las especificaciones y el diseño establecido previamente, es el proceso de traducir los diseños y las ideas en un producto de software funcional, esta fase implica programar en el lenguaje de programación elegido y seguir las mejores prácticas de codificación.
- Pruebas: Una vez que se ha completado la codificación, el software se somete a pruebas exhaustivas para detectar errores y garantizar que cumple con los requisitos establecidos, esto puede incluir pruebas de unidad, pruebas de integración y pruebas de aceptación por parte del cliente, el objetivo es identificar y corregir cualquier defecto antes de que el software se ponga en producción.
- Lanzamiento: Una vez que el software ha sido probado y se han corregido los errores, se implementa en el entorno de producción, esto implica la instalación del software en los sistemas de los usuarios finales y la puesta en marcha de este, también puede implicar actividades como la formación del usuario y la creación de documentación (Rashina Hoda, 2020)..

2.5. Ingeniería Web Basada en UML (UWE)

La Ingeniería Web basada en UML (UWE) es un enfoque metodológico que utiliza el Lenguaje Unificado de Modelado (UML) como herramienta principal para el diseño, desarrollo y mantenimiento de aplicaciones web, UWE se centra en la aplicación de los principios de ingeniería de software al contexto específico de la

web, permitiendo a los ingenieros web modelar de manera precisa y completa los aspectos estructurales y de comportamiento de las aplicaciones web.

Esta metodología abarca la modelización de diferentes aspectos de una aplicación web, incluyendo la estructura de la información, la navegación del usuario, la presentación de la interfaz de usuario, la lógica de negocio y la integración de servicios web, al utilizar UML como lenguaje de modelado, UWE facilita la comunicación entre los diferentes miembros del equipo de desarrollo y proporciona una base sólida para la implementación y el mantenimiento de aplicaciones web robustas y escalables (Cáceres García de Marina, 2019).

2.5.1 Diagrama de Casos de Uso

Este diagrama se utiliza para representar las interacciones entre el sistema y sus actores externos, los actores son entidades externas que interactúan con el sistema, mientras que los casos de uso representan las diferentes formas en que estos actores utilizan el sistema para lograr sus objetivos, el diagrama de casos de uso proporciona una vista de alto nivel de las funcionalidades del sistema y ayuda a identificar los requisitos del usuario (Cáceres García de Marina, 2019).

2.5.2. Diagrama de Clases

Este diagrama se utiliza para representar la estructura estática del sistema, mostrando las clases del sistema, sus atributos, métodos y las relaciones entre ellas, las relaciones pueden incluir asociaciones, herencias, agregaciones, entre otras, el diagrama de clases es fundamental para comprender la organización y la arquitectura del sistema, así como para facilitar el diseño y la implementación de software orientado a objetos (Cáceres García de Marina, 2019).

2.6. Modelo Entidad Relación

El Modelo Entidad-Relación (MER) es una técnica de modelado utilizada en el diseño de bases de datos para representar las entidades, sus atributos y las relaciones entre ellas, en el modelo entidad-relación, una entidad representa un

objeto del mundo real o una abstracción del mundo real, como una persona, un lugar o un concepto, cada entidad tiene atributos que describen características específicas de la entidad, las relaciones en el modelo entidad-relación describen cómo las entidades están conectadas entre sí, pueden ser de varios tipos, como uno a uno, uno a muchos o muchos a muchos, y pueden tener restricciones adicionales, como la cardinalidad y la participación, las relaciones ayudan a reflejar las asociaciones y conexiones entre las entidades en el dominio del problema que está siendo modelado, el modelo entidad-relación se representa visualmente mediante diagramas entidad-relación, que muestran las entidades como rectángulos, los atributos como elipses y las relaciones como líneas que conectan las entidades, estos diagramas proporcionan una forma clara y comprensible de visualizar la estructura y las relaciones de una base de datos antes de su implementación (Cano, 2019).

2.7. Calidad de Software

La calidad del software se refiere a la medida en que un producto de software cumple con los requisitos especificados, satisfaciendo las necesidades y expectativas del cliente de manera efectiva y confiable, implica la ausencia de defectos significativos, la capacidad de realizar las funciones previstas de manera precisa y eficiente, así como la capacidad de mantenerse y adaptarse a medida que evolucionan los requisitos y el entorno operativo, la calidad del software no solo se limita a la funcionalidad del producto, sino que también abarca aspectos como la usabilidad, la fiabilidad, el rendimiento y la seguridad (Silvia Abrahäo, 2022).

2.8. Seguridad en el Software

La calidad del software en términos de seguridad se refiere a la capacidad del software para resistir y mitigar ataques cibernéticos, proteger los datos sensibles, y mantener la integridad, confidencialidad y disponibilidad de los sistemas y la información, esto implica la ausencia de vulnerabilidades que puedan ser explotadas por atacantes, así como la implementación de controles y mecanismos de seguridad efectivos que reduzcan los riesgos de exposición a amenazas, una medida de la calidad del software en cuanto a seguridad se basa en la robustez de

sus defensas contra vulnerabilidades comunes, como inyecciones de código, ataques de denegación de servicio, divulgación de información sensible y manipulación de sesiones, además, la calidad del software en seguridad también se evalúa en términos de la capacidad del software para adaptarse y responder a nuevas amenazas mediante actualizaciones y parches oportunos, así como a través de prácticas de desarrollo seguro y pruebas rigurosas de seguridad durante todo el ciclo de vida del software (Ortega Candel, 2020).

2.9. Herramientas de Desarrollo

2.9.1 HTML

HTML (HyperText Markup Language) es el lenguaje estándar utilizado en la creación de páginas web, se trata de un lenguaje de marcado que define la estructura y el contenido de una página web mediante una serie de etiquetas o elementos, cada elemento en HTML está rodeado por etiquetas que indican al navegador web cómo debe mostrar ese contenido en la página, estas etiquetas proporcionan una variedad de funciones, desde definir encabezados, párrafos e imágenes hasta crear enlaces a otras páginas web y formularios interactivos, HTML también ofrece elementos semánticos que ayudan a los motores de búsqueda y otros sistemas a comprender mejor el contenido de la página, lo que contribuye a una mejor accesibilidad y optimización para los motores de búsqueda.(España, 2023).

2.9.2 CSS

CSS (Cascading Style Sheets) es un lenguaje de diseño utilizado para controlar el aspecto visual y la presentación de las páginas web, junto con HTML y JavaScript, CSS es una de las tecnologías fundamentales en el desarrollo web, la principal función de CSS es definir estilos para los elementos HTML, como colores, fuentes, márgenes, alineaciones y tamaños de texto, esto permite separar el contenido de la estructura y el diseño de una página web, lo que facilita la creación de sitios web más atractivos, consistentes y accesibles, CSS utiliza reglas de estilo que se

aplican a los elementos HTML mediante selectores, estas reglas de estilo pueden ser definidas en archivos CSS externos, en línea en el documento HTML o dentro de etiquetas HTML específicas, además, CSS ofrece la posibilidad de crear estilos reutilizables mediante el uso de clases y ID, lo que facilita la aplicación de estilos consistentes en toda la página web, la cascada en CSS se refiere al proceso mediante el cual se resuelven los conflictos de estilos entre diferentes reglas y selectores, la cascada sigue un orden de prioridad específico, que determina qué regla prevalece sobre otras en caso de conflictos, esto permite controlar la apariencia y el diseño de una página web de manera precisa y flexible (España, 2023).

2.9.3 JavaScript

JavaScript es un lenguaje de programación de alto nivel, interpretado y orientado a objetos, que se utiliza principalmente en el desarrollo web para agregar interactividad y dinamismo a las páginas web, junto con HTML y CSS, JavaScript es una de las tecnologías fundamentales en el desarrollo web moderno, la principal función de JavaScript es permitir a los desarrolladores crear aplicaciones web interactivas y dinámicas, mediante la manipulación del Document Object Model (DOM) de la página web, esto permite a los desarrolladores agregar funcionalidades como animaciones, efectos visuales, validación de formularios, interacciones de usuario en tiempo real y comunicación con servidores, entre otras cosas, JavaScript se ejecuta en el navegador web del usuario, lo que significa que puede interactuar con el contenido de la página y responder a eventos del usuario, como clics, desplazamientos y entradas de teclado además, JavaScript también se utiliza en el desarrollo del lado del servidor, mediante el uso de plataformas como Node.js, lo que permite a los desarrolladores crear aplicaciones web completas utilizando un único lenguaje de programación, JavaScript es un lenguaje versátil y flexible que ofrece una amplia gama de funcionalidades y características, incluyendo tipado dinámico, funciones de primera clase, manipulación de arrays y objetos, expresiones regulares, y manejo de errores, su sintaxis está

inspirada en otros lenguajes de programación como Java y C, lo que facilita su aprendizaje para los desarrolladores que ya están familiarizados con estos lenguajes (España, 2023).

2.9.4 TypeScript

TypeScript es un lenguaje de programación de código abierto desarrollado por Microsoft que se basa en JavaScript y amplía su funcionalidad mediante la adición de características como tipado estático, clases, interfaces y módulos, la principal característica de TypeScript es su sistema de tipos estáticos, que permite a los desarrolladores definir tipos de datos para variables, parámetros de funciones y valores de retorno, esto ayuda a detectar errores en tiempo de compilación y a mejorar la robustez y la escalabilidad del código, especialmente en proyectos grandes y complejos, además del tipado estático, TypeScript ofrece otras características que lo hacen más adecuado para el desarrollo de aplicaciones web y de servidor, estas características incluyen el soporte para clases y herencia de clases, interfaces para definir estructuras de datos y contratos de tipo, y módulos para organizar y reutilizar código de manera modular, TypeScript compila a código JavaScript estándar, lo que significa que puede ser ejecutado en cualquier navegador web o entorno que admita JavaScript, esto permite a los desarrolladores utilizar todas las características de TypeScript en el desarrollo de aplicaciones web, sin necesidad de cambiar o actualizar el entorno de ejecución (Talaminos Barroso, 2022).

2.9.5 NodeJs

Node.js es un entorno de ejecución de JavaScript del lado del servidor, basado en el motor de JavaScript V8 de Google Chrome, permite a los desarrolladores crear aplicaciones web y de red escalables y de alto rendimiento utilizando JavaScript como lenguaje de programación tanto en el lado del cliente como en el lado del servidor, una de las características más destacadas de Node.js es su capacidad para realizar operaciones de entrada/salida de manera asíncrona y no bloqueante, esto significa que

puede manejar un gran número de conexiones simultáneas sin bloquear el hilo de ejecución, lo que lo hace ideal para aplicaciones en tiempo real y basadas en eventos, node.js se basa en un modelo de programación basado en eventos y en el sistema de módulos CommonJS, que permite a los desarrolladores organizar su código en módulos reutilizables y mantener una arquitectura modular en sus aplicaciones, además, Node.js cuenta con un ecosistema de paquetes npm (Node Package Manager) que proporciona acceso a miles de bibliotecas y herramientas de código abierto para simplificar el desarrollo de aplicaciones, otra ventaja de Node.js es su compatibilidad con múltiples plataformas, incluyendo Linux, macOS y Windows, lo que permite a los desarrolladores construir aplicaciones web y de red que se ejecuten en una variedad de entornos de implementación ,además, Node.js se integra fácilmente con tecnologías y frameworks populares como Express.js, Socket.io y MongoDB, lo que permite a los desarrolladores crear aplicaciones web completas utilizando un único lenguaje de programación (Alves, 2021).

2.9.6 Express.Js

Express.js es un framework de desarrollo web para Node.js que simplifica la creación de aplicaciones web y APIs (interfaces de programación de aplicaciones), se basa en Node.js y proporciona una capa adicional de abstracción y funcionalidades para simplificar tareas comunes en el desarrollo web, como el enrutamiento, la gestión de peticiones HTTP y la renderización de vistas, express.js es conocido por su simplicidad y su enfoque minimalista, lo que permite a los desarrolladores crear aplicaciones web de manera rápida y eficiente, proporciona un conjunto de funciones y middleware predefinidos que facilitan la creación de rutas, la gestión de sesiones, la autenticación de usuarios y la manipulación de datos, una de las características clave de Express.js es su sistema de enrutamiento, que permite a los desarrolladores definir rutas para manejar diferentes tipos de peticiones HTTP, como GET, POST, PUT y DELETE, esto facilita la creación de APIs RESTful y la gestión de recursos en una

aplicación web, express.js también es altamente personalizable y extensible, lo que permite a los desarrolladores agregar funcionalidades adicionales mediante el uso de middleware y complementos de terceros, además, Express.js se integra fácilmente con otras tecnologías y frameworks populares, como MongoDB, Socket.io y Passport.js, lo que permite a los desarrolladores construir aplicaciones web completas utilizando un conjunto de herramientas coherentes (Express, 2019).

2.9.7 Git

Git es un sistema de control de versiones distribuido ampliamente utilizado en el desarrollo de software, desarrollado por Linus Torvalds en 2005, Git se ha convertido en una herramienta esencial para la gestión eficiente y colaborativa de proyectos de código abierto y propietario la principal característica de Git es su capacidad para realizar un seguimiento preciso de los cambios en el código fuente y facilitar la colaboración entre desarrolladores de manera transparente, una de las ventajas más destacadas de Git es su sistema de ramificación y fusión, que permite a los desarrolladores trabajar en paralelo en diferentes características o correcciones de errores sin interferir en el trabajo de los demás, Git proporciona un historial completo de cambios, lo que facilita la identificación y corrección de errores, así como la auditoría de cambios en el código, git también ofrece una serie de características adicionales que mejoran la eficiencia y la calidad del desarrollo de software, esto incluye la capacidad de realizar confirmaciones atómicas y etiquetar versiones, lo que permite a los equipos mantener un control preciso sobre el estado del proyecto en diferentes etapas de desarrollo, Git es altamente escalable y puede manejar proyectos de cualquier tamaño, desde pequeños proyectos personales hasta grandes proyectos empresariales, el ecosistema de herramientas y servicios basados en Git, como GitHub, GitLab y Bitbucket, proporciona a los desarrolladores una amplia gama de herramientas para la gestión de proyectos, revisión de código, integración y entrega continuas, estas plataformas facilitan la colaboración entre

equipos distribuidos geográficamente y promueven las mejores prácticas en el desarrollo de software (Aguirre, 2022).

2.9.8 Github

GitHub es una plataforma de desarrollo colaborativo basada en la nube que utiliza Git como sistema de control de versiones, fundada en 2008, GitHub ha crecido hasta convertirse en uno de los sitios web más populares para el alojamiento de proyectos de código abierto y privados, así como en una herramienta esencial para la colaboración en el desarrollo de software a nivel mundial, una de las características principales de GitHub es su capacidad para alojar repositorios Git de forma remota, lo que permite a los desarrolladores almacenar, compartir y colaborar en proyectos de software de manera eficiente, además del alojamiento de repositorios, GitHub proporciona una serie de herramientas y características que mejoran la colaboración y la productividad de los equipos de desarrollo, una de las características más utilizadas de GitHub es su sistema de seguimiento de problemas (issue tracking), que permite a los desarrolladores informar sobre errores, solicitar nuevas características y discutir ideas de manera organizada y transparente, esto facilita la comunicación entre los miembros del equipo y fomenta la participación de la comunidad en el desarrollo del proyecto, gitHub también ofrece funcionalidades para la revisión de código (code review), que permiten a los desarrolladores revisar y comentar los cambios propuestos en el código fuente antes de fusionarlos en el repositorio principal, esto ayuda a mejorar la calidad del código y a detectar posibles problemas antes de que se implementen en el producto final, además de estas características, GitHub proporciona herramientas para la integración continua (CI) y la entrega continua (CD), que permiten a los equipos automatizar el proceso de construcción, pruebas y despliegue de sus aplicaciones, esto ayuda a reducir los errores y a acelerar el ciclo de desarrollo, permitiendo a los equipos entregar software de alta calidad de manera rápida y eficiente, el impacto de GitHub en el desarrollo de software ha sido significativo,

democratizando el acceso a herramientas de colaboración de alta calidad y facilitando la contribución a proyectos de código abierto a nivel mundial, su amplia adopción en la comunidad de desarrollo es un testimonio de su utilidad y su papel en la promoción de la transparencia, la colaboración y la innovación en el desarrollo de software (Aguirre, 2022).

2.10 Sistema

Un sistema es un conjunto de elementos interrelacionados que trabajan juntos para lograr un objetivo común, estos elementos pueden incluir personas, procesos, herramientas, tecnologías y recursos físicos que interactúan de manera coordinada para realizar funciones específicas, los sistemas pueden ser de naturaleza física o abstracta, y pueden encontrarse en una amplia variedad de contextos, desde sistemas biológicos y mecánicos hasta sistemas informáticos y sociales, un aspecto fundamental de los sistemas es que exhiben cierto grado de estructura y organización, así como una serie de entradas, procesos y salidas que regulan su funcionamiento, los sistemas pueden ser simples o complejos, dependiendo de la cantidad de componentes que los componen y de la naturaleza de las relaciones entre ellos, la teoría de sistemas proporciona un marco conceptual para comprender cómo funcionan los sistemas, identificando patrones y principios que se aplican a través de diferentes disciplinas y campos de estudio, el análisis de sistemas permite examinar cómo interactúan los diferentes componentes de un sistema, identificar posibles puntos de mejora y diseñar soluciones para optimizar su rendimiento y eficiencia (Carmen de Pablos Heredero, 2019).

2.10.1 Sistema de Información

Un sistema de información es un conjunto organizado de componentes interrelacionados que recopilan, procesan, almacenan y distribuyen datos para apoyar las actividades de una organización, empresa o entidad, estos componentes pueden incluir hardware, software, bases de datos, redes, personas y procedimientos que trabajan en conjunto para recopilar información, transformarla en datos procesables y proporcionar salidas útiles para la toma de decisiones y el cumplimiento de objetivos

organizacionales, los sistemas de información pueden abarcar una amplia gama de funciones y áreas de aplicación, como la gestión de recursos humanos, la contabilidad, la producción, la logística, el marketing, entre otros, estos sistemas pueden ser transaccionales, es decir, centrados en el procesamiento de transacciones comerciales, o pueden ser sistemas de soporte a la toma de decisiones, que proporcionan información analítica y reportes para ayudar a los gerentes y líderes en la toma de decisiones estratégicas, la implementación y gestión efectiva de un sistema de información puede mejorar la eficiencia operativa, aumentar la productividad, facilitar la comunicación interna y externa, y proporcionar una ventaja competitiva en el mercado, sin embargo el diseño y la implementación de un sistema de información eficaz requiere una cuidadosa planificación, coordinación y gestión de recursos para garantizar que cumpla con los requisitos y necesidades específicas de la organización (Carmen de Pablos Heredero, 2019).

2.10.2 Sistema Transaccional

Un sistema transaccional es una infraestructura tecnológica diseñada para facilitar y gestionar transacciones comerciales o financieras, estas transacciones pueden incluir compras, ventas, pagos, transferencias de fondos, entre otras actividades relacionadas con la operación de una organización, los sistemas transaccionales son fundamentales para garantizar la precisión, integridad y seguridad de las transacciones comerciales en tiempo real o casi real, estos sistemas suelen estar respaldados por bases de datos transaccionales que registran y gestionan cada transacción de manera individual, asegurando que se completen de manera adecuada y que se mantenga la consistencia de los datos en todo momento, demás, los sistemas transaccionales suelen estar diseñados para manejar grandes volúmenes de transacciones de manera eficiente, garantizando un procesamiento rápido y confiable de las operaciones comerciales (Fandiño, 2023).

2.11. Aplicación Web

Una aplicación web es un software alojado en servidores y accesible a través de un navegador web, a diferencia de las aplicaciones de escritorio que se instalan localmente en un dispositivo, las aplicaciones web se ejecutan en servidores remotos y son accesibles a través de Internet, estas aplicaciones web pueden ofrecer una amplia gama de funcionalidades y servicios, desde herramientas de productividad como procesadores de texto y hojas de cálculo, hasta aplicaciones empresariales complejas como sistemas de gestión de relaciones con clientes (CRM) o sistemas de gestión de contenido (CMS), las aplicaciones web pueden ser estáticas o dinámicas, las estáticas son aquellas que muestran contenido fijo y no requieren interacción con el usuario más allá de la navegación por las páginas, las dinámicas, por otro lado, pueden interactuar con el usuario, procesar datos en tiempo real, y adaptarse a las acciones del usuario, para desarrollar aplicaciones web se utilizan una variedad de tecnologías como HTML, CSS y JavaScript para el desarrollo del frontend (interfaz de usuario), y lenguajes de programación como PHP, Python o Ruby para el desarrollo del backend (Express, 2019).

2.12 Arquitecturas de Software

La arquitectura de software se refiere a la estructura organizativa y conceptual de un sistema de software, que define la forma en que sus componentes están diseñados, interactúan y se relacionan entre sí, esta estructura proporciona un marco para el desarrollo, la implementación y el mantenimiento del software, estableciendo patrones, principios y directrices que guían su diseño y evolución, la arquitectura de software aborda aspectos como la distribución de responsabilidades, la separación de preocupaciones, el modularidad, la escalabilidad, el rendimiento y la seguridad del sistema, entre otros, en última instancia, una arquitectura de software bien definida y diseñada contribuye a la creación de sistemas de software eficientes, robustos y adaptables que cumplen con los requisitos y objetivos del proyecto (Zhamak Dehghani, 2023).

2.12.1 Clean Architecture

La Clean Architecture, o Arquitectura Limpia, es un enfoque de diseño de software propuesto por Robert C. Martin que se centra en la creación de sistemas de software altamente mantenibles, flexibles y sostenibles a lo largo del tiempo, esta arquitectura promueve la separación de preocupaciones y la alta cohesión entre los componentes del sistema, lo que facilita su comprensión, prueba y evolución, en la Clean Architecture, el diseño del sistema se organiza en capas concéntricas, con las entidades de negocio en el centro, rodeadas por capas de casos de uso, interfaces de usuario y adaptadores de infraestructura externa, estas capas están diseñadas de manera que las dependencias fluyan hacia adentro, desde los detalles de implementación hacia las políticas y reglas de negocio del sistema, al seguir los principios de la Clean Architecture, los desarrolladores pueden crear sistemas de software que sean independientes de frameworks, bases de datos y tecnologías externas, lo que los hace más flexibles y adaptables a los cambios en los requisitos del negocio y en el entorno tecnológico, además, esta arquitectura fomenta la reutilización de código, el modularidad y la escalabilidad del sistema, lo que contribuye a un desarrollo de software más eficiente y sostenible a largo plazo (Walker, 2021).

2.13. Clean Code

Clean Code, o Código Limpio, representa un enfoque de desarrollo de software que se centra en escribir código claro, legible y fácil de entender y mantener, el objetivo fundamental de Clean Code es producir software de alta calidad que sea comprensible, modificable y extensible, lo que contribuye a la creación de sistemas más robustos y menos propensos a errores, este enfoque pone énfasis en principios y prácticas que promueven la claridad y la simplicidad en el código, lo que facilita su comprensión por parte de los desarrolladores y su mantenimiento a lo largo del tiempo, uno de los principios fundamentales de Clean Code es utilizar

nombres descriptivos para las variables, funciones y clases, de modo que reflejen claramente su propósito y función en el código, esta práctica ayuda a los desarrolladores a entender rápidamente el propósito de cada componente y a navegar eficientemente por el código, además, Clean Code promueve la escritura de funciones pequeñas y cohesivas, que realicen una sola tarea y la realicen bien, esto ayuda a reducir la complejidad del código, evitando la duplicación y facilitando su comprensión y mantenimiento, otro aspecto importante de Clean Code es el uso de comentarios claros y concisos para explicar el propósito y el funcionamiento del código, los comentarios bien escritos pueden ayudar a los desarrolladores a comprender el contexto y la intención detrás de ciertas partes del código, lo que facilita su comprensión y modificación, Clean Code promueve el mantenimiento de un formato consistente en todo el proyecto, utilizando convenciones de estilo y de formato que faciliten la lectura y comprensión del código, eliminar código redundante y escribir pruebas automatizadas también son aspectos clave de Clean Code, al eliminar código muerto o redundante, se reduce la complejidad del sistema y se minimiza el riesgo de errores, por otro lado, escribir pruebas automatizadas ayuda a validar el comportamiento del código y garantizar su correcto funcionamiento en diferentes escenarios, lo que aumenta la confianza en la calidad del software, estos principios y prácticas de Clean Code contribuyen a la creación de software de alta calidad que es fácil de entender, modificar y mantener a lo largo del tiempo (Media, 2024).

2.14 Base de Datos

Una base de datos es un conjunto organizado de datos relacionados entre sí y almacenados de forma estructurada en un sistema informático, estos datos pueden ser de diversos tipos, como textos, números, imágenes o multimedia, y están diseñados para ser accedidos, gestionados y actualizados de manera eficiente, la estructura de una base de datos generalmente se define mediante un esquema que especifica los tipos de datos, las relaciones entre ellos y las restricciones que deben cumplirse para garantizar la integridad y consistencia de los datos, las bases de datos son ampliamente utilizadas en una variedad de aplicaciones y sectores,

desde la gestión de empresas y organizaciones hasta la investigación científica y el entretenimiento, proporcionan un medio centralizado para almacenar y gestionar información, facilitando el acceso rápido y preciso a los datos, así como la realización de análisis y procesamiento de la información para obtener conocimientos útiles (Pulido Romero, 2019).

2.14.1 Bases de Datos Relacionales

Las bases de datos relacionales son un tipo de sistema de gestión de bases de datos (SGBD) que se basa en el modelo relacional, propuesto por Edgar F. Codd en la década de 1970, en este modelo, los datos se organizan y se acceden a través de tablas que contienen filas y columnas, cada tabla representa una entidad y sus atributos, y las relaciones entre las entidades se establecen mediante claves primarias y claves externas, las bases de datos relacionales se caracterizan por su estructura flexible y su capacidad para gestionar datos complejos de manera eficiente, utilizan un lenguaje de consulta estándar, como SQL (Structured Query Language), para realizar operaciones como la inserción, modificación, eliminación y consulta de datos, la arquitectura relacional permite la normalización de datos, lo que reduce la redundancia y mejora la integridad y consistencia de la información almacenada, las bases de datos relacionales son ampliamente utilizadas en aplicaciones empresariales, sistemas de gestión de contenido, sistemas de información geográfica, entre otros, debido a su capacidad para manejar grandes volúmenes de datos y su flexibilidad para adaptarse a diferentes necesidades de negocio (O'Donnell, 2020).

2.14.2 SQL (Structured Query Language)

SQL es un lenguaje de programación utilizado para gestionar y manipular bases de datos relacionales, fue desarrollado originalmente por IBM en la década de 1970 y se ha convertido en el estándar de facto para interactuar con bases de datos relacionales,

SQL proporciona un conjunto de comandos y sintaxis estandarizados que permiten realizar diversas operaciones en una base de datos, como la creación y modificación de tablas, la inserción, actualización y eliminación de datos, y la realización de consultas para recuperar información específica de la base de datos, el lenguaje SQL se compone de varios tipos de instrucciones, incluyendo DDL (Data Definition Language) para definir la estructura de la base de datos, DML (Data Manipulation Language) para realizar operaciones sobre los datos, DQL (Data Query Language) para realizar consultas, y DCL (Data Control Language) para gestionar los permisos de acceso a la base de datos, SQL es ampliamente utilizado en una variedad de aplicaciones y entornos, desde sistemas empresariales y aplicaciones web hasta análisis de datos y procesamiento de transacciones, su capacidad para manejar grandes volúmenes de datos y su facilidad de uso lo convierten en una herramienta invaluable para la gestión de bases de datos relacionales (Fehily, 2020).

2.14.3 MYSQL

MySQL es un sistema de gestión de bases de datos relacional (RDBMS, por sus siglas en inglés) de código abierto ampliamente utilizado en el desarrollo de aplicaciones web y empresariales, desarrollado inicialmente por MySQL AB (adquirida posteriormente por Sun Microsystems y luego por Oracle Corporation), MySQL es conocido por su rendimiento, confiabilidad y facilidad de uso, como un sistema de gestión de bases de datos relacional, MySQL organiza los datos en tablas que se relacionan entre sí mediante claves primarias y claves externas, utiliza un lenguaje de consulta estructurado (SQL) para realizar operaciones como inserción, actualización, consulta y eliminación de datos en estas tablas, MySQL es compatible con una amplia gama de sistemas operativos, incluyendo Linux, Windows y

macOS, y se integra fácilmente con lenguajes de programación como PHP, Python, Java y JavaScript, entre otros.

MySQL ofrece una variedad de herramientas y características, incluyendo replicación, particionamiento, seguridad avanzada y optimización de consultas, que lo hacen adecuado para una amplia variedad de aplicaciones y escenarios de uso, MySQL está disponible en varias ediciones, incluyendo la edición comunitaria de código abierto (MySQL Community Server) y las ediciones comerciales que ofrecen características adicionales y soporte técnico, MySQL forma parte de la plataforma LAMP (Linux, Apache, MySQL, PHP/Perl/Python), una pila de software comúnmente utilizada para el desarrollo de aplicaciones web (Herwin Alayn Huillcen Baca, 2022).

2.14.4. MariaDB

MariaDB es un sistema de gestión de bases de datos relacional (RDBMS) de código abierto que se deriva de MySQL, fue desarrollado por los creadores originales de MySQL después de que esta última fuera adquirida por Oracle Corporation en 2009, con el objetivo de proporcionar una alternativa comunitaria y compatible con MySQL, MariaDB comparte muchas similitudes con MySQL en términos de sintaxis SQL, estructura de datos y características, lo que hace que la migración de aplicaciones de MySQL a MariaDB sea relativamente sencilla, además, MariaDB ha seguido evolucionando con el tiempo, incorporando nuevas características y mejoras de rendimiento que no están presentes en MySQL, Algunas de las características distintivas de MariaDB incluyen el almacenamiento de datos en tablas InnoDB por defecto (antes conocido como XtraDB), compatibilidad mejorada con estándares SQL y la inclusión de nuevas características como columnas virtuales y tablas de temporales, MariaDB ha enfocado esfuerzos en mejorar la seguridad y la escalabilidad del sistema,

MariaDB se distribuye bajo los términos de la Licencia Pública General de GNU (GPL), lo que significa que es de código abierto y gratuito para su uso y distribución, MariaDB cuenta con una activa comunidad de desarrolladores y usuarios que contribuyen al desarrollo del proyecto y proporcionan soporte y recursos adicionales (Blokdyk, 2019).

3. Marco Metodológico

3.1. Metodología de Investigación

3.1.1. Método Científico

El método científico se aplicará en este proyecto comenzando con la observación y descripción detallada de los problemas operativos del Comité de Agua Potable de Rancho Norte, seguido de la formulación de hipótesis sobre cómo un sistema de información podría abordar estos desafíos, luego, se diseñará y planificará el sistema, recopilando datos relevantes y desarrollando el sistema en función de los requisitos identificados, posteriormente, se procederá con la implementación del sistema en el entorno del comité, seguida de pruebas exhaustivas para evaluar su funcionamiento, finalmente, se analizarán los resultados obtenidos tras la implementación del sistema para extraer conclusiones sobre su eficacia en la mejora de la gestión del agua potable, siguiendo un enfoque sistemático y riguroso para obtener resultados significativos.

3.1.2. Investigación Exploratoria

La investigación exploratoria se aplicará en este proyecto para comprender más a fondo el contexto y los posibles enfoques para abordar los problemas identificados en la gestión del agua potable por parte del Comité de Agua Potable de Rancho Norte, esto implicará la revisión de literatura relacionada con sistemas de gestión de agua potable, tecnologías de información y comunicación aplicadas a servicios públicos, así como casos de estudio de implementaciones similares en otras comunidades, además, se llevarán a cabo entrevistas con expertos en el campo de la gestión del agua potable y sistemas de información para recopilar información cualitativa sobre mejores prácticas, desafíos y lecciones aprendidas, esta investigación exploratoria proporcionará una base sólida para la definición de requisitos y la toma de decisiones en las etapas posteriores del proyecto.

3.1.3. Investigación Descriptiva

La investigación descriptiva se aplicará en este proyecto para obtener una comprensión detallada de la situación actual del Comité de Agua Potable de Rancho Norte y su entorno operativo, esto implicará la recopilación de datos cuantitativos y cualitativos sobre aspectos específicos de la gestión del agua potable, como la cantidad de usuarios atendidos, los patrones de consumo de agua, los procesos administrativos actuales y los recursos disponibles, se utilizarán métodos como encuestas, entrevistas estructuradas y análisis documental para recopilar estos datos de manera sistemática, la investigación descriptiva nos permitirá generar perfiles detallados de la situación actual del comité, lo que servirá como punto de referencia para el diseño y la implementación del sistema de información.

3.2. Metodología de la ingeniería

3.2.1. Programación Extrema (XP)

La Programación Extrema (XP) se aplicaría en este proyecto mediante un enfoque iterativo e incremental, pruebas automatizadas, integración continua, refactorización constante y colaboración activa para desarrollar un sistema de información que satisfaga las necesidades del Comité de Agua Potable de Rancho Norte de manera efectiva y eficiente.

3.2.2. Ingeniería web Basada en UML (UWE)

En este proyecto, la aplicación de la Ingeniería Web basada en UML implicaría la representación de los requisitos, la estructura arquitectónica y el flujo operativo del sistema, se emplearía para la generación de código y la creación de casos de prueba, este enfoque contribuiría a asegurar que el sistema final satisfaga las necesidades y expectativas del Comité de Agua Potable de Rancho Norte"

Referencias

(CAPRN), E. C. (2015). *Reglamento interno CAPRN.* San Lorenzo, Tarija: El Comité de Agua Potable Rancho Norte (CAPRN) .

Aguirre, S. (2022). *git GitHub.* n.p.: RedUSERS.

Alves, C. (2021). *Node. js: NODEJS para Principiantes.* Independently Published.

Blokdyk, G. (2019). *MariaDB a Complete Guide - 2019 Edition.* n.p.: Emereo Pty Limited.

Cáceres García de Marina, P. G. (2019). *Especificando software mediante casos de USO y UML.* España: Editorial Universitaria Ramón Areces.

Cano, I. M. (2019). *Ingeniería de requisitos.* España: Editorial Universidad de Almería.

Carmen de Pablos Heredero, J. J.-R. (2019). *Organización y transformación de los sistemas de información en la empresa.* España: ESIC Editorial.

España. (2023). *Construcción y diseño de páginas web con HTML, CSS y JavaScript. Edición 2023.* España: Ra-Ma S.A. Editorial y Publicaciones.

Express, W. D. (2019). *Express. Js: Web App Development with Node. Js Framework.* n.p: Independently Published.

Fandiño, V. M. (2023). *Sistemas de Big Data.* España: Ra-Ma S.A. Editorial y Publicaciones.

Fehily, C. (2020). *SQL Database Programming (Fifth Edition).* n.p.: Questing Vole Press.

Gómez Palomo, S. G. (2020). *Aproximación a la ingeniería del software.* España: Centro de Estudios Ramón Areces.

Herwin Alayn Huillcen Baca, F. d. (2022). *Introducción a las Bases de Datos con MySQL.* n.p.: Herwin Alayn Huillcen Baca.

Media, M. (2024). *Summary of Martin Robert C.'s Clean Code.* n.p.: Milkyway Media.

O'Donnell, T. (2020). *Design and Use of Relational Databases in Chemistry.* Reino Unido: Taylor & Francis Limited (Sales).

Ortega Candel, J. M. (2020). *Desarrollo seguro en ingeniería del software: Aplicaciones seguras con Android, NodeJS, Python y C++.* Colombia: Alpha Editorial.

Porras, A. A. (2023). *Metodologías ágiles para el desarrollo de software.* Colombia: Editorial Universidad Distrital Francisco José de Caldas.

Pulido Romero, E. D. (2019). *Bases de datos.* México: Patria Educación.

Rashina Hoda, M. P. (2020). *Agile Processes in Software Engineering and Extreme Programming.* Alemania: Springer International Publishing.

Silvia Abrahäo, C. C. (2022). *Calidad y sostenibilidad de sistemas de información en la práctica.* España: RA-MA S.A. Editorial y Publicaciones.

Talaminos Barroso, A. (2022). *TypeScript para todo.* España: Books on Demand.

Walker, J. (2021). *Clean Architecture.* n.p: Independently Published.

Zhamak Dehghani, N. F. (2023). *Arquitectura de software: las partes difíciles. Análisis moderno de ventajas y desventajas para arquitecturas distribuidas.* España: ANAYA MULTIMEDIA.