UNIVERSIDAD TÉCNICA DE AMBATO

FACULTAD DE INGENIERÍA EN SISTEMAS ELECTRÓNICA E INDUSTRIAL

CARRERA DE INGENIERÍA EN SISTEMAS COMPUTACIONALES E INFORMÁTICOS

Tema:

APLICACIÓN WEB PARA AUTOMATIZAR EL PROCESO DE RECAUDACIÓN DE VALORES Y CONSULTA DE PLANILLAS MENSUALES POR CONSUMO DE AGUA POTABLE EN LA COOPERATIVA DE AHORRO Y CRÉDITO SAN MARTÍN DE TISALEO LTDA.

Trabajo de Titulación. Modalidad: Proyecto de Investigación, presentado previo a la obtención del título de Ingeniero en Sistemas Computacionales e Informáticos

ÁREA: Software
LÍNEA DE INVESTIGACIÓN: Desarrollo de Software

AUTOR: Christian Gustavo Verdesoto Guaman
TUTOR: Ing. David Omar Guevara Aulestia, Mg.

Ambato - Ecuador
marzo - 2023

APROBACIÓN DEL TUTOR

En calidad de Tutor del Trabajo de Titulación con el Tema: APLICACIÓN WEB PARA AUTOMATIZAR EL PROCESO DE RECAUDACIÓN DE VALORES Y CONSULTA DE PLANILLAS MENSUALES POR CONSUMO DE AGUA POTABLE EN LA COOPERATIVA DE AHORRO Y CRÉDITO SAN MARTÍN DE TISALEO LTDA., desarrollado en la modalidad Proyecto de Investigación por el señor Christian Gustavo Verdesoto Guaman, estudiante de la Carrera de Ingeniería en Sistemas Computacionales e Informáticos, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, de la Universidad Técnica de Ambato, me permito indicar que el estudiante ha sido tutorado durante todo el desarrollo del trabajo hasta su conclusión, de acuerdo a lo dispuesto en el Artículo 15 del Reglamento para obtener el Título de Tercer Nivel, de Grado de la Universidad Técnica de Ambato, y el numeral 7.4 del respectivo instructivo.

Ambato, marzo 2023

Firmado electrónicamente por:
DAVID OMAR GUEVARA AULESTIA

Ing. David Omar Guevara Aulestia, Mg.

TUTOR

AUTORÍA

El presente Proyecto de Investigación titulado: APLICACIÓN WEB PARA AUTOMATIZAR EL PROCESO DE RECAUDACIÓN DE VALORES Y CONSULTA DE PLANILLAS MENSUALES POR CONSUMO DE AGUA POTABLE EN LA COOPERATIVA DE AHORRO Y CRÉDITO SAN MARTÍN DE TISALEO LTDA. es absolutamente original, auténtico y personal. En tal virtud, el contenido, efectos legales y académicos que se desprenden del mismo son de exclusiva responsabilidad del autor.

Ambato, marzo 2023

Christian Gustavo Verdesoto Guaman

C.C. 1805140793

AUTOR

DERECHOS DE AUTOR

Autorizo a la Universidad Técnica de Ambato, para que haga uso de este Trabajo de Titulación como un documento disponible para la lectura, consulta y procesos de investigación.

Cedo los derechos de mi Trabajo de Titulación en favor de la Universidad Técnica de Ambato, con fines de difusión pública. Además, autorizo su reproducción total o parcial dentro de las regulaciones de la institución.

Ambato, marzo 2023

Christian Gustavo Verdesoto Guaman

C.C. 1805140793

AUTOR

APROBACIÓN DEL TRIBUNAL DE GRADO

En calidad de par calificador del Informe Final del Trabajo de Titulación presentado por el señor Christian Gustavo Verdesoto Guaman, estudiante de la Carrera de Ingeniería en Sistemas Computacionales e Informáticos, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, bajo la Modalidad Proyecto de Investigación, titulado APLICACIÓN WEB PARA AUTOMATIZAR EL PROCESO DE RECAUDACIÓN DE VALORES Y CONSULTA DE PLANILLAS MENSUALES POR CONSUMO DE AGUA POTABLE EN EL COOPERATIVA DE AHORRO Y CRÉDITO SAN MARTÍN DE TISALEO LTDA., nos permitimos informar que el trabajo ha sido revisado y calificado de acuerdo al Art. 17 del Reglamento para obtener el título Terminal de Tercer Nivel, de Grado de la Universidad Técnica de Ambato, y al numeral 7.6 del respectivo instructivo. Para cuya constancia suscribimos, conjuntamente con la señora Presidenta del Tribunal.

Firmado electrónicamente por:
ELSA PILAR URRUTIA URRUTIA

Ing. Pilar Urrutia, Mg.

PRESIDENTA DEL TRIBUNAL

Firmado electrónicamente por:
CARLOS ISRAEL NUNEZ MIRANDA

Ing. Carlos Nuñez
PROFESOR CALIFICADOR

Firmado electrónicamente por:
DENNIS VINICIO CHICAIZA CASTILLO

Ing. Dennis Chicaiza
PROFESOR CALIFICADOR

DEDICATORIA

El presente trabajo de investigación es dedicado en primer lugar a Dios por la fuerza y sabiduría otorgada a lo largo de este camino.

A mis padres Ana y Washington por las enseñanzas de vida que, en su momento me inculcaron.

A mis hermanas, por alentarme en cada momento que lo necesite, por brindarme su apoyo emocional en el hogar, de manera especial a Lizbeth por estar conmigo en todo momento. ¡Hermana lo logré!

Por último, a toda mi familia, quienes confiaron en mí para formarme como profesional y persona, por sus consejos que me inspiraron a seguir superándome día a día.

Christian Gustavo Verdesoto Guaman

AGRADECIMIENTO

Agradezco a Dios por las abundantes bendiciones que derramó todos los días de mi vida y por la salud brindada a toda mi familia.

A los docentes, quienes en las aulas impartieron conocimientos, que influyeron en mi formación académica, de manera especial a mi tutor Ing. David Guevara un agradecimiento profundo por su disponibilidad y guía constante que me permitieron culminar el presente proyecto de investigación.

A la Cooperativa de Ahorro y Crédito "San Martín", muy particularmente al gerente general Ing. Francisco Moreta por brindarme el apoyo total para el desarrollo de mi proyecto.

Por último, agradezco a mis amigos y compañeros quienes se convirtieron en mi segunda familia y compartieron grandes momentos que los guardare en mi corazón.

Christian Gustavo Verdesoto Guaman

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN EJECUTIVO

El presente proyecto fue desarrollado con el objetivo de automatizar los procesos que participan en la emisión de planillas mensuales, con el fin de proceder con la recaudación de valores en la institución, que permitirá incrementar la afluencia de clientes que buscan cumplir sus obligaciones en un solo lugar, ya que la institución cuenta con sistemas externos para el cobro de otros servicios básicos como luz, teléfono entre otras.

La aplicación web se divide en componentes backend y frontend, para el primero se utilizó entity framework de ASP.NET en su versión 3.1, en el cual se creó varios servicios web, los mismos que son consumidos por el frontend que está desarrollado en base al framework Angular.

Finalmente, la metodología de desarrollo que se utilizó en el presente proyecto fue Xtreme Programming (XP), lo cual permitió gestionar tiempos de entrega, mediante pruebas de ciclos rápidos continuos e incrementales para conseguir los resultados esperados por el cliente.

Palabras clave: Angular, SQL Server, entity framework, backend, frontend

ABSTRACT

This project was developed with the aim of automating the processes involved in the issuance of monthly payrolls, in order to proceed with the collection of securities in the institution, which will increase the influx of customers seeking to fulfill their obligations in one place, since the institution has external systems for the collection of other basic services such as electricity, telephone among others.

The web application is divided into backend and frontend components, for the first ASP.NET entity framework was used in its version 3.1, in which several web services were created, the same ones that are consumed by the frontend that is developed based on the Angular framework.

Finally, the development methodology used in this project was Xtreme Programming (XP), which allowed to manage delivery times, through tests of continuous and incremental fast cycles to achieve the results expected by the client.

Keywords: Angular, SQL Server, entity framework, backend, frontend

INTRODUCCIÓN

En el presente proyecto de investigación denominado "APLICACIÓN WEB PARA AUTOMATIZAR EL PROCESO DE RECAUDACIÓN DE VALORES Y CONSULTA DE PLANILLAS MENSUALES POR CONSUMO DE AGUA POTABLE EN LA COOPERATIVA DE AHORRO Y CRÉDITO SAN MARTÍN DE TISALEO LTDA.", se encuentra dividido en los capítulos expuestos a continuación.

CAPITULO I: En este capítulo se plantea el tema de investigación, sus antecedentes y lo más importante, sus objetivos a cumplir durante el desarrollo del proyecto.

CAPITULO II: En este capítulo se incluyen los materiales, modalidad de investigación y los resultados de la entrevista que aportaron al planteamiento de una solución tecnológica.

CAPÍTULO III: En este capítulo se especifica el proceso del desarrollo del proyecto, herramientas y tecnologías aplicadas en el desarrollo del proyecto.

CAPITULO IV: En este capítulo se describen las conclusiones y recomendaciones encontradas durante el desarrollo del proyecto.

CAPÍTULO I

MARCO TEÓRICO

1.1. Tema de Investigación

APLICACIÓN WEB PARA AUTOMATIZAR EL PROCESO DE RECAUDACIÓN DE VALORES Y CONSULTA DE PLANILLAS MENSUALES POR CONSUMO DE AGUA POTABLE EN LA COOPERATIVA DE AHORRO Y CRÉDITO SAN MARTÍN DE TISALEO LTDA.

1.2. Antecedentes Investigativos

La presente investigación está basada en los siguientes antecedentes investigativos, los mismos que fueron obtenidos de la revisión bibliográfica de repositorios digitales en Universidades Ecuatorianas, las fuentes son trabajos relacionados con el tema de investigación, luego se menciona los aportes investigativos más relevantes.

Según Jefferson Torres en su tesis "APLICACIÓN MÓVIL MULTIPLATAFORMA PARA LA GESTIÓN DE INFORMACIÓN GEORREFERENCIAL Y SERVICIO TÉCNICO COMUNITARIO DE PLOMERÍA, APLICANDO GEOLOCALIZACIÓN OFFLINE, EN LA JUNTA ADMINISTRADORA DE AGUA POTABLE DE LOS BARRIOS OCCIDENTALES DE ALAUSÍ" trabajo realizado como tesis en la Universidad Técnica de Ambato. En el año 2021 manifiesta que la aplicación móvil está enfocada a mejorar la calidad de difusión de la información y la atención en el servicio de asistencia técnica comunitaria que ofrece la institución, además el autor recalca la importancia de la metodología de desarrollo ágil Extreme Programming(XP) que le ha permitido optimizar los tiempos entrega y desarrollo. Torres recomienda investigar nuevas tecnologías de desarrollo o frameworks, que se encuentran en auge como React, Flutter y Angular que favorecen a la investigación [1].

Según Jean Toa en su tesis "SISTEMA PARA LA RECAUDACIÓN DE TARIFAS POR EL SUMINISTRO DE AGUA POTABLE EN LA JUNTA ADMINISTRADORA DE AGUA LAS AMÉRICAS CANTÓN Y PROVINCIA DE PASTAZA" trabajo realizado como tesis en le Universidad Regional

Autónoma de los Andes. En el año 2017 evidencia la utilización de una metodología de desarrollo como lo es SCRUM, que le permitió mantener contacto directo entre el desarrollador y el cliente, con el objetivo de minimizar confusiones de procesos y estableciendo una planificación por cada módulo programado. En cuanto al sistema desarrollado para el administrador resulta muy versátil y confiable puesto que es una aplicación web lo cual permite reducir el consumo de memoria y almacenamiento del ordenador. Así mismo el sistema lleva un registro actualizado de todos los procesos realizados en relación a los cobros de tarifas, sutituyendo los procesos manuales y reduciendo inconsistencias en la información [2].

Según Kerly Chabla en su tesis "IMPLEMENTACIÓN DE UN SISTEMA WEB DE FACTURACIÓN Y CONSULTA PARA LA JUNTA ADMINISTRADORA DE AGUA POTABLE DE MOBILOIL" trabajo realizado como tesis en la Universidad Tecnológica Indoamérica. En el año 2017 menciona la importancia de establecer tarifas especiales que permitan, respetar los derechos de las personas de tercera edad, discapacitados igualmente en la generación de reportes mensuales de las recaudaciones, para lo cual el desarrollo del sistema esta basado en la estructura 3 capas, Net Framework y como motor de base de datos SQL Server, el sistema web fue diseñado para proveer reportes de valores recaudados por conceptos de: consumo de agua potable, deudas por cobrar, multas y descuentos con el fin de mantener un registro y control de la facturación de clientes [3].

1.2.1. Contextualización del problema

El agua es uno de los recursos naturales más valiosos para todos los seres vivos de este planeta, por ende debe ser gestionado con responsabilidad social con el fin de evitar que escasee este recurso hídrico.

El Ecuador dispone de abundante cantidad de agua, distribuida de manera irregular, la cobertura de agua potable ha aumentado en los últimos años; sin embargo en las comunidades rurales existe una baja calidad, ineficiencia en el servicio y una inadecuada recuperación de costos [4].

La Secretaría Nacional del Agua (SENAGUA) es el ente encargado de asesorar, financiar y construir pequeñas plantas de tratamiento de agua, una vez construida la planta potabilizadora la directiva elegida en la Junta Administradora de Agua Potable (JAAP) administra, comercializa y distribuye transformándose en un ente

autónomo no recibe recursos económicos del estado. Las plantas potabilizadoras no cuentan con instrumentos de medición que les permita verificar el estado del agua, peor aún un sistema de monitorización y control permanente en los subprocesos del sistema de potabilización como la dosificación de líquidos, el tanque de reserva, por los costos elevados de los equipos y el desconocimiento de alternativas [5].

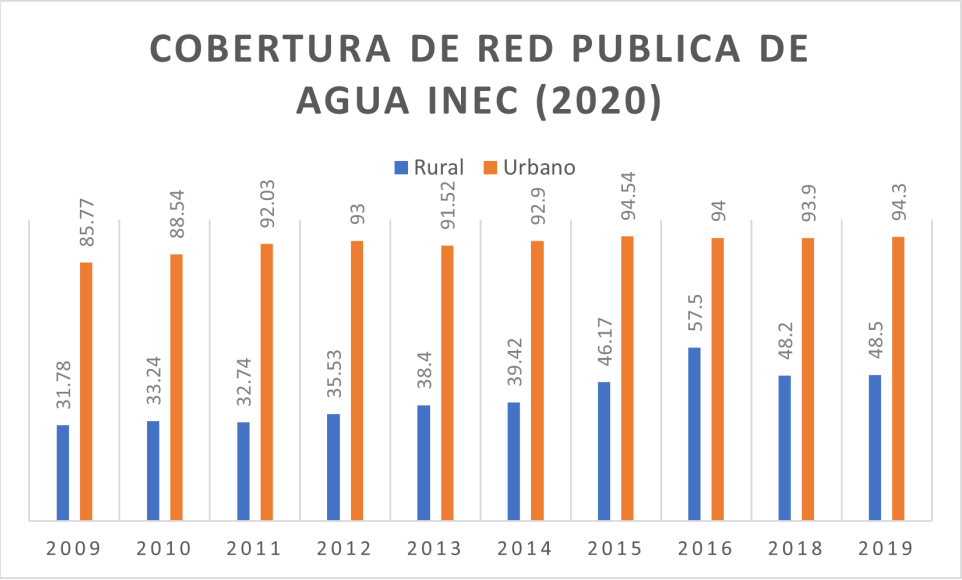


Figura 1.1: Medición de los indicadores de agua, saneamiento e higiene en el Ecuador

Elaborado por: INEC (2020)

El 35,30 % de las instituciones llevan sus registros de forma manual. Además, no disponen de un computador para almacenar información en formato digital. Considerando que el 64,60 % de las juntas que operan hace más de 10 años, ya disponen de un sistema de información, pero algunos sistemas no son escalables para cubrir las necesidades y requerimientos de los procesos administrativos y legales [6].

En las juntas de Agua Potable y Saneamiento de la Provincia Tungurahua no cuentan con capacitaciones que en la gran mayoría de temas cubre el Proceso Administrativo lo cual hace que los directivos no cumplan sus tareas adecuadamente [7], además que carecen de manuales de operación y mantenimiento de sistemas. A esto se suma la escasa supervisión de SENAGUA.

En la Provincia de Tungurahua existen más de 200 Juntas Administradoras de Agua Potable dispersas en los cantones de las cuales apenas el 10 % cuentan con un sistema informático para el cobro de tarifas, la gran mayoría de las Juntas de Agua Potable realizan los cobros tarifarios de forma manual lo que ocasiona: proceso de recaudación inexacto, deficiente atención a los contribuyentes, no obtención de los resultados deseados en el momento oportuno, etc. [8].

En la actualidad la gestión del servicio de agua potable es administrada por los Gobiernos Autónomos Descentralizados (GAD), sin embargo, para el sector rural, existen otras empresas u organizaciones que realizan la gestión del servicio de agua potable. A medida que las tecnologías mejoran, las empresas requieren de mayor tecnificación para implementar herramientas que faciliten la cobranza del servicio y toma de lecturas de los consumos de agua potable dentro de su jurisdicción. Por consiguiente esta investigación plantea como objetivo desarrollar una aplicación web que se ajuste a las necesidades y requerimientos de la institución.

1.2.2. Fundamentación teórica

1.2.2.1. Sistema de Información

Conjunto formal de procesos que, operando sobre una colección de datos estructurada de acuerdo a las necesidades de la empresa, recopila, elabora y distribuyen selectivamente la información necesaria para la operación de dicha empresa y para las actividades de dirección y control correspondientes, apoyando, al menos en parte, los procesos de toma de decisiones necesarios para desempeñar funciones de negocio de la empresa de acuerdo con su estrategia [9].

El sistema de información que opera sobre un conjunto de datos estructurado, los cuales son almacenados, procesados y como un resultado final se muestra al grupo de usuarios que tienen acceso al sistema. Asi mismo la información sirve como retroalimentación o feedback, que servirá a la organización en la toma de decisionessobre su lógica de negocio.

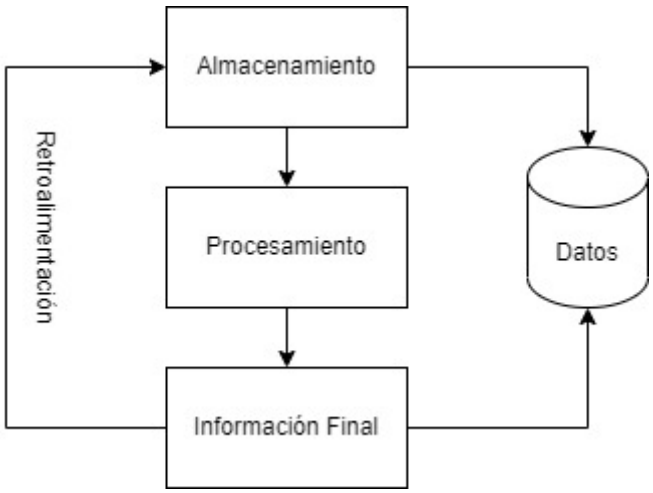


Figura 1.2: Sistema de Información
Elaborado por: Investigador

1.2.2.2. Aplicación Web

Una aplicación web es un programa de software que se almacena en un servidor remoto y utiliza tecnologías web (JavaScript, HTML, CSS) y navegadores web para ofrecer una o más funciones para el usuario final. Los diferentes niveles de utilidad proporcionados por los sitios web han provocado cierto debate sobre qué califica como una aplicación web. El factor determinante es si esta proporciona un servicio o una función, si realiza una tarea, por insignificante que sea, es una aplicación web y debe gestionarse como tal [10].

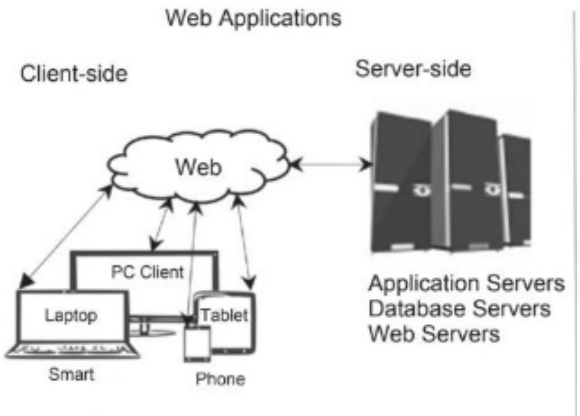


Figura 1.3: Funcionamiento de Aplicaciones Web
Elaborado por: [10]

1.2.2.3. Backend

Un desarrollador Backend es quien trabaja en el lado del servidor, utilizando lenguajes como Hypertext Pre-Processor (PHP), Python, .NET, Java, etc. Interactuando con bases de datos, verificando sesiones de usuario y montando una página en el servidor [11].

1.2.2.4. Frontend

Desarrollo de sitios web del lado del cliente. En este tipo de desarrollo se maqueta la estructura semántica del contenido, se codifica el diseño de las hojas de estilo y se agrega interacción con el usuario [11].

El desarrollo de aplicaciones web se conforma del Back y Frontend, el primero se compone por el servidor y su comunicación directa o capa de acceso a los datos mientras el front se dedica a la interfaz de usuario o capa de presentación mostrada desde un navegador.

1.2.2.5. Servicios Web

La Word Wide Web Consortium (W3C) lo define como un sistema de software diseñado para soportar interacción interoperable máquina a máquina sobre una red. Este tiene una interface descrita en un formato procesable por una máquina (específicamente WSDL). Otros sistemas interactúan con el servicios web en una manera prescrita por su descripción usando mensajes SOAP, típicamente enviados usando HTTP con una serialización XML en relación con otros estándares relacionados con la web [12].

Pueden ser desarrollados en una gran variedad de lenguajes para ser implementados sobre muchos tipos de redes de computadores. El éxito de la interoperabilidad se consigue gracias a la adopción de protocolos y estándares abiertos que básicamente son utilizados para definir, localizar, implementar y hacer que un servicio interactúe con otro. Este conjunto de tecnologías esta conformado por subconjuntos de servicio de transporte, mensajería XML, descripción del servicio y descubrimiento de servicios [12].

Servicio de Transporte

Encargado del transporte de mensajes entre aplicaciones sobre la red. Incluyendo protocolos de nivel de aplicación como HyperText Transfer Protocol (HTTP)

define la sintaxis y semántica utilizada para la arquitectura web, por otro lado esta, File Transfer Protocol (FTP) para transmisión de archivos a través de redes TCP y Simple Mail Transfer Protocol (SMTP) utilizado para envió de mensajes de correo electrónico [12].

Mensajería XML

Encargado de la codificación de los mensajes en XML estándar y pueda así ser interpretado en cualquiera de los nodos de la red. Los componentes más utilizados en este conjunto son: Representational State Transfer (REST), favorito por su escalabilidad y compatibilidad con otros componentes, Remote Procedure Calls (RPC), eXtended Markup Language (XML) ocupado generalmente para intercambio de datos sobre la red, Simple Object Access Protocol (SOAP) para intercambio de mensajes basados en XML [12].

Descripción del Servicio

Cuenta con una interfaz pública la cual es descrita por un formato llamado Web Services Descripción Languages (WSDL) describiendo lo que hace el servicio web, donde se encuentra y la forma de ser invocado, sirve de gran ayuda a los desarrolladores por la información que se puede obtener [12].

1.2.2.6. API

Lo primero que se debe saber es que una API (siglas en inglés de Interfaz de Programación de Aplicaciones) es una interfaz o zona de contacto de un conjunto de bibliotecas o paquetes de software capacitados para que otro software o programa pueda "verlos" y ejecutarlos. Es decir, una API es la herramienta que permite que un software se comunique o interactúe con otro. La importancia de las API es que permiten a diferentes programas, dispositivos y aplicaciones trabajar en conjunto, compartir información y así crear la verdadera conectividad que significa Internet [13].

1.2.2.7. Patrón Modelo-Vista-Controlador

MVC es un patrón de diseño que considera dividir una aplicación en tres módulos identificados y cada parte con una funcionalidad definida. Según Cecilia Ávila es un esquema de arquitectura por capas muy utilizado en el desarrollo de software basado en aplicaciones web. El modelo (M) controla todo lo relacionado con los

datos, la vista (V) lo relacionado con las interfaces de usuario y el controlador (C) se encarga de la manipulación del M para mostrar información en la V [14].

Modelo

Es el objeto que representa los datos del programa. Maneja los datos y controla todas sus transformaciones. El Modelo no tiene conocimiento específico de los Controladores o de las Vistas, ni siquiera contiene referencias a ellos. Es el propio sistema el que tiene encomendada la responsabilidad de mantener enlaces entre el Modelo y sus Vistas, y notificar a las Vistas cuando cambia el Modelo [15].

Vista

Es el objeto que maneja la presentación visual de los datos representados por el Modelo. Genera una representación visual del Modelo y muestra los datos al usuario. Interactúa preferentemente con el Controlador, pero es posible que trate directamente con el Modelo a través de una referencia al propio Modelo [15].

Controlador

Es el objeto que proporciona significado a las órdenes del usuario, actuando sobre los datos representados por el Modelo, centra toda la interacción entre la Vista y el Modelo. Cuando se realiza algún cambio, entra en acción, bien sea por cambios en la información del Modelo o por alteraciones de la Vista. Interactúa con el Modelo a través de una referencia al propio Modelo [15].

Tabla 1.1: Ventajas y Desventajas de patrón MVC

Elaborado por: Investigador

Ventajas	**Desventajas**
▪ Aplicación modular ▪ Modificable y mantenible ▪ Orientado a objetos ▪ Ideal para proyectos grandes	▪ Tiempo de desarrollo mayor en el modelo ▪ Requiere de una arquitectura de clases e interfaces

1.2.2.8. Single Page Aplication

Single Page Aplication (SPA) es una aplicación web que carga completamente todos los recursos en la solicitud inicial y luego los componentes de la página son

reemplazados por otro componente dependiendo de la interacción con el usuario. La SPA se compone de componentes individuales que se pueden reemplazar / actualizar de forma independiente, sin actualizar / recargar toda la página, por lo que no es necesario volver a cargar la página completa [16].

Una SPA al visitar la página por primera vez, cargará completamente la estructura y cuando el usuario navegue, solo realizará la petición al servidor para procesar los datos, puesto que el contenido html y css están ya cargados mejorando la experiencia de usuario, mientras que un sitio web tradicional consiste en que cada vez que el usario visite una página específica se cargará desde cero un gran volumen de contenido.

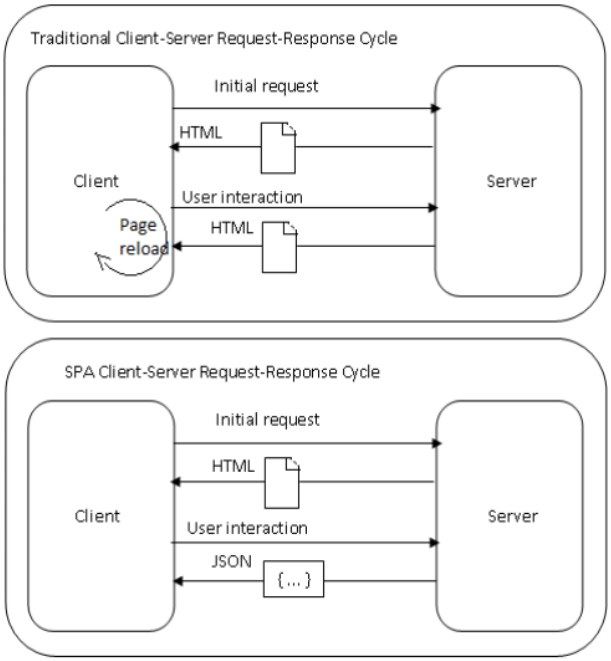


Figura 1.4: Funcionamiento entre un sitio web tradicional y SPA
Elaborado por: [16]

1.2.2.9. Framework

En el desarrollo de software, un framework es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. En programación, es un set de funciones o código genérico que realiza tareas comunes y frecuentes en todo tipo de aplicaciones (creación de objetos, conexión a base de datos, etc.). Esto brinda una base sólida sobre la cual desarrollar aplicaciones concretas y permite obviar los componentes más triviales y genéricos del desarrollo y abre camino a que diseñadores y programadores puedan pasar más tiempo

identificando requerimientos de software [17].

Un framework agrega funcionalidad extendida a un lenguaje de programación, automatiza muchos de los patrones de programación para orientarlos a un determinado propósito, proporcionando una estructura al código, mejorándolo y haciéndolo más entendible y sostenible, y permite separar en capas la aplicación. En general, divide la aplicación en tres capas [17]:

- La lógica de presentación que administra las interacciones entre el usuario y el software.
- La Lógica de datos que permite el acceso a un agente de almacenamiento persistente u otros.
- La lógica de dominio o de negocio, que manipula los modelos de datos de acuerdo a los comandos recibidos desde la presentación.

1.2.2.10. Entity Framework

ADO.NET entity framework es un marco de trabajo para la plataforma .NET que permite superponer varias capas de abstracción sobre un almacén relacional con el n de hacer posible una programación más conceptual (basada en los conceptos del dominio con el que se trabaja) y de reducir a una mínima expresión el desajuste de impedancias causado por las diferencias entre los modelos de programación relacional y orientado a objetos [18].

La gran ventaja de Entity Framework (EF) es que hace transparente la base de datos con la que se trabaja, pues el modelo de EF genera las sentencias Structured Query Language (SQL) nativas para cada Sistema Gestor de Base de Datos (SGBD), compatible contra SQL Server, Oracle, DB2, MySQL [19].

Características

- Accesibilidad a la comunidad de desarrolladores.
- Multiplaforma para Windows, Linux y MAC.
- Permite trabajar con diferentes bases de datos (SQL, MySQL, Oracle).
- Seguimineto a cambios mediante las migraciones.
- Ejecuta métodos Create Read Update Delete (CRUD)

- Gestión automática de Transacciones.
- Crea un Entity Data Model (EDM) basado en entidades.

Base de Datos Primero

Trabaja sobre una base de datos existente con la que se desea ocupar, EF se encargará de generar las clases necesarias para mapear los datos además las actualizará en caso de que haya cambios en las entidades.

Modelo Primero

Se refiere a crear un modelo de datos de manera visual, por medio del Diseñador de Modelos de Visual Studio, EF se encarga de generar la base de datos o modificarla.

Código Primero

Empieza desde cero, es decir que define las clases mediante código y EF genera la base de datos, este enfoque esta orientado a tener el control del comportamiento de las clases, permitiendo implentar métodos de la clase DbContext como operaciones CRUD, además permite dar seguimiento a los cambios mediante migraciones.

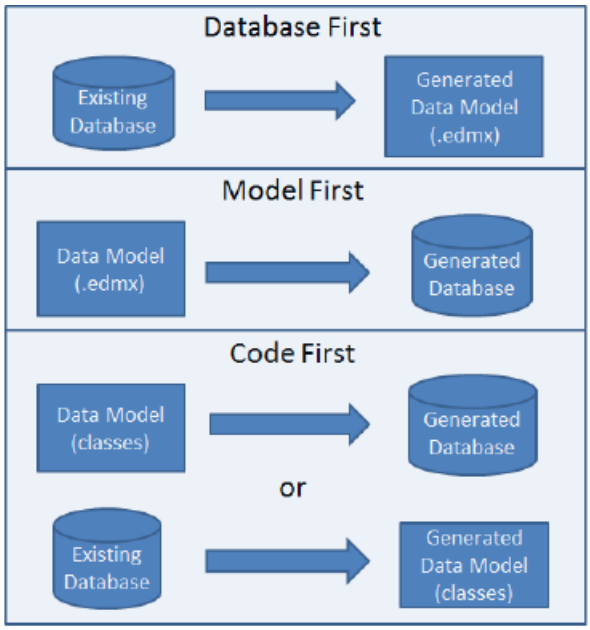


Figura 1.5: Tipos de Trabajo EF
Elaborado por: [18]

1.2.2.11. Framework Web Api

Con la llegada de dispositivos se necesita un punto final web para dar servicio a cualquier tipo de contenidos (JSON, XML, Imágenes, PDF) a cualquier tipo de cliente. Cualquier código que solicite una petición HTTP es un cliente potencial para un punto final web [20].

El framework Web Api se ofrece para dar solución a la demanda de capas de los servidores capaces de presentar la interfaz RESTful e interactuar con cualquier cliente HTTP sin suposiciones y restricciones. El marco Web Api es un conjunto alternativo de clases para crear puntos finales HTTP diseñados para ser conscientes de la sintaxis y semántica HTTP completas. El framework web api ofrece una interfaz de programación casi idéntica a ASP.NET MVC; incorpora controladores, rutas y modelos, pero los ejecuta dentro de un entorno de ejecución completamente nuevo. Con la API web asp.net, el punto de crear un marco web desacoplado del servidor web comenzó a echar raíces, y esto llevó a la definición de la Interfaz Web Abierta para el estándar .Net (OWIN), donde especifica las reglas para que un servidor web y una aplicación web interoperen. [20].

1.2.2.12. ASP .NET Core

ASP.NET Core, es un escenario multiplataforma de gran utilidad de código abierto, con alto beneficio en la elaboración de aplicaciones en relación a la nube, servicios web, asimismo aplicaciones IoT (internet de las cosas) y backends móviles, también el manejo al desarrollar con las herramientas preferidas de Windows, MacOS y Linux. ASP.NET Core se puede ejecutar en la nube de Microsoft y servicios de las varias plataformas que se muestran en su página [21].

Tabla 1.2: Ventajas y Desventajas de ASP .NET Core

Elaborado por: Investigador

Ventajas	Desventajas
▪ Multiplataforma ▪ Soporte continuo ▪ Combinación con otros lenguajes ▪ Emplea MVC ▪ Comunicación nativa con SQL	▪ Menor rendimiento sobre Linux ▪ Ocupa IDE Visual Studio ▪ Mayor tiempo de desarrollo

1.2.2.13. React

Es una biblioteca de interfaz de usuario desarrollada en Facebook para facilitar la creación de aplicaciones interactivas, con estado y componentes de interfaz de usuario reutilizables. Esto es utilizado por Facebook en producción. ReactJS es mejor para renderizar interfaces de usuario complejas con alto rendimiento. El fundamento básico detrás de React es el concepto de DOM virtual. ReactJS usa DOM virtual de manera efectiva, que se puede representar en el lado del cliente o en el lado del servidor y comunicarse de un lado a otro. El Virtual DOM renderiza subárboles de nodos basados en cambios de estado. Hace la menor cantidad de manipulación DOM posible para para mantener sus componentes actualizados. React es más ligero que Angular, se llena con el mínimo condiciones y elimina la necesidad de utilizar elementos adicionales como complementos. React está en contra de dos vías vinculante, intencionalmente se mantiene alejado de él y utiliza actualizaciones explícitas en su lugar [22].

Virtual DOM

Al igual que el DOM real, el DOM virtual es un árbol de nodos que enumera elementos y sus atributos y contenido como objetos y propiedades. El método render() de React crea un árbol de nodos a partir de React componentes y actualiza este árbol en respuesta a mutaciones en el modelo de datos, causadas por acciones. Cada vez que los datos subyacentes cambian en una aplicación React, una nueva representación DOM virtual de la se crea la interfaz de usuario [22].

Características

Actualmente es uno de los frameworks que se encuentra en auge entre sus principales características estan:

- Permite crear aplicaciones nativas para Android e iOS.
- Óptima experiencia de usuario.
- Componentes Multiplaforma.

1.2.2.14. Angular

Es un marco JavaScript de código abierto mantenido por Google y la comunidad que puede ayudar a los desarrolladores a crear aplicaciones de una sola página que se construyen sobre el JavaScript facilitando mucho la vida de los desarrolladores. La idea detrás del uso de Angular en una aplicación web es hacer que su aplicación web sea modular y fácil de mantener. Su propósito es ayudar desarrollo de aplicaciones web con capacidad de controlador de vista de modelo (MVC) en un esfuerzo por facilitar el desarrollo, el mantenimiento y las pruebas. Después de usar archivos minificados en su aplicación, el tamaño se reduce a algunos KB y las páginas se cargan mucho más rápido [22].

Two-way Binding

El enlace de datos en AngularJS es la sincronización entre el modelo y la vista. Cuando los datos en el modelo cambia, la vista refleja el cambio, y cuando los datos en la vista cambian, el modelo es actualizado también. Esto sucede de forma inmediata y automática [22].

Vistas y Rutas

Cuando una aplicación web depende de muchas páginas HTML (vistas), se trabaja en una SPA (single page application), lo que significa que se puede agregar un solo elemento HTML (contenedor) dentro de un solo archivo HTML, y dentro de éste poder renderizar cada página que tenga que mostrarse, sin tener que recargar la URL. El render entre páginas dentro del contenedor es inmediato. Esto se logra con otra directiva llamada ng-view o ui-view, y una librería llamada angular-route o angular-ui-router [23].

1.2.2.15. Bases de Datos

Son conjuntos de datos almacenados sin redundancia que son innecesarias en un soporte informático y accesible simultáneamente por múltiples usuarios y aplicaciones. Los datos deben estar de forma estructurada y almacenada de forma totalmente independiente de las aplicaciones que la utilizan [24].

Las bases de datos son contenedores de información. Existen distintas formas de ordenar esa información; uno de los lenguajes que se usa para organizar esa información es SQL, modelo relacional y la no relacional (NoSQL) [25].

SQL

SQL es considerado un estándar en la manera en que se deben tratar y administrar todos los datos de una aplicación. La aparición de este modelo relacional llevó consigo la aparición de conceptos nuevos como tablas, relaciones, claves, filas, etc; lo cual facilitó la tarea del programado desde 1970 hasta la actualidad [25].

NOSQL

Las bases de datos NOSQL son sistemas de almacenamientos de información que no cumplen con los esquemas entidad/relación es decir que no se imponen una estructura de datos en forma de tablas y relaciones entre ellas, por lo tanto estos sistemas son más flexibles, ya que permiten almacenar información [25].

1.2.2.16. Sistema de Gestión de Base de Datos

El sistema de gestión de la base de datos (SGBD) es una aplicación que permite a los usuarios definir, crear y mantener bases de datos, proporcionando acceso controlado a las mismas. Es una herramienta que sirve de interfaz entre el usuario y las bases de datos. Es decir que por un lado están los datos organizados según ciertos criterios y por otro, un software que permite o facilita su gestión con distintas herramientas y funcionalidades [26].

SQL Server

El motor de bases de datos de Microsoft, inicialmente fue adquirido de Sybase por 1989. Con el paso de los años SQL Server ha evolucionado actualmente posicionarse entre las bases de datos más populares [25].

Oracle

Tuvo su origen en 1979 en la empresa SDL, para con el tiempo convertirse en la base de datos más usada a nivel empresarial. Oracle ofrece el conjunto de herramientas más completo que se va desde la base de datos, aplicaciones comerciales, herramientas de desarrollo, herramientas de soporte de decisiones o business inteligentes [25].

MongoDB

Una base de datos documental, de alto desempeño, no utiliza esquema de bases de datos. Permite almacenar la información de forma mas natural mediante documentos auto contenidos, es decir al no usar tablas con relaciones cada unidad de datos contiene en si mismo las dependencias necesarias [25].

1.3. Objetivos

1.3.1. Objetivo General

Desarrollar una aplicación web que permita optimizar el proceso de recaudación de valores y consulta de planillas mensuales en la Cooperativa de Ahorro y Crédito San Martín de Tisaleo Ltda.

1.3.2. Objetivos Específicos

- Describir los procesos que intervienen en el procedimiento de emisión de planillas mensuales por concepto de consumo de agua potable.
- Analizar las funcionalidades del framework Angular para el desarrollo de la aplicación.
- Definir la metodología de desarrollo ágil que se ajuste a las necesidades de la organización.
- Implantar la aplicación web en la institución financiera.

CAPÍTULO II

METODOLOGÍA

2.1. Materiales

2.1.1. Institucionales

- Repositorios Digitales de la Universidad Técnica de Ambato.
- Repositorios Digitales de las Universidades Ecuatorianas y Extranjeras.
- Bibliotecas Virtuales.
- Infraestructura Tecnológica de la COAC San Martín de Tisaleo.

2.1.2. Humano

- Docente Tutor.
- Investigador.
- Involucrados en el proyecto (Gerente General, Personal Administrativo y Operativo).

2.1.3. Otros

- Impresora.
- Servidor de Archivos.
- Servidor de Aplicaciones.
- Libros y documentos relacionados al tema de investigación.
- Suministros de Oficina.

2.1.4. Económicos

Tabla 2.1: Presupuesto Económico

Elaborado por: Investigador

N°	Detalle	Unidad	Cantidad	Valor Unitario	Valor Total
1	Resma de papel	c/u	2	5.00	10.00
2	Portaminas	c/u	3	1.25	3.75
3	Impresiones	c/u	500	0.15	75.00
4	Esferográficos	c/u	4	0.60	2.40
5	Internet	mes	12	28.50	342.00
6	Copias	c/u	250	0.02	5.00
7	Anillados	c/u	2	4.00	8.00
8	Cuaderno	c/u	1	1.50	1.50
9	Borrador	c/u	2	0.20	0.40
10	Computador	c/u	1	1, 000.00	1, 000.00
11	Disco Duro	c/u	1	80.00	80.00
12	Transporte	mes	6	60.00	360.00
13	Capacitaciones	mes	2	75.00	150.00
				Subtotal	2, 038.05
				Imprevistos (10 %)	203.81
				Total	2, 241.86

2.2. Métodos

2.2.1. Modalidad de la Investigación

Las modalidades de investigación a emplearse en el presente proyecto serán, bibliográfica, de campo y aplicada.

Investigación Bibliográfica

Porque se realizará búsquedas bibliográficas en fuentes como trabajoos de pregrado nacionales, documentos y libros en el área informática para detallar de la contextualización, marco teórico y análisis de frameworks que contribuyan al mantenimiento y escalabilidad de la aplicación en caso de ser necesarios.

Investigación de Campo

Se realizará una investigación de campo debido a que se empleará un estudio en el lugar principal, es decir la Cooperativa de Ahorro y Crédito San Martín de Tisaleo, con el apoyo de gerencia general.

Investigación Aplicada

La investigación será aplicada, ya que los conocimientos adquiridos durante la carrera universitaria se utilizarán para desarrollar una solución tecnológica como lo es el aplicativo web.

2.2.2. Población y Muestra

De acuerdo con el tema de investigación no se requiere muestra, ya que es menor a 100, por tal motivo se trabajará con el gerente, personal administrativo y operativo de la cooperativa. En la siguiente tabla se muestra el número de personas con las cuales se trabajará. No se tomará en cuenta a los usuarios finales ya que son parte del proceso de recaudación de valores, pero no entran a manipular el sistema para el manejo de datos.

Tabla 2.2: Población

Elaborado por: Investigador

Población	**Cantidad**	**Porcentaje**
Gerente	1	20 %
Administrativo	2	40 %
Operativo	2	40 %
Total	5	100 %

2.2.3. Recolección de Información

La técnica a emplearse será la entrevista dirigida al personal de la cooperativa.

Tabla 2.3: Recolección de Información

Elaborado por: Investigador

Preguntas Básicas	**Explicación**
¿Para qué?	Para alcanzar los objetivos de la investigación
¿De qué personas u objetos?	Personal Administrativo y Operativo
¿Sobre qué aspectos?	Proceso de recaudación y consulta de valores
¿Quién, Quiénes?	Investigador: Christian Gustavo Verdesoto Guaman
¿Cuándo?	Período Académico abril - septiembre 2022
¿Dónde?	Cooperativa de Ahorro y Crédito “San Martín”
¿Cuántas veces?	Una
¿Qué técnicas de recolección?	Entrevista
¿Con qué?	Cuestionario
¿En qué situación?	En condiciones normales de trabajo

2.2.4. Procesamiento y análisis de datos

Para la recolección y análisis de datos se realizará una reunión y posteriormente se procederá con una entrevista dirigida al Gerente General porque es la persona responsable de gestionar la lógica de negocio de la cooperativa, asi como, fomentar el desarrollo de nuevos productos y servicios que permitan ampliar la cartera de ingresos, optimizando los recursos que posee su infraestructura.

2.2.4.1. Resultados de la Entrevista

Una vez realizada la entrevista se obtuvo los siguientes resultados:

Pregunta 1

¿Cuáles son los servicios que ofrece la cooperativa?

Respuesta

"La cooperativa San Martin es una institución de intermediación financiera enfocada administrar de una manera responsable los fondos de nuestros socios, mediante productos como Ahorros, inversiones, créditos, pagos y servicios."

Pregunta 2

¿Cuál es la disponibilidad del servicio de internet en la cooperativa?

Respuesta

"El servicio de internet en la institución es primordial para el desarrollo operativo de la misma, razón por la cual, el departamento de tecnologías de la información mediante proveedores de internet garantiza la conectividad constante del servicio de internet y red interna de comunicación en los diversos puestos de trabajo de la institución."

Pregunta 3

¿La cooperativa cuenta con un sistema para cobro de servicios básicos?

Respuesta

"Gracias al servicio de internet constante se ha contratado dos sistemas web externos para prestación de servicios como cobros CNT, servicio de luz eléctrica,

recargas, impuestos, matriculación, pensiones alimenticias, tiendas de ropa, cosméticos, pago de IESS entre otros. permitiendo a los socios y clientes optimizar su tiempo realizando sus pagos en la cooperativa, situación que antes lo realizaban en la ciudad de Ambato"

Pregunta 4

¿Dónde se puede cancelar el valor de una planilla mensual de agua potable del sector?

Respuesta

"El servicio de agua potable del sector exclusivamente se lo puede realizar en la oficina única de recaudación del municipio de Tisaleo, mientras que para los sectores rurales o comunidades se lo realiza en una oficina determinada. Sin embargo, los sistemas externos mencionados anteriormente ofrecen la posibilidad de cobro de agua potable de diferentes cantones como Ambato, Pelileo, Riobamba, Quito entre otros."

Pregunta 5

¿Qué características se busca en un sistema web para la recaudación de valores?

Respuesta

"Se busca ofrecer a los clientes y socios la posibilidad de cumplir con sus obligaciones mediante el cobro en línea de servicios varios, optimizando tiempo y recursos."

Pregunta 6

¿La cooperativa cuenta con la infraestructura tecnológica para el desarrollo de aplicaciones web?

Respuesta

"Cumpliendo con nuestra misión de administrar responsablemente los fondos de nuestros socios la cooperativa San Martín ha invertido en seguridad física y electrónica, dotando de equipos tecnológicos que permitan el desarrollo de productos como la aplicación móvil denominada San Martín Móvil y web transaccional SM en Línea."

Pregunta 7

¿Existe disponibilidad de servidores para la publicación y consumo de servicios web?

Respuesta

"Como se mencionó en la pregunta anterior, la cooperativa cuenta con servidores donde se alojan los productos desarrollados, cabe mencionar que estos equipos se encuentran en óptimas condiciones para agregar la publicación de nuevos productos con tecnologías web que aporten a la expansión de la misma."

Pregunta 8

¿Qué módulos considera necesarios para la aplicación web?

Respuesta

"Se necesitan la administración de contribuyentes, cuentas, lecturas, recaudaciones o cobros y finalmente reportes que reflejen los datos registrados en la base de datos."

Pregunta 9

¿Por qué considera que es importante desarrollar una aplicación web para el cobro de planillas mensuales por consumo de agua potable en la cooperativa?

Respuesta

"El desarrollo de un aplicativo web aporta significativamente a la cooperativa y a los contribuyentes que no cuentan con un sistema de recaudación. porque nos ayuda a incrementar la afluencia de clientes, captando recaudaciones del servicio automatizando y organizando la información de la entidad recaudadora."

Interpretación

Con la entrevista realizada se determinan los requerimientos funcionales para el desarrollo de la aplicación web. La cooperativa "San Martín" busca aportar con herramientas tecnológicas en el campo de la programación. Por consiguiente, este deberá solventar todos los procesos que se requiere para la emisión de planillas mensuales y luego se pueda proceder con la recaudación de valores.

Dado que la cooperativa cuenta la infraestructura tecnológica necesaria, se preocederá con el desarrollo de la aplicación web, esta tendrá un gran impacto para la institución y los contribuyentes de los sectores rurales que no cuentan con una recaudación automatizada por el consumo del recurso hídrico. Con la implementación se podrá replicar el modelo de gestión en escenarios similares.

2.2.5. Desarrollo del proyecto

Las actividades que se llevarán a cabo en el desarrollo del proyecto se detallan a continuación:

1. Descripción de los procesos que intervienen el la emisión de planillas mensuales.
2. Descripción de funcionalidades y beneficios de framework Angular para el desarrollo de aplicaciones web.
3. Estudio comparativo de metodologías ágiles de desarrollo y elección de la que mejor se ajuste a los requerimientos de la cooperativa.
4. Desarrollo de la aplicación web.
5. Pruebas de Rendimiento.

CAPÍTULO III

RESULTADOS Y DISCUSIÓN

3.1. Análisis y discusión de resultados

3.1.1. Descripción de Procesos

Para el presente proyecto, se utilizó diagramas de flujo de proceso definido por la Cooperativa de Ahorro y Crédito San Martín de Tisaleo Ltda., lo que permitió determinar el procedimiento a seguir para la emisión de las planillas por el servicio de agua potable previo a la recaudación de valores.

De esta manera se podrá establecer los requerimientos de procesos mediante módulos necesarios, con el objetivo de mantener un orden secuencial en el desarrollo de la aplicación web.

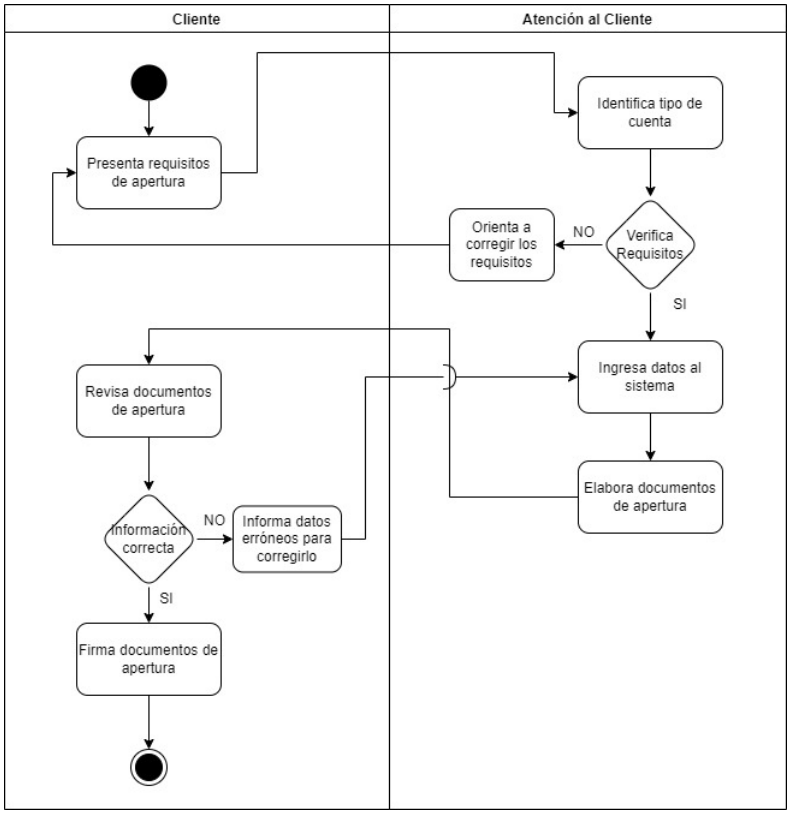


Figura 3.1: Proceso de apertura de cuenta
Elaborado por: Investigador

Apertura de Cuenta

El proceso como se muestra en la figura 3.1, realiza la apertura de cuenta cuando el cliente se acerca con los requisitos y documentos personales, luego el asesor de atención al cliente verifica la documentación e identifica el tipo de cuenta para proceder con el ingreso de datos al sistema e imprime el documento de apertura, para que el cliente revise que la información reflejada en el informe sean correctos.

Posteriormente el asesor procede al ingreso de datos como nombres, identificación, fecha de nacimiento, ubicación del domicilio y datos de conyugue en caso de que existiera, el sistema le asigna un código único denominado CUC, este será el identificador para asignarle una cuenta con su tarifa correspondiente.

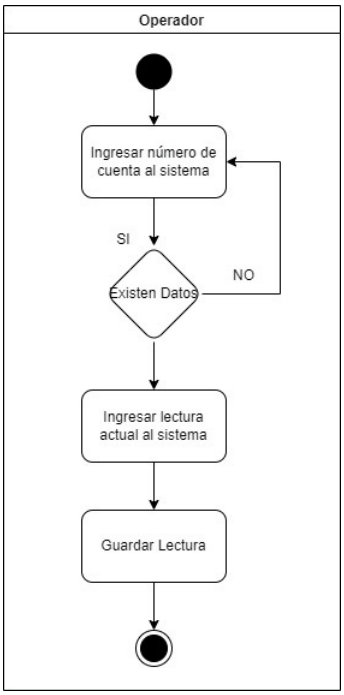


Figura 3.2: Proceso de registro de lectura
Elaborado por: Investigador

Registro de Lectura

El proceso como se muestra en la figura 3.2, registra la lectura de un medidor a la cuenta correspondiente, cuando el operador ingresa el número de cuenta en el sistema, el mismo que arrojará los datos del contribuyente y del medidor, esto le permitirá al operador corroborar la información para proceder con el registro de la lectura.

Posterior al registro de lectura, la aplicación se encarga de generar todas las planillas de las cuentas existentes en función del producto de la diferencia entre la lectura anterior y actual, con el valor de la tarifa establecida en la apertura de la cuenta.

Una vez generado las planillas, la aplicación reflejará el valor de cada una. Cabe mencionar que se podrá consultar por número de cuenta o nombres del contribuyente, la información que se reflejará serán fechas de lectura, emisión, lectura anterior, actual, cantidad consumida entre otros. El estado de las planillas se manejará como cancelada o pendiente.

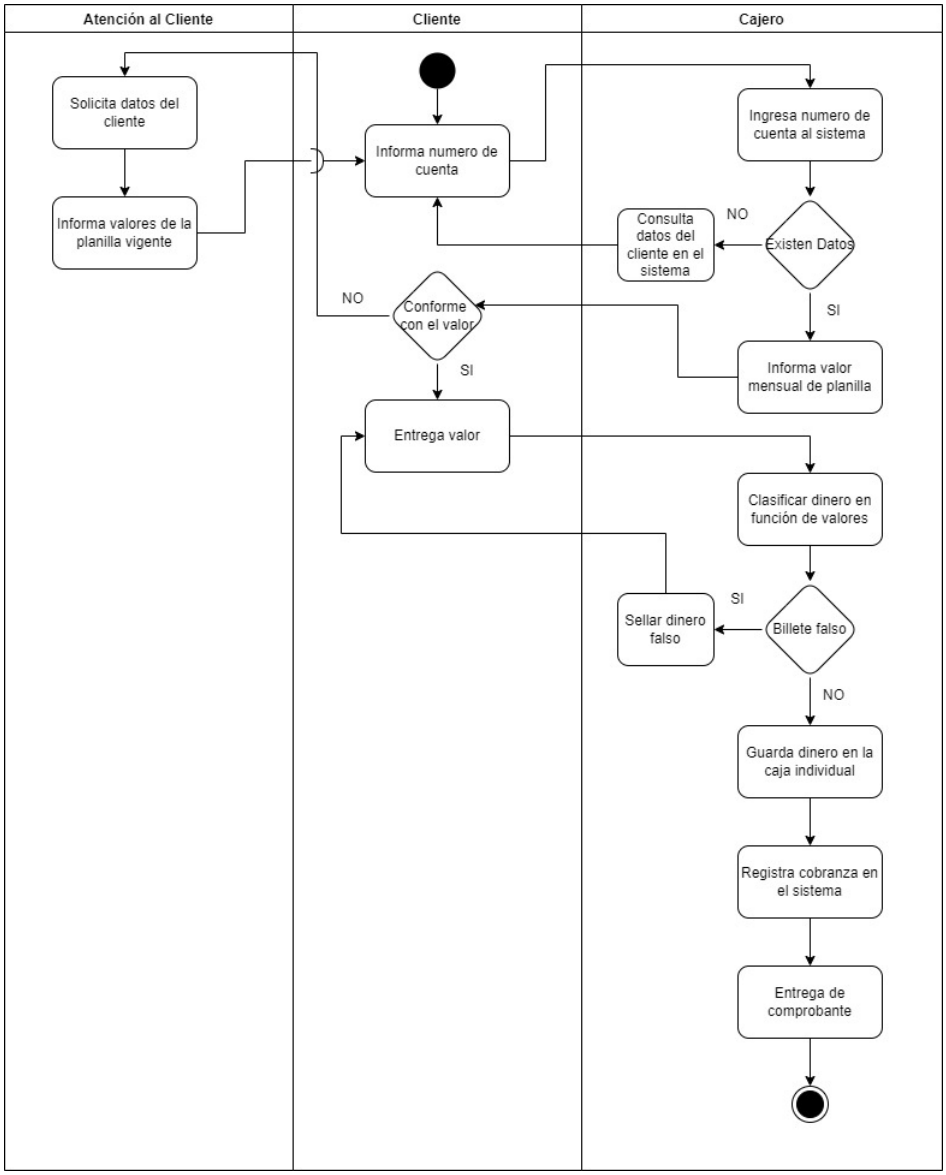


Figura 3.3: Proceso de recaudación de valores
Elaborado por: Investigador

Recaudación de Valores

El proceso como se muestra en la figura 3.3, describe la recaudación de valores cuando el cliente se acerca a pagar el valor por la cantidad de metros cúbicos consumida durante el periodo, informa el número de cuenta al cajero, para verificar que corresponde al contribuyente e informar los valores adeudados, una vez que el cliente entregue el dinero en ventanilla se procede a registrar la cobranza en la aplicación, para generar el comprobante de pago el mismo se le entregará para su respaldo.

Una vez que las planillas se hayan generado correctamente, se actualizará el saldo pendiente en la cuenta, en caso de que no se cancele y se genere una nueva planilla este saldo seguirá aumentando hasta cancelar los valores pendientes. Los comprobantes de pago reflejarán datos de la cuenta, contribuyente y valores monetarios por los que pagó.

Por cada recaudación se añade un registro, con el fin de saber el valor recaudado durante el día por cada usuario registrado bajo el perfil de cajero en la aplicación web.

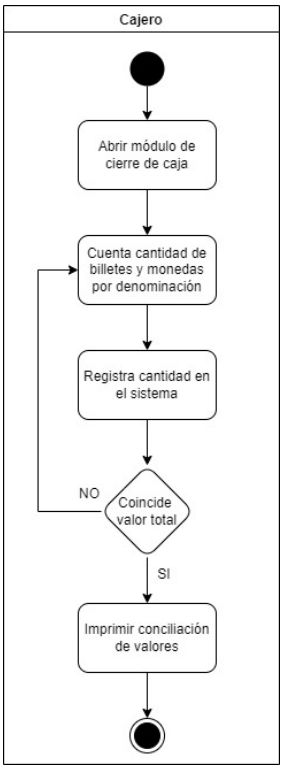


Figura 3.4: Proceso de cierre de caja
Elaborado por: Investigador

Cierre de Caja

El proceso como se muestra en la figura 3.4, describe el cierre de caja que se realizará al final del día, el dinero físico deberá ser igual al valor que refleja el módulo cierre de caja. Luego de ello deberá imprimir la conciliación de valores para entregarlo al supervisor o jefe inmediato.

Luego que se haya realizado el cierre de caja de manera correcta, se añadirá un registro por cada cierre con el valor inicial, valor final, la fecha en la que se realizó la conciliación de valores, esto aplica para los cajeros.

El objetivo de realizar un cierre de caja, se lo realiza con el fin de llevar un control del flujo de valores recaudados durante una fecha determinada. La aplicación sumará los valores de cada planilla recaudada por el cajero y lo mostrará en el módulo cierre de caja. Esto aplica para todos los usuarios que manejen el perfil de cajero.

3.1.2. Metodologías de desarrollo ágil

Los procesos de desarrollo del software rápido se diseñan para producir rápidamente un software útil. El software no se desarrolla como una serie de incrementos, y cada uno de ellos incluye una nueva funcionalidad del sistema. comparten algunas características fundamentales [27].

- Los procesos de especificación, diseño e implementación están entrelazados.
- El sistema se desarrolla en diferentes versiones, con la colaboración de usuarios finales y colaboradores que intervienen en la especificación y evaluación del sistema.
- Las interfaces de usuario del sistema se desarrollan usando con frecuencia un sistema de elaboración interactivo, que permita que el diseño de la interfaz se cree rápidamente en cuanto se dibujan y colocan iconos en la interfaz.

Manifiesto Ágil

La filosofía detrás de los métodos ágiles se refleja en el manifiesto ágil que se define como la colaboración con el cliente sobre la negociación del contrato, respuesta al cambio sobre el seguimiento de un plan [27].

Tabla 3.1: Principios de desarrollo ágil

Elaborado por: [27]

Principio	**Descripción**
Participación del cliente	Los clientes deben intervenir estrechamente durante el proceso de desarrollo
Entrega Incremental	El software se desarrolla en incrementos y el cliente especifica los requerimientos que se van a incluir en cada incrementos
Personas, No Procesos	Tienen que reconocerse y aprovecharse las habilidades del equipo de desarrollo
Adoptar el cambio	Esperar a que cambien los requerimientos del sistema y, de este modo, diseñar el sistema para adaptar dichos cambios
Mantener Simplicidad	Enfocarse en la simplicidad tanto en el software a desarrollar como en el proceso de desarrollo

3.1.3. Programación Extrema XP

eXtreme Programing nace como nueva disciplina de desarrollo de software hace aproximadamente seis años, y ha causado un gran revuelo entre el colectivo de programadores del mundo. Fue propuesto por Kent Beck un programador que ha trabajado en múltiples empresas y que actualmente lo hace como programador en la conocida empresa automovilística DaimlerChrysler. La programación extrema se basa en la simplicidad, la comunicación y el reciclado continuo de código, para algunos no es más que aplicar una pura lógica [28].

Tabla 3.2: Valores Fundamentales de XP

Elaborado por [29]

Valor	Descripción
Comunicación Efectiva	▪ Colaboración cercana e informal ▪ Uso de metáforas para comunicar conceptos importantes ▪ Evitar documentación voluminosa
Simplicidad	▪ Diseñar solo para las necesidades inmediatas
Retroalimentación	▪ Desde la validación del software implementado ▪ Desde el cliente ▪ Desde los miembors del equipo
Valentía	▪ Tener la disciplina de diseñar para el hoy ▪ Los requerimientos futuros pueden cambiar dramáticamente

3.1.3.1. Roles XP

Programador

Escribe las pruebas unitarias y produce el código del sistema. Debe existir una comunicación y coordinación adecuada entre los programadores y otros miembros del equipo [30].

Cliente

El cliente escribe las historias de usuario y las pruebas funcionales para validar su implementación. Además, asigna la prioridad a las historias de usuario y decide cuáles se implementan en cada iteración centrándose en aportar mayor valor al negocio. El cliente es sólo uno dentro del proyecto pero puede corresponder a un interlocutor que está representando a varias personas que se verán afectadas por el sistema [30].

Encargado de pruebas (Tester)

El encargado de pruebas ayuda al cliente a escribir las pruebas funcionales. Ejecuta las pruebas regularmente, difunde los resultados en el equipo y es responsable de las herramientas de soporte para pruebas [30].

Encargado de seguimiento (Tracker)

El encargado de seguimiento proporciona realimentación al equipo en el proceso XP. Su responsabilidad es verificar el grado de acierto entre las estimaciones realizadas y el tiempo real dedicado, comunicando los resultados para mejorar futuras estimaciones. También realiza el seguimiento del progreso de cada iteración y evalúa si los objetivos son alcanzables con las restricciones de tiempo y recursos presentes. Determina cuándo es necesario realizar algún cambio para lograr los objetivos de cada iteración [30].

Entrenador (Coach)

Es responsable del proceso global. Es necesario que conozca a fondo el proceso XP para proveer guías a los miembros del equipo de forma que se apliquen las prácticas XP y se siga el proceso correctamente [30].

Consultor

Es responsable del proceso global. Es necesario que conozca a fondo el proceso XP para proveer guías a los miembros del equipo de forma que se apliquen las prácticas XP y se siga el proceso correctamente [30].

Gestor (Big boss)

Es el vínculo entre clientes y programadores, ayuda a que el equipo trabaje efectivamente creando las condiciones adecuadas. Su labor esencial es de coordinación [30].
El ciclo de desarrollo consiste en los siguientes pasos:

1. El cliente define el valor de negocio.
2. El programador estima el esfuerzo necesario.
3. El cliente selecciona qué construir.
4. El programador construye ese valor de negocio

Tabla 3.3: Ventajas y Desventajas de XP

Elaborado por: Investigador

Ventajas	Desventajas
▪ Programación organizada ▪ Software estable debido a continuas pruebas ▪ Minimiza errores ▪ Aplicación rápida de cambios	▪ Mayor esfuerzo de trabajo ▪ Autodisciplina en el desarrollo ▪ Requeriere mucha atención y tiempo

3.1.4. Scrum

Scrum es un proceso en el que se aplican de manera regular un conjunto de buenas prácticas para trabajar colaborativamente, en equipo, y obtener el mejor resultado posible de un proyecto. En Scrum se realizan entregas parciales y regulares del producto final, priorizadas por el beneficio que aportan al receptor del proyecto. Un proyecto de desarrollo se puede llevar a cabo mediante uno o más equipos Scrum. Un equipo Scrum está formado por personas que juegan tres tipos de roles: Product Owner, Scrum Master y Development team menmber [27].

3.1.4.1. Roles Scrum

Product Owner

Un product owner es uno de los futuros usuarios del sistema, alguien que sabe lo que quieren los usuarios del sistema en desarrollo. El product owner es clave en un proyecto ágil y si no realiza su trabajo puede poner en riesgo el proyecto [27].

Scrum Master

Encargado de velar por que todos los participantes del proyecto sigan los valores y principios ágiles, las reglas y proceso de Scrum y guiar la colaboración intra-equipo y con el cliente [27].

Equipo

El equipo se encarga del desarrollo de las diferentes funcionalidades del sistema [27].

Historias de Usuario

Describe la funcionalidad que será útil para el usuario, o comprador, de un sistema software. La conversación que conllevan, ya que son una herramienta para interactuar. Se caracteriza por ser pequeña, memorizable y que pueda ser desarrollada en corto tiempo [27].

Tabla 3.4: Artefactos SCRUM

Elaborado por: [27]

Artefacto	**Característica**
Sprint	▪ Su duración máxima es de 30 días. ▪ Se llevan a cabo las tareas pre establecidas y no se puede modificar el trabajo acordado ▪ Sólo el Scrum Master puede abortar un sprint si lo considera no viable
Product Backlog	▪ Especifica la serie de tareas que se van a desarrollar según los requisitos señalados ▪ Estas tareas tienen una duración de entre 4 y 6 hrs. de trabajo ▪ Las de mayor duración intentar descomponerlas en Sub-Tareas dentro de ese rango de tiempo

Tabla 3.5: Ventajas y desventajas de SCRUM

Elaborado por: Investigador

Ventajas	**Desventajas**
▪ Gestión de expectativas del usuario ▪ Flexibilidad y adaptación a cambios ▪ Muestra el progreso del sprint	▪ Dominio a fondo de principios y roles del equipo ▪ Depende del SCRUM Master ▪ Requiere tareas y plazos

3.1.5. Comparativa entre Scrum y XP

Se muestra una comparativa entre las metodologías de desarrolo ágil, destacando las principales características de cada metodología.

Tabla 3.6: Comparativa entre Scrum y XP

Elaborado por: Investigador

SCRUM	XP
▪ No se admiten cambios en el sprint ▪ Una vez que se establecen las tareas para un determinado sprint, el equipo determina la secuencia en la que se desarrollarán. ▪ El Scrum Master es el responsable de terminar un sprint ▪ La validación del software se completa al final de cada sprint	▪ Siempre que el equipo no haya comenzado a trabajar en una función en particular, se puede intercambiar una nueva función de tamaño equivalente en la interacción a cambio de una función no iniciada ▪ Las tareas se toman en un estricto orden de prioridad ▪ Los desarrolladores pueden modificar o refactorizar partes del código según sea necesario ▪ El software debe validarse en todo momento

En la investigación realizada por Carmen Penadés y Oatricio Orlando Letelier, se obtiene la tabla 3.7 en cual se compara las distintas aproximaciones ágiles en base a tres parámetros: vista del sistema como algo cambiante, tener en cuenta la colaboración entre los miembros del equipo y características más específicas de la propia metodología como son simplicidad, excelencia técnica, resultados, adaptabilidad [30]. se asigan valoraciones numéricas del 1 al 1, siendo 1 valor mas bajo y 5 el valor mas alto.

Tabla 3.7: Ranking de agilidad

Elaborado por: [30]

Parámetros	**Scrum**	**XP**
Sistema como algo cambiante	5	5
Colaboración	5	5
Resultados	5	5
Simplicidad	5	5
Adaptabilidad	4	3
Excelencia técnica	3	4
Prácticas de Colaboración	4	5
Total	**31**	**32**

3.1.6. Metodología de desarrollo Seleccionada

Una vez analizado las metodologías de desarrollo ágiles con mayor popularidad en el ámbito del desarrollo de sistemas. Se ha seleccionado el enfoque XP como la mejor metodología a ser aplicada teniendo en cuenta lo siguiente.

- Contribuye la construcción de un producto muy ajustado a los requerimientos del cliente, ya que algunas especificaciones pueden variar durante el desarrollo del proyecto.
- La metodología XP se centra en la comunicación entre el equipo de desarrollo y el cliente, además de reutilizar el código y retroalimentación de este.
- Al trabajar con iteraciones cortas, permiten revisar de manera periódica el avance del proyecto, corrigiendo a tiempo los cuellos de botella, de esta manera se minimiza la tasa de error en el desarrollo del proyecto.

3.1.7. React

React, también conocida como React.js o ReactJS es una biblioteca JavaScript de código abierto diseñada para crear interfaces de usuario con el objetivo de facilitar el desarrollo de aplicaciones en una sola página. Es mantenido por Facebook y la comunidad de software libre. En el proyecto hay más de mil desarrolladores libres [31].

Intenta ayudar a los desarrolladores a construir aplicaciones que usan datos que cambian todo el tiempo. Su objetivo es ser sencillo, declarativo y fácil de combinar. React sólo maneja la interfaz de usuario en una aplicación; React es la Vista en un

contexto en el que se use el patrón MVC (Modelo-Vista-Controlador) o MVVM (Modelo-vista-modelo de vista). También puede ser utilizado con las extensiones de React-based que se encargan de las partes no-UI (que no forman parte de la interfaz de usuario) de una aplicación web [31].

Tabla 3.8: Ventajas y Desventajas de React

Elaborado por: Investigador

Ventajas	**Desventajas**
▪ Gran nivel de flexibilidad y máxima 'responsividad'. ▪ DOM Virtual (Modelo de Objetos del Documento) que permite convertir documentos en formato HTML, XHTML o XML ▪ Librería JavaScript 100 % código abierto con frecuentes actualizaciones ▪ Framework muy ligero ya que los datos del lado del usuario pueden representarse en el servidor simultáneamente	▪ Ausencia de documentación oficial. ▪ Requiere mucho tiempo para dominarlo. ▪ Se requiere conocimiento profundo para integración de interfaces.

3.1.8. Angular

Diseñado para aplicaciones web dinámicas, mantenido por Google, utiliza HTML como lenguaje de plantillas. . Los conceptos de enlace de datos e inyección de dependencias permiten reducir líneas de código y su ejecución es realizada dentro del navegador. Angular es 100 % JavaScript del lado del cliente, no es un simple sistema de plantillas, la razón se debe a "bidirectional data binding", enlace de datos bidireccional, que posee y se ejecuta de forma automática. La plantilla se compila en el navegador y el paso de compilación da como resultado una vista en vivo. Ya no se necesita que los desarrolladores deban sincronizar constantemente la vista con el modelo o viceversa [31].

Angular permite el uso de cualquier tipo de backend, desde Java, Python, Ruby, C# entre otros. Al trabajar con Angular es necesario utilizar una forma de

comunicación de ida y vuelta con el servidor, peticiones HTTP XML o JSON.

Tabla 3.9: Ventajas y Desventajas de Angular

Elaborado por: Investigador

Ventajas	**Desventajas**
▪ Gestión de dependencias. ▪ Arquitectura MVC. ▪ Simplicidad en la administración de componentes web. ▪ Corrección de errores en tiempo real ya que está escrito en TypeScript	▪ Curva de aprendizaje elevada ▪ Detallado y Complejo. ▪ Angular Cli carece de detalle

3.1.9. Comparativa entre Angular y React

Tabla 3.10: Comparativa entre Angular y React

Elaborado por: Investigador

Caracterísiticas	**React**	**Angular**
Tipo	Javascript	Superset de Javascript
Vinculación de datos	Unidireccional	Bidireccional
Modelo	DOM Virtual	Modelo MVC
Código	Abierto	Abierto
Creador	Facebook	Google
Escritura	Javascript	Typescript
Abstracción	Fuerte	Medio

3.1.10. Determinación de la tecnología Front-End

Una vez realizado la investigación pertinente sobre los framework Front-End que se encuentran en auge, en función de sus ventajas y desventajas, se ha seleccionado utilizar la tecnología Angular, esto ayudará al desarrollo de la aplicación web

Con Angular se puede desarrollar aplicaciones web moderna, y la mejor organización de componentes como ruteo, peticiones web api, inyección de dependencias, además usa HTML para la construcción de interfaces de usuario, separando la funcionalidad en archivos de typescript.

Dentro de las ventajas potenciales se destaca la reusabilidad que permite la creación de componentes reutilizables, además del testeo puesto que al crear componentes individuales se lleva un control de testeos de manera independiente. Finalmente al ser creado por Google la comunidad de desarrolladores dan soporte continuos a dicho framework que permite agilizar la construcción de aplicaciones.

3.1.11. PHP

PHP se usa ampliamente para el desarrollo de portales web y ofrece una amplia gama de marcos de alto rendimiento y rápidos de usar. Por lo tanto, los desarrolladores prefieren PHP para crear sitios web dinámicos a un costo asequible del mercado. Además, Zend Engine es un conjunto de componentes que forman PHP. El componente más importante de Zend Engine es Zend Virtual Machine, que incluye los componentes Zend Compiler y Zend Executor. También podemos agregar la extensión zend OPCache en esta categoría. Estos tres componentes son el corazón de PHP (o el cerebro de su elección), y son las partes básicas y más complejas de todo el código fuente de PHP el cual proporciona el compilador estándar de PHP, que se publica de forma gratuita bajo la licencia de PHP [32].

Tabla 3.11: Ventajas y Desventajas de PHP

Elaborado por: Investigador

Ventajas	**Desventajas**
▪ Programación de aplicaciones web flexible. ▪ Variedad de conexión de base de datos. ▪ Simplicidad en la administración de componentes web.	▪ Documentación inconsistente. ▪ Propenso a errores si no se tiene experiencia.

3.1.12. ASP.NET

Es un entorno para aplicaciones web desarrollado y comercializado por Microsoft, y los desarrolladores pueden usar este marco para crear sitios web, aplicaciones web y servicios web dinámicos. Enero de 2002 con la versión 1.0 de .NET Framework, que es el sucesor de la tecnología Active Server Pages (ASP). ASP.NET se basa en Common Language Runtime, lo que permite a los

programadores escribir código ASP.NET en cualquier idioma compatible con .NET Framework [33].

Tabla 3.12: Ventajas y Desventajas de ASP.NET

Elaborado por: Investigador

Ventajas	**Desventajas**
▪ Disminución de código. ▪ Soporte por comunidad de Microsoft. ▪ Arquitectura MVC ▪ Estabilidad en conexión SQL server. ▪ Codigficación algo fácil. ▪ Escrito en lenguajes .NET como c#.	▪ Problemas de compatibildiad con sistemas operativos libres. ▪ Velocidad de rendimiento menor en comparación a Linux.

3.1.13. Comparativa entre ASP.NET Y PHP

Tabla 3.13: Comparativa entre ASP.NET y PHP

Elaborado por: Investigador

Caracterísiticas	**ASP.NET**	**PHP**
Aprendizaje	Relativamente fácil	Medio
Código	Pagado	Abierto
Compatibilidad	Multiplataforma	Multiplataforma
Tareas	Multihilo	Hilo
Lenguaje	Orientado a objetos	No
Rendimiento	Medio	Medio
Bases de Datos	SQL, MYSQL	MYSQL

3.1.14. Determinación de la tecnología Back-End

La tecnología seleccionada para el presente proyecto será ASP.NET debido a sus grandes beneficios para el desarrollo de aplicaciones web y su conexión nativa con el sistema de gestión de base de datos SQL Server, el mismo que se encuentra licenciado en la Cooperativa San Martin.

ASP.NET es compatible con múltiples sistemas operativos como Linux, Windows y macOS, basado en el marco del lado servidor está diseñado para cumplir con los requisitos de páginas web dinámicas, así como servicios web.

3.1.15. Arquitectura de la aplicación

Mediante la arquitectura de la aplicación web se comprende la interacción de los componentes que lo conforman, a su vez, permite conocer las tecnologías que mantienen la comunicación. Para el presente proyecto se aplica el modelo cliente – servidor.

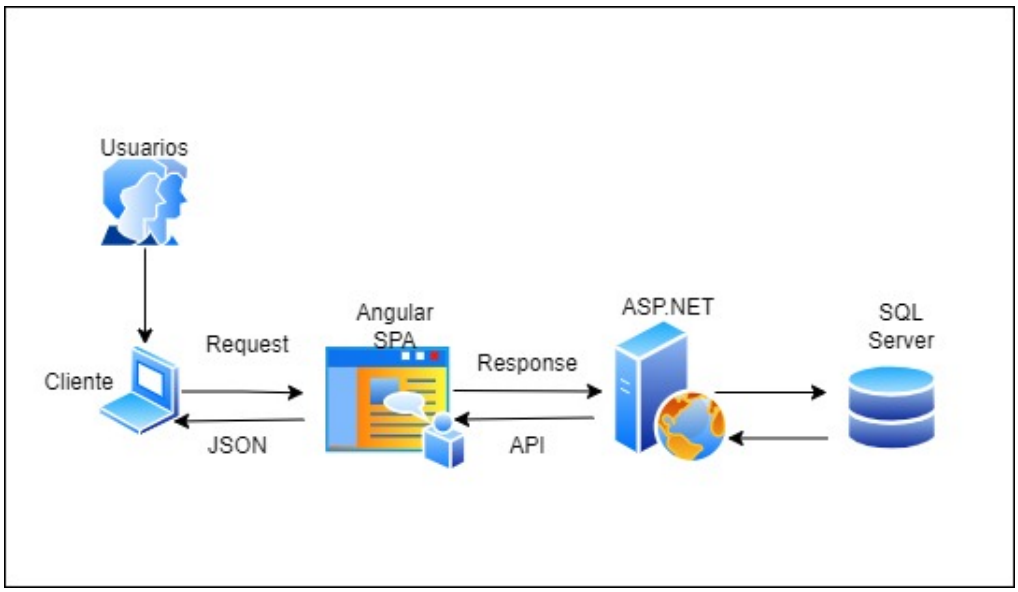


Figura 3.5: Arquitectura de la aplicación web
Elaborado por: Investigador

Cabe mencionar que el framework Angular adopta la arquitectura MVC (Modelo Vista Controlador), el mismo que interactúa mediante el consumo de API (Application Programming Interface) con el back-end basado en ASP.NET, quién se encargar de gestionar procesos y peticiones hacia la Base de Datos alojada en Sql Server.

3.2. Desarrollo de la propuesta

3.2.1. Fase I. Planificación

3.2.1.1. Definición de roles

Tabla 3.14: Definición de Roles

Elaborado por: Investigador

Nombre	Rol	Función	Rol XP
Gustavo Verdesoto	Tesista	Responsable de desarrollar la aplicación web, cumpliendo las fases necesarias para cubrir los requerimientos del cliente.	Programador
Ing. David Guevara, Mg	Tutor del trabajo de investigación	Responsable del seguimiento y cumplimiento de las actividades para la finalización de la aplicación web.	Coach
Ing. MSc. Francisco Moreta	Gerente General de la cooperativa "San Martín"	Persona encargada de validar y realizar las pruebas de aceptación.	Tester

3.2.1.2. Levantamiento de información

Luego de constatar la infraestructura y recursos tecnológicos necesarios para el desarrollo de la aplicación web, se mantuvo conversaciones y una entrevista semiestructurada ajustada al entrevistado con el objetivo de obtener la información necesaria para el levantamiento de requerimientos que necesita la cooperativa "San Martín", en la recaudación de valores por consumo de agua potable.

Mediante el levantamiento de requerimientos, usando modelado de procesos, se puede identificar claramente los actores, así como las funciones que cumplen dentro de cada proceso, facilitando al desarrollador a determinar roles de usuario con sus respectivas tareas.

3.2.1.3. Análisis de Resultados

En la entrevista aplicada al gerente general de la cooperativa "San Martín", el Ing. MSc. Francisco Moreta se logró obtener información como que la

institución, busca captar clientes a través de soluciones tecnológicas que permitan la recaudación de valores por consumo de agua potable.

Mediante la descripción de procesos, se identificó los roles principales que regirán en el sistema, de manera que existirán jerarquías a fin de establecer responsabilidades acordes a sus funciones. Dentro de los roles se determina los siguientes:

- Administrador
- Operador
- Recaudador
- Atención al Cliente

3.2.1.4. Historias de usuario

Es importante la descripción de historias de usuario para un desarrollo satisfactorio del proyecto, ya que se detalla la funcionalidad del software desde la perspectiva del usuario final.

Para la elaboración de cada una de las historias de usuario, se utilizará el siguiente modelo:

Tabla 3.15: Plantilla de historia de usuario

Elaborado por: Investigador

Historia de Usuario	
Número:	**Usuario:**
Nombre:	
Prioridad en el negocio:	**Puntos estimados:**
Riesgo en el desarrollo:	**Iteración asignada:**
Programador responsable:	
Descripción:	
Observación:	

Esta plantilla está formada por los siguientes elementos:

- **Número:** Identificador alfanumérico que se asigna a cada historia de usuario.
- **Usuario:** Persona responsable asignada a la historia de usuario.

- **Nombre:** Título que se le da a la historia de usuario.
- **Prioridad en el negocio:** Valores que se asignan de acuerdo a la necesidad del usuario final (Alta, Media, Baja).
- **Puntos estimados:** Es el número de días estimado en el que se desarrollará la historia de usuario.
- **Riesgo en el desarrollo:** Valores que se asignan de acuerdo al riesgo que pueda tener al desarrollar la historia de usuario.
- **Iteración asignada:** Número de iteración en la que se establece la historia de usuario.
- **Programador responsable:** Nombre de la persona encargada de desarrollar la historia de usuario.
- **Descripción:** El usuario detalla con sus propias palabras lo que se va a hacer más no el cómo hacer.
- **Observación:** Detalle adicional que se le asigne a la historia de usuario.

Historias de Usuario

Tabla 3.16: Historia de Usuario - Establecer la estructura del proyecto

Elaborado por: Investigador

Historia de Usuario	
Número: 01	**Usuario:** Desarrollador
Nombre: Establecer la estructura del proyecto	
Prioridad en el negocio: Alta	**Puntos estimados:** 5
Riesgo en el desarrollo: Alto	**Iteración asignada:** 1
Programador responsable: Christian Gustavo Verdesoto Guaman	
Descripción: Identificar la estructura que tendrá el proyecto, así como las herramientas necesarias para la comunicación entre componentes.	
Observación:	

Tabla 3.17: Historia de Usuario - Diseño de la base de datos

Elaborado por: Investigador

Historia de Usuario	
Número: 02	**Usuario:** Desarrollador
Nombre: Diseño de la base de datos	
Prioridad en el negocio: Alta	**Puntos estimados:** 5
Riesgo en el desarrollo: Alto	**Iteración asignada:** 1
Programador responsable: Christian Gustavo Verdesoto Guaman	
Descripción: Se diseñará el modelado relacional de la base de datos.	
Observación: La base de datos se diseñará en SQL Server.	

Tabla 3.18: Historia de Usuario - Inicio de sesión

Elaborado por: Investigador

Historia de Usuario	
Número: 03	**Usuario:** Desarrollador
Nombre: Inicio de sesión	
Prioridad en el negocio: Alta	**Puntos estimados:** 5
Riesgo en el desarrollo: Alto	**Iteración asignada:** 1
Programador responsable: Christian Gustavo Verdesoto Guaman	
Descripción: El usuario podrá acceder a la aplicación web digitando sus credenciales como lo son usuario y contraseña.	
Observación: Usuario debe ser creado previamente por el administrador de la aplciación web.	

Tabla 3.19: Historia de Usuario - Pantalla principal

Elaborado por: Investigador

Historia de Usuario	
Número: 04	**Usuario:** Desarrollador
Nombre: Pantalla principal	
Prioridad en el negocio: Alta	**Puntos estimados:** 5
Riesgo en el desarrollo: Alto	**Iteración asignada:** 1
Programador responsable: Christian Gustavo Verdesoto Guaman	
Descripción: La pantalla principal o dashboard que tendrá el sistema para que el usuario pueda interactuar mediante módulos. ▪ Usuarios ▪ Personas ▪ Lecturas ▪ Recaudaciones ▪ Reportes	
Observación: Los módulos se habilitarán en función del rol de cada usuario.	

Tabla 3.20: Historia de Usuario - Registrar un nuevo usuario

Elaborado por: Investigador

Historia de Usuario	
Número: 05	**Usuario:** Desarrollador
Nombre: Registrar un nuevo usuario	
Prioridad en el negocio: Alta	**Puntos estimados:** 4
Riesgo en el desarrollo: Alto	**Iteración asignada:** 2
Programador responsable: Christian Gustavo Verdesoto Guaman	
Descripción: En esta pantalla el administrador podrá registrar un nuevo usuario para acceder a la aplicación web llenando los siguientes campos. ▪ Nombre Usuario ▪ Agencia ▪ Empleado ▪ Perfil	
Observación: El nuevo usuario a ser creado deberá estar asociado a un empleado, el mismo que estará registrado previamente.	

Tabla 3.21: Historia de Usuario - Cambiar contraseña

Elaborado por: Investigador

Historia de Usuario	
Número: 06	**Usuario:** Desarrollador
Nombre: Cambiar contraseña.	
Prioridad en el negocio: Alta	**Puntos estimados:** 4
Riesgo en el desarrollo: Alto	**Iteración asignada:** 2
Programador responsable: Christian Gustavo Verdesoto Guaman.	
Descripción: En ésta pantalla los usuarios registrados, tienen la opción de cambiar su contraseña.	
Observación: La nueva contraseña debe cumplir parámetros básicos de seguridad como longitud mínima y caracteres especiales.	

Tabla 3.22: Historia de Usuario - Activar y cancelar usuarios

Elaborado por: Investigador

Historia de Usuario	
Número: 07	**Usuario:** Desarrollador
Nombre: Activar y cancelar usuarios	
Prioridad en el negocio: Media	**Puntos estimados:** 3
Riesgo en el desarrollo: Bajo	**Iteración asignada:** 2
Programador responsable: Christian Gustavo Verdesoto Guaman	
Descripción: En esta pantalla el usuario puede cancelar y activar usuarios para el acceso a la aplicación web.	
Observación: Para realizar esta acción, el usuario deberá estar registrado previamente por el administrador.	

Tabla 3.23: Historia de Usuario - Añadir nueva persona natural

Elaborado por: Investigador

Historia de Usuario	
Número: 08	**Usuario:** Desarrollador
Nombre: Añadir nueva persona natural.	
Prioridad en el negocio: Alta	**Puntos estimados:** 4
Riesgo en el desarrollo: Alto	**Iteración asignada:** 2
Programador responsable: Christian Gustavo Verdesoto Guaman.	
Descripción: En esta pantalla el usuario puede ingresar todos los campos necesarios para el registro de una persona natural. ▪ Numero de Identificación ▪ Apellidos y Nombres ▪ Estado Civil ▪ Sexo ▪ Fecha de Nacimiento ▪ Instrucción ▪ Número de Teléfono ▪ Correo Electrónico ▪ Datos Conyugue ▪ Dirección	
Observación: Se definirán campos obligatorios para el registro de un nuevo contribuyente natural, los datos de conyugue serán opcionales al igual que números de teléfono y correo electrónico.	

Tabla 3.24: Historia de Usuario - Listado de personas naturales

Elaborado por: Investigador

Historia de Usuario	
Número: 09	**Usuario:** Desarrollador
Nombre: Listado de personas naturales	
Prioridad en el negocio: Media	**Puntos estimados:** 3
Riesgo en el desarrollo: Medio	**Iteración asignada:** 2
Programador responsable: Christian Gustavo Verdesoto Guaman	
Descripción: En esta pantalla el usuario puede observar el listado de los contribuyentes naturales registrados en la aplicación web.	
Observación: Se creará un cuadro de búsqueda con filtros como nombres o número de identificación. Además la tabla permitirá ordenar de manera ascendente o descendente por nombres.	

Tabla 3.25: Historia de Usuario - Editar datos de un contribuyente natural

Elaborado por: Investigador

Historia de Usuario	
Número: 10	**Usuario:** Desarrollador
Nombre: Editar datos de un contribuyente natural.	
Prioridad en el negocio: Alta	**Puntos estimados:** 4
Riesgo en el desarrollo: Medio	**Iteración asignada:** 2
Programador responsable: Christian Gustavo Verdesoto Guaman-	
Descripción: En esta pantalla el usuario puede actualizar datos de un contribuyente como son: ▪ Nombres ▪ Fecha de Nacimiento ▪ Estado Civil ▪ Instrucción ▪ Dirección ▪ Datos de Conyugue	
Observación: Para acceder a esta pantalla el usuario debera buscar a un contribuyente del listado donde podrá presionar sobre el botón editar.	

Tabla 3.26: Historia de Usuario - Añadir nueva persona jurídica

Elaborado por: Investigador

Historia de Usuario	
Número: 11	**Usuario:** Desarrollador
Nombre: Añadir nueva persona jurídica	
Prioridad en el negocio: Alta	**Puntos estimados:** 4
Riesgo en el desarrollo: Alto	**Iteración asignada:** 3
Programador responsable: Christian Gustavo Verdesoto Guaman	
Descripción: En esta pantalla el usuario los campos necesarios para el registro de un nuevo contribuyente jurídico. ▪ Numero de Identificación ▪ Razon Social ▪ Actividad Economica ▪ Fecha de creación ▪ Números de Teléfono ▪ Correo electrónico ▪ Dirección	
Observación: Se definirá campos obligatorios y opcionales como lo son números de teléfono y correo electrónico.	

Tabla 3.27: Historia de Usuario - Añadir representantes a una persona jurídica

Elaborado por: Investigador

Historia de Usuario	
Número: 12	**Usuario:** Desarrollador
Nombre: Añadir representantes a una persona jurídica	
Prioridad en el negocio: Alta	**Puntos estimados:** 4
Riesgo en el desarrollo: Medio	**Iteración asignada:** 3
Programador responsable: Christian Gustavo Verdesoto Guaman	
Descripción: En el módulo de Persona Jurídica, esta disponible el apartado de representantes, donde el usuario digita el número de cédula y se deslegaran los datos de la persona, puede agregar dos o más representantes.	
Observación: Se define dos o más representantes, los mismos que deberán estar registrados previamente en la pantalla de persona natural.	

Tabla 3.28: Historia de Usuario - Listado de personas jurídicas

Elaborado por: Investigador

Historia de Usuario	
Número: 13	**Usuario:** Desarrollador
Nombre: Listado de personas jurídicas	
Prioridad en el negocio: Media	**Puntos estimados:** 3
Riesgo en el desarrollo: Medio	**Iteración asignada:** 3
Programador responsable: Christian Gustavo Verdesoto Guaman	
Descripción: En esta pantalla el usuario puede observar el listado de los contribuyentes jurídicos registrados en la aplicación web.	
Observación: Se creará un cuadro de búsqueda con filtros como razón social o número de ruc. Además la tabla permitirá ordenar de manera ascendente o descendente por nombres.	

Tabla 3.29: Historia de Usuario - Editar datos de una persona jurídica

Elaborado por: Investigador

Historia de Usuario	
Número: 14	**Usuario:** Desarrollador
Nombre: Editar datos de una persona jurídica	
Prioridad en el negocio: Alta	**Puntos estimados:** 4
Riesgo en el desarrollo: Medio	**Iteración asignada:** 3
Programador responsable: Christian Gustavo Verdesoto Guaman	
Descripción: En esta pantalla el usuario puede actualizar datos de un contribuyente jurídico como son: ▪ Razón Social ▪ Actividad Económica ▪ Fecha de Creación ▪ Números de Teléfono ▪ Correo Electrónico.	
Observación: Para acceder a esta pantalla el usuario, deberá seleccionar un contribuyente jurídico del listado, presionar la opción editar.	

Tabla 3.30: Historia de Usuario - Eliminar representante de una persona jurídica

Elaborado por: Investigador

Historia de Usuario	
Número: 15	**Usuario:** Desarrollador
Nombre: Eliminar representante de una persona jurídica	
Prioridad en el negocio: Alta	**Puntos estimados:** 3
Riesgo en el desarrollo: Bajo	**Iteración asignada:** 3
Programador responsable: Christian Gustavo Verdesoto Guaman	
Descripción: En esta pantalla el usuario puede actualizar a representantes de un contribuyente jurídico, eliminar o agregar representantes.	
Observación: Para seleccionar a un representante, este deberá estar registrado previamente en el listado de persona natural.	

Tabla 3.31: Historia de Usuario - Registrar una nueva cuenta

Elaborado por: Investigador

Historia de Usuario	
Número: 16	**Usuario:** Desarrollador
Nombre: Registrar una nueva cuenta	
Prioridad en el negocio: Alta	**Puntos estimados:** 4
Riesgo en el desarrollo: Alto	**Iteración asignada:** 4
Programador responsable: Christian Gustavo Verdesoto Guaman	
Descripción: En esta pantalla el usuario debe registrar los datos necesarios para una nueva cuenta como son: ▪ Codigo medidor ▪ Datos del contribuyente ▪ Tarifa ▪ Ubicación ▪ Dirección	
Observación: Todos los campos son obligatorios.	

Tabla 3.32: Historia de Usuario - Listado de cuentas

Elaborado por: Investigador

Historia de Usuario	
Número: 17	**Usuario:** Desarrollador
Nombre: Listado de cuentas	
Prioridad en el negocio: Alta	**Puntos estimados:** 3
Riesgo en el desarrollo: Medio	**Iteración asignada:** 4
Programador responsable: Christian Gustavo Verdesoto Guaman	
Descripción: En esta pantalla el usuario puede observar el listado de las cuentas registradas en la aplicación web.	
Observación: Se creará un cuadro de búsqueda con filtros como nombres o número de cuenta. Además la tabla permitirá ordenar de manera ascendente o descendente por nombres.	

Tabla 3.33: Historia de Usuario - Activar y cancelar cuentas

Elaborado por: Investigador

Historia de Usuario	
Número: 18	**Usuario:** Desarrollador
Nombre: Activar y cancelar cuentas	
Prioridad en el negocio: Media	**Puntos estimados:** 3
Riesgo en el desarrollo: Medio	**Iteración asignada:** 4
Programador responsable: Christian Gustavo Verdesoto Guaman	
Descripción: En esta pantalla se mostrará las cuentas activas, y el susuario podrá seleccionar la cuenta que se desea cancelar.	
Observación:	

Tabla 3.34: Historia de Usuario - Registrar una nueva lectura

Elaborado por: Investigador

Historia de Usuario	
Número: 19	**Usuario:** Desarrollador
Nombre: Registrar una nueva lectura	
Prioridad en el negocio: Alta	**Puntos estimados:** 3
Riesgo en el desarrollo: Alto	**Iteración asignada:** 4
Programador responsable: Christian Gustavo Verdesoto Guaman	
Descripción: En esta pantalla se registra la lectura de la cantidad consumida en metros cúbicos, de una cuenta.	
Observación: Se mostrará los datos del contribuyente a la cual esta asociada la cuenta.	

Tabla 3.35: Historia de Usuario - Listado de lecturas

Elaborado por: Investigador

Historia de Usuario	
Número: 20	**Usuario:** Desarrollador
Nombre: Listado de lecturas	
Prioridad en el negocio: Media	**Puntos estimados:** 3
Riesgo en el desarrollo: Medio	**Iteración asignada:** 4
Programador responsable: Christian Gustavo Verdesoto Guaman	
Descripción: En esta pantalla el usuario puede observar el listado de las lecturas registradas en un intervalo de fecha.	
Observación:	

Tabla 3.36: Historia de Usuario - Cobro de planillas

Elaborado por: Investigador

Historia de Usuario	
Número: 21	**Usuario:** Desarrollador
Nombre: Cobro de planillas	
Prioridad en el negocio: Alta	**Puntos estimados:** 5
Riesgo en el desarrollo: Alto	**Iteración asignada:** 4
Programador responsable: Christian Gustavo Verdesoto Guaman	
Descripción: En esta pantalla se podrá realizar las recaudaciones de valores de planillas pendientes de pago.	
Observación: Se mostrará los datos de la cuenta y contribuyente para minimizar errores de cobro.	

Tabla 3.37: Historia de Usuario -Listado de planillas

Elaborado por: Investigador

Historia de Usuario	
Número: 22	**Usuario:** Desarrollador
Nombre: Listado de planillas	
Prioridad en el negocio: Alta	**Puntos estimados:** 3
Riesgo en el desarrollo: Alto	**Iteración asignada:** 5
Programador responsable: Christian Gustavo Verdesoto Guaman	
Descripción: En esta pantalla el usuario puede observar el listado de las recaudaciones que se han efectuado durante un intervalo de fecha.	
Observación:	

Tabla 3.38: Historia de Usuario -Registrar cierre de caja

Elaborado por: Investigador

Historia de Usuario	
Número: 23	**Usuario:** Desarrollador
Nombre: Registrar cierre de caja	
Prioridad en el negocio: Alta	**Puntos estimados:** 4
Riesgo en el desarrollo: Alto	**Iteración asignada:** 5
Programador responsable: Christian Gustavo Verdesoto Guaman	
Descripción: En esta pantalla el recaudador, deberá conciliar los valores recaudados durante el día y verificar que la sumatoria de valores coincida con la cantidad de dinero físico.	
Observación:	

Tabla 3.39: Historia de Usuario - Listado de cierres de caja

Elaborado por: Investigador

Historia de Usuario	
Número: 24	**Usuario:** Desarrollador
Nombre: Listado de cierres de caja	
Prioridad en el negocio: Alta	**Puntos estimados:** 3
Riesgo en el desarrollo: Alto	**Iteración asignada:** 5
Programador responsable: Christian Gustavo Verdesoto Guaman	
Descripción: En esta pantalla el usuario puede observar los cierres de caja registrados, adempas le permitira reimprimir el cierre registrado de un recaudador.	
Observación:	

Tabla 3.40: Historia de Usuario - Generación de Reportes

Elaborado por: Investigador

Historia de Usuario	
Número: 25	**Usuario:** Desarrollador
Nombre: Listado de Recaudaciones	
Prioridad en el negocio: Alta	**Puntos estimados:** 5
Riesgo en el desarrollo: Bajo	**Iteración asignada:** 5
Programador responsable: Christian Gustavo Verdesoto Guaman	
Descripción: Se definirá reportes de valores recaudados en un intervalo de fecha	
Observación:	

3.2.1.5. Estimación de las historias de usuario

Tabla 3.41: Estimación de las Historias de usuario

Elaborado por: Investigador

Número	**Nombre Historia**	**Tiempo Estimado**	
		Días	**Horas**
01	Establecer la estructura del proyecto	5	25
02	Diseño de la base de datos	4	20
03	Inicio de sesión	5	25
04	Pantalla principal	8	40
05	Registrar un nuevo usuario	4	20
06	Cambiar contraseña	3	15
07	Activar y cancelar usuarios	2	10
08	Añadir nueva persona natural	5	25
09	Listado de personas naturales	3	15
10	Editar datos de una persona natural	4	20
11	Añadir nueva persona jurídica	4	20
12	Añadir representantes a una persona jurídica	6	30
13	Listado de personas jurídicas	3	15
14	Editar datos de una persona jurídica	4	20
15	Eliminar representante de una persona jurídica	3	15
16	Registrar una nueva cuenta	4	20
17	Listado de cuentas	3	15
18	Activar y cancelar cuentas	2	10
19	Registrar una nueva lectura	4	20
20	Listado de lecturas	3	15
21	Cobro de planillas	6	30
22	Listado de planillas	3	15
23	Registrar cierre de caja	4	20
24	Listado de cierres de caja	3	15
25	Listado de Recaudaciones	8	40
Tiempo estimado		**103**	**515**

3.2.1.6. Plan de Entrega

El objetivo de generar un plan de entrega es fijar tiempos de finalización de historias de usuario, aplicando la metodlogía XP, tomando en cuenta que cada iteración tendrá un tiempo estimado de cuatro semanas, durante días laborables.

Tabla 3.42: Plan de entrega

Elaborado por: Investigador

N°	**Nombre Historia**	**Tiempo Estimado**		**Iteración Asignada**				
		Días	**Horas**	**1**	**2**	**3**	**4**	**5**
01	Establecer la estructura del proyecto	5	25	X				
02	Diseño de la base de datos	4	20	X				
03	Inicio de sesión	5	25	X				
04	Pantalla principal	8	40	X				
05	Registrar un nuevo usuario	4	20		X			
06	Cambiar contraseña	3	15		X			
07	Activar y cancelar usuarios	2	10		X			
08	Añadir nueva persona natural	5	25		X			
09	Listado de personas naturales	3	15		X			
10	Editar datos de una persona natural	4	20		X			
11	Añadir nueva persona jurídica	4	20			X		
12	Añadir representantes a una persona jurídica	6	30			X		
13	Listado de personas jurídicas	3	15			X		
14	Editar datos de una persona jurídica	4	20			X		
15	Eliminar representante de una persona jurídica	3	15			X		
16	Registrar una nueva cuenta	4	20				X	
17	Listado de cuentas	3	15				X	
18	Activar y cancelar cuentas	2	10				X	
19	Registrar una nueva lectura	4	20				X	
20	Listado de lecturas	3	15				X	
21	Cobro de planillas	6	30				X	
22	Listado de planillas	3	15					X
23	Registrar cierre de caja	4	20					X
24	Listado de cierres de caja	3	15					X
25	Listado de recaudaciones	8	40					X

3.2.1.7. Plan de Iteraciones

Tabla 3.43: Plan de Iteraciones

Elaborado por: Investigador

Iteración	**N°**	**Nombre Historia**	**Prioridad**	**Riesgo**	**Estado de desarrollo**	**Pruebas**
Primera	01	Establecer la estructura del proyecto	Alta	Alto	Completo	Aprobado
	02	Diseño de la base de datos	Alta	Alto	Completo	Aprobado
	03	Inicio de sesión	Alta	Alto	Completo	Aprobado
	04	Pantalla principal	Alta	Alto	Completo	Aprobado
Segunda	05	Registrar un nuevo usuario	Alta	Alto	Completo	Aprobado
	06	Cambiar contraseña	Alta	Alto	Completo	Aprobado
	07	Activar y cancelar usuarios	Media	Bajo	Completo	Aprobado
	08	Añadir nueva persona natural	Alta	Alto	Completo	Aprobado
	09	Listado de personas naturales	Media	Medio	Completo	Aprobado
	10	Editar datos de una persona natural	Alta	Medio	Completo	Aprobado
Tercera	11	Añadir nueva persona jurídica	Alta	Alto	Completo	Aprobado
	12	Añadir representantes a una persona jurídica	Alta	Medio	Completo	Aprobado
	13	Listado de personas jurídicas	Media	Medio	Completo	Aprobado
	14	Editar datos de una persona jurídica	Alta	Medio	Completo	Aprobado
	15	Eliminar representante de una persona jurídica	Alta	Bajo	Completo	Aprobado
Cuarta	16	Registrar una nueva cuenta	Alta	Alto	Completo	Aprobado
	17	Listado de cuentas	Alta	Medio	Completo	Aprobado
	18	Activar y cancelar cuentas	Media	Medio	Completo	Aprobado
	19	Registrar una nueva lectura	Alta	Alto	Completo	Aprobado
	20	Listado de lecturas	Media	Medio	Completo	Aprobado
	21	Cobro de planillas	Alta	Alto	Completo	Aprobado
Quinta	22	Listado de planillas	Alta	Alto	Completo	Aprobado
	23	Registrar cierre de caja	Alta	Alto	Completo	Aprobado
	24	Listado de cierres de caja	Alta	Alto	Completo	Aprobado
	25	Listado de recaudaciones	Alta	Medio	Completo	Aprobado

3.2.2. Fase II. Diseño

3.2.2.1. Tarjetas CRC (Clase, Responsabilidad y Colaboración)

Las tarjetas CRC son parte de la metodología XP. Una forma orientada a objetos de crear una organización es diseñar una tarjeta CRC(Clase-Responsabilidad-Colaboración) por cada historia de usuario. Proporciona funcionalidad directa al negocio, una clase es una persona, cosa, evento, concepto, pantalla o mensaje y una responsabilidad de la clase son las cosas que se conocen y las que se realizan por los atributos y métodos, los colaboradores de una clases con las que trabajan en conjunto para llevar a cabo sus responsabilidades.

Inicio de Sesión

Tabla 3.44: Tarjeta CRC - Inicio de Sesión

Elaborado por: Investigador

Inicio de Sesión	
Responsabilidad	**Colaboradores**
Validar usuario y contraseña	Capa de acceso a datos. Métodos de validación de datos.
Observaciones:	

Pantalla principal

Tabla 3.45: Tarjeta CRC - Pantalla principal

Elaborado por: Investigador

Pantalla Principal	
Responsabilidad	**Colaboradores**
Obtener información menu principal. Visualizar información de usuario autenticado	Capa de acceso a datos. Inicio de sesión.
Observaciones:	

Registrar un nuevo usuario

Tabla 3.46: Tarjeta CRC - Registrar un nuevo usuario

Elaborado por: Investigador

Registrar un nuevo usuario	
Responsabilidad	**Colaboradores**
Obtener la información del usuario. Validar datos ingresados. Guardar información del nuevo usuario.	Capa de acceso a datos. Métodos de validación de datos.
Observaciones:	

Cambiar contraseña

Tabla 3.47: Tarjeta CRC - Cambiar contraseña

Elaborado por: Investigador

Cambiar contraseña	
Responsabilidad	**Colaboradores**
Obtener la contraseña anterior del usuario. Validar la nueva contraseña. Actualizar la contraseña del usuario autenticado.	Capa de acceso a datos. Métodos de validación de datos.
Observaciones:	

Activar y cancelar usuarios

Tabla 3.48: Tarjeta CRC - Activar y cancelar usuarios

Elaborado por: Investigador

Inicio de Sesión	
Responsabilidad	**Colaboradores**
Obtener la información de los usuarios registrados. Cambiar el estado de un usuario.	Capa de acceso a datos.
Observaciones:	

Añadir nueva persona natural

Tabla 3.49: Tarjeta CRC - Añadir nueva persona natural

Elaborado por: Investigador

Añadir nueva persona natural	
Responsabilidad	**Colaboradores**
Obtener información de la persona. Validar los datos ingresados. Guardar los datos de una nueva persona natural	Capa de acceso a datos. Métodos de validación de datos.
Observaciones:	

Listado de personas naturales

Tabla 3.50: Tarjeta CRC - Listado de personas naturales

Elaborado por: Investigador

Listado de personas naturales	
Responsabilidad	**Colaboradores**
Obtener información de las personas naturales registradas. Realizar búsquedas dinámicas.	Capa de acceso a datos.
Observaciones:	

Editar datos de una persona natural

Tabla 3.51: Tarjeta CRC - Editar datos de una persona natural

Elaborado por: Investigador

Editar datos de una persona natural	
Responsabilidad	**Colaboradores**
Cargar la información de la persona natural. Validar los nuevos datos ingresados. Guardar los nuevos datos de la persona natural.	Capa de acceso a datos. Listado de personas naturales. Métodos de validación de datos.
Observaciones:	

Añadir nueva persona jurídica

Tabla 3.52: Tarjeta CRC - Añadir nueva persona jurídica

Elaborado por: Investigador

Añadir nueva persona jurídica	
Responsabilidad	**Colaboradores**
Obtener información de la persona jurídica. Validar los datos ingresados. Guardar temporalmente los datos de la persona jurídica.	Capa de acceso a datos. Métodos de validación de datos.
Observaciones:	

Añadir representantes a una persona jurídica

Tabla 3.53: Tarjeta CRC - Añadir representantes a una persona jurídica

Elaborado por: Investigador

Añadir representantes a una persona jurídica	
Responsabilidad	**Colaboradores**
Obtener información de los representantes. Validar datos ingresados. Guardar los datos de la persona jurídica y sus representantes.	Capa de acceso a datos. Listado de personas naturales. Métodos de validación de datos.
Observaciones:	

Listado de personas jurídicas

Tabla 3.54: Tarjeta CRC - Listado de personas jurídicas

Elaborado por: Investigador

Listado de personas jurídicas	
Responsabilidad	**Colaboradores**
Obtener información de las personas jurídicas registradas. Realizar búsquedas dinámicas.	Capa de acceso a datos.
Observaciones:	

Editar datos de una persona jurídica

Tabla 3.55: Tarjeta CRC - Editar datos de una persona jurídica

Elaborado por: Investigador

Editar datos de una persona jurídica	
Responsabilidad	**Colaboradores**
Cargar la información de la persona jurídica. Validar los nuevos datos ingresados. Guardar temporalmente los datos de la persona jurídica.	Capa de acceso a datos. Listado de personas jurídicas. Métodos de validación de datos.
Observaciones:	

Eliminar representante de una persona jurídica

Tabla 3.56: Tarjeta CRC - Eliminar representante de una persona jurídica

Elaborado por: Investigador

Eliminar representante de una persona jurídica	
Responsabilidad	**Colaboradores**
Eliminar un representante de una persona jurídica. Actualizar representantes.	Capa de acceso a datos. Añadir representante.
Observaciones:	

Registrar una nueva cuenta

Tabla 3.57: Tarjeta CRC - Registrar una nueva cuenta

Elaborado por: Investigador

Registrar una nueva cuenta	
Responsabilidad	**Colaboradores**
Obtener la información de una nueva cuenta, Validar los datos ingresados. Guardar datos de una nueva cuenta.	Capa de acceso a datos. Métodos de validación de datos.
Observaciones:	

Listado de cuentas

Tabla 3.58: Tarjeta CRC - Inicio de Sesión

Elaborado por: Investigador

Listado de cuentas	
Responsabilidad	**Colaboradores**
Obtener información de las cuentas registradas. Aplicar búsquedas dinámicas.	Capa de acceso a datos.
Observaciones:	

Activar y cancelar cuentas

Tabla 3.59: Tarjeta CRC - Activar y cancelar cuentas

Elaborado por: Investigador

Activar y cancelar cuentas	
Responsabilidad	**Colaboradores**
Obtener información de las cuentas. Cambiar el estado de la cuenta.	Capa de acceso a datos. Listado de cuentas.
Observaciones:	

Registrar una nueva lectura

Tabla 3.60: Tarjeta CRC - Registrar una nueva lectura

Elaborado por: Investigador

Registrar una nueva lectura	
Responsabilidad	**Colaboradores**
Obtener información de una lectura. Validar datos ingresados. Guardar datos de una nueva lectura	Capa de acceso a datos. Métodos de validación de datos.
Observaciones:	

Listado de lecturas

Tabla 3.61: Tarjeta CRC - Listado de lecturas

Elaborado por: Investigador

Inicio de Sesión	
Responsabilidad	**Colaboradores**
Obtener información de la lecturas registradas. Aplicar búsquedas dinámicas.	Capa de acceso a datos.
Observaciones:	

Cobro de planillas

Tabla 3.62: Tarjeta CRC - Cobro de planillas

Elaborado por: Investigador

Cobro de planillas	
Responsabilidad	**Colaboradores**
Obtener información de las planillas pendientes. Seleccionar la planilla a recaudar. Actualizar el estado de la planilla.	Capa de acceso a datos. Listado de planillas.
Observaciones:	

Listado de Planillas

Tabla 3.63: Tarjeta CRC - Listado de planillas

Elaborado por: Investigador

Listado de planillas	
Responsabilidad	**Colaboradores**
Obtener información de las planillas. Aplicar filtros de fechas.	Capa de acceso a datos.
Observaciones:	

Registrar cierre de caja

Tabla 3.64: Tarjeta CRC - Registrar cierre de caj

Elaborado por: Investigador

Registrar cierre de caja	
Responsabilidad	**Colaboradores**
Obtener información de las recaudaciones realizadas. Validar datos ingresados. Coincidir valores de cierre.	Capa de acceso a datos. Listado de recaudaciones. Métodos de validación de datos.
Observaciones:	

Listado de cierre de caja

Tabla 3.65: Tarjeta CRC - Listado de cierre de caja

Elaborado por: Investigador

Listado de cierre de caja	
Responsabilidad	**Colaboradores**
Obtener información de cierre de caja. Aplicar filtros de fechas.	Capa de acceso a datos.
Observaciones:	

Listado de recaudaciones

Tabla 3.66: Tarjeta CRC - Listado de recaudaciones

Elaborado por: Investigador

Listado de recaudaciones	
Responsabilidad	**Colaboradores**
Obtener información de recudaciones. Aplicar filtros de fechas.	Capa de acceso a datos.
Observaciones:	

3.2.2.2. Iteraciones

Una vez que se ha completado el plan de entrega, las tareas se planifican y se agrupan a través de iteraciones, para que puedan ser corregidas de manera oportuna y así evitar en lo posible retrasos por errores que no han sido detectados a tiempo.

Iteración I

Tabla 3.67: Estimación de la primera iteración

Elaborado por: Investigador

Iteración	**N°**	**Nombre Historia**	**Tiempo estimado**	
			Días	**Horas**
Primera	01	Establecer la estructura del proyecto	5	25
	02	Diseño de la base de datos	4	20
	03	Inicio de sesión	5	25
	04	Pantalla principal	8	40
		Total	**22**	**110**

Establecer la estructura del proyecto

Hardware

- Computador de escritorio con sistema operativo Windows 11 Pro 64 bits
- Intel(R) Core(TM) i7-9700F CPU @ 3.00GHz 3.00 GHz
- Memoria RAM 16 GB

Software

- Sistema operativo de 64 bits, procesador basado en x64
- Angular CLI 12.2.11
- Node 14.18.1
- Package Manager npm 6.14.15
- Ambiente de desarrollo Visual Studio 2019
- Entity FrameworkCore 3.1.21
- Postman 9.31.9
- Sql Server Managment Studio 18
- Editor de código fuente Visual Studio Code
- Extensiones para Visual Studio Code

Creación del proyecto

Backend

Ingresando al ambiente de desarrollo Visual Studio 2019, se procede a crear el proyecto backend, el mismo que será de tipo ASP.NET Core Web Api. como se muestra en la Figura 3.3.

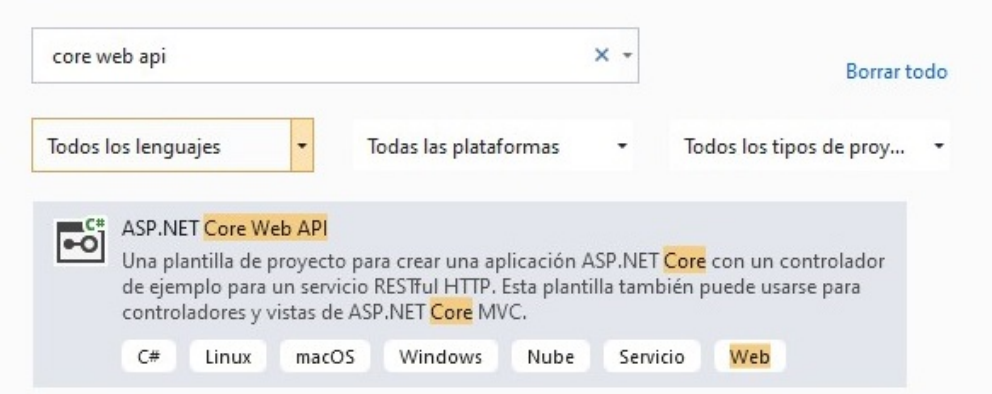


Figura 3.6: Creación del proyecto backend
Elaborado por: Investigador

Una vez seleccionado el tipo de proyecto, se asignará un nombre del proyecto en este caso se denomina ApiAgua Potable.

Figura 3.7: Nombre del proyecto
Elaborado por: Investigador

En la siguiente pantalla se muestra las versiones del framework disponibles, en este caso .NET Core 3.1 como se muestra en ls Figura 3.8

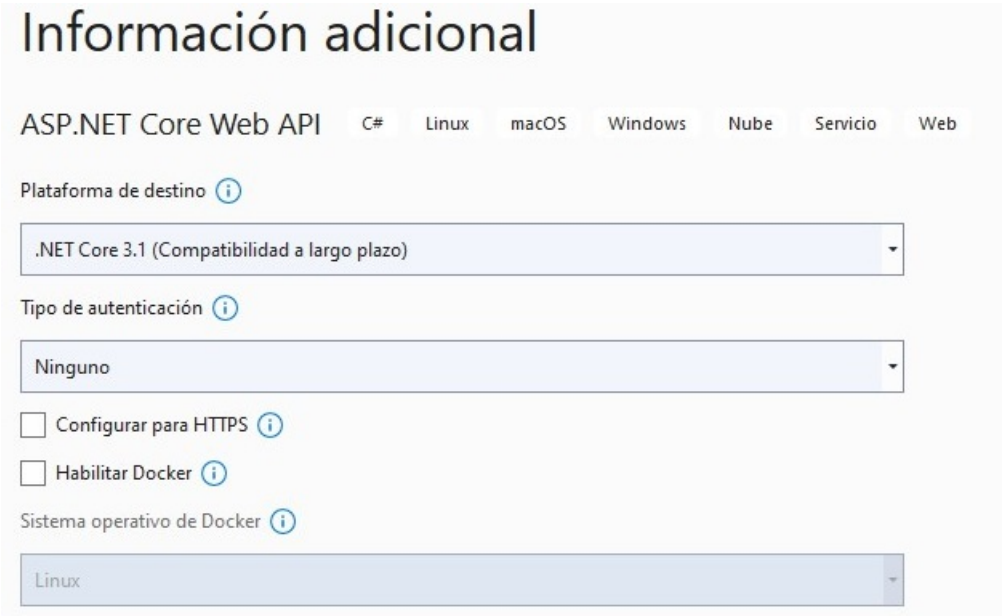


Figura 3.8: Selección de Framework
Elaborado por: Investigador

Organización de directorios del proyecto

Creado el proyecto basado en ASP .NET Core, se muestra un directorio predeterminado Controllers, sin embargo, se agregarán nuevos directorios según sea necesario con el objetivo de mantener una estructura escalable para futuras actualizaciones.

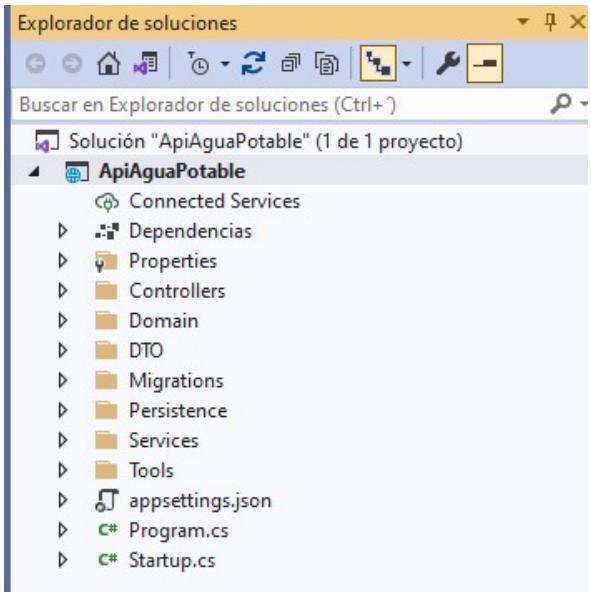


Figura 3.9: Estructura del proyecto backend
Elaborado por: Investigador

Luego se procede con la instalación entity framework, desde el administrador de paquetes, como se muestra en la figura 3.10.

- **EntityFrameworkCore:** Es el framework que se ha seleccionado para trabajar en el proyecto.
- **EntityFrameworkCore.SqlServer:** Permite la conexión con Sql Server.
- **EntityFrameworkCore.Tools:** Herramienta de consola en visual studio para la ejecución de comandos.

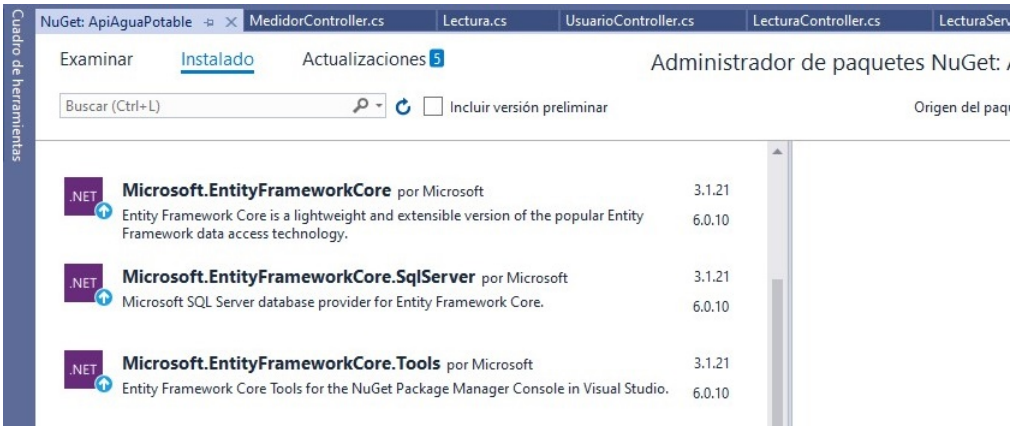


Figura 3.10: Instalación de Entity Framework
Elaborado por: Investigador

Frontend

Para crear el proyecto frontend, se lo realizará desde consola del sistema ingresando el comando ng new nombreProyecto como se muestra en la Figura 3.10. Luego se muestra la opción de agregar el módulo angular routing, y el estilo de hojas que se desea ocupar en el proyecto.

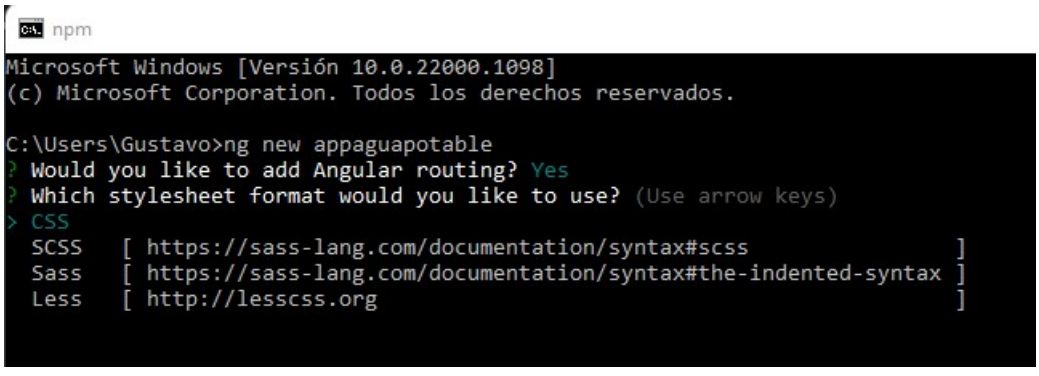


Figura 3.11: Creación del proyecto frontend
Elaborado por: Investigador

Posterior a la creación del proyecto frontend, se procede ingresar en la carpeta creado e ingresar con el editor de código visual studio code. Allí se ingresa una nueva terminal para levantar el servidor de angular y verificar que compile correctamente.

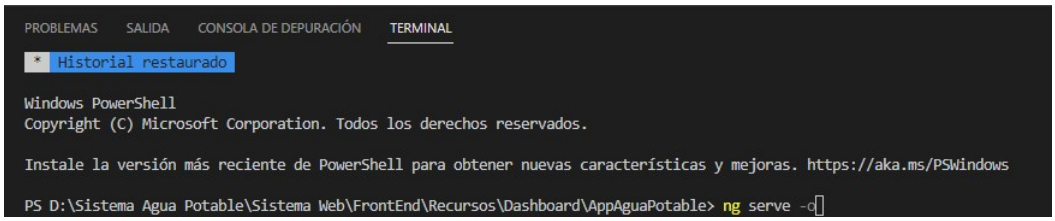


Figura 3.12: Levantar servidor de Angular
Elaborado por: Investigador

Organización de directorios del proyecto

El proyecto creado basado en framework Angular, esta definido por una estructura básica, pero se agregaron nuevos directorios con el fin de mantener una organización escalable según sea conveniente.

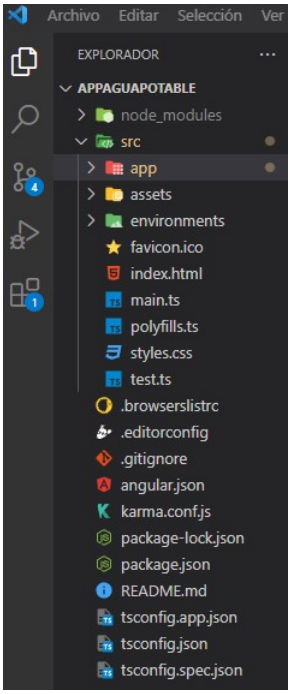


Figura 3.13: Estructura del proyecto frontend
Elaborado por: Investigador

Diseño de la Base de datos

En la figura 3.14, se puede evidencia el diseño de la base de datos relacional, en el cual se encuentran las tuplas necesarias de cada tabla. Con las relaciones o claves foráneas se puede minimizar el ingreso de datos erróneos en la base, y agilizar las consultas. Cabe mencionar que la empresa cuenta con licenciamiento de SQL Server.

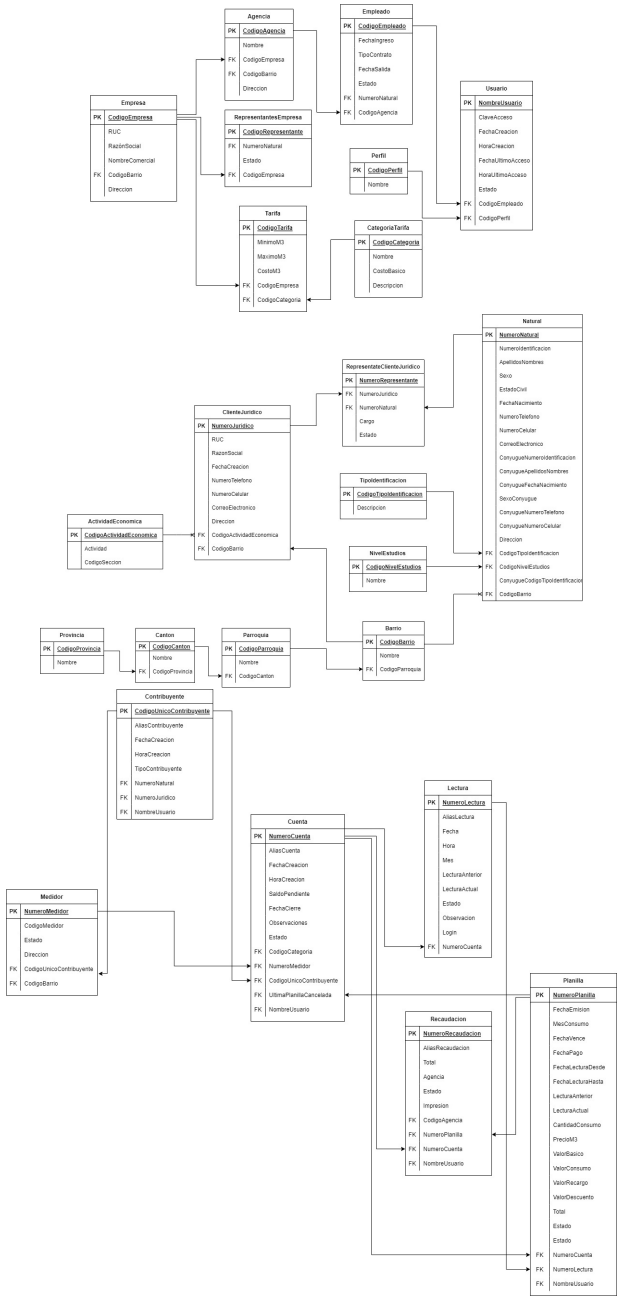


Figura 3.14: Diseño Base de Datos
Elaborado por: Investigador

Inicio de Sesión

Para acceder a la aplicación web, se lo realiza mediante el ingreso de credenciales como nombre de usuario y contraseña registrados en la base de datos, cabe mencionar que en caso de ingreso incorrecto de credenciales no se le permite el acceso al mismo.

Figura 3.15: Inicio de sesión
Elaborado por: Investigador

Pantalla principal

En la pantalla principal, se refleja los datos del usuario que accedió, además que se muestra el menú lateral, con los módulos respectivos su navegación.

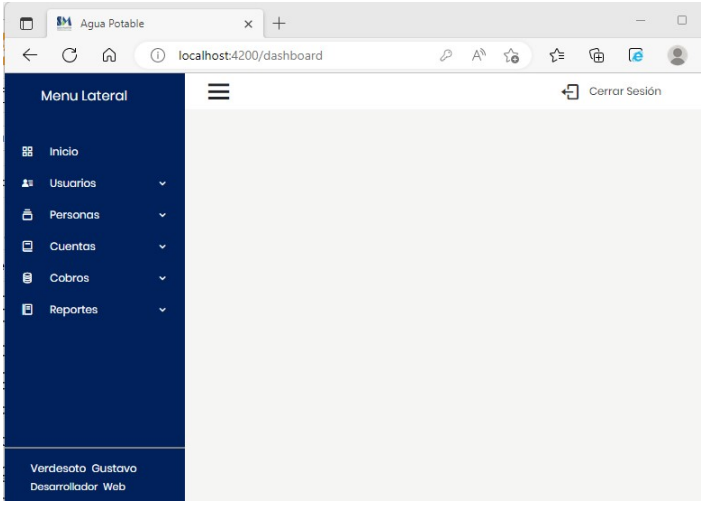


Figura 3.16: Pantalla principal
Elaborado por: Investigador

Iteración II

Tabla 3.68: Estimación de la segunda iteración

Elaborado por: Investigador

Iteración	**N°**	**Nombre Historia**	**Tiempo estimado**	
			Días	**Horas**
Segunda	05	Registrar un nuevo usuario	4	20
	06	Cambiar contraseña	3	15
	07	Activar y cancelar usuarios	2	10
	08	Añadir nueva persona natural	5	25
	09	Listado de personas naturales	3	15
	10	Editar datos de una persona natural	4	20
		Total	**21**	**105**

Registrar un nuevo usuario

En la figura 3.17, se evidencia la pantalla de registro un nuevo usuario, luego que todos los campos hayan sido ingresados de manera correcta, se habilitará el botón Guardar, cabe mencionar que esta acción la puede realizar el administrador de la aplicación web.

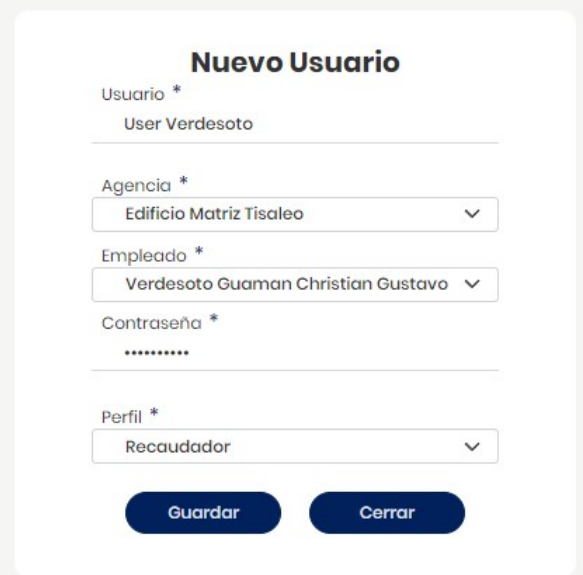


Figura 3.17: Registrar un nuevo usuario
Elaborado por: Investigador

Cambiar contraseña

En la figura 3.18, se refleja la pantalla de cambio de contraseña, se necesita ingresar la contraseña actual y nueva, la misma que deberá cumplir requisitos básicos de seguridad, como mínimo 8 caracteres incluidos números.

Figura 3.18: Cambiar contraseña
Elaborado por: Investigador

Activar y Cancelar usuarios

En la figura 3.19, se observa el listado de usuarios registrados, el administrador luego de registrar el usuario, tiene la tarea de activar el usuario para su respectivo acceso, cabe mencionar que los usuarios cancelados no se les permite el ingreso a la aplicación.

Figura 3.19: Activar y cancelar usuarios
Elaborado por: Investigador

Añadir nueva persona natural

En la figura 3.20, se muestra la pantalla para ingresar una nueva persona natural, se han definido campos obligatorios y opcionales por ejemplo datos de contacto o conyugue en caso de aplicar, para el ingreso de datos del conyugue se habilitará el submenú Conyugue en función del estado civil, además se refleja combos de selección para provincia, cantón, parroquia y barrio de manera dinámica.

Una vez que se han ingresado los datos de manera correcta, se habilitará la opción de guardar, al presionar se muestra un modal con el cuadro de confirmación para proceder al registro en la base de datos.

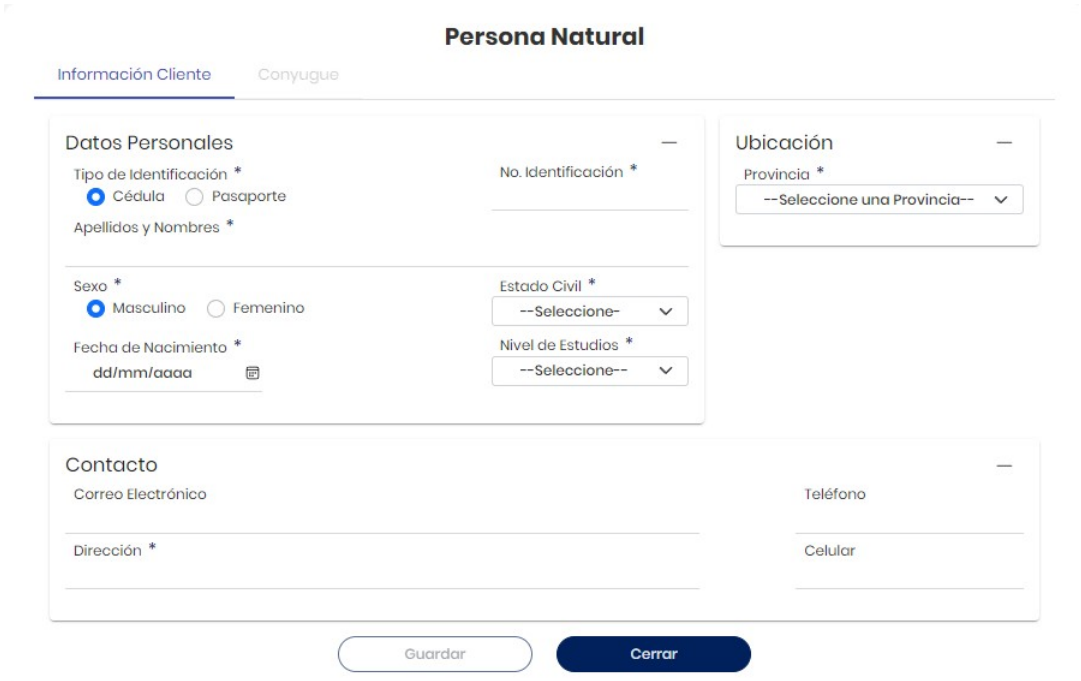


Figura 3.20: Añadir nueva persona natural
Elaborado por: Investigador

Listado de personas naturales

En la figura 3.21, se muestra el listado de las personas naturales que han sido registradas con éxito, se refleja un cuadro de búsqueda, que le permite al usuario filtrar datos por número de cédula o nombres.

En caso de editar los datos de una persona natural, es necesario seleccionar la opción editar del registro que se muestra.

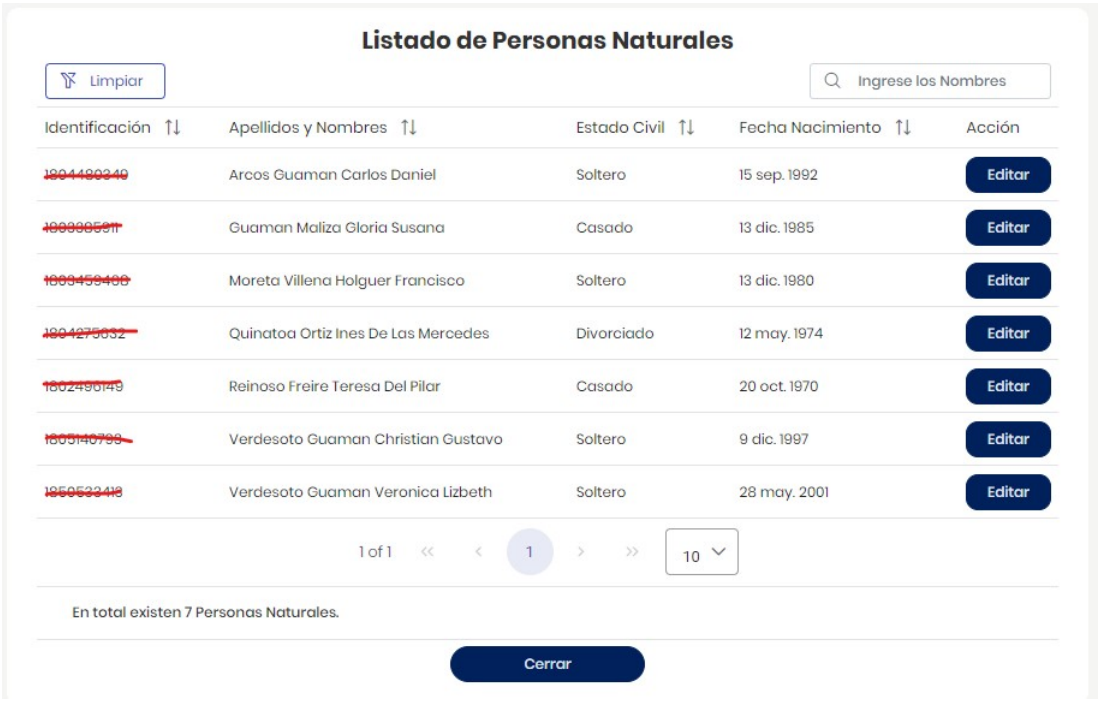


Figura 3.21: Listado de personas naturales
Elaborado por: Investigador

Figura 3.22: Búsqueda por nombres
Elaborado por: Investigador

Figura 3.23: Búsqueda por número de cédula
Elaborado por: Investigador

Editar datos de una persona natural

En la figura 3.24, se evidencia la pantalla de edición de datos de una persona natural, que se ha seleccionado previamente del listado. Se precargarán los datos registrados en la base de datos, para modificar debe llenar los campos de manera correcta.

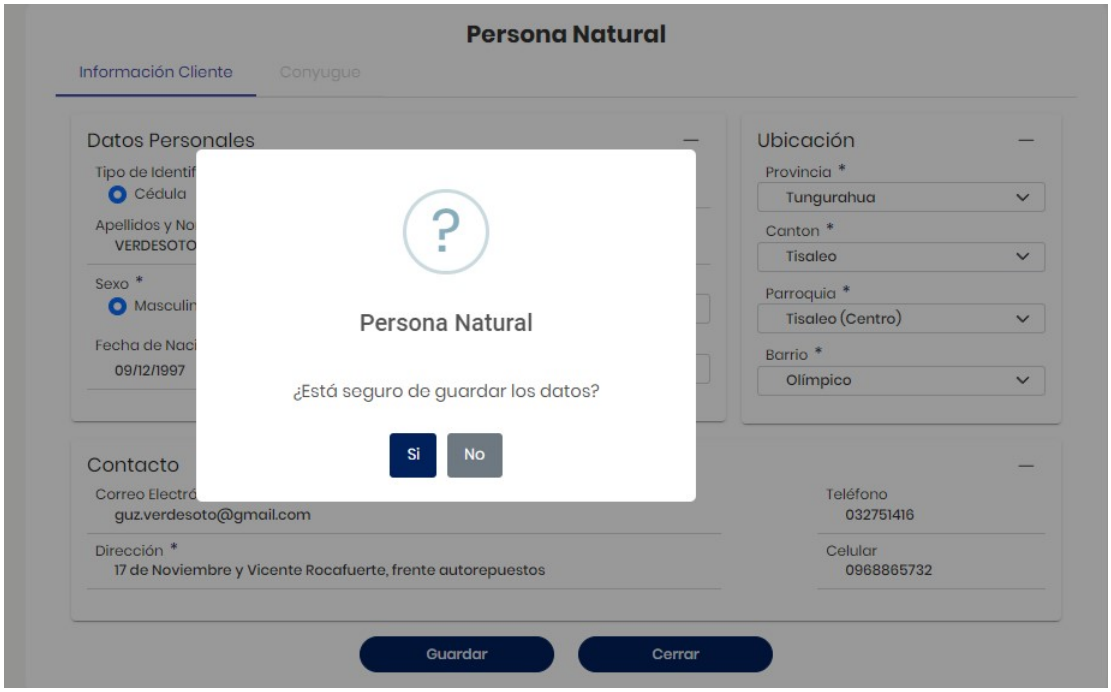


Figura 3.24: Editar datos de una persona natural
Elaborado por: Investigador

Iteración III

Tabla 3.69: Estimación de la tercera iteración

Elaborado por: Investigador

Iteración	N°	Nombre Historia	Tiempo estimado	
			Días	Horas
Tercera	11	Añadir nueva persona jurídica	4	20
	12	Añadir representantes a una persona jurídica	6	30
	13	Listado de personas jurídicas	3	15
	14	Editar datos de una persona jurídica	4	20
	15	Eliminar representante de una persona jurídica	3	15
		Total	**20**	**100**

Añadir nueva persona jurídica

En la figura 3.25, se muestra la pantalla para ingresar una nueva persona jurídica, se han definido campos obligatorios y opcionales como datos de contacto, se muestra el panel de ubicación para seleccionar provincia, cantón, parroquia y barrio de manera dinámica.

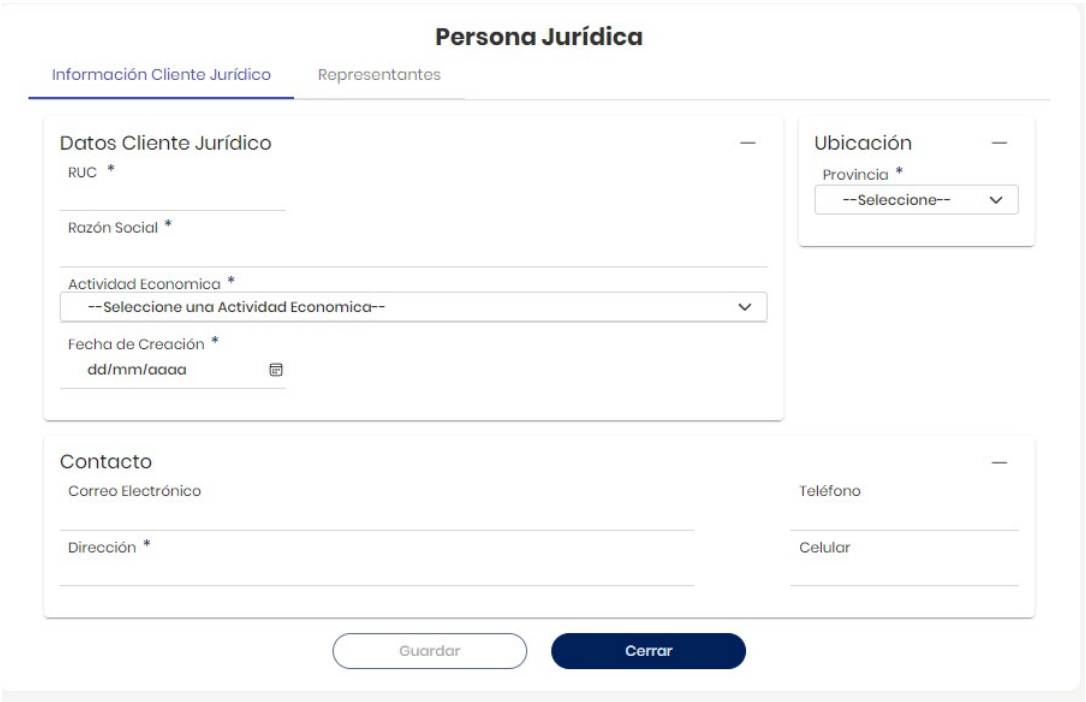


Figura 3.25: Añadir nueva persona jurídica
Elaborado por: Investigador

Añadir representantes a una persona jurídica

En la figura 3.26, se refleja el panel para agregar representantes a una persona jurídica, el usuario deberá ingresar el número de identificación, seleccionar el cargo que ocupa y la aplicación realizará una búsqueda en la base de datos, mostrando en pantalla los datos de la persona que pertenece el número de identificación ingresado. El mismo paso se repetirá para los demás representantes, cabe mencionar que mínimo se necesitan dos representantes, sin embargo, el usuario puede ingresar el número de representantes que considere necesario.

Una vez seleccionado los representantes con sus respectivos cargos o designaciones, se habilitará el botón guardar, el mismo que al presionarlo muestra un modal con el cuadro de confirmación para proceder o no al registro en la base de datos.

Figura 3.26: Añadir representantes a una persona jurídica
Elaborado por: Investigador

Listado de personas jurídicas

En la figura 3.27, se observa el listado de personas jurídicas que han sido registradas con éxito, se muestra un cuadro de búsqueda, que le permite al usuario filtrar datos por RUC o razón social.

Para editar los datos de una persona jurídica, es necesario seleccionar la opción editar, del registro que se desea actualizar datos.

Figura 3.27: Listado de personas jurídicas
Elaborado por: Investigador

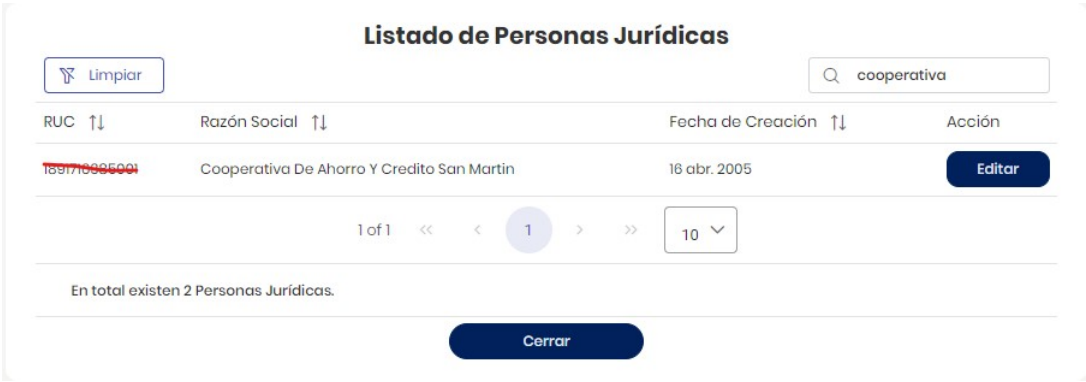


Figura 3.28: Búsqueda por razón social
Elaborado por: Investigador

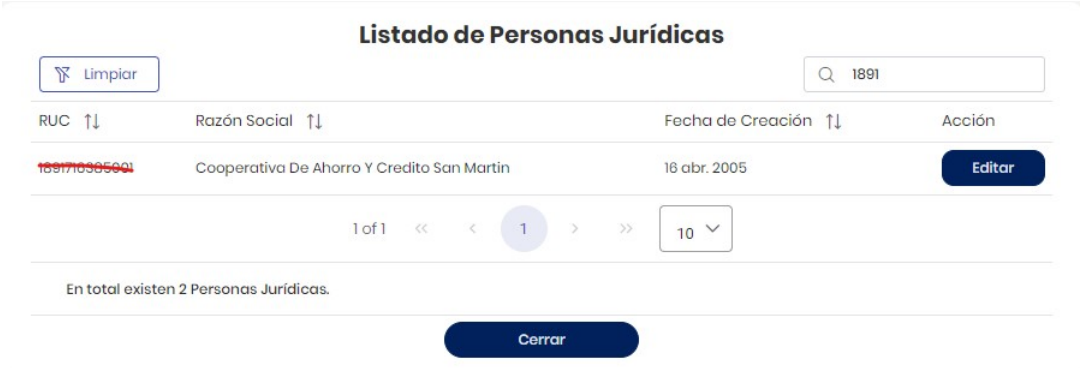


Figura 3.29: Búsqueda por ruc
Elaborado por: Investigador

Editar datos de una persona jurídica

En la figura 3.30, se evidencia la pantalla de edición de datos de una persona jurídica, que se ha seleccionado previamente del listado. Se pre-cargarán los datos registrados en la base de datos, para modificar se debe llenar los campos de manera correcta

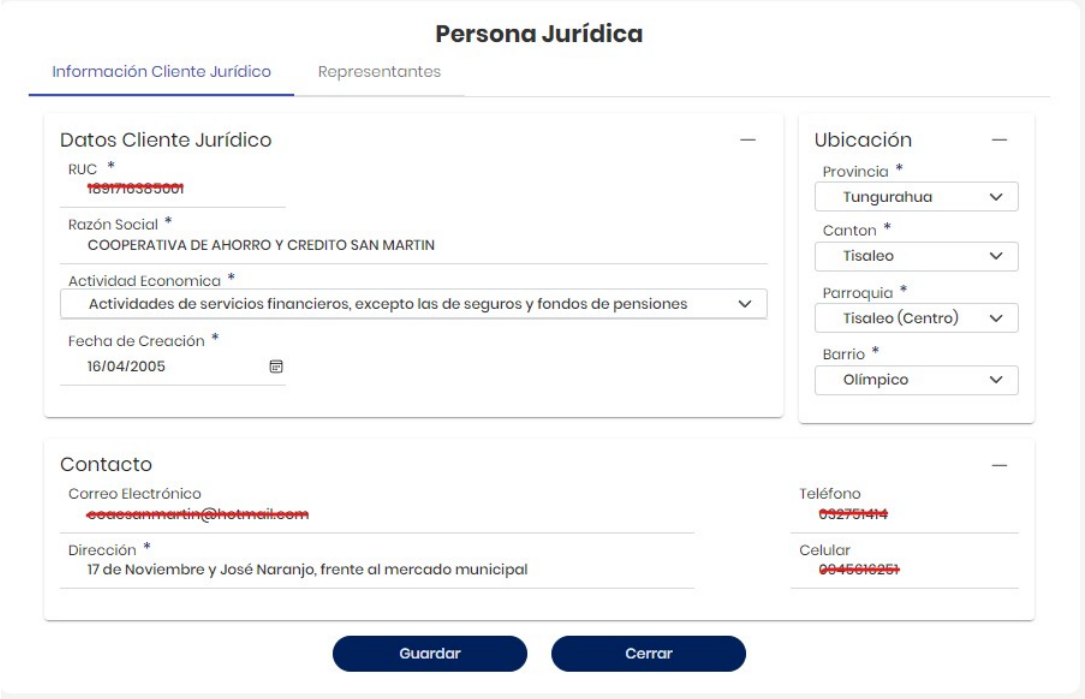


Figura 3.30: Editar datos de una persona jurídica
Elaborado por: Investigador

Eliminar representante de una persona jurídica

En la figura 3.31, se muestra el menú de representantes de una persona jurídica, el usuario puede eliminar uno o más representantes presionando el ícono de basurero que se muestra en la parte superior derecha del panel de cada representante. Cabe mencionar, que cada persona jurídica necesita mínimo dos representantes, por lo tanto, luego de eliminar los representantes el usuario deberá cumplir con este requerimiento.

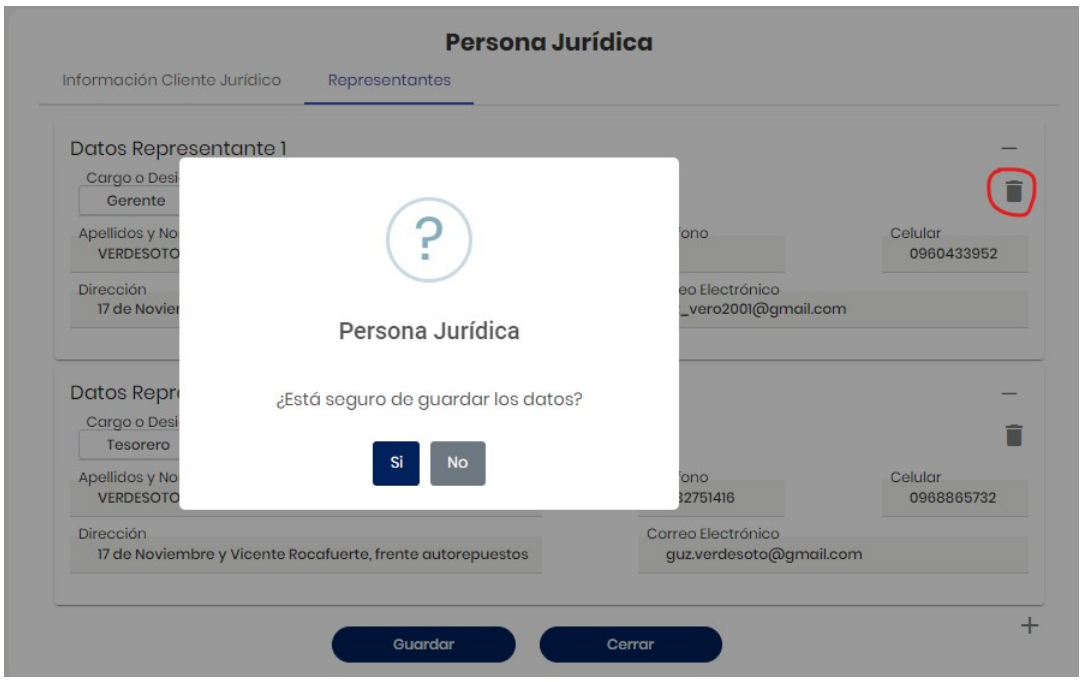


Figura 3.31: Eliminar representante de una persona jurídica
Elaborado por: Investigador

Iteración IV

Tabla 3.70: Estimación de la cuarta iteración

Elaborado por: Investigador

Iteración	**N°**	**Nombre Historia**	**Tiempo estimado**	
			Días	**Horas**
Cuarta	16	Registrar una nueva cuenta	4	20
	17	Listado de cuentas	3	15
	18	Activar y cancelar cuentas	2	10
	19	Registrar una nueva lectura	4	20
	20	Listado de lecturas	3	15
	21	Cobro de planillas	6	30
		Total	**22**	**110**

Registrar una nueva cuenta

En la figura 3.32, se evidencia la pantalla para el registro de una cuenta, para contribuyentes naturales y jurídicos, para este caso se han definido todos los campos de carácter obligatorio, el usuario deberá consultar los datos del contribuyente mediante el número de identificación o ruc, posterior a ello se le asignará la tarifa y medidor que le corresponde, además se requiere la ubicación específica del medidor.

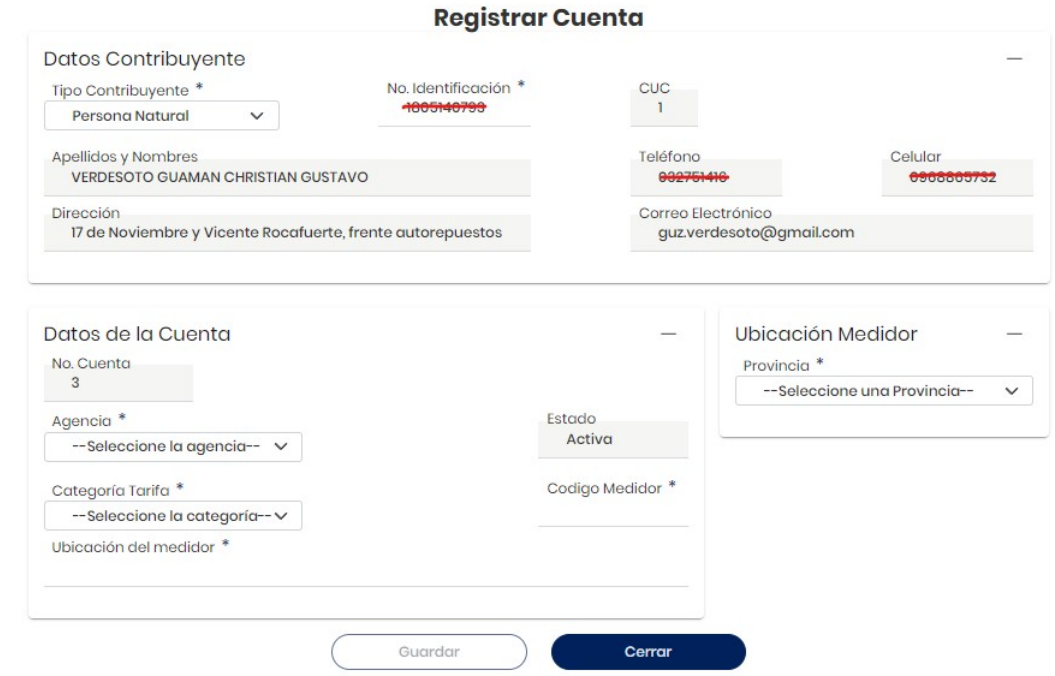


Figura 3.32: Registrar nueva cuenta
Elaborado por: Investigador

Una vez que los campos requeridos se encuentren ingresados de manera correcta, el botón guardar se habilitará para continuar con el registro de la nueva cuenta en la base de datos.

Listado de cuentas

En la figura 3.33, se refleja la pantalla del listado de cuentas que se encuentran registradas en la base de datos, en la parte superior derecha se encuentra un cuadro de búsqueda que permite aplicar varios filtros como, número de cuenta, estado, contribuyente, código o serie del medidor.

Figura 3.33: Listado de cuentas
Elaborado por: Investigador

Figura 3.34: Búsqueda por número de cuenta
Elaborado por: Investigador

Figura 3.35: Búsqueda por contribuyente
Elaborado por: Investigador

Figura 3.36: Búsqueda por código de medidor
Elaborado por: Investigador

Activar y cancelar cuentas

En la figura 3.37, se muestra la pantalla de Activar y cancelar una cuenta registrada, el administrador tendrá privilegios de acceso a esta pantalla, cabe mencionar que las cuentas canceladas, no se permite el acceso a los datos de dicha cuenta, a excepción del administrador, por ende tampoco se permite el registro de lecturas posteriores a la cancelación.

Figura 3.37: Activar y cancelar cuentas
Elaborado por: Investigador

Registrar nueva lectura

En la figura 3.38, se evidencia la pantalla para el registro de una lectura inicial y lectura nueva de una cuenta, para este caso se han definido campos obligatorios, como numero de cuenta y lectura actual, una vez que el usuario ingrese el número de cuenta, se desplegaran los datos del contribuyente al que pertenece la cuenta ingresada.

Se puede ingresar una lectura inicial, esto aplica para todas las cuentas que se encuentren registradas, cabe mencionar que solo puede existir una lectura inicial por cada cuenta, para esto el usuario deberá seleccionar la opción Si en el campo lectura inicial y digitar la cantidad de lectura actual.

Figura 3.38: Registrar lectura inicial
Elaborado por: Investigador

En el caso que exista una lectura inicial registrada, se procede con el registro de una nueva lectura, para ello el usuario deberá seleccionar la opción “No” en el campo lectura inicial, e inmediatamente se desplegara el valor de la lectura anterior, teniendo en cuenta que el valor de la lectura actual deberá ser mayor. Una vez que todos los campos ingresados sean correctos se habilita el botón Guardar para proceder con el registro de la lectura.

Figura 3.39: Registrar nueva lectura
Elaborado por: Investigador

Listado de lecturas

En la figura 3.40, se observa la pantalla del listado de lecturas registradas en la base de datos, en la parte superior de la pantalla se encuentran dos selectores de fechas los mismos que permiten aplicar el filtro de búsqueda entre fechas, es decir que el usuario puede observar el listado de lecturas que se han registrado dentro de un período de tiempo seleccionado.

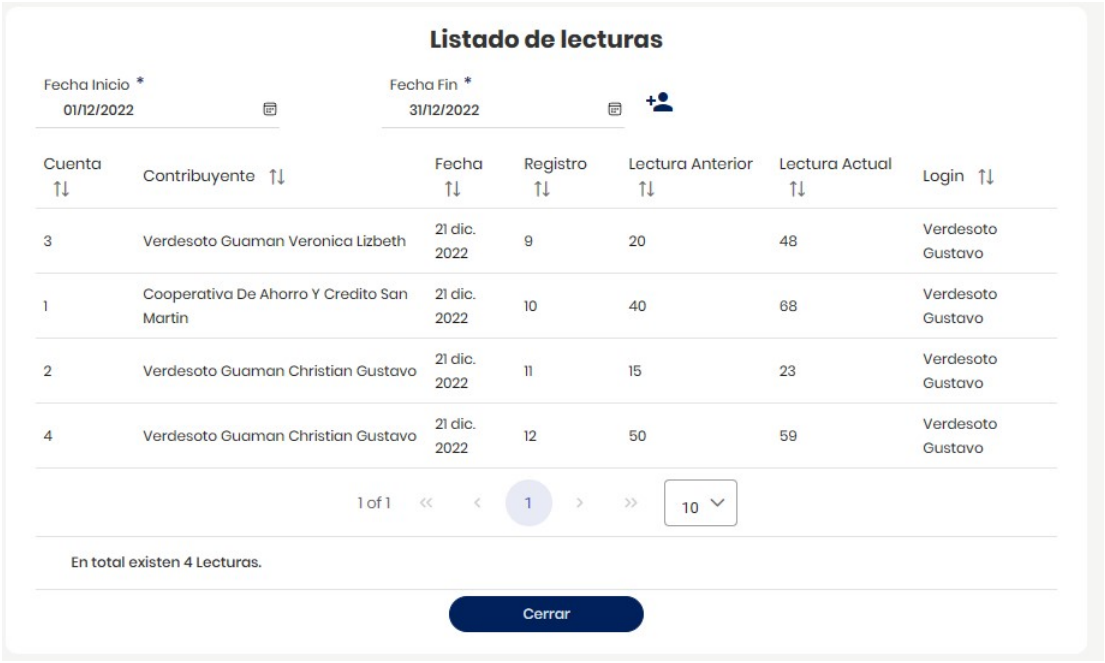


Figura 3.40: Listado de lecturas por fechas
Elaborado por: Investigador

Cobro de planillas

En la figura 3.41, refleja la pantalla para el cobro de planillas mensuales, para ello se ha definido como campo obligatorio el número de cuenta, el mismo que deberá ser ingresado por el usuario para verificar los datos del contribuyente.

Luego de verificar que los datos del contribuyente sean correctos, se desplegará un listado de planillas pendientes de pago asociadas a esa cuenta, el usuario deberá seleccionar la planilla a ser recaudada.

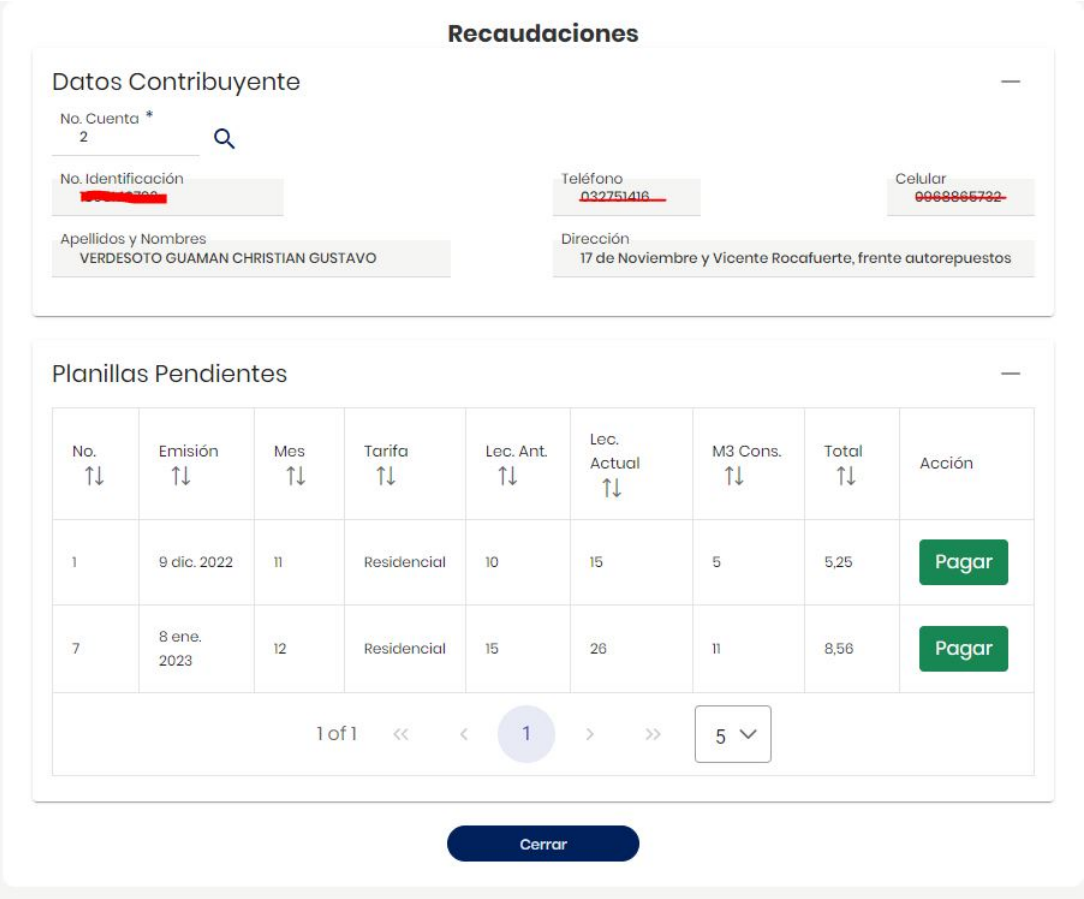


Figura 3.41: Recaudaciones por Cuenta
Elaborado por: Investigador

Iteración V

Tabla 3.71: Estimación de la quinta iteración

Elaborado por: Investigador

Iteración	**N°**	**Nombre Historia**	**Tiempo estimado**	
			Días	**Horas**
Quinta	22	Listado de planillas	3	15
	23	Registrar cierre de caja	4	20
	24	Listado de cierres de caja	3	15
	25	Listado de recaudaciones	3	15
		Total	**22**	**110**

Listado de planillas

En la figura 3.42, se muestra la pantalla de todas las planillas que han sido generadas por la aplicación, en la parte superior derecha se ubica un cuadro

de búsqueda que permite aplicar varios filtros como, nombres o razón social del contribuyente, número de cuenta, estado de la planilla sea este cancelado o esté pendiente de pago.

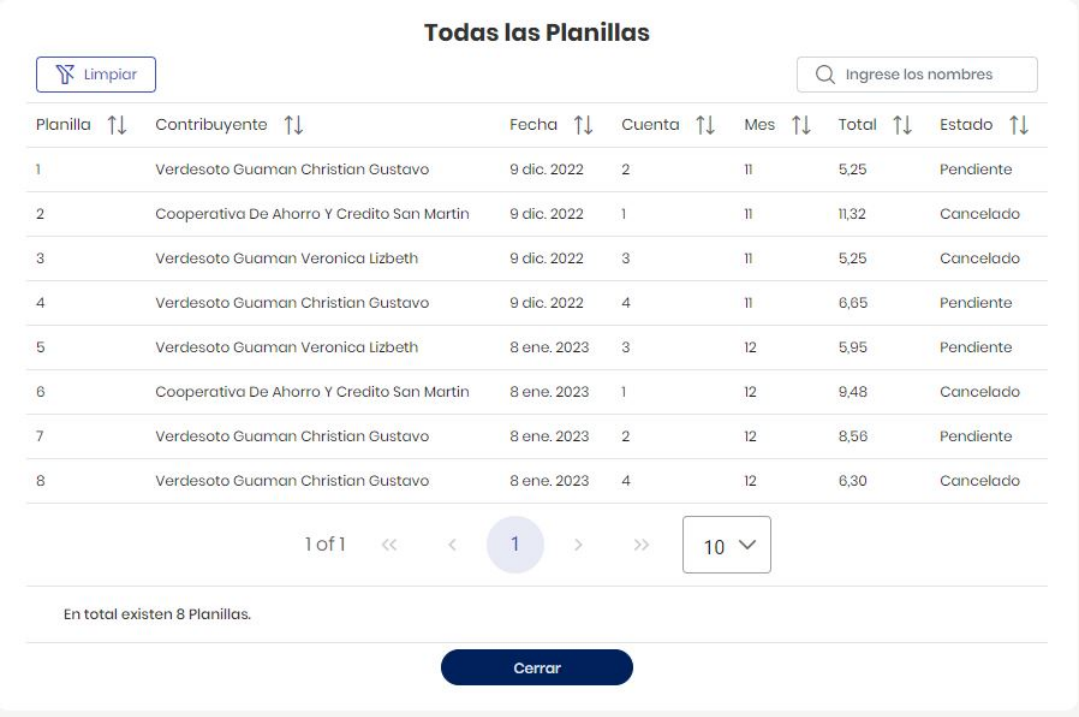


Figura 3.42: Todas las planillas
Elaborado por: Investigador

Figura 3.43: Búsqueda por nombres o razón social del contribuyente
Elaborado por: Investigador

Figura 3.44: Búsqueda por número de cuenta
Elaborado por: Investigador

Figura 3.45: Búsqueda por estado de la planilla
Elaborado por: Investigador

Registrar cierre de caja

En la figura 3.46, se refleja la pantalla para el registro del cierre de caja, en la parte superior se muestran todas las recaudaciones realizadas por un usuario, mientras en la parte inferior izquierda se ubica el panel de las transacciones no registradas por la aplicación, es decir que pueden existir entradas y salidas de dinero, por conceptos de vales de ingreso, recaudaciones y retenciones, mientas al salidas por concepto de vales de egreso y pago de facturas. Cabe mencionar que estos rubros pueden estar o no en cero, dependiendo del flujo de dinero que exista en la caja.

En el panel Detalle Físico, permite al usuario digitar la cantidad de billetes y monedas que tenga en su poder al momento de realizar el cierre, una vez ingresada la cantidad la aplicación realiza el cálculo y devuelve el valor en dólares para las diferentes denominaciones. Al finalizar este proceso, nos muestra el valor total en billetes y monedas, las mismas que al sumar darán como resultado el valor del saldo físico.

Para registrar el cierre de caja, el saldo físico debe ser igual al saldo contable, caso contrario el usuario deberá repetir el proceso hasta encontrar el error del descuadre. Solo así se habilitará el botón guardar.

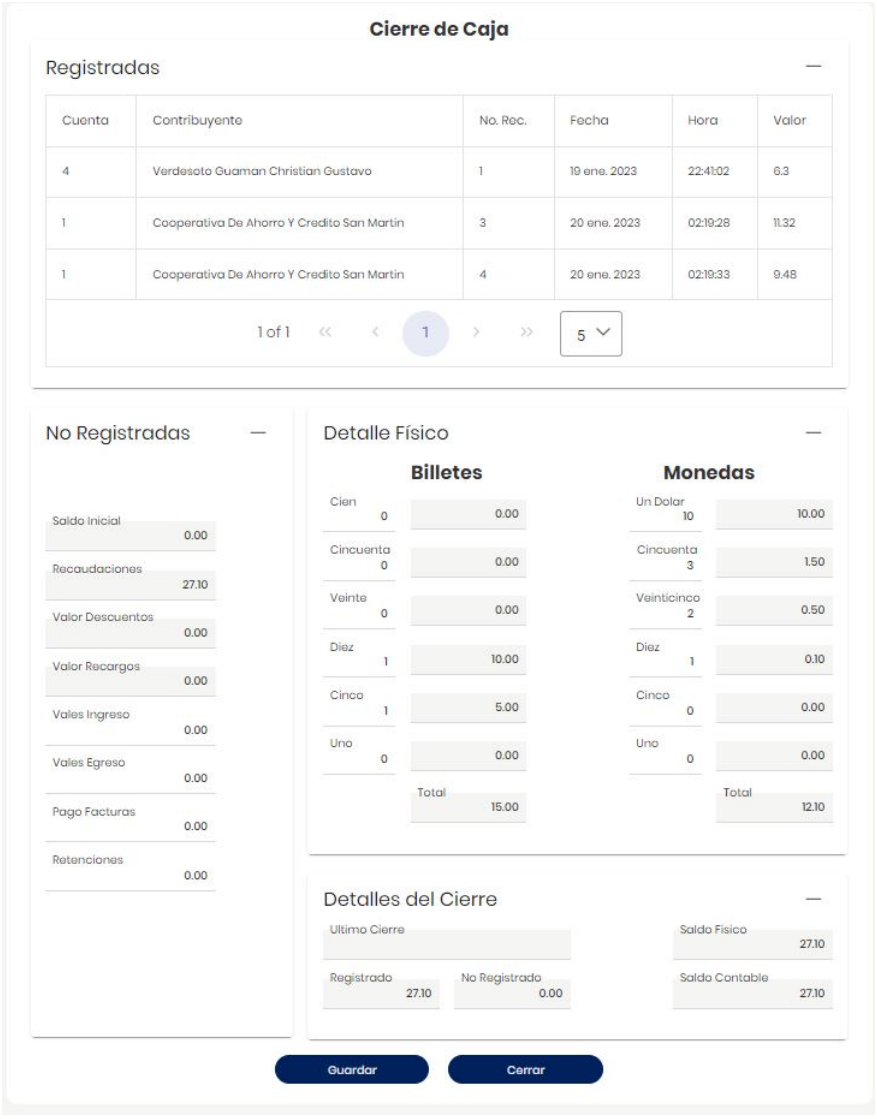


Figura 3.46: Registrar cierre de caja
Elaborado por: Investigador

Listado de cierres de caja

En la figura 3.47, se muestra la pantalla de todos los cierres de caja registrados en la base de datos, en la parte superior izquierda se encuentra los filtros de fecha, es decir que la aplicación me permite buscar los cierres de caja dentro de un tiempo seleccionado.

Figura 3.47: Todos los cierres de caja
Elaborado por: Investigador

Listado de recaudaciones

En la figura 3.48, se refleja la pantalla de todas las recaudaciones registradas por los usuarios, en la parte superior izquierda se puede observar dos selectores de fechas, los cuales permiten aplicar el filtro de búsqueda de todas las recaudaciones en el tiempo seleccionado.

Figura 3.48: Todas las recaudaciones
Elaborado por: Investigador

3.2.3. Fase III. Codificación

3.2.3.1. Métodos de Aplicación Backend

Autenticación de usuarios con JWT para API-REST

Para proporcionar un nivel de seguridad a las rutas o endpoints de la API, se implemento la tecnología Jason Web Token, que permite la transmisión de información segura entre las partes como un objeto JSON. De tal manera que para el acceso a las rutas se necesita autenticar las solicitudes.

Una vez que el usuario ingresa sus credenciales, es decir usuario y contraseña, la api valida sus credenciales y devuelve un token, el mismo que será incluido con cada solicitud que realice a la ruta.

El método "ObtenerToken" genera un token de acceso que contiene una referencia al id de usuario que ingresa sus credenciales, para obtener el acceso a las rutas protegidas.

```
//Recibimos como parametro un usuario
public static string ObtenerToken(Usuario datosUsuario, IConfiguration config)
{
    //accedemos a los parametros del Token que definimos en appsettings.json
    string ClaveSecreta = config["Jwt:ClaveSecreta"];
    string Issuer = config["Jwt:Issuer"];
    string Audiencia = config["Jwt:Audiencia"];

    /*HEADER
     * Codigo Estandar
     * Firmamos las credenciales, recibe la claveSecrtea y el algoritmo
     */
    var securityKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(ClaveSecreta));
    var credentials = new SigningCredentials(securityKey, SecurityAlgorithms.HmacSha256);

    /*Reclamaciones
     Registramos con SUb. que permite pasar el nombre de usuario
     */
    var claims = new[]
    {
        new Claim(JwtRegisteredClaimNames.Sub, datosUsuario.NombreUsuario),
        new Claim("NombreUsuario", datosUsuario.NombreUsuario)
    };

    //Generamos el Token
    var token = new JwtSecurityToken(
        issuer: Issuer,
        audience: Audiencia,
        claims,
        expires: DateTime.Now.AddMinutes(10),
        signingCredentials: credentials
        );

    return new JwtSecurityTokenHandler().WriteToken(token);
}
```

Figura 3.49: Método ObtenerToken
Elaborado por: Investigador

El método “obtenerTokenUsuario”, puede obtener el nombre de usuario autenticado mediante el acceso a los claims de JWT creado anteriormente.

```
//Definimos un metodo que mediante los Claims nos devuelva el nombreUsuario(id)
public static string ObtenerTokenUsuario(ClaimsIdentity identificador)
{
    if (identificador !=null)
    {
        IEnumerable<Claim> claims = identificador.Claims;
        foreach (var claim in claims)
        {
            if (claim.Type== "NombreUsuario")
            {
                return claim.Value;
            }
        }
    }
    return "";
}
```

Figura 3.50: Método “ObtenerTokenUsuario”
Elaborado por: Investigador

ASP.NET Core implementa el patrón de diseño MVC, que separa la lógica en tres capas: Modelo, Vista, Controlador. En este caso se aplica una arquitectura escalable y fácil de mantener como lo es repositoryPattern para minimizar el duplicado de código en la capa de acceso a datos.

Modelo

El modelo “Usuario”, permite definir la estructura y atributos de la tabla para su posterior creación en la base de datos.

```
namespace ApiAguaPotable.Domain.Models
{
    public class Usuario
    {
        [Key]
        [Column (TypeName ="varchar(50)")]
        public string NombreUsuario { get; set; }


        //Relacion a la tabla Empelado
        [ForeignKey ("Empleado")]
        public string CodigoEmpleado { get; set; }
        public Empleado Empleado { get; set; }

        [Column (TypeName ="varchar(300)")]
        public string ClaveAcceso { get; set; }


        [Column(TypeName = "date")]
        public DateTime FechaCreacion { get; set; }

        [Column(TypeName = "time(7)")]
        public TimeSpan HoraCreacion { get; set; }

        [Column(TypeName = "date")]
        public DateTime FechaUltimoAcceso { get; set; }

        [Column(TypeName = "time(7)")]
        public TimeSpan HoraUltimoAcceso { get; set; }


        //Relacion a la tabla Perfil
        [ForeignKey ("Perfil")]
        public string CodigoPerfil { get; set; }
        public Perfil Perfil { get; set; }

        [Column(TypeName = "varchar(20)")]
        public string Estado { get; set; }
    }
}
```

Figura 3.51: Model Usuario
Elaborado por: Investigador

Contexto

Luego de crear el model usuario se procede a dar origen al contexto de la api, esta clase mapea los elementos del modelo creado, por lo tanto permite consultar, crear, eliminar y editar registros en una base de datos.

```
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace ApiAguaPotable.Persistence.Context
{
    public class AplicationDBContext: DbContext
    {
        //Mapeo los elementos que voy a crear en la base

        //Usuario
        public DbSet<Usuario> Usuario { get; set; }

        Models Store Procedure
        public AplicationDBContext(DbContextOptions<AplicationDBContext> options):base(options)
        {
        }

        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            modelBuilder.Entity<TodosLosEmpleadosPorAgencia>().HasNoKey();
        }
    }
}
```

Figura 3.52: AplicationDBContext
Elaborado por: Investigador

La base de datos será alojada en Sql Server en su versión Estándar, para ello se procede agregar la cadena de conexión en el archivo "appsettings.json". Teniendo en cuenta que la Base de datos debe ser creada previamente.

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft": "Warning",
      "Microsoft.Hosting.Lifetime": "Information"
    }
  },
  "ConnectionStrings": {
    "Conexion": "Data Source=DESKTOP-3FF262I\\SQLEXPRESS;Initial Catalog=AguaPotable;Integrated Security=True"
  },
  "AllowedHosts": "*",
```

Figura 3.53: Cadena de conexion
Elaborado por: Investigador

Migración a Sql Server

Las migraciones guardan el historial de las tablas creadas en la base de datos, para esto se debe ingresar por consola el comando "Add-Migration" nombreMigracion, esto dará como resultado código para creación de tabla con su respectiva estructura y sus atributos. Luego se debe ingresar el comando "Update-database" y se creará la tabla en la base de datos.

```
namespace ApiAguaPotable.Migrations
{
    public partial class v11 : Migration
    {
        protected override void Up(MigrationBuilder migrationBuilder)
        {
            migrationBuilder.CreateTable(
                name: "Usuario",
                columns: table => new
                {
                    NombreUsuario = table.Column<string>(type: "varchar(50)", nullable: false),
                    CodigoEmpleado = table.Column<string>(nullable: true),
                    ClaveAcceso = table.Column<string>(type: "varchar(50)", nullable: true),
                    FechaCreacion = table.Column<DateTime>(type: "date", nullable: false),
                    HoraCreacion = table.Column<TimeSpan>(type: "time(7)", nullable: false),
                    FechaUltimoAcceso = table.Column<DateTime>(type: "date", nullable: false),
                    HoraUltimoAcceso = table.Column<TimeSpan>(type: "time(7)", nullable: false),
                    CodigoPerfil = table.Column<string>(nullable: true),
                    Estado = table.Column<string>(type: "varchar(20)", nullable: true)
                },
                constraints: table =>
                {
                    table.PrimaryKey("PK_Usuario", x => x.NombreUsuario);
                    table.ForeignKey(
                        name: "FK_Usuario_Empleado_CodigoEmpleado",
                        column: x => x.CodigoEmpleado,
                        principalTable: "Empleado",
                        principalColumn: "CodigoEmpleado",
                        onDelete: ReferentialAction.Restrict);
                    table.ForeignKey(
                        name: "FK_Usuario_Perfil_CodigoPerfil",
                        column: x => x.CodigoPerfil,
                        principalTable: "Perfil",
                        principalColumn: "CodigoPerfil",
                        onDelete: ReferentialAction.Restrict);
                });
        }

        protected override void Down(MigrationBuilder migrationBuilder)
        {
            migrationBuilder.DropTable(
                name: "Usuario");
        }
```

Figura 3.54: Migración SQL Server
Elaborado por: Investigador

Por cada modelo creado, se debe mapear en el contexto y para su posterior migración, de tal manera que si todo es correcto se creará la tabla con su estructura y atributos en la base de datos, cabe destacar que la migración detecta los nuevos models creados.

Interfaz

La interfaz “ILoginRepository” permite definir las tareas que van a ser implementados en la capa de acceso a datos.

```
using ApiAguaPotable.Domain.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace ApiAguaPotable.Domain.IRepositories
{
    public interface ILoginRepository
    {
        Task<Usuario> ValidarUsuario(Usuario usuario);

        //Defino una tarea para obtener el perfil del usuario
        Task<Usuario> GetDatosUsuario(Usuario usuario);
    }
}
```

Figura 3.55: Interfaz ILoginRepository
Elaborado por: Ibvestigadort

La interfaz "ILoginService" permite definir las tareas que serán invocadas en los controllers que sean necesarios. Cabe mencionar que serán las mismas tareas que se definieron en la interfaz del repositorio.

```
using ApiAguaPotable.Domain.IRepositories;
using ApiAguaPotable.Domain.IServices;
using ApiAguaPotable.Domain.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace ApiAguaPotable.Services
{
    public class LoginService: ILoginService
    {
        private readonly ILoginRepository _loginRepositorio;
        public LoginService(ILoginRepository loginRepositorio)
        {
            _loginRepositorio = loginRepositorio;
        }

        public async Task<Usuario> ValidarUsuario(Usuario usuario)
        {
            //llamamos al metodo de la interfaz ILoginRepository
            return await _loginRepositorio.ValidarUsuario(usuario);
        }
    }
}
```

Figura 3.56: Interfaz ILoginService
Elaborado por: Investigador

Repositorio

La clase "LoginRepository", permite implementar las tareas que fueron definidas en la interfaz, mediante el contexto de la aplicación se puede acceder a los atributos del modelo, para la ejecución de sentencias sql.

```
namespace ApiAguaPotable.Persistence.Repositories
{
    public class LoginRepository:ILoginRepository
    {
        private readonly AplicationDBContext _contexto;
        public LoginRepository(AplicationDBContext contexto)
        {
            _contexto = contexto;
        }

        public async Task<Usuario> ValidarUsuario(Usuario usuario)
        {
            var datosUsuario = await _contexto.Usuario.Where(x =>
                                                              x.NombreUsuario == usuario.NombreUsuario
                                                               && x.ClaveAcceso == usuario.ClaveAcceso
                                                               && x.Estado=="Activo")
                                                      .Include(perfil => perfil.Perfil)
                                                      .Include(empleado => empleado.Empleado)
                                                      .ThenInclude(natural => natural.Natural)
                                                      .FirstOrDefaultAsync();

            return datosUsuario;
        }

        public async Task<Usuario> GetDatosUsuario(Usuario usuario)
        {
            //hago la consulta a la BD
            var perfilUsuario = await _contexto.Usuario.OrderBy(usuario => usuario.NombreUsuario)
                                                       .Include(perfil => perfil.Perfil)
                                                       .FirstOrDefaultAsync();

            return perfilUsuario;
        }
    }
}
```

Figura 3.57: Clase LoginRepository
Elaborado por: Investigador

Servicio

La clase “LoginService”, permite invocar las tareas que fueron definidas en la interfaz del repositorio, las mismas que han sido implementadas en la clase del repositorio, de esta manera se logra comunicar la capa de lógica del negocio con la capa de acceso a datos.

```
using ApiAguaPotable.Domain.IRepositories;
using ApiAguaPotable.Domain.IServices;
using ApiAguaPotable.Domain.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace ApiAguaPotable.Services
{
    public class LoginService: ILoginService
    {
        private readonly ILoginRepository _loginRepositorio;
        public LoginService(ILoginRepository loginRepositorio)
        {
            _loginRepositorio = loginRepositorio;
        }

        public async Task<Usuario> ValidarUsuario(Usuario usuario)
        {
            //llamamos al metodo de la interfaz ILoginRepository
            return await _loginRepositorio.ValidarUsuario(usuario);
        }
    }
}
```

Figura 3.58: Clase LoginService
Elaborado por: Investigador

Controlador

En el controlador "LoginController", contiene los endpoints o rutas de la api que responderán a una solicitud del frontend, estos pueden ser del tipo post (create), put (update), get (read) y delete

Login

El endpoint Post, recibe como parámetro un modelo de tipo Usuario, para validar las credenciales de acceso, en caso que sean correctas genera un token de acceso al sistema caso contrario emite un mensaje de error.

```
//Endpoit
[HttpPost]
public async Task<IActionResult> Post([FromBody]Usuario usuario)  //metodo que devuelve una tarea
{
    try
    {
        //Primero encriptamos la clave que recibimos desde el Front
        usuario.ClaveAcceso = Encriptar.EncriptarPasswordSHA256(usuario.ClaveAcceso);

        //Nos va a devolver un usuario siempre y cuando las credenciales sean correctas
        var datosUsuario =  await _loginServicio.ValidarUsuario(usuario);
        if (datosUsuario == null)
        {
            return BadRequest(new { message = "Usuario o contraseña incorrectos" });
        }
        else
        {
            //Almacenamos el Token del usuario en una cadena de texto
            string tokenCadena = ConfiguracionJWT.ObtenerToken(datosUsuario, _config);

            //Tomo la fecha y hora actual de la api (2)
            TimeSpan hora = System.TimeSpan.Parse(System.DateTime.Now.ToLongTimeString());
            DateTime fecha = System.DateTime.Now.Date;
            datosUsuario.FechaUltimoAcceso = fecha;
            datosUsuario.HoraUltimoAcceso = hora;

            //Esperamos que actualice la fecha de acceso del usuario
            await _usuarioServicio.ActualizarUltimoAcceso(datosUsuario);

            //devuelvo el token, y datos del usuario para el dashboard
            return Ok(new
            {
                token = tokenCadena,
                usuario = usuario.NombreUsuario,
                perfil = datosUsuario.Perfil.Nombre
            });
        }

    }
    catch (Exception ex)
    {
        //En caso de que ocurra un error lo campturamos y mostramos el mensaje
        return BadRequest(ex.Message + "\n" + ex.InnerException.Message);
    }
}
```

Figura 3.59: Login Controller
Elaborado por: Investigador

Nuevo Usuario

El endpoint post recibe como parámetro un json de tipo Usuario, para el nuevo registro en la base de datos, además valida que no se repitan registros, por temas de seguridad la contraseña se encripta en algoritmo sha256.

```
[HttpPost]
public async Task<IActionResult> Post([FromBody]Usuario usuario)
{
    /*(#)
     * Orden de creacion de codigo
     */
    try
    {
        //Valido si existe el usuario (2)
        var existeUsuario = await _usuarioServicio.ExisteUsuario(usuario);
        if (existeUsuario)   //en caso de ser true, mandamos un mensaje
            return BadRequest(new { message = "El usuario "+usuario.NombreUsuario+" ya existe.!!" });

        /*(3)
         * Llamo a la clase Encriptar, donde se encuentra el metodo (MD5 O SHA256)
         * La contraseña que recibo la encripto y la devuelvo encriptada
         */
        usuario.ClaveAcceso = Encriptar.EncriptarPasswordSHA256(usuario.ClaveAcceso);

        //Tomo la fecha y hora actual de la api (1)
        TimeSpan hora = System.TimeSpan.Parse(System.DateTime.Now.ToLongTimeString());
        DateTime fecha = System.DateTime.Now.Date;
        usuario.FechaCreacion = fecha;
        usuario.HoraCreacion = hora;
        usuario.FechaUltimoAcceso = fecha;
        usuario.HoraUltimoAcceso = hora;


        //llamamos al metodo que esta IUsuarioService(1)
        await _usuarioServicio.GuardarUsuario(usuario);

        //Si todo esta correcto, mostramos un mensaje de exito(1)
        return Ok(new { mensaje = "Usuario Registrado con éxito!" });
    }
    catch (Exception ex)
    {
        //En caso de que ocurra un error lo campturamos y mostramos el mensaje
        return BadRequest(ex.Message+"\n"+ ex.InnerException.Message);  //InnerException captura error
    }
}
```

Figura 3.60: Método registrar nuevo usuario
Elaborado por: Investigador

Activar o Cancelar Usuario

Para la actualización del estado de un usuario, se crea el endpoint put, de igual manera recibe un json de tipo usuario

```
[Route("ActualizarEstadoUsuario")]
[HttpPut]
public async Task<IActionResult> Put([FromBody] Usuario usuario)
{
    try
    {
        //Nos va a devolver los datos del usuario siempre y cuando exista en la BD
        var estadoUusarioActualizar = await _usuarioServicio.GetUsuarioPorNombreUsuario(usuario.NombreUsuario);
        if (estadoUusarioActualizar!=null)
        {
            //Actualizo el estado del usuario
            estadoUusarioActualizar.Estado = usuario.Estado;

            //llamamos al metodo que esta IUsuarioService
            await _usuarioServicio.ActualizarEstadoUsuario(estadoUusarioActualizar);

            //Si todo esta correco, mostramos un mensaje de exito
            return Ok(new { mensaje = "Los datos han sido actualizados correctamente..!" });
        }
        else
        {
            return BadRequest("No existen datos");
        }
    }
    catch (Exception ex)
    {
        //En caso de que ocurra un error lo campturamos y mostramos el mensaje
        return BadRequest(ex.Message + "\n" + ex.InnerException.Message);
    }
}
```

Figura 3.61: Método activar o cancelar un usuario
Elaborado por: Investigador

Lista de Usuarios

El endpoint get, retorna una lista de usuarios, para este caso no recibe ningún parámetro

```
[HttpGet]
public async Task<IActionResult> GetListaUsuarios()
{
    try
    {
        //almacenamos en una variable el listado de los usuarios
        var listaUsuarios = await _usuarioServicio.GetListaUsuarios();

        //En caso de exito devolvemos la lista de usuarios
        return Ok(listaUsuarios);
    }
    catch (Exception ex)
    {
        //En caso de que ocurra un error lo campturamos y mostramos el mensaje
        return BadRequest(ex.Message + "\n" + ex.InnerException.Message);  //InnerException
    }
}
```

Figura 3.62: Método lista de usuarios
Elaborado por: Investigador

Añadir Persona Natural

Para el registro de una persona natural, se crea el endpoint post que permite el registro en la tabla Persona y Contribuyente.

```
//Protegemos los endpoints -> Es decir que necesita autenticacion para ingresar
[Authorize(AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme)]  //(4) Esquema de autenticacion JWTBearer que instalamos
[HttpPost]
public async Task<IActionResult> Post([FromBody]Natural natural)
{
    try
    {
        #region Validar que no se duplique la cedula
        var cedulaDuplicada = await _naturalServicio.ExisteCedulaDuplicada(natural);
        if (cedulaDuplicada)
            return BadRequest(new { message = "El número de identificación "+
                              natural.NumeroIdentificacion+" ya existe..!" });
        #endregion

        #region Obtener el id del nuevo registro
        long numeroPersonaNatural;
        try{ numeroPersonaNatural = await _naturalServicio.UltimoNumeroPersonaNatural(natural); }
        catch { numeroPersonaNatural = 1; }
        #endregion

        for (int i = 0; i < natural.contribuyente.Count; i++)
        {
            #region Obtener el id del nuevo registro
            long codigoUnicoContribuyente;
            try{codigoUnicoContribuyente = await _contribuyenteServicio.UltimoCodigoUnicoContribuyente(natural.contribuyente[i]); }
            catch { codigoUnicoContribuyente = 1;  }
            #endregion

            #region Obtener la fecha y hora de la api
            TimeSpan hora = System.TimeSpan.Parse(System.DateTime.Now.ToLongTimeString());
            DateTime fecha = System.DateTime.Now.Date;
            string nombreUsuario,codigoEmpleado,idEmpresa, idTabla,idAlias="", aliasLongitud, alias;
            #endregion

            #region Obtener datos del usuario autenticado
            //JWT para obtener la clave primaria (Utilizamos claims para acceder al data del JWT)
            var identity = HttpContext.User.Identity as ClaimsIdentity;

            //accedemos al metodo para obtener el id del usuario(nombreUsuario)
            nombreUsuario = ConfiguracionJWT.ObtenerTokenUsuario(identity);

            //accedemos al metodo para obtener el codigoEmpleado
            codigoEmpleado = ConfiguracionJWT.ObtenerCodigoEmpleado(identity);
            #endregion

            #region Obtener el alias
            //Obtengo el id de la empresa a la cual pertenece el usuario que inserta un nuevo registro
            idEmpresa = codigoEmpleado.Substring(0, 4);

            //Obtengo los datos de la tabla aliasTabla de la BD
            var listaAlias = await _aliasServicio.GetListaAlias();
            aliasLongitud = listaAlias[0].TablaIdentificador;
            idTabla = listaAlias[0].NumeroAliasTabla.ToString();     //el alias de contribuyente se encuentra en la priem

            for (int j = 4; j <= aliasLongitud.Length; j=j+2)
                idAlias = aliasLongitud.Substring(0, j);
            alias = idEmpresa + idTabla + idAlias+codigoUnicoContribuyente;     //concateno cadenas para obtener el alias
            #endregion

            #region Asignacion de variables al modelo
            natural.contribuyente[i].CodigoUnicoContribuyente = codigoUnicoContribuyente;
            natural.contribuyente[i].AliasContribuyente = alias;
            natural.contribuyente[i].FechaCreacion = fecha;
            natural.contribuyente[i].HoraCreacion = hora;
            natural.contribuyente[i].TipoContribuyente = "Natural";
            natural.contribuyente[i].NumeroNatural = numeroPersonaNatural;
            natural.contribuyente[i].NumeroJuridico = 0;
            natural.contribuyente[i].NombreUsuario = nombreUsuario;
            #endregion

        }

        natural.NumeroNatural = numeroPersonaNatural;
        //llamamos al metodo que esta INaturalService (1)
        await _naturalServicio.GuardarNatural(natural);
        //Si todo esta correcto, mostramos un mensaje de exito(1)
        return Ok(new { message = "Persona Natural ha sido registrado con exito..!" });
    }
    catch (Exception ex)
    {
        //En caso de que ocurra un error lo campturamos y mostramos el mensaje
        return BadRequest(ex.Message +"\n"+ ex.InnerException.Message);
    }
}
```

Figura 3.63: Método añadir persona natural
Elaborado por: Investigador

Lista de Personas Naturales

El endpoint get, retorna una lista de personas naturales que estén registradas en la base de datos

```
[HttpGet]
public async Task<IActionResult> GetListaPersonaNatural()
{
    try
    {
        //almacenamos en una variable el listado de las personas
        var listaPersonas = await _naturalServicio.GetListaPersonasNaturales();

        //En caso de exito devolvemos la lista de personas
        return Ok(listaPersonas);
    }
    catch (Exception ex)
    {
        //En caso de que ocurra un error lo campturamos y mostramos el mensaje
        return BadRequest(ex.Message + "\n" + ex.InnerException.Message);  //InnerException captura
    }
}
```

Figura 3.64: Método lista de personas naturales
Elaborado por: Investigador

Persona Natural por ID

Par invocar otro endpoint de tipo get, es necesario agregar o definir una nueva ruta, de tal manera que evitará confusiones con otro endpoint del mismo tipo. Esta nueva ruta nos retorna los datos de una persona natural mediante su id.

```
//Nuevo endpoint
//localhost:xxxx/api/Natural/GetPersonaNaturalPorNumeroNatural
[Route("GetPersonaNaturalPorNumeroNatural")]
[HttpGet]
public async Task<IActionResult> GetPersonaNaturalPorNumeroNatural(long numeroNatural)
{
    try
    {
        var datosPersona = await _naturalServicio.GetPersonaNaturalPorNumeroNatural(numeroNatural);
        return Ok(datosPersona);
    }
    catch (Exception ex)
    {
        //En caso de que ocurra un error lo campturamos y mostramos el mensaje
        return BadRequest(ex.Message + "\n" + ex.InnerException.Message);  //InnerException captura
    }

}
```

Figura 3.65: Método persona natural por id
Elaborado por: Investigador

Persona Natural por Cédula

De la misma manera se puede obtener los datos de una persona por el número de identificación o cédula, para ello se define una nueva ruta de tipo get, el mismo que recibe como parámetro la cédula para la búsqueda.

```
//Nuevo endpoint
//localhost:xxxx/api/Natural/GetPersonaNaturalPorNumeroNatural
[Route("GetPersonaNaturalPorNumeroIdentificacion")]
[HttpGet]
public async Task<IActionResult> GetPersonaNaturalPorNumeroIdentificacion(string cedula)    //cedula
{
    try
    {
        var datosPersona = await _naturalServicio.GetPersonaNaturalPorNumeroIdentificacion(cedula);
        return Ok(datosPersona);
    }
    catch (Exception ex)
    {
        //En caso de que ocurra un error lo campturamos y mostramos el mensaje
        return BadRequest(ex.Message + "\n" + ex.InnerException.Message);  //InnerException captura e
    }

}
```

Figura 3.66: Método persona natural por cédula
Elaborado por: Investigador

Actualizar datos de una Persona

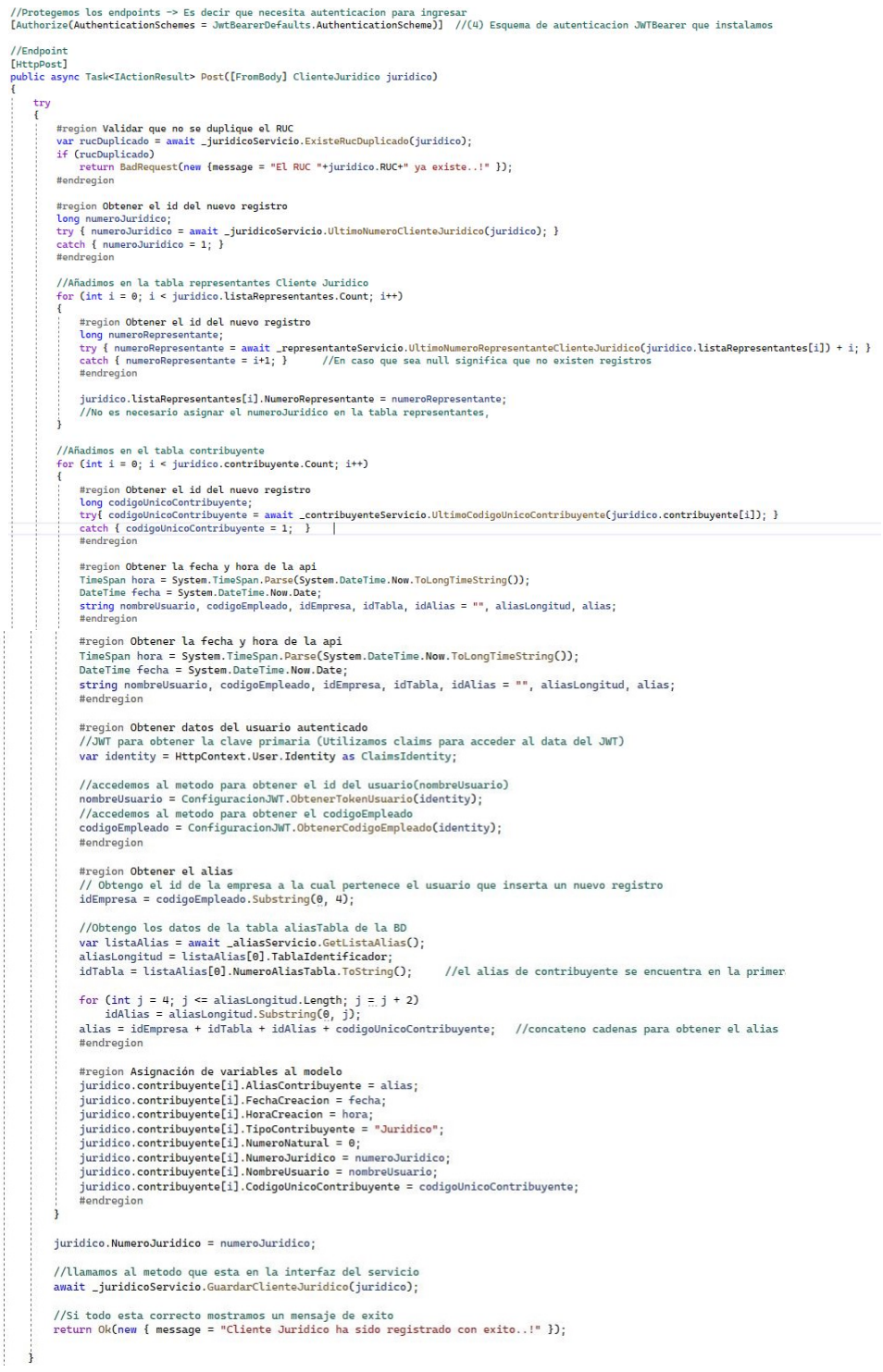
En el caso de actualización no es necesario definir una nueva ruta, debido a que es la única de tipo put, recibe como parámetro un json de tipo Natural y procede con la actualización de datos.

```
[HttpPut]
public async Task<IActionResult> Put([FromBody] Natural natural)
{
    try
    {
        //Nos va a devolver los datos de la persona siempre y cuando exista en la BD
        var datosPersonaActualizar = await _naturalServicio.GetPersonaNaturalPorNumeroNatural(natural.NumeroNatural);
        if (datosPersonaActualizar != null)
        {
            //Asigno los nuevos datos que recibo desde del front
            datosPersonaActualizar.ApellidosNombres = natural.ApellidosNombres;
            datosPersonaActualizar.EstadoCivil = natural.EstadoCivil;
            datosPersonaActualizar.Sexo = natural.Sexo;
            datosPersonaActualizar.FechaNacimiento = natural.FechaNacimiento;
            datosPersonaActualizar.CodigoNivelEstudios = natural.CodigoNivelEstudios;
            datosPersonaActualizar.NumeroTelefono = natural.NumeroTelefono;
            datosPersonaActualizar.NumeroCelular = natural.NumeroCelular;
            datosPersonaActualizar.CorreoElectronico = natural.CorreoElectronico;
            datosPersonaActualizar.ConyugueTipoIdentificacion = natural.ConyugueTipoIdentificacion;
            datosPersonaActualizar.ConyugueNumeroIdentificacion = natural.ConyugueNumeroIdentificacion;
            datosPersonaActualizar.ConyugueApellidosNombres = natural.ConyugueApellidosNombres;
            datosPersonaActualizar.SexoConyugue = natural.SexoConyugue;
            datosPersonaActualizar.ConyugueFechaNacimiento = natural.ConyugueFechaNacimiento;
            datosPersonaActualizar.CodigoBarrio = natural.CodigoBarrio;
            datosPersonaActualizar.Direccion = natural.Direccion;

            //llamamos al metodo que esta INaturalService
            await _naturalServicio.ActualizarDatosPersonaNatural(datosPersonaActualizar);

            //Si todo esta correcto, mostramos un mensaje de exito
            return Ok(new { message = "Los Datos han sido actualizados correctamente..!" });
        }
        else
        {
            return BadRequest("No existen datos");
        }

    }
    catch (Exception ex)
    {
        //En caso de que ocurra un error lo campturamos y mostramos el mensaje
        return BadRequest(ex.Message + "\n" + ex.InnerException.Message);
```

Figura 3.67: Método actualizar datos de una persona natural
Elaborado por: Investigador

Añadir Persona Juridica

Para el registro de una nueva persona jurídica, se crea el endpoint de tipo Post, el mismo que recibe como parámetro un JSON de model Cliente Jurídico, es preciso mencionar que este método inserta una lista de representantes.



```
//Protegemos los endpoints -> Es decir que necesita autenticacion para ingresar
[Authorize(AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme)]  //(4) Esquema de autenticacion JWTBearer que instalamos

//Endpoint
[HttpPost]
public async Task<IActionResult> Post([FromBody] ClienteJuridico juridico)
{
    try
    {
        #region Validar que no se duplique el RUC
        var rucDuplicado = await _juridicoServicio.ExisteRucDuplicado(juridico);
        if (rucDuplicado)
            return BadRequest(new {message = "El RUC "+juridico.RUC+" ya existe..!" });
        #endregion

        #region Obtener el id del nuevo registro
        long numeroJuridico;
        try { numeroJuridico = await _juridicoServicio.UltimoNumeroClienteJuridico(juridico); }
        catch { numeroJuridico = 1; }
        #endregion

        //Añadimos en la tabla representantes Cliente Juridico
        for (int i = 0; i < juridico.listaRepresentantes.Count; i++)
        {
            #region Obtener el id del nuevo registro
            long numeroRepresentante;
            try { numeroRepresentante = await _representanteServicio.UltimoNumeroRepresentanteClienteJuridico(juridico.listaRepresentantes[i]) + i; }
            catch { numeroRepresentante = i+1; }      //En caso que sea null significa que no existen registros
            #endregion

            juridico.listaRepresentantes[i].NumeroRepresentante = numeroRepresentante;
            //No es necesario asignar el numeroJuridico en la tabla representantes,
        }

        //Añadimos en el tabla contribuyente
        for (int i = 0; i < juridico.contribuyente.Count; i++)
        {
            #region Obtener el id del nuevo registro
            long codigoUnicoContribuyente;
            try{ codigoUnicoContribuyente = await _contribuyenteServicio.UltimoCodigoUnicoContribuyente(juridico.contribuyente[i]); }
            catch { codigoUnicoContribuyente = 1;  }
            #endregion

            #region Obtener la fecha y hora de la api
            TimeSpan hora = System.TimeSpan.Parse(System.DateTime.Now.ToLongTimeString());
            DateTime fecha = System.DateTime.Now.Date;
            string nombreUsuario, codigoEmpleado, idEmpresa, idTabla, idAlias = "", aliasLongitud, alias;
            #endregion

            #region Obtener la fecha y hora de la api
            TimeSpan hora = System.TimeSpan.Parse(System.DateTime.Now.ToLongTimeString());
            DateTime fecha = System.DateTime.Now.Date;
            string nombreUsuario, codigoEmpleado, idEmpresa, idTabla, idAlias = "", aliasLongitud, alias;
            #endregion

            #region Obtener datos del usuario autenticado
            //JWT para obtener la clave primaria (Utilizamos claims para acceder al data del JWT)
            var identity = HttpContext.User.Identity as ClaimsIdentity;

            //accedemos al metodo para obtener el id del usuario(nombreUsuario)
            nombreUsuario = ConfiguracionJWT.ObtenerTokenUsuario(identity);
            //accedemos al metodo para obtener el codigoEmpleado
            codigoEmpleado = ConfiguracionJWT.ObtenerCodigoEmpleado(identity);
            #endregion

            #region Obtener el alias
            // Obtengo el id de la empresa a la cual pertenece el usuario que inserta un nuevo registro
            idEmpresa = codigoEmpleado.Substring(0, 4);

            //Obtengo los datos de la tabla aliasTabla de la BD
            var listaAlias = await _aliasServicio.GetListaAlias();
            aliasLongitud = listaAlias[0].TablaIdentificador;
            idTabla = listaAlias[0].NumeroAliasTabla.ToString();     //el alias de contribuyente se encuentra en la primer

            for (int j = 4; j <= aliasLongitud.Length; j = j + 2)
                idAlias = aliasLongitud.Substring(0, j);
            alias = idEmpresa + idTabla + idAlias + codigoUnicoContribuyente;   //concateno cadenas para obtener el alias
            #endregion

            #region Asignación de variables al modelo
            juridico.contribuyente[i].AliasContribuyente = alias;
            juridico.contribuyente[i].FechaCreacion = fecha;
            juridico.contribuyente[i].HoraCreacion = hora;
            juridico.contribuyente[i].TipoContribuyente = "Juridico";
            juridico.contribuyente[i].NumeroNatural = 0;
            juridico.contribuyente[i].NumeroJuridico = numeroJuridico;
            juridico.contribuyente[i].NombreUsuario = nombreUsuario;
            juridico.contribuyente[i].CodigoUnicoContribuyente = codigoUnicoContribuyente;
            #endregion
        }

        juridico.NumeroJuridico = numeroJuridico;

        //llamamos al metodo que esta en la interfaz del servicio
        await _juridicoServicio.GuardarClienteJuridico(juridico);

        //Si todo esta correcto mostramos un mensaje de exito
        return Ok(new { message = "Cliente Juridico ha sido registrado con exito..!" });

    }
```

Figura 3.68: Método añadir persona jurídica
Elaborado por: Investigador

Lista de Personas Jurídicas

En el endpoint de tipo get, para obtener una lista de las personas jurídicas que se han registrado en la base de datos.

```
//Endpoint
[HttpGet]
public async Task<IActionResult> GetListaClientesJuridicos()
{
    try
    {
        //almacenamos en una variable el listado de los clientes juridicos
        var listaJuridicos = await _juridicoServicio.GetListaClientesJuridicos();

        //En caso de exito devolvemos la lista de clientes juridicos
        return Ok(listaJuridicos);
    }
    catch (Exception ex)
    {
        //En caso de que ocurra un error lo campturamos y mostramos el mensaje
        return BadRequest(ex.Message + "\n" + ex.InnerException.Message);  //InnerException
    }
}
```

Figura 3.69: Método lista de personas jurídicas
Elaborado por: Investigador

Persona Jurídica por ID

Para invocar un nuevo endpoint de tipo get, se crea una nueva ruta, la misma que recibe como parámetro un identificador y este nos devolverá los datos de la persona jurídica.

```
//Nuevo Endpoint
//localhost:xxxx/api/ClienteJuridico/GetClienteJuridicoPorNumeroJuridico
[Route("GetClienteJuridicoPorNumeroJuridico")]
[HttpGet]
public async Task<IActionResult> GetClienteJuridicoPorNumeroJuridico(long numeroJuridico)
{
    try
    {
        //almacenamos en una variable los datos del cliente juridico
        var datosJuridico = await _juridicoServicio.GetClienteJuridicoPorNumeroJuridico(numeroJuridico);

        //En caso de exito devolvemos los datos del cliente que se consulto por id
        return Ok(datosJuridico);
    }
    catch (Exception ex)
    {
        //En caso de que ocurra un error lo campturamos y mostramos el mensaje
        return BadRequest(ex.Message + "\n" + ex.InnerException.Message);  //InnerException captura erro
    }
}
```

Figura 3.70: Método persona jurídica por Id
Elaborado por: Investigador

Persona Jurídica por RUC

De igual modo, se crea una nueva ruta, que permita realizar una búsqueda de datos de la persona jurídica por su número de identificación o ruc, el mismo que se envía como parámetro para la respectiva consulta.

```
//Nuevo Endpoint
//localhost:xxxx/api/ClienteJuridico/GetClienteJuridicoPorNumeroJuridico
[Route("GetClienteJuridicoPorNumeroIdentificacionJuridico")]
[HttpGet]
public async Task<IActionResult> GetClienteJuridicoPorNumeroIdentificacionJuridico(string ruc)
{
    try
    {
        //almacenamos en una variable los datos del cliente juridico
        var datosJuridico = await _juridicoServicio.GetClienteJuridicoPorNumeroIdentificacionJuridico(ruc);

        //En caso de exito devolvemos los datos del cliente que se consulto por id
        return Ok(datosJuridico);
    }
    catch (Exception ex)
    {
        // En caso de que ocurra un error lo campturamos y mostramos el mensaje
        return BadRequest(ex.Message + "\n" + ex.InnerException.Message);  //InnerException captura error a n
    }
}
```

Figura 3.71: Método persona jurídica por RUC
Elaborado por: Investigador

Actualizar datos de una Persona Jurídica

En el caso de una actualización no se necesita crear una nueva ruta, ya que es el único método de tipo Put, que recibe como parámetro un JSON de model Cliente Jurídico, lo cual me permite actualizar datos de la persona jurídica y la lista de representantes.

```
[HttpPut]
public async Task<IActionResult> Put([FromBody] ClienteJuridico juridico)
{
    try
    {
        //Nos va a devolver los datos del cliente juridico siempre y cuando exista en la bd
        var datosClienteJuridicoActualizar = await _juridicoServicio.GetClienteJuridicoPorNumeroJuridico(juridico.NumeroJuridico);
        if (datosClienteJuridicoActualizar!=null)
        {
            //Asigno los nuevos datos que recibo desde el front
            datosClienteJuridicoActualizar.RazonSocial = juridico.RazonSocial;
            datosClienteJuridicoActualizar.CodigoActividadEconomica = juridico.CodigoActividadEconomica;
            datosClienteJuridicoActualizar.FechaCreacion = juridico.FechaCreacion;
            datosClienteJuridicoActualizar.NumeroTelefono = juridico.NumeroTelefono;
            datosClienteJuridicoActualizar.NumeroCelular = juridico.NumeroCelular;
            datosClienteJuridicoActualizar.CorreoElectronico = juridico.CorreoElectronico;
            datosClienteJuridicoActualizar.CodigoBarrio = juridico.CodigoBarrio;
            datosClienteJuridicoActualizar.Direccion = juridico.Direccion;

            datosClienteJuridicoActualizar.listaRepresentantes = juridico.listaRepresentantes;

            for (int i = 0; i < juridico.listaRepresentantes.Count; i++)
            {
                var datosRepresentanteActualizar =
                    await _representanteServicio.GetRepresentanteClienteJuridicoPorNumeroRepresentante(juridico.listaRepresentantes[i].NumeroRepresentante);
                if (datosRepresentanteActualizar != null)
                {
                    //Asigno los nuevos datos que recibo desde el front
                    datosRepresentanteActualizar.NumeroNatural = juridico.listaRepresentantes[i].NumeroNatural;
                    datosRepresentanteActualizar.Cargo = juridico.listaRepresentantes[i].Cargo;
                }
                else
                {
                    var numeroRepresentante = await _representanteServicio.UltimoNumeroRepresentanteClienteJuridico(juridico.listaRepresentantes[i]);
                    datosRepresentanteActualizar = juridico.listaRepresentantes[i];
                    datosRepresentanteActualizar.NumeroRepresentante = numeroRepresentante;
                    datosRepresentanteActualizar.NumeroNatural = juridico.listaRepresentantes[i].NumeroNatural;
                    datosRepresentanteActualizar.Cargo = juridico.listaRepresentantes[i].Cargo;

                    await _representanteServicio.GuardarRepresentanteClienteJuridico(datosRepresentanteActualizar);
                }

                //No es necesario asignar el numeroJuridico en la tabla representantes, porq
            }

            //llamamos al metodo que esta en la interfaz del servicio IClienteJuridicoService
            await _juridicoServicio.ActualizarDatosClienteJuridico(datosClienteJuridicoActualizar);

            //Si todo esta correcto, mostramos un mensaje de exito
            return Ok(new { message = "Los Datos han sido actualizados correctamente..!!" });
        }
```

Figura 3.72: Método actualizar datos de una persona jurídica
Elaborado por: Investigador

Añadir Cuenta

El endpoint de tipo Post, recibe como parámetro un JSON de tipo model Medidor, lo cual me permite guardar un medidor y una cuenta asociada, es decir que se insertarán registros en las tablas del mismo nombre mencionadas anteriormente.

```
//Protegemos los endpoints -> Es decir que necesita autenticacion para ingresar
[Authorize(AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme)]
[HttpPost]
public async Task<IActionResult> Post([FromBody]Medidor medidor)
{
    try
    {
        #region Validar que no se duplique el codigo del medidor
        var codigoMedidorDuplicado = await _medidorServicio.ExisteCodigoMedidorDuplicado(medidor);
        if (codigoMedidorDuplicado)
            return BadRequest(new { message = "El código del medidor " + medidor.CodigoMedidor + " ya existe..!" });
        #endregion

        #region Obtener id del nuevo registro
        long numeroMedidor;
        try{ numeroMedidor = await _medidorServicio.UltimoNumeroMedidor(medidor); }
        catch {  numeroMedidor = 1;  }
        #endregion

        //Inicializo variables, para guardar en la tabla maestra y detalle (Medidor - Cuenta)
        string nombreUsuario="", codigoEmpleado, idEmpresa="", idTabla, idAlias = "", aliasLongitud, alias;

        for (int i = 0; i < medidor.cuenta.Count; i++)
        {
            #region Obtener el id del nuevo registro
            long numeroCuenta;
            try { numeroCuenta = await _cuentaServicio.UltimoNumeroCuenta(); }
            catch { numeroCuenta = 1; }
            #endregion

            #region Obtener la fecha y hora de la api
            TimeSpan hora = System.TimeSpan.Parse(System.DateTime.Now.ToLongTimeString());
            DateTime fecha = System.DateTime.Now.Date;

            #endregion

            #region Obtener los datos del usuario autenticado
            //JWT para obtener la clave primaria (Utilizamos claims para acceder al data del JWT)
            var identity = HttpContext.User.Identity as ClaimsIdentity;

            //accedemos al metodo para obtener el id del usuario(nombreUsuario)
            nombreUsuario = ConfiguracionJWT.ObtenerTokenUsuario(identity);

            //accedemos al metodo para obtener el codigoEmpleado
            codigoEmpleado = ConfiguracionJWT.ObtenerCodigoEmpleado(identity);
            #endregion

            #region Obtener el alias de la tabla
            // Obtengo el id de la empresa a la cual pertenece el usuario que inserta un nuevo registro
            idEmpresa = codigoEmpleado.Substring(0, 4);

            //Obtengo los datos de la tabla aliasTabla de la BD
            var listaAlias = await _aliasServicio.GetListaAlias();
            aliasLongitud = listaAlias[1].TablaIdentificador;
            idTabla = listaAlias[1].NumeroAliasTabla.ToString();     //el alias de cuenta se encuentra en la s

            for (int j = 4; j <= aliasLongitud.Length; j = j + 2)
                idAlias = aliasLongitud.Substring(0, j);
            alias = idEmpresa + idTabla + idAlias + numeroCuenta;     //concateno cadenas para obtener el alia
            #endregion

            #region Asignacion de variables al modelo
            medidor.cuenta[i].NumeroCuenta = numeroCuenta;
            medidor.cuenta[i].AliasCuenta = alias;
            medidor.cuenta[i].NumeroMedidor = numeroMedidor;
            medidor.cuenta[i].CodigoUnicoContribuyente = medidor.CodigoUnicoContribuyente;
            medidor.cuenta[i].FechaCreacion = fecha;
            medidor.cuenta[i].HoraCreacion = hora;
            medidor.cuenta[i].SaldoPendiente = 0;
            medidor.cuenta[i].Observaciones = " ";
            medidor.cuenta[i].Estado = "Activo";
            medidor.cuenta[i].CodigoCategoria = medidor.CodigoCategoria;
            medidor.cuenta[i].NombreUsuario = nombreUsuario;
            #endregion
        }

        #region Asignacion de variables al modelo
        medidor.NumeroMedidor = numeroMedidor;
        medidor.Estado = "Activo";
        medidor.CodigoEmpresa = idEmpresa.Substring(0, 2);
        medidor.NombreUsuario = nombreUsuario;
        #endregion

        //llamo al servicio para guardar los datos en la BD
        await _medidorServicio.GuardarMedidor(medidor);
        return Ok(new { message = "Medidor ha sido registrado con exito..!" });
    }
    catch (Exception ex)
    {
        //En caso de que ocurra un error lo campturamos y mostramos el mensaje
        return BadRequest(ex.Message + "\n" + ex.InnerException.Message);
    }
}
```

Figura 3.73: Método añadir cuenta
Elaborado por: Investigador

Activar o Cancelar Cuenta

En el caso de actualizar el estado de una cuenta, es decir que se puede activar o cancelar, el endpoint de tipo Put recibe como parámetro un JSON de tipo model

Cuenta.

```
//Nuevo endpoint
//localhost:xxxx/api/Usuario/ActualizarEstadoCuenta
[Route("ActualizarEstadoCuenta")]
[HttpPut]
public async Task<IActionResult> Put([FromBody]Cuenta cuenta)
{
    try
    {
        //Nos devulebe los datos de la cuenta, siempre y cuando exista en la BD
        var estadoCuentaActualizar = await _cuentaServicio.GetCuentaPorNumeroCuenta(cuenta.NumeroCuenta);
        if (estadoCuentaActualizar != null)
        {
            //Actualizo el estado de la cuenta
            estadoCuentaActualizar.Estado = cuenta.Estado;

            //llamamos al metodo que esta en ICuentService
            await _cuentaServicio.ActualizarEstadoCuenta(estadoCuentaActualizar);

            //Si todo esta correcto, mostramos un mensaje de exito
            return Ok(new { mensaje = "Los datos han sido actualizados correctamnete..!" });
        }
        else
            return BadRequest("No existen datos");

    }
    catch (Exception ex)
    {
        //En caso de que ocurra un error lo campturamos y mostramos el mensaje
        return BadRequest(ex.Message + "\n" + ex.InnerException.Message);
    }
}
```

Figura 3.74: Método activar o cancelar cuenta
Elaborado por: Investigador

Lista de Cuentas

El endpoint de tipo Get, retorna una lista de las cuentas registradas en la aplicación.

```
[HttpGet]
public async Task<IActionResult> GetListaCuentas()
{
    try
    {
        //almacenamos en una variable el listado de cuentas
        var listaCuentas = await _cuentaServicio.GetListaCuentas();

        //En caso de exito devolvemos la lista de cuentas
        return Ok(listaCuentas);
    }
    catch (Exception ex)
    {
        //En caso de que ocurra un error lo campturamos y mostramos el mensaje
        return BadRequest(ex.Message + "\n" + ex.InnerException.Message);  //InnerException c
    }
}
```

Figura 3.75: Método lista de cuentas
Elaborado por: Investigador

Registrar Lectura

Para este caso, el endpoint permite el registro de una lectura inicial por cada cuenta, además el registro de una lectura nueva siempre y cuando se realizan las respectivas validaciones como solo el registro de una lectura por mes, todo esto es posible gracias al model Lectura que recibe como parámetro.

```
//Protegemos los endpoints -> Es decir que necesita autenticacion para ingresar
[Authorize(AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme)]  //Esquema de autenticacion JWTBearer que
[HttpPost]
public async Task<IActionResult> Post([FromBody] Lectura lectura)
{
    try
    {
        #region Obtener la fecha y hora de la api
        TimeSpan hora = System.TimeSpan.Parse(System.DateTime.Now.ToLongTimeString());
        DateTime fecha = System.DateTime.Now.Date;

        lectura.Fecha = fecha;
        lectura.Hora = hora;
        lectura.Año = fecha.Year;
        lectura.NumeroMes = fecha.Month;

        #endregion
        if (lectura.Estado=="Lectura Inicial")
        {
            #region Validar que no exista una lectura inicial registrada en ese numero de cuenta
            var lecturaInicialDuplicada = await _lecturaServicio.ExisteLecturaInicialDuplicada(lectura);
            if (lecturaInicialDuplicada)
                return BadRequest(new { message = "La cuenta #" + lectura.NumeroCuenta +
                    " registra una lectura inicial\nNo se puede duplicar la lectura inicial" });
            #endregion
        }
        else if (lectura.Estado=="Pendiente")
        {
            #region Valido que exista una lectura inicial
            var lecturaInicialExistente = await _lecturaServicio.ExisteLecturaInicialDuplicada(lectura);
            if (lecturaInicialExistente)
            {
                #region Validar que no exista mas de una lectura registrada en el mismo mes
                var lecturaPendienteDuplicada = await _lecturaServicio.ExisteLecturaPendienteDuplicada(lectura);
                if (lecturaPendienteDuplicada)
                    return BadRequest(new { message = "La cuenta # " + lectura.NumeroCuenta +
                        " registra una lectura en el mes " + lectura.NumeroMes + " del " + lectura.Año });
                #endregion

                #region Obtener el valor de la lectura anterior (inicial)
                var lecturaInicial = await _lecturaServicio.GetLecturaAnteriorPorNumeroCuenta(lectura.NumeroCuenta);
                if (lecturaInicial==null)
                    return BadRequest(new { message = "La cuenta # " +lectura.NumeroCuenta+
                        " no registra una lectura inicial" });

                //Asigno el valor de la lectura anterior en el nuevo registro
                lectura.LecturaAnterior = lecturaInicial.LecturaActual;
                #endregion
            }
            #endregion
        }
        #region Obtener el id del nuevo registro
        long numeroLectura;
        try { numeroLectura = await _lecturaServicio.UltimoNumeroLectura();   }
        catch { numeroLectura = 1;   }      //En caso que la consulta me devuelva null, sginfica que no hay registros
        #endregion

        string nombreUsuario = "", codigoEmpleado = "", idEmpresa="", aliasLongitud="", alias="",idTabla="", idAlias="";
        #region Obtener los datos del usuario autenticado
        //JWT para obtener la clave primaria (Utilizamos claims para acceder al data del JWT)
        var identity = HttpContext.User.Identity as ClaimsIdentity;

        //accedemos al metodo para obtener el id del usuario(nombreUsuario)
        nombreUsuario = ConfiguracionJWT.ObtenerTokenUsuario(identity);

        //accedemos al metodo para obtener el codigoEmpleado
        codigoEmpleado = ConfiguracionJWT.ObtenerCodigoEmpleado(identity);
        #endregion

        #region Obtener el alias de la tabla
        // Obtengo el id de la empresa a la cual pertenece el usuario que inserta un nuevo registro
        idEmpresa = codigoEmpleado.Substring(0, 4);

        //Obtengo los datos de la tabla aliasTabla de la BD
        var listaAlias = await _aliasServicio.GetListaAlias();
        aliasLongitud = listaAlias[2].TablaIdentificador;
        idTabla = listaAlias[2].NumeroAliasTabla.ToString();     //el alias de cuenta se encuentra en la tercera posicion

        for (int j = 4; j <= aliasLongitud.Length; j = j + 2)
            idAlias = aliasLongitud.Substring(0, j);
        alias = idEmpresa + idTabla + idAlias + numeroLectura;     //concateno cadenas para obtener el alias
        #endregion

        #region Asignacion de variables al modelo
        lectura.NumeroLectura = numeroLectura;
        lectura.AliasLectura = alias;
        lectura.NombreUsuario = nombreUsuario;
        #endregion

        //Llamo al servicio para guardar los datos en la BD
        await _lecturaServicio.GuardarLectura(lectura);

        return Ok(new { message = "Lectura ha sido registrada con éxito..!" });
    }
    catch (Exception ex)
    {
        //En caso de que ocurra un error lo campturamos y mostramos el mensaje
        return BadRequest(ex.Message + "\n" + ex.InnerException.Message);
    }
}
```

Figura 3.76: Método añadir lectura
Elaborado por: Investigador

Lectura por fechas

El endpoint de tipo Get, recibe como parámetro las fechas de inicio y fin, para devolver una lista de lecturas registras dentro de ese período en la base de datos.

```
//Nuevo Endpoint
//localhost:xxxx/api/Lectura/ObtenerListaLecturaPorFecha
[Route("ObtenerListaLecturaPorFecha")]
[HttpGet]
0 referencias
public async Task<IActionResult> GetListaLecturasPorFecha(DateTime fechaInicio, DateTime fechaFin)
{
    try
    {
        var listaLecturas = await _lecturaServicio.GetListaLecturaPorFecha(fechaInicio, fechaFin);

        return Ok(listaLecturas);
    }
    catch (Exception ex)
    {
        //En caso de que ocurra un error lo campturamos y mostramos el mensaje
        return BadRequest(ex.Message + "\n" + ex.InnerException.Message);  //InnerException captura error a
    }
}
```

Figura 3.77: Método lista lecturas por fechas
Elaborado por: Investigador

Planillas Pendientes de Pago por Numero de Cuenta

El endpoint Get, retorna una lista de planillas pendientes de pago, de un número de cuenta que recibe como parámetro desde el front. Se utilizó una nueva ruta para diferenciarlo.

```
[Route("ObtenerPlanillaPendientePorCuenta")]
//Protegemos los endpoints -> Es decir que necesita autenticacion para ingresar
[Authorize(AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme)]  //(4) Esquema de autenticaci
[HttpGet]
public async Task<IActionResult> GetPlanillasPendientesCuenta(long numeroCuenta)
{
    try
    {
        var listaPlanillas = await _planillaServicio.GetPlanillasPendientesCuenta(numeroCuenta);

        return Ok(listaPlanillas);
    }
    catch (Exception ex)
    {
        // En caso de que ocurra un error lo campturamos y mostramos el mensaje
        return BadRequest(ex.Message + "\n" + ex.InnerException.Message);  //InnerException captura error
    }
}
```

Figura 3.78: Método lista planillas por número de cuenta
Elaborado por: Investigador

Todas las Planillas

Para este caso se utilizó una nueva ruta para evitar confusiones ya que existe otro endpoint de tipo Get, este retorna una lista de todas las planillas.

```
[Route("ObtenerListaPlanillas")]
//Protegemos los endpoints -> Es decir que necesita autenticacion para ingresar
[Authorize(AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme)]  //(4) Esquema de autenticaci
[HttpGet]
public async Task<IActionResult> GetListaPlanillas()
{
    try
    {
        var listaPlanillas = await _planillaServicio.GetPlanillas();

        return Ok(listaPlanillas);
    }
    catch (Exception ex)
    {
        // En caso de que ocurra un error lo campturamos y mostramos el mensaje
        return BadRequest(ex.Message + "\n" + ex.InnerException.Message);  //InnerException captura error
    }
}
```

Figura 3.79: Método lista planillas
Elaborado por: Investigador

Registrar Nueva Recaudación

El endpoint de tipo Post, recibe como parámetro un JSON de tipo model Recaudacion, lo cual permite registrar una nueva recaudación en la tabla de la base de datos. Cabe mencionar que además de realizar la recaudación se actualiza el estado de la planilla a Cancelado, para evitar duplicidad de datos.

```
//Protegemos los endpoints -> Es decir que necesita autenticacion para ingresar
[Authorize(AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme)]  //Esquema de autenticaci
[HttpPost]
public async Task<IActionResult> Post([FromBody] Recaudacion recaudacion)
{
    try
    {
        #region Obtener la fecha y hora de la api
        TimeSpan hora = System.TimeSpan.Parse(System.DateTime.Now.ToLongTimeString());
        DateTime fecha = System.DateTime.Now.Date;

        recaudacion.Fecha = fecha;
        recaudacion.Hora = hora;
        #endregion

        #region Obtener el id del nuevo registro
        long numeroRecaudacion;
        try { numeroRecaudacion = await _recaudacionServicio.UltimoNumeroRecaudacion(); }
        catch { numeroRecaudacion = 1;  }       // En caso que la consulta me devuelva null, sginfica
        #endregion

        string nombreUsuario = "", codigoEmpleado = "", idEmpresa = "", aliasLongitud = "", alias = "
        #region Obtener los datos del usuario autenticado
        //JWT para obtener la clave primaria (Utilizamos claims para acceder al data del JWT)
        var identity = HttpContext.User.Identity as ClaimsIdentity;

        //accedemos al metodo para obtener el id del usuario(nombreUsuario)
        nombreUsuario = ConfiguracionJWT.ObtenerTokenUsuario(identity);

        //accedemos al metodo para obtener el codigoEmpleado
        codigoEmpleado = ConfiguracionJWT.ObtenerCodigoEmpleado(identity);
        #endregion

        #region Obtener el alias de la tabla
        // Obtengo el id de la empresa a la cual pertenece el usuario que inserta un nuevo registro
        idEmpresa = codigoEmpleado.Substring(0, 4);

        //Obtengo los datos de la tabla aliasTabla de la BD
        var listaAlias = await _aliasServicio.GetListaAlias();
        aliasLongitud = listaAlias[5].TablaIdentificador;
        idTabla = listaAlias[5].NumeroAliasTabla.ToString();     //el alias de cuenta se encuentra en
        for (int j = 4; j <= aliasLongitud.Length; j = j + 1)
        {
            if (numeroRecaudacion < 10)
                idAlias = aliasLongitud.Substring(0, j);
            else if (numeroRecaudacion >= 10)
                idAlias = aliasLongitud.Substring(0, j - 1);
            else if (numeroRecaudacion >= 100)
                idAlias = aliasLongitud.Substring(0, j - 2);
            else if (numeroRecaudacion >= 1000)
                idAlias = aliasLongitud.Substring(0, j - 3);
            else if (numeroRecaudacion >= 10000)
                idAlias = aliasLongitud.Substring(0, j - 4);
            else if (numeroRecaudacion >= 100000)
                idAlias = aliasLongitud.Substring(0, j - 5);

        }
        alias = idEmpresa + idTabla + idAlias + numeroRecaudacion;     //concateno cadenas para obtener
        #endregion

        #region Asignacion de variables al modelo
        recaudacion.NumeroRecaudacion = numeroRecaudacion;
        recaudacion.AliasRecaudacion = alias;
        recaudacion.NombreUsuario = nombreUsuario;
        recaudacion.CodigoAgencia = idEmpresa;
        recaudacion.Impreso = false;
        #endregion

        //Llamo al servicio para guardar los datos en la bd
        await _recaudacionServicio.GuardarRecaudacion(recaudacion);

        #region Cambiar el estado cancelado en la tabla planilla
        var planillaCancelada = await _planillaServicio.GetDatosPlanilla(recaudacion.NumeroPlanilla);

        //Asignamos al modelo
        planillaCancelada.Estado = "Cancelado";

        //Esperamos que se actualice el estado de la planilla
        await _planillaServicio.CancelarPlanilla(planillaCancelada);
        #endregion

        return Ok(new { message = "Recaudación ha sido registrada con éxito..!" });
    }
    catch (Exception ex)
    {
        //En caso de que ocurra un error lo campturamos y mostramos el mensaje
        return BadRequest(ex.Message + "\n" + ex.InnerException.Message);
    }
```

Figura 3.80: Método guardar nueva recaudación
Elaborado por: Investigador

Lista de Recaudaciones por Fecha

El endpoint de tipo Get, retorna una lista de recaudaciones que se encuentran registradas en la base de datos, la misma que recibe como parámetro las fechas inicio y fin, es decir que me la lista será dentro de un determinado tiempo. Es preciso indicar que se ha asignado una nueva ruta para identificarlo de mejor manera.

```
// Nuevo Endpoint ...
[Route("ObtenerListaRecaudacionesPorFecha")]
[HttpGet]
public async Task<IActionResult> GetListaRecaudacionesPorFecha(DateTime fechaInicio, DateTime fechaFin)
{
    try
    {
        var listaRecaudaciones = await _recaudacionServicio.GetListaRecaudacionesPorFecha(fechaInicio, fechaFin)

        return Ok(listaRecaudaciones);
    }
    catch (Exception ex)
    {
        //En caso de que ocurra un error lo campturamos y mostramos el mensaje
        return BadRequest(ex.Message + "\n" + ex.InnerException.Message);  //InnerException captura error a nive
    }
}
```

Figura 3.81: Método lista recaudaciones por fecha
Elaborado por: Investigador

Lista de Recaudaciones por Usuario

Para este ednpoint se ha asignado una nueva ruta, la cual retorna una lista de las recaudaciones registradas en la base de datos por el usuario que se encuentre autenticado.

```
//Nuevo Endpoint
//localhost:xxxx/api/Recaudaciones/ObtenerListaRecaudacionesPorUsuario
[Route("ObtenerListaRecaudacionesPorUsuario")]
[HttpGet]
public async Task<IActionResult> GetListaRecaudacionesPorUsuario()
{
    try
    {
        string nombreUsuario;
        #region Obtener los datos del usuario autenticado
        //JWT para obtener la clave primaria (Utilizamos claims para acceder al data del JWT)
        var identity = HttpContext.User.Identity as ClaimsIdentity;

        //accedemos al metodo para obtener el id del usuario(nombreUsuario)
        nombreUsuario = ConfiguracionJWT.ObtenerTokenUsuario(identity);
        #endregion

        var listaRecaudaciones = await _recaudacionServicio.GetListaRecaudacionesPorUsuario(nombreUsuario);
        return Ok(listaRecaudaciones);
    }
    catch (Exception ex)
    {
        //En caso de que ocurra un error lo campturamos y mostramos el mensaje
        return BadRequest(ex.Message + "\n" + ex.InnerException.Message);  //InnerException captura error a ni
    }
}
```

Figura 3.82: Método lista recaudaciones por usuario
Elaborado por: Investigador

Registrar nuevo cierre de caja

Para el registro de un cierre de caja, se crea el endpoint de tipo Post, el mismo que recibe como parámetro un JSON de model CierreCaja, lo cual permite realizar el nuevo registro con toda la información necesaria en la tabla de la base de datos.

```
//Protegemos los endpoints -> Es decir que necesita autenticacion para ingresar
[Authorize(AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme)]  //Esquema de autenticacion
[HttpPost]
public async Task<IActionResult> Post([FromBody] CierreCaja cierre)
{
    try
    {
        #region Obtener la fecha y hora de la api
        TimeSpan hora = System.TimeSpan.Parse(System.DateTime.Now.ToLongTimeString());
        DateTime fecha = System.DateTime.Now.Date;

        cierre.Fecha = fecha;
        cierre.Hora = hora;
        #endregion

        #region Obtener el id del nuevo registro
        long numeroCierre;
        try { numeroCierre = await _cierreServicio.UltimoNumeroCierreCaja(); }
        catch { numeroCierre = 1; }       // En caso que la consulta me devuelva null, sginfica que no h
        #endregion

        string nombreUsuario = "", codigoEmpleado = "", idEmpresa = "", aliasLongitud = "", alias = "",
        #region Obtener los datos del usuario autenticado
        //JWT para obtener la clave primaria (Utilizamos claims para acceder al data del JWT)
        var identity = HttpContext.User.Identity as ClaimsIdentity;

        //accedemos al metodo para obtener el id del usuario(nombreUsuario)
        nombreUsuario = ConfiguracionJWT.ObtenerTokenUsuario(identity);

        //accedemos al metodo para obtener el codigoEmpleado
        codigoEmpleado = ConfiguracionJWT.ObtenerCodigoEmpleado(identity);
        #endregion

        #region Obtener el alias de la tabla
        // Obtengo el id de la empresa a la cual pertenece el usuario que inserta un nuevo registro
        idEmpresa = codigoEmpleado.Substring(0, 4);

        //Obtengo los datos de la tabla aliasTabla de la BD
        var listaAlias = await _aliasServicio.GetListaAlias();
        aliasLongitud = listaAlias[6].TablaIdentificador;
        idTabla = listaAlias[6].NumeroAliasTabla.ToString();     //el alias de cuenta se encuentra en la

        for (int j = 4; j <= aliasLongitud.Length; j = j + 1)
        {
            if (numeroCierre < 10)
                idAlias = aliasLongitud.Substring(0, j);
            else if (numeroCierre >= 10)
                idAlias = aliasLongitud.Substring(0, j - 1);
            else if (numeroCierre >= 100)
                idAlias = aliasLongitud.Substring(0, j - 2);
            else if (numeroCierre >= 1000)
                idAlias = aliasLongitud.Substring(0, j - 3);
            else if (numeroCierre >= 10000)
                idAlias = aliasLongitud.Substring(0, j - 4);
            else if (numeroCierre >= 100000)
                idAlias = aliasLongitud.Substring(0, j - 5);

        }
        alias = idEmpresa + idTabla + idAlias + numeroCierre;     //concateno cadenas para obtener el a
        #endregion

        #region Asignacion de variables al modelo
        cierre.NumeroCierre = numeroCierre;
        cierre.AliasCierre = alias;
        cierre.NombreUsuario = nombreUsuario;
        #endregion

        //Llamo al servicio para guardar los datos en la bd
        await _cierreServicio.GuardarCierreCaja(cierre);

        return Ok(new { message = "Recaudación ha sido registrada con éxito..!" });
    }
    catch (Exception ex)
    {
        //En caso de que ocurra un error lo campturamos y mostramos el mensaje
        return BadRequest(ex.Message + "\n" + ex.InnerException.Message);
    }

}
```

Figura 3.83: Método guardar nuevo cierre de caja
Elaborado por: Investigador

Lista de Cierres de caja por fecha

El endpoint de tipo Get, retorna una lista de cierres de caja registrados en la base de datos, la misma que recibe como parámetro las fechas inicio y fin, de tal

manera que permita filtrar por tiempos específicos, cabe mencionar que dentro del CierreCajaController se ha designado una nueva ruta para identificarlo de mejor manera.

```
//Protegemos los endpoints -> Es decir que necesita autenticacion para ingresar
[Authorize(AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme)]  //Esquema de autenticacion
[Route("ObtenerListaCierresPorFecha")]
[HttpGet]
public async Task<IActionResult> GetListaCierresCajaPorFecha(DateTime fechaInicio, DateTime fechaFin)
{
    try
    {
        var listaCierres = await _cierreServicio.GetListaCierresCajaPorFecha(fechaInicio, fechaFin);

        return Ok(listaCierres);
    }
    catch (System.Exception ex)
    {
        //En caso de que ocurra un error lo campturamos y mostramos el mensaje
        return BadRequest(ex.Message + "\n" + ex.InnerException.Message);  //InnerException captura err
    }
}
```

Figura 3.84: Método lista cierres de caja por fecha
Elaborado por: Investigador

3.2.3.2. Métodos de la Aplicación FrontEnd

La aplicación frontend desarrollada en angular se comunica con el backend mediante la API, todo gracias a endpoints o puntos finales los mismos que se comunica con la lógica de negocio y esta a la capa de acceso a datos.

Ruta de Acceso a la API

Dentro de la carpeta de enviroments, se encuentran archivos de tipos de ambiente, para desarrollo y producción, en este caso se define una variable endpoint para el acceso a la api.

Figura 3.85: Ruta de acceso a la API
Elaborado por: Investigador

Interceptor

El interceptor en Angular brinda un mecanismo de mutar las solicitudes y respuestas http. En este caso se utiliza para el envío del token de autorización del

usuario en la cabecera de cada solicitud que se realice a la API.

Además se aprovecho el interceptor para el manejo de errores, que permite capturar por cada petición en caso de existirlos.

```
@Injectable()
export class AddTokenInterceptor implements HttpInterceptor {

  constructor(private router: Router) {}

  intercept(request: HttpRequest<unknown>, next: HttpHandler): Observable<HttpEvent<unknown>> {
    //almaceno en una variable el token
    const token = localStorage.getItem('token');

    //Si existe el token(autenticacion)
    if(token){
      //agrego al header el token
      request = request.clone({setHeaders:{authorization:`Bearer ${token}`}});
    }
    return next.handle(request).pipe(
      catchError((error:HttpErrorResponse)=>{
        if (error.status === 401) {
          //Muestro mensaje de error
          sweetAlert.fire({
          title:  'Error',
          html:   'Sesión expirada, por favor ingrese de nuevo<br> Ha ocurrido un error',
          icon:   'error'
          });

          //Redirecciono al login
          this.router.navigate(['auth'])
        }

        if (error.status===500) {
          //Muestro mensaje de error
          sweetAlert.fire({
            title:  'Error Servidor',
            html:   'Contactese con el administrador<br> Ha ocurrido un error',
            icon:   'error'
            });
        }

        return throwError(error);
      })
    );
  }
}
```

Figura 3.86: Configuración de interceptor
Elaborado por: Investigador

Guards

Los guards en angular son interfaces que permiten proteger las rutas e indican al enrutador si se permitirá la navegación a una ruta o no.

El método CanActivate devuelve un valor boleano que es el que indica si se debe permitir o no la navegación a una ruta. Si el usuario no está autenticado, será redireccionado a la página de login.

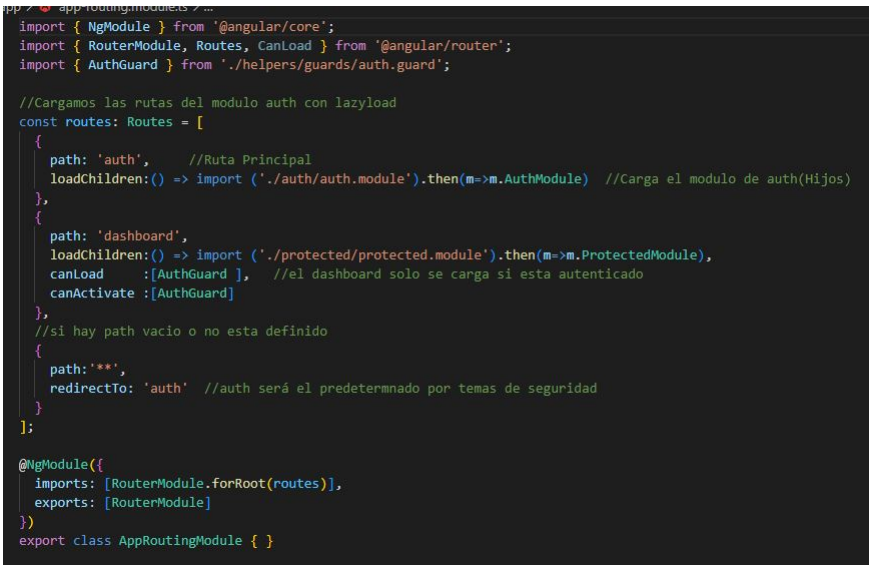
El método CanLoad devuelve un valor boleano que indica si debe cargarse los módulos, los mismos que se cargan con carga perezosa. Es decir que cada módulo contiene rutas hijas.

```
import { Injectable } from '@angular/core';
import { CanActivate, CanLoad, Router } from '@angular/router';
import { AuthService } from '../../auth/services/auth.service';

@Injectable({
  providedIn: 'root'
})
export class AuthGuard implements CanActivate, CanLoad {

  //inyeccion de dependencias
  constructor(private loginService: AuthService, private router:Router){

  }
  //Implementa el canActivate
  canActivate(): boolean{
    //En caso que no exista un token de autenticacion
    if (!this.loginService.getToeknDecode()) {
      this.router.navigate(['auth']); //redirecciono al login
      return false;
    }
    return true   //en caso que si exista un token
  }

  /* route : Es la ruta
    segments: puede retornar un Observable que devuelve un boleano(true | falso
  canLoad(): boolean  {
    if (!this.loginService.getToeknDecode()) {
      this.router.navigate(['auth']); //redirecciono al login
      return false;
    }
    return true;
  }
}
```

Figura 3.87: Configuración de guards
Elaborado por: Investigador

Rutas de Navegación

La aplicación se encuentra organizada mediante módulos, estos contienen las rutas hijas que son cargadas en conjunto con su módulo padre, para este caso se ha definido un módulo auth y protected con el principio de organización que permita el mantenimiento y escalabilidad del proyecto.



```
import { NgModule } from '@angular/core';
import { RouterModule, Routes, CanLoad } from '@angular/router';
import { AuthGuard } from './helpers/guards/auth.guard';

//Cargamos las rutas del modulo auth con lazyload
const routes: Routes = [
  {
    path: 'auth',     //Ruta Principal
    loadChildren:() => import ('./auth/auth.module').then(m=>m.AuthModule)  //Carga el modulo de auth(Hijos)
  },
  {
    path: 'dashboard',
    loadChildren:() => import ('./protected/protected.module').then(m=>m.ProtectedModule),
    canLoad     :[AuthGuard ],   //el dashboard solo se carga si esta autenticado
    canActivate :[AuthGuard]
  },
  //si hay path vacio o no esta definido
  {
    path:'**',
    redirectTo: 'auth'  //auth será el predetermnado por temas de seguridad
  }
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

Figura 3.88: Rutas de navegación principal
Elaborado por: Investigador

El módulo auth, contiene la ruta de navegación a la página login, este archivo puede ser creado en conjunto al módulo con el comando ***ng generate module auth –routing***.

Figura 3.89: Rutas de navegación del módulo auth
Elaborado por: Investigador

Dentro del módulo protected, se definen otros módulos como usuario, personas, cuentas, recaudaciones y reportes, Cabe mencionar que dentro de cada módulo se carga las rutas hijas mediante lazyload (carga perezosa) las mismas que otorgan el acceso a las distintas pantallas de la aplicación web.

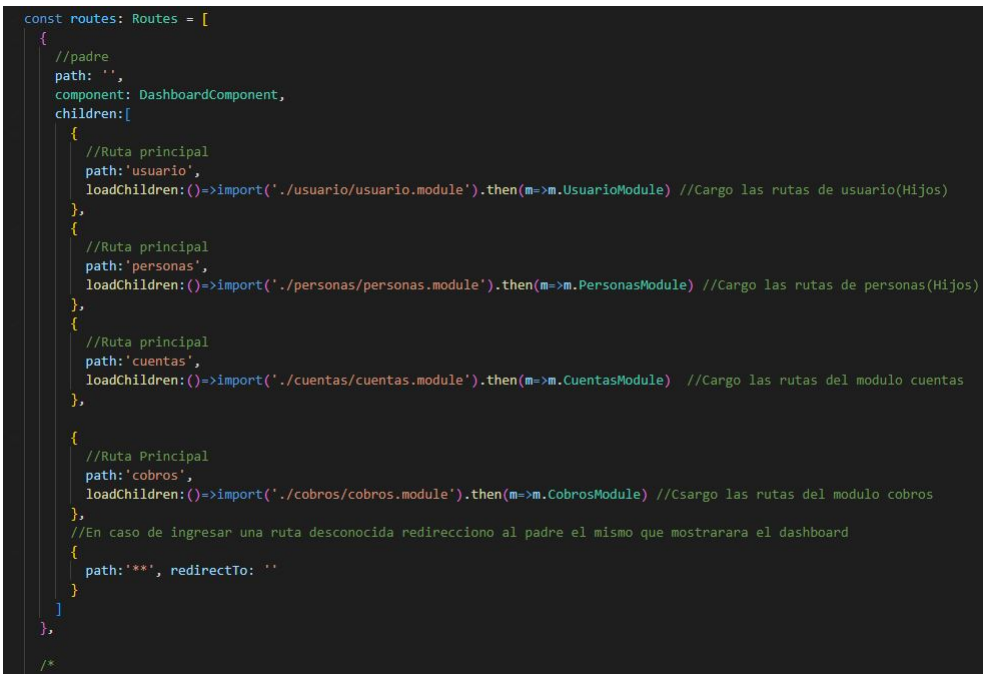


Figura 3.90: Rutas de navegación del módulo protected
Elaborado por: Investigador

Consumo de servicios

En cada módulo se ha configurado servicios, los mismos que pueden ser creados con el comando ng generate service auth/services/auth.

Una vez creado el servicio en angular, se procede a importar la librería HttpClient, para el consumo de servicios. El método "login" utiliza una función de tipo post. Esta función envía una solicitud http y esta devuelve un Observable de tipo AuthResponse que se recibe en respuesta a dicha solicitud.

El método login(usuario : Usuario), toma tres argumentos: URL del endpoint y de la API, además un objeto (usuario), que permite enviar un objeto de tipo JSON al backend para validar las credenciales y porteriormente el acceso a la aplicación web.

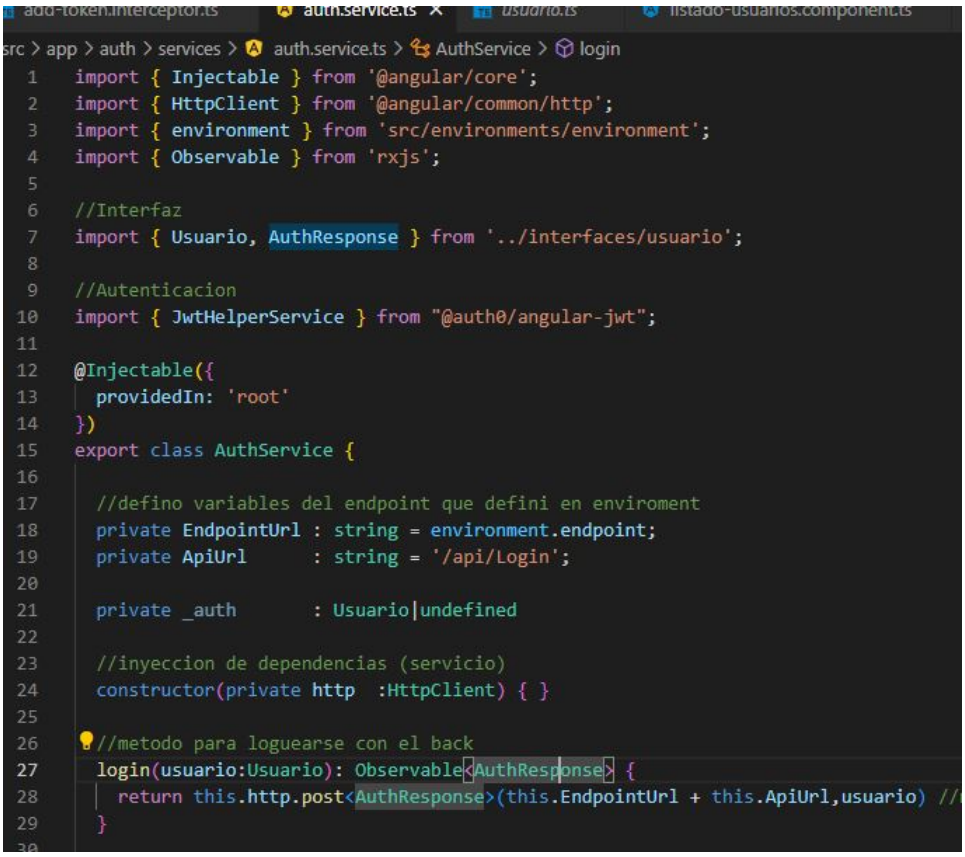


Figura 3.91: Consumo de servicio web - Login
Elaborado por: Investigador

3.2.4. Fase IV: Pruebas

Para la validación de la aplicación web, se procede con las pruebas de aceptación, estas ayudan a evaluar y verificar que las historias de usuarios que fueron definidas anteriormente hayan cumplido con los requerimientos establecidos, de manera que el usuario final confirme el correcto funcionamiento.

Después de finalizar las pruebas de aceptación, la aplicación web está lista para su puesta en producción.

Tabla 3.72: Prueba de aceptación 01

Elaborado por: Investigador

PRUEBA DE ACEPTACIÓN	
Número: 01	**Historia de usuario:** 03
Denominación: Inicio de sesión	
Descripción: Se mostrará una pantalla en la cual el usuario pueda ingresar sus credenciales, mediante autenticación de nombre de usuario y contraseña.	
Condición de ejecución: Ninguna	
Interfaz: La pantalla contiene dos campos de ingreso los cuales son usuario y contraseña, además se muestra un botón para el ingreso a la aplicación. Este botón validará que las credenciales ingresadas se encuentren registradas en la base de datos y sean correctas.	
Resultado esperado: Si las credenciales son correctas, el usuario puede ingresar a la pantalla principal. Al ingresar las credenciales incorrectas muestra un modal con un mensaje de "La contraseña o usuario son incorrectos".	
Resultado de la prueba Prueba aceptada satisfactoriamente.	

Tabla 3.73: Prueba de aceptación 02

Elaborado por: Investigador

PRUEBA DE ACEPTACIÓN	
Número: 02	**Historia de usuario:** 04
Denominación: Pantalla principal	
Descripción: Se mostrará una pantalla donde el usuario podrá encontrar un menú de opciones: Usuarios, Personas, Cuentas, Cobros y Reportes.	
Condición de ejecución: El usuario debe ingresar con el rol de Desarrollador Web para acceder a todas las pantallas que se encuentran organizadas por módulos mendionados anteriormente.	
Interfaz: La pantalla contiene un menú vertical ubicada en la parte izquierda la cual contiene los módulos descritos y en la parte inferior se muestra los datos del usuario.	
Resultado esperado: Si las credenciales son correctas, y se encuentra registrado como desarrollador web, se visualizará la pantalla con el menú especificado, caso contrario retorna a la pantalla de inicio pata el acceso.	
Resultado de la prueba Prueba aceptada satisfactoriamente.	

Tabla 3.74: Prueba de aceptación 03

Elaborado por: Investigador

PRUEBA DE ACEPTACIÓN	
Número: 03	**Historia de usuario:** 05
Denominación: Registrar un nuevo usuario	
Descripción: Se mostrará una pantalla que permita el registro de un nuevo usuario	
Condición de ejecución: El usuario debe ingresar con las credenciales de Desarrollador Web.	
Interfaz: La pantalla contiene varios campos de texto y opciones de selección que son requeridos para el registro. Además contiene dos botones en la parte inferior, el primero permite guardar el usuario, el segundo, que cierra la pantalla sin registrar los cambios en la base de datos.	
Resultado esperado: Una vez que todos los campos sean llenados correctamente, se habilitará el botón guardar que al presionar click se abre un mensajé de éxito y la pantalla se cierra.	
Resultado de la prueba Prueba aceptada satisfactoriamente.	

Tabla 3.75: Prueba de aceptación 04

Elaborado por: Investigador

PRUEBA DE ACEPTACIÓN	
Número: 04	**Historia de usuario:** 06
Denominación: Cambiar contraseña	
Descripción: Se mostrará una pantalla en la cual el usuario pueda actualizar su contraseña de acceso.	
Condición de ejecución: El usuario debe estar autenticado.	
Interfaz: La pantalla contiene tres cajas de texto para el ingreso de la contraseña actual y nueva contraseña la misma que debrá repetir para confirmar. Además contiene dos botones en la parte inferior, el primero permite guardar los cambios, el segundo, que cierra la pantalla sin registrar cambios en la base de datos.	
Resultado esperado: Si la nueva contraseña cumple los parámetros mínimos, se habilitará el botón guardar que al presionar click se abre un mensaje de éxito, y la pantalla se cierra.	
Resultado de la prueba Prueba aceptada satisfactoriamente.	

Tabla 3.76: Prueba de aceptación 05

Elaborado por: Investigador

PRUEBA DE ACEPTACIÓN	
Número: 05	**Historia de usuario:** 07
Denominación: Activar y cancelar usuarios	
Descripción: Se mostrará una pantalla en donde se enlista todos los usuarios registrados en la base de datos, en conjunto con botones de activacion y cancelación por cada fila o usuario.	
Condición de ejecución: El usuario debe ingresar con las credenciales de Desarrollador Web.	
Interfaz: La pantalla refleja una tabla que contiene todos los usuarios registrados, con sus respectivos datos. La última columna de la tabla se observa los botones de Activar y Cancelar, para la respectiva actualización en el estado del usuario seleccionado.	
Resultado esperado: La tabla donde se cargan los datos responde a las actualizaciones realizadas en tiempo real, de tal manera que los cambios serán visibles de manera inmediata.	
Resultado de la prueba Prueba aceptada satisfactoriamente.	

Tabla 3.77: Prueba de aceptación 06

Elaborado por: Investigador

PRUEBA DE ACEPTACIÓN	
Número: 06	**Historia de usuario:** 08
Denominación: Añadir nueva persona natural.	
Descripción: Se mostrará una pantalla en la cual el usuario pueda ingresar los datos de una persona natural.	
Condición de ejecución: El usuario debe ingresar con las credenciales de Desarrollador Web.	
Interfaz: La pantalla contiene campos de texto requeridos y opciones de selección que el usuario deberá ingresar para agregar una nueva persona natural. Además contiene dos botones en la parte inferior, el primero permite guardar los cambios, el segundo, que cierra la pantalla sin registrar cambios en la base de datos.	
Resultado esperado: Si los campos que son requeridos son llenados de manera correcta, el botón para guardar el nuevo registro se habilitará al presionar click sobre ella se abre un mensaje de éxito y procede a cerrarse la pantalla.	
Resultado de la prueba Prueba aceptada satisfactoriamente.	

Tabla 3.78: Prueba de aceptación 07

Elaborado por: Investigador

PRUEBA DE ACEPTACIÓN	
Número: 07	**Historia de usuario:** 09
Denominación: Listado de personas naturales	
Descripción: Se refleja una pantalla con todos los registros de personas naturales registradas.	
Condición de ejecución: El usuario debe ingresar con las credenciales de Desarrollador Web.	
Interfaz: La pantalla contiene una tabla que enlista los datos de las personas registradas. En la parte superior derecha se visualiza un caja de texto que realiza filtros de búsqueda. Finalmente se adjunta un boton para cerrar la pantalla.	
Resultado esperado: En caso de aplicar filtros de busqueda, los registros reducirán de acuerdo con las coincidencias encontradas.	
Resultado de la prueba Prueba aceptada satisfactoriamente.	

Tabla 3.79: Prueba de aceptación 08

Elaborado por: Investigador

PRUEBA DE ACEPTACIÓN	
Número: 08	**Historia de usuario:** 10
Denominación: Editar datos de una persona natural.	
Descripción: Se mostrará una pantalla donde se enlistarán todas las personas naturales registradas en la base de datos, en la parte izquierda de cada fila se encuentra un botón que al presionar click sobre el mismo, permite la modificación de datos.	
Condición de ejecución: El usuario debe ingresar con las credenciales de Desarrollador Web.	
Interfaz: La pantalla contiene varias cajas de texto y opciones de seleccion cargados con los datos actuales. Además contiene dos botones en la parte inferior, el primero permite guardar los cambios, el segundo, que cierra la pantalla sin registrar cambios en la base de datos.	
Resultado esperado: Si los campos y datos nuevos son correctos, se refleja el mensaje de éxito y los cambios de manera satisfactoria.	
Resultado de la prueba Prueba aceptada satisfactoriamente.	

Tabla 3.80: Prueba de aceptación 09

Elaborado por: Investigador

PRUEBA DE ACEPTACIÓN	
Número: 09	**Historia de usuario:** 11
Denominación: Añadir nueva persona jurídica.	
Descripción: Se mostrará una pantalla que permita realizar el registro de una nueva persona jurídica.	
Condición de ejecución: El usuario debe ingresar con las credenciales de Desarrollador Web.	
Interfaz: La pantalla contiene varias cajas de texto y seleccion requeridos para el ingreso de datos de necesarios de una persona jurídica.	
Resultado esperado: Si los datos ingresados son correctos, se habilitará el item o pestaña para añadir los representantes.	
Resultado de la prueba Prueba aceptada satisfactoriamente.	

Tabla 3.81: Prueba de aceptación 10

Elaborado por: Investigador

PRUEBA DE ACEPTACIÓN	
Número: 10	**Historia de usuario:** 12
Denominación: Añadir representantes a una persona jurídica.	
Descripción: En la pantalla de persona jurídica se mostrará un ítem o pestaña que permita añadir los representantes.	
Condición de ejecución: El usuario debe ingresar con las credenciales de Desarrollador Web.	
Interfaz: Dentro de la pantalla persona jurídica se visualiza un item que contiene cajas de texto requeridas para el ingreso del número de cédula, junto con un icono al lado izquierdo que al presionar click, se cargan los datos de la persona a la cual pertenece esa cédula. Finalmente se observa dos botones en la parte inferior, el primero permite guardar los cambios, el segundo, que cierra la pantalla sin registrar cambios en la base de datos.	
Resultado esperado: Si las cédulas ingresadas son correctas y se cargan los datos de los representantes, refleja un modal con mensaje de éxito.	
Resultado de la prueba Prueba aceptada satisfactoriamente.	

Tabla 3.82: Prueba de aceptación 11

Elaborado por: Investigador

PRUEBA DE ACEPTACIÓN	
Número: 11	**Historia de usuario:** 13
Denominación: Listado de personas jurídicas.	
Descripción: Se refleja una pantalla con todos los registros de personas jurídicas registradas.	
Condición de ejecución: El usuario debe ingresar con las credenciales de Desarrollador Web.	
Interfaz: La pantalla contiene una tabla que enlista los datos de las personas jurídicas registradas. Además se adjunta un boton para cerrar la pantalla.	
Resultado esperado: La aplicación lista las personas naturales.	
Resultado de la prueba Prueba aceptada satisfactoriamente.	

Tabla 3.83: Prueba de aceptación 12

Elaborado por: Investigador

PRUEBA DE ACEPTACIÓN	
Número: 12	**Historia de usuario:** 14
Denominación: Editar datos de una persona jurídica	
Descripción: Se mostrará una pantalla donde se enlistan todas las personas jurídicas registradas, en la parte izquierda de cada fila un botón que al presionar click sobre el mismo permite su actualización de datos.	
Condición de ejecución: El usuario debe ingresar con las credenciales de Desarrollador Web.	
Interfaz: La pantalla contiene varias cajas de texto y opciones de seleccion cargados con los datos actuales de la persona jurídica seleccionada. Además contiene dos botones en la parte inferior, el primero permite guardar los cambios, el segundo, que cierra la pantalla sin registrar cambios en la base de datos.	
Resultado esperado: Si los campos y datos nuevos son correctos, se refleja el mensaje de éxito y los cambios de manera satisfactoria.	
Resultado de la prueba Prueba aceptada satisfactoriamente.	

Tabla 3.84: Prueba de aceptación 13

Elaborado por: Investigador

PRUEBA DE ACEPTACIÓN	
Número: 13	**Historia de usuario:** 15
Denominación: Eliminar representante de una persona jurídica.	
Descripción: Se mostrará una pantalla donde se enlistan todas las personas jurídicas registradas, en la parte izquierda de cada fila un botón que al presionar click sobre el mismo permite su actualización de datos.	
Condición de ejecución: El usuario debe ingresar con las credenciales de Desarrollador Web.	
Interfaz: La pantalla contiene el item de representantes, cada uno separado por bloque, en la parte superior de cada bloque se encuentra un icono en forma de basura que permite eliminar el representante de la persona jurídica. Cabe mencionar que se mostrará un modal para la confirmación del registro.	
Resultado esperado: Se eliminan el bloque conjunto con los datos del registro eliminado.	
Resultado de la prueba Prueba aceptada satisfactoriamente.	

Tabla 3.85: Prueba de aceptación 14

Elaborado por: Investigador

PRUEBA DE ACEPTACIÓN	
Número: 14	**Historia de usuario:** 16
Denominación: Registrar una nueva cuenta	
Descripción: Se mostrará una pantalla que permita realizar el registro de una nueva cuenta.	
Condición de ejecución: El usuario debe ingresar con las credenciales de Desarrollador Web.	
Interfaz: La pantalla contiene opción de seleccion y varias cajas de texto requeridas para el ingreso de datos. Además se visualiza dos botones en la parte inferior, el primero guarda los datos en la base de datos y el segundo, cierra la pantalla sin realizar cambios.	
Resultado esperado: Si el número de cédula es correcto se cargarán los datos del contribuyente, al cual se le va asignar una cuenta, posterior a ello se realiza en ingreso de datos, en caso que los campos sean correctos se habilitará el boton guardar.	
Resultado de la prueba Prueba aceptada satisfactoriamente.	

Tabla 3.86: Prueba de aceptación 15

Elaborado por: Investigador

PRUEBA DE ACEPTACIÓN	
Número: 15	**Historia de usuario:** 17
Denominación: Listado de cuentas	
Descripción: Se mostrará una pantalla que enlitará todas las cuentas registradas en la base de datos.	
Condición de ejecución: El usuario debe ingresar con las credenciales de Desarrollador Web.	
Interfaz: La pantalla contiene una table que enlista los datos de las cuentas registradas. En la parte superior derecha en encuentra una caja de texto que permite aplicar varios filtros de búsqueda como nombres, numero de cuenta, etc. Finalmente se adjunta un botón para cerrar la pantalla.	
Resultado esperado: En caso de aplicar los filtros de búsqueda, el listado de registros se reducirán en función de las coincidencias encontradas. Esto lo realiza de manera dinámica conforme va digitando una letra o número.	
Resultado de la prueba Prueba aceptada satisfactoriamente.	

Tabla 3.87: Prueba de aceptación 16

Elaborado por: Investigador

PRUEBA DE ACEPTACIÓN	
Número: 16	**Historia de usuario:** 18
Denominación: Activar y cancelar cuentas	
Descripción: Se mostrará una pantalla en la cual se enlista todas las cuentas registradas en la base de datos, en conjunto con dos botones, uno para la activación y otro para cancelar la cuenta.	
Condición de ejecución: El usuario debe ingresar con las credenciales de Desarrollador Web.	
Interfaz: La pantalla muestra una tabla que contiene todas las cuentas registradas, con sus respectivos datos, al final de cada columna se visualiza los botones Activar y Cancelar, para la respectiva actualización de estado de la cuenta seleccionada.	
Resultado esperado: La tabla donde se cargan los datos responde a las actualizaciones de los estados de manera dinámica, de tal manera que los cambios serán visibles de manera inmediata para el usuario que se enceuntre en esta pantalla.	
Resultado de la prueba Prueba aceptada satisfactoriamente.	

Tabla 3.88: Prueba de aceptación 18

Elaborado por: Investigador

PRUEBA DE ACEPTACIÓN	
Número: 18	**Historia de usuario:** 20
Denominación: Listado de lecturas.	
Descripción: Se mostrará una pantalla para que el usuario pueda revisar las lecturas registradas dentro de una fecha inicio y fecha fin.	
Condición de ejecución: El usuario debe ingresar con las credenciales de Desarrolador Web.	
Interfaz: La pantalla muestra selectores de fechas en la parte superior para aplicar el filtro, mientras en la parte inferior se encuentra una tabla con los datos de las lecturas que se encuentran registradas con mayor detalle para el control continuo de estas lecturas. Al final se muestra un botón para cerrar la pantalla.	
Resultado esperado: Si las fechas seleccionadas son correctas la tabla se cargará de datos en función de las lecturas registradas durante esas fechas, en caso de no econtrar resultados revisar las fechas seleccionadas.	
Resultado de la prueba Prueba aceptada satisfactoriamente.	

Tabla 3.89: Prueba de aceptación 19

Elaborado por: Investigador

PRUEBA DE ACEPTACIÓN	
Número: 19	**Historia de usuario:** 21
Denominación: Cobro de planillas.	
Descripción: Se mostrará una pantalla que le permita recaudar valores por planillas pendientes de pago de una cuenta ingresada.	
Condición de ejecución: El usuario debe ingresar con las credenciales de Recaudador.	
Interfaz: La pantalla evidencia una caja de texto editable, para el numero de cuenta que se desea consultar, luego se muestran los datos del controbuyente al cual pertenece la cuenta. En la parte inferior se refleja las planillas pendientes de pago con su respectivo botón de Pagar.	
Resultado esperado: Al digitar el número de cuenta, se reflejan los datos del contribuyente para evitar errores, además que se despliega la lista de planillas pendientes, luego de presionar click sobre el boton Pagar de la planilla seleccionada, se muestra un mensaje de confirmación, en caso de estar seguro, la tabla se actualiza inmediatamente.	
Resultado de la prueba Prueba aceptada satisfactoriamente.	

Tabla 3.90: Prueba de aceptación 20

Elaborado por: Investigador

PRUEBA DE ACEPTACIÓN	
Número: 20	**Historia de usuario:** 22
Denominación: Listado de planillas.	
Descripción: Se mostrará una pantalla con el registro de todas las planillas registradas en la base da datos.	
Condición de ejecución: El usuario debe ingresar con las credenciales de Desarrollador Web.	
Interfaz: La pantalla muestra un caja de texto en a parte superior derecha, la cual permite aplicar varios filtros de búsqueda como nombres, número de cuenta, etc. Finalmente se ubica en la aprte inferior un botón para cerrar la pantalla.	
Resultado esperado: En el caso de aplicar los fisltros de búsqueda el número de registros a mostrarse se reducirán en función de las concidencias encontradas. es preciso señalar que la búsqueda es de manera dinámica, la tabla realizará la búsqueda cada vez que el usuario digita una letra o número.	
Resultado de la prueba Prueba aceptada satisfactoriamente.	

Tabla 3.91: Prueba de aceptación 21

Elaborado por: Investigador

PRUEBA DE ACEPTACIÓN	
Número: 21	**Historia de usuario:** 23
Denominación: Registrar cierre de caja.	
Descripción: Se refeljará una pantalla que el usuario pueda registrar un nuevo cierre de caja.	
Condición de ejecución: El usuario debe ingresar con las credenciales de Recaudador.	
Interfaz: La pantalla muestra varios paneles para la organización de la información, el primero corresponde a las recaudaciones registradas con el usuario autenticado, el segundo panel respecto a las entradas y salidas de dinero, mientras que el panel de detalle físico para el registro de las cantidad de todas las denominaciónes tanto en billetes como en monedas. Finalmente el panel de datos del cierre, el cual muestra un saldo contable y físico respectivamente.	
Resultado esperado: Una vez que las cantidades ingresadas de las diferentes denominaciones resulta el saldo físico, el cual deberá ser igual al saldo contable, luego se habilitará el botón de guardar y procede con el cuadro de confirmación para el usuario, caso contrario presionar click sobre el botón cerrar para no registrar ningún cambio.	
Resultado de la prueba Prueba aceptada satisfactoriamente.	

Tabla 3.92: Prueba de aceptación 22

Elaborado por: Investigador

PRUEBA DE ACEPTACIÓN	
Número: 22	**Historia de usuario:** 24
Denominación: Listado de cierres de caja.	
Descripción: Se evidenciará una pantalla con todos los registros respecto a los cierres de caja realizados por un usuario dentro de un período de tiempo.	
Condición de ejecución: El usuario debe ingresar con las credenciales de Desarrollado Web.	
Interfaz: La pantalla muestra selectores de fecha ubicados en la parte superior izqueirda, para aplicar filtros de búsqueda por fechas, a continuación la tabla que contiene los registros. Se adjunta al final un botón para cerrar la pantalla.	
Resultado esperado: Al seleccionar las fechas de búsqueda la tabla mostrará un listado de los cierres de caja registrados en el tiempo seleccionado por el usuario.	
Resultado de la prueba Prueba aceptada satisfactoriamente.	

Tabla 3.93: Prueba de aceptación 23

Elaborado por: Investigador

PRUEBA DE ACEPTACIÓN	
Número: 23	**Historia de usuario:** 25
Denominación: Listado de recaudaciones.	
Descripción: Se mostrará una pantalla en la cual el usuario enliste todas las recaudaciones registradas en la base de datos de la aplicación.	
Condición de ejecución: El usuario debe ingresar con las credenciales de Desarrollador Web.	
Interfaz: La pantalla refleja selectores de fecha ubicados en la parte superior izquierda, para aplicar filtros de búsqueda dentro de un tiempo, seguido de una tabla que contiene los registros esperados. Se adjunta un botón para cerrar la pantalla.	
Resultado esperado: Una vez seleccionadas las fechas de inicio y fin, la tabla arrojará los registros que se enceuntren dentro del tiempo seleccionado.	
Resultado de la prueba Prueba aceptada satisfactoriamente.	

3.2.5. Fase V. Lanzamiento

El lanzamiento del aplicativo se lo realizará en días posteriores, ya que el departamento de Tecnologías de la Información de la cooperativa realizar pruebas de testeo y conexión en la red interna.

El proceso inicia con la entrega del proyecto al Gerente de Tecnologías de la Información, donde fue desarrollado la aplicación web, quién será el encargado de avalar que todos los requerimientos han sido solventados de manera exitosa, con el fin de aprobar el proyecto. Posteriormente, es entregado al Analista de Sistemas, quién será el encargado de revisar el código, ejecución de pruebas unitarias y de rendimiento antes de la puesta en producción.

Una vez que el proyecto ha sido revisado y avalado por el departamento, se prosigue con la emisión de informe y puesta en conocimiento a gerencia, después de ser aceptada, el Analista de Sistemas ejecuta las configuraciones en los servidores que se requieran, con el objetivo de subir al servidor de producción y listo para ser utilizado.

Cabe mencionar que el Gerente de Tecnologías de la Información planifica y realiza capacitaciones cobre el manejo correcto del sistema al personal implicado. Se considera un tiempo estimado de 2 meses para culminar con este proceso.

CAPÍTULO IV

CONCLUSIONES Y RECOMENDACIONES

4.1. Conclusiones

La aplicación web permite gestionar el flujo de información de manera estructurada con respecto a contribuyentes, cuentas y lecturas que permiten dar origen a las planillas mensuales, por consiguiente, se procede a la recaudación de valores por consumo de agua potable. Las conclusiones a las cuales se llegó con base a los objetivos alcanzados son:

- Con el modelado de los procesos, se logró identificar los pasos a seguir y el flujo de información previa a la emisión de planillas mensuales, al igual que la conciliación de valores recaudados. De esta manera le permitió al investigador organizar los datos de una manera estructurada, teniendo en cuenta la integridad y disponibilidad de estos. Con la identificación de los procesos mediante diagramas de flujo fue útil para el diseño correcto de la aplicación web.

- Desarrollando la aplicación web con el apoyo del framework Angular, se logró producir una solución escalable, debido a su arquitectura Modelo Vista Controlador (MVC) que separa la lógica de datos de la vista, organizando de mejor manera la estructura de la aplicación, con la creación de módulos y componentes individuales que se cargan únicamente cuando son llamados, evitando así los recargos de página o sobrecargando de componentes innecesarios.

- Se ha desarrollado el Backend del proyecto en ASP.NET, aprovechando al máximo su conexión nativa con el motor de base de datos SQL Server, es preciso señalar que este se encuentra licenciado por la empresa, al igual que en el Frontend aquí se aplicó una arquitectura de diseño denominada RepositoryPattern que da lugar a una organización y separación de la capa de acceso a datos, lógica de negocio y servicios que se son compatibles con múltiples sistemas operativos.

- Desarrollar proyectos utilizando el método ágil XP (Programación Extrema) es muy útil porque permite una programación más organizada, estructurada y mantiene una comunicación constante entre los desarrolladores y los

clientes. Con el objetivo de revisar con cada iteración los resultados obtenidos.

- Gracias a la colaboración de empleados que operan en el área de atención al cliente se determinó que la aplicación es intuitiva y compatible que brinda una buena experiencia al usuario. Además, que abarca con todas las funcionalidades previstas para este proyecto.

4.2. Recomendaciones

- Para agregar nuevas funcionalidades al proyecto, se recomienda utilizar módulos y su ruteo, con carga perezosa (Lazy Loading) ya que mejora notablemente el rendimiento de la aplicación pues sus tiempos de respuesta suelen ser mucho más rápidos.

- Se recomienda utilizar un control de versiones como GitHub, ya que le admite realizar una copia de seguridad de los proyectos y subirlos a la nube, en caso de pérdida de código este servicio permite su recuperación a la última fecha publicada. Además de un trabajo colaborativo entre los desarrolladores de software

- Como recomendación, se debe realizar copias de seguridad de la base de datos cada día, como medida de prevención a un ataque o pérdida de información. Esta tarea se puede automatizarla en el agente administrador de tareas que incorpora SQL server en su versión estándar.

- Se recomienda instalar certificados de seguridad en la página web y mantenerlos vigentes donde se aloja la aplicación, para evitar ataques maliciosos y proteger el ingreso no autorizado.

- Para el desarrollo de nuevos módulos se recomienda, profundizar el uso de entity framework en sus versiones actuales para mejorar el rendimiento de la aplicación a futuro.

Bibliografía

[1] J. P. Torres Bastidas, "Aplicación móvil multiplataforma para la gestión de información georeferencial y servicio técnico comunitario de plomería, aplicando geolocalización offline, en la junta administradora de agua potable de los barrios occidentales de aloasí," B.S. thesis, Universidad Técnica de Ambato. Facultad de Ingeniería en Sistemas, Electrónica e Industrial, 2021.

[2] J. T. Quezada, "Sistema para la recaudación de tarifas por el suministro de agua potable en la junta administradora de agua las américas cantón y provincia de pastaza," *Universidad Regional Autónoma de los Ándes Uniandes*, 2017.

[3] K. P. Chabla Vintimilla, "Implementación de un sistema web de facturación y consulta para la junta administradora de agua potable de mobiloil.," B.S. thesis, Ambato: Universidad Tecnológica Indoamérica, 2017.

[4] G. V. Chiles Arévalo, "Evaluación de la calidad y cantidad de agua de las juntas administradoras de agua potable del cantón montúfar para el diseño de un plan mejoramiento y aprovechamiento adecuado," B.S. thesis, 2015.

[5] W. A. Conde Chicaiza, "Sistema de monitoreo y control para el proceso de potabilización en las juntas administradoras de agua potable," Master's thesis, Universidad Técnica de Ambato. Facultad de Ingeniería en Sistemas, Electrónica e Industrial, 2019.

[6] L. B. I. Robles and M. A. C. Molina, "Sistema de información para la administración de los procesos de las juntas de agua potable," *Polo del Conocimiento: Revista científico-profesional*, vol. 5, no. 1, pp. 23–47, 2020.

[7] T. A. Toapanta Cevallos, "Diagnóstico de la gestión administrativa de las juntas de agua potable y saneamiento del cantón ambato," B.S. thesis, Universidad Técnica de Ambato. Facultad de Ciencias Administrativas, 2017.

[8] E. M. Llerena Ortíz, "Sistema de facturación para el control automatizado de las tarifas recaudadas en las juntas administradoras de agua potable adscritas al parlamento agua del gobierno provincial de tungurahua," B.S. thesis, Universidad Técnica de Ambato. Facultad de Ingeniería en Sistemas, Electrónica e Industrial, 2011.

[9] R. Andreu, J. Ricart, and J. Valor, "Estrategia y sistema de información," in *Estrategia y sistema de información*, 1991.

[10] R. Sturm, C. Pollard, and J. Craig, *Application performance management (APM) in the digital enterprise: managing applications for cloud, mobile, iot and eBusiness*. Morgan Kaufmann, 2017.

[11] F. Luna, C. P. Millahual, and M. Iacono, *PROGRAMACION WEB Full Stack 13-PHP: Desarrollo frontend y backend-Curso visual y práctico*, vol. 13. RedUsers, 2018.

[12] C. A. M. Machuca, "Estado del arte: Servicios web," *Universidad Nacional de Colombia, Tesis de Maestría*, 2010.

[13] C. d. C. de Bogotá, "El mundo conectado por las api," 2019.

[14] C. Ávila Garzón, "Modelo vista controlador," *Ingeniería de Software*, 2019.

[15] Y. Fernández Romero and Y. Díaz González, "Patrón modelo-vista-controlador," *Telem@ tica (La Habana)*, vol. 11, no. 1, pp. 47–57, 2012.

[16] M. A. Jadhav, B. R. Sawant, and A. Deshmukh, "Single page application using angularjs," *International Journal of Computer Science and Information Technologies*, vol. 6, no. 3, pp. 2876–2879, 2015.

[17] G. M. Villalobos, G. D. C. Sánchez, and D. A. B. Gutiérrez, "Diseño de framework web para el desarrollo dinámico de aplicaciones," *Scientia et technica*, vol. 16, no. 44, pp. 178–183, 2010.

[18] U. Z. Castro, *ADO. NET ENTITY FRAMEWORK 4.1-APLICACIONES Y SERVICIOS CENTRADOS EN DATOS.* Krasis Consulting SL, 2011.

[19] C. De la Torre Llorente, U. Z. Castro, M. A. R. Barros, and J. C. Nelson, "Guía de arquitectura n-capas orientada al dominio con .net," 2010.

[20] D. Esposito, *Programming ASP. NET Core*. Microsoft Press, 2018.

[21] R. Anderson, D. Roth, and S. Luttin, "Introduction to asp .net core," 2019.

[22] A. Kumar and R. K. Singh, "Comparative analysis of angularjs and reactjs," *International Journal of Latest Trends in Engineering and Technology*, vol. 7, no. 4, pp. 225–227, 2016.

[23] S. M. Mafla Flores, “Comparativa de los frameworks angular y primefaces para el desarrollo del aplicativo control de materia prima en la empresa mastercubox sa, utilizando la metodología scrum,” B.S. thesis, 2019.

[24] P. N. Moya, R. M. M. Poma, and A. L. A. Anchatuña, “La administración de los sistemas de gestor de base de datos (sgbdâs) de los sistemas de información y su incidencia en el control de las seguridades de las bases de datos,” *REFCalE: Revista Electrónica Formación y Calidad Educativa. ISSN 1390-9010*, vol. 6, no. 1, pp. 57–70, 2018.

[25] J. E. Salazar Cárdenas, “Análisis comparativo de dos bases de datos sql y dos bases de datos no sql,” 2014.

[26] L. H. Ibáñez, *Administración de Sistemas Gestores de Base de Datos*. Grupo Editorial RA-MA, 2015.

[27] R. S. Pressman and J. M. Troya, “Ingeniería del software,” 1988.

[28] M. C. Solís, “Una explicación de la programación extrema (xp),” *V Encuentro usuarios xBase*, 2003.

[29] R. Pressman and T. B. Contreras, “Desarrollo ágil,” *Pablo Roig Vásquez. Ingeniería del software. Ciudad de México: McGrawHill*, p. 69, 2010.

[30] P. Letelier and M. C. Penadés, “Métodologías ágiles para el desarrollo de software: extreme programming (xp),” 2006.

[31] L. A. Anchundia Medrano, “Análisis comparativo de tecnologías front end angular js vs react js, en el modelo de procesos para el desarrollo de aplicaciones web.,” B.S. thesis, Babahoyo: UTB-FAFI. 2022, 2022.

[32] K. J. Mosquera Coronel, “Estudio comparativo de frameworks (java) zk y (php) laravel para desarrollo de aplicaciones web y móviles.,” B.S. thesis, Babahoyo: UTB-FAFI. 2022, 2022.

[33] W. R. Romero García, “Análisis comparativo de los lenguajes de programación node js y asp. net para un sistema de registro de la âfarmacia tu ahorroâ en la ciudad de babahoyo.,” B.S. thesis, Babahoyo: UTB-FAFI. 2022, 2022.

Anexos

Anexo A

Entrevista

Entrevista aplicada al Ing. Francisco Moreta gerente general de la Cooperativa de Ahorro y Crédito "San Martín" de Tisaleo Ltda., quién es el interesado en implementar un software para la recaudación de valores por el consumo de agua potable en la institución.

N°	**Pregunta**
1	¿Cuáles son los servicios que ofrece la cooperativa?
2	¿Cuál es la disponibilidad del servicio de internet en la cooperativa?
3	¿La cooperativa cuenta con un sistema para cobro de servicios básicos?
4	¿Dónde se puede cancelar el valor de una planilla mensual de agua potable del sector?
5	¿Qué características se busca en un sistema web para la recaudación de valores?
6	¿La cooperativa cuenta con la infraestructura tecnológica para el desarrollo de aplicaciones web?
7	¿Existe disponibilidad de servidores para la publicación y consumo de servicios web?
8	¿Qué módulos considera necesarios para la aplicación web?
9	¿Por qué considera que es importante desarrollar una aplicación web para el cobro de planillas mensuales por consumo de agua potable en la cooperativa?