UNIVERSIDAD DEL BÍO-BÍO
FACULTAD DE CIENCIAS EMPRESARIALES
ESCUELA DE INGENIERÍA CIVIL INFORMÁTICA

Sistema de administración y gestión para farmacias comunales

Diego Alexis Navarrete Gutiérrez
Juan Carlos Solís Montecino

Enero de 2019 Chillán-
Chile

Memoria para optar al título de Ingeniería Civil en Informática

UNIVERSIDAD DEL BÍO-BÍO
FACULTAD DE CIENCIAS EMPRESARIALES
ESCUELA DE INGENIERÍA CIVIL INFORMÁTICA

Sistema de administración y gestión para farmacias comunales

Diego Alexis Navarrete Gutiérrez
Juan Carlos Solís Montecino

Profesor Guía
María Antonieta Soto Chico

Enero de 2019 Chillán-
Chile

Memoria para optar al título de Ingeniería Civil en Informática

Agradecimientos

En primer lugar, agradecer a mis padres por su cariño, amor y apoyo que me entregan a diario. A mi madre Gina del Carmen Gutiérrez por orientarme, dar fuerzas y sus consejos, por hacerme más grata esta etapa universitaria.

A mi padre Robinson Navarrete por sus grandes sacrificios por darme la educación, valores, consejos y cariño.

A Paula Espinoza por acompañarme en los momentos más difíciles, aconsejarme y brindarme su cariño incondicional.

.

A mis amigos por brindarme apoyo, momentos de alegría y de diversión durante esta etapa de mi vida.

Por último, a todas las personas que conocí a lo largo de mi vida como universitario, por brindarme la fuerza de llegar hasta aquí y no decaer y cumplir mi sueño de ser un profesional.

Diego Alexis Navarrete Gutiérrez.

En primer lugar, a dios, por brindarme la fuerza de llegar hasta aquí y no decaer y cumplir mi sueño de ser un profesional.

A mis padres por su cariño, amor y apoyo incondicional.
A mi madre Gladys Montecino por orientarme, dar fuerzas en esta etapa universitaria.

A mi padre Juan Solís por sus grandes sacrificios por darme la educación, valores, consejos en los momentos más oportunos

A Ena Villarroel por acompañarme en los momentos más difíciles, aconsejarme y brindarme su amor incondicional

.

A mis amigos en especial a Diego Navarrete y Jehison Villegas por brindarme apoyo, momentos de alegría y de diversión durante esta etapa de mi vida.

A mi abuela Esilda Muñoz a pesar que no esté con nosotros sigue inspirándome, motivándome a ser una mejor persona.

Por último, a todas las personas que me han apoyado en este proceso universitario

Juan Carlos Solís Montecino.

Resumen

Este proyecto es presentado para cumplir con los requisitos exigidos por la Universidad del Bío-Bío, en el proceso de titulación para la carrera de Ingeniería Civil en Informática.

Hoy en día, las farmacias comunales, brindan una ayuda fundamental dentro de la población, ya que, responden a la necesidad de una atención más rápida y acceso a medicamentos a un precio más accesible que las grandes cadenas de farmacias dentro del país. Sin embargo, la farmacia comunal de Chillán carece de un sistema automatizado que apoye de manera conjunta en los procesos de inscripción, venta y gestión en sus medicamentos o clientes.

Debido a lo anterior, el proyecto titulado "Sistema de administración y gestión para farmacias comunales", tiene el objetivo de mejorar la administración, la gestión, y agilizar los procesos de inscripción y ventas, para así ayudar a los trabajadores de la farmacia a optimizar los tiempos y brindar una atención más rápida y personalizada a los clientes.

Para realizar el sistema descrito, se utiliza una metodología de desarrollo iterativo incremental con el objetivo de acercarse a la forma en que trabajan las empresas hoy en día. La metodología escogida es adaptable, comprensible y accesible.

Para el desarrollo del software se utiliza el lenguaje PHP con el framework Laravel, basado en la arquitectura Modelo-Vista-Controlador.

Finalmente, se obtiene un producto software, que contiene todas las funcionalidades requeridas por los funcionarios de la farmacia, de este modo se logra tener un sistema que se adapte a las diferentes tareas que realizan las farmacias comunales.

Abstract

This project is presented to fulfill the requirements demanded by the Universidad del Bío-Bío, in the process of qualification for the career of Civil Engineering in Computing.

Nowadays, the community pharmacies offer a fundamental help within the population, since they respond to the need for faster attention and access to medicines at a more accessible price than the large pharmacy chains within the country. However, the communal pharmacy of Chillán lacks an automated system that jointly supports the registration, sale and management processes of its medications or clients.

Due to the above, the project entitled "Administration and management system for community pharmacies", aims to improve the administration, management, and streamline the registration and sales processes, in order to help pharmacy workers to optimize the times and provide faster and more personalized attention to customers.

In order to carry out the described system, an incremental iterative development methodology is used with the objective of approaching the way in which companies work today. The chosen methodology is adaptable, understandable and accessible.

For the development of the software, the PHP language is used with the Laravel framework, based on the Model-View-Controller architecture.

Finally, a software product is obtained, which contains all the functionalities required by the pharmacy officials, in this way it is possible to have a system that adapts to the different tasks performed by the community pharmacies.

Índice

Índice Tablas

Índice Figuras

Introducción

En la actualidad, el uso de Tecnologías de la Información (TI) es ya una necesidad para las empresas, tanto para diferenciarse de la competencia como para mejorar y optimizar sus procesos productivos. Esto debido a que las TI permiten manejar grandes volúmenes de datos y mejorar la toma de decisiones entregando información precisa y actualizada al instante. Todo esto con el objetivo de mejorar los procesos, el servicio a los clientes y ocupar de forma eficiente los recursos.

Debido a lo anterior, la farmacia comunal de Chillán, requiere gozar de los beneficios que las TI otorgan, mediante un sistema que les permita disponer de información, rápida, actualizada y confiable. La principal información que maneja, son los datos de los medicamentos, clientes, recetas, stock de medicamentos y ventas realizadas.

En este documento se encuentra descrito el proceso realizado para obtener el producto que requiere la farmacia comunal de Chillán. El siguiente capítulo presenta el contexto en el que se desarrolló este proyecto, el problema que se pretende resolver y de qué forma se debe solucionar. Posteriormente, el segundo capítulo estipula el ambiente de ingeniería de software, metodologías de desarrollo y tecnologías a ocupar. El tercer capítulo especifica los requerimientos del software construidos junto al cliente. Luego, el cuarto capítulo corresponde al estudio de factibilidad que determina qué tan viable es el sistema, técnica, operativa y económicamente. El quinto y sexto capítulo especifican el análisis de casos de uso y el diseño tanto de la base de datos como de la interfaz de la aplicación. Posteriormente, el séptimo capítulo corresponde a la seguridad y las pruebas realizadas en el sistema, que permiten verificar la calidad del mismo. Finalmente, el octavo capítulo presenta las conclusiones de este proyecto. Aparte, se incluyen las fuentes bibliográficas usadas y una sección de Anexos que incluye información crucial sobre la cual se construyó este documento.

CAPÍTULO 1
DEFINICIÓN DE LA INSTITUCIÓN

1. Definición Historia de la institución

En este capítulo, se presenta información sobre la institución para la cual se realizará este proyecto, que incluye una descripción de esta institución, su historia y cómo funciona actualmente, los problemas que enfrenta y qué necesitan para logra una mejora en su servicio.

1.1 Historia de la institución

La primera farmacia comunitaria del país fue iniciativa de la Municipalidad de Recoleta. A enero de 2018 son 148 las farmacias comunitarias que se encuentran en funcionamiento en el país, 42 en Santiago y 106 en regiones (Figura 1.). En la región del Bío Bío existen 27 (Fuente Dipol/Minsal). Cabe destacar que no existen datos estadísticos sobre la región de Ñuble, ya que esta se instauró recién en septiembre del 2018.

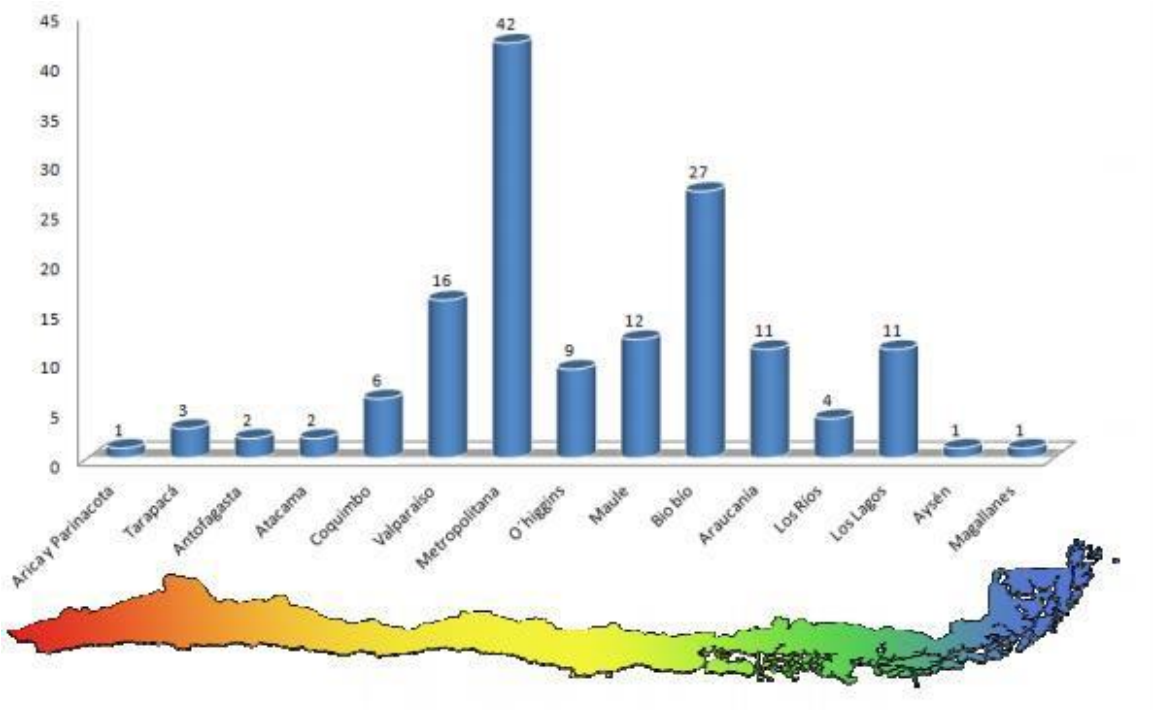

Figura 1. Distribución de farmacias comunales en regiones de chile.

1.2 Descripción de la institución

Una farmacia comunitaria es un establecimiento, por lo general de tipo estatal, que expende o comercializa medicamentos a un precio más bajo que en el mercado farmacéutico. Suelen ser administrados por el gobierno local, en este caso el municipio, siendo considerados como una ayuda social prestada por el estado a personas de bajos ingresos o que gastan un alto porcentaje de su presupuesto familiar en medicamentos.

1.2.1 Organización de la institución

El equipo de personas que trabajan dentro de la farmacia comunal de Chillán se encuentra dentro de las siguientes categorías:

- **Profesionales:** Químico Farmacéutico que ejerce como director de la farmacia.
- **Vendedor:** Técnicos en enfermería nivel superior que desempeñan el rol de auxiliares de farmacias.
- **Personal:** Administrativos, guardia de seguridad y auxiliares de aseo.

1.3 Descripción del problema

A continuación, se presenta la descripción de los problemas que están presente en la farmacia comunal:

Uno de los inconvenientes que pueden presentar estas instituciones es el gran dinamismo y heterogeneidad que existe entre ellas, ya que cada municipio utiliza distintos organigramas y procesos administrativos. Estos problemas son:

- Los mecanismos de abastecimiento.
- El tipo de recetas médicas que se pueden aceptar.
- La Inscripción de los clientes con enfermedades crónicas.
- El tipo de productos que pueden intermediar con los usuarios.

Otro de los problemas en la implementación de estas farmacias es el abastecimiento, ya que el sistema actual de adquisiciones cuanta con varios mecanismos como es el convenio marco, licitación, trato directo, compra directa y Cenabast. Debido a esto, se generan demoras en la entrega en los medicamentos a los clientes de la institución, donde informan sus requerimientos.

Por lo tanto, en la mayoría de los casos estos establecimientos de salud se abastecen con los tratamientos farmacológicos, dispositivos médicos y suplementos que han sido solicitados con antelación por los clientes. Además, cuando un cliente se acerca a una farmacia busca comenzar con su tratamiento tan pronto como le ha sido indicado. Sin embargo, se presenta dificultad en este aspecto debido a que los procesos de adquisición en el sector público requieren tiempos que en muchos casos puede llevar semanas hasta meses, impactando directamente en su salud y dificultando su adherencia al tratamiento indicado.

Estas esperas se pueden ver incrementadas si el mecanismo de adquisición no satisface los requerimientos técnicos de compra. Así, el modelo actual de abastecimiento de la institución presenta falencias significativas que afecta la entrega de un mejor servicio a sus clientes [1].

CAPÍTULO 2
DEFINICIÓN PROYECTO

2. Definición del proyecto

A continuación, se define los objetivos del proyecto general y específicos, el ambiente de software con el cual se realiza el proyecto, herramientas, técnicas y notación.

2.1 Objetivos del proyecto

A continuación, se define el objetivo general y los objetivos específicos los cuales deben ayudar a resolver los problemas que presenta la institución.

2.1.1 Objetivo general

Desarrollar un sistema web que permita tener control de los clientes, medicamentos, registro de ventas y generación de reportes para el apoyo del área de abastecimiento, todo esto para mejorar el servicio de la farmacia comunal.

2.1.2 Objetivos Específicos

1. Diseñar una aplicación web que permita llevar control de los clientes inscritos y medicamentos disponibles.
2. Generar reportes relacionados con el área de abastecimiento.
3. Llevar un registro de ventas de los medicamentos.
4. Permitir el acceso al sistema a solo personal pertinente.

2.2 Ambiente de Ingeniería de Software

A continuación, se describe la metodología de desarrollo, las tecnologías empleadas para el desarrollo del proyecto, así como técnicas y notaciones.

2.2.1 Metodología de desarrollo

Se eligió el uso de la metodología iterativa incremental, ya que permite desarrollar incrementos en un periodo muy corto, usando un enfoque impulsado por el cliente, además de reaccionar mejor al cambio en los requisitos y es ideal considerando un desarrollo individual, permitiendo llevar un mejor seguimiento a las actividades realizadas y ayudando a mejorar la retroalimentación en cada ciclo, logrando evitar los errores de la iteración anterior.

Esta metodología consiste en llevar el desarrollo de software en incrementos y entender estos como un mini proyecto, en cada incremento se realizan una o varias iteraciones, repitiendo un proceso de trabajo similar para proporcionar un resultado completo al final. Además, como cada incremento debe dar como resultado un producto funcional, esto permite gestionar las expectativas del cliente, ya que cada cierto tiempo puede ver los resultados del desarrollo.

2.2.2 Tecnologías

A continuación, se describe las tecnologías y herramientas a emplear para el desarrollo del proyecto:

- **Visual studio code:** Es un editor de texto y código multiplataforma. Admite la apertura de múltiples documentos mediante el uso de pestañas. Este soporta un gran número de lenguajes tales como: C, C++, C#, CSS, HTML, Java, JavaScript, PHP, Python, SQL, etc [3].
- **MariaDB:** MariaDB es un fork de MySQL que nace bajo la licencia GPL. Nos permite administrar bases de datos relacionales, siendo un reemplazo de MySQL con más funcionalidades y mejora de rendimiento [4].
- **Laravel Framework:** Laravel es un framework de código abierto que permite desarrollar aplicaciones y sitios web con PHP. Su principal objetivo es permitir el uso de una sintaxis refinada y expresiva para así crear un código ordenado y sencillo [5].
- **HTML5:** Es un lenguaje utilizado para presentar y estructurar el contenido web. Soporta contenido multimedia (audio y video), lo que posibilita incluir y controlar contenido web [6].
- **CSS3:** Sirve para definir el estilo o la apariencia de las páginas web, escritas con HTML o de los documentos XML. Además, fija la estética de un sitio web o, también, puede cambiarla.
- **JQuery:** JQuery es una librería de JavaScript. Esta librería de código abierto, simplifica la tarea de programar en JavaScript y permite agregar interactividad a un sitio web sin tener conocimientos del lenguaje [7].
- **Sublime text 3:** Es un editor de texto y código multiplataforma. Admite la apertura de múltiples documentos mediante el uso de pestañas. Soporta un gran número de lenguajes tales como: C, C++, C#, CSS, HTML, Java, JavaScript, PHP, Python, SQL, etc.

2.2.3 Técnicas y notaciones

En el trascurso del desarrollo del proyecto seutilizan las siguientes técnicas y notaciones:

- **BPMN (Business Process Model and Notation):** Es unanotacióngrafica que describe la lógica de los pasos de un proceso denegocio.(OMG, 2014)
- **UML (Unified Modeling Languaje):** Es un lenguaje gráfico para visualizar, especificar y documentar cada una de las partes que comprende el desarrollo de software. (OMG)
- **Diagrama de casos de uso:** Un diagrama de caso de uso es una descripción de las acciones de un sistema desde el punto de vista del usuario. Cada caso de uso proporciona uno o más escenarios que indican cómo debería interactuar el sistema con el usuario o con otro sistema para conseguir un objetivo específico.

Todo sistema de software ofrece a su entorno una serie de servicios. Un caso de uso es una forma de expresar cómo alguien o algo externo al sistema lo usa. Cuando se dice "alguien o algo" se hace referencia a que los sistemas son usados no sólo por personas, sino también por otros sistemas de hardware o software.

2.3 Arquitectura de Software

A continuación, de define la arquitectura escogida para el desarrollo.

2.3.1 Laravel y su Arquitectura

Laravel es un framework que sigue la arquitectura Modelo-Vista-Controlador (MVC) pero no utiliza esta arquitectura de por sí. MVC le ayuda a separar las consultas de la base de datos (el Modelo) de la lógica relacionada con la forma en que se deben procesar las solicitudes (el Controlador) y cómo se debe representar el diseño (la Vista), también se implementa las rutas para hacer las peticiones al servidor por medio del controlador. En la figura 2, se muestra el funcionamiento de una aplicación típica de Laravel [8].

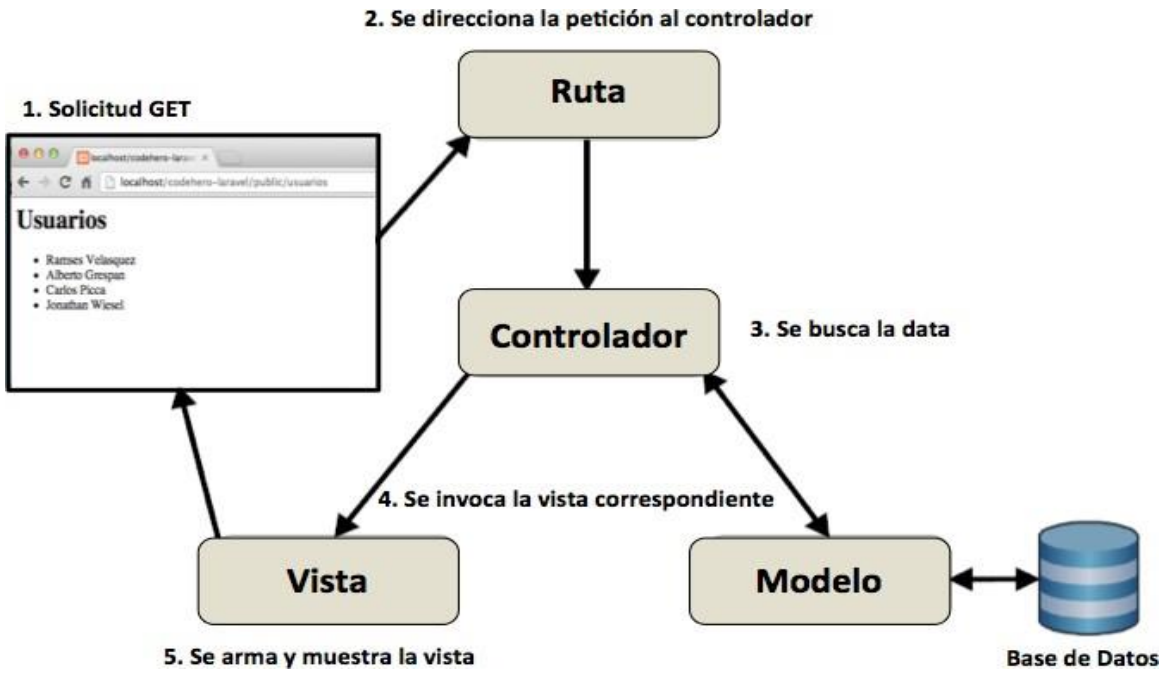

Figura 2. Arquitectura de Laravel [9].

A continuación, se describen los componentes de la arquitectura de Laravel, utilizando ejemplos para una mejor compresión de esta arquitectura [9]:

Rutas

Cuando el servidor recibe una solicitud HTTP, Laravel intenta hacerla coincidir con una ruta registrada dentro de cualquiera de los archivos de ruta. Todos los archivos de ruta se encuentran

dentro del directorio de rutas. Dentro del directorio del proyecto esta routes/web.php aloja las rutas para llamar a los diferentes controladores y dentro de él están cada uno de los métodos a usar.

Para hacer uso de estas rutas se utilizan peticiones HTTP por medio de una URL se determina el método a usar del controlador correspondiente. Las rutas más comunes son:

- GET /medicamentos/: recupera todos los medicamentos.
- POST /medicamentos: crea un nuevo medicamento y lo inserta en la base de datos.
- PUT /medicamentos: actualiza un medicamento existente que coincida con el id.
- DELETE/medicamentos: elimina el medicamento con el id proporcionado.

Modelo

El modelo es la capa que se encuentra en la parte superior de la base de datos, ocultando toda la jerga específica de la base de datos. Laravel usa Eloquent ORM para modelar la base de datos. Este último proporciona una implementación sencilla para trabajar con una base de datos. Cada tabla de la base de datos tiene un "Modelo" correspondiente que se utiliza para interactuar con esa tabla.

Controlador

La clase Controller se compone de varios métodos (índice, mostrar, almacenar, actualizar y eliminar) que corresponden a diferentes peticiones HTTP que son solicitadas desde las rutas. Laravel hacer coincidir el nombre de instancia del controlador con la URL en rutas.

Vista

Las vistas en Laravel son la parte pública de nuestro sistema, se escriben en HTML junto con un motor de plantillas llamado Blade. Laravel, por defecto, trabaja con la idea de que tenemos que escribir la menor cantidad de código repetido, modularizar nuestro código en donde más se pueda, y esto lo aplicamos en nuestros modelos, controladores y demás partes de nuestro proyecto.
Laravel trabaja con un motor de plantillas Blade que se definirá a continuación:

Blade

Blade provee de muchas ventajas además de modularizar nuestras vistas de una forma sorprendente, también nos permite usar estructuras de control y variables de PHP directamente en ellas.

Base de datos

Laravel permite generar las tablas de las bases de datos desde el mismo proyecto a partir de las migraciones, las cuales una vez terminadas son importadas creando cada una de las tablas de la base datos.

2.3.2 REST

En Laravel se utiliza API REST, pero antes de ver cómo es su utilización, se describe qué es REST a continuación [9]:

La Transferencia de Estado Representacional (REST) es cualquier interfaz entre sistemas que use HTTP para obtener datos o generar operaciones sobre esos datos en todos los formatos posibles,

como XML y JSON. Una forma simple de organizar interacciones entre sistemas independientes. Ha estado creciendo en popularidad desde 2005 e inspira el diseño de servicios, como la API de Twitter.

Algunas características de REST:

- Protocolo cliente/servidor sin estado: Cada petición HTTP contiene toda la información necesaria para ejecutarla, por tanto, esto permite que ni cliente ni servidor necesiten recordar ningún estado previo. No obstante, existen algunas excepciones y hay algunas aplicaciones HTTP que incorporan memoria caché, para que así, el cliente pueda ejecutar en un futuro la misma respuesta para peticiones idénticas.

- Cuatro operaciones más importantes: Las operaciones más importantes relacionadas con los datos en cualquier sistema REST y la especificación HTTP son cuatro; POST (crear), GET (leer y consultar), PUT (editar) y DELETE (borrar).

- Objetos en REST manipulados con URI: La URI es el identificador único de cada recurso de un sistema REST. Esta, nos facilita el acceso a la información, para poder modificarla o borrarla. También para compartir su ubicación exacta a terceros.

- Interfaz uniforme: Para poder realizar una transferencia de datos en un sistema REST, este aplica acciones concretas (POST, GET, PUT y DELETE) sobre los recursos, siempre y cuando estén identificados con una URI. Esto lo que permite es facilitar la existencia de una interfaz uniforme que sistematiza el proceso con la información.

- Sistema de capas: Su estructura o arquitectura es jerárquica entre sus componentes, y cada una de estas capas, lleva a cabo una funcionalidad dentro del sistema REST.

Ventaja del uso de REST:

- Separación entre el cliente y el servidor: Esto mejora la portabilidad de la interfaz a otro tipo de plataformas, así como, aumenta la escalabilidad de los proyectos y permite que los distintos componentes desarrollados puedan evolucionar de forma independiente.

- Visibilidad, fiabilidad y escalabilidad: Es evidente que la separación entre clientes y servidor permite que cualquier equipo de desarrollo pueda escalar el producto sin excesivos problemas. Por tanto, permite que se pueda migrar a otros servidores o realizar todo tipo de cambios en la base de datos, siempre y cuando los datos de cada una de las peticiones se envíen de forma correcta.

- Independiente de plataformas o lenguajes: Se adapta al tipo de sintaxis o plataformas con las que se estén trabajando. Esto ofrece una gran libertad a la hora de probar o cambiar nuevos entornos. La API REST permite tener servidores PHP, Java, Python o Node.js.

2.3.3 VUE JS

Si bien Laravel posee un motor de plantillas propio como es blade, en este proyecto es complementado con Vue js para tener una mayor separación de elementos a nivel de front-end, a continuación, se presenta una definición de Vue js:

Vue js es un framework progresivo para crear interfaces de usuario. A diferencia de otros framework monolíticos, Vue está diseñado desde cero para que pueda ser adoptado gradualmente. La biblioteca central se enfoca solo en la capa de vista y es fácil de captar e integrar con otras bibliotecas o proyectos existentes. Por otro lado, Vue también es perfectamente capaz de impulsar aplicaciones sofisticadas de una sola página cuando se usa en combinación con herramientas modernas y bibliotecas de soporte [10].

Algunas características de Vue js [11]:

- Flexibilidad: Le permite al usuario escribir su plantilla en un archivo HTML, un archivo JavaScript y un archivo JavaScript puro utilizando nodos virtuales.
- Optimizado: Su core ocupa 74KB.
- Comunidad: Va creciendo a un ritmo importante con más 66500 estrellas en GitHub y 130 personas contribuyendo al core cada día.

Ventajas del uso de Vue js [12]:

- El dato como centro de todo: En VueJS, los componentes gestionan un modelo de datos interno. Estos componentes están diseñados bajo el patrón MVVM. Esto quiere decir que el desarrollador no tiene que preocuparse tanto por cómo o cuándo renderiza un modelo en pantalla y sí más en cómo tiene que ser la lógica que gestiona ese modelo.
- El sistema de componentes: VueJS sabe comunicarse muy bien por medio de eventos asíncronos. Un componente hijo se puede comunicar con su componente padre por medio de eventos. Dos partes del sistema pueden comunicarse por medio de eventos. Los propios modelos de un componente son capaces de enviar eventos para indicar cuándo renderizarse.

CAPÍTULO 3
ESPECIFICACIÓN DE REQUERIMIENTOS DE SOFTWARE

3. Especificaciones del software

A continuación, se presenta los requerimientos del software, su descripción global, alcances y limitaciones.

3.1 Requisitos funcionales

A continuación, se presenta una lista de los requisitos funciones del software:

1. El Sistema debe mantener un control de los clientes que atiene la institución farmacéutica.
2. El Sistema debe mantener un control de los medicamentos.
3. El Sistema debe mantener un registro de las recetas de sus respectivos clientes.
4. El Sistema debe contar con un ingreso al sistema (login), de modo que solo los funcionarios autorizados tengan acceso a los datos y funcionalidades de este.
5. El Sistema debe mantener un control de las ventas que se realizan.
6. El Sistema debe listar las recetas de los clientes según la fecha de ingreso.
7. El sistema debe generar reportes que apoyen al área de abastecimiento de la institución farmacéutica.

3.1.2 Requisitos No Funcionales

A continuación, se presenta una lista de los requisitos no funciones del software:

1. El sistema debe ser compatible con el sistema operativo Windows.
2. El sistema debe ser intuitivo para los funcionarios con opciones de trabajo claras y precisas.
3. El acceso al sistema debe ser seguro, permitiendo el acceso solo por medio de usuario y contraseña generados por el mismo sistema al momento de registrar al funcionario.

3.2 Requisitos Operacionales

A continuación, se presenta los requisitos operaciones del proyecto:

Para el correcto funcionamiento de este sistema la empresa debe contar con al menos un computador, una conexión a Internet estable y una impresora para la generación de informes.

Además, debe contar con personal que maneje aspectos básicos de computación, para el uso del sistema, ingresar, editar y eliminar datos, generar informes, acceder al sistema.

3.3 Requisitos globales del Software

A continuación, se presentan los requerimientos para el correcto funcionamiento de software del proyecto:

Concepto	**Requerimiento**
Sistema Operativo	Microsoft Windows 7/8.1/10
Navegador	Google Chrome/ Mozilla Firefox
Conexión a internet	Obligatoria
RAM	4 GB (Mínimo)

Tabla 1: Requerimientos de software.

3.3.1 Interfaz de usuario

La interfaz de la aplicación debe ser intuitiva, sencilla y mantener colores sobrios que sean apropiados con un sistema de farmacias, sin que esto afecte la usabilidad del sistema.

3.3.2 Interfaz de hardware

Es necesario tener la computadora conectada a la red y en conjunto con una impresora para poder para imprimir las boletas e informes generados por el sistema. Los requisitos básicos serían una entrada ethernet, conexión USB e impresión blanco y negro.

3.3.3 Interfaz De Software

El sistema web requerirá de un navegador Web de cualquier proveedor, aunque se recomienda usar Google Chrome, Mozilla Firefox, Opera, por la confiabilidad que estos ofrecen.

3.4 Alcance

A continuación, se presenta los alcances que presenta el sistema:

- El sistema sólo estará disponible para los funcionarios de la institución farmacéutica.
- El sistema se desarrollará, según los requerimientos de los funcionarios de la institución farmacéutica.
- El sistema sólo podrá registrar clientes de la comuna en la cual se implementó.

3.5 Limitaciones

A continuación, se presenta las limitaciones del sistema:

- El sistema no puede funcionar sin una conexión a Internet.

CAPÍTULO 4
Estudio De Factibilidad

1. Estudios de factibilidad

En este capítulo, se presenta el estudio de factibilidad técnica del proyecto, en el cual se define el hardware y software requerido, también se presenta el análisis de factibilidad operativa y económica. Este último considera la inversión, costos y beneficios incluyendo el beneficio social que aporta este proyecto.

1.1 Factibilidad técnica

A continuación, se da a conocer el estudio de factibilidad técnica detallando el software y hardware requerido (Tabla 2 y Tabla 3).

Detalle	**Descripción**	**Licencia**
Lenguaje de programación	PHP	Gratuita
Base de datos	MaríaDB	Gratuita
Lenguaje de mercado	HTML 5, JavaScript, CCS3, AJAX y JQuery	Gratuita
Framework	Laravel	Gratuita
Plantilla	CoreUI	Gratuita
Entorno de desarrollo	Sublime Text y Visual Studio Code	Gratuita

Tabla 2. Software Requerido.

Como se aprecia en la tabla 2, el software no requiere la compra de licencias, esto ayuda a reducir el monto de la inversión.

Detalle	**Escificaciones**
Procesador	Intel core i3 / Intel core i7
Memoria Ram	4 GB/ 8GB
Disco Duro	SSD 250 GB/ SSD 500 GB
Sistema Operativo	Windows 10 Home Edition

Tabla 3. Hardware empleado.

Por su parte, en la tabla 3 se describe el hardware requerido. Cabe mencionar que este corresponde a equipos propios, por lo que, no se toman en cuenta para la inversión del proyecto.

1.2 Factibilidad operativa

Los funcionarios que darán uso a este sistema poseen conocimientos de computación básico-medio, se usó como referencia la farmacia comunal de Chillán. Sin embargo, para garantizar el buen funcionamiento y uso del programa, se requerirá de una capacitación básica.

En conclusión, de acuerdo a los antecedentes recabados, este proyecto es factible de implementar operacionalmente.

1.3 Factibilidad económica

A continuación, se presenta la factibilidad económica, detallando la inversión, los costos, así como, los beneficios tangibles y beneficios intangibles del proyecto.

1.3.1 Inversión

Los ítems considerados para determinar la inversión del proyecto son los siguientes (Ver Tabla 4):

- **Sueldo Ingeniero en Informática:** Corresponde a $800.000 al mes (aproximadamente 180 horas de trabajo mensual).
- **Capacitación de funcionarios:** Curso destinado a los usuarios finales, en el que se explica cómo utilizar el software, una vez que se entregue a la farmacia.
- **Hardware:** Costo promedio de un computador con las características detalladas en la Tabla 3. Este tiene un valor aproximado de $300.000.
- **Hosting**: Hosting ofrecido por hostchile.cl, con requisitos suficientes para el funcionamiento del sistema, tiene un costo anual de $29.000.
- **Dominio (primer pago):** Inscripción de dominio al que se asociará nuestro sistema, esta tarifa anual según lo definido por la página de "NIC Chile" es alrededor de $10.000.

Ítem	**Costo ($)**
Sueldo de 2 Ingenieros en informática	$3.200.000 (por 2 meses de trabajo)
Capacitación de usuario	$100.000
Hardware	$300.000
Hosting	$29.000
Dominio	$10.000
Total	$3.621.000

Tabla 4. Resumen inversión.

La inversión será aproximadamente de $3.621.000 (Tabla 4).

1.3.2 Costos

Para los costos asociados al proyecto, se toma en cuenta la mantención del software y pago del dominio.
Mantención: Esta se debe ejecutar por los ingenieros en informática que realizaron el proyecto, estos deben mantener el sistema dos veces en el año (2 días trabajados que corresponde a 18 horas en total, con un valor de $4.500 la hora lo que da un total de $81.000).

Dominio: Compra de dominio donde se alojará nuestro sistema, esta tarifa anual definida por en página "NIC Chile" es alrededor de $10.000.

Hosting: Hosting ofrecido por hostchile.cl, con requisitos suficientes para el funcionamiento del

sistema, tiene un costo anual de $29.000.

DETALLE	COSTO ($)
Mantención	$81.000
Dominio	$10.000
Host	$29.990
Total	$120.990

Tabla 5: Resumen de costos.

El costo anual del proyecto será de $120,990.

1.3.3 Beneficios tangibles

1. **Reducción del tiempo de registro:** Tal como se mencionó, el software ayuda a mejorar los tiempos de registro de los fármacos y los clientes, Así el promedio de tiempo que dedican los funcionarios a estas tareas se reduce.

2. **Reducción de tiempo de atención a clientes:** El sistema ayudará a reducir el tiempo de atención entre cada cliente ya que, hoy en día los tiempos de atención se extienden por falta de software de apoyo para ejecutar las diversas funciones diarias de la farmacia.

   Consideraciones:

   Los datos que se presentan a continuación fueron obtenidos en la farmacia comunal de Chillán. Además, para el cálculo de estos beneficios tangibles consideramos solo a los vendedores, ya que estos interactúan la gran parte del tiempo con los clientes y el sistema [2].

   - Promedio total a pagar en una venta: 12.623
   - Promedio ventas realizadas anuales: 15.844
   - Promedio de personas atendidas diariamente: 60
   - Promedio de minutos de atención: 8
   - Promedio de personas atendidas en una hora: 7
   - Promedio salario del vendedor (TENS): 400.000 por 160 horas de trabajos (20 días de trabajo al mes[1].

Tipo	Detalle	Anual	Total
Reducción de tiempo de atención y registros de clientes	Se produce como consecuencia de la mejora de tiempos de los funcionarios en registrar los fármacos, recetas y clientes.	-Ahorro de tiempo mensual es de 11 horas considerando que se ahorra diariamente 33 minutos y 20 días de trabajo en 1 mes.	-Considerado al vendedor (TENS) con un sueldo de $400.000 mensual por 160 horas de trabajo mensual

[1] Fuente: https://www.indeed.cl/salaries/T%C3%A9cnico/a-en-enfermer%C3%ADa-Salaries

	Además, ayuda a reducir los tiempos de atención de los clientes. Con los datos obtenidos de la farmacia comunal de Chillán la aplicación ayuda a mejorar ese promedio de atención a un total de 7 minutos. Por hora se atenderían a 8 personas. Por lo que el ahorro en tiempo sería de 33 minutos diarios.	-Ahorro de tiempo anual es de 132 horas.	($2.500 la hora de trabajo) el ahorro corresponde a $27.500 mensual. -Anualmente el ahorro es de $330.000 -Considerando que hay 2 vendedores en la farmacia, el ahorro es de $660.000 anual.
Total			$660.000 anual.

Tabla 6: Cálculo de beneficios

Flujo de caja

	Año 0	Año 1	Año 2	Año 3	Año 4	Año 5
(+) Ingresos						
Beneficios	0	660.000	660.000	660.000	660.000	660.000
(-) Costos						
Hosting	0	29.000	29.990	29.990	29.990	29.990
Dominio		10.000	10.000	10.000	10.000	10.000
Mantención		81.000	81.000	81.000	81.000	81.000
Desarrollo	3.621.000					
Total		539.010	539.010	539.010	539.010	539.010

Tabla 7: Cálculo flujo de caja

Cálculo de VAN

A continuación, la Tabla 8 muestra el cálculo del flujo de caja del proyecto y el valor actual neto, para determinar si este es factible económicamente.
Tasa de descuento: 10%
Años: 5

	Año 0	Año 1	Año 2	Año 3	Año 4	Año 5
Inversión	3.621.000					
Gastos		120.990	120.990	120.990	120.990	120.990
Beneficios		660.000	660.000	660.000	660.000	660.000
Flujo		539.010	539.010	539.010	539.010	539.010

Tabla 8: Cálculo de VAN

La fórmula para calcular el van es la siguiente:

$$VAN = -I_0 \sum_{t=1}^{n} \frac{Ft}{(1+k)^t}$$

Figura 3: Fórmula del VAN.

Donde, Ft son los flujos de dinero en cada periodo t, I0 es la inversión realizada en el momento inicial (t = 0), n es el número de periodos de tiempo y k es el tipo de descuento o tipo de interés exigido a la inversión.

En este caso, la tasa de descuento corresponde a un 10%, y la inversión como se presentó anteriormente, corresponde a $3.592.000. Por lo tanto, el VAN a cinco años corresponde a $-1.577.728. Aunque el VAN arroje un resultado negativo, el proyecto está enfocado en los beneficios intangibles.

1.3.4 Beneficios intangibles

A continuación, se presentan los beneficios intangibles del proyecto:

1. Incrementar la satisfacción de los funcionarios: El software suple actividades que antes los funcionarios tenían que hacer de forma paralela en otros programas, por lo que no se tendrán que realizar tareas extras.
2. Brindar al cliente una atención más personalizada: El sistema brindará a los funcionarios mayor conocimiento de los clientes que acuden a la farmacia.

CAPÍTULO 5
ANÁLISIS

2. Análisis

En este capítulo se presenta el análisis del ámbito del proyecto, para comprender claramente las necesidades del cliente.

2.1 Casos De Uso

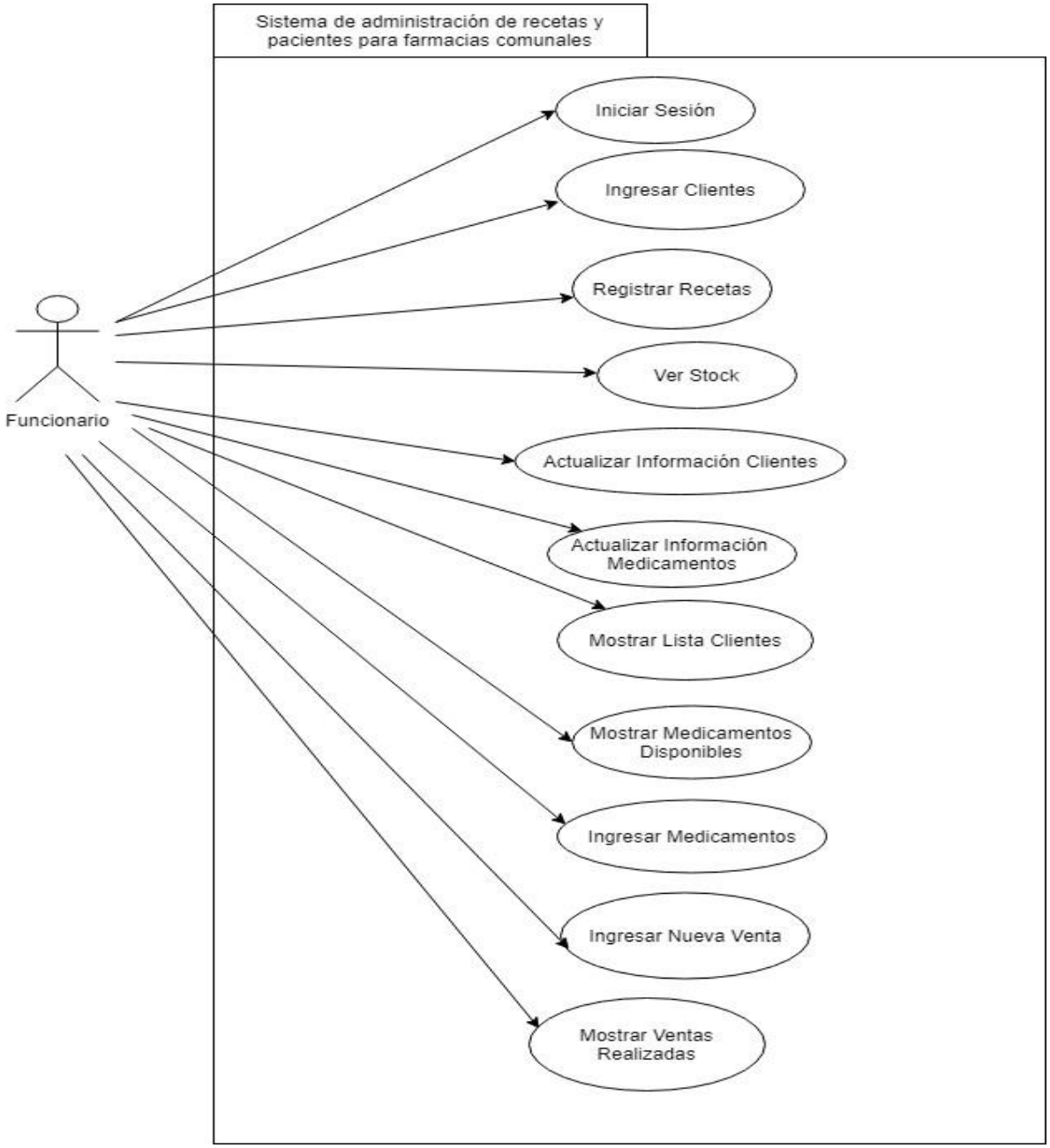

Figura 4. Diagrama casos de uso funcionario.

2.1.1 Descripción actores casos de uso

Cuatro de los actores que se describen en esta sección se representan como funcionarios en el modelo de caso de usos de la figura 4, los demás diagramas de casos de uso se encuentran en el Anexo 1.

- **Farmacéutico:** Usuario del sistema, cuyo rol principal se centra en el editar, eliminar y registrar medicamentos, recetas y clientes, realizar ventas y ver gráficos estadísticos. Este usuario no tendrá acceso al módulo de ingresar usuario y ver los roles que desempeñan.

- **Vendedor:** Usuario del sistema, que hace referencia a la persona que genera ventas de medicamentos. Además, tiene acceso a módulos de ingreso de cliente, registro de recetas, ver medicamentos disponibles, actualizar información de los clientes, listar clientes y generar ventas. Cabe notar que este usuario no podrá actualizar ni ingresar los medicamentos.

- **Bodeguero:** Actor que tiene como rol principal el ingreso, actualización y reporte de los medicamentos del sistema. Además, puede ver las estadísticas de los clientes ingresados y las ventas. Este usuario no tendrá acceso a otras funcionalidades del sistema tales como las ventas, registro de clientes, etc.

- **Administrador:** Actor que tiene como rol principal el registro de funcionarios en el sistema, crear perfiles, borrar usuarios y acceder libremente a la base de datos. Además, posee acceso a todos los módulos y funcionalidades.

2.1.2 Descripción de casos de uso

A continuación, se describen algunos de los casos de uso del sistema (el resto de la descripción de los casos de uso se encuentran en Anexo 2.

Caso de uso: Ingresar Clientes
ID: CU_2
Breve Descripción: Permite al Administrador/Farmacéutico/Vendedor ingresar un nuevo cliente en el sistema.
Actores Principales: Administrador/Farmacéutico/Vendedor
Actores Secundarios: Ninguno
Precondición: Administrador/Farmacéutico/Vendedor debe haber iniciado sesión.
Flujo Principal: 1. El caso de uso comienza cuando el administrador/farmacéutico/vendedor escoge la opción de ingresar un nuevo cliente en el sistema. 2. El sistema presenta formulario para el ingreso del cliente. 3. El sistema verifica que los datos (id, Rut, nombre, apellido, fecha, teléfono, domicilio, estado tratamiento) del nuevo cliente ingresado son correctos y que no se encuentra registrado en el sistema. 4. El sistema registra el nuevo cliente.
Postcondición: El sistema aumenta con un nuevo cliente registrado.
Flujos Alternativos:

3a.1 El sistema detecta errores en los datos ingresados. 3a.2 El sistema muestra un mensaje de error. 3a.3 El sistema vuelve al paso 2. 3b.1 El sistema detecta que el cliente ya está registrado. 3b.2 El sistema mostrará un mensaje avisando que el cliente ya se encuentra registrado. 3b.3 El sistema vuelve al paso 2.

Tabla 9: CU Ingresar Clientes.

Caso de uso: Ingresar Recetas
ID: CU_3
Breve Descripción: Permite al Administrador/Farmacéutico/Vendedor ingresar una nueva receta al sistema.
Actores Principales: Administrador/Farmacéutico/Vendedor
Actores Secundarios: Ninguno
Precondición: Administrador/Farmacéutico/Vendedor debe haber iniciado sesión.
Flujo Principal: 1. El caso de uso comienza cuando el administrador/farmacéutico/vendedor escoge la opción de ingresar una nueva receta en el sistema. 2. El sistema presenta formulario para el ingreso de una nueva receta. 3. El sistema verifica que los datos (id, médico, contacto médico, fecha creación) de la nueva receta ingresada son correctos y que esta no se encuentra registrada en el sistema 4. El sistema registra la receta.
Postcondición: El sistema aumenta con una nueva receta registrada.
Flujos Alternativos: 3a.1 El sistema detecta errores en los datos ingresados. 3a.2 El sistema muestra un mensaje de error. 3a.3 El sistema vuelve al paso 2. 3b.1 El sistema detecta que la receta fue ingresada. 3b.2 El sistema mostrará un mensaje avisando que la receta ya se encuentra registrado. 3b.3 El sistema vuelve al paso 2.

Tabla 10: CU Ingresar Recetas

Caso de uso: Mostrar Clientes
ID: CU_7
Breve Descripción: Permite al
Breve Descripción: Permite al Administrador/Farmacéutico/Vendedor ver los clientes que están inscritos en el sistema.
Actores Principales: Administrador/Farmacéutico/Vendedor
Actores Secundarios: Ninguno
Precondición: 1. Administrador/Farmacéutico/Vendedor debe haber iniciado sesión.
Flujo Principal:

1. El caso de uso comienza cuando el Administrador/Farmacéutico/Vendedor elige la opción Mostrar clientes. 2. El sistema mostrará el listado de los clientes registrados en el sistema.
Postcondición: Ninguna.
Flujos Alternativos: 2a.1 El sistema no posee ningún cliente registrado 2a.2 El sistema no mostrará ningún dato relacionado con los clientes ni las opciones tales como editar, eliminar. 2b.1 No existe cliente al buscar el nombre en el buscador. 2b.2 El sistema no mostrará ningún dato relacionado con ese cliente ni las opciones tales como editar, eliminar.

Tabla 11: CU Listar Clientes

5.2 Descripción de proceso de negocio

A continuación, en la figura 5 se presenta la descripción del proceso de negocio de venta en la farmacia comunitaria [1]:
Primero, el cliente se dirige a la farmacia comunitaria que le corresponde, una vez en el lugar, presenta su receta médica al farmacéutico de turno el cual revisa y ve si está todo correcto y lo incorpora al sistema. Ya inscrito se procede a realizar la venta de los medicamentos solicitados por el cliente. Luego, revisa si dichos fármacos están disponibles y los entrega. En caso de que un medicamento no se encuentra disponible se realiza lo siguiente:

- Se genera una solicitud de compra, la que es administrada por el organismo de gestión de abastecimiento.
- Una vez realizada la solicitud de compra se realizar una cotización a los diferentes laboratorios, luego se recibe la respuesta de estas cotizaciones y se genera la orden de compra que se demora entre 5 a 14 días.
- Finalmente, el o los proveedores realizan el despacho de los medicamentos solicitados hacia la farmacia, esto se demora entre 24 horas hasta 2 semanas.

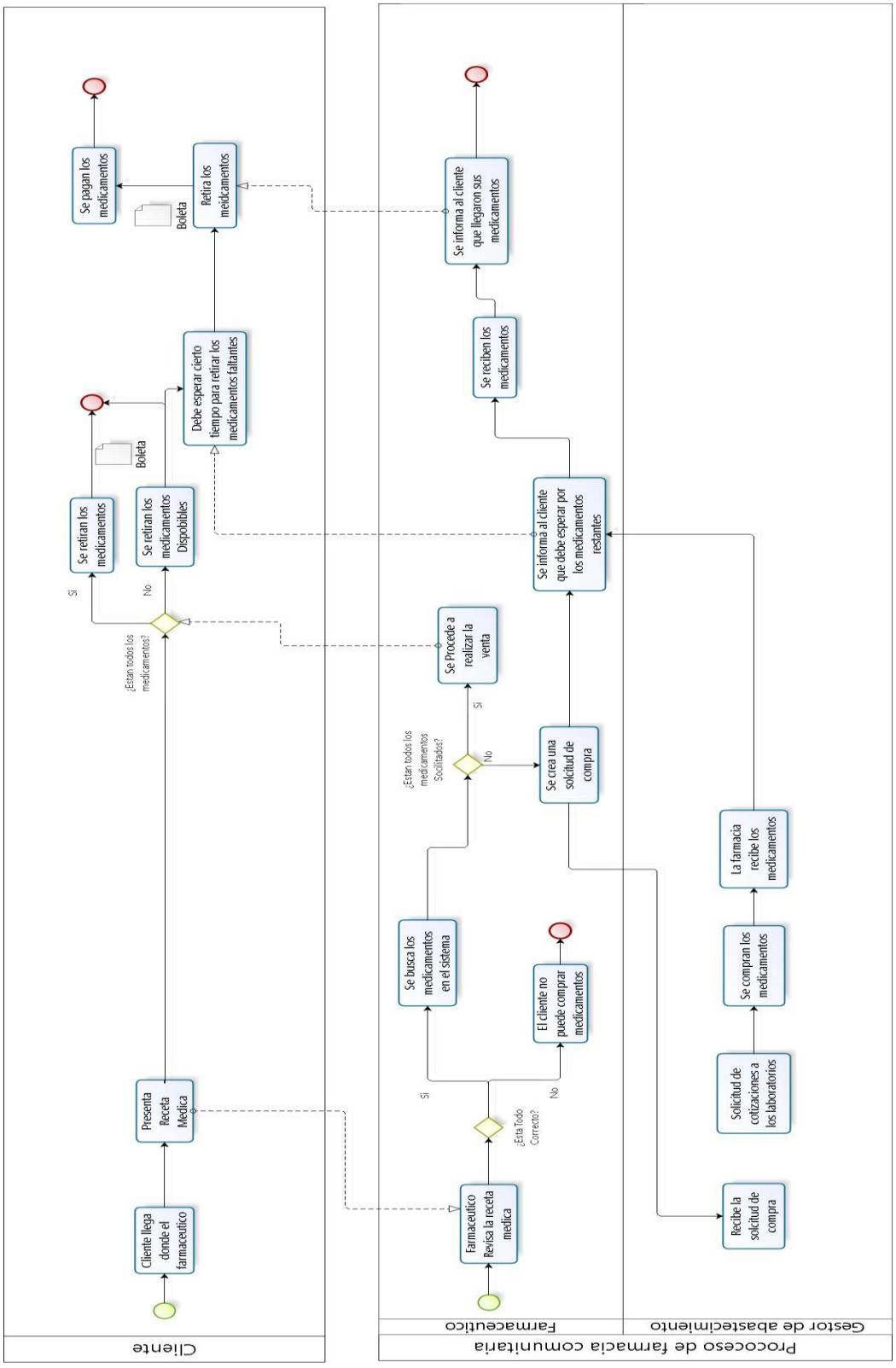

Figura 5: Procesos de negocios farmacia comunal.

5.3 Modelo de datos

El modelo entidad relación es una herramienta para el modelado de datos que permite representar las entidades relevantes de un sistema de información, así como sus interrelaciones y propiedades. Este soporta la lógica de negocio del sistema (Ver Figura 6).

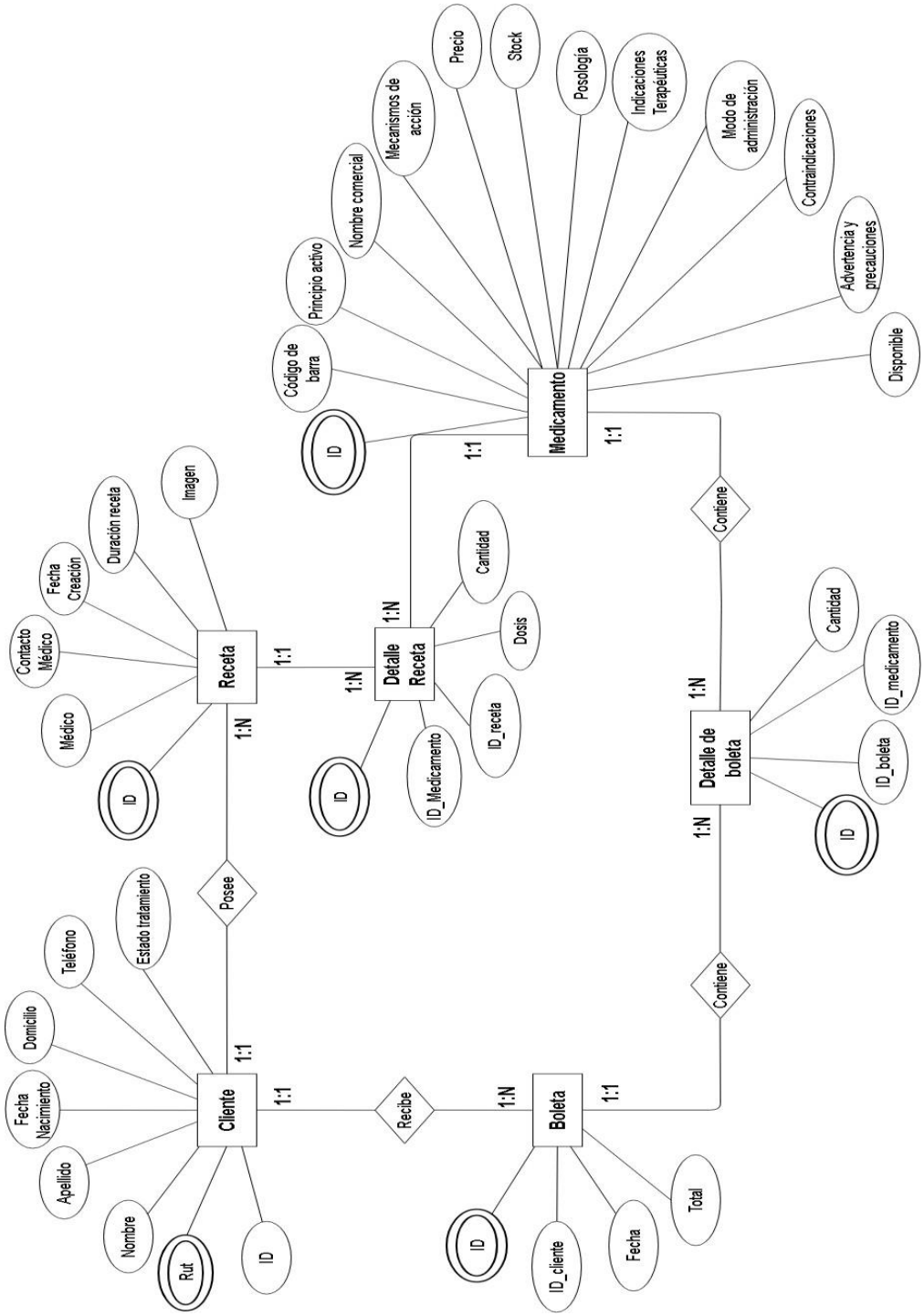

Figura 6. Modelo de datos del sistema.

5.3.1 Descripción de entidades

A continuación, se describen las entidades del sistema:

- **Cliente:** Hace referencia a los clientes de la farmacia, los cuales se inscriben en el sistema para generar las ventas, un cliente posee: ID, Rut (Elemento identificatorio), nombre, apellidos, fecha de nacimiento, domicilio, teléfono y estado del tratamiento, este último nos indica si el cliente sigue con su tratamiento o no. El atributo ID no se utiliza como identificador de un cliente, el ID en este caso se utiliza para ordenar a los clientes dentro del sistema acorde a su ID de ingreso.
- **Boleta:** Comprobante de una venta realizada que tiene como atributos el ID de la boleta, ID del cliente, fecha de ingreso y el total a pagar.
- **Receta:** Entidad del sistema que nos permite conocer los medicamentos que han sido prescritos a los clientes, tiene como atributos: Numero de receta (Elemento identificatorio), nombre del médico, contacto del médico, fecha creación, duración e imagen.
- **Detalle de boleta:** Tabla intermedia que enlaza la cantidad de medicamentos que posee una boleta, esta posee como atributos: ID, id de la boleta, id de un medicamento y la cantidad (referente al número de unidades a vender).
- **Medicamento:** Corresponde a las sustancias que sirven para el tratamiento o la prevención de enfermedades. Tiene como atributos: ID medicamento (Elemento identificatorio), código de barra, principio activo, nombre comercial, mecanismos de acción, precio, stock, posología (Dosis recomendadas), indicaciones terapéuticas, modo de administración, contraindicaciones y advertencias, y disponible, este nos indica si un medicamento está disponible para venta o no.
- **Detalle receta:** Tabla intermedia que relaciona los medicamentos con la receta. Esta posee como atributos un ID, Id del medicamento, id de la receta, dosis y la cantidad de medicamentos en la receta.

CAPÍTULO 6
DISEÑO

3. Diseño

En este capítulo se presenta el diseño de la aplicación, enfocado en el desarrollo de la misma. Los puntos tratados a continuación abordan el diseño físico de la base de datos, REST, rutas y controladores, interfaz de navegación y especificación de módulos.

3.1 Diseño físico de la base de datos

En la figura 7 se muestra el diseño físico de la base de datos, en él se muestran las tablas, relaciones y atributos.

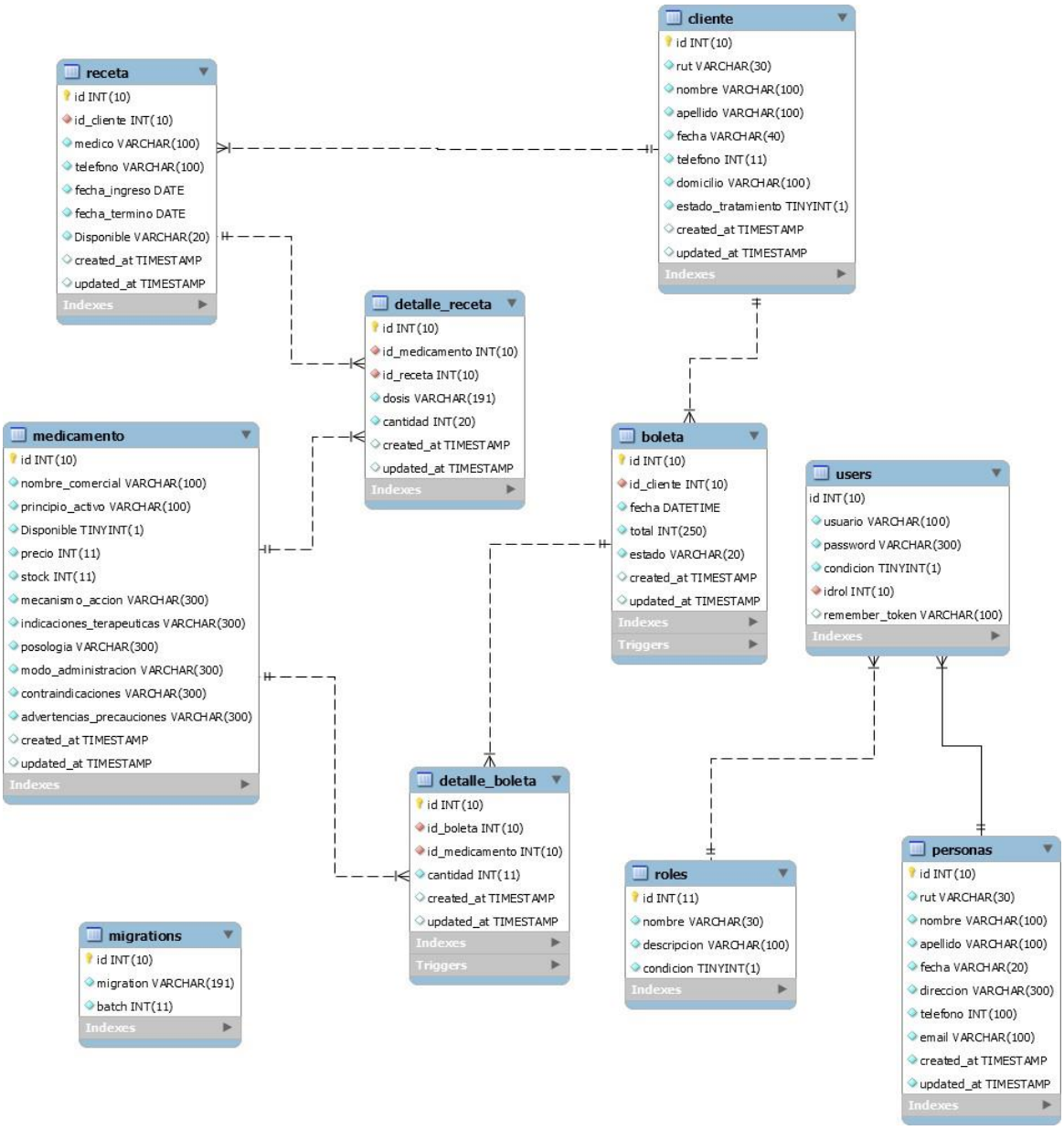

Figura 7: Diseño físico de base de datos.

Como se aprecia en la figura 7 aparecen tablas que no están presentes en el MER de la sección 5.3 esto se debe a que estas tablas tienen relación con la implementación de la solución. A continuación, se describen dichas tablas.

- Personas: Esta tabla contiene los datos de todos los usuarios que interactúan con el sistema.
- Users: Esta tabla almacena los datos de los usuarios que interactúan con el sistema, pero se diferencia de la anterior dado que esta contiene el nombre del usuario y su contraseña.
- Rol: Esta tabla contiene los diferentes roles que existen dentro del sistema.
- Migrations: Esta tabla contiene las migraciones que se han realizado hacia la base de datos.

3.2 Diseño orientado al consumo de servicios

A continuación, se muestra la forma en que opera el sistema, además de los recursos y servicios que deben ser consumidos para cumplir los casos de uso mostrados en la sección 6.1.2.

3.2.1 Recursos y servicios

Para cumplir con los casos de uso del proyecto, en la tabla 12 se han definido los recursos y servicios que el sistema ha de consumir. Antes de definir, se aclara que el término categoría representa a clientes, medicamentos, recetas, boletas y usuarios, dado que usan servicios iguales en cuanto a procesamiento.

ID	Recurso	Servicios	Observaciones
S_Login	Usuario	A partir del Rut y contraseña ingresado se debe ingresar al sistema.	La contraseña esta encriptada con bcrypt. Mensaje de respuesta en caso de datos erróneos.
S_logout	Usuario	Se elige la opción de salir del sistema.	Se vuelve a la vista del login.
S_Cliente	Clientes	Toma todos los datos de los clientes registrados.	Se visualiza todos los datos de los clientes ingresados al sistema.
S_medicamento	Medicamentos	Toma todos los datos de los medicamentos registrados.	Se visualiza todos los datos de los medicamentos ingresados al sistema.
S_receta	Recetas	Toma todos los datos de las recetas registradas.	Se visualiza todos los datos de las recetas ingresadas al sistema.
S_boleta	Boletas	Toma todos los datos de las boletas registradas.	Se visualiza todos los datos de las boletas ingresadas al sistema.
S_user	Usuarios	Toma todos los datos de los usuarios registrados.	Se visualiza todos los datos de los usuarios ingresados al sistema

S_roles	Roles	Toma todos los datos de los roles registrados.	Se visualiza todos los datos de los roles ingresados al sistema
S_registrar	Categoría	Ingresa una nueva categoría (cliente, medicamento, receta, boleta).	Se ingresan los datos en un formulario para el ingreso de una categoría.
S_actualizar	Categoría	Actualiza el dato de una categoría (cliente, medicamento, receta, boleta y usuario).	Se actualiza datos en específico, los que se muestran en un formulario. Se actualiza acorde a la id escogida
S_desactivar	Categoría	Desactiva el dato de una categoría (cliente, medicamento, receta, boleta y usuario).	Se desactiva un registro en específico, para no hacer uso de este dentro del sistema. Se desactiva acorde a la id escogida
S_activar	Categoría	Activa el dato de una categoría (cliente, medicamento, receta, boleta y usuario).	Se activa un registro en específico, para el uso de este nuevamente dentro del sistema. Se activa acorde a la id escogida
S_obtenerCabecera	Categoría	Muestra los detalles específicos de un dato de una categoría (cliente, medicamento, receta, boleta).	Se visualiza los detalles de datos en específico. Se visualizan a acorde a la id escogida.
S_ selectCliente	Muestra el RUT a medida que se escribe.	Muestra el RUT a medida que se va escribiendo.	Se muestra el RUT de un cliente en específico.
S_listarpdf	Cliente, medicamento.	Genera un reporte de PDF de una categoría completa (Cliente, medicamento)	Al presionar el botón reporte se genera automáticamente un reporte en PDF de todos los datos de la categoría.
S_ obtenerDetalles	Receta, boleta.	Muestra los detalles de los medicamentos que posee un registro en una categoría (Receta, boleta).	Se abre una nueva ventana que muestra los detalles de un registro en específico.
S_ selectRol	Rol	Muestra los roles disponibles.	Muestra los roles disponibles a la hora de ingresar un usuario.
S_ pdf/{id}	Venta.	Genera un reporte en PDF de una venta en específico.	Genera un reporte en PDF de una venta en específico, a partir de la id correspondiente de esa venta.

Tabla 12: Recursos y servicios.

3.3 Servicios Web

A continuación, se presenta el uso de REST en el sistema.

Las peticiones de servicios, según la arquitectura REST vista en la sección 2.3.2, nos permite implementar servicios web mediante el protocolo HTTP, dependiendo del tipo de petición se obtienen diferentes tipos de datos, actualizar datos, eliminar, así como implementar validaciones, entre otros servicios.

A continuación, se muestra en las tablas 13 y 14. las peticiones realizadas en el sistema y sus resultados. Estas peticiones corresponden a métodos POST y GET, a su vez se muestra las URL utilizadas para llamar a los servicios, métodos y parámetros usados para el envío de estas peticiones. La descripción del resto de las peticiones se encuentra en el Anexo 3.

ID	S_Cliente
URL	http://127.0.0.1:8000/cliente
Método	GET
Parámetros de la URL	http://127.0.0.1:8000/cliente?page=1&buscar=184513250&criterio=rut
Parámetros de datos	/cliente

Respuesta Exitosa	Codigo:200
	{
	apellido: "navarrete"
	created_at: "2018-11-19 00:00:00"
	domicilio: "chillan"
	estado_tratamiento: 1
	fecha: "1993-07-02"
	id: 1
	nombre: "diego alexis"
	rut: "184513250"
	telefono: 89785634
	updated_at: "2018-11-29 20:11:13"
	}

Respuesta errónea	No hay error solo no se visualizan datos.

Llamada de ejemplo	listarCliente(page, buscar, criterio) {
	let me = this;
	var url ="/cliente?page=" +page +"&buscar=" +buscar +"&criterio=" +criterio;
	axios.get(url).then(function(response) {
	var respuesta = response.data;
	me.arrayCliente = respuesta.clienteController.data;
	me.pagination = respuesta.pagination;
	})
	.catch(function(error) {
	console.log(error);
	});
	},

Tabla 13: Petición de para buscar un cliente.

ID	S_Registrar
URL	http://127.0.0.1:8000/cliente/registrar
Método	POST
Parámetros de la URL	http://127.0.0.1:8000/cliente/registrar
Parámetros de datos	/cliente/registrar

Respuesta Exitosa	Codigo:200
	{
	apellido:"Gonzalez Navarro"
	created_at:"2018-12-14 05:37:04"
	domicilio:"Chillan"
	estado_tratamiento:1
	fecha:"1990-12-03"
	id:10
	nombre:"Ana del carmen"
	rut:"184513257"
	telefono:49658798
	updated_at:"2018-12-14 05:37:04"
	}

Respuesta errónea	El sistema retorna un objeto json con el error.

Llamada de ejemplo	registrarCliente() {
	if (this.validarCliente()) {
	return;
	}
	let me = this;
	axios.post("/cliente/registrar", {
	rut: this.rut,
	nombre: this.nombre,
	apellido: this.apellido,
	fecha: this.fecha,
	telefono: this.telefono,
	domicilio: this.domicilio
	})
	.then(function(response) {
	me.cerrarModal();
	me.listarCliente(1, "", "rut");
	})
	.catch(function(error) {
	console.log(error);
	
	});
	},

Tabla 14: Petición para ingresar un cliente al sistema.

3.4 Controladores y Rutas

A continuación, se presenta los controladores y rutas utilizados en conjunto con los servicios presentados en la sección anterior.

3.4.1 Rutas

Las rutas son una capa muy importante en Laravel, es por ello que el Framework destina un directorio en la carpeta raíz, llamado routes, para ubicar todas las rutas de la aplicación.

Desde la figura 8 a la figura 16 se presenta una lista de las rutas que se utilizan y los diferentes servicios de cada uno de los módulos del sistema.

Rutas clientes

```
Route::get('/cliente', 'clienteController@index');
Route::post('/cliente/registrar', 'clienteController@store');
Route::put('/cliente/actualizar', 'clienteController@update');
Route::put('/cliente/desactivar', 'clienteController@desactivar');
Route::put('/cliente/activar', 'clienteController@activar');
Route::get('/cliente/selectCliente', 'clienteController@selectCliente');
Route::get('/cliente/listaPdf', 'clienteController@listaPdf')->name('cliente_pdf');
```

Figura 8: Rutas para servicios de clientes.

Rutas medicamentos

```
Route::get('/medicamento', 'medicamentoController@index');
Route::post('/medicamento/registrar', 'medicamentoController@store');
Route::put('/medicamento/actualizar', 'medicamentoController@update');
Route::put('/medicamento/desactivar', 'medicamentoController@desactivar');
Route::put('/medicamento/activar', 'medicamentoController@activar');
Route::get('/medicamento/obtenerCabecera', 'medicamentoController@obtenerCabecera');
Route::get('/medicamento/listaPdf', 'medicamentoController@listaPdf')->name('medicamentos_pdf');
```

Figura 9: Rutas para servicios de medicamentos.

Rutas recetas

```
Route::get('/receta', 'recetaController@index');
Route::post('/receta/registrar', 'recetaController@store');
Route::put('/receta/actualizar', 'recetaController@update');
Route::get('/receta/obtenerCabecera', 'recetaController@obtenerCabecera');
Route::get('/receta/obtenerDetalles', 'recetaController@obtenerDetalles');
```

Figura 10: Rutas para servicios de recetas.

Rutas boletas

```
Route::get('/boleta', 'boletaController@index');
Route::post('/boleta/registrar', 'boletaController@store');
Route::put('/boleta/actualizar', 'boletaController@update');
Route::put('/boleta/desactivar', 'boletaController@desactivar');
Route::put('/boleta/activar', 'boletaController@activar');
Route::get('/boleta/obtenerCabecera', 'boletaController@obtenerCabecera');
Route::get('/boleta/obtenerDetalles', 'boletaController@obtenerDetalles');
Route::get('/boleta/pdf/{id}', 'boletaController@pdf')->name('boleta_pdf');
```

Figura 11: Rutas para servicios de boletas.

Rutas usuarios

```
Route::get('/user', 'UserController@index');
Route::post('/user/registrar', 'UserController@store');
Route::put('/user/actualizar', 'UserController@update');
Route::put('/user/desactivar', 'UserController@desactivar');
Route::put('/user/activar', 'UserController@activar');
```

Figura 12: Rutas para servicios de usuarios.

Rutas roles

```
Route::get('/rol', 'RolController@index');
Route::get('/rol/selectRol', 'RolController@selectRol');
```

Figura 13: Rutas para servicios de roles.

Rutas login

```
Route::get('/','Auth\LoginController@showLoginForm');
Route::post('/login', 'Auth\LoginController@login')->name('login');
```

Figura 14: Rutas para servicios de login

Rutas logout

```
Route::post('/logout', 'Auth\LoginController@logout')->name('logout');
```

Figura 15: Rutas de servicios de logout.

Rutas reportes

```
Route::get('/dashboard','DashboardController');
```

Figura 16: Rutas para servicios de reportes.

Como se puede observar en las rutas, además de tener la URL para hacer la petición, indican el controlador, así como el método dentro del controlador (exceptuando la ruta de reportes), todo esto se analizará con más detalle en la próxima sección de controladores.

3.4.2 Controladores

Los Controladores agrupan las peticiones HTTP relacionadas con la manipulación lógica en una clase, y contiene todos los métodos que puede utilizar la clase.

A continuación, en la tabla N°15 se presenta el controlador BoletaController, describiéndose su funcionalidad, los métodos y su propósito en el sistema, la descripción de los demás controladores se encuentra en el Anexo 4.

Controlador	**Funcionalidad**	**Método**	**Objetivo**
BoletaController	Administrar todos los métodos de la clase Boleta.	Index	Mostrar los datos de todas boletas en el sistema.
		store	Actualizar los registros de una boleta determinada.
		ObtenerCabecera	Obtener los detalles de una boleta en específico.
		ObtenerDetalles	Se complementa con ObtenerCabecera y tiene el propósito de mostrar todos los medicamentos que posee una boleta
		Pdf	Genera las boletas de cada venta que se ha realizado en el sistema en formato PDF.
		Desactivar	Cambia el estado de la receta para anular la compra.

Tabla 15: Detalles del controlador de boleta.

3.5 Diseño de la interfaz y navegación

En esta sección se presenta el esquema de interfaz de navegación y ejemplos de la interfaz del sistema.

3.5.1 Esquema de navegación

A continuación, se muestra en la figura 17 el esquema de navegación para el administrador, los demás esquemas se encuentran en el Anexo 5, además se define qué información es relevante en el proceso para determinar los recursos y servicios que debe consumir el sistema.

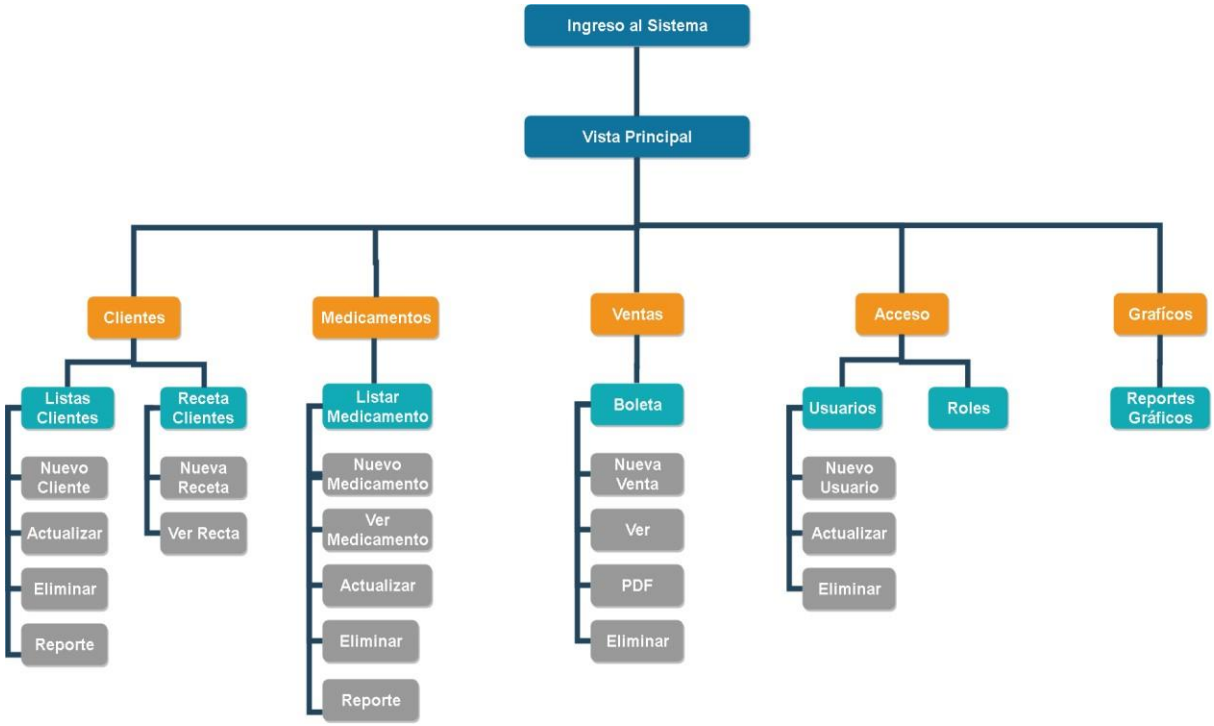

Figura 17: Mapa de navegación de "Administrador"

Cuando se ingresa al sistema, antes que todo, se solicita la autentificación de los usuarios mediante el RUT y contraseña. Una vez que el usuario ha ingresado al sistema, se muestra la pantalla principal, según el Rol al cual pertenezca el usuario autentificado. Además, se muestra en la esquina superior derecha el nombre del usuario, donde se encuentra la opción cerrar sesión de dicho usuario.

El sistema cuenta con un menú lateral en el cual se encuentran todos los módulos al cual el usuario puede acceder, todo esto y como se mencionó anteriormente, es acorde al rol que pertenece dicho usuario.

A continuación, se describen los módulos mencionando las actividades que se realizan en dichos módulos.

Para el módulo cliente, se tiene lo siguiente:

Listar clientes: este módulo cuenta con cuatros actividades.

- Nuevo cliente: Cumple la función de ingresar un nuevo cliente al sistema recopilando todos sus datos.
- Actualizar: Permite actualizar los datos de un cliente existente en el sistema.
- Eliminar: Permite cambiar el estado de un cliente, esto para dar cuenta de si el cliente está acudiendo a la farmacia o no.
- Reporte: Permite generar un reporte en formato PDF con una lista de todos los clientes y el total de clientes registrados en ese momento.

Lista receta: este módulo cuenta con dos actividades:

- Nueva Receta: Cumple la función de ingresar una nueva receta al sistema recopilando todos sus datos.
- Ver receta: Permite ver los detalles de una determinada receta.

Para el módulo medicamento, se tiene siguiente:

Listar medicamentos: Este módulo cuenta con cinco actividades.

- Nuevo Medicamento: Cumple la función de ingresar un nuevo medicamento al sistema recopilando todos sus datos.
- Ver medicamento: Permite ver todos los detalles de un determinado medicamento.
- Actualizar: Permite actualizar los datos de un medicamento existente en el sistema.
- Eliminar: Permite cambiar el estado de un medicamento dejándolo no disponible, independiente de si tiene stock.
- Reporte: Permite generar un reporte en formato PDF que contiene una lista de todos los medicamentos indicando, además, el número total de medicamentos que están registrados en ese momento.

Para el módulo ventas, se tiene lo siguiente:

Boleta: Este módulo cuenta con cuatro actividades.

- Nueva venta: Cumple la función de ingresar una venta al sistema recopilando todos sus datos.
- Ver: Permite ver los detalles de una venta en específico.
- PDF: Permite generar la boleta de una venta en específico.
- Eliminar: Permite anular una venta realizada.

Para el módulo Acceso, se tiene lo siguiente:

Usuarios: Este módulo cuenta con tres actividades.

- Nuevo usuario: Cumple la función de ingresar un nuevo usuario al sistema recopilando todos sus datos.
- Actualizar: Permite actualizar los datos de un determinado usuario.
- Eliminar: Permite eliminar un usuario del sistema, impidiendo que este pueda ingresar al sistema.

Roles: Este módulo solo cumple la función de mostrar los roles disponibles en el sistema.

Para el módulo Gráficos, se tiene lo siguiente:

Reportes Gráficos: Este módulo solo cuenta con una actividad que consiste en mostrar los reportes gráficos de las ventas ingresadas por mes y los clientes ingresados por mes.

3.5.2 Diseño de la interfaz

A continuación, desde la figura 18 a la figura 23, se muestra el diseño de la interfaz del sistema. En particular, se muestra la disposición de menús, botones, logo, formularios, gráficos y otros aspectos de la interfaz del sistema. El detalle de las demás opciones que se muestran en pantalla se encuentra en el Anexo 5.

Ingreso al sistema

La figura 18 muestra la vista de acceso al sistema.

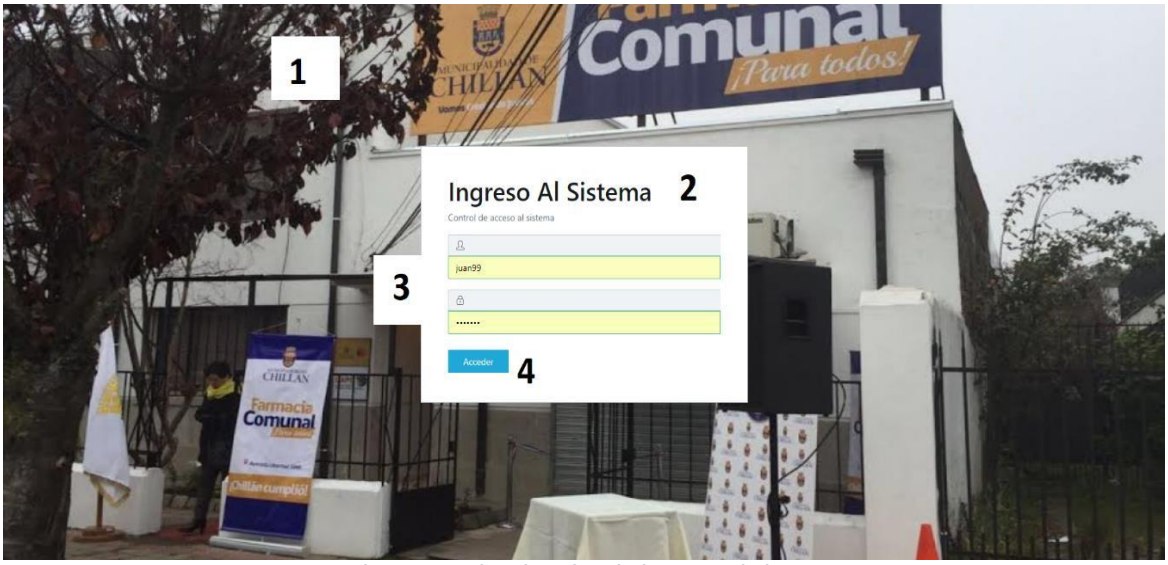

Figura 18: Vista interfaz de ingreso al sistema.

La tabla 16 muestra la descripción de cada una de las cuatro áreas de la vista de acceso al sistema.

N.º Área	Representación
1	Background farmacia comunal de Chillán.
2	Título de ingreso al sistema.
3	Formulario de ingreso.
4	Botón Acceder.

Tabla 16: Descripción vista interfaz de acceso al sistema

Pantalla Principal / Ventas

La figura 19 muestra la vista principal, la cual corresponde a las ventas, esta interfaz es utilizada para ver las ventas realizadas y realizar una nueva venta. El detalle de las demás opciones que se muestran en pantalla se encuentra en el Anexo 5.

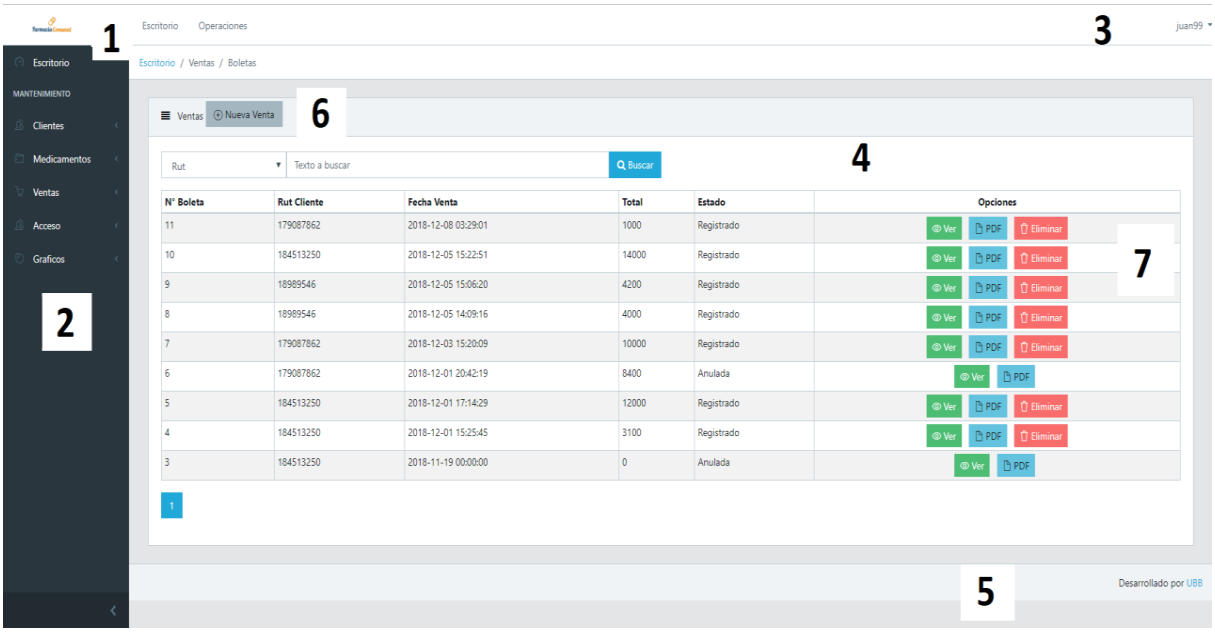

Figura 19: Vista Interfaz de ventas.

La tabla 17 muestra la descripción de cada una de las siete áreas de la vista de ventas de la figura 19.

N.º Área	Representación
1	Logo de la farmacia.
2	Menú Principal.
3	Barra Superior y nombre de perfil.

4	Contenido de la página y buscador.
5	Pie de página.
6	Botón realizar nueva venta.
7	Botón de ver la venta realizada, generar pdf (boleta) y eliminar.

Tabla 17: Diseño de interfaz "Pantalla principal/ Ventas".

Clientes

La figura 20 muestra la Interfaz que permite ver los clientes registrados en el sistema, y el ingreso de un nuevo cliente. El resto de las opciones que se muestran en pantalla se describen en el Anexo 5.

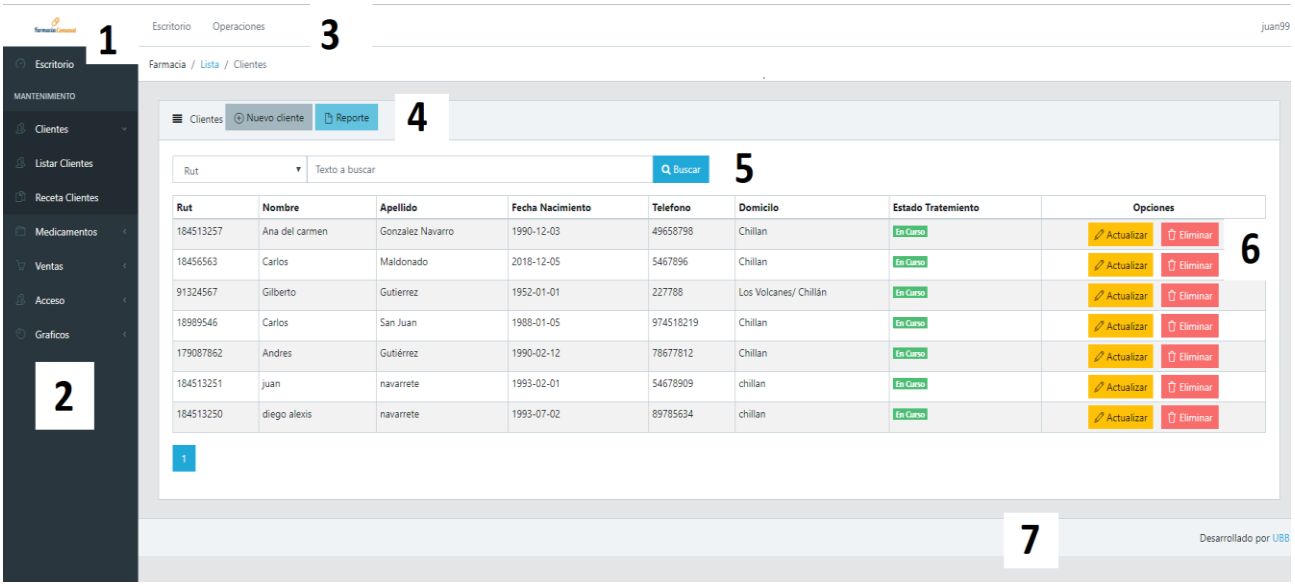

Figura 20: vista interfaz de cliente.

La tabla 18 muestra la descripción de cada una de las siete áreas de la vista de cliente de la figura 20.

N.º Área	**Representación**
1	Logo de la farmacia.
2	Menú Principal.
3	Barra Superior y nombre de perfil.
4	Botón ingresar un nuevo cliente/ Botón generar reporte de clientes.
5	Contenido de la página y buscador.
6	Botón actualizar información del cliente/ Botón eliminar cliente.
7	Pie de página.

Tabla 18: Diseño de interfaz "Clientes".

Medicamentos

La figura 21 muestra la interfaz de medicamentos que permite ver los medicamentos registrados y agregar nuevos medicamentos en el sistema. El detalle de las demás opciones que se muestran en pantalla se encuentra en el Anexo 5.

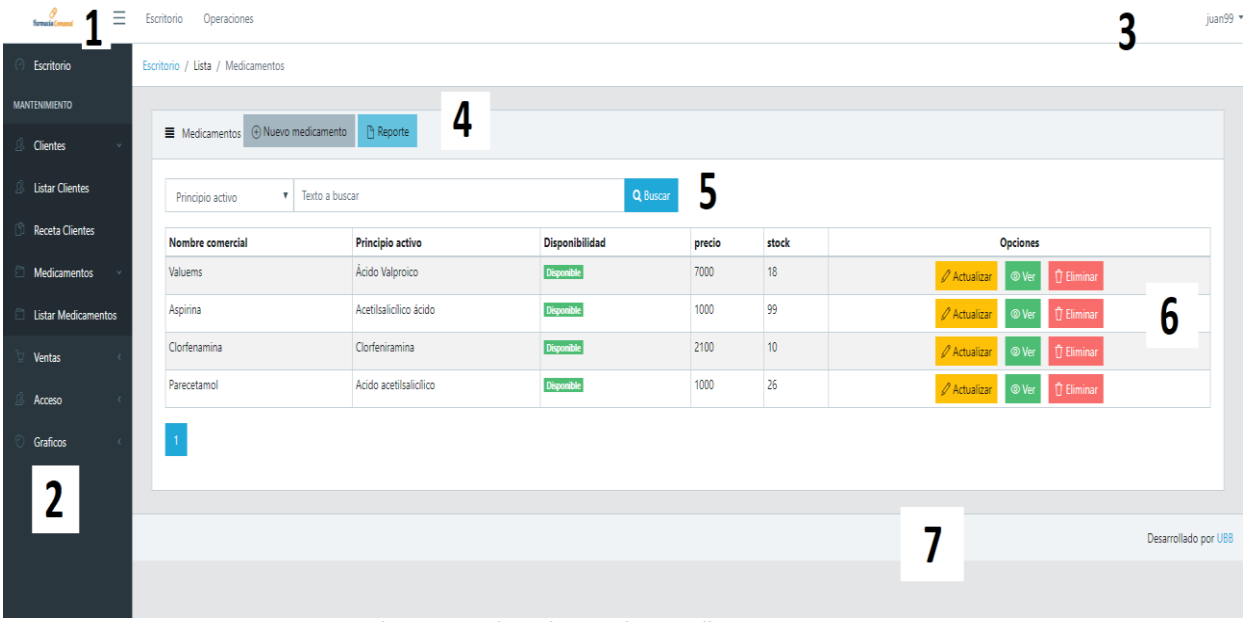

Figura 21: Vista de Interfaz medicamentos.

La tabla 19 muestra la descripción de cada una de las siete áreas de la vista de medicamentos de la figura 21.

N.º Área	Representación
1	Logo de la farmacia.
2	Menú Principal.
3	Barra Superior y nombre de perfil.
4	Botón ingresar un nuevo medicamento/ Botón generar reporte de medicamentos.
5	Contenido de la página y buscador.
6	Botón actualizar información del medicamento/ Botón eliminar medicamento/ Botón ver información detallada del medicamento.
7	Pie de página.

Tabla 19: Diseño de interfaz "Medicamentos"

Recetas

La figura 22 muestra la interfaz de recetas, esta permite listar las recetas registradas en el sistema, agregar una nueva receta y ver en detalle las recetas que están en el sistema. El resto de las opciones mostradas en pantalla se encuentran en el Anexo 5.

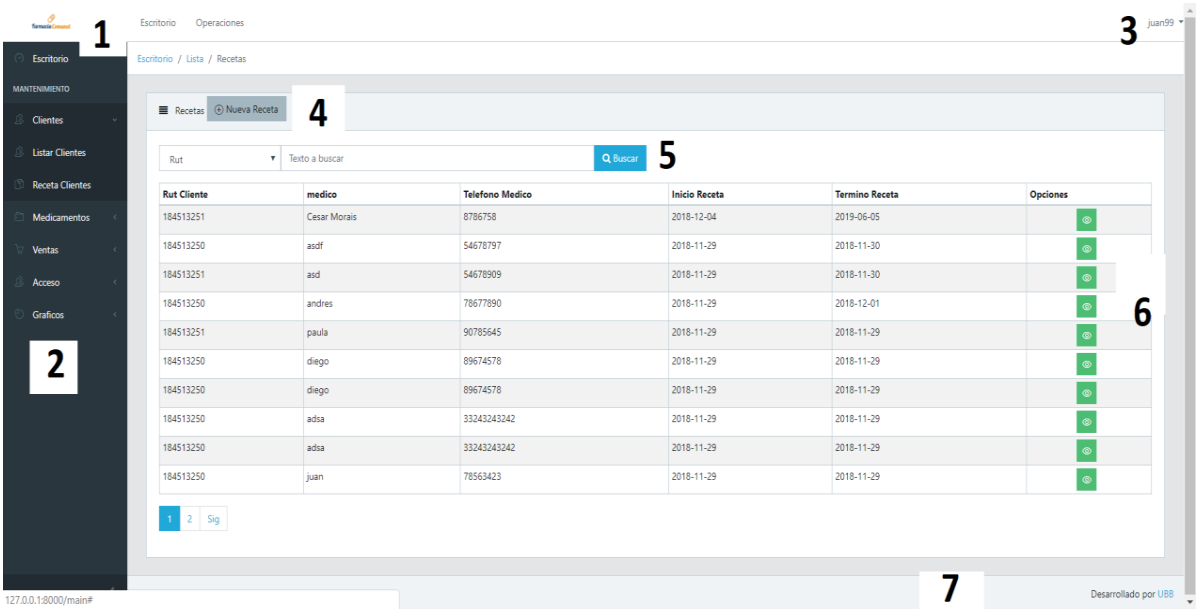

Figura 22: Vista interfaz recetas.

La tabla 20 muestra la descripción de cada una de las siete áreas de la vista de recetas de la figura 22.

N.º Área	Representación
1	Logo de la farmacia.
2	Menú Principal.
3	Barra Superior y nombre de perfil.
4	Botón ingresar una nueva receta.
5	Contenido de la página y buscador.
6	Botón ver información detallada de la receta.
7	Pie de página.

Tabla 20: Diseño de interfaz "Receta"

Gráficos

La figura 23 muestra la interfaz de los gráficos generados por la aplicación. Estos gráficos presentan los ingresos de clientes y ventas realizados por mes.

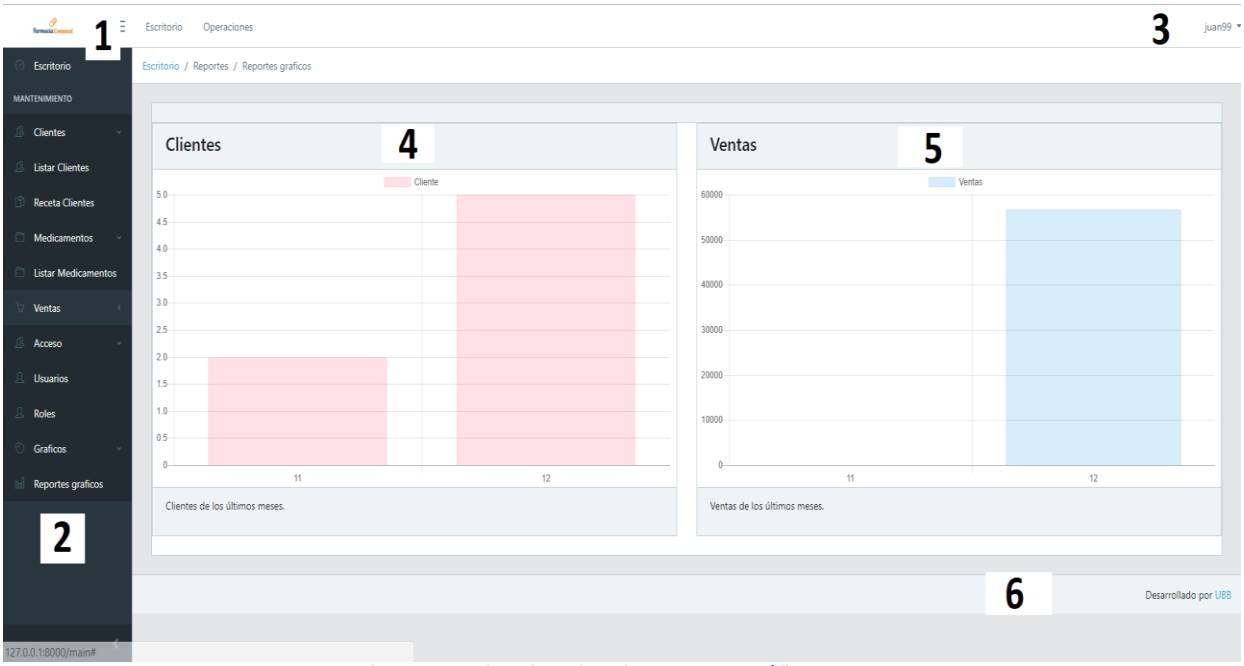

Figura 23: Vista interfaz de reportes gráficos.

La tabla 21 muestra la descripción de cada una de las seis áreas de la vista de reportes gráficos de la figura 23.

N.º Área	Representación
1	Logo de la farmacia.
2	Menú Principal.
3	Barra Superior y nombre de perfil
4	Gráfico de clientes ingresados en un cierto mes.
5	Gráfico con las ventas realizadas en un cierto mes.
6	Pie de página.

Tabla 21: Diseño interfaz "Gráficos"

CAPÍTULO 7
Pruebas

4. Pruebas

En este capítulo se presentan las pruebas del sistema, los elementos de prueba, la metodología usada para realizar las pruebas funcionales y no funcionales.

4.1 Escenarios de prueba

Se realizan pruebas a las principales funcionalidades de cada uno de los módulos del sistema.

A continuación, se listan los escenarios definidos para las pruebas:

1. Inicio de sesión
2. venta
3. cliente
4. receta
5. medicamento
6. usuario

7.1.2 Metodología de pruebas

Para realizar las pruebas se utiliza caja negra, que se describe a continuación.

Las Pruebas de Caja Negra, es una técnica de pruebas de software en la cual la funcionalidad se verifica sin tomar en cuenta la estructura interna de código, detalles de implementación o escenarios de ejecución internos en el software [13].

En las pruebas de caja negra, el enfoque está en las entradas y salidas del sistema, no hay necesidad de conocer la estructura interna del programa de software. Para obtener el detalle de cuáles deben ser esas entradas y salidas, se toma como base los requerimientos de software y especificaciones funcionales [13].

7.2 Pruebas funcionales

A continuación, en la tabla 22 y 23 se muestra las pruebas funcionales para el escenario de prueba inicio de sesión y el ingreso de una venta, respectivamente, las demás pruebas se encuentran en el Anexo 6.

Escenario iniciar sesión:
Fecha de prueba: 28/11/2018

ID	Objetivo	Contexto	Entrada	Salida	Evento	Resultado	Evaluación
P01	Determinar si un usuario puede entrar al sistema.	RUT y contraseña están en la BD.	RUT y contraseña	Se ingresa al sistema.	Al presionar el botón acceder.	Se muestra la pantalla principal.	Éxito.
P02	Determinar si un usuario puede entrar al sistema con datos erróneos	RUT y contraseña están en la BD.	RUT y contraseña incorrectas.	Mensaje de que los datos son inválidos	Al presionar el botón acceder.	No se puede ingresar al sistema.	Éxito.

Tabla 22: Prueba escenario inicio de sesión.

Escenario de venta:
Fecha de prueba: 28/11/2018

ID	Objetivo	Contexto	Entrada	Salida	Evento	Resultado	Evaluación
P03	Ingresar una venta con selección de los medicamentos	RUT y datos de medicamentos están la BD.	RUT y medicamentos.	Se ingresa la venta al sistema.	Al presionar el botón registrar venta.	Se muestra la pantalla de ventas con el nuevo registro.	Éxito.
P04	Ingresar una venta con datos faltantes.	RUT y datos de medicamentos están la BD.	RUT y medicamentos.	Mensaje de que faltan medicamentos.	Al presionar el botón registrar venta	No se puede registrar la venta.	Éxito.

Tabla 23: Prueba de escenario de venta.

Las pruebas anteriores fueron realizadas por los desarrolladores del software.

A continuación, se muestran las pruebas realizadas en conjunto con el profesor guía del proyecto, el detalle del resto de las pruebas de esta sección de trabajo está en el Anexo 6.

Fecha de prueba: 26/12/2018

ID	Objetivo	Contexto	Entrada	Salida	Evento	Resultado	Evaluación
P25	Ingresar una venta con selección de los medicamentos	RUT y datos de medicamentos están la BD.	RUT y medicamentos.	Se ingresa la venta al sistema.	Al presionar el botón registrar venta.	Se muestra la pantalla de ventas con el nuevo registro.	Fallo en la validación del formato del Rut del cliente.
P26	Ingresar una venta a partir de la receta del cliente.	RUT y datos de medicamentos están la BD.	RUT y medicamentos.	ninguno.	Al presionar el botón registrar venta	No se puede registrar la venta, de esta forma.	Fallo.

Tabla 24: Pruebas generales del sistema.

Si bien los fallos corresponden a la falta de validaciones, como son el formato del RUT, esto ocurre por no considerar estas situaciones al momento del desarrollo del software.

7.3 Pruebas no funcionales

A continuación, se presentan las pruebas no funcionales que determinan el cumplimiento de los requisitos no funcionales, presentado en la sección 3.1.2.

7.3.1 Usabilidad

Después de realizar pruebas de usabilidad del sistema en conjunto con el profesor guía, se encontraron los siguientes problemas:

- Errores de escritura en algunas tablas.
- Icono no adecuado para un botón de eliminar un medicamento seleccionado al momento de realizar una compra.
- Algunas columnas que se deben eliminar.
- Menú lateral un poco confuso.
- Falta de validaciones en formularios, como es la validación del Rut y fecha de nacimiento.

Estos problemas en su mayoría fueron corregidos para entregar un producto más limpio en cuanto a interfaz y usabilidad para el usuario, pero quedaron pendiente validaciones de formularios, que por cuestiones de desarrollo no se pudo resolver del todo, quedando estos como un trabajo futuro para tener en cuenta.

Si bien aún quedan detalles de la interfaz por mejorar, se logra tener una interfaz limpia e intuitiva para el usuario, cumpliendo con este requisito no funcional.

7.3.2 Seguridad

En este aspecto el sistema cuenta con una serie de medidas de seguridad que lo hacen bastante seguro para su uso. A continuación, se describen todas las medidas de seguridad con las que cuenta el sistema.

Encriptación de la clave del usuario: Cuando se registra a un nuevo usuario en el sistema, su clave de acceso es guardada de forma encriptada en la base de datos. Esto se hace mediante Bcrypt que proporciona Laravel, a continuación, se detalla que es Bcrypt:

Bcrypt es una función de hashing de contraseñas diseñado por Niels Provos y David Maxieres, basado en el cifrado de Blowfish. Lleva incorporado un valor llamado salt, que es un fragmento aleatorio que se usará para generar el hash asociado a la contraseña, y se guardará junto con ella en la base de datos. Así se evita que dos contraseñas sean iguales, esto se debe al uso de un hash por parte de Bcrypt y así se evita el ataque por fuerza bruta [14].

El poder de cómputo requerido para generar los hashes con bcrypt es altísimo comparado con MD5 y SHa1. Además, la cantidad de rondas que corre el algoritmo es adaptable como un parámetro de la función [15].

Protección contra ataque CSRF: Laravel facilita la protección de sus frentes a ataques de falsificación de solicitudes entre sitios (CSRF). Las falsificaciones de solicitud entre sitios son un tipo de explotación maliciosa en la que se realizan comandos no autorizados en nombre de un usuario autenticado [16].

Solo se permite peticiones http para el acceso a los controladores: Agregando la línea if (!$request->ajax()) a cada una de las funciones de nuestros controladores nos aseguramos que solo se acepten peticiones Ajax [16].

7.3.3 Eficiencia

En este punto, se mide el tiempo que tarde el sistema en mostrar los datos en pantalla para las diferentes peticiones realizadas. El sistema presenta un adecuado tiempo de respuestas para cada una de las peticiones, no superando los 3 segundos. Sin embargo, se debe tener en cuenta que estos tiempos pueden variar, por factores como la velocidad de conexión a Internet o el servidor en donde se aloja el sistema. No se toma en cuenta el volumen de peticiones al servidor, dado que este sistema no va ser usado por más de 10 personas simultáneamente.

7.4 Conclusiones de pruebas

Como se apreció en las pruebas la gran mayoría de estas cumplieron con sus objetivos, sin embrago quedan elementos por mejorar, como son la usabilidad del sistema. En este punto quedan pendientes algunas validaciones y mejorar el menú lateral.

En cuanto a la seguridad, el sistema cuenta con un adecuado nivel de seguridad, lo que hace que sea muy confiable para los usuarios que interactúan con él.

Las pruebas realizadas permiten una mejora notable en la entrega final del sistema, ya que entregan información importante sobre qué se está realizando bien y qué es posible mejorar, para así asegurar un sistema de calidad y que cumple con los objetivos impuestos inicialmente.

CAPÍTULO 8
CONCLUSIONES

Conclusiones

El desarrollo iterativo e incremental fue una buena elección al momento de escoger la metodología de desarrollo, puesto que, nos guio apropiadamente para cumplir con las tareas que abarcó el proyecto de software, desde la recopilación de los requerimientos, hasta las pruebas finales del software.

Con tiempo reducido, se logró desarrollar el proyecto en un plazo razonable y cumpliendo con todos los requisitos y preferencias de la farmacia comunal. Además, no se retrasó el proyecto al utilizar esta metodología ya que se adapta a los requisitos cambiantes.

En el caso de la arquitectura usada, el Modelo-Vista-Controlador fue una excelente opción ya que, divide la persistencia de los datos, la lógica del sistema y la interfaz que se le presenta al usuario.

Las pruebas realizadas al sistema arrojaron resultados positivos, sin embargo, quedan elementos por mejorar, así como realizar pruebas exhaustivas en cuanto a rendimiento de software y su seguridad. Todo esto ayudará a generar un producto más funcional y confiable para los usuarios.

Considerando lo anterior, el desarrollo iterativo e incremental nos ayuda a mitigar en fases tempranas riesgos que pueden afectar el desarrollo armónico del proyecto, facilita a la mejora y perfeccionamiento del producto, lo que da como resultado un producto más sólido y, sin duda debería ser tratada en proyectos de este tipo.

Como equipo de trabajo, esta ha sido una experiencia bastante enriquecedora, puesto que se tenía conocimiento en el desarrollo en PHP, pero no así usando el framework Laravel y un desconocimiento total en la implementación de servicios web en conjunto con MVC. Por otra parte, el desarrollo de este proyecto nos ayudó a comprender un poco mejor cómo operan las empresas en el mercado actual y sus necesidades.

Finalmente, se concluye que el proyecto fue realizado de buena manera, cumpliendo con los objetivos definidos y las expectativas personales.

Trabajos Futuros

Considerando trabajos futuros, se pueden agregar otras funcionalidades que incorporen más actividades de la farmacia comunal de Chillán. A continuación, se exponen las siguientes funcionalidades a elaborar en un futuro:

- Incluir un módulo de inventario que ayude a generar órdenes de compra, verificar estado de los pedidos y tener registro de los proveedores.

- Otro aspecto a mejorar a futuro es la interfaz, para que esta sea más intuitiva para los usuarios.

- Mejorar las validaciones del sistema para que solo se puedan ingresar datos correctos.

- Adaptar el formato de la boleta que se le entrega al cliente, de forma de que esté acorde a las exigencias legales impuestas por el SII.

- Mejorar los gráficos para que estos muestren datos por año.

- Finalmente, añadir alertas cuando exista bajo stock disponible de un medicamento en particular (Stock crítico), ya que ayudaría a prevenir el desabastecimiento por descuido o falta de atención. Como complemento de la alerta sería interesante enviar automáticamente un email al Farmacéutico/Bodeguero/Administrador/Vendedor con información del stock del insumo.

Bibliografía

[1] Anónimo. Minsal.cl. [En Línea] [Citado el: 2 de Octubre de 2018.] https://www.minsal.cl/wp-content/uploads/2015/09/Recomendaciones-Farmacias-Populares.pdf

[2] Chávez, Jorge. La Discusión. [En Línea] [Citado el: 13 de Octubre de 2018.]http://www.ladiscusion.cl/detalle/13285/32-mil-f%C3%A1rmacos-ha-entregado-la-farmacia-popular-de-Chill%C3%A1n?fbclid=IwAR2rzTaydjeBgF7n9WKMwOBYSZbUddbWG3tLlW3ZRytk zDtmav0Ojl0co#sthash.vmAheh7O.JF2J3SSt.dpbs

[3] Anónimo. [En Línea] [Citado el: 15 de Octubre de 2018.] https://code.visualstudio.com/docs

[4] Anónimo. [En Línea] [Citado el: 15 de Octubre de 2018.] https://mariadb.org/learn/

[5] Anónimo. [En Línea] [Citado el: 15 de Octubre de 2018.] https://es.wikipedia.org/wiki/Laravel

[6] Anónimo. [En Línea] [Citado el: 15 de Octubre de 2018.] https://en.wikipedia.org/wiki/HTML5

[7] Anónimo. [En Línea] [Citado el: 15 de Octubre de 2018.] https://jquery.com/

[8] Robles, Victor. Victor Web. [En Línea] [Citado el: 24 de Octubre de 2018.] https://victorroblesweb.es/2013/11/18/tutorial-mvc-en-php-nativo/

[9] Ribas, Ester. Iedschool. [En Línea] [Citado el: 24 de Octubre de 2018.] https://www.iebschool.com/blog/que-es-api-rest-integrar-negocio-business-tech/

[10] VueJS. VueJS Introduction. [En Línea] [Citado el: 24 de Octubre de 2018.] https://vuejs.org/v2/guide/

[11] Anónimo, Hackermoon. [En Línea] [Citado el: 25 de Octubre de 2018.] https://hackernoon.com/what-is-vue-js-and-what-are-its-advantages-4071b7c7993d

[12] Dongil Sánchez, José Antonio. Genbeta. [En Línea] [Citado el 25 de Octubre de 2018.] https://www.genbeta.com/desarrollo/por-que-elegir-vuejs-5-razones-para-considerarlo-nuestro-proximo-framework-de-referencia

[13] Terrera, Gustavo. Testingbaires. [En Línea] [Citado el 7 de Diciembre de 2018.] https://testingbaires.com/pruebas-caja-negra-enfoque-practico/

[14] Vicente, David. Solidgeargroup. [En Línea] [Citado el 7 de Diciembre de 2018.] https://solidgeargroup.com/password-nodejs-mongodb-bcrypt?lang=es

[15] Sánchez, Orlando. Medium web. [En Línea] [Citado el 8 de Diciembre de 2018.] https://medium.com/universo-mutante/si-guardas-contrase%C3%B1as-as%C3%AD-pones-en-peligro-a-tus-usuarios-e567cc2e1ec5

[16] Laravel. CSRF Protection. [En Línea] [Citado el 8 de Diciembre de 2018.] https://laravel.com/docs/5.6/csrf

ANEXOS

Anexo 1. Diagrama De Casos de uso

Farmacéutico

Figura 24: Diagrama de caso de uso farmacéutico

Vendedor

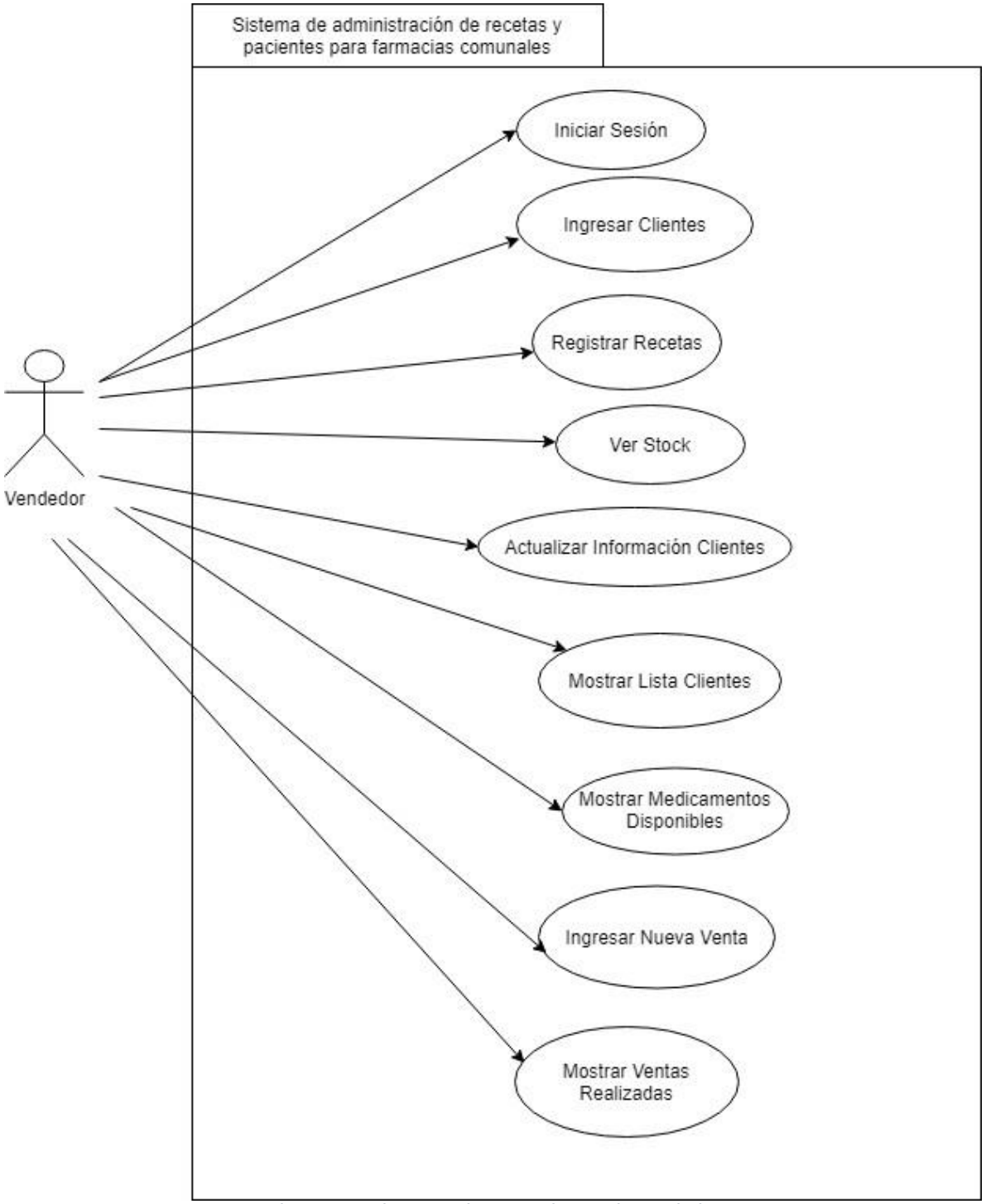

Figura 25.. Diagrama de casos de uso de vendedor

Administrador

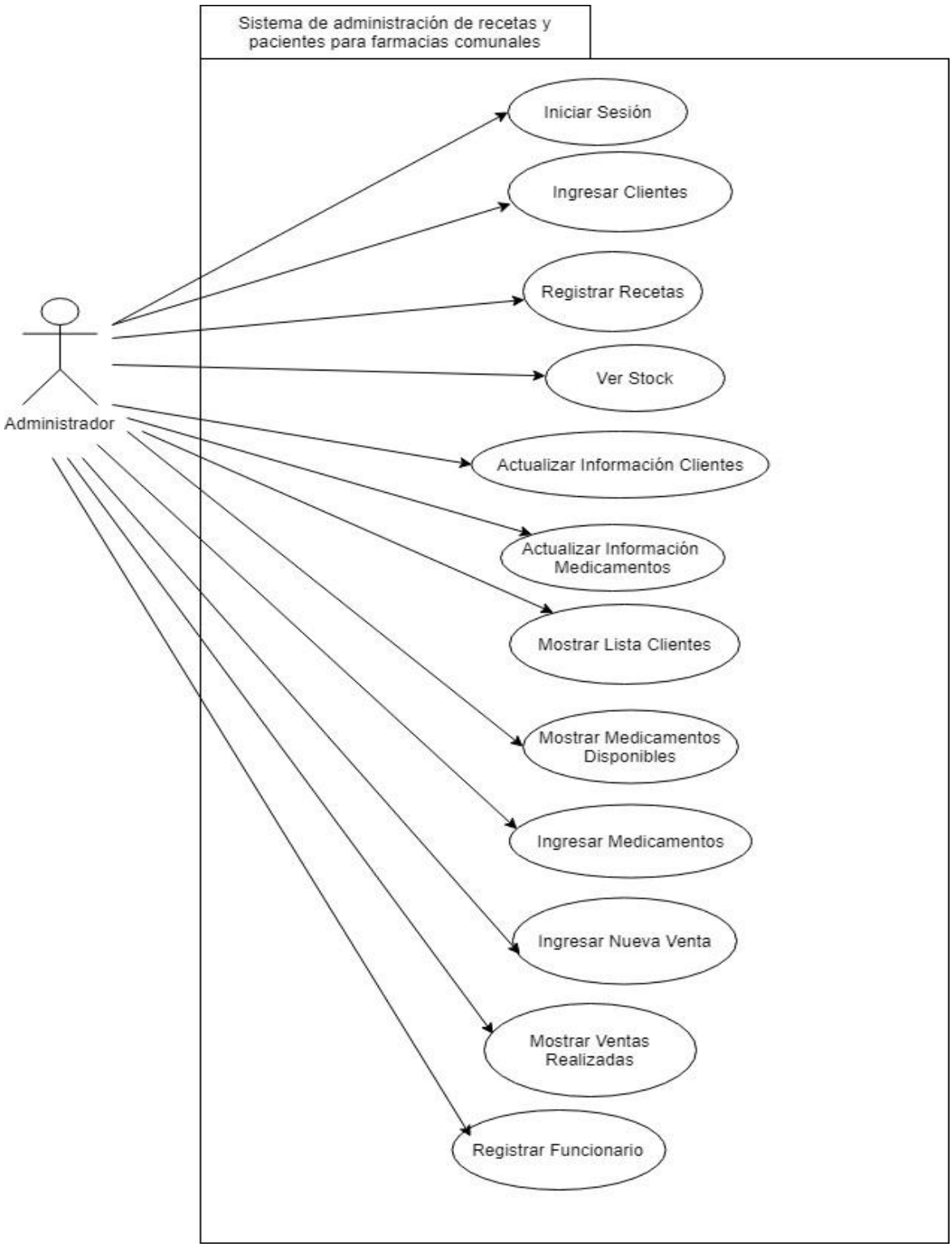

Figura 26. Diagrama de casos de uso administrador.

Bodeguero

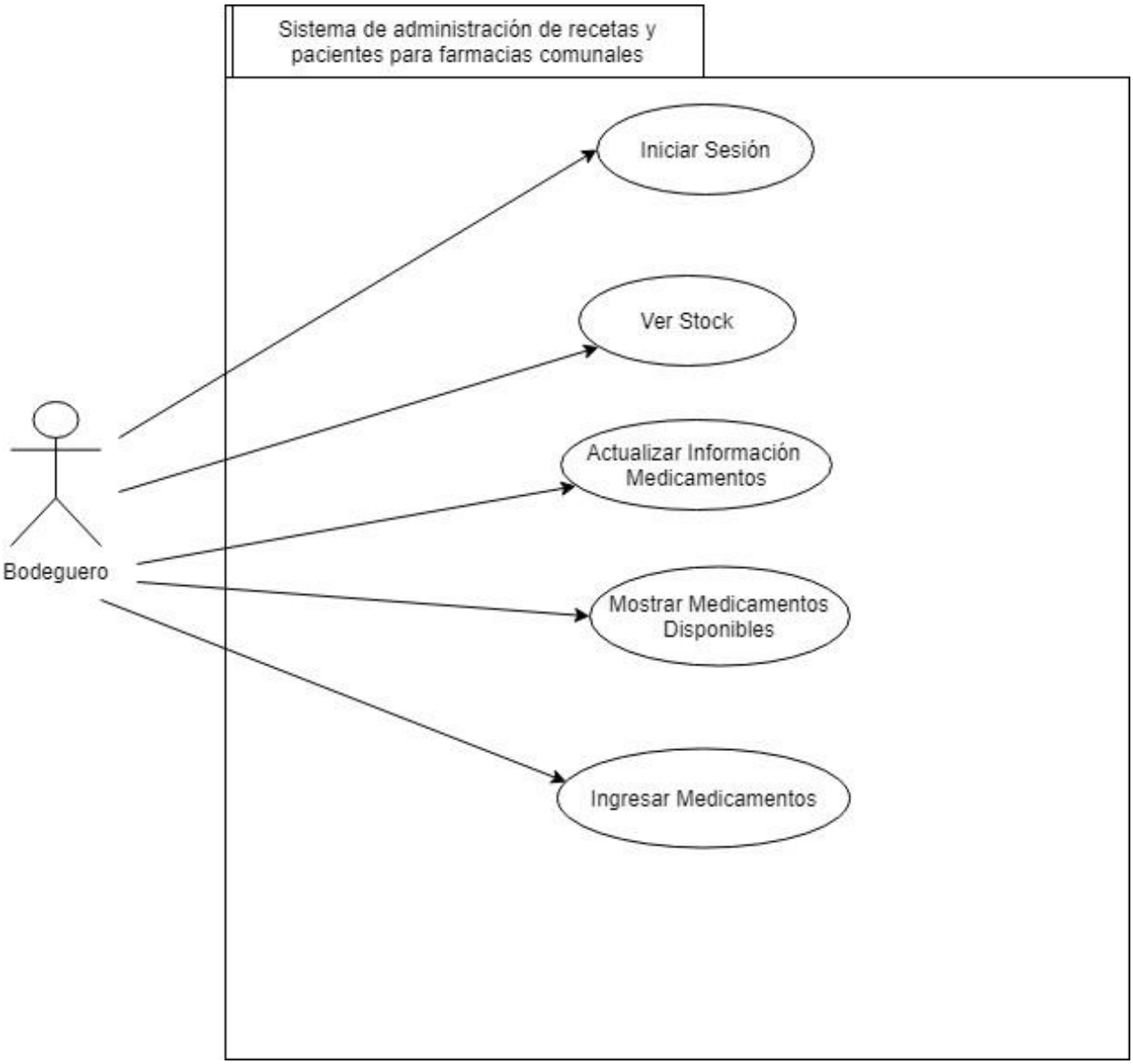

Figura 27. Diagrama de caso de uso bodeguero.

Anexo 2. Especificación de casos de uso

Caso de uso: Iniciar Sesión
ID: CU_1
Breve Descripción: Permite al Administrador/Farmacéutico/Vendedor /Bodeguero iniciar sesión en el sistema de farmacias.
Actores Principales: Administrador/Farmacéutico/Vendedor /Bodeguero
Actores Secundarios: Ninguno
Precondición: El Administrador/Farmacéutico/Vendedor/Bodeguero debe existir en la base de datos.
Flujo Principal: 1. El caso de uso comienza cuando el Administrador/Farmacéutico/Vendedor /Bodeguero se encuentra en la página de login para iniciar sesión. 2. El sistema presenta formulario para el ingreso 3. El sistema verifica que los datos (Nombre, Contraseña) sean correctos. 4. Se hace ingreso a la página.
Postcondición: Una vez ya ingresado el funcionario, el sistema mostrará los módulos correspondientes a la función que desempeña cada trabajador.
Flujos Alternativos: 3a.1 El sistema detecta errores en los datos ingresados. 3a.2 El sistema muestra un mensaje de error. 3a.3 El sistema vuelve al paso 2.

Tabla 25: CU inicio de sesión.

Caso de uso: Registrar Funcionario
ID: CU_12 Administrador
Breve Descripción: Permite al administrador registrar un nuevo funcionario en el sistema.
Actores Principales: Administrador
Actores Secundarios: Ninguno
Precondición: Administrador debe haber iniciado sesión.
Flujo Principal: 1. El caso de uso comienza cuando el administrador escoge la opción de registrar un nuevo funcionario en el sistema. 2. El sistema presenta formulario para el ingreso 3. El sistema verifica que los datos (Nombre, Contraseña) del nuevo funcionario ingresado son correctos y que no se encuentra registrado en el sistema 4. El sistema registra el nuevo funcionario.
Postcondición: El sistema aumenta con un nuevo funcionario registrado.
Flujos Alternativos: 3a.1 El sistema detecta errores en los datos ingresados. 3a.2 El sistema muestra un mensaje de error. 3a.3 El sistema vuelve al paso 2. 3b.1 El sistema detecta que el funcionario fue ingresado. 3b.2 El sistema mostrará un mensaje avisando que el funcionario ya se encuentra registrado. 3b.3 El sistema vuelve al paso 2.

Tabla 26. CU registrar funcionario.

Caso de uso: Ver Stock
ID: CU_4
Breve Descripción: Permite ver el stock de un medicamento.
Actores Principales: Administrador/Farmacéutico/Vendedor /Bodeguero
Actores Secundarios: Ninguno
Precondición: 1. El Administrador/Farmacéutico/Vendedor /Bodeguero ha ingresado al sistema.
Flujo Principal: 1. El caso de uso comienza cuando el técnico o farmacéutico elige la opción de ver stock. 2. El sistema mostrará un listado con la cantidad restante y nombre de los medicamentos.
Postcondición: Ninguna.
Flujos Alternativos: 2a.1 El sistema no posee ningún medicamento registrado 2a.2 El sistema no mostrará ningún dato relacionado con los medicamentos ni las opciones tales como editar, eliminar. 2b.1 No existe medicamento al buscar el nombre de el en el buscador. 2b.2 El sistema no mostrará ningún dato relacionado con ese medicamento ni las opciones tales como editar, eliminar.

Tabla 27: CU ver stock.

Caso de uso: Actualizar Información Clientes
ID: CU_5
Breve Descripción: Permite al Administrador/Farmacéutico/Técnico actualizar la información de los clientes.
Actores Principales: Administrador/Farmacéutico/Vendedor
Actores Secundarios: Ninguno
Precondición: El Administrador/Farmacéutico/Técnico ha ingresado al sistema.
Flujo Principal: 1. El caso de uso comienza cuando el Administrador/Farmacéutico/Vendedor elige la opción de actualizar información paciente de la tabla de clientes. 2. El sistema presenta formulario para actualizar la información del cliente solicitando los datos de este. 3. El sistema verifica que los datos (Rut, Nombre, Apellido, Fecha nacimiento, Domicilio, Teléfono) sean correctos. 4. El sistema actualiza la información del cliente.
Postcondición: Se han actualizado los datos de un cliente en el sistema.
Flujos Alternativos: 3a.1 El sistema detecta errores en los datos ingresados. 3a.2 El sistema muestra un mensaje de error. 3a.3 El sistema vuelve al paso 2.

Tabla 28: CU actualizar cliente.

Caso de uso: Actualizar Información Medicamentos
ID: CU_6
Breve Descripción: Permite al Administrador/Farmacéutico/Bodeguero actualizar la información de los medicamentos.
Actores Principales: Administrador/Farmacéutico/Bodeguero
Actores Secundarios: Ninguno
Precondición: El Administrador/Farmacéutico/Bodeguero ha ingresado al sistema.
Flujo Principal: 1. El caso de uso comienza cuando el Administrador/Farmacéutico/Bodeguero elige la opción de actualizar información medicamentos. 2. El sistema presenta formulario para actualizar la información del medicamento. 3. El sistema verifica que los datos (Principio activo, Nombre comercial, Mecanismos de acción, Precio, Stock, Posología, Indicaciones Terapéuticas, Modo de administración, Contraindicaciones, Advertencias y Disponibilidad) sean correctos. 4. El sistema actualiza información de los medicamentos.
Postcondición: Se han actualizado los datos de un medicamento en el sistema.
Flujos Alternativos: 3a.1 El sistema detecta errores en los datos ingresados. 3a.2 El sistema muestra un mensaje de error. 3a.3 El sistema vuelve al paso 2.

Tabla 29: CU actualizar medicamento.

Caso de uso: Mostrar Medicamentos Disponibles
ID: CU_8
Breve Descripción: Permite al Administrador/Farmacéutico/Vendedor /Bodeguero ver los medicamentos disponibles en el sistema.
Actores Principales: Administrador/Farmacéutico/Vendedor /Bodeguero
Actores Secundarios: Ninguno.
Precondición: 1. El Administrador/Farmacéutico/Vendedor /Bodeguero ha ingresado al sistema. 2. En la base de datos del sistema deben existir medicamentos registrados.
Flujo Principal: 1. El caso de uso comienza cuando el Administrador/Farmacéutico/Técnico/Bodeguero elige la opción Mostrar medicamentos disponibles. 2. El sistema mostrará el listado de los medicamentos que posee el sistema.
Postcondición: Ninguna.
Flujos Alternativos: 2a.1 El sistema no posee ningún medicamento registrado 2a.2 El sistema no mostrará ningún dato relacionado con los medicamentos ni las opciones tales como editar, eliminar. 2b.1 No existe medicamento al buscar el nombre de el en el buscador. 2b.2 El sistema no mostrará ningún dato relacionado con ese medicamento ni las opciones tales como editar, eliminar.

Tabla 30: CU listar medicamento.

Caso de uso: Ingresar Medicamentos
ID: CU_9
Breve Descripción: Permite al Administrador/Farmacéutico/ Bodeguero ingresar un nuevo medicamento al sistema.
Actores Principales: Administrador/Farmacéutico/ Bodeguero
Actores Secundarios: Ninguno
Precondición: El Administrador/Farmacéutico/ Bodeguero ha ingresado al sistema.
Flujo Principal: 1. El caso de uso comienza cuando el Administrador/Farmacéutico/Bodeguero escoge ingresar un nuevo medicamento en el sistema. 2. El sistema presenta formulario para el ingreso del medicamento solicitando los datos de este. 3. El sistema verifica que los datos (Principio activo, Nombre comercial, Mecanismos de acción, Precio, Stock, Posología, Indicaciones Terapéuticas, Modo de administración, Contraindicaciones, Advertencias y Disponibilidad) del nuevo medicamento ingresado sean correctos y que se encuentre registrado en el sistema. 4. El sistema registra el nuevo medicamento.
Postcondición: El sistema aumenta con un nuevo medicamento registrado.
Flujos Alternativos: 3a.1 El Sistema detecta errores en los datos ingresados. 3a.2 El sistema muestra mensaje de error. 3.a.3 Vuelve al formulario. 3b.1 El sistema detecta que el medicamento se encuentra registrado. 3b.2 El sistema mostrará un mensaje avisando que el medicamento ya se encuentra registrado. 3b.3 Sistema vuelve al paso 2.

Tabla 31: CU ingresar medicamento.

Caso de uso: Ingresar Nueva Venta
ID: CU_10
Breve Descripción: Permite al Administrador/Farmacéutico/Vendedor realizar una venta.
Actores Principales: Administrador/Farmacéutico/Vendedor
Actores Secundarios: Ninguno
Precondición: 1. Administrador/Farmacéutico/Vendedor debe haber iniciado sesión. 2. En la base de datos del sistema deben existir clientes registrados. 3. En la base de datos del sistema deben existir medicamentos registrados.
Flujo Principal: 1. El caso de uso comienza cuando el Administrador/Farmacéutico/Vendedor elige la opción de ingresar una nueva venta. 2. El sistema presenta formulario para el ingreso de una nueva venta. 3. El sistema verifica que los datos (Cantidad de medicamentos) de la nueva venta ingresada son correctos. 4. El sistema registra la venta.
Postcondición: El sistema aumenta con una nueva venta registrada.
Flujos Alternativos:

3a.1 El sistema detecta errores en los datos ingresados. 3a.2 El sistema muestra un mensaje de error. 3a.3 El sistema vuelve al paso 2. 3b.1 El sistema detecta que no hay ningún medicamento ingresado en el sistema 3b.2 El sistema no podrá realizar la venta por falta de medicamentos. 3b.3 El sistema vuelve al paso 2. 3b.1 El sistema detecta que no hay ningún cliente ingresado en el sistema 3b.2 El sistema no podrá realizar la venta porque no hay clientes ingresados. 3b.3 El sistema vuelve al paso 2.

Tabla 32: CU ingresar venta.

Caso de uso: Mostrar Ventas Realizadas
ID: CU_11
Breve Descripción: Permite al Administrador/Farmacéutico/Vendedor /Bodeguero ver las ventas realizadas en el sistema.
Actores Principales: Administrador/Farmacéutico/Vendedor
Actores Secundarios: Ninguno
Precondición: 1. El Administrador/Farmacéutico/Vendedor ha ingresado al sistema.
Flujo Principal: 1. El caso de uso comienza cuando el Administrador/Farmacéutico/Técnico elige la opción de ventas 2. El sistema mostrará el listado con las ventas realizadas que posee el sistema.
Postcondición: Ninguna.
Flujos Alternativos: 2a.1 El sistema no posee ninguna venta registrada 2a.2 El sistema no mostrará ningún dato relacionado con las ventas ni las opciones tales como anular ventar, generar la boleta o ver los detalles de una venta. 2b.1 No existe ninguna venta al buscar el Rut del cliente en el buscador. 2b.2 El sistema no mostrará ningún dato relacionado con el Rut ingresado ni las opciones tales como anular ventar, generar la boleta o ver los detalles de una venta.

Tabla 33: CU mostrar ventas.

Anexo 3 Servicios web

ID	S_Login
URL	http://127.0.0.1:8000/login
Método	POST
Parámetros de la URL	http://127.0.0.1:8000/login
Parámetros de datos	/login
Respuesta Exitosa	Codigo:302 Solo se hace una redirección a la página principal.
Respuesta errónea	El sistema retorna un objeto json con el error.

Llamada de ejemplo	public function login(Request $request){
	$this->validateLogin($request);
	
	if (Auth::attempt(['usuario' => $request->usuario,
	'password' => $request->password,'condicion'=>1])){
	return redirect()->route('main');
	}
	return back()
	->withErrors(['usuario' =>trans('auth.failed')])
	->withInput(request(['usuario']));
	
	}
	
	protected function validateLogin(Request $request){
	$this->validate($request,[
	'usuario' => 'required|string',
	'password' => 'required|string'
	]);
	
	}

Tabla 30: Petición de ingreso al sistema.

ID	S_logout
URL	http://127.0.0.1:8000/logout
Método	GET
Parámetros de la URL	http://127.0.0.1:8000/logout
Parámetros de datos	/logout
Respuesta Exitosa	Codigo:302 Solo se hace una redirección a la página principal.
Respuesta errónea	No hay error solo no se visualizan datos.

Llamada de ejemplo	public function logout(Request $request){
	Auth::logout();
	$request->session()->invalidate();
	return redirect('/');
	}

Tabla 31: Petición para cerrar sesión en el sistema.

ID	S_main
URL	http://127.0.0.1:8000/main
Método	GET
Parámetros de la URL	http://127.0.0.1:8000/main
Parámetros de datos	/main
Respuesta Exitosa	Código: 200 Muestra la vista principal del sistema.
Respuesta errónea	Código: 404 No se encuentra la página.

Llamada de ejemplo	Route::get('/main', function () {
	return view('contenido/contenido');
	})->name('main');

Tabla 32: petición muestra vista principal.

ID	S_medicamento
URL	http://127.0.0.1:8000/ medicamento
Método	GET
Parámetros de la URL	http://127.0.0.1:8000/medicamento?page=1&buscar=&criterio=principio_activo
Parámetros de datos	/ medicamento?page=1&buscar=&criterio=principio_activo
Respuesta Exitosa	Codigo:200 { data: [,…] 0: {id: 4, nombre_comercial: "Valuems", principio_activo: "Ácido Valproico", Disponible: 1, precio: 7000,…} 1: {id: 3, nombre_comercial: "Aspirina", principio_activo: "Acetilsalicílico ácido", Disponible: 1,…} 2: {id: 2, nombre_comercial: "Clorfenamina", principio_activo: "Clorfeniramina",

	Disponible: 1,…} 3: {id: 1, nombre_comercial: "Parecetamol", principio_activo: "Acido acetilsalicílico", Disponible: 1,…} }
Respuesta errónea	No hay error solo no se visualizan datos.

Llamada de ejemplo	listarMedicamento(page, buscar, criterio) {
	let me = this;
	var url ="/medicamento?page=" +page +"&buscar=" +buscar +"&criterio=" +criterio;
	axios.get(url).then(function(response) {
	var respuesta = response.data;
	me.arrayMedicamento = respuesta.medicamentoController.data;
	me.pagination = respuesta.pagination;
	})
	.catch(function(error) {
	console.log(error);
	});
	},

Tabla 33: Petición listar datos de medicamentos.

ID	S_receta
URL	http://127.0.0.1:8000/receta
Método	GET
Parámetros de la URL	http://127.0.0.1:8000/receta?page=1&buscar=&criterio=rut
Parámetros de datos	/receta?page=1&buscar=&criterio=rut
Respuesta Exitosa	Codigo:200 { 0: {id: 18, id_cliente: 5, medico: "Cesar Morais", telefono: "8786758", fecha_ingreso: "2018-12-04",…} 1: {id: 17, id_cliente: 1, medico: "asdf", telefono: "54678797", fecha_ingreso: "2018-11-29",…} 2: {id: 16, id_cliente: 5, medico: "asd", telefono: "54678909", fecha_ingreso: "2018-11-29",…} 3: {id: 15, id_cliente: 1, medico: "andres", telefono: "78677890", fecha_ingreso: "2018-11-29",…} 4: {id: 14, id_cliente: 5, medico: "paula", telefono: "90785645", fecha_ingreso: "2018-11-29",…} 5: {id: 13, id_cliente: 1, medico: "diego", telefono: "89674578", fecha_ingreso: "2018-11-29",…} 6: {id: 12, id_cliente: 1, medico: "diego", telefono: "89674578", fecha_ingreso: "2018-11-29",…} 7: {id: 11, id_cliente: 1, medico: "adsa", telefono: "33243243242", fecha_ingreso: "2018-11-29",…} 8: {id: 10, id_cliente: 1, medico: "adsa", telefono: "33243243242", fecha_ingreso: "2018-11-29",…} 9: {id: 9, id_cliente: 1, medico: "juan", telefono: "78563423",

	fecha_ingreso: "2018-11-29",…} }
Respuesta errónea	No hay error solo no se visualizan datos.

Llamada de ejemplo	listarReceta (page,buscar,criterio){
	let me=this;
	var url= '/receta?page=' + page + '&buscar='+
	buscar + '&criterio='+ criterio;
	axios.get(url).then(function(response) {
	console.log(response);
	var respuesta= response.data;
	me.arrayReceta = respuesta.recetas.data;
	me.pagination= respuesta.pagination;
	})
	.catch(function (error) {
	console.log(error);
	});
	}

Tabla 34: Petición mostrar datos de recetas.

ID	S_user
URL	http://127.0.0.1:8000/user
Método	GET
Parámetros de la URL	http://127.0.0.1:8000/user?page=1&buscar=&criterio=rut
Parámetros de datos	/user?page=1&buscar=&criterio=rut

Respuesta Exitosa	Código: 200
	{
	data: [{id: 9, rut: "162346781", nombre: "Andres",
	apellido: "Navarro", fecha: "1980-11-21",…},…]
	0: {id: 9, rut: "162346781", nombre: "Andres", apellido:
	"Navarro", fecha: "1980-11-21",…}
	1: {id: 8, rut: "190250377", nombre: "Paula", apellido:
	"Espinoza", fecha: "1994-04-12",…}
	2: {id: 7, rut: "183456782", nombre: "Pedro", apellido:
	"Navarrete", fecha: "1998-12-03",…}
	3: {id: 6, rut: "198976752", nombre: "Juan", apellido:
	"Solis", fecha: "1994-01-13", direccion: "Ninhue",…}
	}

Respuesta errónea	No hay error solo no se visualizan datos.

Llamada de ejemplo	listarPersona (page,buscar,criterio){
	let me=this;
	var url= '/user?page=' + page + '&buscar='+
	buscar + '&criterio='+ criterio;
	axios.get(url).then(function (response) {
	var respuesta= response.data;
	me.arrayPersona = respuesta.personas.data;
	me.pagination= respuesta.pagination;
	})

	.catch(function (error) {
	console.log(error);
	});
	},

Tabla 35:Petición mostrar datos de los usuarios.

ID	S_roles
URL	http://127.0.0.1:8000/rol
Método	GET
Parámetros de la URL	http://127.0.0.1:8000/rol?page=1&buscar=&criterio=nombre
Parámetros de datos	/rol?page=1&buscar=&criterio=nombre

Respuesta Exitosa	Código: 200
	{
	data: [{id: 1, nombre: "Administrador", descripcion: "Administrador del sistema", condicion: 1},…]
	0: {id: 1, nombre: "Administrador", descripcion: "Administrador del sistema", condicion: 1}
	1: {id: 2, nombre: "Farmaceutico", descripcion: "Encargado de la farmacia", condicion: 1}
	2: {id: 3, nombre: "Vendedor", descripcion: "Tecnico en enfermeria", condicion: 1}
	3: {id: 4, nombre: "Bodegero", descripcion: "Encargado del inventario", condicion: 1}
	}

Respuesta errónea	No hay error solo no se visualizan datos.

Llamada de ejemplo	listarRol (page,buscar,criterio){
	let me=this;
	var url= '/rol?page=' + page + '&buscar='+ buscar + '&criterio='+ criterio;
	axios.get(url).then(function (response) {
	var respuesta= response.data;
	me.arrayRol = respuesta.roles.data;
	me.pagination= respuesta.pagination;
	})
	.catch(function (error) {
	console.log(error);
	});
	},

Tabla 36: Petición mostrar los datos de todos los roles.

ID	S_ actualizar
URL	http://127.0.0.1:8000/cliente/actualizar
Método	PUT
Parámetros de la URL	http://127.0.0.1:8000/cliente/actualizar
Parámetros de datos	/cliente/actualizar
Respuesta Exitosa	Código: 200 Se actualiza al cliente
Respuesta errónea	No hay error, solo no se visualizan datos.

Llamada de ejemplo	actualizarCliente() {
	if (this.validarCliente()) {
	return;
	}
	let me = this;
	axios
	.put("/cliente/actualizar", {
	rut: this.rut,
	nombre: this.nombre,
	apellido: this.apellido,
	fecha: this.fecha,
	telefono: this.telefono,
	domicilio: this.domicilio,
	id: this.id
	})
	.then(function(response) {
	me.cerrarModal();
	/* para buscar */
	me.listarCliente(1, "", "rut");
	})
	.catch(function(error) {
	console.log(error);
	});
	},

Tabla 37: Petición actualizar un registro en el sistema.

ID	S_desactivar
URL	http://127.0.0.1:8000/cliente/desactivar
Método	PUT
Parámetros de la URL	http://127.0.0.1:8000/cliente/desactivar
Parámetros de datos	/ cliente/desactivar
Respuesta Exitosa	Código: 200 Se cambia el estado del cliente.
Respuesta errónea	No hay error, solo no se visualizan datos.

Llamada de ejemplo	desactivarCliente(id) {
	swal({
	title: "¿Esta seguro de cambiar el estado

	del cliente?",
	type: "warning",
	showCancelButton: true,
	confirmButtonColor: "#3085d6",
	cancelButtonColor: "#d33",
	confirmButtonText: "Aceptar!",
	cancelButtonText: "Cancelar",
	confirmButtonClass: "btn btn-success",
	cancelButtonClass: "btn btn-danger",
	buttonsStyling: false,
	reverseButtons: true
	/* se da la funcionalidad */
	}).then(result => {
	if (result.value) {
	let me = this;
	axios.put("/cliente/desactivar", {
	id: id
	})
	.then(function(response) {
	me.listarCliente(1, "", "rut");
	swal(
	"Desactivado!",
	"El estado ha sido cambiado con éxito.",
	"success"
	);
	})
	.catch(function(error) {
	console.log(error);
	});
	} else if (
	// Read more about handling dismissals
	result.dismiss === swal.DismissReason.cancel
	) {
	}
	});
	},

Tabla 38: Petición eliminar un registro del sistema.

ID	S_activar
URL	http://127.0.0.1:8000/cliente/activar
Método	PUT
Parámetros de la URL	http://127.0.0.1:8000/cliente/activar
Parámetros de datos	/ cliente/activar
Respuesta Exitosa	Código: 200 Se cambia el estado del cliente.
Respuesta errónea	No hay error, solo no se visualizan datos.

Llamada de ejemplo	activarCliente(id) {
	swal({
	title: "¿Esta seguro de cambiar el estado del cliente?",
	type: "warning",
	showCancelButton: true,
	confirmButtonColor: "#3085d6",
	cancelButtonColor: "#d33",
	confirmButtonText: "Aceptar!",
	cancelButtonText: "Cancelar",
	confirmButtonClass: "btn btn-success",
	cancelButtonClass: "btn btn-danger",
	buttonsStyling: false,
	reverseButtons: true
	}).then(result => {
	if (result.value) {
	let me = this;
	axios
	.put("/cliente/activar", {
	id: id
	})
	/* si sale bien */
	.then(function(response) {
	me.listarCliente(1, "", "rut");
	swal(
	"Activado!",
	"El estado ha sido cambiado con éxito.",
	"success"
	);
	})
	/* si sale mal */
	.catch(function(error) {
	console.log(error);
	});
	} else if (
	// Read more about handling dismissals
	result.dismiss ===

	swal.DismissReason.cancel
	) {
	}
	});
	},

Tabla 39: Petición para activar un registro del sistema.

ID	S_ obtenerCabecera
URL	http://127.0.0.1:8000/medicamento/obtenerCabecera
Método	GET
Parámetros de la URL	http://127.0.0.1:8000/medicamento/obtenerCabecera?id=4
Parámetros de datos	/medicamento/obtenerCabecera?id=4

Respuesta Exitosa	Código: 200
	
	{
	id: 4, nombre_comercial: "Valuems", principio_activo: "Ácido Valproico", Disponible: 1, precio: 7000,…}
	Disponible: 1
	advertencias_precauciones: "No consumir con alcohol"
	contraindicaciones: "Consumir bajo solo indicacion medica"
	id: 4
	indicaciones_terapeuticas: "Tomar cada 12 horas"
	mecanismo_accion: "inhibe las enzimas que catabolizan este neurotransmisor o bloquea la recaptación del GABA por el sistema nervioso central."
	modo_administracion: "Oral"
	nombre_comercial: "Valuems"
	posologia: "500 mg"
	precio: 7000
	principio_activo: "Ácido Valproico"
	stock: 18
	}

Respuesta errónea	No hay error, solo no se visualizan datos.

Llamada de ejemplo	verMedicamento1(id){
	let me=this;
	me.listado=3;
	
	var arrayMedicamentoT=[];
	var url='/medicamento/obtenerCabecera?id=' + id;
	
	axios.get(url).then(function (response) {
	console.log(response);
	var respuesta= response.data;
	arrayMedicamentoT = respuesta.medicamento;
	
	me.nombre_comercial = arrayMedicamentoT[0]['nombre_comercial'];
	me.principio_activo = arrayMedicamentoT[0]['principio_activo'];
	me.Disponible = arrayMedicamentoT[0]['Disponible'];
	me.precio=arrayMedicamentoT[0]['precio'];
	me.stock=arrayMedicamentoT[0]['stock'];
	me.mecanismo_accion=arrayMedicamentoT[0]['mecanismo_accion'];
	
	me.indicaciones_terapeuticas=arrayMedicamentoT[0]['indicaciones_terapeuticas'];
	me.posologia=arrayMedicamentoT[0]['posologia'];
	
	me.modo_administracion=arrayMedicamentoT[0]['modo_administracion'];
	me.contraindicaciones=arrayMedicamentoT[0]['contraindicaciones'];
	
	me.advertencias_precauciones=arrayMedicamentoT[0]['advertencias_precauciones'];
	})
	.catch(function (error) {
	console.log(error);
	});
	
	//Obtener los datos de los detalles
	
	},

Tabla 40: Petición para mostrar los detalles de un registro determinado.

ID	S_ selectCliente
URL	http://127.0.0.1:8000/cliente/selectCliente
Método	GET
Parámetros de la URL	http://127.0.0.1:8000/cliente/selectCliente?filtro=184513250
Parámetros de datos	/cliente/selectCliente?filtro=184513250

Respuesta Exitosa	Código: 200
	{
	id: 1, rut: "184513250", nombre: "diego alexis", apellido: "navarrete"
	}

Respuesta errónea	No hay error, solo no se visualizan datos.

Llamada de ejemplo	selectCliente(search,loading){
	let me=this;
	loading(true)
	
	var url= '/cliente/selectCliente?filtro='+search;
	axios.get(url).then(function (response) {
	let respuesta = response.data;
	q: search
	me.arrayCliente=respuesta.clientes;
	loading(false)
	})
	.catch(function (error) {
	console.log(error);
	});
	},

Tabla 41: Petición para seleccionar un RUT de un cliente.

ID	S_listapdf
URL	http://127.0.0.1:8000/medicamento/listarpdf
Método	GET
Parámetros de la URL	http://127.0.0.1:8000/medicamento/listarpdf
Parámetros de datos	/medicamento/listarpdf
Respuesta Exitosa	Código: 200 Se genera un reporte en PDF.
Respuesta errónea	No hay error, solo no se visualizan datos.

Llamada de ejemplo	cargarPdf(){
	window.open('http://127.0.0.1:8000/medicamento/listaPdf','_blank');
	
	},

Tabla 42: Petición para generar reporte PDF.

ID	S_ obtenerDetalles
URL	http://127.0.0.1:8000/ /boleta/obtenerDetalles
Método	GET
Parámetros de la URL	http://127.0.0.1:8000/boleta/obtenerDetalles?id=11
Parámetros de datos	/ /boleta/obtenerDetalles?id=11

Respuesta Exitosa	Código: 200
	{
	cantidad: 1, nombre_comercial: "Aspirina", principio_activo: "Acetilsalicílico ácido", precio: 1000
	}

Respuesta errónea	No hay error solo no se visualizan datos.

Llamada de ejemplo	verBoleta(id){
	let me=this;
	me.listado=2;
	
	//Obtener los datos del ingreso
	var arrayBoletaT=[];
	var url= '/boleta/obtenerCabecera?id=' + id;
	
	axios.get(url).then(function (response) {
	console.log(response);
	var respuesta= response.data;
	arrayBoletaT = respuesta.boletaController;
	me.id=arrayBoletaT[0]['id'];
	me.rut=arrayBoletaT[0]['rut'];
	
	me.created_at=arrayBoletaT[0]['created_at'];
	me.estado=arrayBoletaT[0]['estado'];
	me.total=arrayBoletaT[0]['total'];
	})
	.catch(function (error) {
	console.log(error);
	});
	
	//Obtener los datos de los detalles
	var urld= '/boleta/obtenerDetalles?id=' + id;
	
	axios.get(urld).then(function (response) {
	console.log(response);
	var respuesta= response.data;
	me.arrayDetalle = respuesta.detalleBoleta;
	})
	.catch(function (error) {
	console.log(error);
	});
	},

Tabla 43: Petición para ver los detalles de una boleta en específico.

ID	S_selectRol
URL	http://127.0.0.1:8000//rol/selectRol
Método	GET
Parámetros de la URL	http://127.0.0.1:8000//rol/selectRol
Parámetros de datos	/rol/selectRol
Respuesta Exitosa	Código: 200 { 0: {id: 1, nombre: "Administrador"}

	1: {id: 4, nombre: "Bodegero"} 2: {id: 2, nombre: "Farmaceutico"} 3: {id: 3, nombre: "Vendedor"} }
Respuesta errónea	No hay error solo no se visualizan datos.

Llamada de ejemplo	selectRol(){
	let me=this;
	var url= '/rol/selectRol';
	axios.get(url).then(function
	(response) {
	var respuesta= response.data;
	me.arrayRol = respuesta.roles;
	})
	.catch(function (error) {
	console.log(error);
	});
	},

Tabla 44: Petición para seleccionar un rol.

ID	S_ pdf/{id}
URL	http://127.0.0.1:8000/boleta/pdf/{id}
Método	GET
Parámetros de la URL	http://127.0.0.1:8000/boleta/pdf/{id}
Parámetros de datos	/pdf/{id}
Respuesta Exitosa	Código: 200 Se genera una boleta en PDF.
Respuesta errónea	No hay error, solo no se visualizan datos.
Llamada de ejemplo	pdfVenta(id){ window.open('http://127.0.0.1:8000/boleta/pdf/'+ id); },

Tabla 45: Petición para generar una boleta.

ID	S_boleta
URL	http://127.0.0.1:8000/boleta
Método	GET
Parámetros de la URL	http://127.0.0.1:8000/boleta?page=1&buscar=&criterio=rut
Parámetros de datos	/boleta?page=1&buscar=&criterio=rut
Respuesta Exitosa	Codigo:200 { data: [{id: 11, created_at: "2018-12-08 03:29:01", estado: "Registrado", total: 1000, rut: "179087862",…},…] 0: {id: 11, created_at: "2018-12-08 03:29:01", estado: "Registrado", total: 1000, rut: "179087862",…}

	1: {id: 10, created_at: "2018-12-05 15:22:51", estado: "Registrado", total: 14000, rut: "184513250",…} 2: {id: 9, created_at: "2018-12-05 15:06:20", estado: "Registrado", total: 4200, rut: "18989546",…} 3: {id: 8, created_at: "2018-12-05 14:09:16", estado: "Registrado", total: 4000, rut: "18989546",…} 4: {id: 7, created_at: "2018-12-03 15:20:09", estado: "Registrado", total: 10000, rut: "179087862",…} 5: {id: 6, created_at: "2018-12-01 20:42:19", estado: "Anulada", total: 8400, rut: "179087862",…} 6: {id: 5, created_at: "2018-12-01 17:14:29", estado: "Registrado", total: 12000, rut: "184513250",…} 7: {id: 4, created_at: "2018-12-01 15:25:45", estado: "Registrado", total: 3100, rut: "184513250",…} 8: {id: 3, created_at: "2018-11-19 00:00:00", estado: "Anulada", total: 0, rut: "184513250",…} }
Respuesta errónea	No hay error, solo no se visualizan datos.

Llamada de ejemplo	listarBoleta (page,buscar,criterio){
	let me=this;
	var url= '/boleta?page=' + page + '&buscar='+ buscar + '&criterio='+ criterio;
	axios.get(url).then(function (response) {
	var respuesta= response.data;
	me.arrayBoleta = respuesta.boletaController.data;
	me.pagination= respuesta.pagination;
	})
	.catch(function (error) {
	console.log(error);
	});
	},

Tabla 46: Petición para ver todas las boletas.

Anexo 4 Descripción de Controladores

Controlador	Funcionalidad	Funciones	Objetivo
ClienteController	Administrar todos los métodos de la clase cliente.	Index	Mostrar los datos de todos los clientes en el sistema.
		store	Actualizar los registros de una boleta determinada.
		SelectCliente	muestra el RUT del cliente a medida que se va escribiendo.
		Update	Actualiza los registros de un determinado cliente.
		listarPdf	Genera un informe de formato PDF de todos los clientes registrados en el sistema.
		Desactivar	Cambia el estado de un cliente para ver si aún acceder a la farmacia.
		Activar	Cambia el estado de un cliente para ver si aún acceder a la farmacia.

Tabla 47: Detalle de controlador cliente.

Controlador	Funcionalidad	Funciones	Objetivo
DashboardController	Administra el metodo que reúne todos los datos que son usados para generar los reportes gráficos.	_invoke	Esta es una función que presta Laravel, se utiliza cuando un controlador solo tiene una única función. Tomar los datos de la base de datos para mostrarlos en los reportes gráficos.

Tabla 48: Detalle de controlador dashboard.

Controlador	Funcionalidad	Funciones	Objetivo
MedicamentoController	Administrar todos los métodos de la clase medicamento.	Index	Mostrar los datos de todos los medicamentos en el sistema.
		store	Actualizar los registros de un medicamento determinado.
		Update	Actualiza los registros de un determinado medicamento.
		listarPdf	Genera un informe de formato PDF de todos los medicamentos registrados en el sistema.
		Desactivar	Cambia el estado de un medicamento para ver si está disponible.
		Activar	Cambia el estado de un medicamento para ver si está disponible.
		obtenerCabecera	Toma los datos de un medicamento determinado.

Tabla 49: Detalles del controlador medicamentos.

Controlador	Funcionalidad	Funciones	Objetivo
RecetaController	Administrar todos los métodos de la clase receta.	Index	Mostrar los datos de todas las recetas en el sistema.
		store	Actualizar los registros de una receta determinado.
		obtenerCabecera	Muestra los datos de una receta determinada.
		Desactivar	Cambia el estado de una receta para ver si está disponible.
		obtenerDetalles	Se complementa con ObtenerCabecera y tiene el propósito de mostrar todos los medicamentos que posee una boleta

Tabla 50:Detalles de controlador de Receta.

Controlador	Funcionalidad	Funciones	Objetivo
UserController	Administrar todos los métodos de la clase usuario.	index	Mostrar los datos de todos los usuarios en el sistema.
		store	Actualizar los registros de un usuario determinado.
		update	Actualiza los registros de un determinado usuario.
		desactivar	Cambia el estado de un usuario para que este tenga acceso al sistema.
		activar	Cambia el estado de un usuario para el acceso al sistema.

Tabla 51: Detalles de controlador de usuarios.

Controlador	Funcionalidad	Funciones	Objetivo
RolController	Administrar todos los métodos de roles	index	Mostrar los datos de todos los usuarios en el sistema.
		SelectRol	Permite seleccionar el tipo de rol que tendrá un usuario al ser ingresado al sistema.

Tabla 52: Detalles de controlador de roles.

Controlador	Funcionalidad	Funciones	Objetivo
LoginController	Administrar todos los métodos del login.	showLoginForm	Retorna la vista login.
		login	Valida los datos del usuario para el ingreso al sistema y también si los datos están incorrectos.
		validateLogin	Valida que los datos ingresados sean string.
		logout	Cierra la sesión del usuario.

Tabla 53: Detalles de controlador login.

Anexo 5 Navegación y diseño de la interfaz del sistema

Navegación del sistema

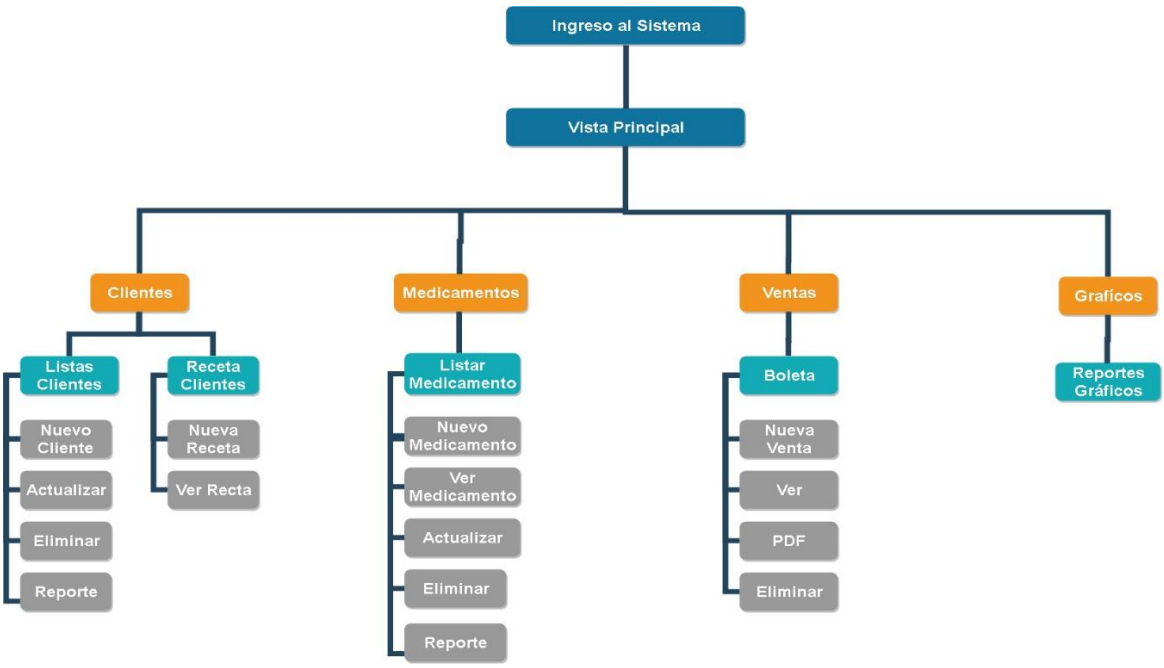

Figura 28: Mapa de navegación de "Farmacéutico".

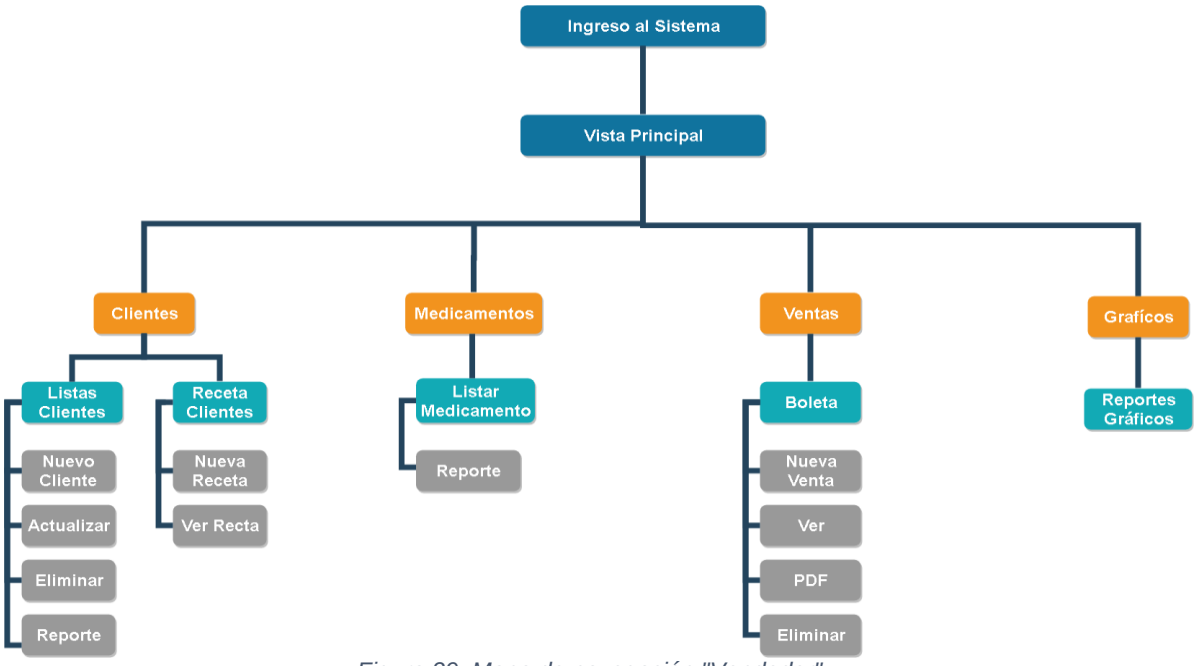

Figura 29. Mapa de navegación "Vendedor".

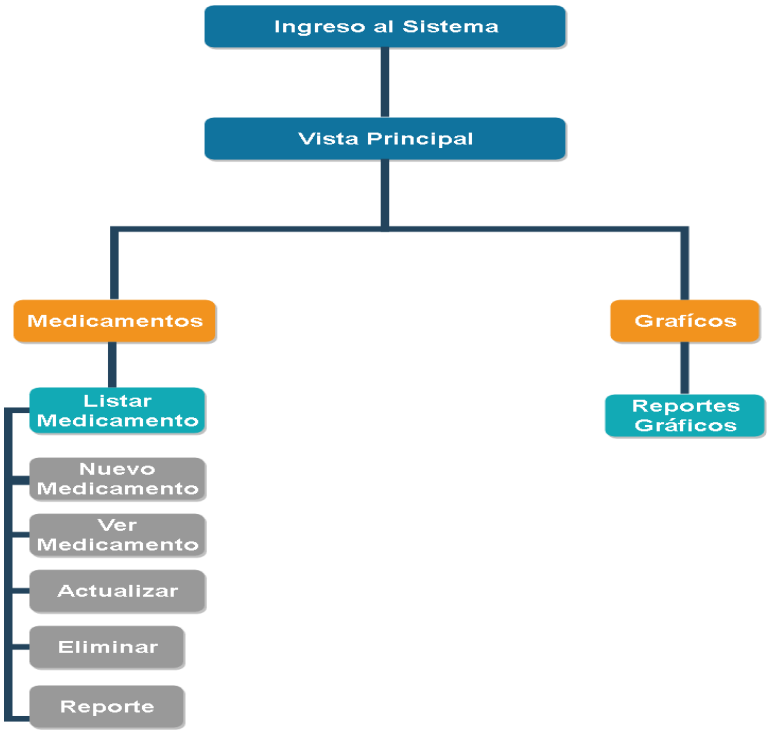

Figura 30: Mapa de navegación "Bodeguero".

Diseño de interfaz del sistema

Ventana modal registrar clientes
Ventana modal que nos permite registrar un nuevo cliente en el sistema.

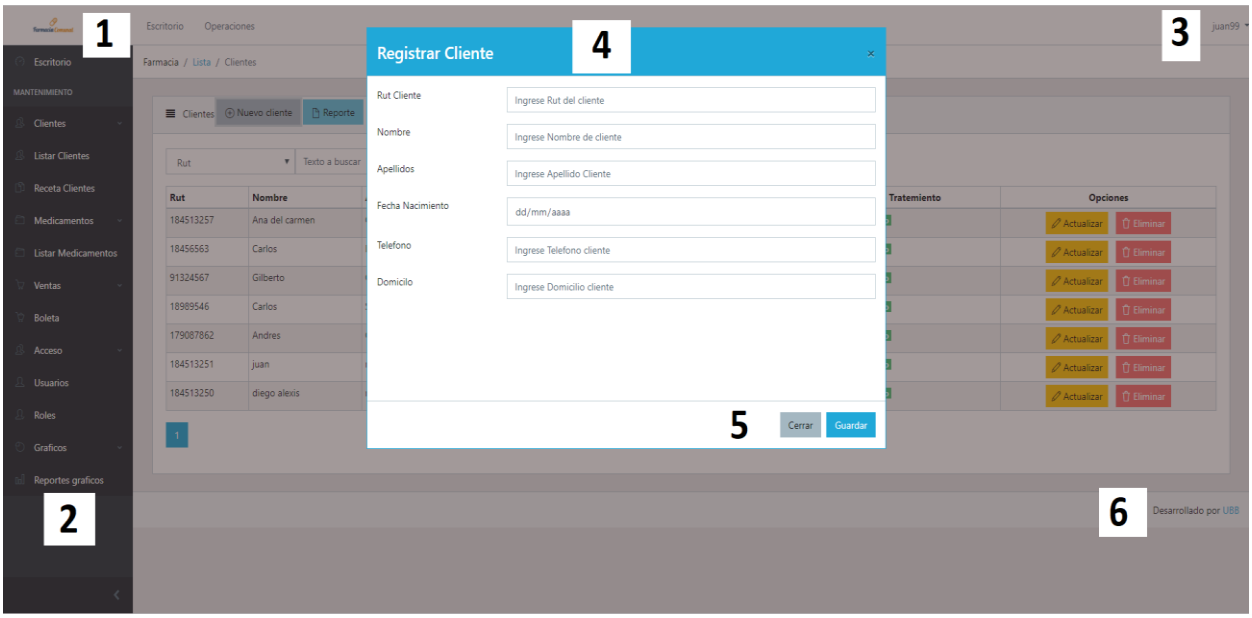

Figura 31. Interfaz de ingreso cliente.

N.º Área	Representación
1	Logo de la farmacia.
2	Menú Principal.
3	Barra Superior y nombre de perfil.
4	Formulario de registro de nuevo cliente.
5	Botón de guardar cliente/ Botón cancelar acción.
6	Pie de página.

Tabla 54: Diseño interfaz "Registrar Cliente"

Vista agregar receta

Módulo que nos permite agregar una receta en el sistema.

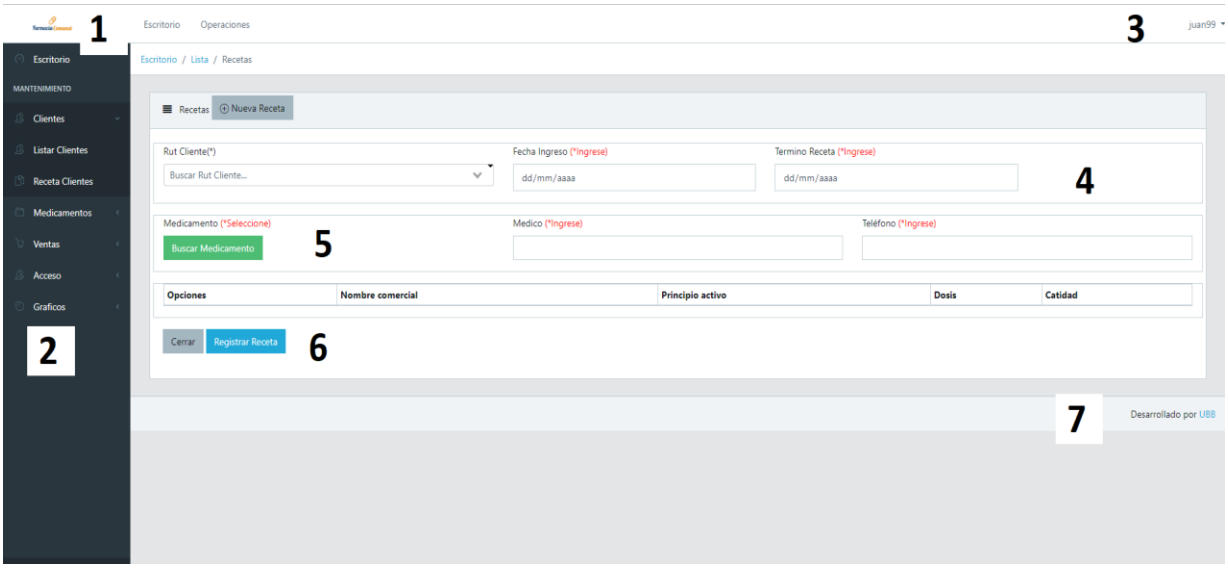

Figura 32. Interfaz vista agregar receta.

N.º Área	Representación
1	Logo de la farmacia.
2	Menú Principal.
3	Barra Superior y nombre de perfil.
4	Formulario de registro nueva receta.
5	Botón buscar medicamento en el sistema.
6	Botón de cerrar ventana/ Botón cancelar acción.
7	Pie de página.

Tabla 55: Diseño interfaz "Agregar receta"

Vista agregar ventas

Modulo que nos permite realizar una nueva venta en el sistema.

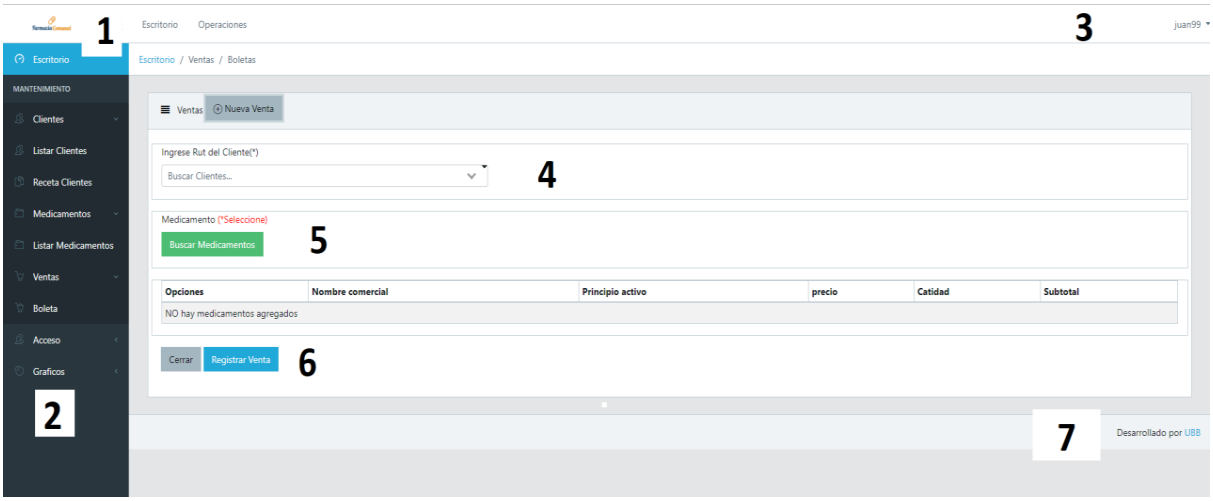

Figura 33. Interfaz de agregar venta.

N.º Área	Representación
1	Logo de la farmacia.
2	Menú Principal.
3	Barra Superior y nombre de perfil.
4	Formulario de registro nueva venta.
5	Botón buscar medicamento en el sistema.
6	Botón de cerrar ventana/ Botón cancelar acción
7	Pie de página.

Tabla 56: Diseño interfaz "Agregar nueva venta"

Ventana modal agregar nuevo medicamento.

Ventana modal que nos permite agregar un nuevo medicamento en el sistema.

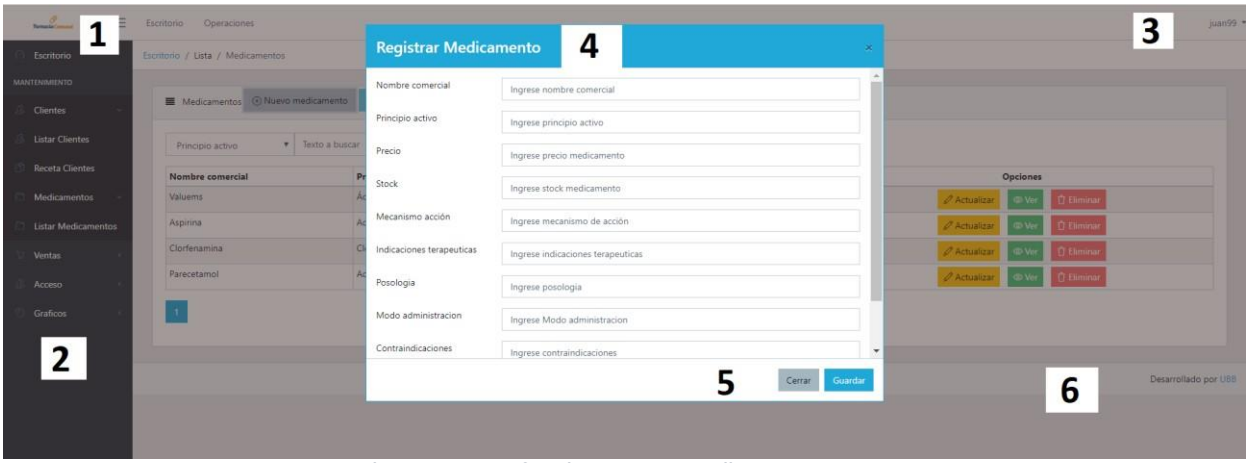

Figura 34. Interfaz de agregar medicamento.

N.º Área	Representación
1	Logo de la farmacia.
2	Menú Principal.
3	Barra Superior y nombre de perfil.
4	Formulario de registro de nuevo medicamento.
5	Botón de guardar cliente/ Botón cancelar acción.
6	Pie de página.

Tabla 57: Diseño interfaz "Registrar nuevo medicamento"

Ventana acceso usuarios

Módulo que nos permite ver el listado de usuarios del sistema con sus respectivos roles.

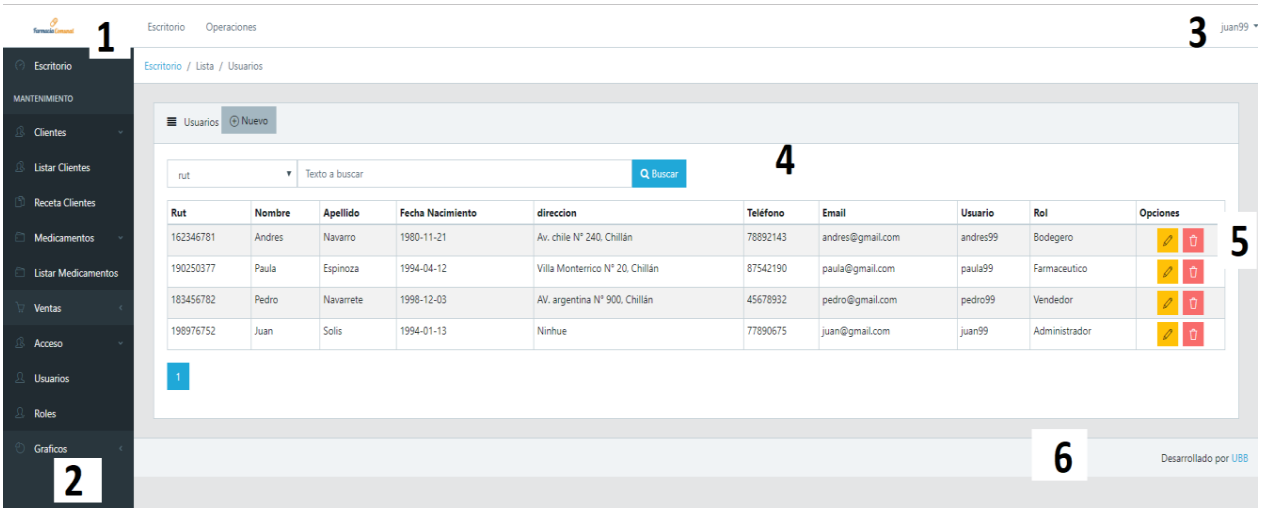

Figura 35. interfaz mostrar usuarios.

N.º Área	Representación
1	Logo de la farmacia.
2	Menú Principal.
3	Barra Superior y nombre de perfil.
4	Contenido de la página/ Buscador de usuarios.
5	Botón editar un usuario/ Botón Eliminar usuario.
6	Pie de página.

Tabla 58: Diseño interfaz "Acceso usuarios"

Ventana registrar usuario del sistema

Ventana modal que nos permite registrar un nuevo usuario del sistema.

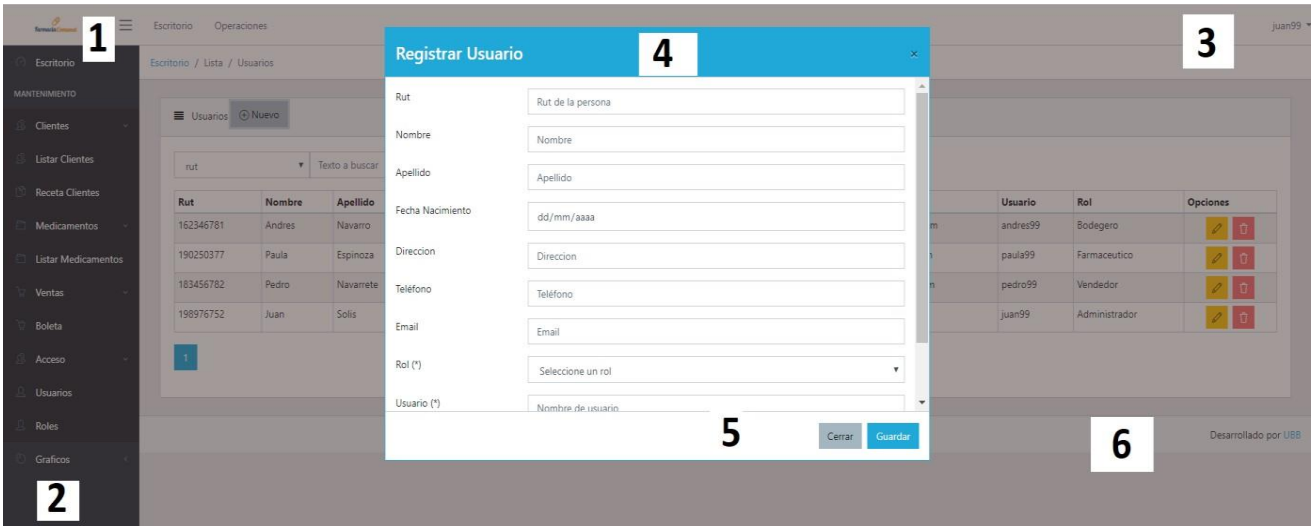

Figura 36. Interfaz de ingresar usuario.

N.º Área	Representación
1	Logo de la farmacia.
2	Menú Principal.
3	Barra Superior y nombre de perfil.
4	Formulario de registro de nuevo usuario del sistema.
5	Botón de guardar cliente/ Botón cancelar acción.
6	Pie de página.

Tabla 59: Diseño interfaz "Registrar nuevo usuario".

Módulo Roles:

Permite ver los roles que poseen los usuarios en el sistema. Este módulo solo lo puede ver el usuario que posea el rol de administrador del sistema.

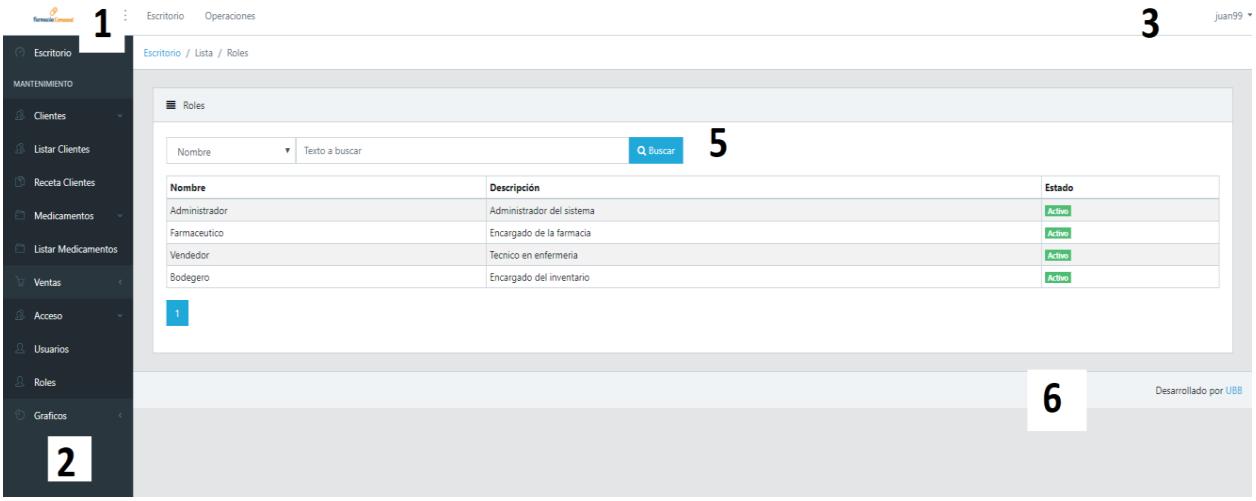

Figura 37. Interfaz mostrar roles.

N.º Área	Representación
1	Logo de la farmacia.
2	Menú Principal.
3	Barra Superior y nombre de perfil.
4	Formulario de registro de nuevo usuario del sistema.
5	Botón de guardar cliente/ Botón cancelar acción.
6	Pie de página.

Tabla 60: Diseño interfaz "Roles"

Anexo 6 Pruebas

Escenario cliente:
Fecha de prueba: 28/11/2018

ID	Objetivo	Contexto	Entrada	Salida	Evento	Resultado	Evaluación
P05	Ingresar un cliente al sistema.	Ingresar los datos por medio de un formulario.	Todos los datos relacionados con un cliente (RUT, nombres, apellido, entre otros).	Se muestra la pantalla de los clientes ingresados con el nuevo registro.	Al presionar el botón guardar.	Se ingresar a un nuevo cliente.	Éxito.
P06	Ingresar un cliente al sistema que ya está ingresado.	Ingresar los datos por medio de un formulario.	Todos los datos relacionados con un cliente (RUT, nombres, apellido, entre otros).	No permite el ingreso, pero hay errores.	Al presionar el botón guardar.	No se ingresa al sistema.	Fallo.
P07	Actualizar los datos de un cliente ingresado al sistema	Actualizar los datos por medio de un formulario.	Los datos que se modifican en el formulario.	Actualización de los datos del cliente.	Al presionar el botón actualizar.	Se actualizan los registros.	Éxito.
P08	Buscar un cliente por medio del RUT.	RUT está contenido en la BD.	Ingresar el RUT del cliente en la barra de búsqueda.	Se muestra el cliente a quien le corresponda el RUT.	Al presionar el botón buscar.	Se muestra al cliente.	Éxito.
P09	Buscar un cliente por medio de un RUT incorrecto.	RUT está contenido en la BD.	Ingresar el RUT del cliente en la barra de búsqueda.	No se muestran registros.	Al presionar el botón buscar.	No se muestra al cliente.	Éxito.

Tabla 61: Prueba de escenario cliente.

Escenario receta:
Fecha de prueba: 28/11/2018

ID	Objetivo	Contexto	Entrada	Salida	Evento	Resultado	Evaluación
P10	Ingresar una receta al sistema.	Ingresar los datos por medio de un formulario.	Todos los datos relacionados con un cliente (RUT, nombres, apellido, medicamentos, entre otros).	Se muestra la pantalla de las recetas ingresadas con el nuevo registro.	Al presionar el botón registrar receta.	Se ingresar a una nueva receta.	Éxito.
P11	Ingresar una receta al sistema que ya está ingresada.	Ingresar los datos por medio de un formulario.	Todos los datos relacionados con una receta (RUT, nombres, apellido, medicamentos, entre otros).	No permite el ingreso, pero hay errores.	Al presionar el botón registrar receta.	No se ingresa al sistema.	Fallo.
P12	Buscar una receta por medio del RUT.	RUT está contenido en la BD.	Ingresar el RUT del cliente en la barra de búsqueda.	Se muestra la receta de quien le corresponda el RUT.	Al presionar el botón buscar.	Se muestra la receta.	Éxito.
P13	Buscar una receta por medio de un RUT incorrecto.	RUT está contenido en la BD.	Ingresar el RUT del cliente en la barra de búsqueda.	No se muestran registros.	Al presionar el botón buscar.	No se muestra la receta.	Éxito.

Tabla 62: Prueba de escenario de receta.

Escenario medicamento:
Fecha de prueba: 28/11/2018

ID	Objetivo	Contexto	Entrada	Salida	Evento	Resultado	Evaluación
P14	Ingresar un medicamento al sistema.	Ingresar los datos por medio de un formulario.	Todos los datos relacionados con un medicamento (nombre comercial, principio activo, precio, stock, entre otros).	Se muestra la pantalla de los medicamentos ingresados con el nuevo registro.	Al presionar el botón guardar.	Se ingresar a un nuevo medicamento.	Éxito.
P15	Ingresar un medicamento	Ingresar los datos por	Datos del medicamento a ingresar.	Mensaje de que faltan medicamentos.	Al presionar	No se puede registrar el medicamento.	Éxito.

	con datos faltantes.	medio de un formulario.			el botón guardar.		
P16	Actualizar los datos de un medicamento ingresado al sistema	Actualizar los datos por medio de un formulario.	Los datos que se modifican en el formulario.	Actualización de los datos del medicamento.	Al presionar el botón actualizar.	Se actualizan los registros.	Éxito.
P17	Buscar un medicamento por medio del principio activo.	Principio activo está contenido en la BD.	Ingresar el principio activo del medicamento en la barra de búsqueda.	Se muestra el medicamento de quien le corresponda el principio activo.	Al presionar el botón buscar.	Se muestra el medicamento.	Éxito.
P18	Buscar un medicamento por medio del principio activo incorrecto.	Principio activo está contenido en la BD.	Ingresar el principio activo del medicamento en la barra de búsqueda.	No se muestran registros.	Al presionar el botón buscar.	No se muestran registros.	Éxito.
P19	Generar informe PDF de todos los medicamentos registrados en el sistema.	Todos los datos de los medicamentos están la BD.	Ejecución del botón reporte.	Se genera un reporte PDF con todo los medicamentos y el total de estos.	Al presionar el botón reporte.	Se descargar el reporte.	Éxito.

Tabla 63: Prueba de escenario de medicamento.

Escenario usuario:
Fecha de prueba: 28/11/2018

ID	Objetivo	Contexto	Entrada	Salida	Evento	Resultado	Evaluación
P20	Ingresar un usuario al sistema.	Ingresar los datos por medio de un formulario.	Todos los datos relacionados con un usuario (RUT, nombres, apellido, entre otros).	Se muestra la pantalla de los usuarios ingresados con el nuevo registro.	Al presionar el botón guardar.	Se ingresar a un nuevo usuario.	Éxito.
P21	Ingresar un usuario al sistema que ya está ingresado.	Ingresar los datos por medio de un formulario.	Todos los datos relacionados con un usuario (RUT, nombres, apellido, entre otros).	No permite el ingreso, pero hay errores.	Al presionar el botón guardar.	No se ingresa al sistema.	Fallo.

P22	Actualizar los datos de un usuario. ingresado al sistema	Actualizar los datos por medio de un formulario.	Los datos que se modifican en el formulario.	Actualización de los datos del usuario.	Al presionar el botón actualizar.	Se actualizan los registros.	Éxito.
P23	Buscar un usuario por medio del RUT.	RUT está contenido en la BD.	Ingresar el RUT del usuario en la barra de búsqueda.	Se muestra el usuario a quien le corresponda el RUT.	Al presionar el botón buscar.	Se muestra al usuario.	Éxito.
P24	Buscar un usuario por medio de un RUT incorrecto.	RUT está contenido en la BD.	Ingresar el RUT del usuario en la barra de búsqueda.	No se muestran registros.	Al presionar el botón buscar.	No se muestran registros.	Éxito.

Tabla 64: Prueba de escenario de usuario.

Fecha de prueba: 26/12/2018

ID	Objetivo	Contexto	Entrada	Salida	Evento	Resultado	Evaluación
P27	Ingresar un medicamento con unidad de medida.	Ingresar los datos por medio de un formulario.	Datos del medicamento.	No existe salida.	Al presionar el botón registrar medicamento.	Sin resultados.	Fallo por falta de sección para seleccionar el tipo de unidad de medida del medicamento.
P28	Ingresar un cliente con su fecha de nacimiento.	Ingresar los datos por medio de un formulario.	Datos del cliente.	Se muestra la lista de clientes ingresados.	Al presionar el botón registrar venta	No se puede registrar la venta, de esta forma.	Fallo en la validación de la fecha de nacimiento.
P29	Ver detalle de ventas realizadas por año.	Ver el total de ventas realizadas por cada año.	Datos de las ventas están en la BD.	Se muestra el grafico por mes.	Al ingresar a la vista reportes gráficos.	Gráficos por mes de las ventas realizadas.	Fallo, por falta de un filtro para mostrar las ventas realizadas por año.
P30	Formato de boleta al ingresar una venta	Generar una boleta al realizar una venta.	Los datos son ingresados por medio de un formulario.	Se muestra la boleta en formato PDF.	Al ingresar la venta.	Se genera la boleta en formato PDF.	Fallo en el formato de la boleta, adaptar al formato impuesto por el Sii.

Tabla 65: Pruebas generales del sistema.