Universidad Nacional de Loja

Facultad de la Energía, las Industrias y los Recursos Naturales no Renovables

Carrera de Ingeniería en Sistemas

Aplicación Web para la automatización del cobro de consumo de agua potable en el Gobierno Autónomo Descentralizado Rural de la Parroquia Milagro del Cantón Atahualpa

Trabajo de Titulación previo a la obtención del título de Ingeniera en Sistemas

AUTOR:

Edhisson Alexis Sanmartín Freire

DIRECTOR:

Ing. Roberth Gustavo Figueroa Díaz, Mg. Sc.

LOJA – ECUADOR

2023

Certificación

Loja, 27 de febrero de 2023

Ing. Roberth Gustavo Figueroa Díaz Mg. Sc.
DIRECTOR DEL TRABAJO DE TITULACIÓN

CERTIFICO:

Que he revisado y orientado todo el proceso de elaboración del Trabajo de Titulación denominado: **Aplicación Web para la automatización del cobro de consumo de agua potable en el Gobierno Autónomo Descentralizado Rural de la Parroquia Milagro del Cantón Atahualpa,** previo a la obtención del título de **Ingeniero en Sistemas,** de la autoría del estudiante: **Edhisson Alexis Sanmartín Freire**, con cédula de identidad Nro. **0705643740,** una vez que el trabajo cumple con todos los requisitos exigidos por la Universidad Nacional de Loja, para el efecto, autorizo la presentación del mismo para su respectiva sustentación y defensa.

Firmado electrónicamente por:
ROBERTH GUSTAVO
FIGUEROA DIAZ

Ing. Roberth Gustavo Figueroa Díaz Mg. Sc.
DIRECTOR DEL TRABAJO DE TITULACIÓN

Autoría

Yo, **Edhisson Alexis Sanmartín Freire**, declaro ser el autor del presente Trabajo de Titulación y eximo expresamente a la Universidad Nacional de Loja y a sus representantes jurídicos de posibles reclamos o acciones legales, por el contenido del mismo. Adicionalmente acepto y autorizo a la Universidad Nacional de Loja, la publicación de mi trabajo de titulación en el Repositorio Institucional - Biblioteca Virtual.

Firma:

Cédula de Identidad: 0705643740

Fecha: 28 de marzo de 2023

Correo electrónico: easanmartinf@unl.edu.ec

Celular: 0962553853

Carta de autorización por parte del autor, para la consulta, reproducción parcial y/o total, publicación electrónica de texto completo del Trabajo de Titulación

Yo **Edhisson Alexis Sanmartín Freire,** declaro ser autor del Trabajo de Titulación denominado: **Aplicación Web para la automatización del cobro de consumo de agua potable en el Gobierno Autónomo Descentralizado Rural de la Parroquia Milagro del Cantón Atahualpa,** como requisito para optar el título de **Ingeniero en Sistemas,** autorizo al sistema Bibliotecario de la Universidad Nacional de Loja para que con fines académicos muestre la producción intelectual de la Universidad, a través de la visibilidad de su contenido de la siguiente manera en el Repositorio Institucional.

Los usuarios pueden consultar el contenido de este trabajo en el Repositorio Institucional, en las redes de información del país y del exterior, con las cuales tenga convenio la Universidad.

La Universidad Nacional de Loja, no se responsabiliza por el plagio o copia del Trabajo de Titulación que realice un tercero.

Para constancia de esta autorización, en la ciudad de Loja, a los veintiocho días del mes de marzo del dos mil veintitrés.

Firmado electrónicamente por:
EDHISSON ALEXIS SANMARTIN FREIRE

Firma:

Autor: Edhisson Alexis Sanmartín Freire

Cédula: 0705643740

Dirección: Parroquia Milagro – Atahualpa – El Oro

Correo electrónico: easanmartinf@unl.edu.ec

Celular: 0962553853

DATOS COPLEMENTARIOS:

Director del Trabajo de Titulación: Ing. Roberth Gustavo Figueroa Díaz Mg. Sc.

Dedicatoria

Dedico este trabajo principalmente a Dios quien me ha brindado de la sabiduría necesaria a lo largo de mi vida universitaria para afrontar cada adversidad; a mi familia, a mis padres Franco y Elsi quienes son un pilar fundamental en mi vida, brindándome el apoyo necesario para cada día dejar el mejor esfuerzo y sacar lo mejor de mí; a mis hermanos Bryan y Nicole quienes me brindaron su cariño y fortaleza en cada momento; a mi querida Nayeli quien con su amor y apoyo incondicional llego para aportar y consolidar cada una de mis metas y sueños; a mi pequeña Dafne Antonella que fue mi motivación principal para culminar este importante logro; a cada uno de mis tíos, tías, primos y abuelitos que con cada uno de sus consejos y ayuda supieron llevarme en cada una de las adversidades a lo largo de mi carrera académica y vida personal; sin cada uno de sus aportes este logro jamás hubiese sido alcanzado, gracias y mil gracias, este logro es de ustedes.

Edhisson Alexis Sanmartín Freire

Agradecimiento

Primeramente, quiero agradecer a Dios por brindarme la sabiduría y la fortaleza necesaria para pasar cada uno de los peldaños y propósitos que han ido surgiendo a lo largo de mi vida, por poner en nuestro camino a familiares, amigos, docentes, compañeros; gente maravillosa que ha estado contribuyendo en cada paso de nuestro crecimiento.

A mi familia, por ser el pilar fundamental en cada una de las situaciones presentadas en el transcurso de mi formación como profesional, apoyándome en cada paso dado, sirviendo de apoyo fundamental, convirtiéndose así; en uno de los factores más importantes para el desarrollo y logro de los objetivos propuestos a lo largo y ancho de mi vida.

Sin lugar a duda a la prestigiosa Universidad Nacional de Loja, a la Facultad de la Energía, las Industrias y los Recursos Naturales no Renovales y a la carrera de Ingeniería en Sistemas por formarme como excelente persona, con un alto valor ético y moral, como futuro líder del mañana.

De igual manera, expresar mis más sinceros agradecimientos al Ing. Roberth Figueroa Díaz, director del presente trabajo de titulación que ha sabido guiarme de la mejor manera para el desarrollo del mismo con cada una de sus experiencias y conocimientos. A cada uno de los docentes de la carrera de Ingeniería en Sistemas por haber compartido cada uno de sus conocimientos y experiencias, de la mejor manera posible brindándonos su amistad y apoyo cuando se lo ha requerido.

Por último, pero sin menor crédito, a nuestros compañeros con los que he compartido buenas y malas experiencias a lo largo de nuestra formación académica.

Edhisson Alexis Sanmartín Freire

Índice de contenidos

Índice de tablas:

Índice de figuras:

Índice de anexos:

1. Título

Aplicación Web para la automatización del cobro de consumo de agua potable en el Gobierno Autónomo Descentralizado Rural de la Parroquia Milagro del Cantón Atahualpa

2. Resumen

Una de las principales ventajas o privilegios que ofrece la tecnología y el mundo moderno es la automatización o ayuda en cada uno de los procesos de las organizaciones, estos parámetros son de vital importancia para ahorrar tiempo, dinero y esfuerzo humano en la realización de tareas constantes. Referente a ello, el objetivo principal del presente trabajo de titulación es desarrollar una plataforma web como servicio para la automatización del cobro de consumo de agua potable en la parroquia Milagro del cantón Atahualpa, con la finalidad de responder la pregunta de investigación: ¿Qué importancia tiene proponer una aplicación web para la automatización del cobro de consumo de agua potable en el Gobierno Autónomo Descentralizado Rural de la Parroquia Milagro del Cantón Atahualpa?

El presente Trabajo de Titulación constó de tres fases, en la primera fase se aplicó la técnica de la entrevista para la obtención de requisitos de la aplicación web, obteniendo el documento de especificación de requerimientos según el estándar IEEE 830. Así mismo, se llevó a cabo el diseño de cada uno de los artefactos tecnológicos de la aplicación con la ayuda de la metodología de desarrollo de software XP y utilizando el modelo arquitectónico 4+1 que se basa en las vistas de escenarios, lógica, física, de despliegue y de procesos con los diagramas de casos de uso, clases, despliegue, componentes y de actividad respectivamente a cada una de ellas para tener una mejor organización del proyecto.

En la segunda fase se desarrolló la aplicación web a través del modelo vista controlador que trae por defecto Django y por ende en este mismo Framework con la ayuda de PostgreSQL como Base de datos, codificando funcionalidades como ingresos de clientes del servicio, ingreso de cifras consumidas por los clientes, registro de cobros del servicio, notificaciones a correos electrónicos de clientes por deudas pendientes del servicio, entre otras funcionalidades más. Finalmente, como última fase, se realizó la prueba de aceptación de la aplicación web con el cliente, misma que se fundamentó en realizar una entrevista donde se valida la aplicación tomando como referencia los requerimientos de la misma y por otro lado la usabilidad del software. De igual manera, la recodificación de la aplicación se basó en desarrollar de mejor manera algunas de las funcionalidades que no cumplían por completo las expectativas del cliente, en este caso, implementar la recuperación de contraseña de ingreso. Así mismo, los test unitarios determinaron que los modelos del software funcionan correctamente; gracias a las pruebas de carga y estrés se definió que la aplicación tiene una cantidad de fallos mínima y, cuenta tiempos de respuesta óptimos entre el cliente y el servidor.

Palabras clave: Desarrollo de software, Aplicaciones web, Framework Django, Servicio de agua.

2.1. Abstract

One of the main advantages or privileges offered by technology and the modern world is automation, which helps to optimize each of the processes of the organizations. These parameters are of vital importance to save time, money and human effort in the realization of constant tasks. Regarding this, the main objective of this titling work is to develop a web platform as a service for the automation of drinking water consumption collection in the parroquia Milagro del cantón Atahualpa, in order to answer the following research question: What Is it important to propose a web application to automate the collection of drinking water consumption in the Rural Decentralized Autonomous Government of the Parroquia Milagro del Cantón Atahualpa?

This Degree Work consisted of three phases, in the first phase the interview technique was applied to obtain requirements of the web application, obtaining the requirements specification document according to the IEEE 830 standard. Likewise, the design of each of the technological artifacts of the application was carried out with the help of the XP software development methodology and using the 4 + 1 architectural model that is based on thes views of scenarios, logic, physics, deployment and processes with the diagrams of use cases, classes, deployment, components and activity respectively to each of them to have a better organization of the project.

In the second phase the web application was developed through the model view controller that brings by default Django and therefore in this same Framework with the help of PostgreSQL as a Database, coding functionalities such as customer income of the service, entry of figures consumed by customers, record of service charges, Notifications to customer emails for outstanding debts of the service, among other functionalities. Finally, as a last phase, the acceptance test of the web application was carried out with the client, which was based on conducting an interview where the application is validated taking as reference the requirements of the same and on the other hand the usability of the software. Similarly, the recoding of the application was based on better developing some of the functionalities that did not completely meet the client's expectations, in this case, implementing login password recovery. Likewise, the unit tests determined that the software models work correctly; Thanks to the load and stress tests, it was defined that the application has a minimum number of failures and has optimal response times between the client and the server.

keywords: Software Development, Web Applications, Django Framework, Water Service.

3. Introducción

En la actualidad el progreso tecnológico, la innovación dentro de las organizaciones y la sociedad ha ido evolucionando con vías claras hacia el futuro, lo que repercute en una transcendencia importante en el campo de la dinámica competitiva y la viabilidad de cada una de las instituciones [1], la implementación de nuevas tecnologías dentro de los procesos en las organizaciones es de vital importancia para mejorar su calidad con las virtudes que nos ofrece la tecnología.

El Gobierno Autónomo Descentralizado Parroquial Rural de Milagro es una importante institución que actualmente cuenta con el proceso de proveer el servicio de agua potable a la comunidad, este proceso consta desde la conexión del servicio hasta la recaudación mes a mes de las cifras consumidas por cada uno de los usuarios del servicio y posteriormente se calcula los valores a cancelar, toda esta secuencia de actividades que lleva actualmente la organización es realizada manualmente, sin las funcionalidades que un software podría brindar.

El presente trabajo de titulación tiene como propósito el desarrollo de una aplicación web que ayude a mejorar el proceso de cobros por el servicio brindado por la Institución a la parroquia, mejorando así notablemente cada una de las actividades y ofreciendo funcionalidades acordes a las necesidades que requiere la institución.

Para el desarrollo de la aplicación web es importante conocer algunos temas fundamentales que serán abordados en el presente trabajo, la metodología de desarrollo de software XP será la encargada de guiar de manera ordenada cada una de las fases del proyecto; de igual manera es importante hacer uso de una arquitectura de software para tener una idea y visión general de cómo se llevará el desarrollo de la aplicación, se ha optado por llevar la arquitectura basada en el modelo arquitectónico 4+1 que hace uso de las vistas para cada uno de los interesados del proyecto. Para la codificación de la aplicación se ha optado por el framework de desarrollo Django, este potente entorno permite crear aplicaciones o sitios web complejos, atractivos, de manera dinámica, en un tiempo corto y con un alto grado de complejidad [2]. Por último, es importante realizar cada una de las pruebas del sistema, tanto de parte del desarrollador como del cliente, lo que permite tener fiabilidad, aceptación y confianza en el proyecto desarrollado.

Por lo anteriormente mencionado, el presente trabajo de titulación propone el desarrollo de una aplicación web con el objetivo de dar respuesta a la pregunta de investigación**: ¿Qué importancia tiene proponer una aplicación web para la automatización del cobro de**

consumo de agua potable en el Gobierno Autónomo Descentralizado Rural de la Parroquia Milagro del Cantón Atahualpa?

El presente TT está compuesto por las siguientes secciones: Marco teórico donde se engloban todos los conceptos necesarios que fueron utilizados como guía para el entendimiento del presente trabajo; la sección de Materiales y métodos detalla el procedimiento que fue necesario para la realización del trabajo de titulación, de igual manera se muestra el detalle de los participantes que intervinieron en el desarrollo del mismo y los recursos utilizados durante su desarrollo. En los resultados se detalla cada uno de los hallazgos encontrados al cumplir cada uno de los objetivos específicos, lo que engloba el cumplimiento del objetivo general; la sección de Discusión brinda la interpretación por parte del autor de cada uno de los resultados obtenidos. Finalmente se muestra una sección con las conclusiones encontradas al desarrollar cada uno de los objetivos y las Recomendaciones para futuros trabajos relacionas al presente trabajo de titulación.

4. Marco teórico

Esta sección consta de los primordiales conceptos afines al desarrollo de una aplicación web que generan un aporte directo aclarando dudas y definiendo términos relevantes dentro de la investigación.

4.1. Software como servicio (Saas)

Saas (Software-as-service), Software como servicio es un modelo que se ocupa de la distribución de software proporcionando a los usuarios acceso a las aplicaciones a través de internet. El software se proporciona como servicio ayudando a optimizar los costos, que repercute en la optimización de recursos e infraestructura por parte del cliente y por parte del proveedor, brinda economías a escala [3].

Las aplicaciones que se ofrecen a los usuarios mediante este modelo (Saas) no incluyen un soporte físico, están alojadas y se ejecutan en servidores remotos del proveedor del servicio en forma de hosting, y se puede acceder a ellas a través de la red utilizando un navegador web de computadora o teléfonos celular inteligente [3]. En la actualidad, se cuenta con una gran variedad de proveedores de servicios para alojamiento de aplicaciones en la nube, entre los cuales se distingue a Heroku, Google Cloud, Amazon Aws, Microsoft Azure, Render, entre muchos más.

Tabla 1. *Proveedores de servicios para alojamiento de aplicaciones en la Nube.*

Proveedor	Descripción	Precios
Heroku	Trabaja con la Base de datos PostgreSQL. Se integra con GitHub.	Básico – Eco: 5 dólares y más por mes. Producción: 25 dólares y más por mes. Avanzado: 250 dólares y más por mes. Empresa: más de 250 dólares y más por mes.
Google Cloud	Ofrece gran fiabilidad al brindar la misma infraestructura que los servicios ofrecidos por Google. Brinda precios detallados y transparentes.	Plan gratuito hasta alcanzar los 300 dólares. Incluye cargos por operaciones, la tarifa depende del consumo y la región.
Amazon Aws	Permite pagar por la tarifa consumida. Múltiples formas de pago.	Ofrece niveles gratuitos para pruebas. Permite pagar por la tarifa consumida. Desarrollador: 29 dólares. Negocio: 100 dólares y más.
Microsoft Azure	Ofrece gran seguridad. Tiene herramientas sofisticadas de control de costos	Ofrece niveles gratuitos para pruebas. Pagos por la tarifa consumida

Render	Implementaciones con Git. Simplicidad y actualizaciones automáticas.	Ofrece planes gratuitos. Existes tarifas por consumo de algunos servicios.

El alojamiento de la aplicación web se realizó en los servidores de render, esto debido al gran servicio gratuito que ofrece para realizar pruebas de despliegue del proyecto, por tener un gran acoplamiento con GitHub permitiendo desplegar automáticamente la aplicación con cada actualización del código fuente y por brindar tarifas económicas de consumo [4].

4.2. Metodología de desarrollo ágil Extreme Programming (XP)

Extreme Programming XP es una metodología de desarrollo ágil que permite reunirse y comunicarse constantemente con los clientes, en función a este contacto constante proporciona al desarrollador de software una solución al problema planteado tomando como base la retroalimentación que ofrece la misma metodología. La documentación está basada primordialmente en las historias de usuarios y las tarjetas CRC, de igual manera fomenta en los proyectos la implementación de un diseño simple y sin complejidad que no compliquen a los desarrolladores [5].

Algunas de las principales características que se encuentran dentro de la metodología Extreme Programming son [6]:

- Frecuentes reuniones de trabajo donde hay una interacción directa entre el cliente y el equipo de desarrolladores.
- Código simple, recurre primordialmente por la simplicidad y sencilles con la capacidad de agregar fácilmente funciones complejas más adelante cuando sea necesario.
- Reescribir partes del código sin la necesidad de afectar su funcionalidad.
- Permite que todos los miembros del proyecto se concentren en la misma tarea, brindando soluciones y correcciones a fallos del proyecto.
- Añadir nuevas funcionalidades luego de la corrección de errores.
- Entregas con mejoras frecuentes.
- Recomienda el trabajo y la programación en pareja, que ayuden a diseñar y discutir características que ofrezcan código de alta calidad.
- Pruebas unitarias continuas.

Las fases que consta la metodología XP son [6]:

- **Planificación. –** En esta fase, la funcionalidad del software se define utilizando historias de usuario basadas en las necesidades del cliente. El tiempo de desarrollo de la aplicación se puede calcular o estimar en función a las historias de usuario que

requiera el cliente. Las historias de usuario utilizadas en esta fase son muy similares a los casos de uso, solo que aquí los requisitos del cliente se definen en un lenguaje común.

- **Diseño. –** La metodología XP recomienda y promueve un diseño simple y claro para que los desarrolladores puedan implementar fácilmente las funcionalidades del programa de software y así reducir el tiempo de desarrollo.
- **Desarrollo.** – El desarrollo de software es una tarea conjunta entre programadores y clientes, en este punto se necesita fundamentalmente que el cliente este constante debido a que es él quien verifica y aprueba funcionalidad del software. Las historias de usuario son responsables de ser la norma fundamental de la codificación para cada funcionalidad.
- **Pruebas. –** Cuentan con la finalidad de evaluar el código para verificar si existen errores y mitigarlos. Las pruebas se realizan antes de que se escriba el código y se realizan continuamente ante cambios realizados en el código de alguna de las funcionalidades.

4.3. Aplicación web

Una aplicación web, es un software al que acceden los clientes por la web por una determinada red a través de una conexión a un servidor, como internet o intranet. De manera más amplia, se puede definir como un programa de computadora que se ejecuta por un cliente desde un navegador web que admite un lenguaje de programación específico para ese propósito. Los navegadores web pueden interpretar e interactuar con las aplicaciones para mostrar información a los usuarios [7].

La comunicación entre el cliente y el servidor realiza mediante el protocolo HTTP (Protocolo de transferencia de hipertexto) y sigue un sistema de solicitud-respuesta que permite transferir información a través de la web. Para realizar una petición, el cliente la realiza mediante las rutas URL [8]. En la imagen descrita a continuación se puede visualizar el intercambio de peticiones entre un cliente y un servidor:

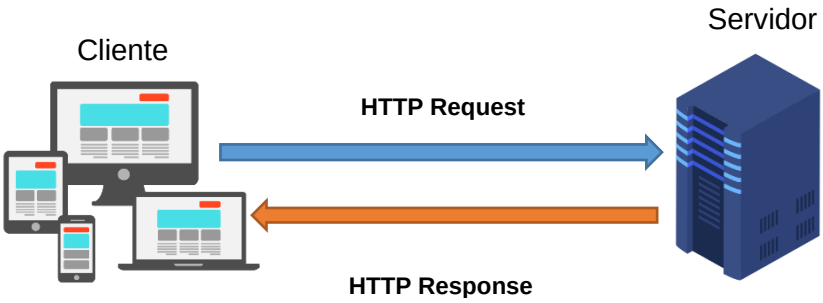


Figura 1. *Petición - respuesta HTTP*

4.4. Especificación de requisitos según el estándar IEEE – 830

Una de las tareas más importantes en el ciclo de vida del proyecto es determinar los requisitos del software, en este punto se definen los “planos” para el desarrollo de la nueva aplicación. Los requisitos del software representan las necesidades del usuario en la aplicación que está en vías de desarrollo, de aquí nace y se crea un documento que describe cada requisito y explica lo que se supone que debe hacer el programa [9]. Actualmente el estándar el estándar IEEE 830 a evolucionado al ISO/IEC/IEEE 29148.

La especificación de los requisitos del sistema – ERS, cuenta con los siguientes objetivos:

- Ayudar al cliente a describir los procesos que se quieren plasmar mediante un software. El cliente es participante directo en toda la fase que comprende la ERS debido a que el conoce específicamente cada uno de los procesos a determinar.
- Ayudar a los desarrolladores a saber en detalle que es lo que exactamente quiere el cliente en el software, en base a ello el desarrollador de la aplicación tiene una base fundamental en la que puedan trabajar. Es importante realizar una buena especificación de requisitos debido a que es la guía fundamental de los desarrolladores, si la misma está mal definida surgirán cambios a futuro en el desarrollo de la aplicación lo que implica una pérdida de tiempo.
- Servir de base para desarrollos estándares de cada organización.

La ERS debe estar plasmada de cierta y determinada manera, por ello el Instituto de Ingenieros Eléctricos y Electrónicos IEEE ha realizado una guía a seguir, a continuación se puede visualizar el esquema del estándar IEEE 830:

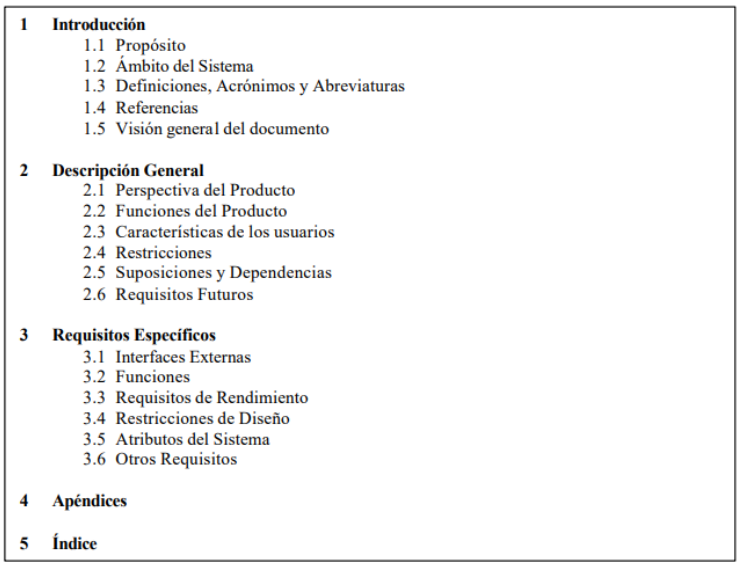


Figura 2. *Estructura de la Especificación de Requisitos del sistema - IEEE 830 [9].*

4.5. Modelo arquitectónico 4+1

El modelo “4+1” propuesto por el profesor Philippe Kruchten para el desarrollo de software se compone de abstracción, análisis, de composición y descomposición, de patrones y estética, de cada uno de los escenarios con los que cuenta el software haciendo uso de los distintos diagramas en cada una de las vistas del modelo [10]. La arquitectura 4+1 utiliza múltiples vistas, que incluyen: la vista lógica, de procesos, física, y de desarrollo; las mismas permiten que cada parte involucrada en el sistema plantee individualmente sus intereses, abordando los requisitos funcionales y no funcionales de los clientes, administradores del proyecto, ingenieros de sistemas, desarrolladores y otros involucrados con el proyecto que se está desarrollando.

En la imagen se puede observar los “Stakeholders” que intervienen en cada uno de los escenarios:

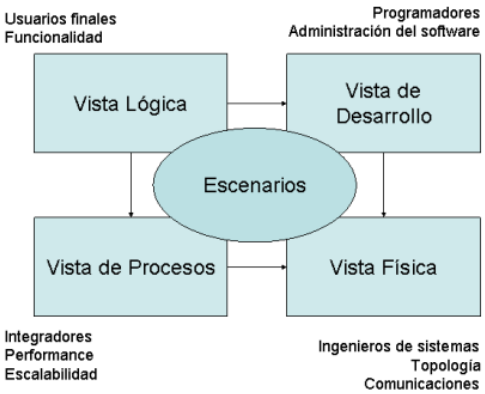


Figura 3. *Modelo 4+1, escenarios [10].*

- **Vista lógica:** se enfoca en los usuarios finales y describe el modelo de diseño basado en el método orientado a objetos. Los diagramas de entidad-relación son utilizados predominantemente en esta vista. La vista lógica es la encargada de brindar soporte a los requerimientos funcionales, es decir, lo que el software en cuestión debe ofrecer a los usuarios en términos de servicio [11].
- **Vista de procesos:** se encarga de describir el flujo de los datos durante el procesamiento de los mismos en el software, detallando los algoritmos que se emplean para ofrecer las funcionalidades requeridas por el cliente [11]. Se centra en garantizar la integridad, escalabilidad, tolerancia a fallos y distribución del sistema mediante el uso de diagramas de procesos. Los procesos se definen como un conjunto de tareas que forman una unidad.
- **Vista física:** mapea el software a los diferentes componentes de hardware identificados en una red de computadoras. La vista física muestra la distribución de los componentes en los diferentes nodos del sistema, es decir, explica cómo se posiciona cada parte del software en un nodo de tal manera que se mapean software y hardware [11].
- **Vista de desarrollo:** se centra en el entorno de desarrollo y organiza los módulos de software de manera sistemática. La vista de desarrollo se enfoca en organizar los módulos de software en el entorno de desarrollo del software. El software se divide en pequeñas partes, bibliotecas de programas o subsistemas, que pueden ser desarrollados por un solo desarrollador o un pequeño grupo de ellos [10].

La quinta vista ***"+1"*** se compone de los casos de uso o escenarios donde los diseñadores de software presentan sus decisiones. Su función consiste en establecer la relación entre los elementos presentes en las otras cuatro perspectivas, mediante la utilización de casos de uso o escenarios que ejemplifiquen la interacción entre todos estos elementos [12].

4.6. Herramientas de desarrollo

Para el desarrollo de la aplicación web del presente trabajo de titulación se utilizó el framework de desarrollo Django basado en el lenguaje de programación Python, las herramientas de diseño Bootstrap, el patrón Modelo Vista Controlador aplicado a la lógica de Django.

4.6.1. Framework Django

Django es un framework de desarrollo web de alto nivel que se basa en el lenguaje de programación Python. Originalmente diseñado por el equipo de desarrollo del diario Lawrence Journal-World, ha sido modificado varias veces para adaptarse a sus necesidades. De ahí la idea de seguir el principio DRY (Don't Repeat Yourself) en español no te repitas, que permite

la reutilización del código ya escrito, lo que ahorra tiempo, esfuerzo y conduce a un diseño más limpio [13].

Entre algunas ventajas de usar el framework Django se cuenta con:

- Facilidad de uso: ayuda a los desarrolladores a llevar las aplicaciones desde el concepto hasta su culminación lo más rápido posible.
- Seguridad: ayuda a los desarrolladores a evitar errores de seguridad comunes a través de un conjunto de acciones.
- Escalabilidad: gracias a su capacidad, se puede ampliar de forma rápida y flexible.
- Versatilidad: puede crear cualquier cosa, desde un sistema de administración de contenido hasta una red social.

Este poderoso framework cuenta con la gran ventaja de que su equipo se encarga de desarrollar muchas características principales de la mejor manera posible, lo que se puede entender a que el desarrollador se ocupe a codificar con un menor grado de dificultad algunas funcionalidades específicas. En la **figura 4**, se puede observar el funcionamiento general con el que cuenta Django:

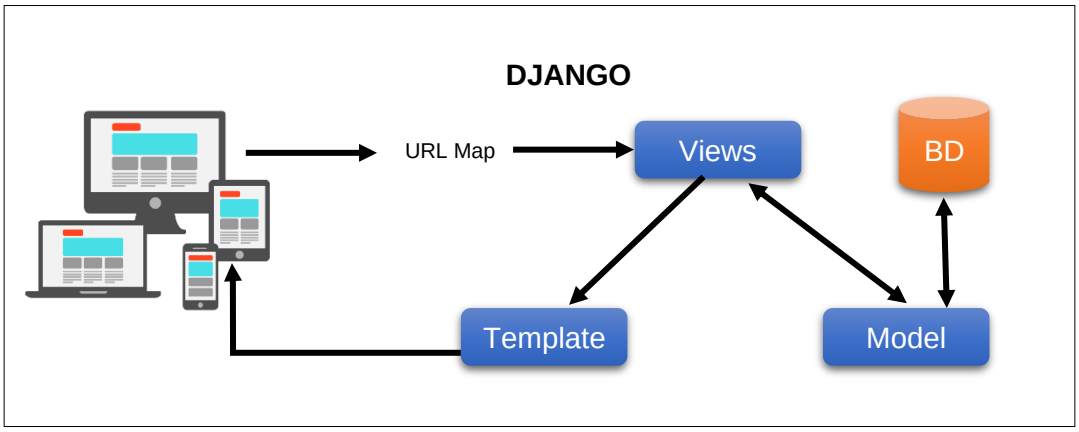


Figura 4. *Funcionamiento de Django.*

4.6.2. Python

Python es un lenguaje de programación orientado a objetos de alto nivel que incluye algunas estructuras de datos implícitas que permiten realizar algunas tareas complejas como lo son los diccionarios, listas, tuplas, conjuntos, etc., con pocas líneas de código y de manera legible. Fue desarrollado por el programador Guido van Rossum en 1990 con la idea de Software Libre. Se utiliza para desarrollar todo tipo de software como entre los cuales se tiene a: Spotify, Netflix, Instagram, entre otros [14]. Las ventajas al desarrollar con este lenguaje de programación son:

- Librería estándar: contiene un sin número de funciones que satisfacen las necesidades básicas de la mayoría de los desarrolladores, las más importantes son el manejo de excepciones, estructuras de datos, funciones numéricas y matrices.
- Rendimiento: todas las librerías estándar están implementadas en el lenguaje C de alto rendimiento.
- Documentación: incluye un sistema de documentación completo e integral para ayudar a corregir errores, desarrollar y mejorar funcionalidades de mejor manera.
- Extensibilidad: permite la reutilización de código escrito en los lenguajes C y C++.
- Licencias: es posible producir software que se redistribuya libremente, lo que permite al desarrollador proporcionar solo su código fuente si así lo desea.

4.6.3. Base de datos

Una base de datos es una estructura de datos interrelacionados que permiten la gestión de información en entornos específicos diseñados para satisfacer las necesidades de información de una empresa o cualquier otra organización [15].

Una base de datos se puede considerar como un gran almacén de datos que se define y crea una sola vez, pero que muchos usuarios pueden usar al mismo tiempo y se puede combinar dentro de una organización para cada departamento sin duplicar datos.

El autor [16] en su informe “PostgreSQL”, realiza una tabla comparativa con dos de los principales gestores de bases de datos con las principales características de cada uno de ellos, misma que se describe a continuación:

Tabla 2. *Características principales de PostgreSQL y MySQL (Fuente: PostgreSQL – Patricio Denzer).*

Sistema	MySQL	PostgreSQL
Licencia	GPL	BSD
Cumplimiento con estándar SQL	Medio	Alto
Velocidad	Media / Alta	Media
Estabilidad	Alta / Muy Alta	Alta
Seguridad	Alta	Alta
Interfaces de programación	ODBC, JDBC, C/C++, OLEDB, Delphi, Perl, Python, PHP	ODBC, JDBC, C/C++, SQL embebido (en C), Tcl/Tk, Perl, Python, PHP

Tipos de tablas alternativas	ISAM, MYISAM, BerkeleyDB, InnoDB, HEAP, MERGE, Gemini	PostgreSQL mantiene su propio sistema de tipos de tablas
Propiedad DBMS	Sistema de gestión de bases de datos puramente relacional (RDBMS)	Sistema de gestión de bases de datos puramente relacional de objetos (ORDBMS)

El gestor de Base de Datos PostgreSQL al soportar netamente conceptos orientados a la programación orientada a objetos, manejar grandes volúmenes de información, ofrecer grandes facilidades en su manejo, brindar seguridad y acoplarse de mejor manera al desarrollo de la aplicación web, es el que se empleó en el presente proyecto.

4.6.3.1. PostgreSQL

PostgreSQL es un administrador de base de datos de código abierto avanzado que se usa ampliamente en entornos de software gratuito, debido a sus características y beneficios avanzados es uno de los mejores administradores de bases de datos, incluso en la misma posición o mejor que los administradores de bases de datos pagadas, esto por sus desarrolladas funcionalidades y prestaciones [17]. Es una base de datos relacional, esto significa que utiliza álgebra relacional, lo que se traduce a que los datos se almacenan en tablas con columnas y filas, y estas tablas se relacionan entre sí mediante el uso de llaves [16].

Según el sitio oficial de PostgreSQL [18], algunas de las primordiales funcionalidades con el que cuenta este gestor son las que se mencionan a continuación:

- Varios tipos de datos: básicos (entero, numérico, booleano), estructurados (fecha/hora, matriz), documento (JSON, XML), geométricos y personalizados.
- Integridad de los datos: valores únicos, valores no nulos, claves primarias y restricciones de exclusión.
- Concurrencia y rendimiento: aislamientos en las transacciones, transacciones anidadas, particionamientos de tablas y demás.
- Seguridad: funciones de autenticación de todo tipo, sistema robusto para la verificación de identidad y autenticaciones de múltiples factores haciendo uso de certificados.
- Confiabilidad y recuperación ante desastres: dispone de recuperaciones oportunas en un determinado punto en el tiempo, replicaciones de la BD.

- Extensibilidad: expresiones JSON/SQL, interfaz SQL y conexión a otras bases de datos, soporte multilingüe.
- Texto: operaciones de coincidencia de mayúsculas, minúsculas y no sensibles a la tilde, compatible con caracteres internacionales.

4.6.4. Modelo Vista Controlador

El patrón de diseño Modelo Vista Controlador (MVC), es la arquitectura del software que divide la lógica de negocio de la interfaz de usuario para desarrollarlos por separado [19], este patrón permite la evolución de manera aislada tanto de la interfaz como de la lógica de negocio, así mismo, extiende la reutilización y flexibilidad en el software.

El MVC en las aplicaciones web se define de la siguiente forma:

- **Modelo**
  Esta capa trata con datos o información que está almacenada en bases de datos o XML, tiene los mecanismos necesarios para acceder y actualizar la información, forma el núcleo de la aplicación. Se compone de un conjunto de clases que representan información del mundo real que el sistema necesita para procesar [20].
- **Controlador**
  Esta parte de código es responsable de recibir los eventos de entrada, recuperar datos de forma dinámica, responder a las solicitudes de los clientes y realizar las acciones adecuadas para representar el HTML. Esta capa actúa como enlace entre la vista y el modelo, su función es servir de enlace.
- **Vista**
  Es la página HTML que el navegador interpreta para mostrar al cliente. Esta capa es responsable de representar los datos del modelo, por lo que los utiliza, pero no los puede cambiar. Las vistan son las responsables de recibir la información procesada por el controlador o el modelo y mostrar al cliente [21].

4.6.4.1. Patrón Modelo Vista Controlador en Django

Django usa una arquitectura MVC propietaria que se implementa y diseña de tal manera que todas las partes están débilmente acopladas, se pueden realizar cambios en una parte específica de la aplicación sin tener que afectar al resto.

El controlador en las aplicaciones en Django es manejado internamente por el framework, por lo que se le da un mayor protagonismo a los modelos (Models), vistas (Views) y las plantillas (Templates), debido a esta acción el framework es conocido con el patrón Models Views

Templates (MTV). En la **figura 5** se puede visualizar el significado correspondiente a modelo vista controlador.

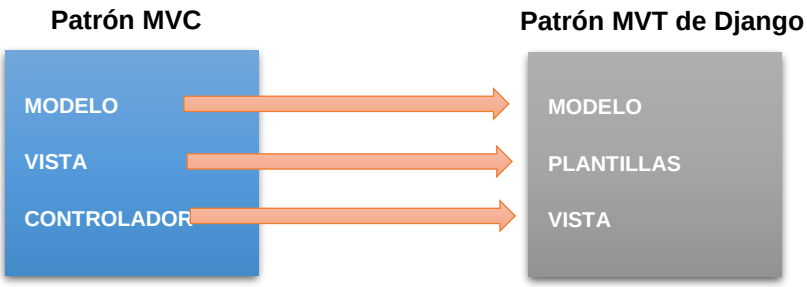


Figura 5. *Correspondencia entre MVC - MTV*

En Django se conoce a la vista como plantillas *"templates"* y al controlador como vistas *"Views".* Por consiguiente, el que decide cuales datos ve el usuario es la vista y el que decide como se visualizan esos datos se lo encarga a las plantillas. La vista es la capa que contiene la lógica del negocio, las reglas que acceden al modelo y delegan a la plantilla adecuada. Por último, el modelo no cambia con respecto al patrón MVC, igualmente en MTV modelo es la capa encargada de acceso a las bases de datos.

A continuación, se puede observar una gráfica de como interactúan cada uno de los componentes entre sí:

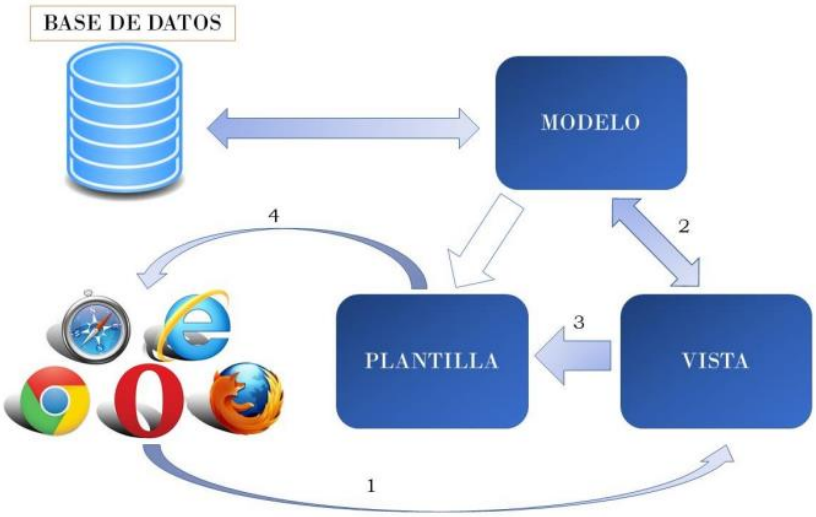


Figura 6. *Interacción entre los componentes MTV [19].*

1. Un cliente de un navegador web realiza una solicitud al servicio web de la aplicación y establece una conexión con la capa de visualización.
2. La capa de vista se comunica con el modelo para garantizar que se recuperen los datos solicitados por el cliente. La capa del modelo es responsable de pasar la

solicitud a la base de datos y devolverla a la vista, así como de procesar y filtrar el objeto o la información.

3. Después de completar el procesamiento de objetos, la capa de presentación envía la información u objetos preprocesados a la capa de plantilla.
4. Finalmente, la capa de plantilla se encarga de mostrar los datos solicitados por el cliente y poner la información procesada en la plantilla adecuada para que el navegador pueda procesarla correctamente.

Siguiendo estos cuatro pasos es como de manera general internamente funciona el framework Django con su lógica de patrón MTV, esto va desde la solicitud realizada por el cliente desde un navegador web, hasta la presentación de la información requerida en el mismo.

4.7. Herramientas de Prueba

Para realizar los test de la Aplicación Web se utilizó las herramientas Apache JMeter por parte de las pruebas de carga y estrés; y, por otro lado, para los test unitarios se hizo uso de la herramienta TestCase que ofrece Django.

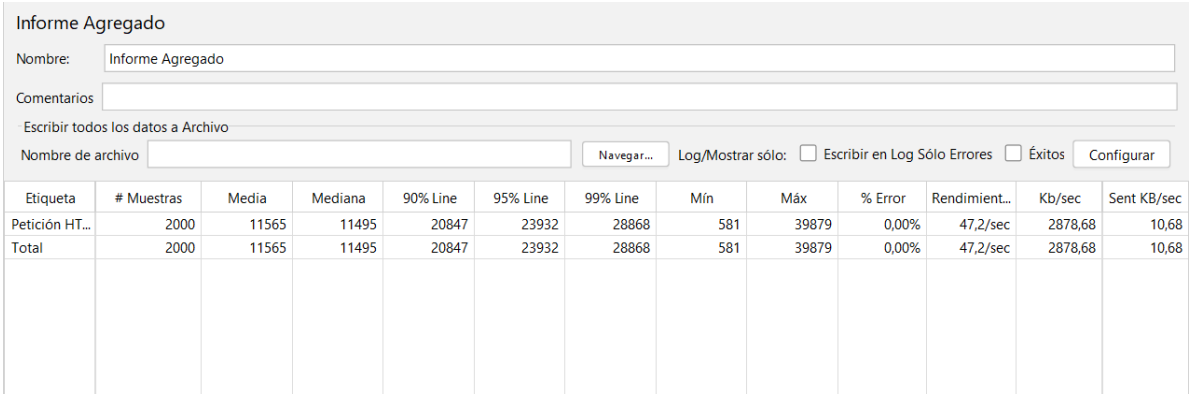
4.7.1. Apache JMeter

JMeter es una herramienta escrita en el lenguaje de programación Java para pruebas de carga y estrés de servidores como servicios web. Este procedimiento se utiliza para evaluar empíricamente el comportamiento y el desempeño. Inicialmente tenía como finalidad evaluar las aplicaciones web, pero actualmente se expande hacia otras funciones [22].

Entre algunas de las características más importantes y relevantes que tiene Apache JMeter se encuentran las descritas a continuación:

- Capacidad para cargar y probar el rendimiento de servidores, aplicaciones, protocolos.
- Interfaz fácil de usar para todas las funciones que permiten editar, resumir e incluso obtener resultados con gráficos.
- Extracción de información de los formatos de respuesta más utilizados en la actualidad: JSON, HTML, XML, entre muchos más.
- Cuenta con varias funcionalidades, novedades y muchas otras extensiones de todo tipo.

En la **figura 7** se puede visualizar un ejemplo de informe de resultados de peticiones realizadas a la red social Facebook, así mismo, en la **figura 8** se puede observar una gráfica con los resultados de la herramienta.



Etiqueta	# Muestras	Media	Mediana	90% Line	95% Line	99% Line	Mín	Máx	% Error	Rendiment...	Kb/sec	Sent KB/sec
Petición HT...	2000	11565	11495	20847	23932	28868	581	39879	0,00%	47,2/sec	2878,68	10,68
Total	2000	11565	11495	20847	23932	28868	581	39879	0,00%	47,2/sec	2878,68	10,68

***Figura 7**. Informe de resultados de JMeter.*

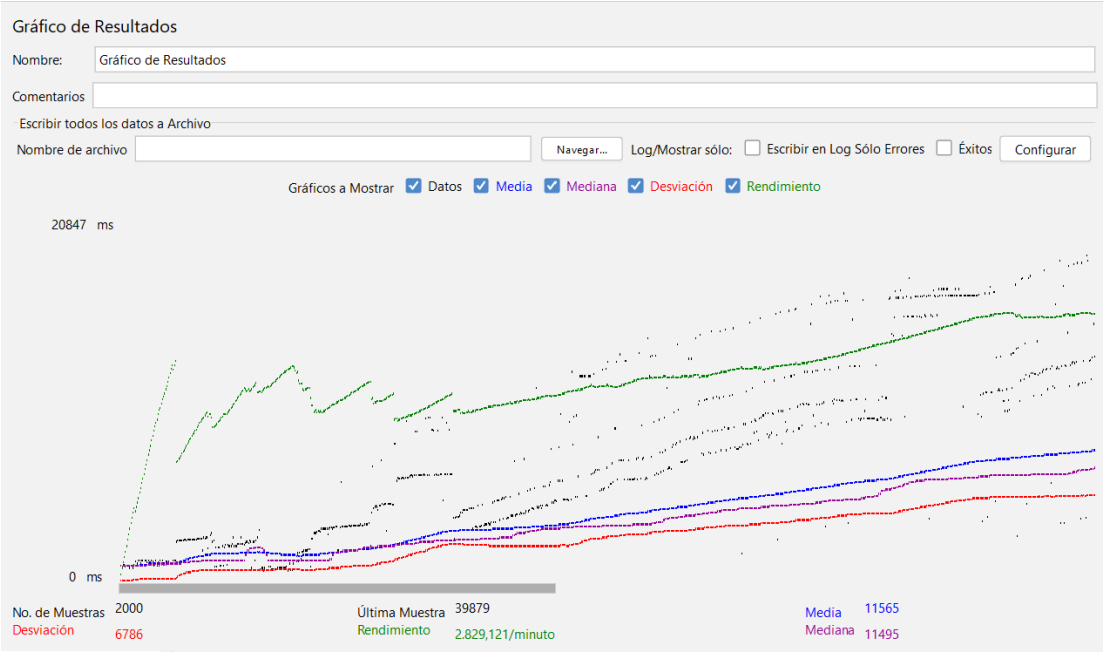


***Figura 8.** Gráfico de resultados de JMeter.*

La gráfica anterior muestra los resultados obtenidos al realizar la prueba de carga y estrés al sitio facebook.com, se puede visualizar que el número de muestras realizadas al sitio es de 2000, la última muestra fue realizada en el segundo 39, se demoró un promedio de 11 segundos en entregar cada una de las muestras, con un rango de 6 segundos por encima o debajo de la media y el rendimiento indica que puede el sitio soportar 2829 peticiones por minuto, todos estos valores son debido al gran número de muestras al que se sometió a la página.

4.7.2. TestCase

La Librería Django TestCase fue introducida por el framework Django para crear pruebas unitarias, esta librería se basa en unittest que es la librería modelo de Python para la creación de los test unitarios [23].

Lo que hace TestCase es probar si una determinada función funciona como se esperaba, para ello crea una base de datos ficticia si es necesario y hace uso de los assert para probar que las funcionalidades den valores true o false dependiendo de los resultados obtenidos. A continuación, se muestra la ejecución del test con un resultado negativo:

```
======================================================================
FAIL: test_was_published_recently_with_future_poll (polls.tests.PollMethodTests)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/dev/mysite/polls/tests.py", line 16, in test_was_published_recently_with_future_poll
    self.assertIs(future_poll.was_published_recently(), False)
AssertionError: True is not False

----------------------------------------------------------------------
Ran 1 test in 0.003s

FAILED (failures=1)
```

Figura 9. *Ejecución de test unitario con resultado negativo.*

5. Metodología

El tipo de investigación es aplicada debido a que permite realizar soluciones informáticas para resolver problemas que tenga la sociedad, en este caso, el Gobierno Autónomo Descentralizado Parroquial Rural de Milagro, ayudando a realizar de manera automatizada los cobros por el servicio de agua potable.

En esta sección del documento se detalla el proceso metodológico para culminar el desarrollo del presente Trabajo de Titulación, para esto se define el contexto que da a conocer donde se desarrolló, el proceso que define como se dio el cumplimiento de los objetivos planteados, los recursos utilizados, los participantes del proyecto y, por último, los materiales.

5.1. Contexto

El presente Trabajo de Titulación fue desarrollado en el Gobierno Autónomo Descentralizado Parroquial Rural de Milagro; el mismo que se enfocó para que esta institución pueda brindar un mejor servicio de cobro por el consumo de agua potable a cada uno de sus usuarios, con la finalidad de cumplir con el objetivo general planteado, el cual es: “Desarrollar una plataforma web como servicio para la automatización del cobro de consumo de agua potable en la parroquia Milagro del cantón Atahualpa”, donde se desglosa en tres objetivos específicos con sus respectivas actividades, mismas que se detallan a continuación.

5.2. Proceso

El proceso que ayudó al cumplimiento del objetivo general del presente Trabajo de Titulación en base a la metodología XP es el siguiente:

Fase 1. Especificar los requerimientos y diseñar los artefactos tecnológicos mediante la metodología de software eXtreme Programming.

Se realizó el levantamiento de requisitos funcionales y no funcionales por medio de la técnica de la entrevista, la cual fue necesaria para definir el alcance de la aplicación.

- Se elaboró el documento de especificación de requisitos de software utilizando el estándar IEEE 830.
- Se desarrolló la fase de planificación y diseño de la metodología XP mediante el desarrollo del prototipado básico de la aplicación y su arquitectura fundamentándose en el modelo arquitectónico 4+1 que describe la arquitectura de la aplicación basándose en vista lógica, vista de procesos, vista de despliegue, vista física y vista de escenarios.

Fase 2. Construir la aplicación web mediante el framework Django y la librería Bootstrap.

- Se desarrolló la fase de codificación de cada una de las funcionalidades definidas de la aplicación para el cobro por el servicio de agua potable en el framework Django.

Fase 3. Probar y validar la aplicación web mediante pruebas unitarias y pruebas de estrés en ambiente simulado.

- Se realizó la codificación de los test unitarios con la ayuda de la herramienta TestCase de Django para cada uno de los modelos de datos de la aplicación; se desplegó la aplicación en los servidores web de render y; se solucionó los fallos suscitados en el momento de despliegue de la aplicación. Así mismo, se ejecutó las pruebas de carga y estrés con la ayuda de la herramienta Apache JMeter.
- Se efectuó las primeras visualizaciones de la aplicación desplegada con el cliente y el docente tutor, a partir de ello se realizó la codificación agregando la funcionalidad de recuperación de contraseña de ingreso al sistema.
- Se realizó la validación de la aplicación web a partir de una entrevista al cliente con los requisitos funcionales definidos en un inicio.

5.3. Recursos

Para el cumplimiento de las fases anteriormente mencionadas, se empleó una serie de recursos que se muestran a continuación:

5.3.1. Recursos científicos

5.3.1.1. Experimento

Para la aplicación de este método se siguió el proceso experimental, el cual indica cada una de las actividades a realizarse, así como también, las entradas y salidas de cada actividad. Este método fue usado en el tercer objetivo, para la realización de las pruebas unitarias.

5.3.2. Recursos técnicos

Para el desarrollo del presente Trabajo de Titulación se hizo uso de la metodología de desarrollo Extreme-Programming (XP), la misma que se definió desde un inicio en el alcance debido a las múltiples prestaciones que nos brinda y por ser la que de mejor manera se acopla al proyecto.

5.3.2.1. Entrevista

Permitió obtener y definir los requerimientos de la aplicación web, de igual forma permitió evaluar la funcionalidad y la aceptación del software por parte del cliente encargado en el GADPR de Milagro.

5.3.2.2. Investigación

Permitió definir los conceptos básicos importantes para el desarrollo del presente Trabajo de Titulación, en base a ello se logró indagar sobre aspectos realmente necesarios a lo largo de todo el proyecto.

5.4. Participantes

El presente Trabajo de Titulación se desarrolló por los siguientes participantes:

- Edhisson Alexis Sanmartín Freire como investigador del presente TT, empezando desde la formulación del proyecto, hasta la culminación del mismo cumpliendo cada uno de los objetivos planteados.
- Ing. Roberth Figueroa Díaz como director del TT, aportando desde la formulación del anteproyecto, orientando en su desarrollo hasta la finalización con el cumplimiento de cada uno de los objetivos.
- Franco Sanmartín Gómez como cliente y colaborador para el desarrollo de la aplicación, ente fundamental desde el levantamiento de los requerimientos de la aplicación hasta su aceptación.

5.5. Materiales

Los materiales utilizados para el desarrollo del presente Trabajo de Titulación se detallan en la **tabla 3**.

Tabla 3. *Materiales utilizados en el trabajo de titulación.*

Recursos Hardware	
Nombre	**Descripción**
Laptop Lenovo	Dispositivo utilizado para el desarrollo del TT.
Recursos Software	
Nombre	**Descripción**
Draw.io	Software gratuito para elaboración de los diagramas.
GitHub	Plataforma web para el alojamiento del código fuente de la aplicación.
Visual Studio Code	Editor de código, herramienta indispensable para el desarrollo del proyecto.

Zoom	Plataforma de comunicación utilizada para las reuniones y revisiones del TT.
Pencil	Software utilizado para el prototipado de la aplicación para el cobro de servicios de agua.
JMeter	Herramienta gratuita para realizar las pruebas de carga y estrés de la aplicación.
Tecnología	
Nombre	**Descripción**
Django	Framework utilizado para el desarrollo de la aplicación web usando el lenguaje de programación Python.
Bootstrap	Herramienta de código abierto utilizada para diseñar la parte visual de la aplicación.

6. Resultados

En este apartado se describe en detalle el cumplimiento de los objetivos planteados en el Trabajo de Titulación.

6.1. Objetivo 1: Especificar los requerimientos y diseñar los artefactos tecnológicos mediante la metodología de software eXtreme Programming.

En el presente objetivo se realizó una entrevista al presidente de la organización y se tuvo varias reuniones para conocer en detalle el proceso del cobro por el consumo de agua en la Parroquia Milagro, esta interacción con el cliente ayudó y fue fundamental para obtener la información necesaria para especificar los requerimientos y diseñar los artefactos tecnológicos requeridos en el presente trabajo de titulación, ver Anexo 1**: Entrevista – Establecimiento de Requerimientos.**

A continuación, se describen cada una de las fases que se desarrollaron para el cumplimiento del presente objetivo:

6.1.1. Especificar los requerimientos de la aplicación a través del estándar IEEE 830.

Se utilizó el estándar IEEE 830 para realizar un correcto levantamiento y especificación de requerimientos funcionales y no funcionales en base a la entrevista y las reuniones con el cliente, requerimientos que han sido fundamentales y han permitido el correcto desarrollo de la aplicación web. El estándar IEEE 830 tiene como propósito documentar los acuerdos entre el cliente y los desarrolladores, para ver en detalle Anexo 2: **Especificación de requisitos de software.**

- **Requerimientos funcionales**

En la **tabla 4** se muestran los requerimientos funcionales del software como servicio.

Tabla 4. *Requerimientos funcionales de la aplicación.*

Requerimientos funcionales		
Número	**Nombre**	**Breve descripción**
RF001	Inicio de sesión	El Software contará con un inicio de sesión para validar credenciales del administrador.
RF002	Ingreso de usuarios	Se permitirá a través del administrador ingresar usuarios del servicio.
RF003	Cobros del servicio	El administrador podrá realizar los cobros del servicio de cada uno de los usuarios antes ingresados y su respectivo consumo.

RF004	Ingreso de cifras consumidas	Cada mes el administrador tendrá la opción de subir las cifras de consumo.
RF005	Comprobante de pago	Se generará un comprobante de pago que será impreso o enviado al correo del usuario.
RF006	Configuraciones del sistema	Se podrá cambiar la base mínima de consumo, así como la tarifa extra y demás.
RF007	Notificar deuda	El administrador podrá enviar correo de aviso a los usuarios que tengan deudas pendientes.

- **Requerimientos no funcionales**

En la **tabla 5** se muestran los requerimientos no funcionales de la aplicación.

***Tabla 5**. Requerimientos no funcionales de la aplicación.*

Requerimientos no funcionales		
Número	**Nombre**	**Breve descripción**
RN001	Compatibilidad	El sistema permitirá poder ser usado en los distintos entornos de navegación web.
RN002	Desempeño	Se garantizará a los usuarios la realización de las tareas de una manera eficaz.
RN003	Disponibilidad	El sistema estará disponible siempre que se requiera de sus servicios.
RN004	Fiabilidad	Se garantizará a los usuarios seguridad en cuanto a la información que se provee al sistema.
RN005	Tiempo de respuesta	Se responderá de una manera rápida y correcta en el menor tiempo posible a las peticiones realizadas.
RN006	Usabilidad	Será de fácil manejo, contará con una interfaz de usuario sencilla y amigable.

Historias de usuario

La metodología XP recomienda utilizar la técnica de Historias de Usuario para especificar los requisitos de la aplicación y obtener una breve descripción de cada uno de los requisitos. La HU deber ser clara, precisa y estar limitada, para que el programador pueda saber claramente qué función debe ejecutarse lo más rápido posible [24]. La metodología XP propone definir las HU a través de tablas, el grafico para la definición que se toma como referencia es el que presenta [25] y se describe a continuación:

***Tabla 6.** Esquema de Historias de usuario.*

Historia de Usuario	
Número:	**Nombre de Historia de Usuario:**
Usuario:	
Modificación de Historia Número:	**Interacción Asignada:**
Prioridad en el negocio (Alta/Media/Baja):	**Puntos estimados:**
Riesgo en el desarrollo (Alta/Media/Baja):	**Puntos reales:**

Descripción:
Observaciones:
Criterios de aceptación:

Descripción de cada campo de las Historias de Usuario:

- **Número:** Número asignado a cada una de las Historias de Usuario.
- **Nombre de Historia de usuario:** un nombre breve que describa la HU, asignad a cada una de ellas.
- **Usuario:** responsable del proceso o la actividad a realizar.
- **Modificación de Historia de usuario:** Prioridad de la HU, se establece de acuerdo con la prioridad de la actividad.
- **Riesgo de desarrollo:** complejidad del desarrollo de la actividad.
- **Interacción asignada:** número de interacciones de la actividad.
- **Puntos estimados:** Se refiere a la cantidad de días de desarrollo.
- **Puntos reales:** Los puntos reales que se utilizaron en la actividad.
- **Descripción:** Información detallada de las actividades.
- **Observaciones:** aspectos relevantes de cada HU.
- **Criterios de aceptación:** criterios que debe cumplir una actividad para ser aceptada.

A continuación se presentan las Historias de Usuario de la aplicación:

Tabla 7. *Historia de usuario - Iniciar sesión.*

Historia de Usuario	
Número: 001	**Nombre de Historia de Usuario:** Inicio de sesión
Usuario: Administrador	
Modificación de Historia Número: 1	**Interacción Asignada:** 1
Prioridad en el negocio (Alta/Media/Baja): Alta	**Puntos estimados:** 2
Riesgo en el desarrollo (Alta/Media/Baja): Media	**Puntos reales:** 2
Descripción: La aplicación web debe permitir al administrador iniciar sesión para acceder al sistema a través de un formulario en la página principal, este acceso es exclusivo para los administradores.	
Observaciones:	
Criterios de aceptación: 1. Mostrar un formulario de acceso en la página principal del sistema. 2. Comprobar que el usuario y la contraseña coincidan con los registros de la base de datos. 3. Si las credenciales no coinciden con los registros mostrar un mensaje diciendo que los datos ingresados son incorrectos. 4. Dar acceso dependiendo de si el administrador tiene privilegios o no, si el administrador no cuenta con privilegios no podrá acceder al panel administrativo.	

Tabla 8. *Historia de usuario - Ingresar usuarios.*

Historia de Usuario	
Número: 002	**Nombre de Historia de Usuario:** Ingreso de usuarios
Usuario: Administrador	
Modificación de Historia Número: 1	**Interacción Asignada:** 1
Prioridad en el negocio (Alta/Media/Baja): Alta	**Puntos estimados:** 4
Riesgo en el desarrollo (Alta/Media/Baja): Alta	**Puntos reales:** 5
Descripción: El administrador podrá registrar usuarios del servicio, así como también modificarlos y eliminarlos.	
Observaciones:	
Criterios de aceptación: 1. Mostrar el formulario de registro de clientes a través de la opción nuevo. 2. Validar que los campos necesarios para el registro sean ingresados. 3. Mostrar un mensaje que indique que campos obligatorios para el registro están vacíos. 4. Controlar que exista un único número de medidor por cliente. 5. Mostrar un mensaje que indique que el número de medidor ya ha sido registrado. 6. Crear un cliente con los datos ingresados por el administrador. 7. Mostrar un mensaje indicando que el cliente se ha creado correctamente. 8. Editar campos de clientes ingresados. 9. Guardar correctamente los campos actualizados por el administrador. 10. Mostrar un mensaje indicando que el cliente se ha actualizado correctamente. 11. Eliminar clientes que el administrador quiera. 12. Mostrar un mensaje para confirmar la eliminación del cliente alertando cada una de las consecuencias de esta acción. 13. Mostrar un mensaje indicando que el cliente se ha eliminado de una forma correcta.	

Tabla 9. *Historia de usuario - Cobros del servicio.*

Historia de Usuario	
Número: 003	**Nombre de Historia de Usuario:** Cobros del servicio
Usuario: Administrador	
Modificación de Historia Número: 1	**Interacción Asignada:** 1
Prioridad en el negocio (Alta/Media/Baja): Alta	**Puntos estimados:** 5
Riesgo en el desarrollo (Alta/Media/Baja): Alta	**Puntos reales:** 5
Descripción: El administrador podrá realizar los cobros del servicio de cada uno de los usuarios dependiendo de las cifras consumidas.	
Observaciones:	
Criterios de aceptación: 1. Calcular los todos los valores que el cliente posea. 2. Comprobar que existan valores a pagar. 3. Mostrar un mensaje que indique que el cliente no cuenta con valores pendientes junto con un registro de todos los pagos realizados. 4. Realizar cálculo de consumo del mes restando la cifra ingresada actualmente de la cifra del mes anterior. 5. Calcular valores dependiendo de los valores base que posee esa cifra, la cifra del	

mes anterior puede tener unos valores base diferentes a los valores base de la cifra actual. 6. Agregar valores por mora a las cifras que no han sido pagadas en el tiempo dado. 7. Agregar valores adicionales a cada metro que exceda la cifra base en cada uno de los diferentes meses que tenga deudas pendientes. 8. Dar valores base a clientes que no excedan la cifra base. 9. Mostrar los meses pendientes que el cliente tenga.

Tabla 10. *Historia de Usuario - Ingreso de cifras.*

Historia de Usuario	
Número: 004	**Nombre de Historia de Usuario:** Ingreso de cifras consumidas
Usuario: Administrador	
Modificación de Historia Número: 1	**Interacción Asignada:** 1
Prioridad en el negocio (Alta/Media/Baja): Alta	**Puntos estimados:** 5
Riesgo en el desarrollo (Alta/Media/Baja): Alta	**Puntos reales:** 5
Descripción: El administrador podrá realizar los cobros del servicio de cada uno de los usuarios dependiendo de las cifras consumidas.	
Observaciones:	
Criterios de aceptación: 1. Cada mes el administrador podrá ingresar la cifra del cliente. 2. La cifra solo se puede ingresar una vez por mes. 3. Listar todas las cifras ingresadas que tiene el cliente junto con su estado. 4. Verificar que la cifra que se ingresa no sea mayor a la cifra del mes anterior. 5. Mostrar un mensaje que indique que la cifra ingresada es inferior a la cifra del mes anterior. 6. Mostrar un mensaje indicando que la cifra se ha ingresado correctamente. 7. Las cifras ingresadas solo se podrán editar si aún no han sido pagadas por el cliente. 8. Indicar un mensaje que indique que el campo a ingresar está vacío.	

Tabla 11. *Historia de Usuario - Comprobante de pago.*

Historia de Usuario	
Número: 005	**Nombre de Historia de Usuario:** comprobante de pago
Usuario: Administrador	
Modificación de Historia Número: 1	**Interacción Asignada:** 1
Prioridad en el negocio (Alta/Media/Baja): Alta	**Puntos estimados:** 4
Riesgo en el desarrollo (Alta/Media/Baja): Media	**Puntos reales:** 2
Descripción: Una vez realizado el pago, el sistema generará una comprobante de pago que podrá ser impreso y se enviará al correo del cliente.	
Observaciones:	
Criterios de aceptación: 1. Registrar el pago en la base de datos y actualizar el estado de las cifras a pagadas. 2. Generar un pdf en una nueva pestaña con el recibo de pago del cliente. 3. Enviar el recibo de pago al correo electrónico del cliente. 4. Recargar la página donde se muestra los valores a cancelar, mostrar un mensaje indicando que el pago ha sido realizado correctamente y enviado al correo del	

cliente.
5. Mostrar todos los pagos realizados por el cliente hasta la fecha.

Tabla 12. *Configuraciones del sistema.*

Historia de Usuario	
Número: 006	**Nombre de Historia de Usuario:** Valores base (configuraciones del sistema)
Usuario: Administrador	
Modificación de Historia Número: 1	**Interacción Asignada:** 1
Prioridad en el negocio (Alta/Media/Baja): Alta	**Puntos estimados:** 4
Riesgo en el desarrollo (Alta/Media/Baja): Baja	**Puntos reales:** 2
Descripción: El administrador podrá realizar las configuraciones del sistema agregado nuevos valores base del servicio.	
Observaciones:	
Criterios de aceptación: 1. Mostrar un formulario para el ingreso de nuevos valores base. 2. Los valores base no se pueden editar ni eliminar. 3. Mostrar un mensaje indicando que, si se agregan nuevos valores base, antes se debe estar seguro de haber ingresado todas las cifras del mes actual. 4. Mostrar un mensaje indicando que los nuevos valores base han sido ingresado correctamente.	

Tabla 13. *Historia de Usuario – Notificar deudas.*

Historia de Usuario	
Número: 007	**Nombre de Historia de Usuario:** notificar deudas
Usuario: Administrador	
Modificación de Historia Número: 1	**Interacción Asignada:** 1
Prioridad en el negocio (Alta/Media/Baja): Baja	**Puntos estimados:** 3
Riesgo en el desarrollo (Alta/Media/Baja): Baja	**Puntos reales:** 3
Descripción: El administrador del sistema podrá notificar a los clientes las deudas pendientes por el servicio, esto una vez por mes.	
Observaciones:	
Criterios de aceptación: 1. Mostrar un listado de todos los usuarios con deudas pendientes. 2. Permitir enviar una sola notificación al correo electrónico del usuario por mes. 3. Dar la opción de notificar a un único usuario. 4. Dar la opción de notificar a todos los usuarios. 5. Mostrar un mensaje que indique que se ha notificado al usuario seleccionado. 6. Mostrar un mensaje que indique que todos los usuarios con deudas pendientes han sido notificados.	

Estimación de Historias de Usuario

En la **tabla 14** se presenta las historias de usuario con el tiempo estimado de cada una de ellas, tomando como referencia una semana de 5 días con 8 horas al día:

Tabla 14. *Tiempo estimado de historias de usuario.*

Nro.	Nombre de Historia de Usuario	Tiempo estimado		
		Semanas	**Días**	**Horas**
001	Inicio de sesión	0.8	4	32
002	Ingreso de usuarios	1.2	6	48
003	Cobro del servicio	2	10	80
004	Ingreso de cifras consumidas	1.2	6	48
005	Comprobante de pago	1	5	40
006	Configuraciones del sistema	2	10	80
007	Notificar deudas	1	5	40
	Tiempo estimado total	9.2	46	368

6.1.2. Diseñar los diagramas básicos y el prototipado de la aplicación junto con su validación.

Para la ejecución de la presente fase se desarrolló el prototipado básico de la aplicación y su arquitectura fundamentándose en el modelo arquitectónico 4+1, estos elementos son una parte y guía fundamental dentro del desarrollo y la iteración con el cliente en base a las funcionalidades que el mismo desea en el software.

Diseño general de la aplicación

En la **figura 10** se puede observar una vista general del sistema de cobro del servicio de agua potable, mismo que se desarrolló en el framework Django y en la parte visual con la librería Bootstrap, aplicando el patrón de diseño Modelo Vista Controlador ofrecido por Django, el sistema se conecta con la base de datos Postgresql con la ayuda de las librerías de conexión ofrecida por el mismo framework.

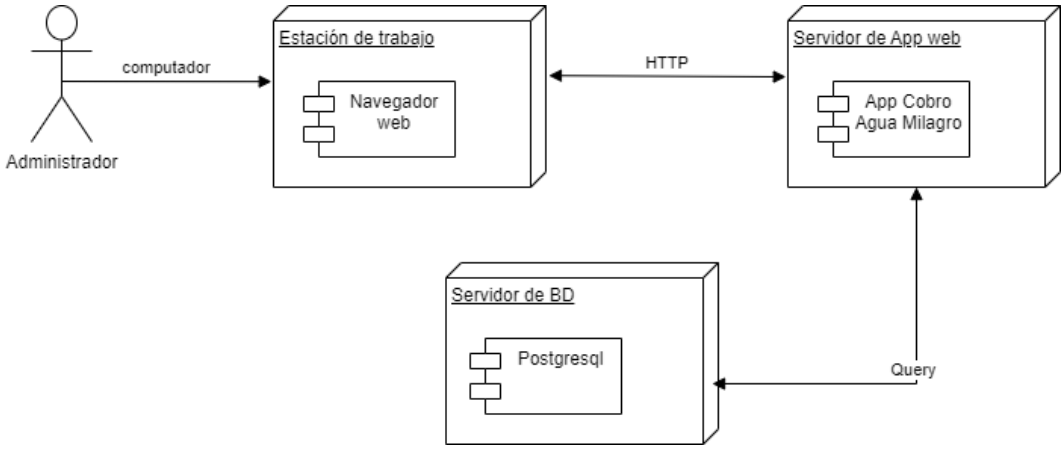


Figura 10. *Vista general del sistema.*

Arquitectura de la aplicación

En la **tabla 15** se puede observar cómo se encuentra definida la arquitectura de la aplicación, para ver esta información de forma más detallada ver el anexo 3: **Arquitectura del software como servicio para la automatización del cobro de consumo de agua.**

Tabla 15. *Arquitectura de la aplicación (4+1).*

Vista	Elemento modelado	Descripción
Vista de escenarios	Casos de uso	Muestra las actividades que debe realizar el administrador para llevar a cabo el proceso de cobro del servicio de agua.
Vista lógica	Diagrama de clases	Muestras las funcionalidades de la aplicación.
Vista física	Diagrama de despliegue	Muestra los componentes físicos del sistema
Vista de despliegue	Diagrama de componentes	Describe los diferentes componentes para que el desarrollador tenga una clara comprensión del sistema.
Vista de procesos	Diagrama de actividad	Describe los procesos del sistema.

Vista de escenarios

En la **figura 11** se puede observar el diagrama de casos de uso del sistema de cobros de agua potable, en este interviene el administrador como actor directo del sistema y el usuario del servicio como actor indirecto, las actividades que realiza el administrador del sistema son los siguientes:

- Ingresar al sistema
- CRUD de usuarios del servicio
- Buscar usuarios

- Ingresar cifras consumidas
- Realizar cobros del servicio
- Emitir recibos de pago
- Notificar deudas a los usuarios
- Configuraciones del sistema.

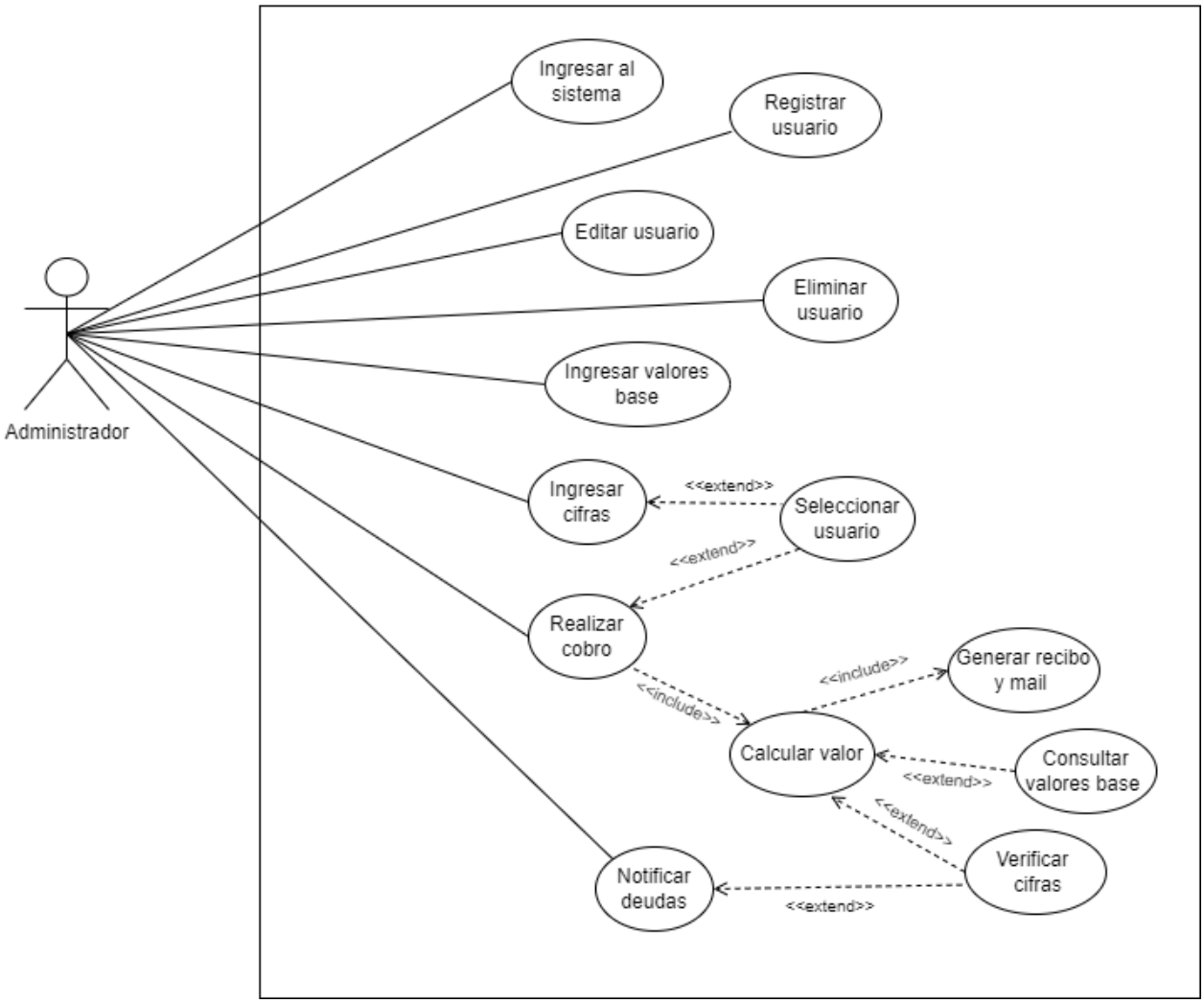


Figura 11. *Diagrama de casos de uso de la aplicación.*

Vista lógica

En la **figura 12** se visualiza el diagrama de clases, el mismo tiene las funcionalidades y el comportamiento de cada una de las clases.

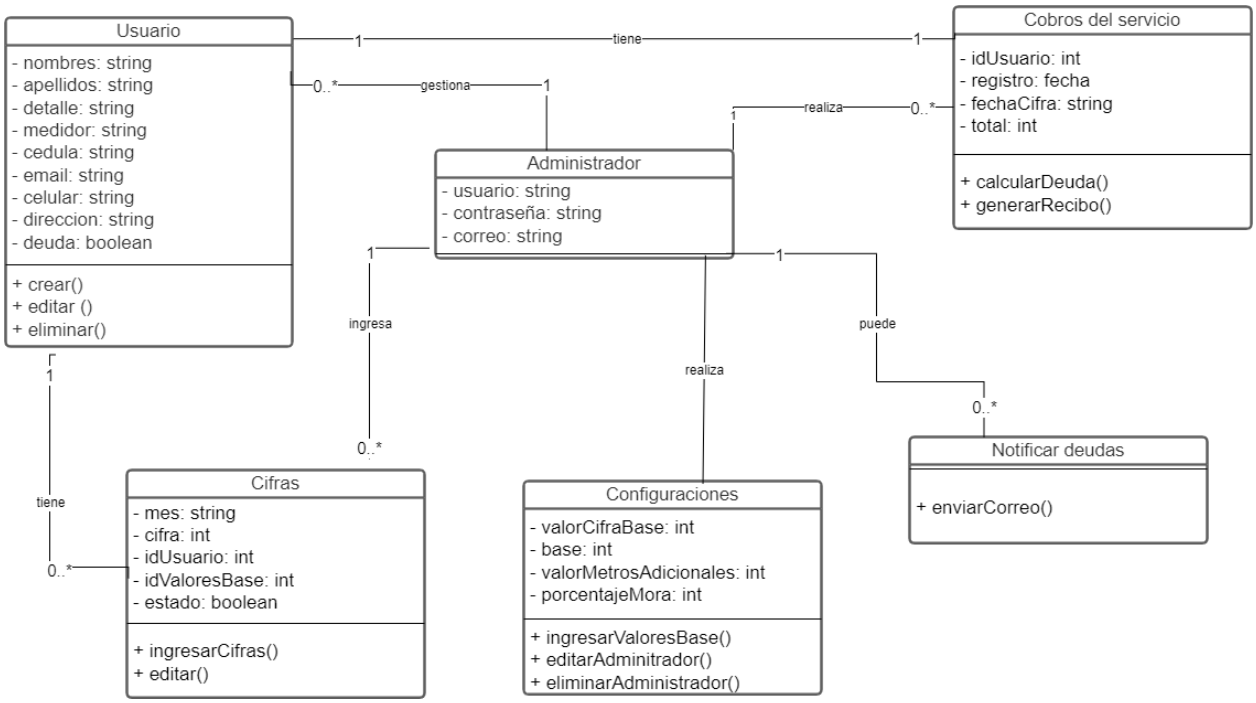


Figura 12. *Diagrama de clases de la aplicación.*

Vista de procesos

A continuación se presenta los diagramas de actividades, en este diagrama interactúa el usuario con el sistema, en la **figura 13** que se observa a continuación se encuentra el diagrama de actividades para realizar un cobro por el servicio, especificando cada uno de los pasos que debe de seguir la aplicación para su correcto funcionamiento. Para ver en detalle cada diagrama de actividades de los demás procesos ver el anexo 3: **Arquitectura del software como servicio para la automatización del cobro de consumo de agua.**

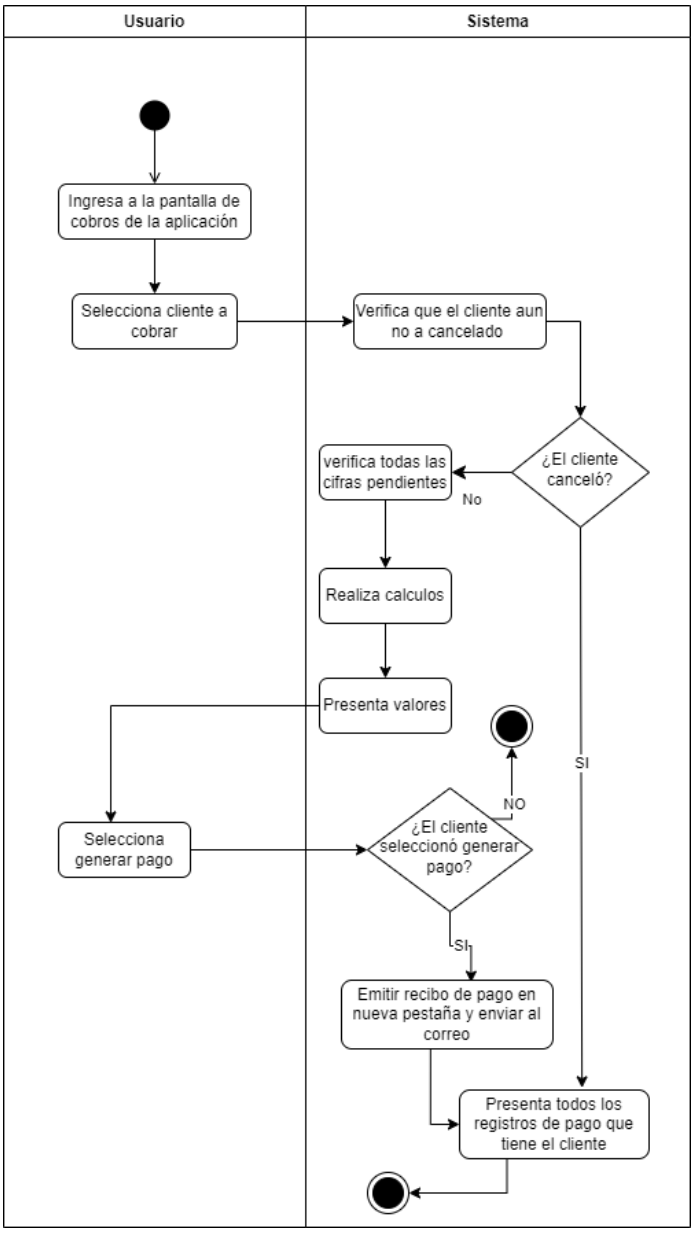


Figura 13. *Diagrama de actividades para realizar un cobro en la aplicación.*

Vista de despliegue

En la **figura 14** se puede visualizar como se organiza cada uno de los componentes de la aplicación, la interacción entre el cliente, el servidor de aplicaciones y el servidor de bases de datos.

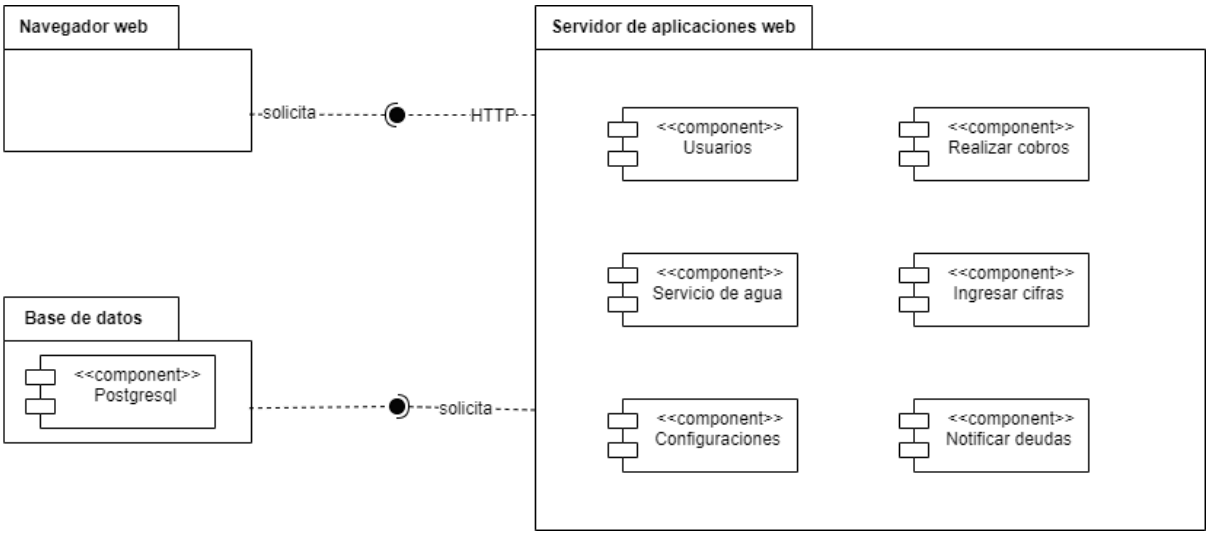


Figura 14. *Diagrama de componentes.*

Vista física

El diagrama de despliegue que se observa en la **figura 15** consta del cliente web, el navegador que se comunica con el servidor de aplicaciones web que contiene a la aplicación de cobros de servicio de agua.

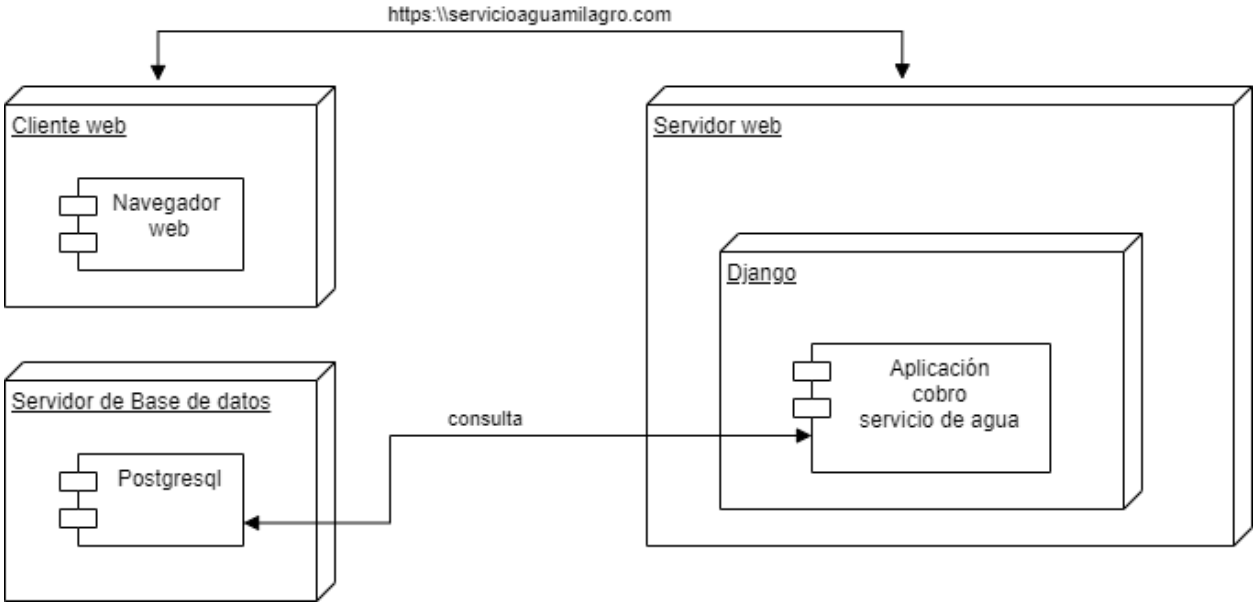


Figura 15. *Diagrama de despliegue.*

Arquitectura de la aplicación

La arquitectura utilizada en la Aplicación Web es cliente/servidor donde los demandantes llamados clientes realizan peticiones a un proveedor de recursos llamado servidor que a su vez realiza consultas a una Base de Datos, ver **figura 16**.

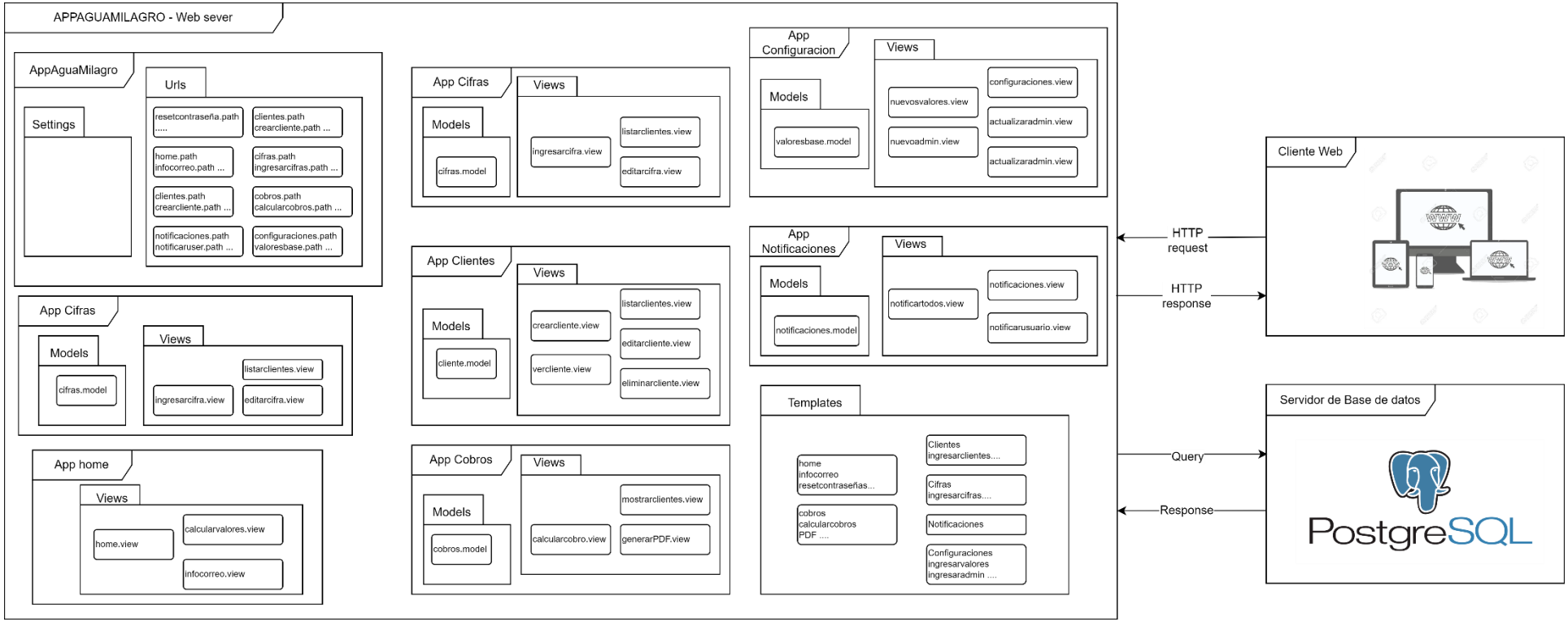


Figura 16. *Arquitectura de la aplicación.*

Prototipado de la interfaz de la aplicación web para el cobro por consumo de agua

Se realizó el prototipado de las principales interfaces de la aplicación con la ayuda del software de diseño de interfaces pencil, el diseño ayudó a tener una idea global de como quedaría el software una vez desarrollado. En la **figura 17** se tiene la interfaz de inicio de la aplicación una vez el administrador ha iniciado sesión, en la **figura 18** se puede observar la interfaz cuando el administrador quiera generar un cobro por el servicio. Para ver en detalle cada una de las interfaces ver el anexo 4: **Prototipos del software como servicio para el cobro por el consumo de agua.**

Figura 17. *Pantalla inicial de la aplicación web.*

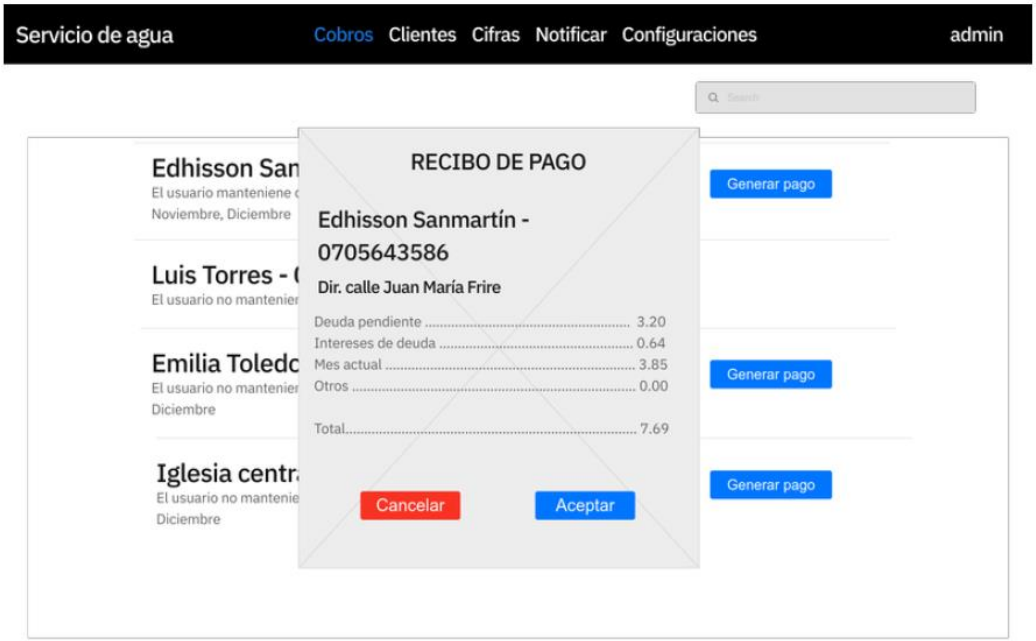


Figura 18. *Pantalla de cobros del servicio y recibo de pago.*

6.2. Objetivo 2: Construir la aplicación web mediante el framework Django y la librería Bootstrap.

En esta sección se desarrolló la codificación de la aplicación para el cobro del servicio de Agua – Milagro utilizando el framework Django basado en el lenguaje de programación Python, el desarrollo de la aplicación web se realizó haciendo uso del patrón de diseño Modelo Vista Controlador (MVC) aplicada a lógica de Django, en la **figura 19** se puede visualizar la estructura del proyecto, de igual forma en la **figura 20** se muestra el patrón MVC utilizado.

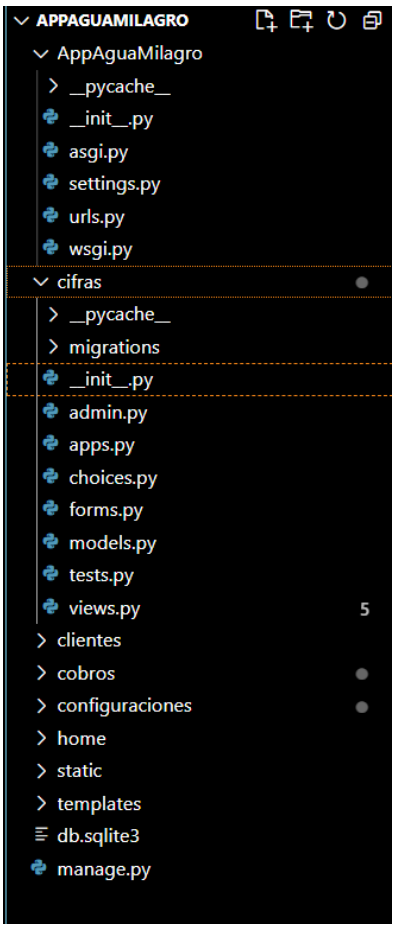


Figura 19. *Estructura de la aplicación web.*

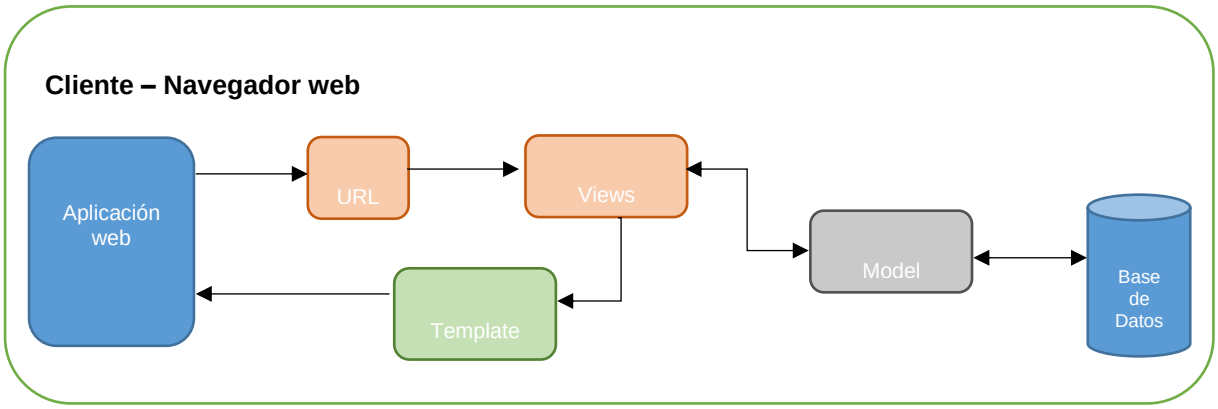


Figura 20. *MVC aplicado a la lógica de Django.*

- En Django las URL se encargan de redirigir hacia cada una de las Views dependiendo de la solicitud requerida por el usuario.

La **figura 21** muestra cada una de las URL para la aplicación web, las mismas se encargan de redirigir hacia cada view de cada aplicación la solicitud realizada (Django define como aplicación a cada módulo interno del proyecto).

```
from django.contrib import admin
from django.urls import path
from home.views import home, cerrar_sesion, inicio
from clientes.views import crear_cliente, listar_clientes, editar_cliente, actualizar_cliente, ver_cliente, eliminar_cliente,
from cifras.views import seleccion_clientes,ingresar_cifra, editar_cifras, actualizar_cifra
from cobros.views import mostrar_clientes, calcular_cobro

urlpatterns = [
    path('admin/', admin.site.urls),
    #Para la app home
    path('', home, name="home"),
    path('logout/', cerrar_sesion, name="logout"),
    path('inicio/', inicio, name="inicio"),
    #Para la app clientes
    path('crearcliente/', crear_cliente, name="crearcliente"),
    path('clientes/', listar_clientes, name="listarclientes"),
    path('editarcliente/<int:id>',editar_cliente),
    path('actualizarcliente/<int:id>',actualizar_cliente),
    path('vercliente/<int:id>',ver_cliente),
    path('eliminarcliente/<int:id>',eliminar_cliente),
    path('confirmareliminarcliente/<int:id>',confirmar_eliminar_cliente),
    #Para la app cifras
    path('cifras/', seleccion_clientes),
    path('ingresarcifra/<int:id>',ingresar_cifra),
    path('editarcifra/<int:id>',editar_cifras),
    path('actualizarcifra/<int:id>',actualizar_cifra),
    #Para la app cobros
    path('cobros/',mostrar_clientes),
    path('calcularcobro/<int:id>',calcular_cobro),
]
```

Figura 21. *Urls de la aplicación web de cobros del servicio de agua.*

- Las Views contienen cada una de las clases con la lógica de determinada parte de la aplicación, estas clases son accedidas mediante las urls.

La **figura 22** muestra el código de las Views, las Views dentro de la lógica de Django vendrían a ser el controlador, en este apartado se procesa los datos de la solicitud realizada por el usuario, procesados los datos se recupera y se envía la información hacia la plantilla para mostrarse en el navegador.

```
def editar_cliente(request, id):
    cliente = Cliente.objects.get(id=id)
    f = formulario_cliente(initial={'nombres':cliente.nombres,
                                    'apellidos':cliente.apellidos,
                                    'detalle':cliente.detalle,
                                    'medidor':cliente.medidor,
                                    'cedula':cliente.cedula,
                                    'email':cliente.email,
                                    'direccion':cliente.direccion,
                                    'celular':cliente.celular})
    context = {
        "form":f,
        "cliente":cliente,
    }
    return render(request, "editarCliente.html", context)

def actualizar_cliente(request, id):
    cliente = Cliente.objects.get(pk=id)
    form = formulario_cliente(request.POST, instance=cliente)
    if form.is_valid():
        form.save()
        messages.success(request, "Usuario '"+ cliente.nombres + " " + cliente.apellidos + " medidor numero: "+ cliente.medidor
        return redirect('/clientes')
    else:
        messages.success(request, "A ocurrido un error! Puede que el número de medidor ya este registrado.")
    context = {
        "form":form,
        "cliente":cliente,
    }
    return render(request, "editarCliente.html", context)
```

Figura 22. *Views de la app clientes (app interna de la aplicación web).*

- Las funciones editar y actualizar cliente son pertenecientes a la app cliente, se procesan los datos y se redirige hacia la plantilla solicitada.

La **figura 23** muestra el formulario de cliente, estos se encargan de mostrar la información a los clientes a través de las plantillas.

```
class formulario_cliente(forms.ModelForm):
    class Meta:
        model = Cliente
        fields=["nombres", "apellidos","detalle","medidor","cedula","email","direccion","celular","deuda","ultima_cifra"]
```

Figura 23. *Formulario de cliente.*

Para la generación de archivos con formato pdf en Django se hizo uso de la librería render_to_pdf, esta poderosa librería genera el archivo a partir de una plantilla a la cual se le incrusta un diccionario con variables. Este pdf una vez generado se lo muestra en una pestaña nueva del navegador con la opción de guardarlo o imprimirlo. Así mismo, se envía el recibo de pago al correo del cliente haciendo uso de la librería EmailMultiAlternatives de Django. En la **figura 24** se puede visualizar de mejor manera lo anteriormente mencionado:

```
template_name="recibo.html"
pdf = render_to_pdf(template_name,context)
#enviar correo electronico
template2 = get_template('recibo.html')
content = template2.render(context)
mail = EmailMultiAlternatives(
subject='Recibo de Pago del Servicio de Agua del GADPR Milagro - '+cifra.mes,
body='',
from_email=settings.EMAIL_HOST_USER,
to=[
    cliente.email
],
cc=[])
mail.attach_alternative(content,'text/html')
mail.send(fail_silently=False)
```

Figura 24. *Código para generar recibo de pago en formato PDF y envío de Mails.*

Proceso de ingreso de cifras de los clientes

A continuación, se describe brevemente el proceso junto con el código fuente y la lógica de negocio que sigue la aplicación web para realizar el registro de las cifras consumidas por los usuarios mes a mes. El proceso empieza con el listado de todos los clientes, cada uno de los clientes cuenta con una acción entre las cuales está: ingresar nueva cifra o ver cifras ingresadas, en la **figura 25** se observa los clientes y sus acciones:

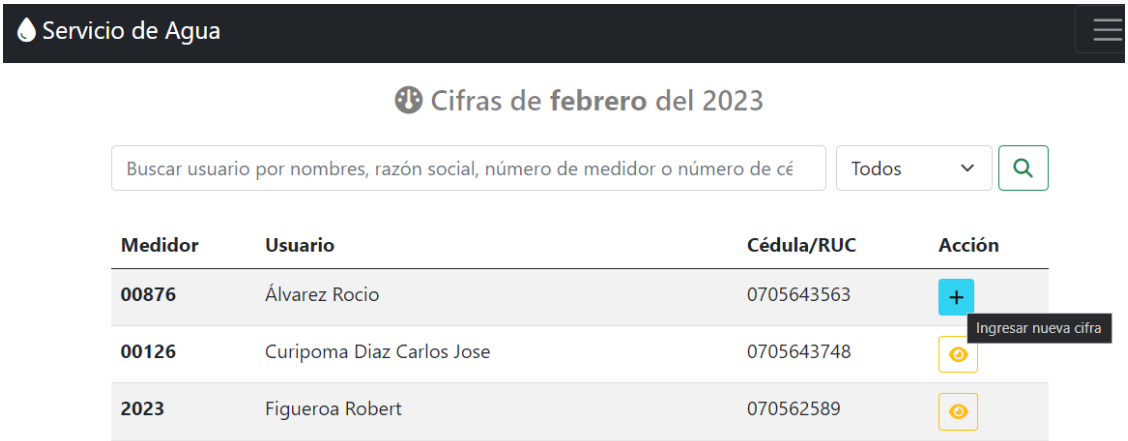


Figura 25. *Listado de clientes con acciones para ingresar cifras o ver cifras ingresadas.*

Para realizar esta acción fue necesario implementar la función seleccionar clientes, dentro de esta se obtienen todos los clientes registrados en la base de datos, así mismo, se obtiene la fecha actual del servidor, esto para colocar la acción debida a cada cliente, estos datos se envían a la plantilla html para la visualización del usuario. En la **figura 26** se observa la obtención de todos los clientes haciendo uso del modelo cliente y la función all que ofrece Django para la obtención de todos los elementos de la BD del modelo solicitado.

```python
def seleccion_clientes(request):
    busqueda = request.GET.get("buscar")
    filtro = request.GET.get("filtro")
    try:
        clientes = Cliente.objects.all()
    except Cliente.DoesNotExist:
        raise Http404
    #para obtener el año y mes
    date = datetime.date.today()
    año = date.strftime("%Y")
    mes = date.strftime("%B")
    fecha = mes+año
```

Figura 26. *Código fuente para listar clientes e ingresar su respectiva cifra.*

Para mostrar la acción correspondiente de cada usuario se comparó el campo última cifra ingresada del objeto cliente de cada uno de los clientes con la fecha actual obtenida en la función anteriormente indicada. Si la comparación realizada es certera (la fecha actual coincide con la fecha de la última cifra), la acción a mostrar es ver cifras ingresadas, el administrador solo puede visualizar las cifras porque la cifra actual ya ha sido ingresada; si por el contrario las fechas no coinciden, la acción es ingresar nueva cifra, el administrador puede ingresar la cifra correspondiente al mes tal, esta comparativa se muestra en la **figura 26***:*

```
{% if cliente.ultima_cifra == fecha %}
    <a title="Ver cifras ingresadas" class="btn btn-sm btn-outline-warning "
    href="/ingresarcifra/{{cliente.id}}"><i class="fa-solid fa-eye"></i></a>
{% else %}
    <a title="Ingresar nueva cifra" class="btn btn-sm btn-info "
    href="/ingresarcifra/{{cliente.id}}"><i class="fa-solid fa-plus"></i></a>
{% endif %}
```

Figura 27. *Comparativa para mostrar acciones de cifras.*

Una vez seleccionada la acción se muestra todas las cifras ingresadas del usuario, y la opción de ingresar la cifra del mes actual si aún no se ha ingresado, tal como se muestra en la **figura 28***:*

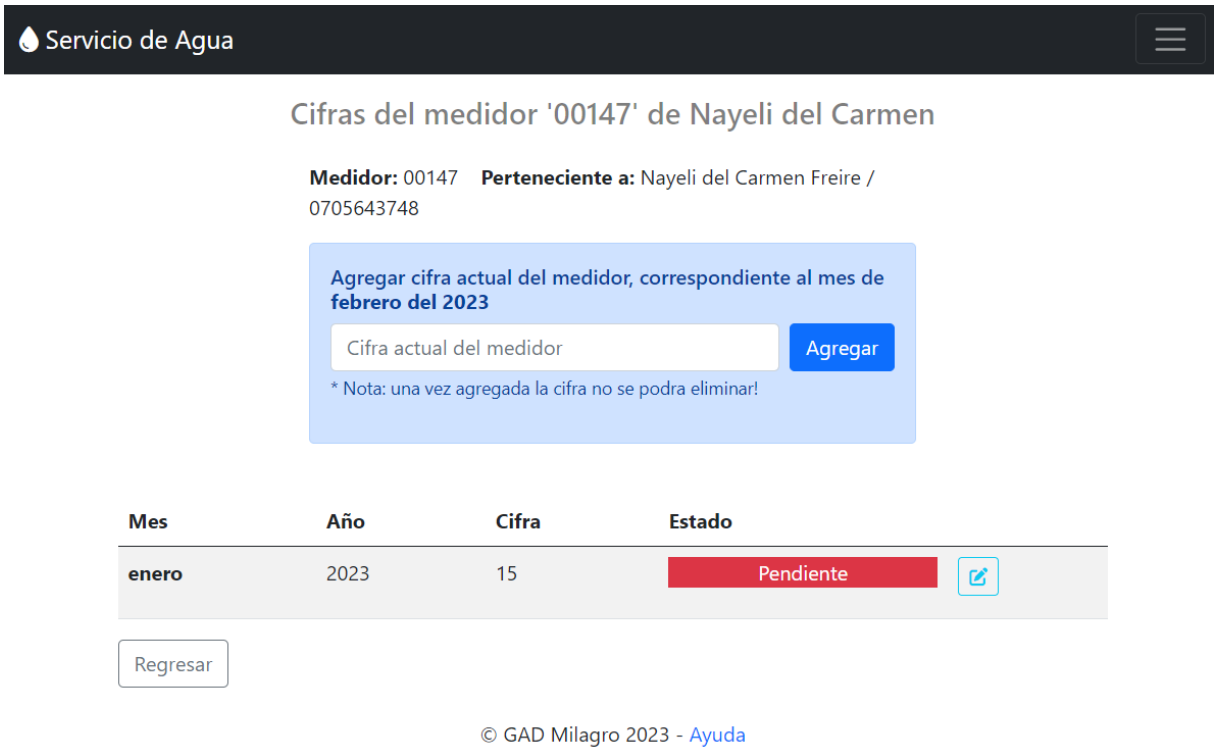


Figura 28. *Listado de cifras ingresadas e ingreso de nueva cifra.*

Para la implementación de lo anteriormente mencionado fue necesario la creación de la función ingresar cifra, la cual primeramente obtiene de la base de datos el cliente seleccionado datos necesarios para que el administrador tenga constancia de que realmente está ingresando la cifra al usuario correcto, así mismo, se obtiene la fecha actual para verificar si la cifra actual del usuario ha sido ingresada. Se obtiene todas las cifras que tiene el usuario, esto con el modelo cifras y la función de django para filtrar elementos (filter) comparando el id del actual cliente con el id del cliente en la tabla cifras de la BD, en la **figura 29** se puede visualizar lo anterior mencionado.

```
def ingresar_cifra(request, id):
    try:
        cliente = Cliente.objects.get(id=id)
    except Cliente.DoesNotExist:
            raise Http404
    #para obtener el año y mes
    date = datetime.date.today()
    año = date.strftime("%Y")
    mes = date.strftime("%B")
    fecha_ucifra = mes+año
    #para filtar y mostrar las cifras de cada usuario
    tcifras = Cifras.objects.filter(id_usuario = cliente).distinct()
```

Figura 29. *Función ingresar cifras.*

Posteriormente, a través de un for se recorrió toda la lista de cifras para obtener la última cifra ingresada, esto es necesario para controlar que el usuario no ingrese cifras inferiores a la

última ingresada. De igual forma, se obtiene los valores base actuales del sistema para almacenar el id de estos en la nueva cifra, ver **figura 30**.

```
#para obtener el ultimo mes ingresado
for cifras in tcifras.iterator():
    ultimomes = cifras.mes
if len(tcifras) == 0: #ayuda a combertir el dato si no hay elementos
    ultimomes=""
#para obtener el ultimo registro de los valores base
try:
    idv = ValoresBase.objects.latest('id')
    valores = ValoresBase.objects.get(id=idv.id)
except ValoresBase.DoesNotExist:
    messages.success(request, "Asegurese que esten ingresados los val
    return redirect("/cifras/")
#agregar una nueva cifra
```

Figura 30. *Recorrido de cifras y obtención de los valores base.*

Finalmente se comparó si existe una respuesta con el método POST, si esto es así significa que el administrador está tratando de ingresar una nueva cifra al sistema, se recupera a través de get la cifra ingresada en pantalla por el administrador para luego comparar con la última cifra ingresada, si la actual cifra es mayor se la registra en la base de datos junto con los valores de fecha, id de valores base, id del cliente y demás datos relevantes; si la cifra es menor se envía una advertencia al administrador, tal como se muestra en la *figura 31:*

```
#agregar una nueva cifra
if request.method == 'POST':
    #obtengo cifra ingresada en pantalla
    cifra = request.POST.get("cifra")
    # obtener la ultima cifra
    longitud = len(tcifras)
    if longitud != 0:
        utimacifra = tcifras[longitud-1]
        auxcifra = int(utimacifra.cifra)
    else:
        auxcifra = 0
    if auxcifra < int(cifra):
        #guardo valores
        c =Cifras()
        c.id_usuario = cliente
        c.mes = mes
        c.anio = año
        c.cifra = cifra
        c.id_valores = valores.id
        c.estado = 'n'
        if cifra != "":
            if c.save() != True:
                messages.success(request, "La cifra
```

Figura 31. *Captura y guardado de cifra.*

Diseño final de las interfaces de usuario de la aplicación web para el cobro del servicio de agua

En la **figura 32** se muestra el listado de clientes que están atrasados con sus pagos, de igual forma en la **figura 33** se puede observar el recibo de pago con todos los valores pendientes de pago que posee un usuario del sistema, este recibo es enviado al correo del cliente y a su vez es generado en archivo pdf para descargar o imprimir. Para ver más en detalle cada una de las vistas, ver anexo 5: **Prototipo final de interfaz** de la aplicación web para el cobro del servicio de agua.

Figura 32. *Pantalla principal con el listado de usuarios con pagos pendientes.*

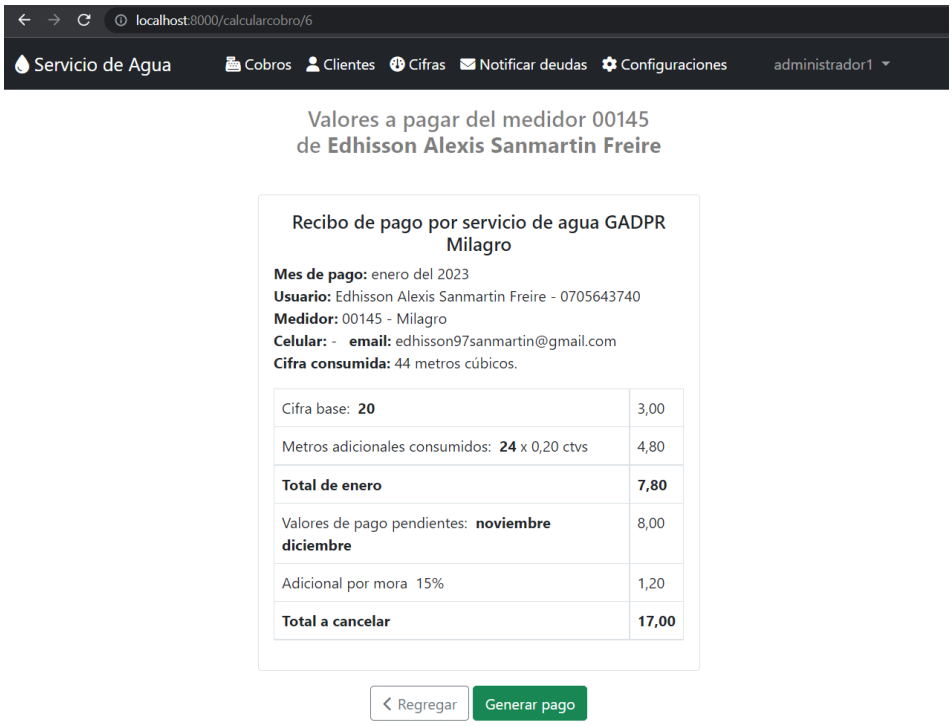


Figura 33. *Recibo de pago del servicio de agua.*

A continuación, en la **figura 34** se puede observar el recibo de pago por el servicio de agua potable enviado al correo del cliente.

Figura 34. *Correo receptado por el cliente con el recibo de pago.*

Con la finalidad de ayuda al usuario del sistema y para conocer más a detalle cada una de las de las interfaces y funcionalidades que tiene la aplicación web, ver Anexo 8: **Manual de usuario de la aplicación web para el cobro por el consumo de agua de la Parroquia Milagro.** En la **figura 35** se puede visualizar las acciones con las que cuenta la sección de cobros de la aplicación web.

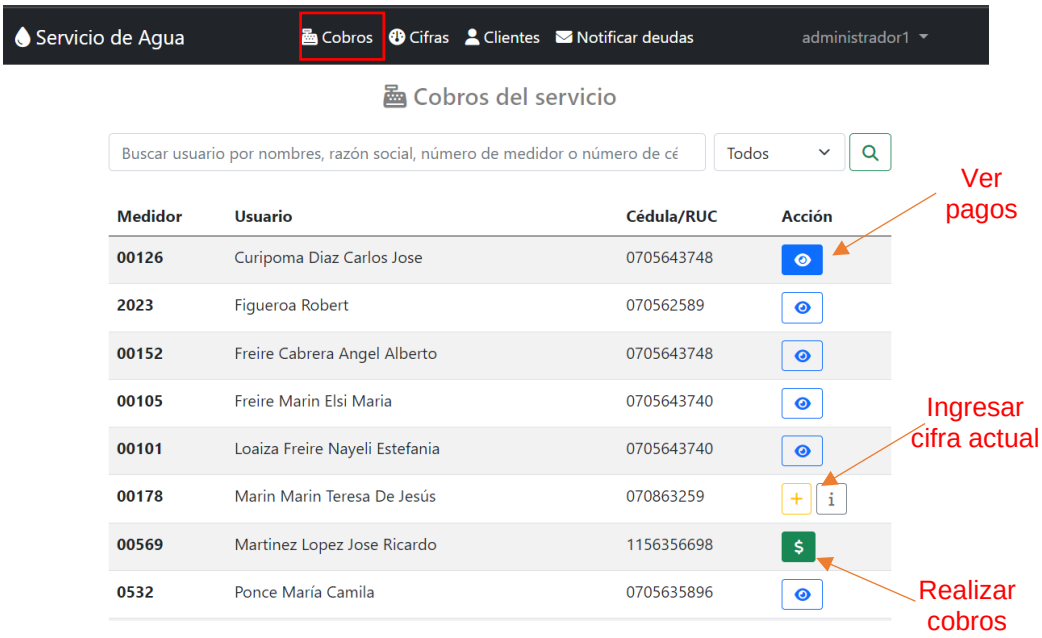


Figura 35. *Acciones para el listado de clientes de la sección de cobros.*

6.3. Objetivo 3: Probar y validar la aplicación web mediante pruebas unitarias y pruebas de estrés en ambiente simulado.

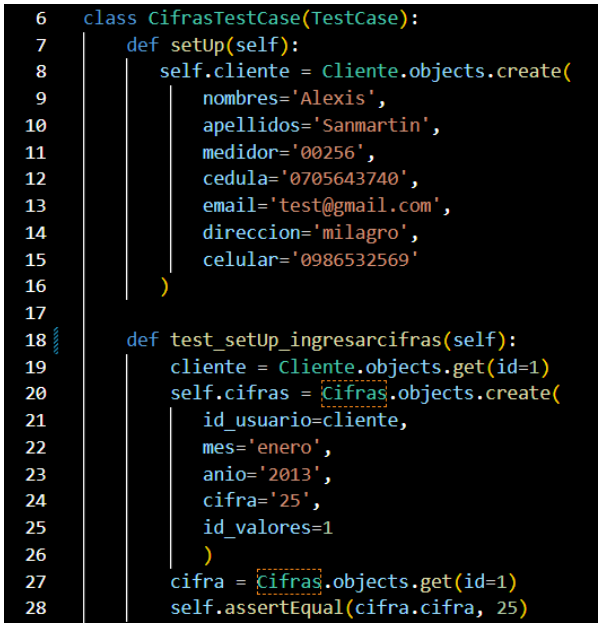
En esta sección se describe la tercera fase para el desarrollo y cumplimiento del presente Trabajo de Titulación donde se realizó la codificación de las pruebas unitarias utilizando la herramienta TestCase que nos ofrece el framework Django, se realizó la recodificación en cuanto a errores y funcionalidades para posteriormente desplegar la aplicación en los servidores de Render. Alojada la aplicación en la web se realizó las pruebas de carga con el uso de la herramienta Apache JMeter y, finalmente, se realizó pruebas de funcionalidad con el cliente a través de una entrevista.

6.3.1. Realizar los test de la aplicación web y corregir los errores.

Las pruebas que se realizaron a la aplicación web son los Test Unitarios y las pruebas de estrés mismas que se describen a continuación:

Pruebas Unitarias

Para la realización de los Test Unitarios de la aplicación se hizo uso de la herramienta TestCase que nos ofrece Django, esta herramienta ejecuta todos los test que se encuentren dentro de la aplicación, crea una Base de Datos para probar modelos si es necesario y, las destruye una vez ejecutado los test. En la **Figura 36** se puede visualizar el test unitario para ingresar la cifra a partir de un cliente ficticio, lo que realiza el test es, a partir de una base de datos de prueba creada únicamente para estas pruebas, crear un cliente haciendo uso del modelo para Clientes, posteriormente se recupera este cliente para crear una cifra consumida a partir del modelo de Cifras, finalmente, se obtiene la cifra de la base de datos y se compara con uno de los datos ingresados, si esta comparación es correcta se pasa el test. Para la visualización en detalle de cada una de las pruebas unitarias realizadas a cada uno de los modelos de la aplicación, se puede observar el Anexo 6: **Pruebas Unitarias de la Aplicación para el Cobro del Servicio de Agua Potable de la Parroquia Milagro.**



```
6   class CifrasTestCase(TestCase):
7       def setUp(self):
8           self.cliente = Cliente.objects.create(
9               nombres='Alexis',
10              apellidos='Sanmartin',
11              medidor='00256',
12              cedula='0705643740',
13              email='test@gmail.com',
14              direccion='milagro',
15              celular='0986532569'
16          )
17
18      def test_setUp_ingresarcifras(self):
19          cliente = Cliente.objects.get(id=1)
20          self.cifras = Cifras.objects.create(
21              id_usuario=cliente,
22              mes='enero',
23              anio='2013',
24              cifra='25',
25              id_valores=1
26              )
27          cifra = Cifras.objects.get(id=1)
28          self.assertEqual(cifra.cifra, 25)
```

Figura 36. *Test Unitario para el ingreso de cifras.*

En la **figura 37** se visualiza la ejecución de los test para las Cifras, donde, se crea la base de datos ficticia, se ingresan los datos a probar, se realizan las respectivas comparaciones y, finalmente, independientemente de los resultados obtenidos, se destruye la base de datos como se muestra a continuación:

```
(venv) C:\Users\USER\Documents\ProjectsDjango\AppAguaMilagro>py manage.py test cifras
Found 2 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
..
----------------------------------------------------------------------
Ran 2 tests in 0.007s

OK
Destroying test database for alias 'default'...
```

Figura 37. *Ejecución de Test Unitarios para el módulo de cifras.*

Despliegue de la aplicación

Para el despliegue de la aplicación en la web se hizo uso del servicio SaaS que nos ofrece render en sus servidores en la web y que provee una BD gratuita con 1Gb de almacenamiento, RAM de 256Mb y CPU 100m. Por parte del desarrollo se realizó algunas configuraciones necesarias que apunten al despliegue en el servicio de render y a la base de datos PostgreSQL en el archivo settings del proyecto.

Render hace el despliegue de la aplicación a través de repositorios de código fuente, en este caso, se hizo uso y se alojó el código fuente de la aplicación web en los repositorios de GitHub: https://github.com/servicioaguamilagro/SAguaMilagro. Para la conexión con la base de datos se utilizó las variables de entorno a través de render, así mismo, se implementa la librería dabase_url al entorno Django, esto permito conectar la base de datos deseada a través de una url de las variables de entorno. Como el despliegue es de prueba se realiza una configuración para producción de PostgreSQL y para el entorno de desarrollo se configura sqlite3 como se puede observar en la **figura 38**.

```
106  # Database
107  DATABASES = {
108      'default': dj_database_url.config(
109          default='sqlite:///db.sqlite3',
110          conn_max_age=600
111      )
112  }
```

Figura 38. *Conexión a BD en Django.*

En render se agregó una variable de entorno con el nombre DATABASE_URL y con la url de conexión de la BD PostgreSQL. La aplicación web desplegada hace uso de la url: https://servicioaguamilagro.onrender.com/.

Pruebas de Carga y Estrés

Para realizar las pruebas de carga y estrés de la Aplicación se utilizó la herramienta Apache JMeter, misma que se configuró como se muestra en la **figura 39** definiendo sus parámetros iniciales.

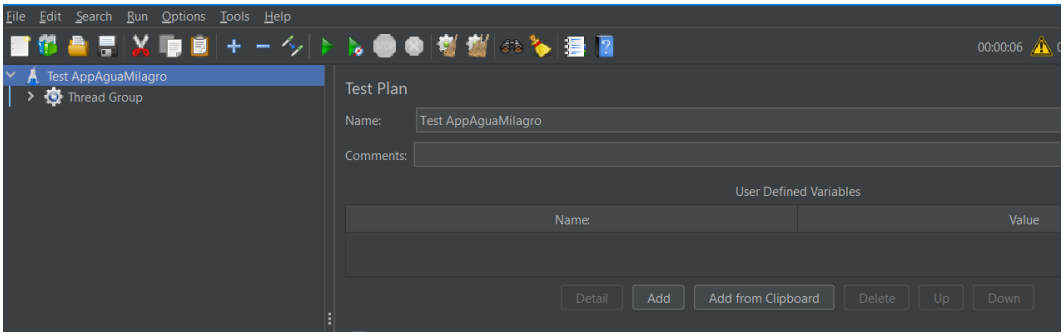


Figura 39. *Creación de la prueba de estrés en JMeter.*

En la **figura 40** se visualiza la configuración de HTTP Request a la Aplicación Web con el grupo de hilos, donde se ingresan 500 peticiones en un periodo de un segundo que se ejecutarán concurrentemente.

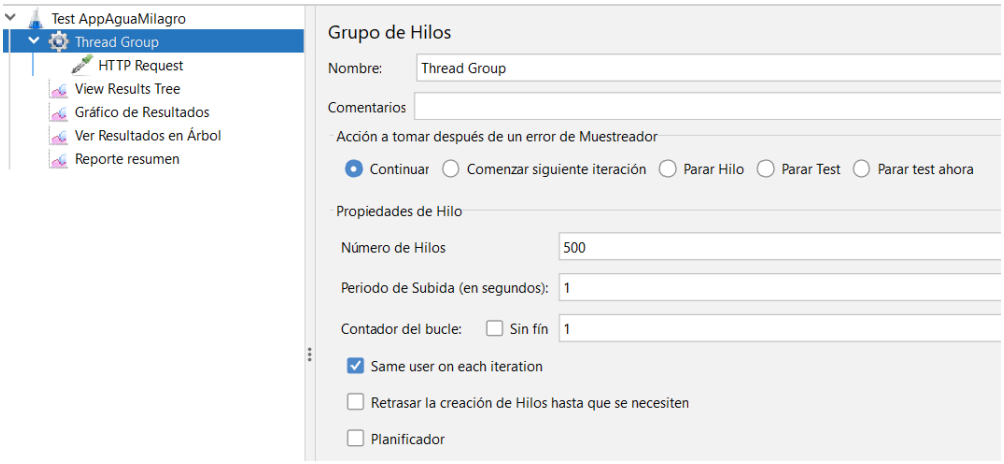


Figura 40. *Configuración de peticiones hacia la Aplicación Web.*

Al ejecutar la prueba de carga a la aplicación se obtuvo un gráfico donde se puede visualizar que al procesar 500 peticiones por segundo la aplicación logra mantener un rendimiento estable a lo largo de la ejecución. En la **figura 41** se puede observar más en detalle cada uno de los apartados que toma en cuenta la prueba:

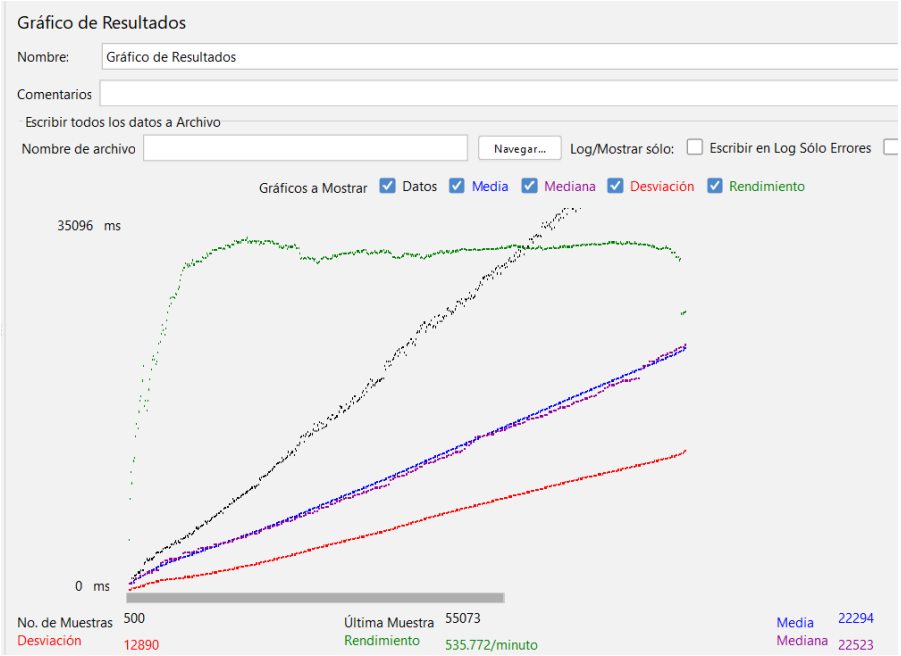
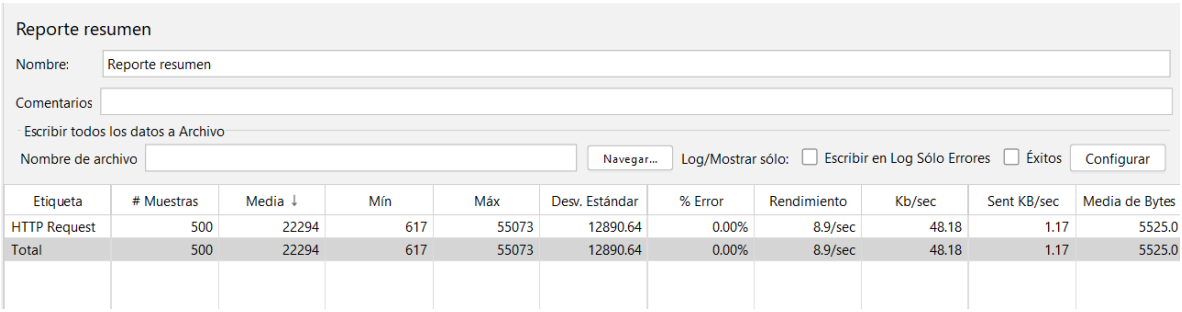


Figura 41. *Gráfico de resultados de la prueba de carga y estrés.*

De igual manera, en la **figura 42** se observa el resultado de la prueba, en resumen, destacando que el error obtenido es del 0% para las 500 peticiones realizadas, con un rendimiento de 8.9/sec por lo que se puede afirmar que la aplicación web tiene un funcionamiento correcto con tiempos de respuesta mínimos de 617 milisegundos y tiempos de respuesta máximos de 55073 milisegundos.



Etiqueta	# Muestras	Media ↓	Mín	Máx	Desv. Estándar	% Error	Rendimiento	Kb/sec	Sent KB/sec	Media de Bytes
HTTP Request	500	22294	617	55073	12890.64	0.00%	8.9/sec	48.18	1.17	5525.0
Total	500	22294	617	55073	12890.64	0.00%	8.9/sec	48.18	1.17	5525.0

Figura 42. *Reporte de la prueba de carga y estrés.*

Corrección de errores de la Aplicación

El principal error en esta fase de testeo y despliegue de la aplicación es el de obtención de las fechas en el idioma español, la configuración del idioma en Django no fue factible, esto debido a las limitaciones del servidor gratuito de despliegue y la compatibilidad con Django. Para mitigar este problema se procedió a hacer uso de la función de traducción que ofrece Django, para ello se implementa la etiqueta de internacionalización en cada una de las

plantillas que se va a traducir y, por ende, la etiqueta de traducciones en cada una de las variables a traducir, tal como se muestra en la ilustración a continuación:

```
4
5   {% block content %}
6   {% load i18n %}
7       <div class="container mb-1 " style="max-width: 800px;">
8           <h4 class="text-black-50 text-center py-3">Valores a pagar del medidor {{ cliente.medidor }} <p>de <b>{{ cliente.nombres }} {% if
9       </div>
10  {% if total != None %}
11      <div class="container mb-1 " style="max-width: 500px;">
12          <div class="card" style="max-width: 500px;">
13              <div class="card-body">
14                  <h5 class="card-title text-center">Recibo de Pago del Servicio de Agua Potable del GAD Parroquial de Milagro </h5>
15                  <p class="card-text"><b>Mes de pago: </b>{% filter slugify %}{% trans mes %}{% endfilter %} del {{ año }}<br>
16                                      <b>Usuario: </b>{{ cliente.nombres }} {% if cliente.apellidos == None %}  {% else %} {{ cliente.apell
17                                      <b>Medidor: </b>{{ cliente.medidor }} - {{ cliente.direccion }}<br>
18                                      <b>Celular: </b>{% if cliente.celular == None %} - {% else %} {{ cliente.celular }} {% endif %} 
```

***Figura 43**. Traducción de variables en plantillas.*

6.3.2. Realizar la recodificación de la aplicación web en cuanto a fallos y funcionalidades que varíen.

Al realizar las pruebas y visualizar la aplicación desplegada con el cliente, una de las observaciones que se obtuvo es que; al consultar los clientes desde la base de datos, los listados de estos no tienen un orden definido, los clientes se muestran conforme se van ingresando al sistema. Para solventar este problema se procedió a la edición del modelo de clientes agregando una clase meta donde se define el valor por el que se desea ordenar el listado, tal como se muestra en la **figura 44**, al efectuar estos cambios en los modelos es necesario realizar las migraciones para que los cambios se reflejen correctamente.

```
15
16      def __str__(self):
17          #apunta con email
18          return  self.medidor
19      class Meta:
20          db_table = 'cliente'
21          verbose_name = 'Cliente'
22          verbose_name_plural = 'clientes'
23          ordering = ['apellidos']
24
```

***Figura 44.** Edición del modelo cliente.*

De igual manera, al simular la aplicación en un entorno real se pudo indagar que, para efectuar una acción en específico como por ejemplo realizar los cobros del servicio, se necesita tener un listado con más visibilidad del panorama, por lo cual es necesario filtrar los usuarios por grupos. Para corregir esta acción se realizó un filtro de clientes en cada apartado, así, en

cobros se cuenta con un filtro para un grupo de clientes que ya realizado el pago del servicio y otro que agrupa los usuarios con deudas pendientes, tal como se observa a continuación:

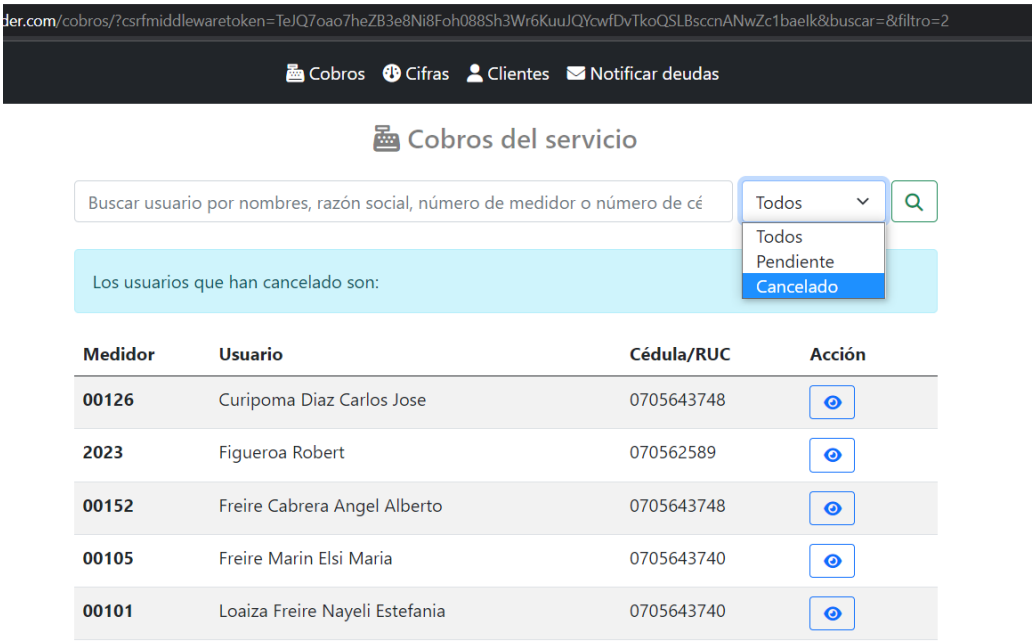


Figura 45. *Filtro de clientes.*

Para realizar esta acción fue necesario listar los clientes haciendo una consulta a la base de datos donde se agrupan de acuerdo al grupo al que le corresponde, tal como se muestra en la **figura 46** a continuación:

```
if filtro:
    if str(filtro) == '1':
        clientes = Cliente.objects.filter(
            Q(deuda__icontains = 's')).distinct()
        messages.success(request, "Los usuarios con deudas pendientes son: ")
    if str(filtro) == '2':
        clientes = Cliente.objects.filter(
            Q(deuda__icontains = 'n')).distinct()
        messages.success(request, "Los usuarios que han cancelado son: ")
```

Figura 46. *Filtrado de clientes.*

Otra de las funcionalidades que hacía falta y que sin duda es sumamente importante de acuerdo a la indagación realizada con el tutor, es la de recuperar la contraseña de acceso al sistema. Esta funcional Django la trae incorporada, para realizar su activación es necesario especificar las url en el archivo de configuración de rutas, tal como se observa en la siguiente figura, además, se personalizó las plantillas que trae por defecto el framework para que estén acorde a la aplicación.

```
#reseteo contraseñas
path('reset_password/',auth_views.PasswordResetView.as_view(template_name="contraseñaResetear.html"), name='password_reset'),
path('reset_password_send/',auth_views.PasswordResetDoneView.as_view(template_name="contraseñaEnvio.html"), name='password_reset_done'),
path('reset/<uidb64>/<token>',auth_views.PasswordResetConfirmView.as_view(template_name="contraseñaNueva.html"),name='password_reset_confirm'),
path('reset_password_complete',auth_views.PasswordResetCompleteView.as_view(template_name="contraseñaCompletado.html"),name='password_reset_comp
```

Figura 47. *Activación para recuperar contraseña de Django.*

La activación para recuperar la contraseña en Django consta de 4 vistas, la primera muestra un pequeño formulario para ingresar el correo electrónico del usuario, la segunda se encargar de enviar al correo del usuario las instrucciones para cambiar la contraseña, la tercera muestra un formulario que se envía al correo para el ingreso de la nueva contraseña y, la última confirma el cambio de la nueva contraseña. A continuación, se puede observar el correo que se envía al usuario junto con el formulario para ingresar las nuevas contraseñas:

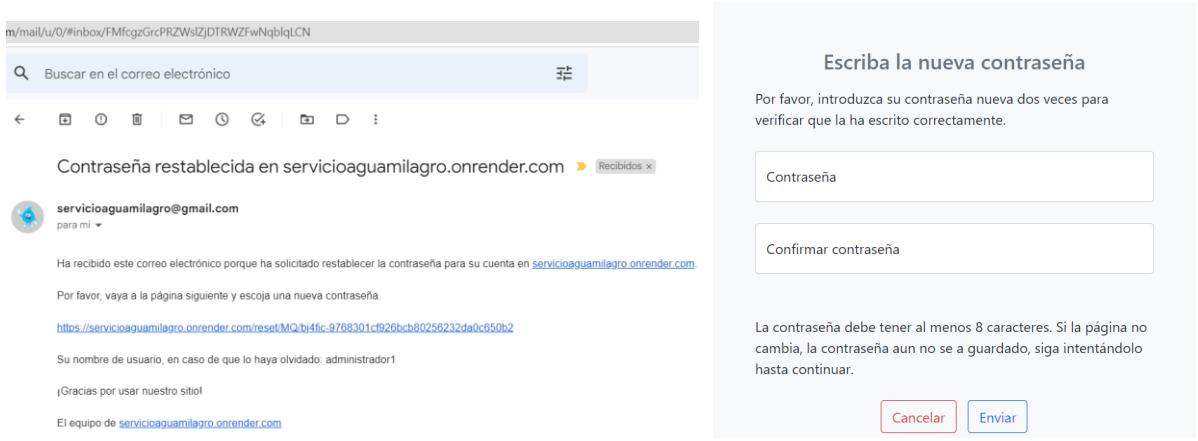


Figura 48. *Restablecimiento de contraseña.*

6.3.3. Validar la aplicación con el cliente.

La validación de la aplicación web para el cobro por el servicio de agua en la Parroquia Milagro se la hizo a través de una entrevista realizada al cliente, obteniendo como resultado que el software cumple con cada uno de los requisitos funcionales y no funcionales definidos al inicio del desarrollo del presente proyecto.

A continuación, en la **tabla 16** se observa la validación del cliente de cada uno de los requisitos funcionales de la aplicación web, para ver más a detalle la entrevista ver Anexo 7: **Entrevista – Validación de la Aplicación Web para el cobro por el consumo del servicio de agua potable en el Gobierno Autónomo Descentralizado Parroquial Rural de Milagro.**

Tabla 16. *Validación de los requisitos funcionales del sistema.*

#	Nombre	Cumple	No Cumple
RF001	Inicio de sesión	**X**	
RF002	Ingreso de usuarios	**X**	
RF003	Cobros del servicio	**X**	
RF004	Ingreso de cifras consumidas	**X**	
RF005	Comprobante de pago	**X**	
RF006	Configuraciones del sistema	**X**	
RF007	Notificar deuda	**X**	

En la **tabla 17** se puede visualizar la validación de cada uno de los requisitos no funcionales del software.

Tabla 17. *Validación de los requisitos no funcionales del sistema.*

#	Nombre	Cumple	No Cumple
RN001	Compatibilidad	**X**	
RN002	Desempeño	**X**	
RN003	Disponibilidad	**X**	
RN004	Fiabilidad	**X**	
RN005	Tiempo de respuesta	**X**	
RN006	Usabilidad	**X**	

Para la estimación de los tiempos y conocer la usabilidad de la aplicación web se lo realizó a través de la entrevista donde el entrevistado destacó que el software es fácil de entender y utilizar, es amigable al usuario, cuenta con tiempos de respuesta óptimos y mejora considerablemente el proceso de cobros. Además, para la estimación de tiempos se tomó una muestra de estos en los casos que tiene en el servicio, como se muestra en la **tabla 18**, minimizando los tiempos del cobro del servicio, con valores inferiores al minuto mientras que manualmente los valores rondan los 4 minutos. Para ver en detalle la entrevista y la muestra de tiempos ver ***Anexo 9. Entrevista – Usabilidad de la Aplicación Web para el cobro por el consumo del servicio de agua potable.***

Tabla 18. *Tiempos de cálculos de valores del servicio.*

Acción realizada	Cálculos manuales	Cálculos en la aplicación web
Usuario con última cifra pendiente sin exceder los valores base	3:20	0:31
Usuario con última cifra pendiente con valores base excedidos	3:54	0:18
Usuario con deudas pendientes y valores excedidos	5:53	0:22
Tiempo promedio	**4:33**	**0:23**

7. Discusión

Para el desarrollo exitoso del presente Trabajo de titulación fue necesario el cumplimiento de cada una de las etapas o fases de los objetivos, mismos que se describen a continuación:

Objetivo 1: Especificar los requerimientos y diseñar los artefactos tecnológicos mediante la metodología de software eXtreme Programming.

Las fases de la metodología XP permitieron desarrollar de manera exitosa el Trabajo de Titulación. La técnica de la entrevista al responsable de la institución ayudó a la obtención y especificación de los requisitos del sistema, consiguiendo siete requisitos funcionales y seis requisitos no funcionales. Entre los requisitos funcionales más destacados se cuenta con: ingreso de cifras consumidas por cada usuario, notificación de deudas pendientes y generación de comprobantes de pago del servicio con envío al correo electrónico del cliente. Como resultado final se obtuvo el documento de especificación de Requerimientos de software en base al estándar IEEE 830.

Para el diseño del sistema, se hizo uso del modelo arquitectónico 4+1 que se fundamenta en la vista de escenarios, vista lógica, vista de despliegue, vista física y la vista de procesos; lo cual ayudo a modelar y documentar de la mejor manera el diseño de la aplicación web. De igual manera, para el diseño de la interfaz se hizo uso de la herramienta pencil, esto ayudo a tener una idea clara para el desarrollo de la aplicación y para que el cliente pueda observar algunas funcionalidades definidas.

Objetivo 2: Construir la aplicación web mediante el framework Django y la librería Bootstrap.

La fase de codificación de la aplicación de cumplió de manera exitosa, la curva de aprendizaje del lenguaje Python para el uso del framework Django junto con la librería Bootstrap que ayuda al diseño fue fácil con un grado alto de dificultad, esto gracias a la gran variedad de información y documentación para desarrollo en la web que ofrece el framework, lo que ayudó a cumplir con los tiempos estimados de desarrollo.

A lo largo de la fase de codificación algunas funcionalidades complicaron el desarrollo, como la de generar recibos en formato pdf y enviar estos recibos al correo del cliente, el framework al ser libre y tener gran equipo de desarrollo detrás, ofrece una gran variedad de librerías que suplantan cualquier funcionalidad en mente, si una librería no satisface a cabalidad una de las funcionalidades planteadas, fácilmente se encuentra otra que se acople mejor a lo deseado. Esto ayudó a suplir de manera correcta y con rapidez alguna de las funcionalidades con

dificultad, además de existir una ayuda realmente importante por la variedad de información existente y todo el grupo de comunidades de desarrollo del framework.

Objetivo 3: Probar y validar la aplicación web mediante pruebas unitarias y pruebas de estrés en ambiente simulado.

Se realizó los test unitarios para comprobar el correcto ingreso, edición y eliminación de datos de cada uno de los modelos de datos. Cada uno de los test cumplió a la perfección el objetivo por el que fue creado. De igual manera, al realizar el despliegue de la aplicación web en los servidores de render no hubo mayor problema, la producción de la aplicación se realizó con éxito para posteriormente realizar las pruebas de validación y, de carga y estrés.

Al realizar una de las primeras reuniones con el cliente para validar las funcionalidades del sistema se pudo llegar a la conclusión de que una de las funcionalidades que debían mejorar es la de listar los clientes, inicialmente se los listaba en el orden de ingreso, pero, con fines de ofrecer facilidades al usuario administrador se optó por realizar el listado alfabéticamente. Este cambio se logró modificando el modelo de datos del cliente agregando una función de ordenar. Así mismo, con el docente tutor se puedo observar que una de las funcionalidades faltantes en la aplicación web es la de recuperar la contraseña de ingreso de los usuarios, esto se logró fácilmente activando la funcionalidad que el mismo framework ofrece.

Las pruebas de carga y estrés dieron un resultado positivo con un rendimiento alto, la herramienta que ayudó a realizar los test es Apache JMeter, con la única novedad de que no se puede estresar mucho al sistema debido a que el servicio de alojamiento en la nube que ofrece render es limitado al ser gratuito, por su parte, esto limita un poco el rendimiento del sitio, sin embargo, con peticiones un poco altas al uso normal que recibirá el sistema, el mismo responde más que a la perfección.

Finalmente, la validación con el cliente se la realizó a partir de una entrevista donde se enumeran cada uno de los requisitos funcionales, no funcionales, y si cumple o no el software con cada una de ellas; obteniendo un resultado positivo en el que se aprueba cada uno de los requisitos definidos inicialmente.

Valoración Técnica económica ambiental

Para el desarrollo del presente Trabajo de titulación, se emplearon recursos tanto técnicos, como económicos y ambientales.

Los recursos técnicos como lo es el Framework Django con el lenguaje de programación Python permitieron llevar a cabo el desarrollo de la aplicación web. Las librerías y

herramientas de software utilizadas, al ser de versión libre; así como algunos servicios, al ser versión gratuita, no influyeron a que los costos no sean elevados, esto permitió tener un perfecto balance y correcto desarrollo. En el aspecto ambiental, se reduce la impresión de comprobantes de pago, utilizando a estos físicamente solo si el usuario desea, esto debido a que la aplicación envía los recibos de pago directamente al correo del cliente. En la siguiente tabla se puede observar más en detalle los recursos utilizados.

Tabla 19. *Recursos humanos, materiales, técnicos y tecnológicos.*

RECURSOS HUMANOS			
Cargo	**Número de horas**	**Precio/Hora**	**Valor Total**
Estudiante	400	5	$ 2000
Tutor	48	12	$ 576
Profesor de la asignatura	192	12	$ 2304
		Total	**$ 4480**
RECURSOS MATERIALES			
Descripción	**Meses**	**Precio unitario**	**Valor total**
Internet	4	$ 25	**$ 100**
RECURSOS TECNOLÓGICOS Y TÉCNICOS			
Descripción	**Cantidad**	**Precio Unitario**	**Valor Total**
Laptop	1	$ 800	**$ 800**
Draw.io	1	$ -	**$ -**
Django	1	$ -	**$ -**
GitHub	1	$ -	**$ -**
Zoom	1	$ -	**$ -**
Pencil	1	$ -	**$ -**
Visual Studio Code	1	$ -	**$ -**
Lenguaje de programación	1	$ -	**$ -**
		Total	**$ 800**

En la tabla descrita a continuación se puede observar en resumen el costo final de los recursos utilizados en el Trabajo de Titulación.

Tabla 20. *Costo total del Trabajo de Titulación.*

DESCRIPCIÓN	VALOR TOTAL
Recursos humanos	$ 4480
Recursos Materiales	$ 100
Recursos técnicos y tecnológicos	$ 800
Subtotal	**$ 5380**
Imprevistos (10%)	$ 538
TOTAL	**$ 5918**

Al ser un Trabajo de Titulación los recursos humanos son solventados por la Universidad Nacional de Loja, teniendo así en la siguiente tabla el resultado final real del costo del presente Trabajo de Titulación.

Tabla 21. *Costo final total del Trabajo de Titulación.*

DESCRIPCIÓN	VALOR TOTAL
Recursos Materiales	$ 100
Recursos técnicos y tecnológicos	$ 800
Subtotal	**$ 900**
Imprevistos (10%)	$ 90
TOTAL	**$ 990**

8. Conclusiones

Una vez finalizado el presente trabajo de titulación, se concluye que:

- El desarrollo de la aplicación web para el cobro por el servicio de agua potable de la parroquia Milagro fue importante porque mediante la entrevista se constató que se minimizaron los tiempos del cobro del servicio con valores inferiores al minuto mientras que manualmente los valores rondan los 4 minutos, se mitigó la pérdida de información almacenando datos importantes en la BD PostgreSQL, además ayudó a reducir el esfuerzo humano automatizando todos los cálculos de los valores a cancelar de cada cliente y, realizó un orden de la información ante la confusión y el desorden existente llevando un registro de cifras, valores cancelados, clientes y deudas pendientes.
- La entrevista directa al cliente es una herramienta fundamental en la construcción del software para la especificación de requisitos, esta importante técnica ayudó a la recopilación de siete requisitos funcionales y seis requisitos no funcionales que deben estar contemplados en la aplicación web, quedando establecidos en el documento IEEE 830 para la especificación de requisitos de software.
- La metodología de desarrollo XP me ayudó de manera vital a organizar y llevar un desarrollo ordenado en cada una de sus fases: planificación, diseño, codificación y pruebas, esto significó un desarrollo ágil y exitoso del proyecto; de igual forma, para tener una visión general del comportamiento y desarrollo del sistema la arquitectura 4+1 fue fundamental.
- Django es un framework de desarrollo potente que ayuda a la creación de aplicaciones web de cualquier tipo sin la necesidad de tener un conocimiento exhaustivo de la herramienta debido a toda la información y ayuda que su comunidad ofrece, esto combinado con la librería de diseño Bootstrap ayuda a generar interfaces sencillas y elegantes en poco tiempo que agradan a la vista del usuario, teniendo así un gran diseño sin la necesidad de tener un conocimiento a fondo de diseño web.
- La fase de pruebas del proyecto es una de las fases fundamentales, debido a que; ayudó a validar que la aplicación cumpla con cada uno de sus requisitos ya sean funcionales o no y la respectiva aceptación del software. Además, la realización de los test corroboró que la aplicación tiene un funcionamiento adecuado y se comporta de manera exitosa con tiempos de respuesta aceptables.
- Uno de los artefactos tecnológicos que más ayudó al desarrollo de la aplicación web son los diagramas de actividades debido a que de manera general reflejan el proceso de las funcionalidades a desarrollar.

9. Recomendaciones

Una vez finalizado el presente trabajo de titulación, se recomienda lo siguiente:

- Realizar una revisión a fondo de las tecnologías que se van a utilizar en el desarrollo del proyecto para evitar problemas futuros que se pueden presentar al desarrollar alguna funcionalidad que tenga un alto grado de complejidad.
- Tener reuniones constantes con cada uno de los interesados del proyecto de software debido a que, las funcionalidades pueden variar o se pueden mal interpretar por parte del desarrollador lo que repercute en no implementar mayores cambios cuando se realice la validación y aceptación del sistema con el cliente y, a no tener retrasos en la entrega del producto final.
- Desarrollar aplicaciones web con el framework Django debido a que ofrece la creación de aplicaciones web en tiempo récord y con funcionalidades de un elevado grado de dificultad, este potente entorno de desarrollo cuenta con algunas funcionalidades incorporadas que en otros frameworks habría que desarrollarlas desde cero.
- Aplicar correctamente cada una de las fases de la metodología de software, esto permitirá llevar un entorno de desarrollo ordenado, cumplir con los tiempos estimados y ofrecer un producto de calidad que satisfaga todas las necesidades del cliente.

9.1. Trabajos futuros

- Desarrollar una aplicación móvil que permita a los usuarios consultar deudas pendientes que tengan por el servicio, con la posibilidad de realizar los pagos del mismo.
- Implementar un módulo para que los usuarios puedan realizar directamente los pagos por el servicio a través de los diferentes métodos de pago existentes en la actualidad: PayPal, tarjetas de crédito, monedas virtuales, entre otros.
- Agregar los procesos de solicitud del servicio, multas y reconexión, esto dentro del servicio de agua; así como servicios externos a este como: funeraria, limpieza a lugares públicos de la parroquia, multas, entre otros.

10. Referencias

[1] F. Álvarez, Implementación de nuevas tecnologías, San Salvador: UFG Editores, 2015.

[2] S. García , La guía definitiva de Django: Desarrolla aplicaciones web de forma rápida y sencilla, Celaya: Apress, 2015.

[3] A. Bravo, El SaaS y el Cloud-Computing: una opción innovadora para tiempos de crisis, Revista Española de Innovación, Calidad e Ingeniería del Software*,* vol. 5, nº 1, p. 38, 2009.

[4] Render, Cloud application hosting for developers, [En línea]. Available: https://render.com/. [Último acceso: 15 01 2023].

[5] C. Penadés y P. Letelier, Metodologías ágiles para el desarrollo de software: extreme programming (xp), nº 2017, p. 17.

[6] R. Salgado, Metodologías de desarrollo de proyectos informáticos en entornos web, 2010.

[7] J. Segui, Arquitectura y Diseño de Sistemas Web y C/S, Departamento Ciencias de la Computación, 2016.

[8] Mdn, Mozilla Corporation, [En línea]. Available: https://developer.mozilla.org/. [Último acceso: 04 01 2023].

[9] R. Agut, Especificación de Requisitos Software según el estándar de IEEE 830, *E78.* Ingeniería del software*,* 2001.

[10] P. Kruchten, Planos Arquitectónicos: El Modelo de “4+1” Vistas de la Arquitectura del Software, IEEE Software *,* vol. 6, nº 12, 1995.

[11] M. Sánchez, N. Silega y O. Rojas, Revisión de elementos conceptuales para la representación de las arquitecturas de referencias de software, Revista Cubana de Ciencias Informáticas, vol. 13, nº 1, pp. 143-157, 2019.

[12] H. Cervantes, P. Velasco y L. Castro, Arquitectura de software. Conceptos y ciclo, México DF: Cengage Learning , 2016.

[13] J. Kaplan-Moss y A. Holovaty, El libro de Django 1.0, Apres., 2007.

[14] C. Pérez, El lenguaje de programación Python/The programming language Python, Ciencias Holguín, 2014.

[15] M. Marqués, Bases de datos, Castellon de la Plana: Publicacions de la Universitat Jaume I, 2011.

[16] P. Denzer, PostgreSQL, Universidad Técnica Federico Santa María, Valparaíso, 2002.

[17] R. Camps, L. Casillas, D. Costal, G. Ginestà, C. Martín y O. Pérez, Bases de datos en PostgreSQL, UOC Formación de posgrado, 2015.

[18] Grupo de Desarrollo de PostgreSQL, PostgreSQL, [En línea]. Available: https://www.postgresql.org/. [Último acceso: 05 01 2023].

[19] P. Mestras, Estructura de las Aplicaciones Orientadas a Objetos, Universidad Complutense Madrid, 2008.

[20] E. Bascón, El patrón de diseño Modelo-Vista-Controlador (MVC) y su implementación en Java Swing, Acta Nova, vol. 2, nº 4, 2004.

[21] Y. Fernández y Y. Díaz, Patrón Modelo-Vista-Controlador, Telemática, vol. 11, nº 1, pp. 47-57, 2012.

[22] Apache Software Foundation, Apache JMeter, [En línea]. Available: https://jmeter.apache.org/. [Último acceso: 07 01 2023].

[23] Django Project, «Documentación de Django,» Django Software Foundation. [En línea].

[24] L. Gil, Estudio comparativo de metodologías tradicionales y ágiles para proyectos de desarrollo de software, Valladolid, 2018.

[25] S. Katerine, Definición de equivalencias entre historias de usuario y especificaciones en UN - LENCEP para el desarrollo ágil de software, p. 96, 2014.

11. Anexos

Anexo 1. Entrevista – Establecimiento de Requerimientos.

ENTREVISTA – ESTABLECIMIENTO DE REQUERIMIENTOS

Departamento: Oficinas del Gobierno Autónomo Descentralizado Parroquial Rural de Milagro, Cantón Atahualpa

Entrevistado: Sr. Franco Antonio Sanmartín Gómez

Objetivo: Conocer el proceso del cobro del consumo de agua potable de la Parroquia Milagro.

Fecha de revisión: 28-11-2022

Autores: Edhisson Sanmartín Freire.

Resumen de la Entrevista:

El proceso de cobro por el consumo de agua en la Parroquia Milagro se lo realiza de forma manual teniendo en cuenta los siguientes aspectos.

- ✓ Cada usuario del servicio cuenta necesaria e indispensablemente con un medidor de agua, mismo que da las cifras de los metros consumidos mensualmente por el usuario.
- ✓ Si ingresa un nuevo usuario al servicio, el mismo deberá contar con un medidor de agua.
- ✓ Mensualmente se recolectan las cifras de cada uno de los medidores.
- ✓ Existe una persona encargada de realizar los cobros y registrarlos, misma que hace los cálculos a pagar dependiendo de la cifra base.
- ✓ La tarifa básica de consumo de cada mes es de 20 metros cúbicos que tiene un valor de 3 dólares.
- ✓ Se cobra un valor adicional de 0.20 centavos por cada metro cúbico de agua excedido a partir de la tarifa básica.

- ✓ Si el usuario no paga el consumo de un mes, se le hace un recargo del 20% del valor de la deuda.

Mediante esta información proporcionada se ha podido comprender el proceso para el cobro por consumo de agua dentro del Gobierno Autónomo Descentralizado Parroquial Rural de Milagro, misma información que contribuye en la Fase de Planificación y Diseño del Presente Trabajo de Titulación.

Firmado electrónicamente por:
FRANCO ANTONIO SANMARTIN GOMEZ

Franco Antonio Sanmartín Gómez

Presidente del Gobierno Autónomo Descentralizado

Parroquial Rural de Milagro.

Anexo 2. Especificación de requisitos de software.

Especificación de requisitos de software según el estándar IEEE 830

Proyecto: Aplicación Web para la automatización del cobro de consumo de agua potable en el Gobierno Autónomo Descentralizado Rural de la Parroquia Milagro del Cantón Atahualpa

Ficha del documento

Fecha	Revisión	Autor	Verificado dep. Calidad.
22/11/2022		**Edhisson A. Sanmartín Freire**	

Documento validado por las partes en fecha:

Por la comunidad	Por la universidad
GADPR de Milagro	**Universidad Nacional de Loja** **Ing. Roberth Figueroa Diaz Mg. SC**

1. Introducción

Mediante del presente documento se realizó la especificación de Requisitos del Software (ERS) de la “Aplicación Web para la automatización del cobro de consumo de agua potable en el Gobierno Autónomo Descentralizado Rural de la Parroquia Milagro del Cantón Atahualpa” para el comité encargado de brindar el servicio de agua en la Parroquia Milagro del cantón Atahualpa. Esta especificación se basa en la estructura establecida por la IEEE, para la Especificación de Requisitos de Software ANSI/IEEE 830/1998.

1.1. Propósito

En la Especificación de Requisitos de Software (ERS) definimos los requerimientos funcionales y no funcionales, las mismas que serán la base para el Desarrollo del Software. Por todo esto el propósito es generar una solución informática que permita optimizar tiempo, esfuerzo, trabajo y llevar un correcto manejo de recursos por parte de la persona encarda de realizar los cobros del servicio mes a mes, contando así con una herramienta indispensable en los actuales tiempos tecnológicos que permite realizar cobros, notificar deudas mediante correo, tener un registro de clientes que brinda comodidad y facilidad al recaudador.

1.2. Alcance

El desarrollo del Software como solución informática para el cobro por el servicio de agua es el objetivo primordial del proyecto, aplica al tesorero del comité encargado de brindar el servicio o a la persona que designada que se le brinde las facultades de administrar el sistema, misma que hace de recaudadora y es responsable de los cobros efectuados mediante la aplicación.

Al llevar a cabo el proyecto se obtendrán grandes beneficios como:

- Mayor comodidad al usuario del sistema para generar cobros del servicio desde la comodidad de un computador o dispositivo móvil de una manera ágil, calculando automáticamente los valores a pagar del usuario del servicio.
- Podrán realizar el ingreso de las cifras consumidas por los usuarios mes a mes.
- Tener un registro con los usuarios del servicio.
- Generar boletas o comprobantes de pago que serán enviados al correo del cliente o imprimir si se desea.
- Mediante el software se podrá notificar las deudas pendientes que tiene el usuario del servicio mediante correo electrónico.

1.3. Personal involucrado

Nombre	Edhisson Sanmartin
Rol	Analista, diseñador y programador
Categoría Profesional	Estudiante
Responsabilidad	Análisis de información, diseño y programación del SIS-I
Información de contacto	easanmartinf@unl.edu.ec

1.4. Definiciones, acrónimos y abreviaturas

Nombre	*Descripción*
ERS	Especificación de Requisitos de Software
SAAS	Software como servicio
UNL	Universidad Nacional de Loja
RF	Requerimiento Funcional
RNF	Requerimiento No Funcional

1.5. Referencias

Título del Documento	Referencia
Standard IEEE 830 - 1998	IEEE

2. Descripción general

2.1. Perspectiva del producto

El Módulo de Homologación de estudios usando el mecanismo de Análisis Comparativo a implementar será dependiente ya que necesariamente requiere de otro sistema, modulo para su correcta eficiencia.

2.2. Características de los usuarios

Tipo de usuario	Administrador
Formación	Bachiller
Actividades	Control y manejo del sistema en general

2.3. Restricciones

Puesto que se trata de un SaaS, la nube en la que estará alojado el software aún no se encuentra definida, otra limitación que se presenta es que para que funcione el servicio se deberá contar siempre con una conexión a internet.

2.4. Suposiciones y dependencias

- Se asume que los requisitos aquí descritos son estables.
- Los equipos en los que se vaya a ejecutar el sistema deben cumplir los requisitos antes indicados para garantizar una ejecución correcta de la misma.

3. Requisitos específicos

El aplicativo al ser de uso específico contará con el único usuario que se encargará de realizar las actividades y que hará las veces de **administrador,** pudiendo así contar con uno o más administradores que accedan al sistema.

3.1. Requerimientos Funcionales

3.1.1. Inicio de sesión.

***Tabla 22**. Requerimiento Funcional - Inicio de sesión.*

Identificación del requerimiento:	**RF1.1**
Nombre del Requerimiento:	El Software contará con un inicio de sesión para validar credenciales del administrador.
Descripción del requerimiento:	Para ingresar al sistema y realizar cualquiera de las actividades, la persona encargada deberá validar sus credenciales.
Tipo	☐ Requisito ☒ Restricción
Prioridad del requerimiento:	☒ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional

3.1.2. Ingreso de usuarios.

***Tabla 23.** Requerimiento Funcional - Ingreso de usuarios.*

Identificación del requerimiento:	**RF1.2**
Nombre del Requerimiento:	Ingreso de usuarios.
Descripción del requerimiento:	El Administrador del Software será la persona encargada de ingresar (crear, editar, eliminar) los usuarios del servicio. Para el ingreso de Usuarios se requiere de los siguientes datos: Nombres y Apellidos o razón social, Cédula o ruc, correo electrónico, numero de celular.
Tipo	☒ Requisito ☐ Restricción
Prioridad del requerimiento:	☒ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional

3.1.3. Cobros del servicio.

Tabla 24. *Requerimiento Funcional - Cobros del servicio.*

Identificación del requerimiento:	**RF1.3**
Nombre del Requerimiento:	Cobros por el servicio.
Descripción del requerimiento:	El Administrador realizar los cobros del servicio de cada uno de los usuarios, para esto antes debió haber ingresado al sistema las cifras del consumo. Para realizar el cobro se deberá verificar si el usuario a consumido su cifra o aumentarle el adicional, así mismo se deberá verificar si posee deudas de los meses anteriores.
Tipo	[X] Requisito [ ] Restricción
Prioridad del requerimiento:	[X] Alta/Esencial [ ] Media/Deseado [ ] Baja/ Opcional

3.1.4. Ingreso de cifras.

Tabla 25. *Requerimiento Funcional - Ingreso de cifras.*

Identificación del requerimiento:	**RF1.4**
Nombre del Requerimiento:	Ingresar cifras de los usuarios.
Descripción del requerimiento:	El Administrador deberá ingresar cada mes al sistema las cifras que se han recolectado, esto servirá para luego poder calcular los cobros del servicio.
Tipo	[X] Requisito [ ] Restricción
Prioridad del requerimiento:	[X] Alta/Esencial [ ] Media/Deseado [ ] Baja/ Opcional

3.1.5. Comprobante de pago.

Tabla 26. *Requerimiento Funcional - Comprobante de pago.*

Identificación del requerimiento:	**RF1.5**
Nombre del Requerimiento:	El sistema emitirá un comprobante de pago.
Descripción del requerimiento:	El administrador una vez realizado el cobro del servicio, el software emitirá un comprobante o recibo de pago que se enviará al usuario del servicio por correo electrónico o impreso.
Tipo	[X] Requisito [ ] Restricción
Prioridad del requerimiento:	[ ] Alta/Esencial [X] Media/Deseado [ ] Baja/ Opcional

3.1.6. Configuraciones.

Tabla 27. *Requerimiento Funcional - Configuraciones.*

Identificación del requerimiento:	**RF1.6**
Nombre del Requerimiento:	Configuraciones del sistema.
Descripción del requerimiento:	El administrador tendrá la posibilidad de editar sus credenciales. Así mismo tendrá la posibilidad de hacer configuraciones al sistema (cambiar la cifra máxima de consumo de cada usuario, cambiar el valor de cada cifra).
Tipo	[X] Requisito [ ] Restricción
Prioridad del requerimiento:	[X] Alta/Esencial [ ] Media/Deseado [ ] Baja/ Opcional

3.1.7. Notificar deudas.

Tabla 28. *Requerimiento Funcional - Notificar deudas.*

Identificación del requerimiento:	**RF1.7**
Nombre del Requerimiento:	Notificar deudas pendientes de los usuarios.
Descripción del requerimiento:	El usuario tendrá la posibilidad de notificar a los usuarios las deudas pendientes del servicio mediante un correo electrónico.
Tipo	[X] Requisito [ ] Restricción
Prioridad del requerimiento:	[ ] Alta/Esencial [ ] Media/Deseado [X] Baja/ Opcional

3.2. Requerimientos No Funcionales.

3.2.1. Compatibilidad

Tabla 29. *Requerimiento No Funcional - Compatibilidad.*

Identificación del requerimiento:	**RNF01**
Nombre del Requerimiento:	Navegadores compatibles.
Características:	La aplicación podrá ser usada en los diferentes entornos de navegación web disponibles.
Descripción del requerimiento:	La aplicación web debe tener una interfaz sencilla y fácil de usar en cualquier navegador disponible.
Prioridad del requerimiento: **Alta**	

3.2.2. Desempeño

Tabla 30. *Requerimiento No Funcional - Desempeño.*

Identificación del requerimiento:	**RNF02**
Nombre del Requerimiento:	Desempeño
Características:	La aplicación permitirá a los usuarios realizar tareas de manera eficiente.
Descripción del requerimiento:	Asegurar el funcionamiento de la aplicación a los distintos usuarios. En este sentido la información almacenada o los registros creados pueden ser solicitados y actualizados de manera constante y sincrónica, sin afectar el tiempo de respuesta.
Prioridad del requerimiento:	**Alta**

3.2.3. Disponibilidad

Tabla 31. *Requerimiento No Funcional - Disponibilidad.*

Identificación del requerimiento:	**RNF03**
Nombre del Requerimiento:	Disponibilidad – El Sistema estará en funcionamiento las 24/7
Características:	La aplicación tiene que estar funcionando las 24 horas al día y los 7 días a la semana.
Descripción del requerimiento:	La disponibilidad de la aplicación debe brindar a los usuarios un nivel de servicio continuo los 7 días de la semana, las 24 horas del día, asegurando que los responsables de su desarrollo cuenten con un plan en caso de falla de cualquiera de sus componentes.
Prioridad del requerimiento: **Alta**	

3.2.4. Fiabilidad

Tabla 32*. Requerimiento No Funcional - Fiabilidad.*

Identificación del requerimiento:	**RNF04**
Nombre del Requerimiento:	Fiabilidad
Características:	La aplicación velará por la seguridad del usuario respecto a la información procesada en el sistema.
Descripción del requerimiento:	Garantizar la seguridad del sistema en relación con la información almacenada, los datos personales y contraseñas.
Prioridad del requerimiento: **Alta**	

3.2.5. Tiempo de respuesta

Tabla 33. *Requerimiento No Funcional - Tiempo de respuesta.*

Identificación del requerimiento:	**RNF05**
Nombre del Requerimiento:	Tiempo de respuesta
Características:	La aplicación responderá de forma rápida y precisa a sus consultas lo antes posible.
Descripción del requerimiento:	El tiempo de respuesta será instantáneo en la realización de acciones del sistema, para lo cual se necesitará contar con una conexión a internet necesaria con características optimas.
Prioridad del requerimiento: **Alta**	

3.2.6. Usabilidad

Tabla 34. *Requerimiento No Funcional - Usabilidad.*

Identificación del requerimiento:	**RNF06**
Nombre del Requerimiento:	Usabilidad
Características:	La aplicación será fácil de usar y tendrá una interfaz simple, sencilla e intuitiva.
Descripción del requerimiento:	La aplicación web será fácil de utilizar con una pequeña guía de usuario incluida, se brindará una capacitación a los usuarios sobre cómo usarla de ser necesario
Prioridad del requerimiento: **Alta**	

Anexo 3. Arquitectura del software como servicio para la automatización del cobro de consumo de agua.

Proyecto: Aplicación Web para la automatización del cobro de consumo de agua potable en el Gobierno Autónomo Descentralizado Rural de la Parroquia Milagro del Cantón Atahualpa.

Documento Arquitectura de Software

1. Introducción

El presente documento tiene como objetivo presentar la arquitectura de la Aplicación Web para la automatización del cobro de consumo de agua potable en el Gobierno Autónomo Descentralizado Rural de la Parroquia Milagro del Cantón Atahualpa. Es sumamente importe diseñar una buena arquitectura debido a que presenta el comportamiento y la interacción del sistema, de igual forma ayuda a tener una clara idea del software que se va a desarrollar.

2. Propósito

El propósito del presente documento es definir la arquitectura del software basándose en un modelo arquitectónico 4+1 el cual mediante diferentes vistas detalla el comportamiento de la aplicación.

3. Alcance

El modelo arquitectónico 4+1 propone realizar una arquitectura en base a las vistas, donde se describirán la vista de escenarios, la vista lógica, la vista física, la vista de despliegue y finalmente la vista de procesos.

4. Referencias

Tabla 35. *Referencias de la arquitectura de software.*

Referencia	Título del documento
Anexo 2	Especificación de requisitos. – IEEE 830
Modelo	Arquitectura de software 4+1

5. Vista global

En el presente documento se detalla cómo se encuentra estructura la arquitectura de la aplicación para el cobro del consumo de agua tomando como referencia el modelo 4+1, el cual hace uso de múltiples vistas, la vista de escenarios a través de los casos de uso muestra cada una de las actividades del usuario, la vista de lógica a través del diagrama de clases muestra las funcionalidades, la vista física a través del diagrama de despliegue muestra los componentes físicos, la vista de despliegue con la ayuda de los diagramas de componentes muestra los diferentes componentes al desarrollador y la vista de procesos con la ayuda del diagrama de secuencia muestra la secuencia de cada uno de los procesos del sistema.

6. Representación de la arquitectura

El modelo arquitectónico 4+1 propone 5 vistas las mismas que se detallan en la siguiente tabla.

Tabla 36. *Vistas de la arquitectura 4+1.*

Vista	Elemento modelado	Descripción
Vista de escenarios	Casos de uso	Muestra las actividades que debe realizar el administrador para llevar a cabo el proceso de cobro del servicio de agua.
Vista lógica	Diagrama de clases	Muestras las funcionalidades de la aplicación.
Vista física	Diagrama de despliegue	Muestra los componentes físicos del sistema
Vista de despliegue	Diagrama de componentes	Describe los diferentes componentes para que el desarrollador tenga una clara comprensión del sistema.
Vista de procesos	Diagrama de secuencia	Muestra la secuencia de cada una de los procesos del sistema.

7. Objetivos de la arquitectura

El software cumplirá con los siguientes puntos:

- **Rendimiento:** la aplicación tendrá tiempos de respuesta inferiores a 6 segundos en cada uno de sus procesos.
- **Seguridad:** el framework Django cuenta con algunas seguridades por defecto, entre otras.
    - Protección en los formularios post (CSRF).
    - Protección a inyecciones SQL.
    - El software funcionara bajo el protocolo HTTPS.
    - La aplicación restringirá el acceso a cualquier usuario que no esté autorizado.
- **Fiabilidad:** los datos ingresados por los usuarios serán encriptados y no se los usara para otros fines externos a la aplicación.
- **Disponibilidad:** la aplicación estará disponible siempre para el usuario, 24 horas al día los 365 días a la semana, esto estará restringido al servicio de aplicaciones, hosting.
- **Portabilidad:** el sistema web estará disponible dentro de cualquier navegador que cuente con una conexión a internet, tanto en computadores, smartphones o tables.
- **Usabilidad:** el software contara con una interfaz sencilla y amigable que no complicara en sus funcionalidades al usuario.

8. Vista de escenarios

En esta vista se muestran los casos de uso, mismo que presentan la iteración del usuario del sistema con el sistema.

Diagrama de casos de uso

El diagrama de caso de uso muestra el actor directo que vendría a ser el administrador o usuario del sistema y el actor indirecto que vendría a ser el usuario del servicio. El administrador tiene las capacidades de realizar las siguientes tareas en el sistema:

- Ingresar al sistema
- CRUD de usuarios del servicio
- Buscar usuarios
- Ingresar cifras consumidas
- Realizar cobros del servicio
- Emitir recibos
- Notificas deudas a los usuarios.

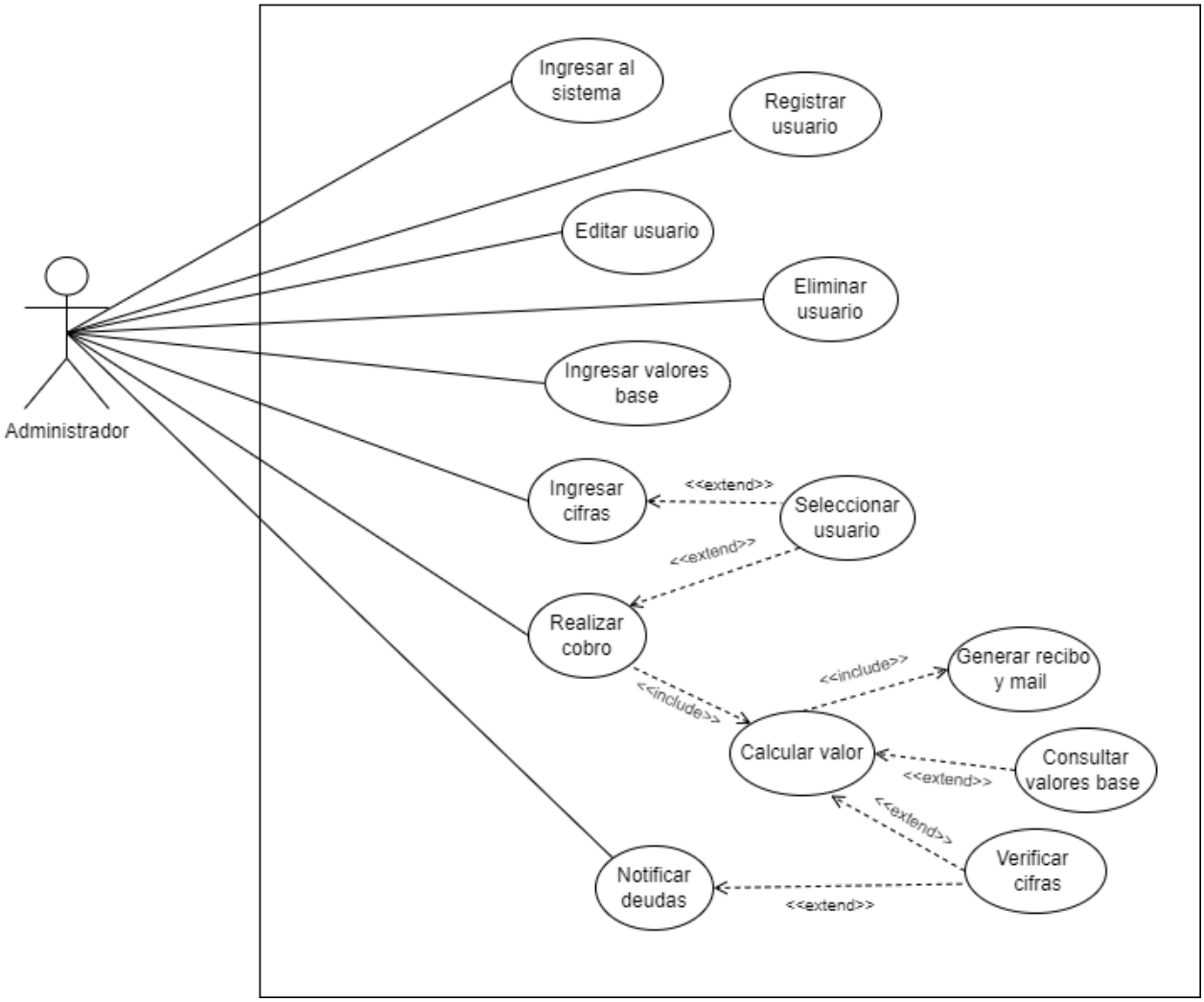


Figura 49. *Diagrama de casos de uso de la aplicación.*

9. Vista lógica

Mediante esta vista se muestran las funcionalidades de la aplicación a través del uso del diagrama de clases.

Diagrama de clases

El diagrama de clases que se visualiza en la *figura 50* nos el comportamiento de cada una de las clases con cada uno de sus atributos.

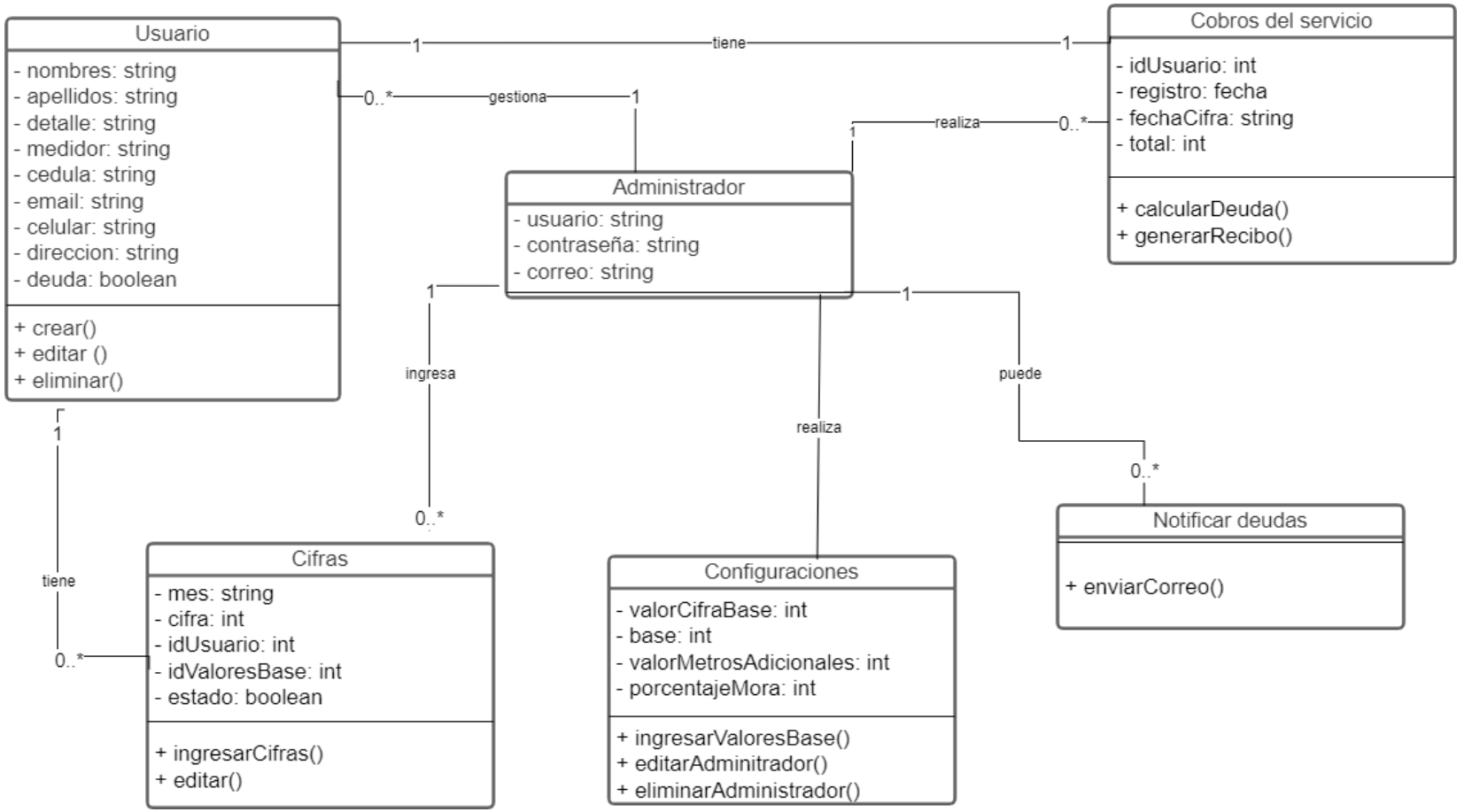


Figura 50. *Diagrama de clases de la aplicación.*

10. Vista física

La vista física hace referencia a los componentes del sistema y hace uso de los diagramas de despliegue.

Diagrama de despliegue

En la figura a continuación se puede visualizar como se organiza cada uno de los componentes de la aplicación, la interacción entre el cliente, el servidos de aplicaciones y el servidor de bases de datos.

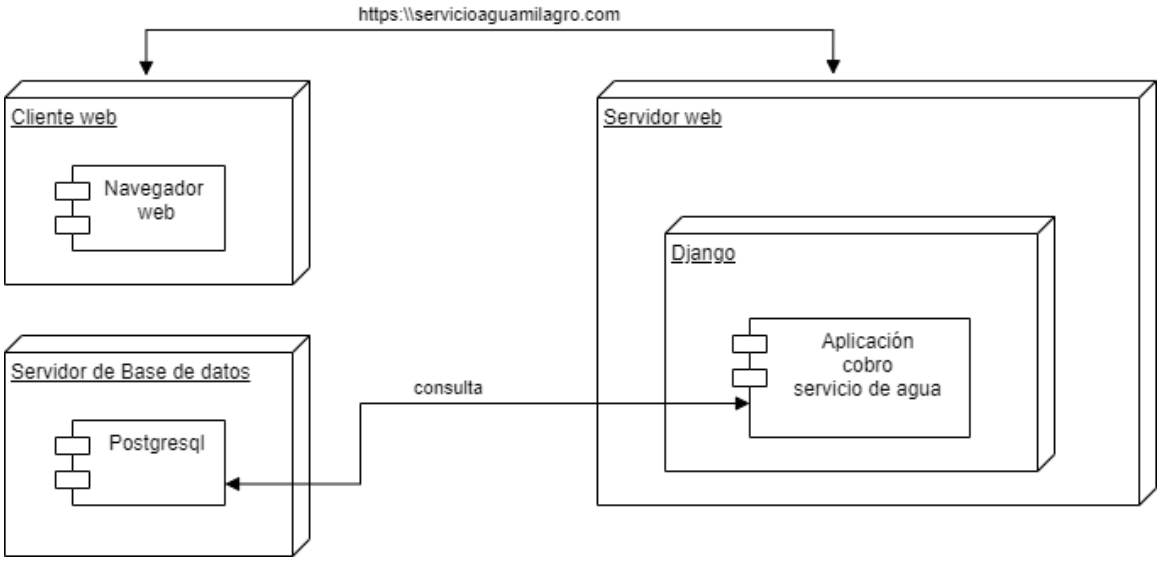


***Figura 51**. Diagrama de despliegue.*

11. Vista de despliegue

A través de los diagramas de componentes describe los diferentes componentes para que el desarrollador tenga una clara comprensión del sistema

Diagrama de componentes

En la **figura 52** se muestra cada uno de los componentes que componen al sistema del cobro por consumo de agua, estos componentes se interrelacionan entre sí para cumplir todas las funcionalidades requeridas por el usuario, django ofrece un componente exclusivo que nos facilita la conexión hacia las bases de datos.

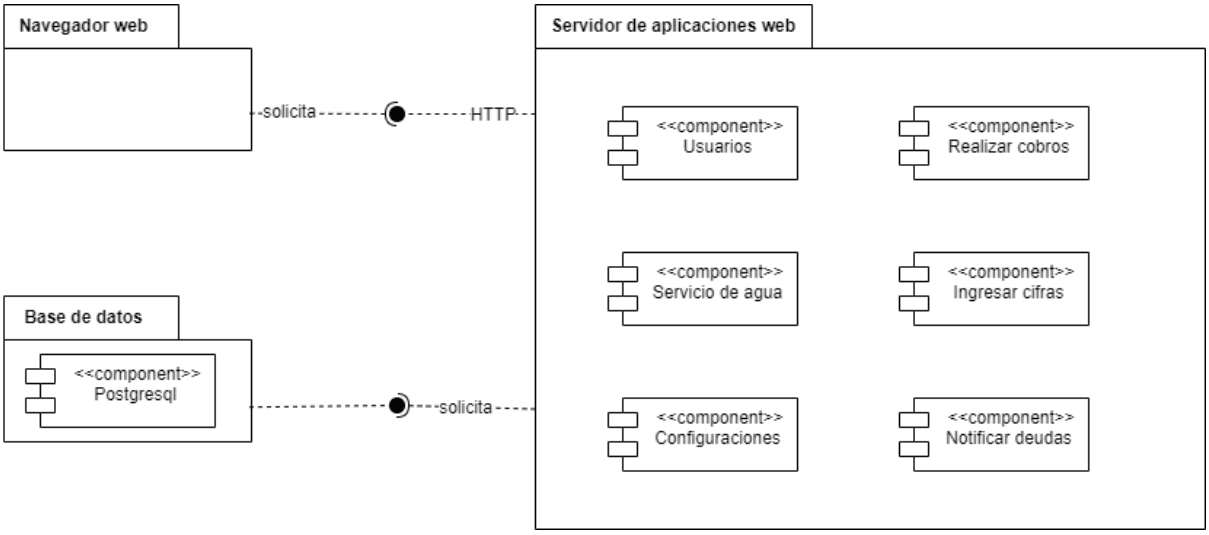


Figura 52. *Diagrama de componentes.*

12. Vista de procesos

A través de los diagramas de actividades se muestra cada uno de los procesos con los que cuenta la aplicación web.

Diagrama de actividades

En la **figura 53** se muestra el diagrama de actividades para realizar un cobro por el servicio, especificando cada uno de los pasos que debe de seguir la aplicación para el cumplimiento de este proceso.

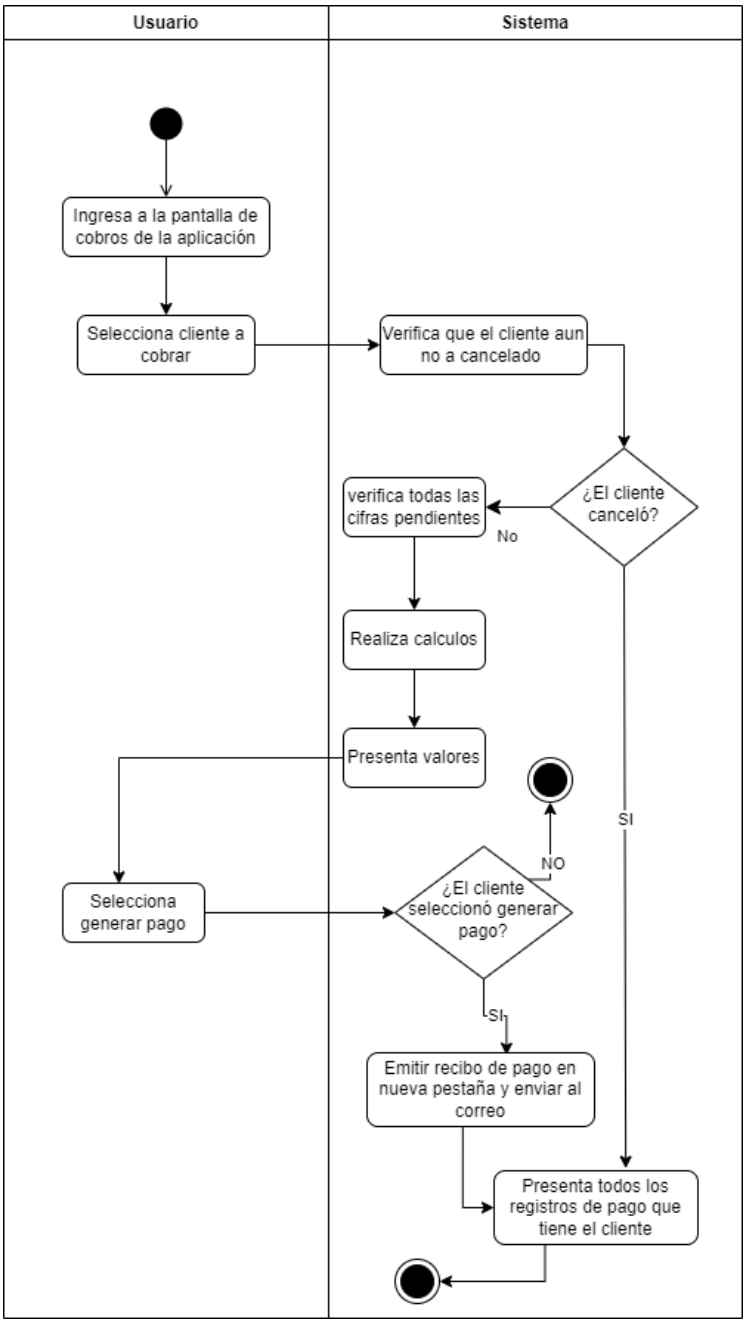


Figura 53. *Diagrama de actividades para realizar un cobro del servicio.*

En siguiente figura se muestra el diagrama de actividades para ingresar cifras de cada uno de los usuarios del servicio, detallando cada uno de los pasos que debe de seguir la aplicación para el cumplimiento de esta actividad.

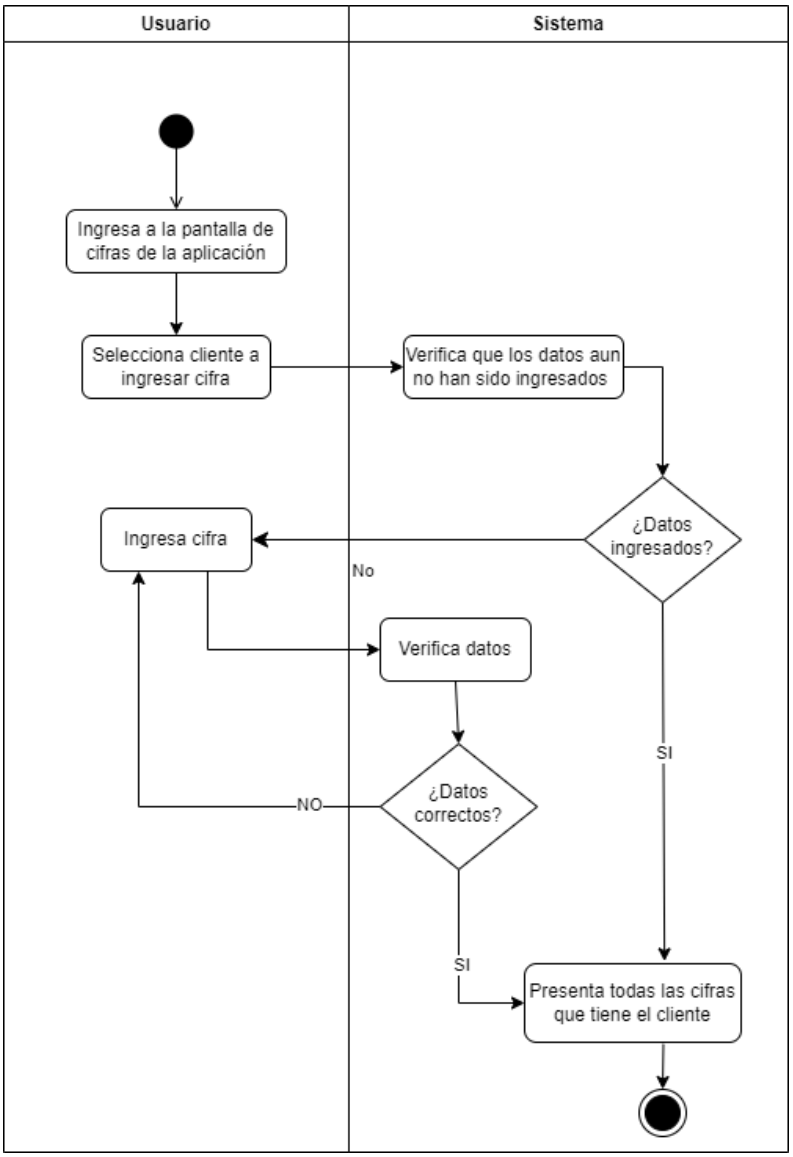


Figura 54. *Diagrama de actividades para el ingreso de cifras.*

A continuación se puede visualizar el diagrama de actividades para ingresar clientes del servicio del servicio a la aplicación web, detallando cada uno de los procesos del software para tener una ejecución correcta.

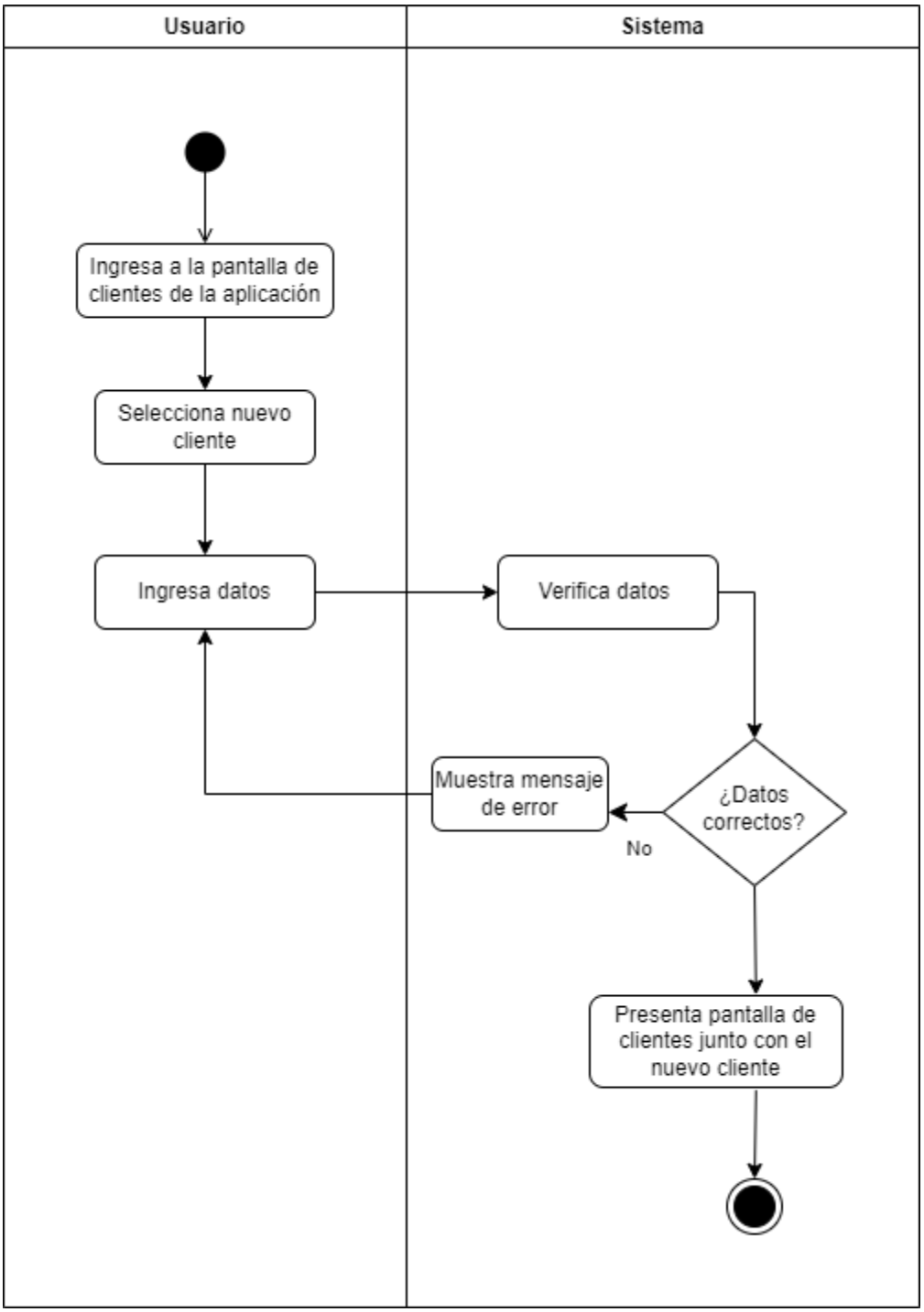


Figura 55. *Diagrama de actividades para ingresar clientes del servicio.*

En la siguiente figura se puede visualizar el diagrama de actividades para ingresar nuevos valores base al servicio, se detalla cada una de las actividades que debe cumplir el software para tener una ejecución correcta de la aplicación.

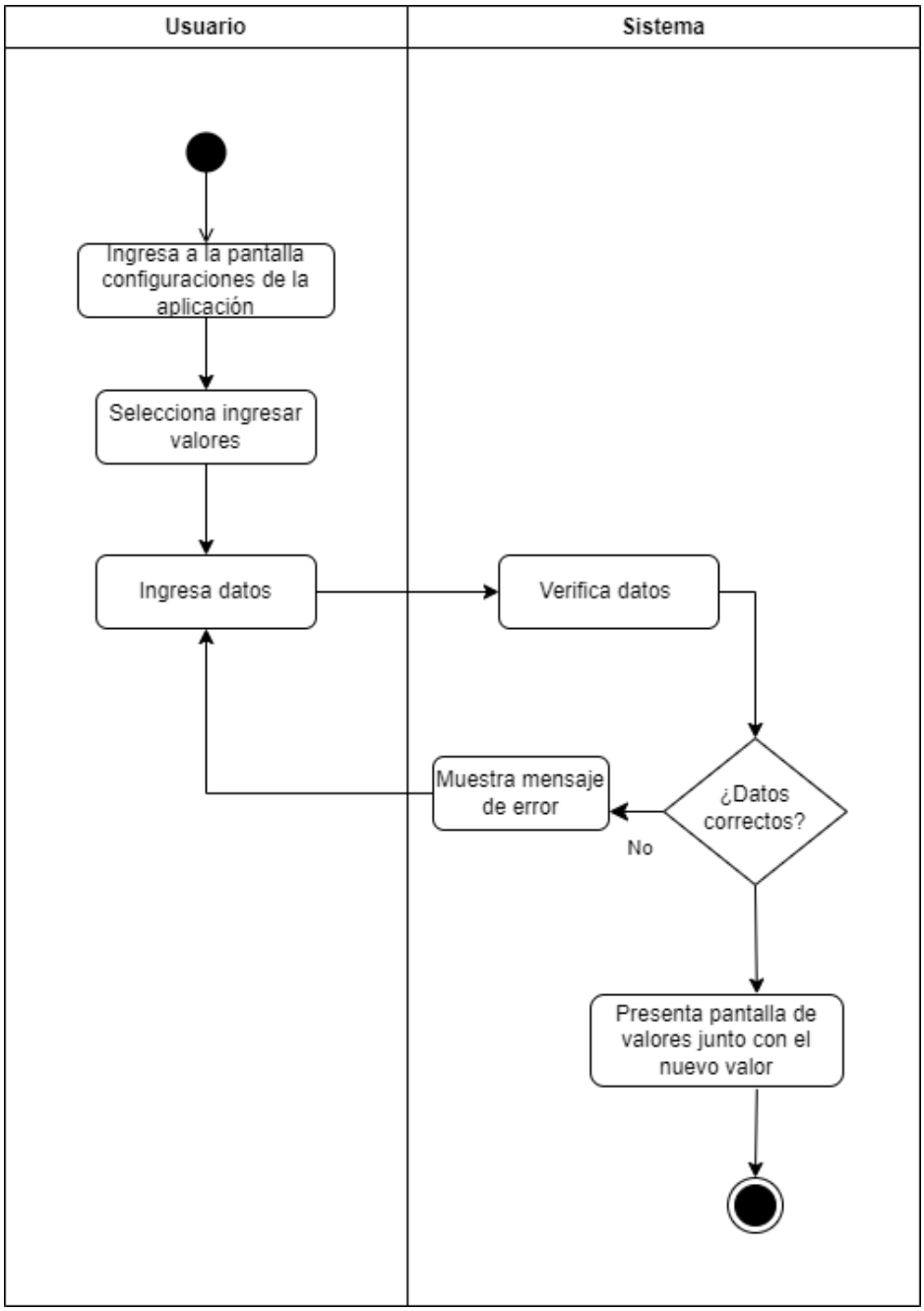


Figura 56. *Diagrama de actividades para ingresar nuevos valores base del servicio.*

En la siguiente ilustración se puede observar en detalle el diagrama de actividades para iniciar sesión en la aplicación web, detallando cada uno de los procesos a cumplir para tener una ejecución correcta del software.

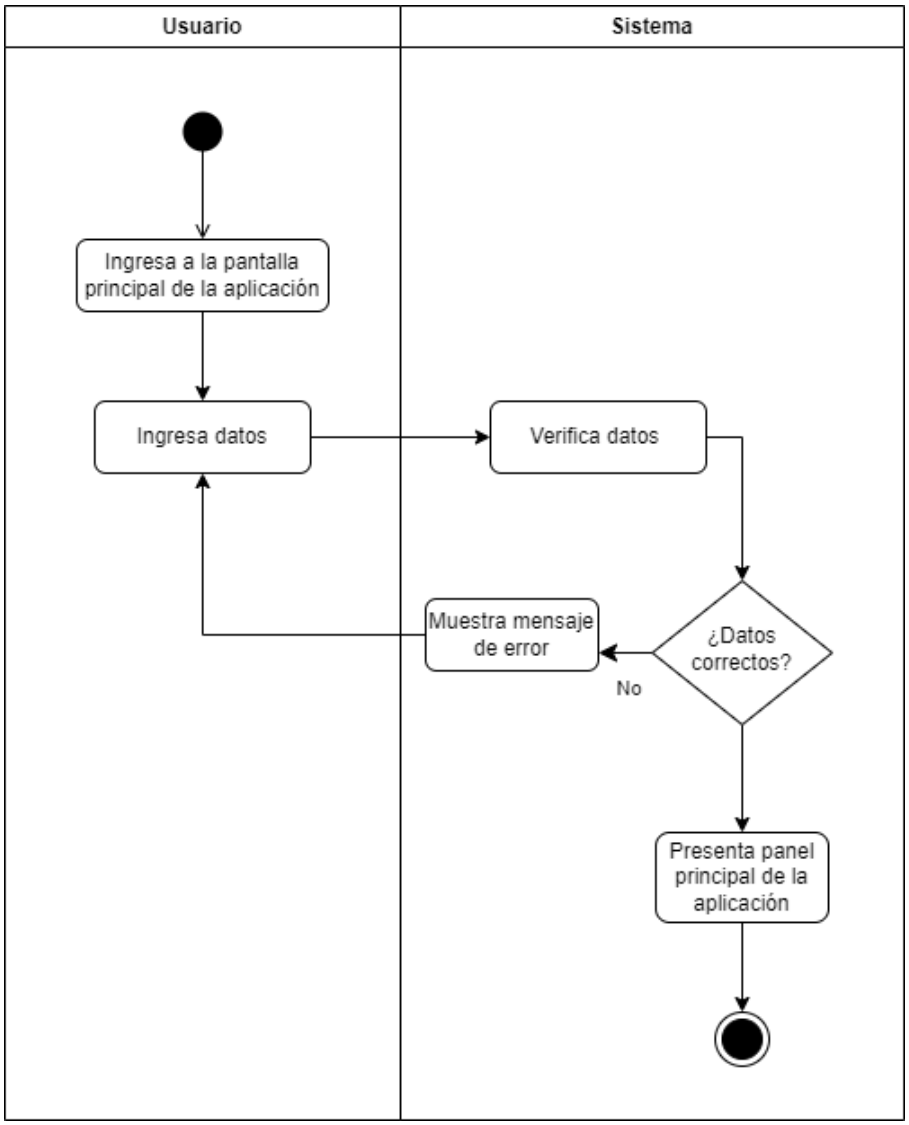


Figura 57. *Diagrama de actividades para iniciar sesión en la aplicación web.*

En la ilustración siguiente se puede visualizar en detalle el diagrama de actividades para notificar las deudas de los usuarios por el servicio en la aplicación web, mostrando en detalle cada uno de los procesos a seguir para una adecuada ejecución del software.

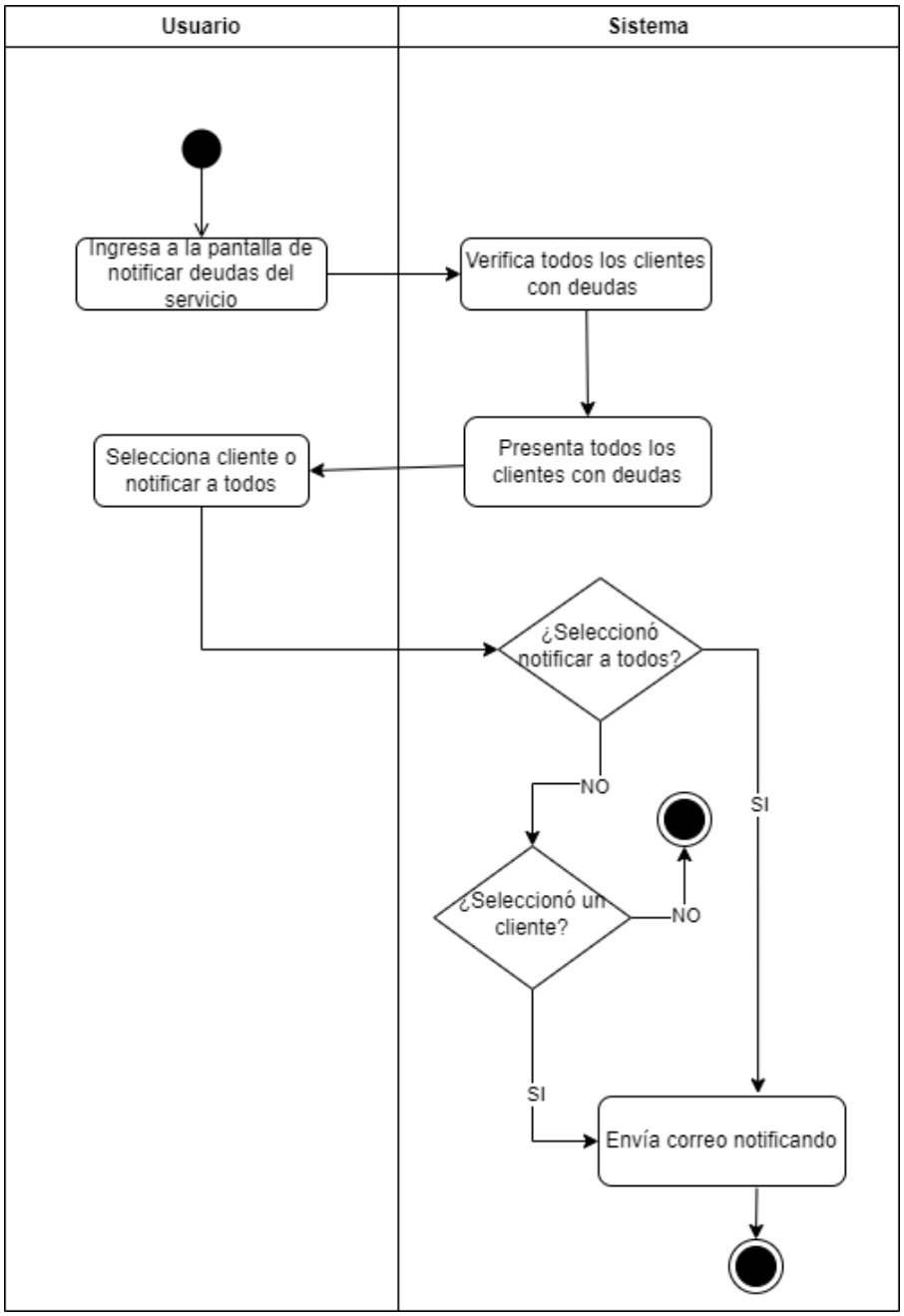


Figura 58. *Diagrama de actividades para notificar deudas pendientes por el servicio.*

Diagrama de secuencia

Para complementar el entendimiento de los procesos se optó por implementar los diagramas de secuencia. En la **figura 59** se muestra la secuencia para poder realizar el crud de cada uno de los usuarios del servicio.

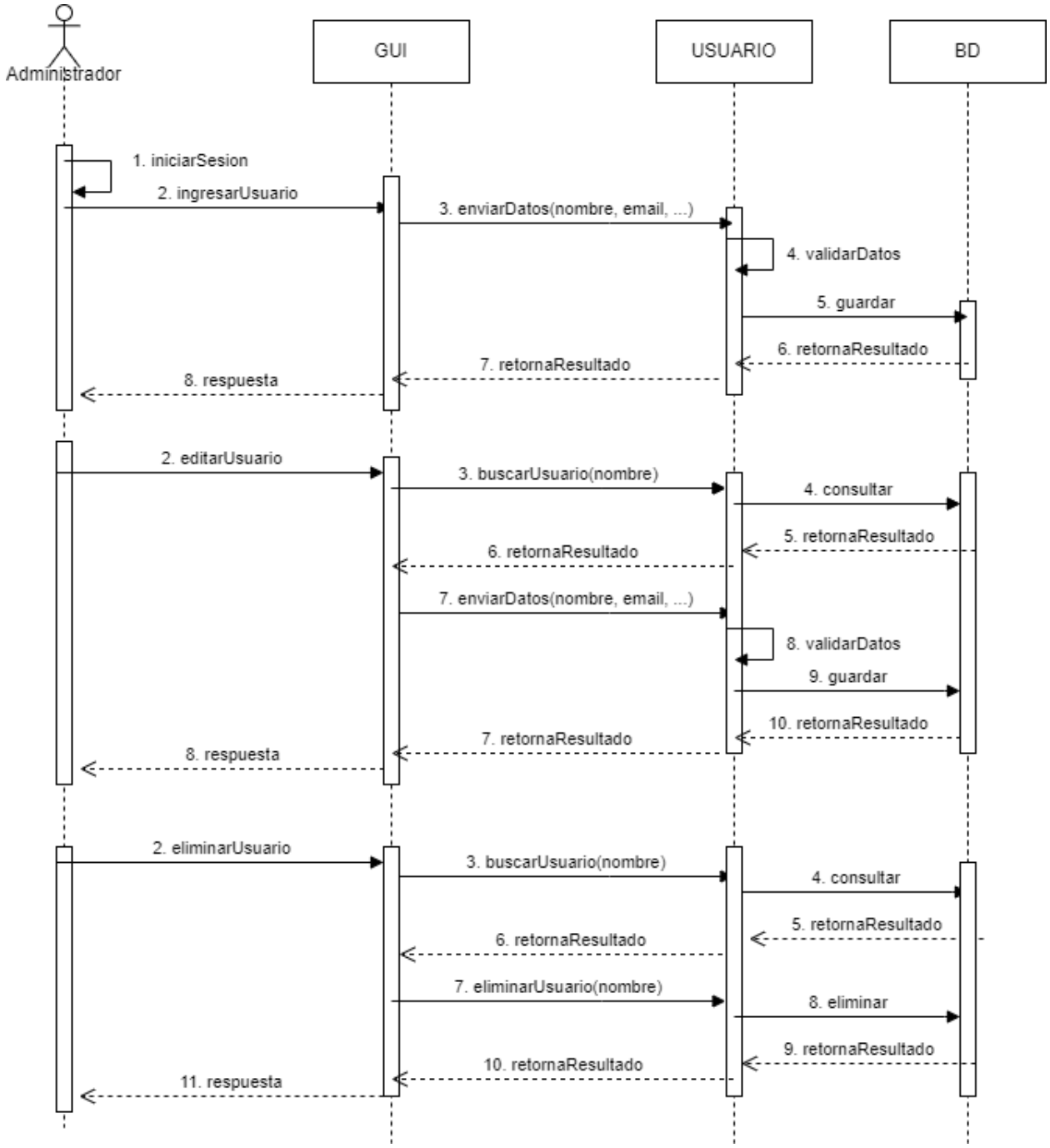


Figura 59. *Diagrama de secuencia para el CRUD de usuarios del sistema.*

En la **figura 60** se muestra la secuencia que tiene que realizar la aplicación para poder realizar un cobro a un determinado usuario.

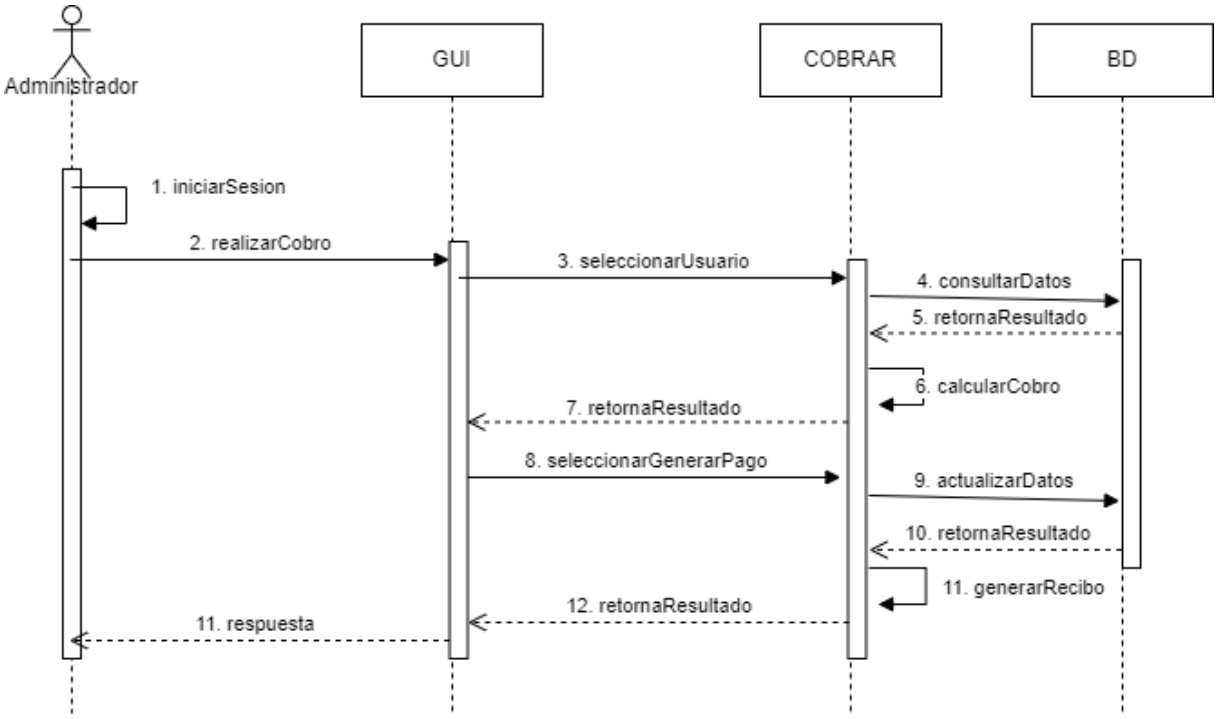


Figura 60. *Diagrama de secuencia para realizar el cobro del servicio.*

En la **figura 61** se muestra la secuencia que se realiza para poder ingresar una cifra consumida por el usuario en la aplicación.

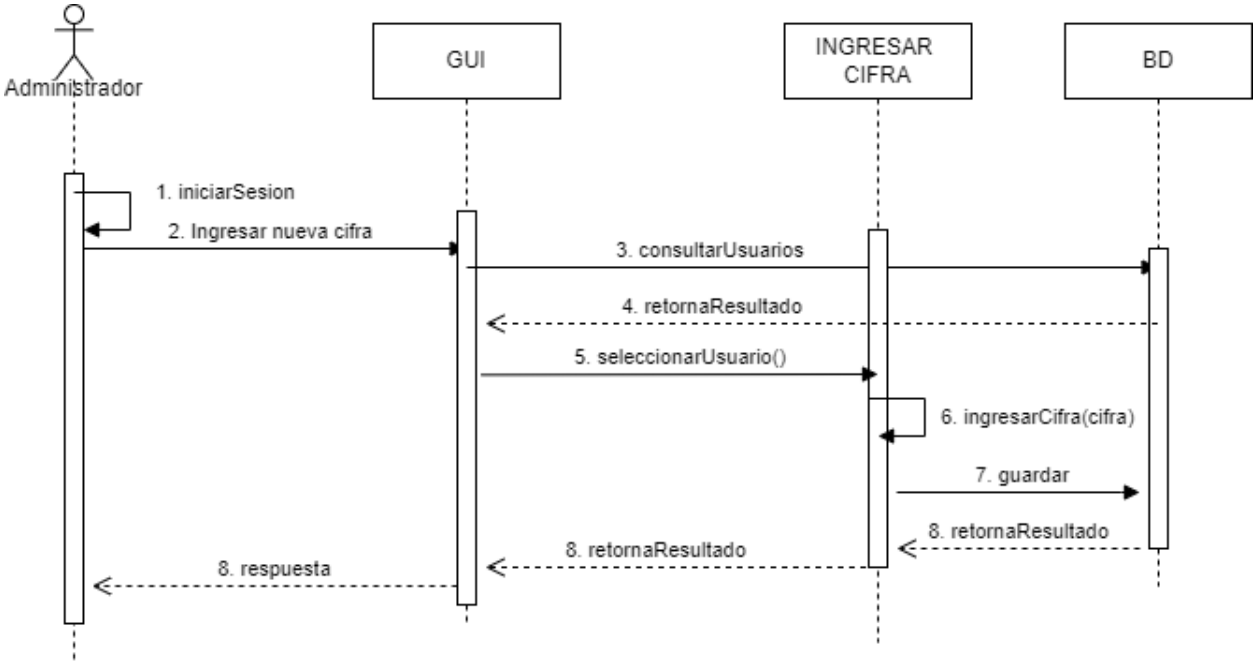


Figura 61. *Diagrama de secuencia para ingresar cifras consumidas.*

En la **figura 62** se muestra la secuencia para poder acceder a la aplicación mediante el inicio de sesión.

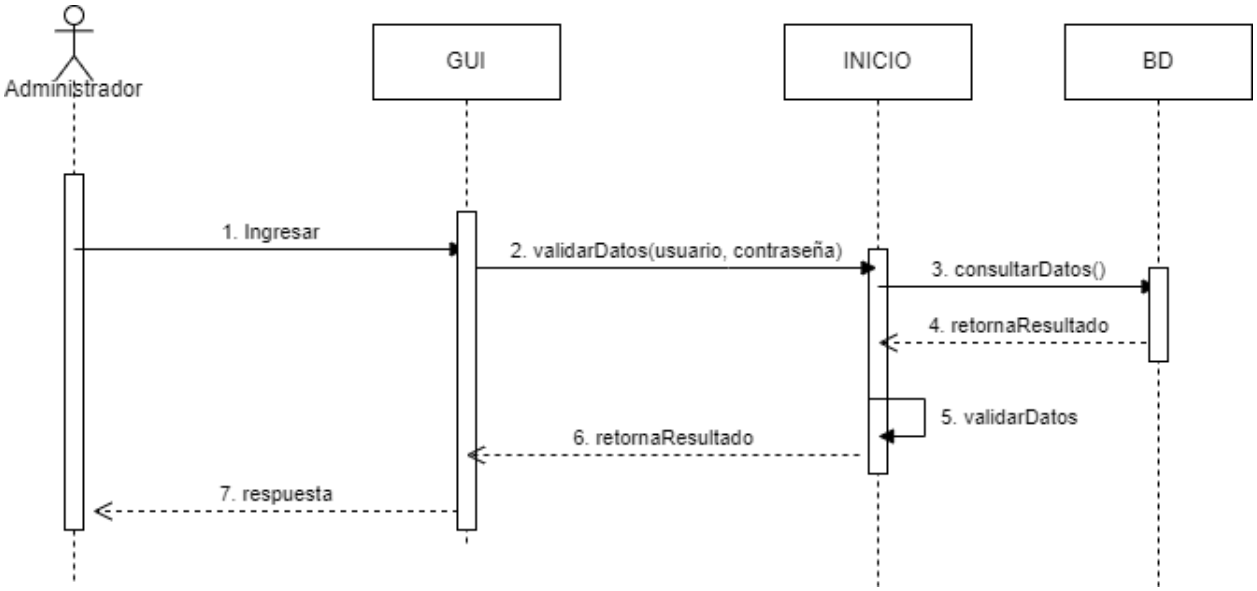


Figura 62. Diagrama de secuencia para iniciar sesión en la aplicación.

13. Arquitectura de la Aplicación

La arquitectura utilizada en la Aplicación Web es cliente/servidor donde los demandantes llamados clientes realizan peticiones a un proveedor de recursos llamado servidor que a su vez realiza consultas a una Base de Datos, en la siguiente figura se muestra en detalle la arquitectura.

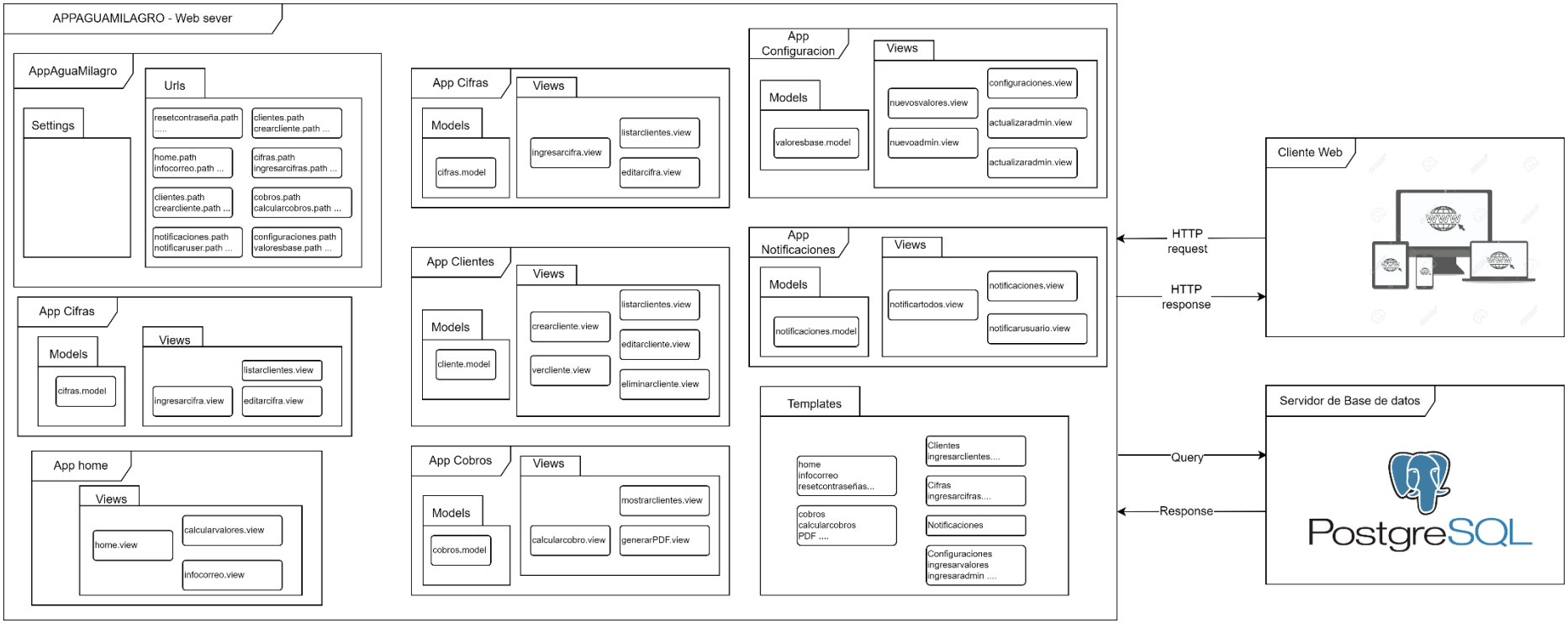


Figura 63. *Arquitectura de la Aplicación Web para el cobro del servicio de agua potable.*

Anexo 4. Prototipos del software como servicio para el cobro por el consumo de agua.

Prototipado de la aplicación web para el cobro por el consumo de agua de la Parroquia Milagro

Proyecto: Aplicación Web para la automatización del cobro de consumo de agua potable en el Gobierno Autónomo Descentralizado Rural de la Parroquia Milagro del Cantón Atahualpa.

PROTOTIPADO DE LAS INTERFACES DE LA APLICACIÓN WEB PARA EL COBRO POR EL CONSUMO DE AGUA DE LA PARROQUIA MILAGRO

El presente documento contiene las interfaces de cada una de las pantallas de las interfaces de la aplicación para el cobro por el consumo de agua, para su creación se usó la herramienta pencil, y se validó el diseño con el cliente.

Pantalla de inicio de sesión

Observaciones:

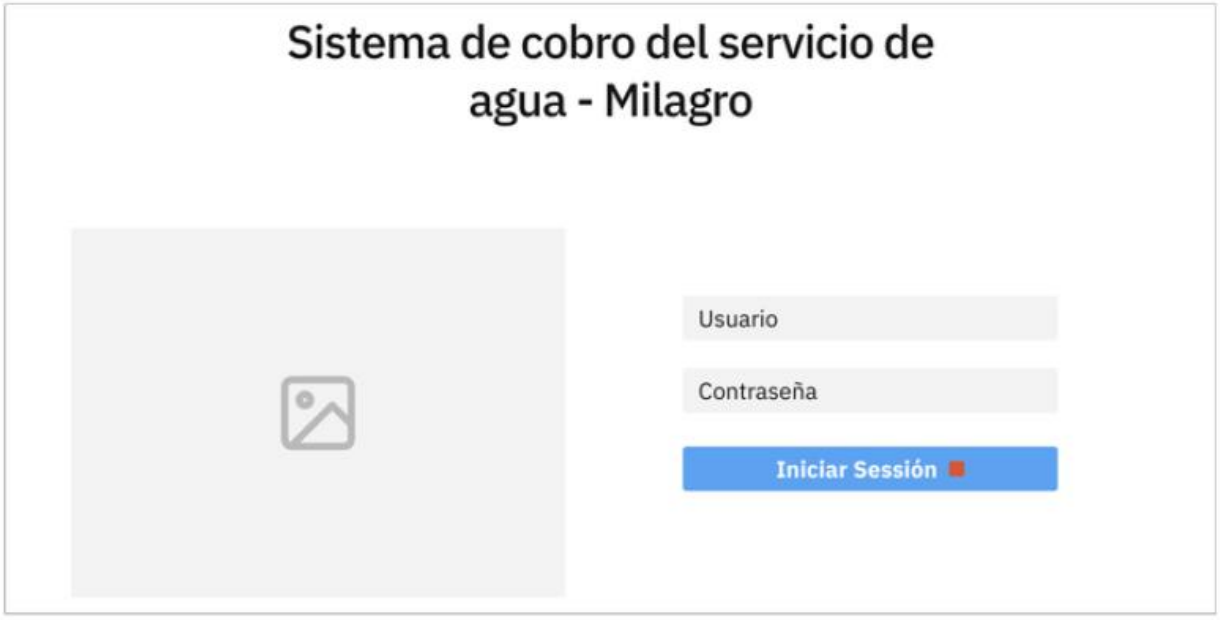


Figura 64. *Pantalla de inicio de sesión.*

Pantalla de inicio de la aplicación

Observaciones:

Figura 65. *Pantalla de inicio del sistema.*

Pantalla de cobros del servicio

Observaciones:

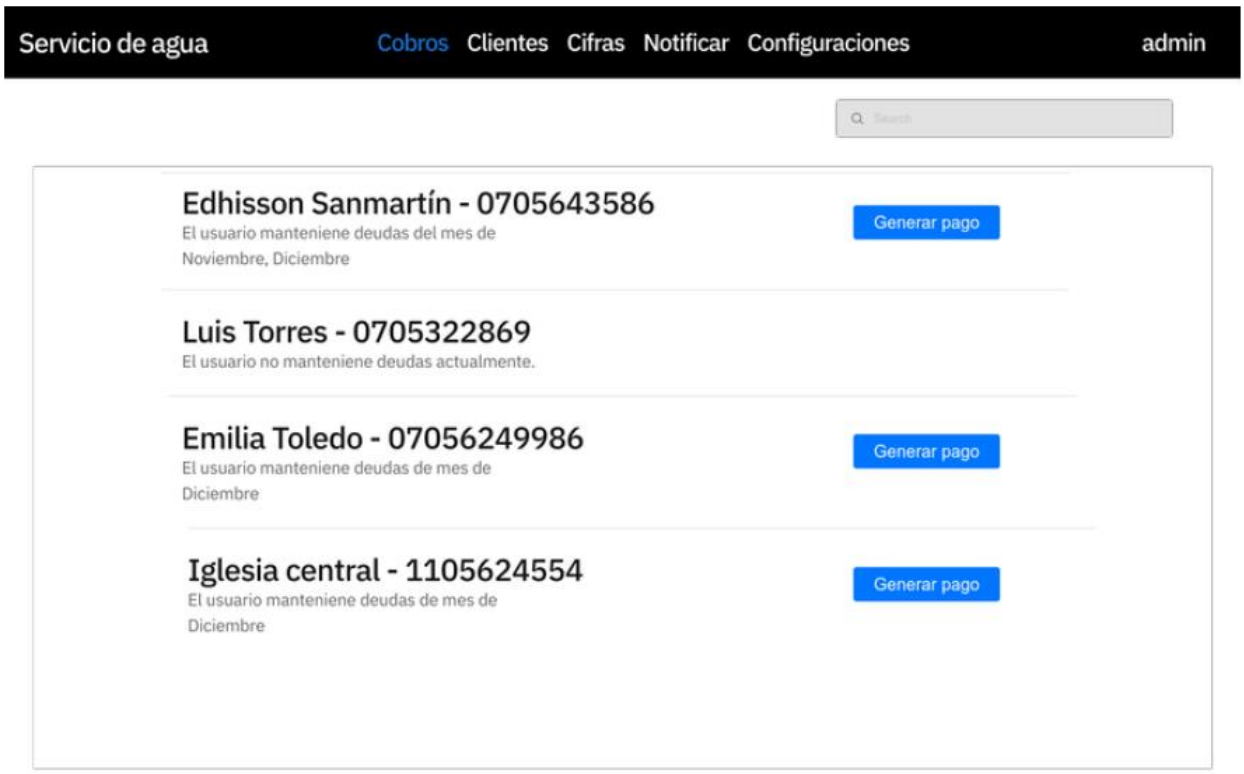


Figura 66. *Pantalla de cobros del servicio*

Pantalla de recibo de cobro

Observaciones:

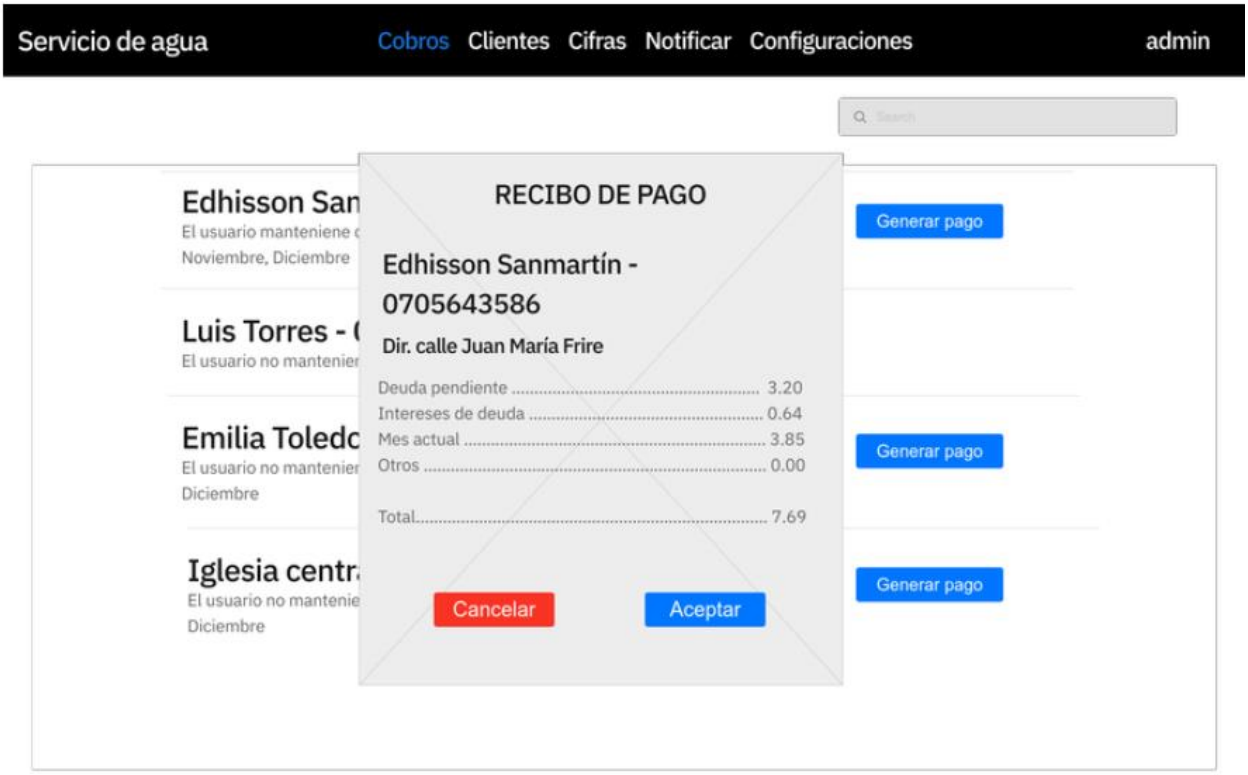


Figura 67. *Pantalla del recibo generado por el cobro del servicio.*

Pantalla de clientes

Observaciones:

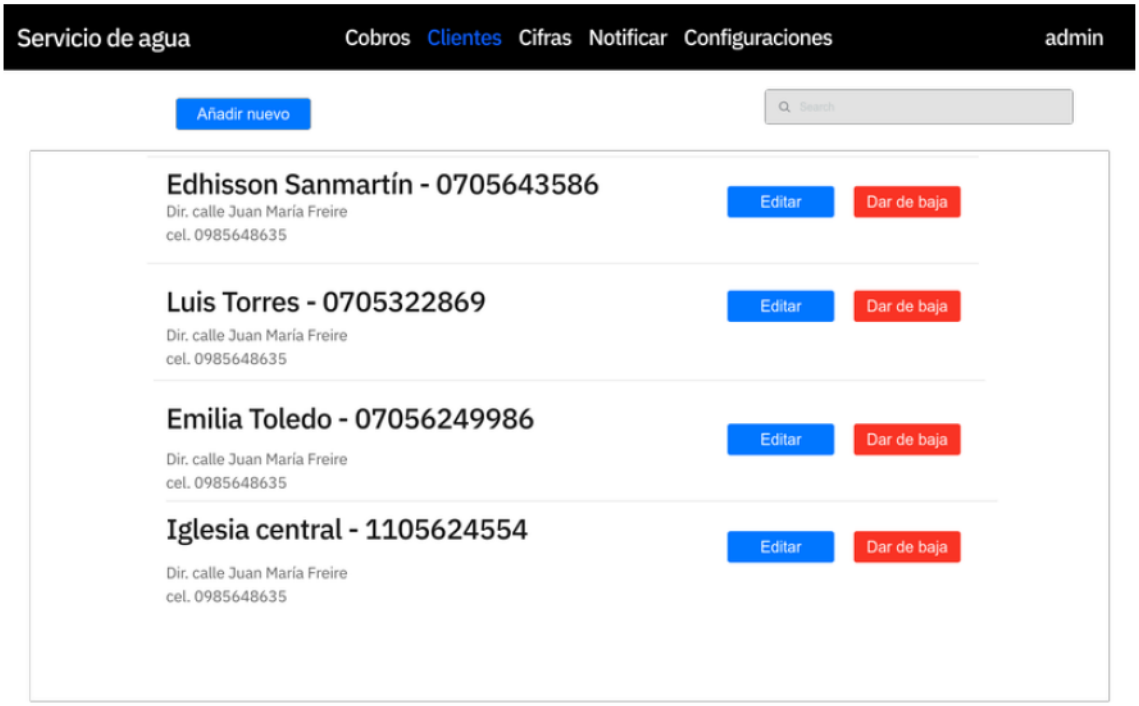


Figura 68. *Pantalla de gestión de los clientes del servicio.*

Pantalla de cifras consumidas

Observaciones:

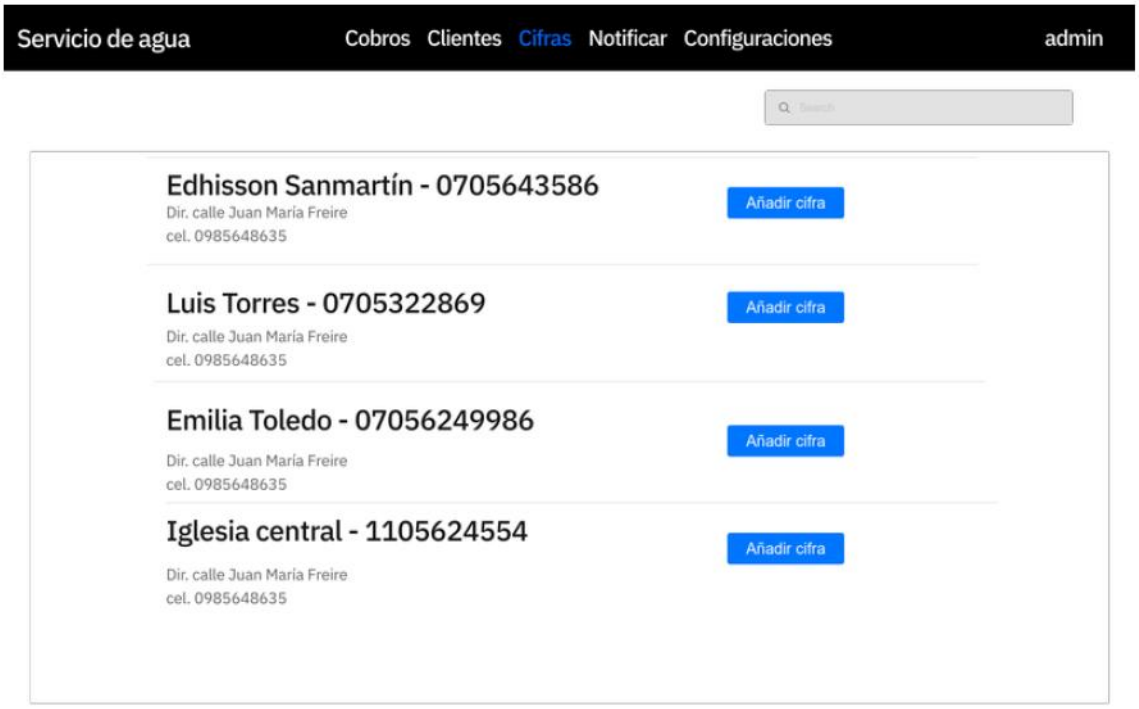


Figura 69. *Pantalla de cifras del servicio.*

Pantalla de ingreso de nuevas cifras

Observaciones:

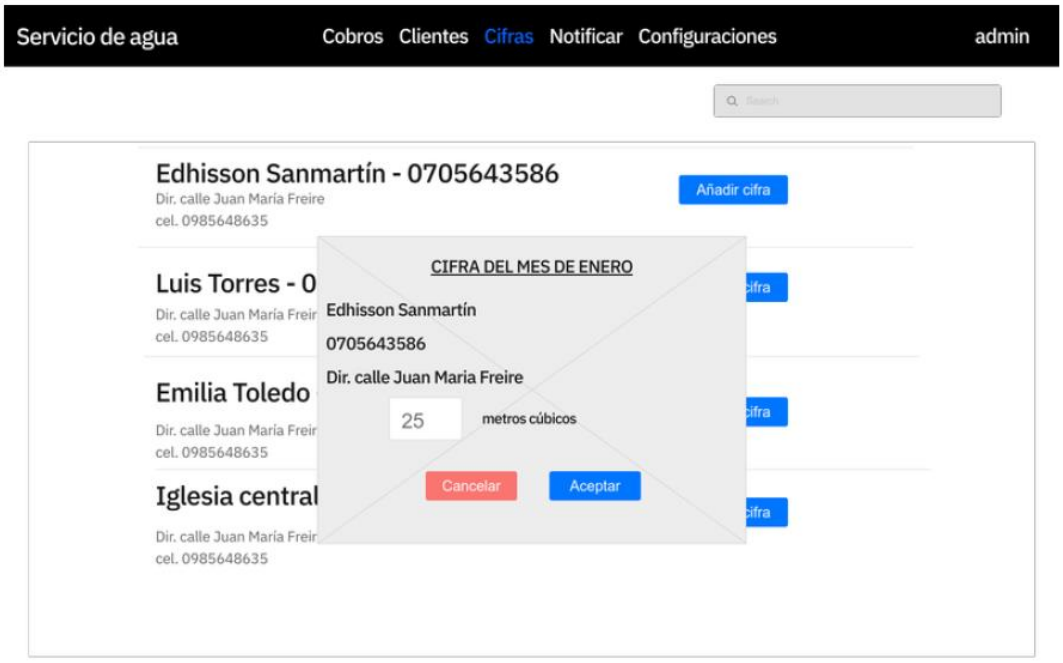


***Figura 70**. Pantalla de ingreso de cifras de los clientes.*

Pantalla para notificar deudas

Observaciones:

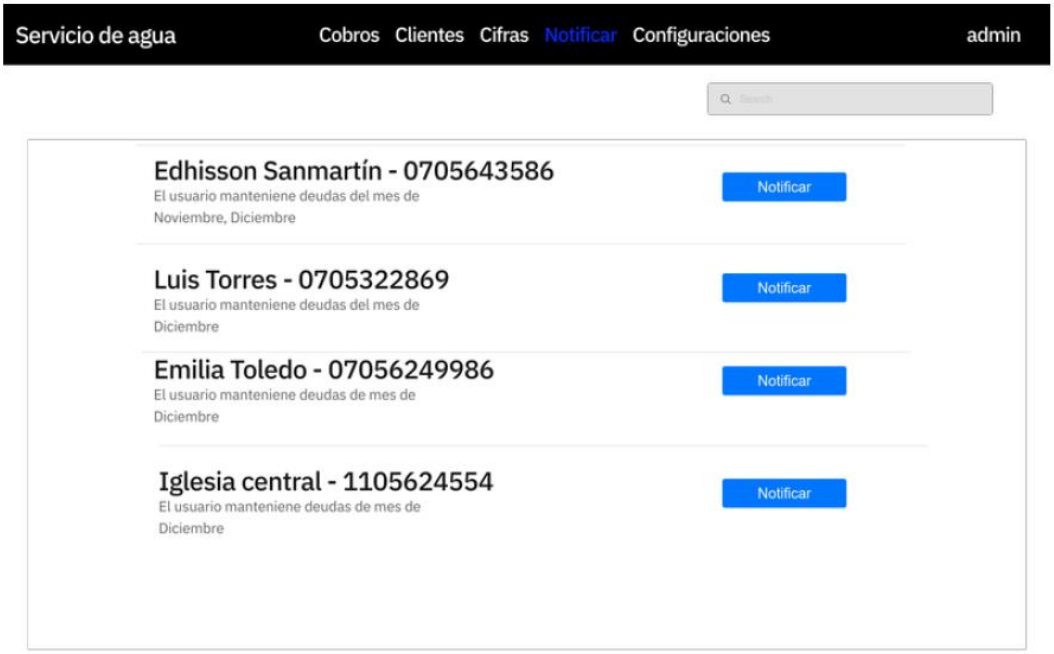


***Figura 71.** Pantalla para notificar deudas pendientes a los usuarios.*

Pantalla para las configuraciones del sistema

Observaciones:

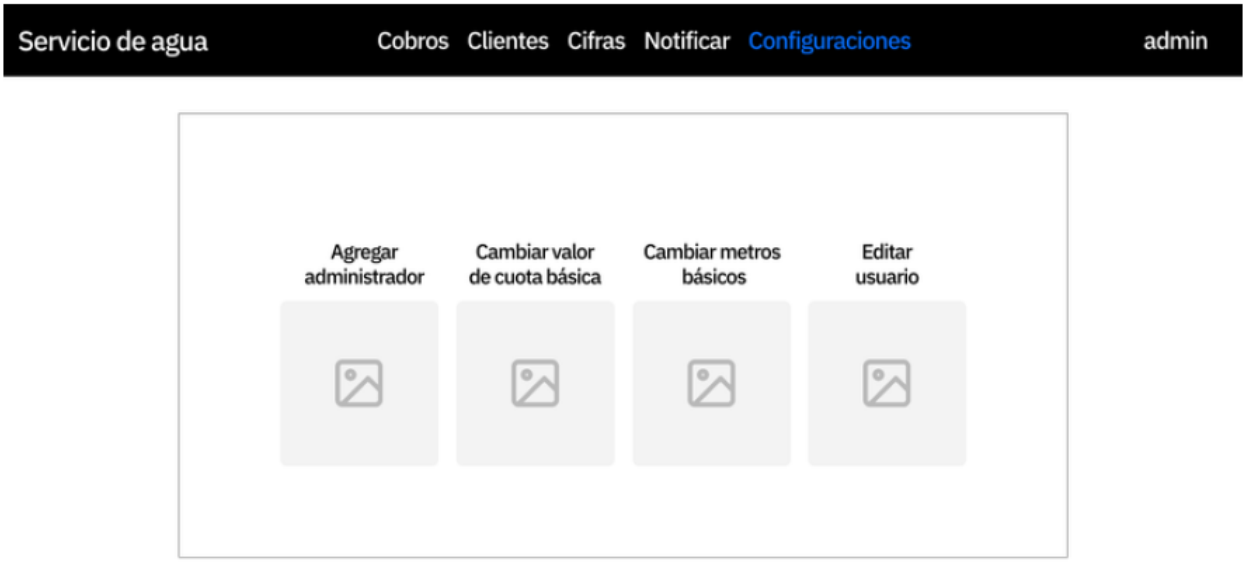


Figura 72. *Pantalla con las configuraciones que tiene la aplicación.*

Anexo 5. Prototipo final de interfaz de la aplicación web para el cobro del servicio de agua.

Prototipado final de interfaz de la aplicación web para el cobro por el consumo de agua de la Parroquia Milagro

Proyecto: Aplicación Web para la automatización del cobro de consumo de agua potable en el Gobierno Autónomo Descentralizado Rural de la Parroquia Milagro del Cantón Atahualpa.

PROTOTIPADO FINAL DE LAS INTERFACES DE LA APLICACIÓN WEB PARA EL COBRO POR EL CONSUMO DE AGUA DE LA PARROQUIA MILAGRO

El presente documento contiene las interfaces finales de cada una de las pantallas de la aplicación web para el cobro por el consumo de agua, concluyendo así la versión final de las pantallas de la aplicación.

Pantalla de inicio de sesión

Observaciones:

Figura 73. *Pantalla de inicio de sesión.*

Pantalla de inicio de la aplicación

Observaciones:

Figura 74. *Pantalla de inicio del sistema.*

Pantalla de cobros del servicio

Observaciones:

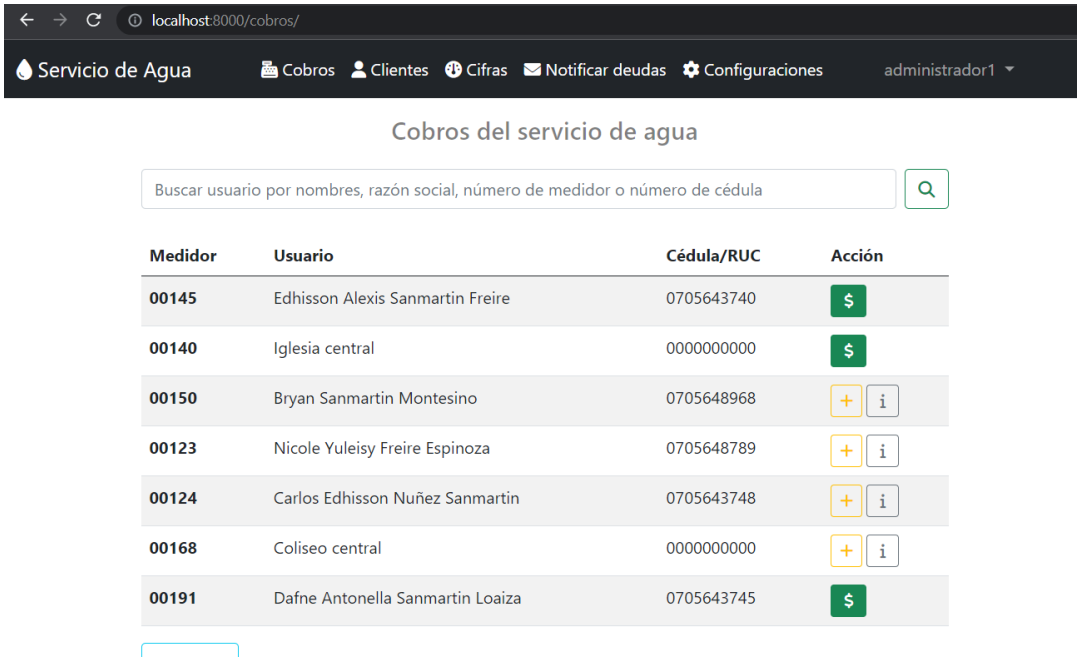


Figura 75. *Pantalla de cobros del servicio*

Pantalla de recibo de cobro

Observaciones:

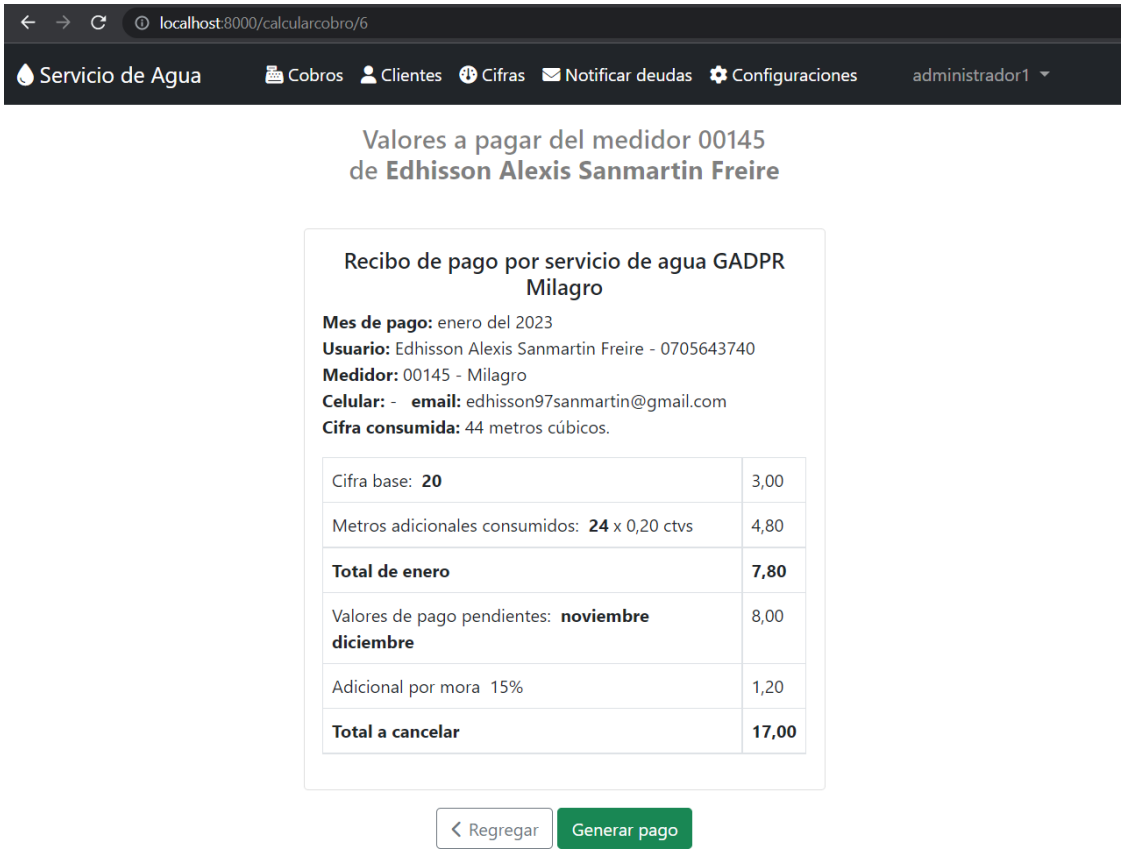


Figura 76. *Pantalla del recibo generado por el cobro del servicio.*

Pantalla de clientes

Observaciones:

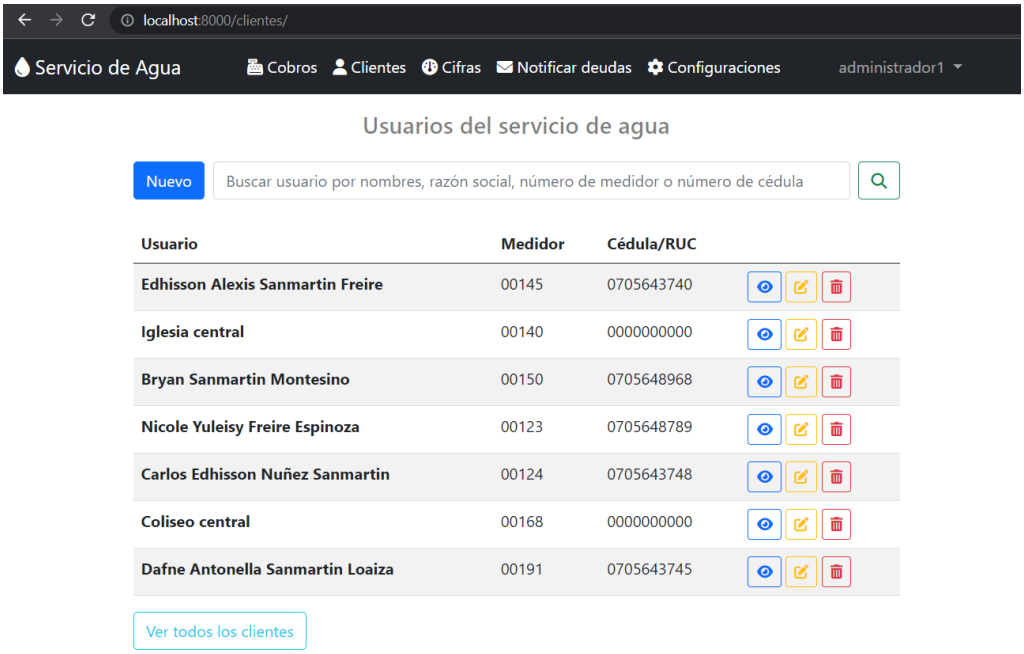


Figura 77. *Pantalla de gestión de los clientes del servicio.*

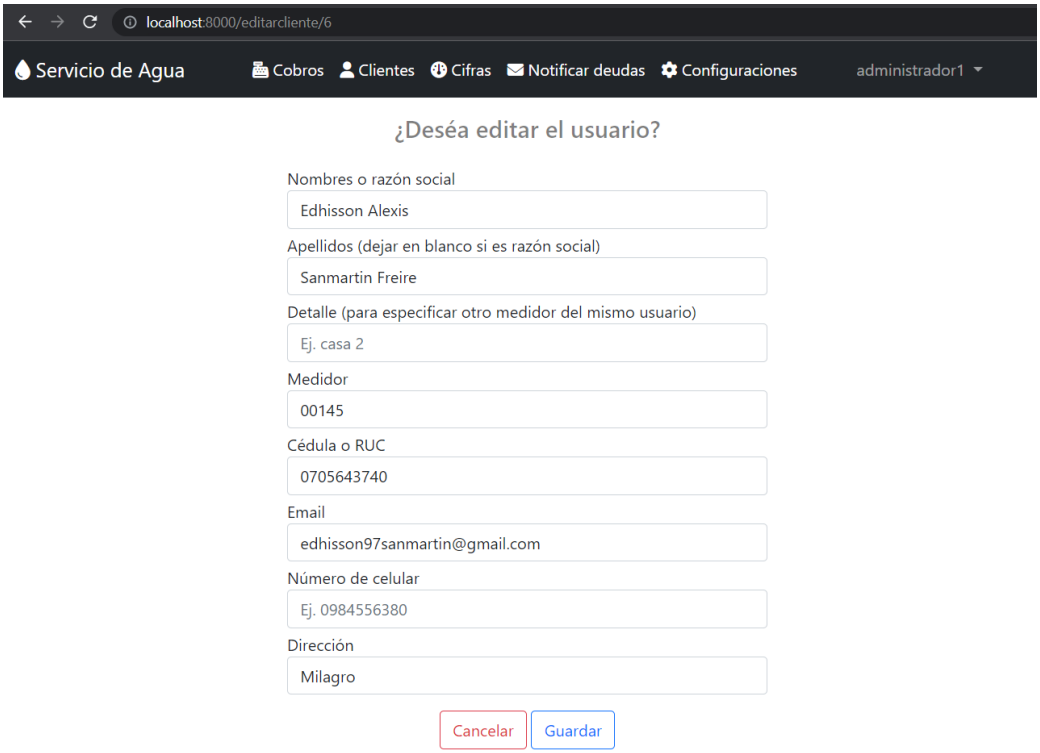


Figura 78. *Editar cliente.*

Pantalla de cifras consumidas

Observaciones:

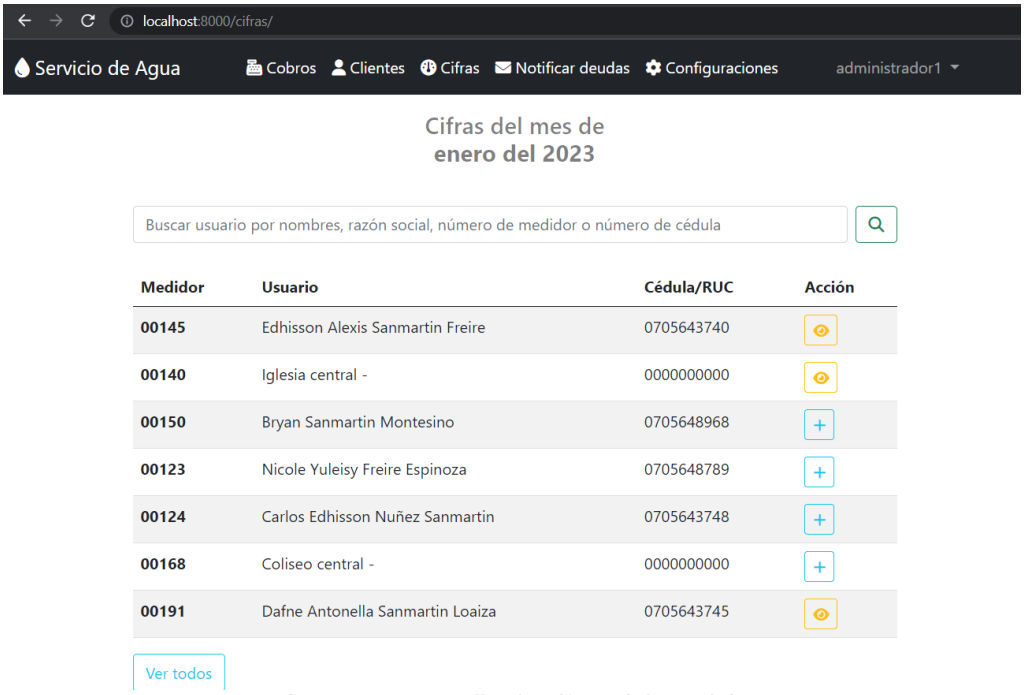


Figura 79. *Pantalla de cifras del servicio.*

Pantalla de ingreso de nuevas cifras

Observaciones:

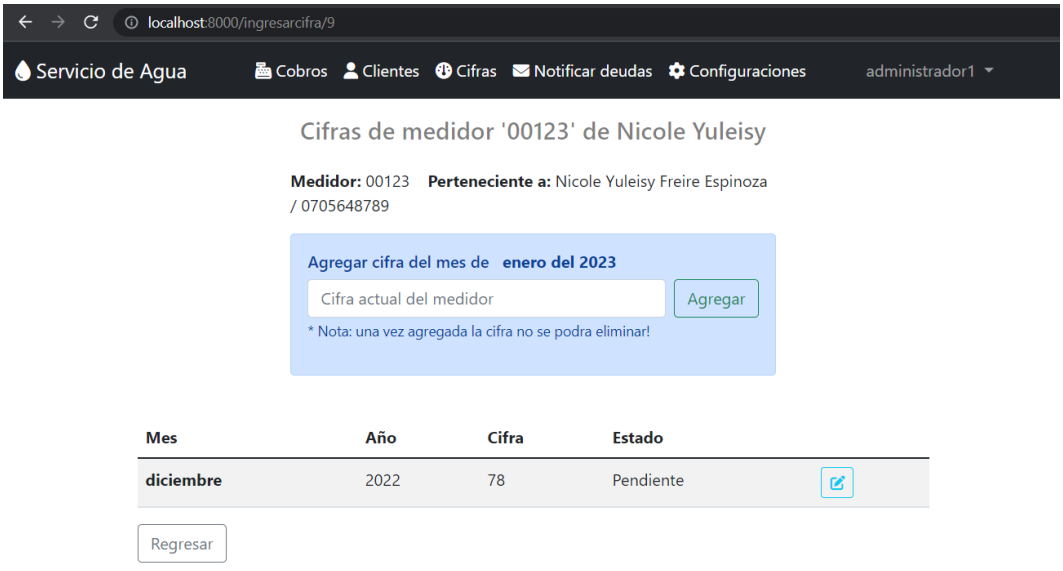


Figura 80. *Pantalla de ingreso de cifras de los clientes.*

Anexo 6. Pruebas Unitarias de la Aplicación para el Cobro del Servicio de Agua Potable de la Parroquia Milagro.

Pruebas Unitarias de la Aplicación Web para el Cobro del Servicio de Agua Potable de la Parroquia Milagro

Proyecto: Aplicación Web para la automatización del cobro de consumo de agua potable en el Gobierno Autónomo Descentralizado Rural de la Parroquia Milagro del Cantón Atahualpa.

PRUEBAS UNITARIAS DE LA APLICACIÓN WEB PARA EL COBRO POR EL CONSUMO DE AGUA DE LA PARROQUIA MILAGRO

El presente documento contiene las pruebas unitarias de cada uno de los modelos de la aplicación, la realización de estos test se lo hizo gracias a la herramienta TestCase que nos ofrece Django, a continuación, se observa una breve descripción junto con el código y su ejecución.

Test de clientes

- Creación de clientes

  Para la realización de los test se crea un cliente a partir del modelo Clientes y se ingresa a la base de datos. El test creación de clientes realiza una consulta a la base de datos del cliente ingresado, posteriormente compara los datos ingresados con los datos obtenidos, si esta comparación es correcta se pasa el test, caso contrario falla.

```
4   class ClienteTestCase(TestCase):
5       def setUp(self):
6           self.cliente = Cliente.objects.create(
7               nombres='Alexis',
8               apellidos='Sanmartin',
9               medidor='00256',
10              cedula='0705643740',
11              email='test@gmail.com',
12              direccion='milagro',
13              celular='0986532569'
14          )
15
16      def test_setUp_creation(self):
17          cliente = Cliente.objects.get(id=1)
18          self.assertEqual(cliente.medidor, '00256')
19
```

Figura 81. *Creación de clientes y test de creación.*

- Editar clientes

  El test editar clientes obtiene un cliente de la base de datos y altera uno de los campos ingresados, posteriormente guarda estos datos en la base de datos. Para verificar que se guardaron los datos correctamente se consulta a la base de datos el nuevo dato ingresado con el obtenido de la BD, esto significa que el test es correcto.

```
19
20      def test_setUp_edit(self):
21          cliente = Cliente.objects.get(id=1)
22          cliente.medidor='36523'
23          cliente.save()
24          c=Cliente.objects.get(id=1)
25          self.assertEqual(cliente.medidor, '36523')
26
```

Figura 82. *Test Unitario para editar clientes.*

- Eliminar clientes

  La prueba eliminar clientes, obtiene un cliente de la base de datos y luego lo elimina, posteriormente se procede a la comparación de la base de datos con uno de los datos del cliente, si la comparación no es igual, esto se traduce a que el test pasó con éxito.

```
26
27      def test_setUp_delete(self):
28          cliente = Cliente.objects.get(id=1)
29          cliente.delete()
30          self.assertIsNot(cliente.apellidos,'Sanmartin')
31
```

Figura 83. *Test Unitario para eliminar clientes.*

- Ejecución de test

  Al realizar la ejecución de los test, la herramienta crea una BD ficticia y luego procede a destruirla.

```
(venv) C:\Users\USER\Documents\ProjectsDjango\AppAguaMilagro>py manage.py test clientes
Found 3 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
...
----------------------------------------------------------------------
Ran 3 tests in 0.008s

OK
Destroying test database for alias 'default'...
```

Figura 84. *Ejecución de los test para cliente.*

Test de cifras

Para el ingreso de cifras, primero se ingresó un cliente ficticio a la Base de datos.

```
class CifrasTestCase(TestCase):
    def setUp(self):
        self.cliente = Cliente.objects.create(
            nombres='Alexis',
            apellidos='Sanmartin',
            medidor='00256',
            cedula='0705643740',
            email='test@gmail.com',
            direccion='milagro',
            celular='0986532569'
        )
```

Figura 85. *Ingreso de cliente a la base de datos ficticia.*

- Ingreso de cifras

  Para el ingreso de cifras primero se obtiene el cliente de la base de datos y luego procede a crear la cifra con el id del cliente, para que el test este correcto se compara los datos ingresados con los de la BD.

```
17
18      def test_setUp_ingresarcifras(self):
19          cliente = Cliente.objects.get(id=1)
20          self.cifras = Cifras.objects.create(
21              id_usuario=cliente,
22              mes='enero',
23              anio='2013',
24              cifra='25',
25              id_valores=1
26              )
27          cifra = Cifras.objects.get(id=1)
28          self.assertEqual(cifra.cifra, 25)
```

Figura 86. *Test Unitario para ingresar cifras.*

- Editar cifras

  Para editar cifras, se obtiene la cifra a editar y posteriormente se edita uno de los campos, para pasar el test el campo editado debe ser igual al que está en la base de datos.

```
30      def test_setUp_editValores(self):
31          cliente = Cliente.objects.get(id=1)
32          self.cifras = Cifras.objects.create(
33              id_usuario=cliente,
34              mes='enero',
35              anio='2013',
36              cifra='25',
37              id_valores=1
38          )
39          cifr = Cifras.objects.get(id=1)
40          cifr.anio='2014'
41          cifr.save()
42          c=Cifras.objects.get(id=1)
43          self.assertEqual(c.anio, '2014')
```

Figura 87. *Test Unitario para editar cifras.*

- Ejecución de los test

  Se ejecuta los test a través de la consola y se obtiene los siguientes resultados.

```
(venv) C:\Users\USER\Documents\ProjectsDjango\AppAguaMilagro>py manage.py test cifras
Found 2 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
..
----------------------------------------------------------------------
Ran 2 tests in 0.012s

OK
Destroying test database for alias 'default'...
```

Figura 88. *Ejecución de los test para cifras.*

Test de cobros

- Ingresar nuevo cobro

  Para el ingreso de cobros, se crea un cliente ficticio, se consume este cliente desde la base de datos y posteriormente se le agrega un nuevo cobro, se compara uno de los valores agregados con los obtenidos de la BD, si esto resulta, el test es correcto.

```python
class CobrosTestCase(TestCase):
    def setUp(self):
        self.cliente = Cliente.objects.create(
            nombres='Alexis',
            apellidos='Sanmartin',
            medidor='00256',
            cedula='0705643740',
            email='test@gmail.com',
            direccion='milagro',
            celular='0986532569'
        )

    def test_setUp_creationcobros(self):
        date = datetime.date.today()
        cliente = Cliente.objects.get(id=1)
        self.cobros = Cobros.objects.create(
            id_usuario=cliente,
            registro=date,
            fechacifra='enero 2013',
            total=5.31
            )
        cobros = Cobros.objects.get(id=1)
        self.assertEqual(cobros.fechacifra, 'enero 2013')
```

Figura 89. *Test Unitario para ingresar nuevos cobros.*

- Ejecución del test

  Al realizar la ejecución del test se puede observar que es correcto y pasa la prueba.

```
(venv) C:\Users\USER\Documents\ProjectsDjango\AppAguaMilagro>py manage.py test cobros
Found 1 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
.
----------------------------------------------------------------------
Ran 1 test in 0.006s

OK
Destroying test database for alias 'default'...
```

Figura 90. *Ejecución del test para cobros.*

Test para configuraciones

- Test para crear usuarios administradores

  Para la creación de un usuario administrador primero se crea en la BD el usuario y luego se compara si el usuario esta activo y tiene los privilegios, esto para saber en qué nivel esta y si es correcta su creación.

```python
class UserTestCase(TestCase):
    def setUp(self):
        self.user = User.objects.create_user(
            username='Secretaria349',
            password='12345',
            is_staff=False
        )
    def test_setUp_creation(self):
        self.assertEqual(self.user.is_active,True)
        self.assertEqual(self.user.is_staff,False)
        self.assertEqual(self.user.is_superuser,False)

```

Figura 91. *Test Unitario para la creación de un usuario administrador del sistema.*

- Test para crear super usuario administrador

  Para realizar este test se hizo uso de la función para crear super usuarios misma que activa todos los privilegios que el administrador posee en el sistema, una vez realizado esto, se comprueba si el usuario creado tiene los privilegios de SU, si es así, el test es correcto.

```python
class SuperTestCase(TestCase):
    def setUp(self):
        self.user =User.objects.create_superuser(
            username='Secretaria349',
            password='12345',
            is_staff=True
        )
    def test_setUp_creationSU(self):
        self.assertEqual(self.user.is_active,True)
        self.assertEqual(self.user.is_staff,True)
        self.assertEqual(self.user.is_superuser,True)

```

Figura 92. *Test Unitario para la creación de Super Usuarios.*

- Test para ingresar valores base del sistema

  La realización de este test consta de crear los valores base y luego compara esos valores con uno de los ingresados, si esto es correcto, el test está bien.

```python
class ValoresTestCase(TestCase):
    def setUp(self):
        self.valores = ValoresBase.objects.create(
            valor_cifra_base=3.00,
            valor_adicional=0.25,
            base=25,
            porcentaje_mora=15
        )

    def test_setUp_creation(self):
        valores = ValoresBase.objects.get(id=1)
        self.assertEqual(valores.base, 25)
```

Figura 93. *Test Unitario para crear valores base del sistema.*

- Ejecución de los test

  Finalmente se ejecuta los test de las configuraciones de la aplicación y se puede observar que se ejecutan sin ningún problema.

```
(venv) C:\Users\USER\Documents\ProjectsDjango\AppAguaMilagro>py manage.py test configuraciones

Found 3 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
...
----------------------------------------------------------------------
Ran 3 tests in 0.504s

OK
Destroying test database for alias 'default'...
```

Figura 94*. Ejecución de las pruebas de configuraciones.*

Anexo 7. Entrevista – Validación de la Aplicación Web para el cobro por el consumo del servicio de agua potable en el Gobierno Autónomo Descentralizado Parroquial Rural de Milagro.

Entrevista – Validación de la Aplicación Web para el cobro por el consumo del servicio de agua potable en el Gobierno Autónomo Descentralizado Parroquial Rural de Milagro

Departamento: Oficinas del Gobierno Autónomo Descentralizado Parroquial Rural de Milagro, Cantón Atahualpa

Entrevistado: Sr. Franco Antonio Sanmartín Gómez

Objetivo: Validar la aplicación web para el consumo de agua en la parroquia Milagro.

Fecha de revisión: 03-02-2023

Autor: Edhisson Sanmartín Freire.

Descripción:

La siguiente entrevista tiene como finalidad y objetivo validar con el cliente cada uno de los requisitos funcionales y no funcionales de la Aplicación Web para el cobro por el consumo de agua en la Parroquia Milagro definidos al inicio del desarrollo del presente trabajo de titulación.

Tabla 37. *Requisitos funcionales validados por el cliente.*

Validación de los requerimientos funcionales					
#	**Nombre**	**Breve descripción**	**Cumple**	**No Cumple**	**Observaciones**
RF001	Inicio de sesión	El Software contará con un inicio de sesión para validar credenciales del administrador.	**X**		
RF002	Ingreso de usuarios	Se permitirá a través del administrador	**X**		

		ingresar los usuarios del servicio.			
RF003	Cobros del servicio	El administrador podrá realizar los cobros del servicio de cada uno de los usuarios antes ingresados y su respectivo consumo.	**X**		
RF004	Ingreso de cifras consumidas	Cada mes el administrador tendrá la opción de subir las cifras de consumo.	**X**		
RF005	Comprobante de pago	Se generará un comprobante de pago que será impreso o enviado al correo del usuario.	**X**		
RF006	Configuraciones del sistema	Se podrá cambiar la base mínima de consumo, así como la tarifa extra y demás.	**X**		
RF007	Notificar deuda	El administrador podrá enviar correo de aviso a los usuarios que tengan deudas pendientes.	**X**		

Además, la validación de los requisitos no funcionales se muestra a continuación:

Tabla 38. Requisitos no funcionales validados por el cliente.

Requerimientos no funcionales					
#	**Nombre**	**Breve descripción**	**Cumple**	**No Cumple**	**Observaciones**
RN001	Compatibilidad	El sistema permitirá poder ser usado en los distintos entornos de navegación web.	**X**		
RN002	Desempeño	Se garantizará a los usuarios la realización de las tareas de una manera eficaz.	**X**		
RN003	Disponibilidad	El sistema estará disponible siempre que se requiera de sus servicios.	**X**		
RN004	Fiabilidad	Se garantizará a los usuarios seguridad en cuanto a la información que se provee al sistema.	**X**		
RN005	Tiempo de respuesta	Se responderá de una manera rápida y correcta en el menor tiempo posible a las peticiones realizadas.	**X**		

RN006	Usabilidad	Será de fácil manejo, contará con una interfaz de usuario sencilla y amigable.	**X**		

Firmado electrónicamente por:
FRANCO ANTONIO SANMARTIN GOMEZ

Franco Antonio Sanmartín Gómez

Presidente del Gobierno Autónomo Descentralizado

Parroquial Rural de Milagro.

Anexo 8. Manual de usuario de la aplicación web para el cobro por el consumo de agua de la Parroquia Milagro.

Manual de usuario de la aplicación web para el cobro por el consumo de agua de la Parroquia Milagro

Proyecto: Aplicación Web para la automatización del cobro de consumo de agua potable en el Gobierno Autónomo Descentralizado Rural de la Parroquia Milagro del Cantón Atahualpa.

MANUAL DE USUARIO DE LA APLICACIÓN WEB PARA EL COBRO POR EL CONSUMO DE AGUA DE LA PARROQUIA MILAGRO

El presente documento contiene cada una de las pantallas de la aplicación web con la especificación de lo que realiza cada funcionalidad, describe las funciones del sistema como una ayuda para el usuario final.

Inicio de sesión a la aplicación

El administrador tiene la facultad de iniciar sesión a la aplicación web, así mismo, ofrece credenciales de inicio de sesión a los demás usuarios del sistema, los mismos que pueden acceder al servicio.

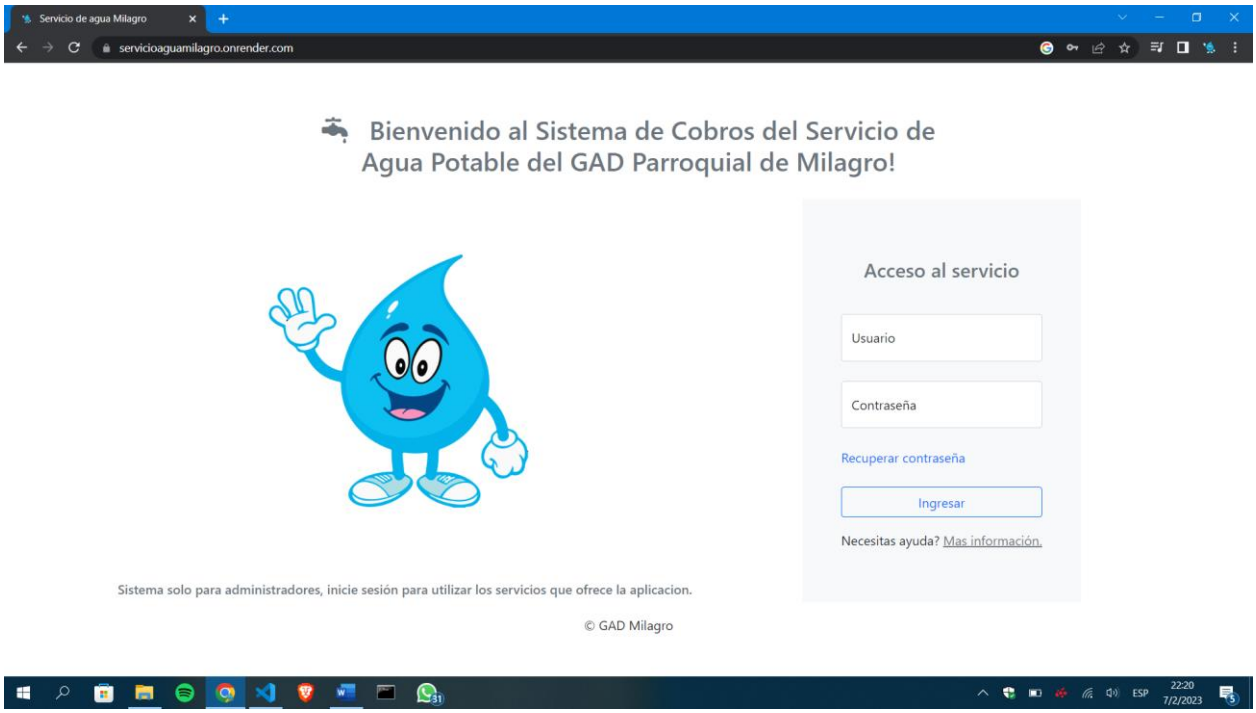


Figura 95. *Pantalla de inicio de sesión.*

- **Recuperación de contraseña**

Para recuperar la contraseña de acceso al servicio ingresar al item *Recuperar contraseña:*

***Figura 96**. Recuperación de acceso al sistema*

Ingresar la dirección de correo electrónico con el que se creó la cuenta:

***Figura 97.** Ingreso de correo electrónico para recuperación de contraseña de acceso.*

Al presionar enviar, se enviará un correo con las instrucciones de recuperar contraseña, en la bandeja del correo electrónico, donde se debe acceder a la url enviada:

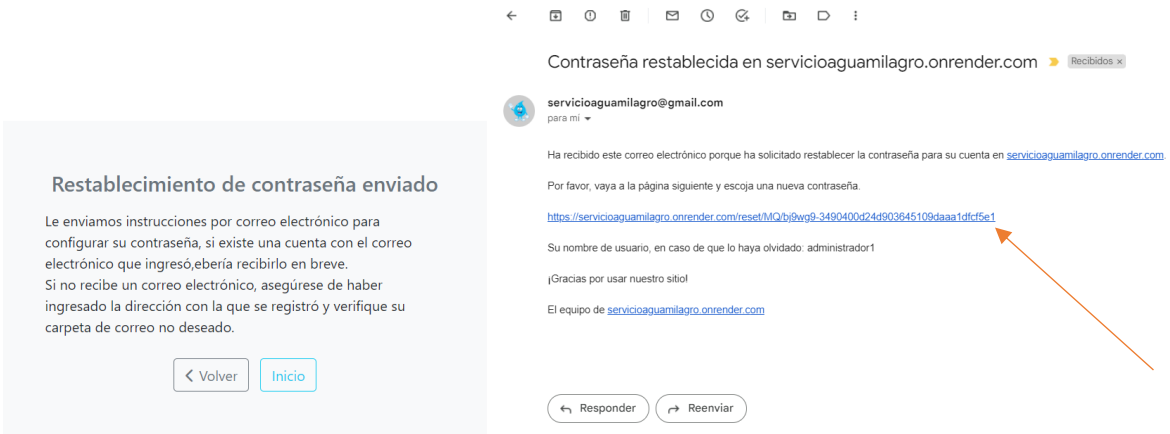


***Figura 98.** Correo de recuperación de acceso a la aplicación.*

Una vez que se accede al link, se abrirá una nueva pestaña donde se debe ingresar la nueva contraseña y confirmarla, con esto estaría todo el proceso culminado.

Figura 99. *Ingreso de la nueva contraseña de acceso.*

Panel principal de la aplicación

La aplicación web cuenta con las funcionalidades que se muestran a continuación:

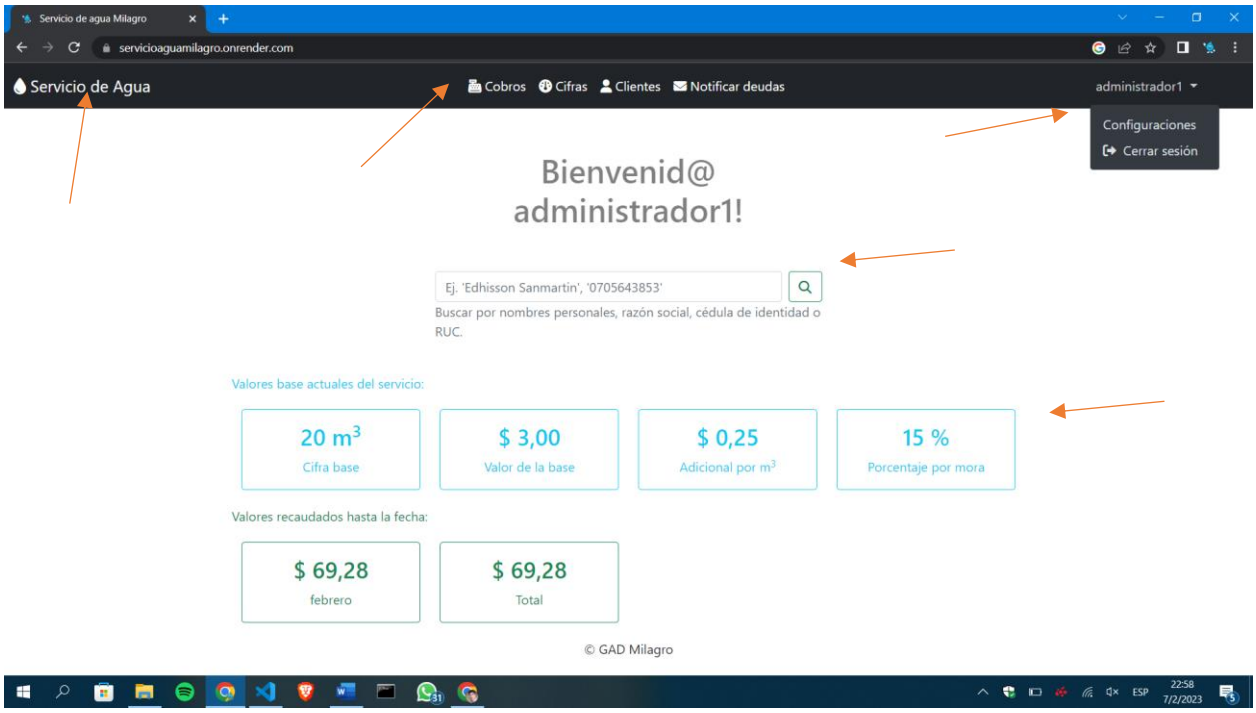


Figura 100. *Panel principal de la aplicación.*

a. Ícono de la página principal
b. Panel administrativo de cobros
c. Buscador rápido de clientes
d. Configuraciones del sistema
e. Información base de la aplicación

Clientes

La sección de clientes realiza un listado con todos los clientes del sistema, da la opción de crear, ver, editar y eliminar un determinado usuario del servicio:

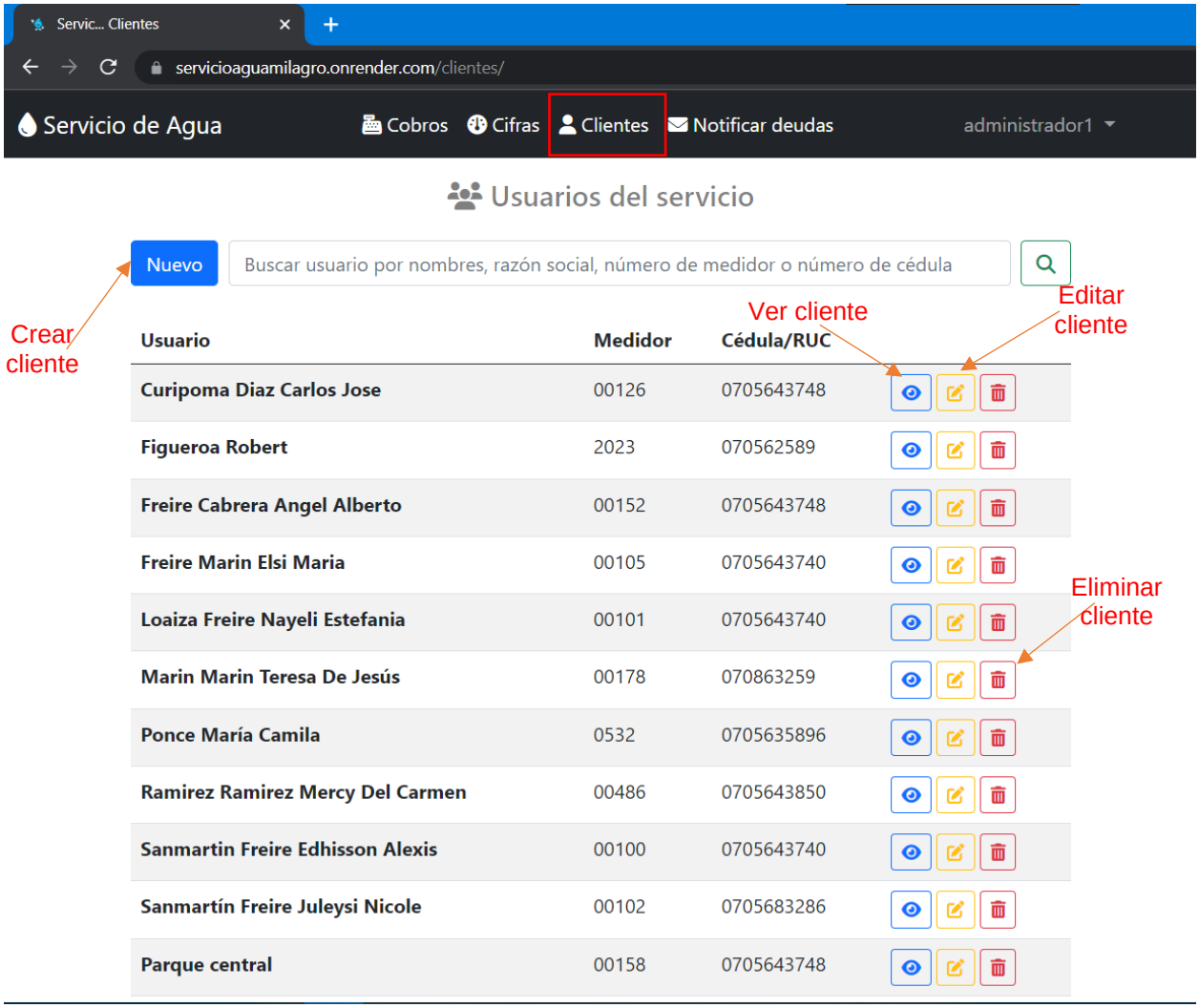


Figura 101. *Vista de clientes.*

Cifras

La vista de cifras muestra todos los clientes ingresados al sistema, el cliente muestra la acción a realizar según su estado, entre las cuales se tiene: ingresar cifra y ver cifras ingresadas.

Figura 102. *Cifras del servicio.*

- **Ver cifras ingresadas y editar**

Al ingresar a la sección para ver las cifras ingresadas actualmente de cada usuario, se puede realizar la acción de editar cifra siempre que la misma no haya sido cancelada por el cliente:

Figura 103. *Editar cifras ingresadas.*

❖ **Ingresar nuevas cifras**

El sistema permite ingresar una cifra mensualmente los primeros días del mes al ingresar a la acción ingresar cifras:

***Figura 104**. Ingreso de cifras al sistema.*

Realizar cobros

La sección de realizar cobros cuenta con las acciones de: ver los pagos realizados, realizar cobros a los usuarios e ingresar cifras de usuarios que aún no hayan sido ingresadas en el actual mes:

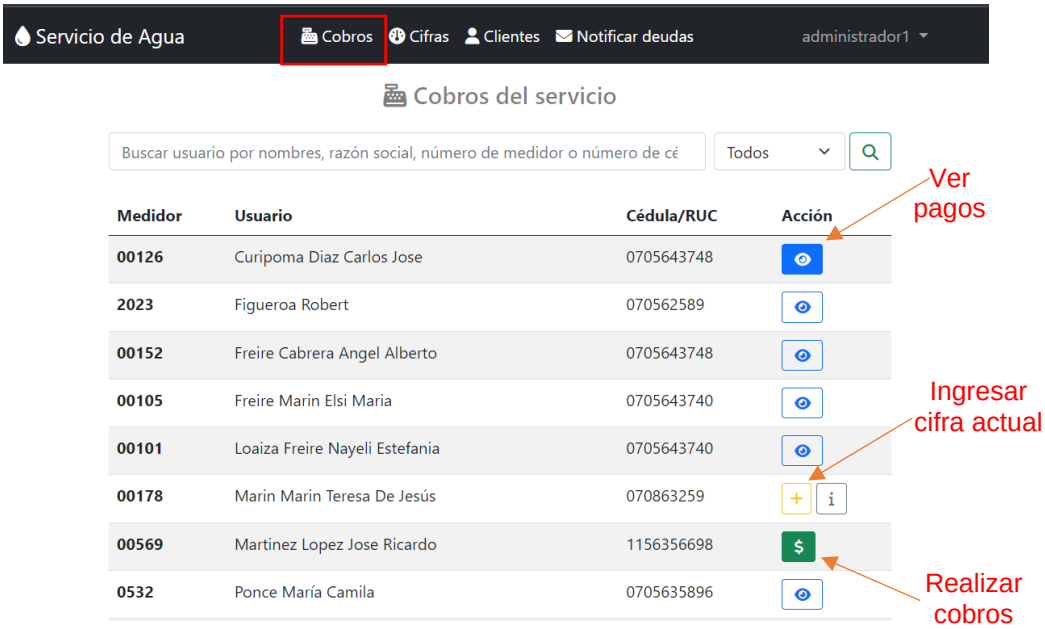


***Figura 105**. Listado de clientes de la sección de cobros.*

❖ **Ver pagos**

La acción de ver pagos muestra cada uno de los pagos realizados hasta la fecha por el usuario:

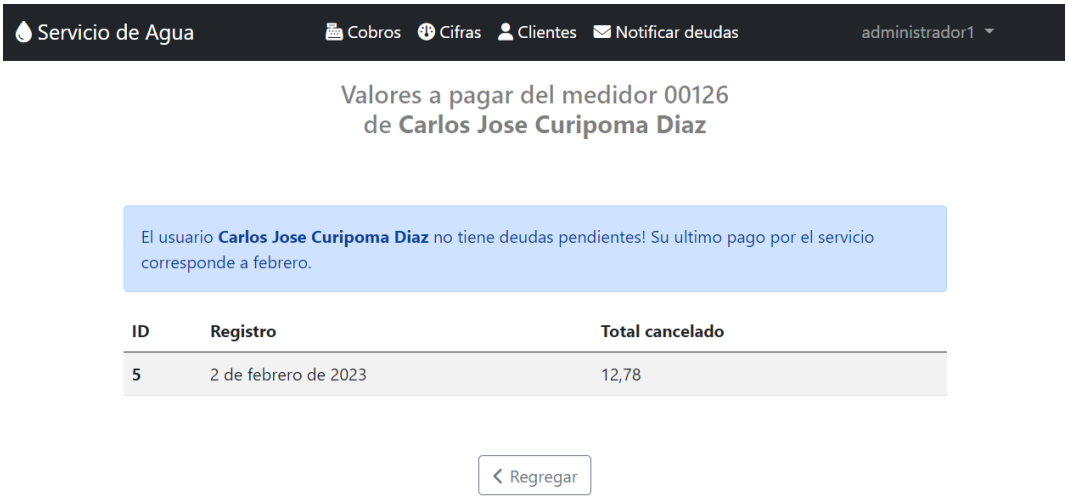


Figura 106. *Ver pagos del usuario*

❖ **Realizar cobro**

La acción de realizar cobro genera los valores a pagar del usuario, al realizar la acción generar pago, se genera el respectivo pago, antes no:

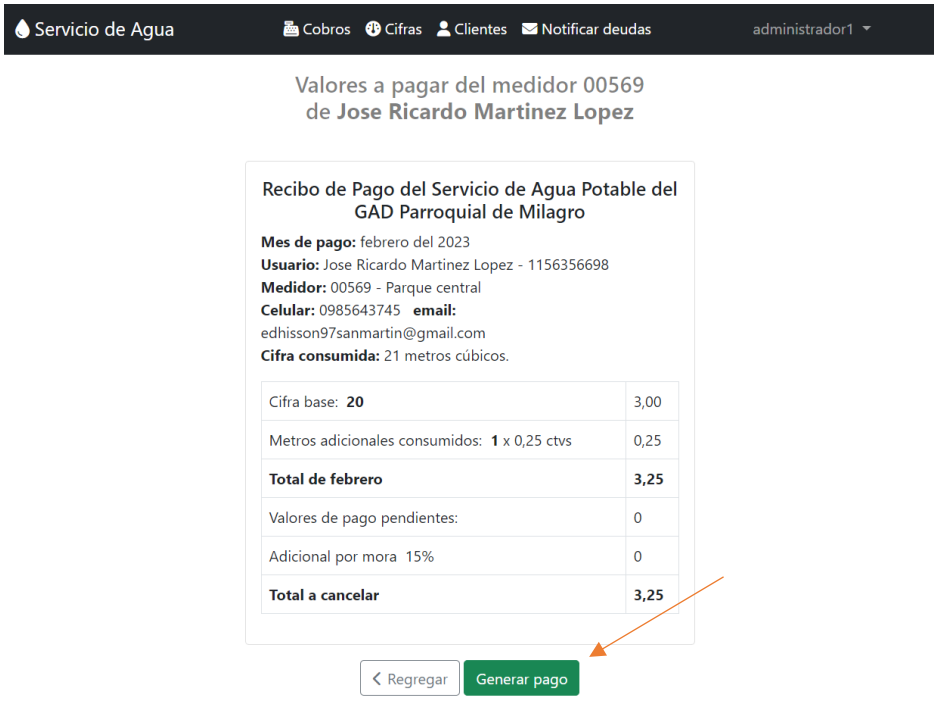


Figura 107. *Generar pago del usuario.*

❖ Notificar deudas

La sección notificar deudas a los usuarios cuenta con las acciones de notificar la deuda a un usuario en específico o a todos los usuarios que tengan deudas pendientes. Si un usuario fue notificado individualmente y se luego se procede a notificar a todos los usuarios, el usuario antes notificado no se volverá a notificar, esto con la finalidad de no invadir el correo de los usuarios.

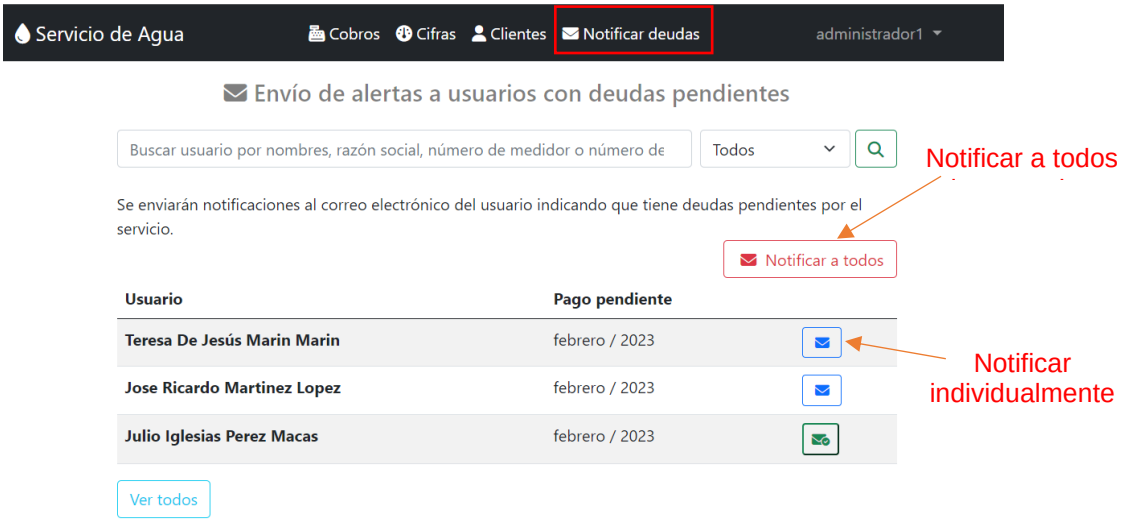


Figura 108. *Notificar deudas de los usuarios.*

❖ Configuraciones del sistema

La sección de configuraciones cuenta con las acciones de:

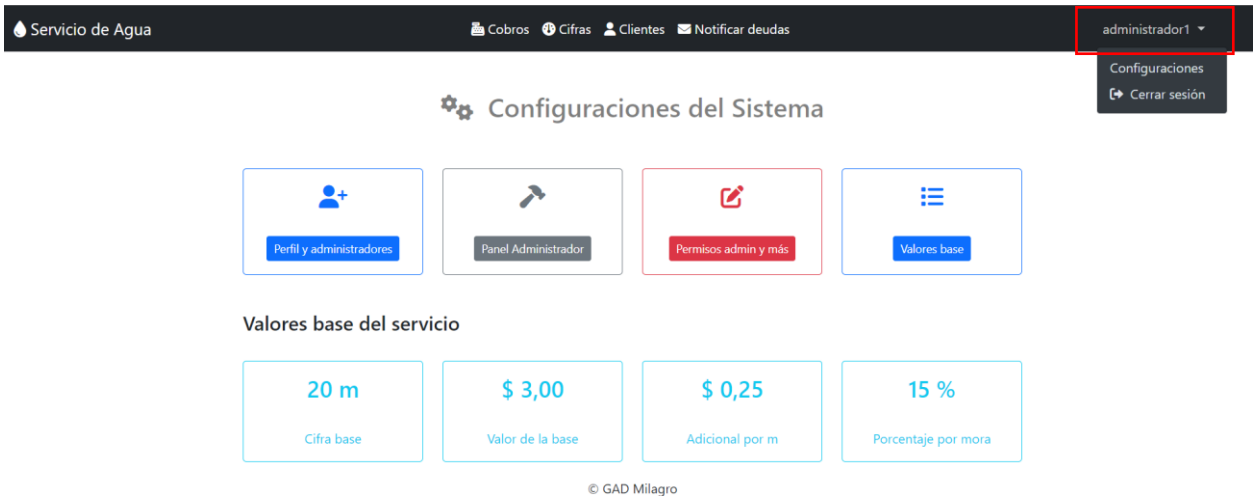


Figura 109. *Configuraciones del sistema.*

❖ **Perfil y administradores**

La sección de perfil y administradores cuenta con las acciones de ingresar nuevos usuarios de acceso al sistema, de editar el usuario actual y eliminarlo.

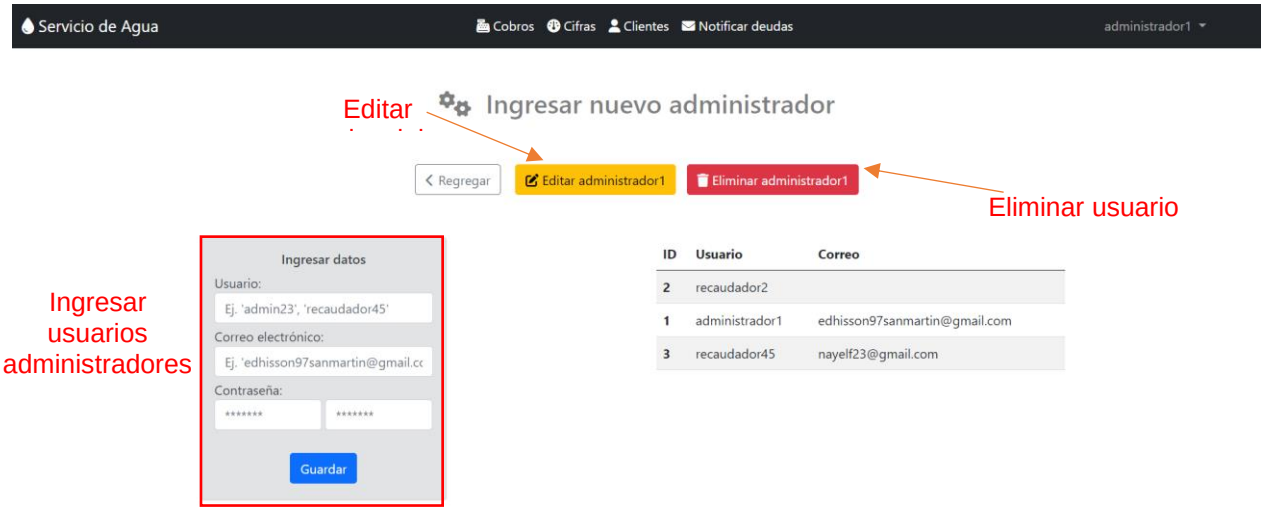


Figura 110. *Perfil y administradores.*

❖ **Panel administrador**

El panel administrador cuenta con acceso a las tablas de la base de datos de la aplicación, para entrar a esta sección es necesario tener los privilegios de un super usuario. Aquí se puede modificar cualquier tipo de datos por lo que solo el administrador general del sistema tiene acceso.

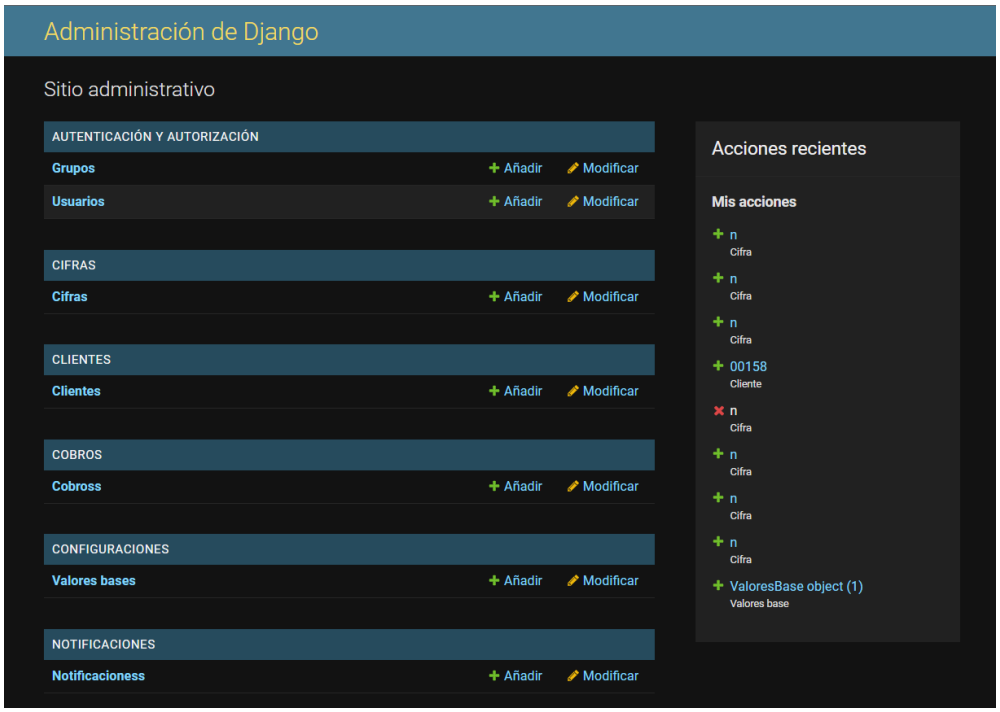


Figura 111. *Panel administrador Django.*

❖ Permisos admin y más

En esta sección se puede agregar permisos de super usuario a cualquier tipo de usuario o a su vez denegar los permisos de acceso y eliminar un usuario predeterminado. A esta sección solo tiene permitido el acceso un super usuario del sistema:

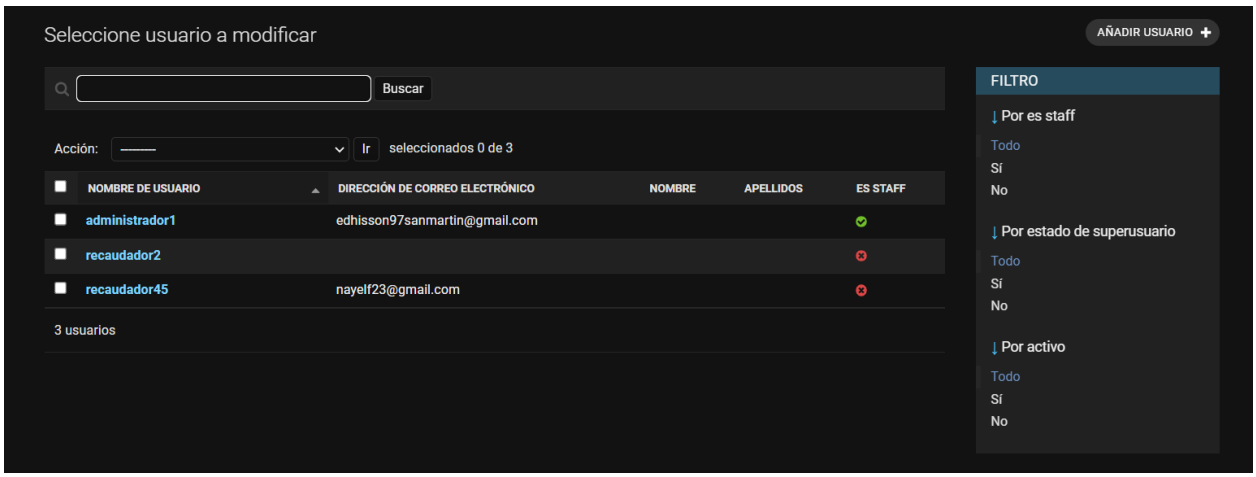


Figura 112. *Permisos de usuarios administradores.*

❖ **Valores base del sistema**

Esta sección cuenta con la facultad de ingresar nuevos valores base al servicio, cabe destacar que los valores base agregados solo afectarán a las cifras agregadas luego de haber ingresado los nuevos valores base, esto debido a que usuarios con cifras anteriores ingresadas deberían cancelar por la cifra con los valores base ingresados en ese momento.

Figura 113. *Ingreso de nuevos valores base.*

Anexo 9. Entrevista – Usabilidad de la Aplicación Web para el cobro por el consumo del servicio de agua potable en el Gobierno Autónomo Descentralizado Parroquial Rural de Milagro.

Entrevista – Usabilidad de la Aplicación Web para el cobro por el consumo del servicio de agua potable en el Gobierno Autónomo Descentralizado Parroquial Rural de Milagro

Departamento: Oficinas del Gobierno Autónomo Descentralizado Parroquial Rural de Milagro, Cantón Atahualpa

Entrevistado: Sra. Laura Estefanía Loaiza Romero

Objetivo: Validar la usabilidad de la aplicación web para el consumo de agua en la parroquia Milagro.

Fecha de revisión: 10-02-2023

Autor: Edhisson Sanmartín Freire.

Descripción:

La siguiente entrevista tiene como finalidad y objetivo conocer la usabilidad de la Aplicación Web para el cobro por el consumo de agua potable en la Parroquia Milagro, si cumple con las expectativas que tiene el cliente final del proyecto.

- **¿Considera que la aplicación web es fácil de entender y utilizar? ¿Por qué?**

  Si, porque la aplicación cuenta con una vista de todas las funciones lo cual permite que sea fácil de entender y de utilizar.

- **¿Considera que la aplicación web es amigable? ¿Por qué?**

  Si, porque visualmente es agradable y entendible.

- **¿La aplicación web optimiza los tiempos para el cálculo de valores a pagar del servicio?**

Si, la aplicación es de gran ayuda, realiza cálculos en cuestión de segundos que antes se tardaban varios minutos.

- **¿Cuál es el tiempo estimado que se tardaba en calcular los valores a mano de un usuario que no tenga deudas pendientes, solo el valor del mes actual?**

  Entre buscar el listado de usuarios, comprobar que no cuente con valores pendientes y realizar los cálculos de los valores, un estimado sería de 4 minutos.
- **¿Cuál es el tiempo estimado que se tardaba en calcular los valores a mano de un usuario que deude la cifra actual y esta supere el valor base?**

  El tiempo estimado es de hasta 5 minutos debido a que se debe calcular el excedente de la base.
- **¿Cuál es el tiempo estimado que se tardaba en calcular los valores a mano de un usuario con deudas pendientes?**

  Aquí se complica un poco porque hay que buscar los registros de los meses pendientes, realizar los cálculos y sumar los valores, un valor estimado sería más de 5 minutos.
- **¿Cuál es el tiempo estimado que se tarda en calcular los valores con la aplicación web de un usuario que no tenga deudas pendientes, solo el valor del mes actual?**

  Se tarda menos de un minuto.
- **¿Cuál es el tiempo estimado que se tarda en calcular los valores con la aplicación web de un usuario que deude la cifra actual y esta supere el valor base?**

  Se tarda menos de un minuto.
- **¿Cuál es el tiempo estimado que se tarda en calcular los valores en la aplicación web de un usuario con deudas pendientes?**

  Se tarda menos de un minuto.
- **¿La aplicación web tiene un tiempo de respuesta óptimo?**

  Si, la aplicación no tarda en responder, cuando se presiona una función la respuesta es casi al instante.
- **¿Considera que la aplicación web mejora el cobro del servicio de agua potable? ¿Por qué?**

  Considero que la aplicación web mejora notablemente el cobro de consumo de agua, esto debido a que calcula los valores automáticamente, no se necesitan registros a mano y buscar en ellos la cifra de los usuarios, entrega una constancia de los valores cancelados y guarda registros fáciles de indagar.

A continuación se describe en resumen los tiempos tomados al realizar los cálculos manualmente por el entrevistado:

Tabla 39. *Tiempos al realizar los cálculos manualmente.*

Acción realizada	Minutos
Usuario con última cifra pendiente sin exceder los valores base	3:20
Usuario con última cifra pendiente con valores base excedidos	3:54
Usuario con deudas pendientes y valores excedidos	5:53
Tiempo promedio	4:33

A continuación se describe en resumen los tiempos tomados al realizar los cálculos en la aplicación web por el entrevistado:

Tabla 40. *Tiempos al realizar los cálculos en la Aplicación Web.*

Acción realizada	Minutos
Usuario con última cifra pendiente sin exceder los valores base	0:31
Usuario con última cifra pendiente con valores base excedidos	0:18
Usuario con deudas pendientes y valores excedidos	0:22
Tiempo promedio	0:23

Firmado electrónicamente por:
LAURA ESTEFANIA LOAIZA ROMERO

Laura Estefanía Loaiza Romero

Secretaria del Gobierno Autónomo Descentralizado

Parroquial Rural de Milagro.

Anexo 10. Certificado de traducción de resumen

Tania Del Rocio Loaiza Dávila

LICENCIADA EN CIENCIAS DE LA EDUCACIÓN MENCIÓN INGLÉS

Registro Profesional SENESCYT N°: 1011-13-1256991

CERTIFICA

Que, en la ciudad de Machala, a los 26 días del mes de marzo de dos mil veintitrés, se ha procedido a realizar la revisión del documento RESUMEN del Trabajo de Titulación denominado **"Aplicación Web para la automatización del cobro de consumo de agua potable en el Gobierno Autónomo Descentralizado Rural de la Parroquia Milagro del Cantón Atahualpa"**, perteneciente al señor EDHISSON ALEXIS SANMARTÍN FREIRE, con número de cedula: 0705643740. Documento que consta de 438 palabras en español y 404 palabras en idioma inglés y que está traducido en su integridad, manteniendo el mismo mensaje de su originalidad en español.

Es todo lo que puedo certificar en honor a la verdad.

Machala, 26 de marzo de 2023

Lcda. Tania Del Rocio Loaiza Dávila

LICENCIADA EN CIENCIAS DE LA EDUCACIÓN MENCION INGLÉS

Cédula: 0703742601