
UNIVERSIDAD TECNOLÓGICA NACIONAL

FACULTAD REGIONAL SANTA FE

# DESARROLLO DE SISTEMA DE GESTIÓN DE MUTUALES

# TESIS

**INGENIERO EN SISTEMAS DE INFORMACIÓN**

PRESENTA

CÉSAR LUIS DEAN

DIRECTORA DE TESIS: DRA. LUCIANA BALLEJOS

SANTA FE, AGOSTO DE 2023

**Agradecimientos**

A mi madre, por haberme inculcado los valores que me guiaron hasta este día, como así también haberme ayudado económica y motivacionalmente para acceder a una educación de nivel superior.

A mis hermanos que siempre me marcaron el camino y me apoyaron para terminar la carrera.

A mi directora de tesis, Dra. Luciana Ballejos por su apoyo y asesoramiento permanente.

A mis compañeros de facultad que siempre estuvieron cerca para ayudarme, dándome consejos desinteresados.

# Tabla de contenido

# 1. Introducción

## *1.1.* Delimitación del tema

El mutualismo fue adquiriendo un carácter organizado e integrado con la comunidad. Su capacidad de complementar -y a veces suplir- la falta de acción del Estado, ha generado nuevos cauces de acción, en la tarea de contribuir al bienestar común. Las actuales prestaciones presentan un espectro muy variado de posibilidades y beneficios.

Las mutuales han crecido en cantidad, en número de socios, en número de prestaciones y en patrimonio. Las prestaciones se han ido refinando y ampliando, y en la actualidad existen infinidad de entidades que prestan reales servicios a la comunidad.

En los últimos tiempos, el crecimiento en las estructuras organizativas de las mutuales ha generado nuevas necesidades. Entre ellas, por ejemplo, las más sobresalientes son la de facilitar la rapidez en las comunicaciones e integrar consistentemente la información gestionada, para lograr una mejor y más eficiente administración potenciando su productividad administrativa.

## *1.2.* Objetivo general

El objetivo de este proyecto es desarrollar e implementar un sistema de gestión web para la Mutual General Obligado, a fin de satisfacer los requerimientos de circuitos administrativos de la entidad, como ser: administración de la información del asociado, cobranza de servicios, administración del cajero, entre otros; integrando y distribuyendo la información gestionada a lo largo de toda su estructura organizativa que se encuentra distribuida geográficamente. El sistema contemplará tanto la lógica propia del negocio, como la solución administrativa/contable para el seguimiento y tratamiento de los distintos servicios que se brindan a los asociados desde distintas partes del país, mediante las facilidades brindadas por una aplicación web.

## *1.3.* Objetivos específicos

- Analizar las necesidades reales de la organización aplicando técnicas y metodologías de Ingeniería de Requerimientos.
- Generar un Documento de Requerimientos de la aplicación (a ser aprobado por los gerentes/directores/etc. de la organización.)
- Diseñar y desarrollar el sistema de información de acuerdo con la metodología de

desarrollo propuesta.

- Obtener un sistema implementado y aprobado por el usuario a partir de un proceso de pruebas.
- Estudiar el uso de lenguajes de desarrollos y herramientas para su implementación.
- Estudiar el sistema de base de datos que sea la mejor solución para este sistema.
- Generar y entregar un Manual de Usuario como parte del producto final.
- Capacitar a los usuarios en base al Manual de Usuario generado.

## *1.4.* Ámbito del sistema

El alcance de este proyecto se extenderá a una sucursal en la provincia de Santa Fe. Sin embargo, se contempla que en un futuro pueda abarcar otras sucursales ubicadas en otras provincias del país, como, por ejemplo, Santiago del Estero, Chaco, Corrientes y Córdoba, para lo cual no serían necesarias modificaciones de fondo al sistema.

También se espera que en el futuro el sistema aumente su alcance y tamaño para incorporar soluciones a más servicios, como el de ayudas económicas, farmacia, proveedurías, seguros, etc.

## *1.5.* Descripción del Problema

En la actualidad, el esfuerzo global de las organizaciones, y en particular de las mutuales, está centrado en lograr un rápido acceso a la información necesaria y vital para su funcionamiento, así como también entender que la mutualidad se encuentra ante el desafío de rediseñarse en un contexto de profundos cambios económicos, geopolíticos, culturales y tecnológicos. Para ello, las mismas deben contar con recursos económicos para desarrollar o adquirir los productos de software que de alguna manera colaboren a disminuir las ineficiencias y a soportar los procedimientos de manera automatizada.

En particular, el mutualismo argentino, se distingue del que se practica en otros países por la gran diversidad de servicios que presta, teniendo como guía todo aquello relacionado al bienestar del ser humano. Esto hace que sea imprescindible adoptar las innovaciones tecnológicas necesarias, para que las mutuales puedan adaptarse a los tiempos actuales.

En este momento, la Mutual General Obligado se encuentra en los primeros pasos de su creación. El aporte económico necesario para afrontar los costos del desarrollo de un sistema

en esta etapa es escaso. Por otra parte, los paquetes de software que hay en el mercado con funcionalidades similares, no se adaptan a las necesidades específicas de esta institución. Por lo tanto, esta institución optó por solicitarme el desarrollo de un producto a medida, considerando el resultado como una oportunidad de negocio, ya que contará con un sistema de gestión a su medida que considere todas sus necesidades y requerimientos.

## *1.6.* Solución al Problema

La propuesta de sistema que se presenta como solución a dicha Mutual, es tener un sistema corriendo en un servidor en internet, administrado por una empresa que ofrezca dichos servicios, con una base de datos única, la cual también esté administrada por un servicio de bases de datos en la nube de internet, para que toda la información esté accesible en línea. Esto permitirá a las sucursales controlar sus números y a la comisión directiva poder tomar decisiones de manera muy rápida sobre los servicios en todas las sucursales.

La utilización de un sistema informático que disponga de información actualizada y en línea a través de internet, será aprovechada para minimizar los costos y mejorar la eficiencia, las relaciones con los asociados y con la comunidad; además de optimizar los procesos de los servicios.

La mejor utilización de los recursos tecnológicos, cumplimiento de los valores, principios y códigos de ética Mutual, asegura el mayor de los éxitos en la dirección de dicha entidad; a su vez, redundará en beneficio de la mutual y de la comunidad en la que actúe.

Este sistema mantendrá la contabilidad completamente en la nube de internet, permitiendo monitorear y tomar decisiones rápidas.

También cada comprobante que se genere por los cajeros estará en línea para ser auditado y la información registrada permanecerá sin borrarse para llevar un control seguro.

La información personal de cada asociado, así como el estado de los servicios que este recibirá, se mantendrán actualizados y disponibles en línea para cualquier persona del staff comercial, que la necesite para brindar los servicios.

Los ahorros que posea un asociado, al igual que los demás servicios, también se encontrarán en línea para que un asociado pueda gozar de realizar operaciones de depósitos o extracciones desde la sucursal que le sea más conveniente.

En resumen, mis servicios brindan la posibilidad a la mutual de contar con un sistema personalizado que se adapte a sus necesidades, minimizando costos y aumentando la cantidad de información en línea.

## *1.7.* Antecedentes a la solución

Existen diversas soluciones de software para la gestión de los servicios que ofrecen las mutuales. La mayoría de las aplicaciones poseen un sistema cliente-servidor con una base de datos por cada una de las sucursales. Aparejado a esto, la información de los asociados no queda disponible a otras sucursales de manera automática. Si un asociado se encuentra registrado en una sucursal, al viajar a otra, no podrá ser atendido, ya que su información no se encontrará disponible en la misma.

En cuanto a la información contable de los sistemas actuales, cada sucursal suele mantener almacenada en la base de datos local su propia contabilidad, así como también los mecanismos necesarios para conciliarla periódicamente, con las demás sucursales. Por ende, al no mantenerse en una base de datos central en la nube la contabilidad de cada sucursal, imposibilita realizar una toma de decisiones rápida.

A continuación, se describirán algunas soluciones existentes en el mercado.

- **SOFTWARE PARA MUTUALES SIGMA de GRUPO NEOSISTEMAS**

En su página web, se promueve como una innovación tecnológica para hacer crecer a una mutual. Sin embargo, este sistema es una solución cliente servidor donde cada sucursal de una mutual posee un servidor. De este modo, cada sucursal administra su propia base de datos, permitiendo conciliar sólo la información contable, no la información comercial y administrativa. También como otro punto en contra, podemos observar que toda la parametrización y configuración de los puestos de trabajo se hacen en cada computadora donde se necesite operar. Esto va en contra de la evolución que ofrecen los sistemas web, tales como la solución que se propone en este documento, donde la información y las aplicaciones sean accesibles desde cualquier plataforma, y solo se tenga que configurar los accesos a los distintos tipos de usuarios.

También desde la empresa Neosistemas, mencionan que poseen respaldo y confianza, pero el sistema posee una manera de realizar copias de seguridad de la información de la base de datos, que debe ser realizada manualmente por el cliente durante una hora determinada del día, pudiendo ocurrir que una falla mecánica de la electricidad o un

problema ocasionado con un virus pueda eliminar o corromper la información de la base de datos, y hasta la copia de seguridad anteriormente realizada.

En el sistema que se propone como solución en este Proyecto Final de Carrera ese no es un inconveniente, ya que se propone el servicio de bases de datos en internet administrado por una compañía que brinda respaldos de información automatizados y seguridad tanto contra virus, así como también ante interrupciones de electricidad que pueda corromper la información.

También desde Neosistemas, dicen que facilitan una mayor información para la toma de decisiones. La verdad es que, estudiando el funcionamiento del sistema, podemos observar que, posiblemente por ser una solución que con el correr de los años se fue actualizando para cumplir ciertas normas, posee muchos listados con información incompleta y sin un objetivo claro. Por otro lado, al no unificar la información de las sucursales para tener una visión global de los distintos sectores comerciales, la toma de decisiones se vuelve compleja, debiendo tener que recurrir a nuevos listados con otras herramientas para poder ir unificando criterios y poder así llegar a tener una visión global.

- **NEXA MUTUALES**

Software específico para asociaciones y mutuales.

En el mismo se encuentra volcado toda la experiencia para poder cumplir con las necesidades de información y para poder gestionar el universo de actividades dentro de los diversos tipos de entidades (asociativas, financieras, médicas, prestacionales, proveedurías, etc.).

El sistema cuenta por varios módulos parametrizables para cubrir la amplia variedad de servicios que ofrece una mutual. Entre los módulos encontramos el de Ayudas Económicas (Similar al de Préstamos del sistema SIGMA), Órdenes de compras, Nichos, Plazos Fijos, Módulos de estadísticas INAES/AFIP, Proveeduría/Manejo de mercadería, Turismo.

A diferencia de SIGMA, este sistema ya posee una versión que corre en la nube. Esto es una de las ventajas fundamentales por las cuales se decidió que el sistema a ser desarrollado se ejecute en la nube.

Entre las ventajas que menciona NEXA se encuentra el acceso desde cualquier sitio y con varios dispositivos. Los programas y archivos están en la nube, con lo que basta una conexión a Internet para acceder a ellos y usarlos de modo remoto. La copia de seguridad se

realiza automáticamente en forma diaria brindando más seguridad en el resguardo de los datos. Con la instalación en la nube se ahorra en recursos y gana en comodidad.

Este sistema cuenta con muchas de las ventajas que se mención como propuestas para desarrollar un sistema, pero cuenta también con una de las desventajas por las cuales se solicitó el desarrollo de un sistema nuevo: el costo mensual de la contratación del mismo.

En este sentido, la Mutual solicita el desarrollo porque se ahorraría el alquiler del sistema mensualmente costoso, pero además contar con la ventaja de ir desarrollando por etapas para ir automatizando los diferentes servicios. Entonces a medida que se avance con la puesta en producción de los primeros módulos, esto dará un panorama a los usuarios de las mejoras o personalizaciones que se requieran del sistema, para que mejor se adapten a la cultura de La Mutual.

## *1.8.* Metodología

La elección del modelo de desarrollo evolutivo en espiral se debe principalmente a que la organización no tiene un sistema informático. Por ello, existe un gran desconocimiento inicial de los cambios y riesgos que pueden surgir de “informatizar” una organización cuyos procedimientos actualmente son completamente manuales. Por lo tanto, con un modelo evolutivo la adaptación a los cambios introducidos a lo largo del desarrollo es más fácil.

La metodología de trabajo consiste en varios ciclos de desarrollo de software, donde cada ciclo consta de las siguientes etapas:

- Etapa 1: Determinación de objetivos, alternativas y restricciones.
- Etapa 2: Evaluación de alternativas e identificación y resolución de riesgos.
- Etapa 3: Desarrollo del siguiente nivel de producto.
- Etapa 4: Planificación del siguiente paso.

En el primer ciclo, se espera determinar los objetivos y límites del sistema, definiendo su estructura y funcionamiento para establecer qué es lo que debería hacer. Mediante la descripción de requerimientos funcionales y no funcionales se identificarán los elementos básicos y las relaciones entre sí y con el entorno. También se determinarán los riesgos y se hará una evaluación de estos para mitigarlos.

Por otro lado, se espera también obtener un conocimiento más profundo de la terminología del negocio, incorporar un análisis preliminar de requisitos, identificar y delimitar la funcionalidad esperada para el sistema, distinguir los usuarios y las relaciones de estos con el sistema, y se trabajará en identificar riesgos del proyecto y cómo mitigarlos. Al finalizar este ciclo, se espera obtener los siguientes resultados:

- Análisis preliminar de requisitos funcionales y no funcionales
- Modelo preliminar de casos de usos
- Glosario de términos
- Modelo del dominio de los principales elementos del sistema.

Como paso final se planificará el siguiente ciclo.

En un segundo ciclo, se refinará el modelo de casos de uso del ciclo anterior, dando como resultado una especificación de éstos, también se especificará y detallará el modelo de datos identificado anteriormente. Por otro lado, se estudiarán las herramientas más adecuadas para el desarrollo y para la puesta en producción del sistema. Por último, se especificará la arquitectura propuesta para el sistema. Luego se instalarán las herramientas principales de desarrollo y los entornos necesarios. El objetivo de este ciclo es obtener las siguientes especificaciones del sistema:

- Especificación de los Casos de uso.
- Especificación de la Arquitectura.
- Tecnología para utilizar en el proyecto.
- Diagrama de Bases de Datos.
- Diagrama preliminar de interfaces de usuario.

También se priorizarán los requerimientos en el tiempo, para planificar su implementación y obtener un plan preliminar de desarrollo de software. Como paso final se planificará el siguiente ciclo, donde se van a implementar los casos de uso seleccionados.

Los siguientes ciclos, serán de construcción de software. Como primer paso, se planificará la implementación, identificando los casos de uso a implementar en cada ciclo y los plazos propuestos, obteniendo un plan preliminar de desarrollo del software. Así se conseguirá obtener un primer, segundo y tercer prototipo de software, que incorpore mayor funcionalidad. En cada ciclo, se comenzará identificando cuáles son los riesgos que conlleva implementar los casos de usos seleccionados, para luego mitigarlos. Las implementaciones

resultantes de cada ciclo serán probadas por los usuarios finales en entornos de pruebas, para luego ser desplegadas en entornos productivos.

Se finalizará cada ciclo planificando el siguiente ciclo, donde se seleccionarán nuevos casos de uso para ir ampliando la funcionalidad del prototipo de software.

Se elaborarán manuales de usuario y ayudas del sistema.

Finalmente se instalará el sistema en el entorno previsto.

Se redactará el documento final para la correspondiente presentación del proyecto.

### *1.9.* Aportes que se espera realizar con este trabajo

Se espera que este proyecto contribuya a optimizar no sólo los recursos invertidos por la mutual en la gestión de sus procesos, sino también, la relación de la mutual con la sociedad, reduciendo tiempos de procesos de negocio y mejorando el acceso y distribución de la información.

Desde el punto de vista de la formación profesional, sin dudas el desarrollo de este sistema permitirá afianzar, aplicar y llevar a la práctica los conocimientos adquiridos en la carrera y los incorporados durante el desarrollo del proyecto.

# 2. Primer Ciclo del Proyecto

### *2.1.* Identificación de Riesgos

Se trabaja en factores que ponen en riesgo el proyecto en este primer ciclo. Como resultado del trabajo se logró obtener en conjunto con los dirigentes de La Mutual los siguientes riesgos:

- Desconocimiento de la terminología del negocio, sus procesos, y que es lo que los clientes esperan que del sistema.
- Al no tener un sistema informático, no se tiene un conocimiento claro de los requisitos tanto funcionales como no funcionales. Muchos de los problemas en la ingeniería de software (hablando sobre el proceso de desarrollo en sí mismo) comienzan con especificaciones de requisitos inexactas.

## *2.2.* Evaluación de los Riesgos

- Para comprender mejor el funcionamiento del negocio, se pautaron reuniones con los clientes, donde estos explicaron cómo funciona la entidad y cuáles son los procesos tanto administrativos, como comerciales. También se presenciaron jornadas de trabajo para comprender las explicaciones que fueron relevadas de los clientes. Como resultado, se comprendió con mayor claridad el soporte que tiene que brindar el sistema a dicha entidad, y se detalló un glosario de términos, el cual se encuentra en el "Anexo 4: Glosario de Términos"
- Para mitigar el problema en la identificación de los requisitos funcionales y no funcionales, debido a que los clientes no operan con un sistema informático, se realiza una primera identificación de los requisitos escrita, la cual es usada con los clientes para aclarar las necesidades que tienen éstos. Dicho trabajo dejó como resultado la identificación de los actores del sistema, los requisitos funcionales para cada uno de los actores, y un detalle de requisitos no funcionales esperados. Por último, se obtiene un diagrama de casos de uso donde se encuentran todas las funcionalidades identificadas.

## *2.3.* Actores

Se detallarán los actores que representan un conjunto de roles que juegan los usuarios que interactúan con el sistema:

**Usuario**: Representa el rol que ejecuta las funcionalidades comunes a todos los usuarios del sistema.

**Comercial**: Hereda todas las funcionalidades de Usuario y ejecuta las funcionalidades comunes a los empleados que interactúan con los socios de La Mutual para brindar algún servicio, a excepción de aquellas funciones del cajero.

**Cajero:** Hereda todas las funcionalidades de Usuario y ejecuta las funcionalidades comunes a los empleados que operan una caja de La Mutual, donde se producen los ingresos y egresos de dinero.

**Administrador:** Hereda las funcionalidades de Usuario, Comercial y Cajero, y como su nombre lo indica, desempeña las tareas de administración del sistema.

## *2.4.* Requerimientos del Sistema

### 2.4.1. Requerimientos Funcionales:

Se detallan los requisitos funcionales que los actores mencionados en el punto anterior deberán tener disponibles en el sistema:

**Usuario:**

- Identificarse y autenticarse con el sistema cada vez que desee iniciar sesión en el mismo y deberá poder modificar la contraseña del sistema cuando lo deseen.

**Comercial:**

- Dar de alta, baja y reincorporar a un socio.
- Modificar la información personal.
- Buscar un socio a través del apellido.
- Dar de alta una caja de ahorro.
- Buscar una caja de ahorros a través del apellido del titular
- Modificar tasas o responsables de una caja de ahorro.
- Realizar un depósito o extracción de la caja de ahorros.
- Anular un depósito o extracción de la caja de ahorros.
- Deshabilitar y habilitar caja de ahorros.

**Cajero:**

- Consultar información de la caja en el día de la fecha, que es saldo inicial y final y comprobantes cargados en el día.
- Registrar un comprobante.
- Anular un comprobante.

**Administrador:**

- Mostrar el Listado de Cuentas Contables.
- Mostrar los saldos Cuentas Contables mensuales.
- Alta Cuenta Contable.
- Modificar Cuenta Contable.
- Eliminar Cuenta Contable.

- Listar comprobantes confirmados en la caja entre dos fechas determinadas.
- Listar los movimientos de todas las cajas de ahorros entre dos fechas determinadas.

### 2.4.2. **Requerimientos No Funcionales:**

**Portabilidad**

El sistema diseñado y sus componentes deben ser portables entre distintas plataformas, de modo que se pueda cambiar las plataformas de los servidores en los que se aloja el sistema.

**Disponibilidad**

El sistema debe soportar una operación en alta disponibilidad, no debe presentar ningún punto de fallo, es decir, debe estar provisto de mecanismos o componentes que aseguren la continuidad del servicio, con procesamiento distribuido y almacenamiento en múltiples servidores. Por lo que al momento de realizar el diseño detallado se debe validar la arquitectura física en la que funcionará el sistema.

**Mantenibilidad**

Se debe documentar todo el proceso de desarrollo de software, en especial las especificaciones en la etapa de diseño, y llevar un control de versiones en la etapa de codificación del software. También se deberán codificar componentes de tal modo de aumentar la reusabilidad y la modularidad.

De este modo, se mejorará la capacidad del producto de software para ser modificado efectiva y eficientemente, debido a necesidades evolutivas, correctivas o perfectivas.

**Seguridad**

Todas las comunicaciones externas entre los servidores de datos, la aplicación y el cliente del sistema deben estar cifradas. El sistema controlará el acceso a usuarios registrados en las bases de datos. Una vez que un usuario inicia sesión, tendrá un rol, el cual determinará qué acciones tiene permitido realizar; además el sistema controlará qué datos son visibles para ese usuario y para ese rol.

**Fiabilidad**

Se deberá minimizar al máximo la presencia de fallos futuros. Pero cuando estos ocurran, se deberá contar con mecanismos de recuperación de datos para garantizar que no haya pérdida de información importante. Para ello se contará con un sistema que mantenga la consistencia transaccional.

## *2.5.* Diagrama preliminar de Casos de Uso

Luego de un análisis exhaustivo pero preliminar de los requisitos funcionales, se llegó a la conclusión de delimitar el sistema en los siguientes casos de usos, que cubren todas las funcionalidades planteadas en la sección anterior.

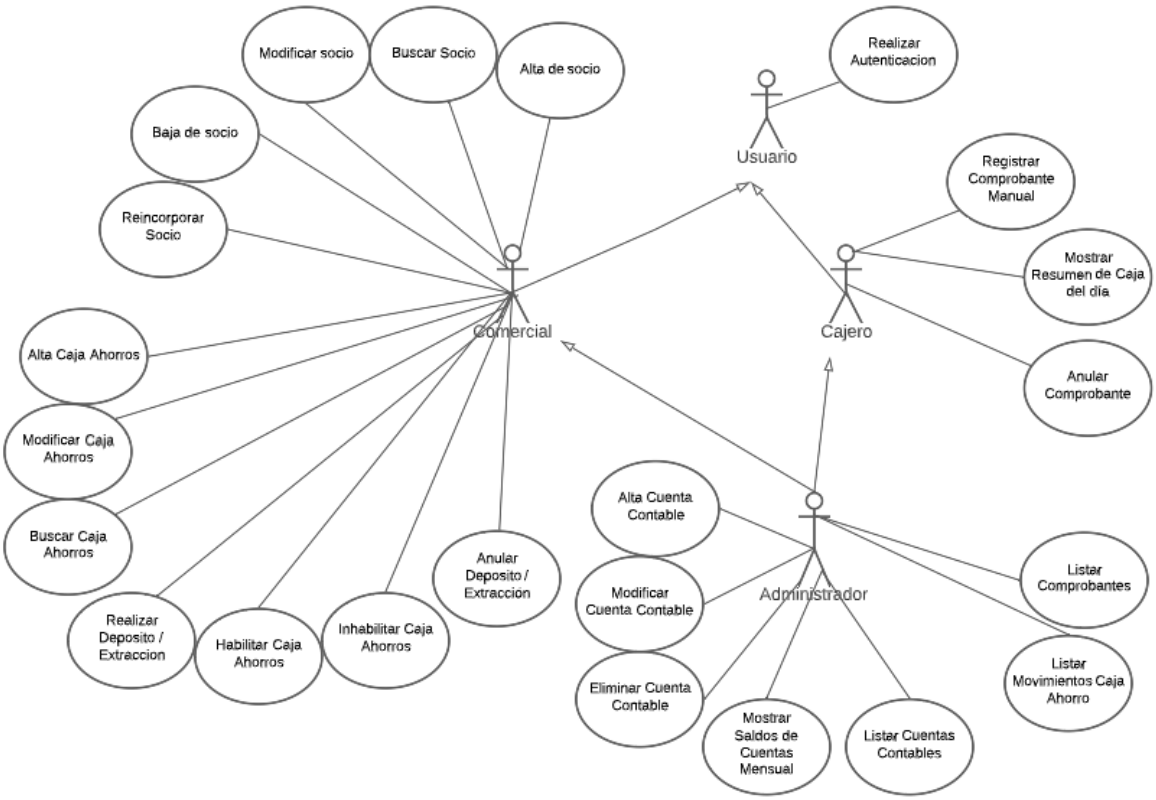



**Figura 1: Diagrama de casos de uso.**

## *2.6.* Modelo del Dominio

Se identificaron los principales elementos que intervienen en el sistema y sus relaciones, dando como resultado el siguiente modelo del dominio.

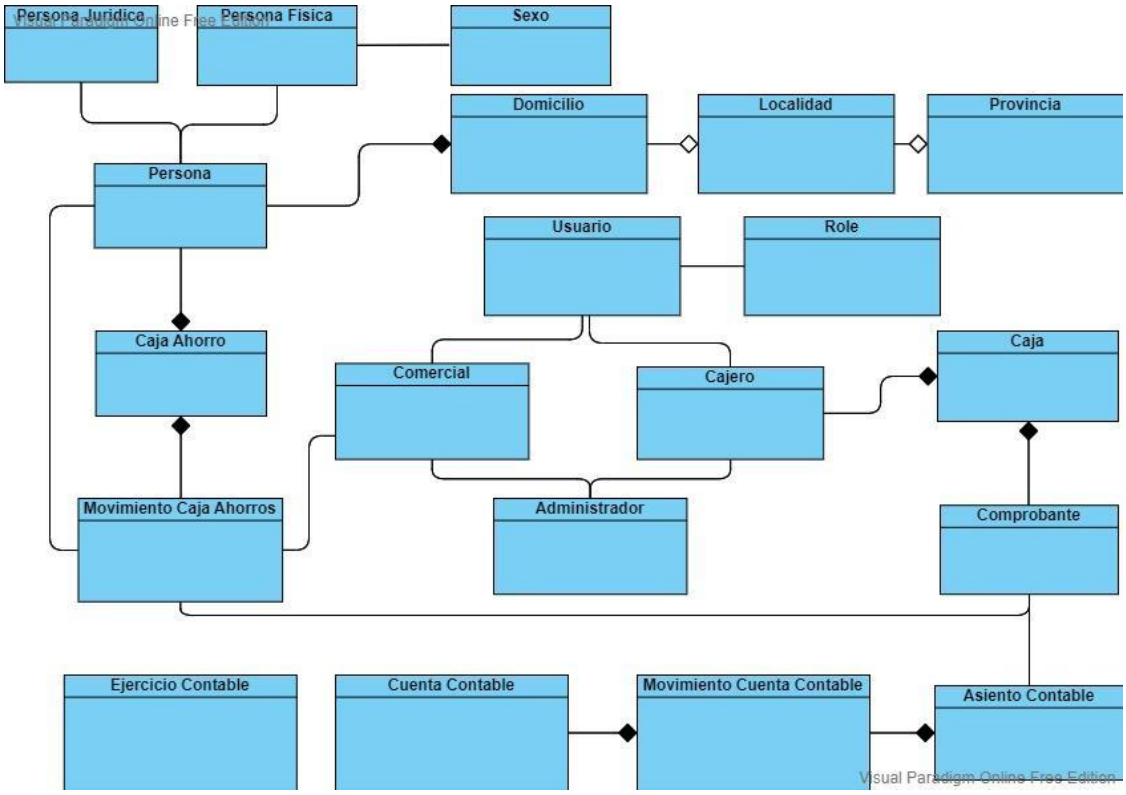



**Figura 2: Modelo del Dominio.**

Un socio es el cliente que posee la mutual y recibe los servicios y beneficios por ser socio de ésta. Aquí se decidió representar al socio como una **persona**, que se especializa en una **persona física** o **persona jurídica**.

Una **persona** estará asociada a un **domicilio**, el cual pertenecerá a una **localidad**, donde esta última formará parte de una **provincia**.

Una **caja de ahorros** estará compuesta de uno o más socios (personas), que serán los responsables de los **movimientos de caja de ahorros** que se realicen en la misma. A su vez, un **movimiento de caja de ahorros** pertenecerá a una sola caja de ahorros, y estará será generada por un u**suario** con el **rol** comercial.

Un **movimiento de caja de ahorros** será un tipo de **comprobante** que impactará en la caja de la mutual.

Un **comprobante** será confirmado (validado en la caja) por un **usuario** con el **rol** de Cajero, que operará una caja de la mutual.

Un **comprobante** estará asociado a un **asiento contable**.

Un **asiento contable** estará compuesto de **movimientos de cuentas contables**. Cada **movimiento de cuenta contable** estará asociado con una y solo una **cuenta contable**.

Los **comprobantes** serán validados en la caja por un **usuario** con el **rol** de cajero y que será el responsable de controlar el ingreso y egreso del dinero. También los usuarios con dicho rol podrán anular un comprobante.

Un **usuario** con el **rol** de administrador además de tener todos los permisos que poseen los roles comerciales y cajeros, representará a la persona con los permisos necesarios, para listar y controlar movimientos de cajas de ahorros y comprobantes de cajas de distintas fechas.

# 3. Segundo Ciclo

## *3.1.* Identificación de Riesgos

Se trabaja en factores que ponen en riesgo el proyecto en este segundo ciclo. Como resultado del trabajo logramos obtener los siguientes riesgos:

- Poca experiencia en el uso de las herramientas que intervienen en el desarrollo del software y en la puesta en producción.
- Escasa información de interacción esperada entre el sistema y los usuarios finales y la apariencia y usos de sus pantallas o interfaces.
- Diseño inadecuado de la arquitectura del sistema.

## *3.2.* Evaluación de los Riesgos

- Para mitigar el riesgo de la falta de experiencia en el uso de herramientas en el desarrollo de software, se trabajó en la investigación y elección de las herramientas que se consideraron más adecuadas.
- Para mitigar el riesgo de la escasa información de interacción entre el sistema y los usuarios finales, se realizó una especificación del modelo de casos de uso para describir en detalle el flujo de la interacción que existe entre los usuarios y el sistema. También se realizó un diseño preliminar de interfaces de usuarios, de este modo poder acordar con la mutual las pantallas que poseerá el sistema y las funcionalidades a las que dará soporte este último.

- Para mitigar el riesgo de la arquitectura inadecuada, se trabajó en visualizar y describir cómo será la arquitectura del sistema final para que el sistema funcione correctamente de acuerdo con los requisitos.

## *3.3.* Especificación de Casos de usos

En el ciclo anterior, se delimitaron las funcionalidades que cubrirá el sistema a través de su modelado en un diagrama preliminar de casos de usos. En este ciclo, se refinará la especificación detallando la secuencia de pasos que ocurre en la ejecución de éstos. En el Anexo "2: Especificación de Casos de Usos" se encuentra el resultado de este trabajo.

## *3.4.* Tecnología por utilizar en el proyecto

En este apartado explicarán las tecnologías que fueron elegidas y utilizadas tanto en las etapas de desarrollo del proyecto, como así también las utilizadas para la implementación del sistema. Como el sistema web es un cliente-servidor, primero se va a explicar la tecnología utilizada para el Frontend y luego para el Backend.

### 3.4.1. Para el Frontend se utilizaron las siguientes tecnologías:

#### 3.4.1.1. HTML

HTML (Lenguaje de marcado de hipertexto o HyperText Markup Language por sus siglas en inglés) es un lenguaje descriptivo que especifica la estructura de las páginas web.

Está compuesto por una serie de etiquetas que el navegador interpreta y da forma en la pantalla. HTML dispone de etiquetas para imágenes, hipervínculos que nos permiten dirigirnos a otras páginas, saltos de línea, listas, tablas, etc.

Es un estándar a cargo del World Wide Web Consortium (W3C) o Consorcio WWW, organización dedicada a la estandarización de casi todas las tecnologías ligadas a la web, sobre todo en lo referente a su escritura e interpretación. Es el estándar que se ha impuesto en la visualización de páginas web y es el que todos los navegadores actuales han adoptado.

HTML permite incorporar otros lenguajes para definir el formato con el que se tienen que presentar las webs, como CSS.

Actualmente HTML se encuentra en su versión HTML5 (HyperText Markup Language, versión 5), que es la quinta revisión importante del lenguaje básico de la World Wide Web, HTML. HTML5 establece una serie de nuevos elementos y atributos que reflejan el uso típico de los sitios web modernos.

HTML5 no sólo es una nueva versión del lenguaje de HTML, es también un grupo de especificaciones para el desarrollo de web. Se ha convertido en un sistema que permite dar formato y diseño a los layout de los sitios web.

#### 3.4.1.2. CSS

Hojas de estilo en cascada (CSS) es un mecanismo simple para agregar estilo (por ejemplo, fuentes, colores, espaciado) a los documentos Web.

CSS es uno de los lenguajes base de la Open Web y posee una especificación estandarizada por parte del W3C.

Es muy usado para establecer el diseño visual de los documentos web e interfaces de usuario escritas en HTML o XHTML; el lenguaje puede ser aplicado a cualquier documento XML. Junto con HTML y JavaScript, CSS es una tecnología usada por muchos sitios web para crear páginas visualmente atractivas.

CSS está diseñado principalmente para marcar la separación del contenido del documento y la forma de presentación de este, características tales como las capas o layouts, los colores y las fuentes.4 Esta separación busca mejorar la accesibilidad del documento, proveer más flexibilidad y control en la especificación de características presentacionales, permitir que varios documentos HTML compartan un mismo estilo usando una sola hoja de estilos separada en un archivo .css, y reducir la complejidad y la repetición de código en la estructura del documento.

La especificación CSS describe un esquema prioritario para determinar qué reglas de estilo se aplican si más de una regla coincide para un elemento en particular. Estas reglas son aplicadas con un sistema llamado de cascada, de modo que las prioridades son calculadas y asignadas a las reglas, así que los resultados son predecibles.

#### 3.4.1.3. Javascript y Typescript

JavaScript (también conocido como ECMAScript) comenzó su vida como un lenguaje de programación simple para navegadores. En el momento en que se inventó, se esperaba que se usará para pequeños fragmentos de código incrustados en una página web; escribir más de unas pocas docenas de líneas de código habría sido algo inusual. Debido a esto, los primeros navegadores web ejecutaban dicho código con bastante lentitud. Sin embargo, con el tiempo, JS se hizo cada vez más popular y los desarrolladores web comenzaron a usarlo para crear experiencias interactivas.

Los desarrolladores de navegadores web respondieron a este mayor uso de JS optimizando sus motores de ejecución (compilación dinámica) y ampliando lo que se podía hacer con ellos (agregando API), lo que a su vez hizo que los desarrolladores web lo usaran aún más. En los sitios web modernos, su navegador ejecuta con frecuencia aplicaciones que abarcan cientos de miles de líneas de código. Se trata de un crecimiento prolongado y gradual de "la web", que comienza como una simple red de páginas estáticas y evoluciona hacia una plataforma para *aplicaciones* ricas de todo tipo.

Más que esto, JS se ha vuelto lo suficientemente popular como para usarse fuera del contexto de los navegadores, como la implementación de servidores JS usando node.js. La naturaleza de "ejecutar en cualquier lugar" de JS lo convierte en una opción atractiva para el desarrollo multiplataforma.

En resumen, tenemos un lenguaje que fue diseñado para usos rápidos y luego se convirtió en una herramienta completa para escribir aplicaciones con millones de líneas.

TypeScript es un lenguaje que es un superconjunto de JavaScript: la sintaxis JS es, por lo tanto, correcta en TS. La sintaxis se refiere a la forma en que escribimos texto para formar un programa. Al ser un superconjunto, significa que agrega definiciones de tipos estáticos y reglas sobre cómo se pueden usar diferentes tipos de valores.

TypeScript tiene una relación inusual con JavaScript. TypeScript ofrece todas las funciones de JavaScript y una capa adicional además de estas: el sistema de tipos de TypeScript.

Los tipos proporcionan una forma de describir la forma de un objeto, proporcionando una mejor documentación y permitiendo que TypeScript valide que su código está funcionando correctamente.

Los tipos pueden ser opcionales en TypeScript, pero la inferencia de tipos le permite obtener mucha potencia sin escribir código adicional.

TypeScript no considera que ningún código JavaScript sea un error debido a su sintaxis. Esto significa que puede tomar cualquier código JavaScript que funcione y colocarlo en un archivo TypeScript sin preocuparse por cómo está escrito exactamente. De esta manera, se garantiza que se ejecutará de la misma manera, incluso si TypeScript piensa que el código tiene errores de tipo.

El principal beneficio de TypeScript es que puede resaltar comportamientos inesperados en su código, lo que reduce la posibilidad de errores.

#### 3.4.1.4. Angular

Angular es una plataforma y un framework Javascript para la creación de páginas web SPA (Single Page Application) usando HTML y TypeScript. A su vez, que una web sea SPA quiere decir que cuando el usuario entra en la web se carga todo el contenido de todas las páginas a la vez. Esto quiere decir que la primera carga nada más entrar es más lenta pero luego los cambios entre páginas son instantáneos. Además, solo se utiliza un fichero con el código de la web por lo que no se requiere de muchas llamadas al servidor.

Otra de las características de Angular es que utiliza Typescript, un lenguaje que luego se traduce a Javascript y que sirve para tener tipado de variables, interfaces, inyección de dependencias y más cosas.

Una de los objetivos de Angular es fortalecer la estructura MVC en el desarrollo web y las SPA.

La arquitectura de una aplicación en Angular se basa en ciertos conceptos fundamentales. Los bloques de construcción básicos son los NgModules, que proporcionan un contexto de compilación para los componentes. Los NgModules recopilan código relacionado en conjuntos funcionales; una aplicación de Angular se define por un conjunto de NgModules. Una aplicación siempre tiene al menos un módulo raíz que permite el arranque y generalmente tiene muchos más módulos de funcionalidad.

Los componentes definen vistas, que son conjuntos de elementos de la pantalla que Angular puede elegir y modificar de acuerdo con la lógica y los datos de tu programa.

Los componentes usan servicios, los cuales proporcionan una funcionalidad específica que no está directamente relacionada con las vistas. Los proveedores de servicios pueden inyectarse en componentes como dependencias, haciendo que tu código sea modular, reutilizable y eficiente.

Los módulos, componentes y servicios son clases que usan decoradores. Estos decoradores indican su tipo y proporcionan metadatos que le indican a Angular cómo usarlos.

Los metadatos para una clase componente son asociados con una plantilla que define una vista. Una plantilla combina HTML ordinario con directivas de Angular y enlace markup que permiten a Angular modificar el HTML antes de mostrarlo para su visualización.

Los metadatos para una clase servicio proporcionan la información que Angular necesita para que esté disponible para los componentes a través de la Inyección de Dependencia (ID).

Los componentes de una aplicación suelen definir muchas vistas, ordenadas jerárquicamente. Angular proporciona el servicio Router para ayudarte a definir rutas de navegación entre vistas. El enrutador proporciona capacidades de navegación sofisticadas en el navegador.

Ventajas de usar Angular

- Lenguaje Typescript, tiene una sintaxis muy parecida a Java, con tipado estático. Esto te va a ayudar a que tu código tenga menos errores.
- Sigue el patrón MVC (Modelo - Vista - Controlador), con la vista separada de la lógica de negocio.
- Basado en componentes, es decir, piezas de código con vista que puedes reutilizar en otras páginas.
- Comunidad muy grande con multitud de tutoriales y librerías.
- Inyección de dependencias, un patrón de diseño que se basa en pasar las dependencias directamente a los objetos en lugar de crearlas localmente.
- Programación reactiva, la vista se actualiza automáticamente a los cambios en las variables.
- Dispone de asistente por línea de comandos para crear proyectos base (Angular cli).
- Se integra bien con herramientas de testing.
- Buena integración con Ionic, para adaptar aplicaciones web a dispositivos móviles.
- Framework completo. No necesitas instalar demasiadas librerías externas para hacer cosas comunes.

#### 3.4.1.5. Firebase

Es un servicio de Google que nos proporciona un backend ya listo para el desarrollo de aplicaciones web y apps para dispositivos. Con Firebase podemos realizar un desarrollo acelerado de aplicaciones, ya que no necesitamos desarrollar la parte del servidor. Este tipo de servicios de computación en la nube se conoce como BaaS (Backend as a Service) en el

que la tarea principal para el desarrollo backend será la configuración, en vez de la programación.

Este servicio "Backend as a Service" nos ofrece varias piezas que vamos a describir y usar en el manual, necesarias para cualquier tipo de aplicación. Te las ofrecen listas para que solo tengas que configurar y comenzar a usar, de modo que el desarrollo de aplicaciones se realiza de una manera mucho más rápida.

Ofrece numerosos servicios, entre los que se encuentra una base de datos en tiempo real, sistema de Login de usuarios, notificaciones, almacenamiento de archivos, hosting, analítica, etc.

Es compatible con la web, mediante programación Javascript y con numerosas librerías para trabajar en sistemas diversos como AngularJS, React o Polymer. Pero también es compatible con dispositivos iOS y Android, así como -por medio de un API REST- en cualquier otra plataforma o tecnología que deseemos trabajar.

Esta plataforma, está integrada con Google Cloud Platform, que usa un conjunto de herramientas para la creación y sincronización de proyectos que serán dotados de alta calidad, haciendo posible el crecimiento del número de usuarios y dando resultado también a la obtención de una mayor monetización.

Permite sincronizar fácilmente los datos de sus proyectos sin tener que administrar conexiones o escribir lógica de sincronización compleja.

Usa la infraestructura de Google y escala automáticamente para cualquier tipo de aplicación, desde las más pequeñas hasta las más potentes.

Crea proyectos sin necesidad de un servidor: Las herramientas se incluyen en los SDK para los dispositivos móviles y web, por lo que no es necesario la creación de un servidor para el proyecto.

Firebase dota a sus usuarios de una gran documentación para crear aplicaciones usando esta plataforma. Aparte de esto, ofrece soporte gratuito mediante correo electrónico para todos sus usuarios, y además sus desarrolladores participan activamente en plataformas como Github y StackOverflow, así como poseen un canal de Youtube explicando el funcionamiento de varias de sus herramientas.

### 3.4.2. Para el Backend se utilizaron las siguientes tecnologías:

#### 3.4.2.1. Java

Java es un lenguaje de programación y una plataforma informática que fue comercializada por primera vez en 1995 por Sun Microsystems. Hay muchas aplicaciones y sitios web que no funcionarán, probablemente, a menos que tengan Java instalado y cada día se crean más. Java es rápido, seguro y fiable. Desde ordenadores portátiles hasta centros de datos, desde consolas para juegos hasta computadoras avanzadas, desde teléfonos móviles hasta Internet, Java está en todas partes, si es ejecutado en una plataforma no tiene que ser recompilado para correr en otra. Java es, a partir de 2012, uno de los lenguajes de programación más populares en uso, particularmente para aplicaciones de cliente-servidor de web, con unos diez millones de usuarios.

Su sintaxis deriva en gran medida de C y C++, pero tiene menos utilidades de bajo nivel que cualquiera de ellos. Las aplicaciones de Java son compiladas a bytecode (clase Java), que puede ejecutarse en cualquier máquina virtual Java (JVM) sin importar la arquitectura de la computadora subyacente reportados.

El lenguaje Java se creó con cinco objetivos principales:

- Debería usar el paradigma de la programación orientada a objetos.
- Debería permitir la ejecución de un mismo programa en múltiples sistemas operativos.
- Debería incluir por defecto soporte para trabajo en red.
- Debería diseñarse para ejecutar código en sistemas remotos de forma segura.
- Debería ser fácil de usar y tomar lo mejor de otros lenguajes orientados a objetos, como C++.

La primera característica, orientado a objetos (“OO”), se refiere a un método de programación y al diseño del lenguaje. Aunque hay muchas interpretaciones para OO, una primera idea es diseñar el software de forma que los distintos tipos de datos que usen estén unidos a sus operaciones. Así, los datos y el código (funciones o métodos) se combinan en entidades llamadas objetos. Otra de las grandes promesas de la programación orientada a objetos es la creación de entidades más genéricas (objetos) que permitan la reutilización del software entre proyectos, una de las premisas fundamentales de la Ingeniería del Software.

La segunda característica, la independencia de la plataforma, significa que programas escritos en el lenguaje Java pueden ejecutarse igualmente en cualquier tipo de hardware. Este es el significado de ser capaz de escribir un programa una vez y que pueda ejecutarse en cualquier dispositivo, tal como reza el axioma de Java, "write once, run anywhere".

Para ello, se compila el código fuente escrito en lenguaje Java, para generar un código conocido como “bytecode” (específicamente Java bytecode), instrucciones máquina simplificadas específicas de la plataforma Java. Esta pieza está “a medio camino” entre el código fuente y el código máquina que entiende el dispositivo destino. El bytecode es ejecutado entonces en la máquina virtual (JVM), un programa escrito en código nativo de la plataforma destino (que es el que entiende su hardware), que interpreta y ejecuta el código. Además, se suministran bibliotecas adicionales para acceder a las características de cada dispositivo (como los gráficos, ejecución mediante hebras o threads, la interfaz de red) de forma unificada. Se debe tener presente que, aunque hay una etapa explícita de compilación, el bytecode generado es interpretado o convertido a instrucciones máquina del código nativo por el compilador JIT (Just In Time).

#### 3.4.2.2. PostgreSQL

PostgreSQL es un poderoso sistema de base de datos relacional de objetos de código abierto que usa y extiende el lenguaje SQL combinado con muchas características que almacenan y escalan de manera segura las cargas de trabajo de datos más complicadas. Los orígenes de PostgreSQL se remontan a 1986 como parte del proyecto POSTGRES en la Universidad de California en Berkeley y tiene más de 30 años de desarrollo activo en la plataforma central.

PostgreSQL se ha ganado una sólida reputación por su arquitectura probada, confiabilidad, integridad de datos, conjunto de características robustas, extensibilidad y la dedicación de la comunidad de código abierto detrás del software para ofrecer soluciones innovadoras y de alto rendimiento de manera consistente. PostgreSQL se ejecuta en todos los principales sistemas operativos, cumple con ACID desde 2001 y tiene complementos potentes como el popular extensor de base de datos geoespacial PostGIS. No es de extrañar que PostgreSQL se haya convertido en la base de datos relacional de código abierto elegida por muchas personas y organizaciones.

PostgreSQL viene con muchas características destinadas a ayudar a los desarrolladores a crear aplicaciones, a los administradores a proteger la integridad de los datos y a crear

entornos tolerantes a fallas, y a administrar sus datos sin importar cuán grande o pequeño sea el conjunto de datos. Además de ser gratuito y de código abierto, PostgreSQL es altamente extensible. Por ejemplo, puede definir sus propios tipos de datos, crear funciones personalizadas, ¡incluso escribir código desde diferentes lenguajes de programación sin volver a compilar su base de datos!

PostgreSQL intenta ajustarse al estándar SQL donde tal conformidad no contradice las características tradicionales o podría conducir a malas decisiones de arquitectura. Se admiten muchas de las características requeridas por el estándar SQL, aunque a veces con una sintaxis o función ligeramente diferente. Se pueden esperar más avances hacia la conformidad con el tiempo. A partir del lanzamiento de la versión 14 en septiembre de 2021, PostgreSQL cumple con al menos 170 de las 179 características obligatorias para la conformidad con SQL: 2016 Core. En el momento de redactar este documento, ninguna base de datos relacional cumple plenamente con este estándar.

A continuación, se muestra una lista inagotable de varias características que se encuentran en PostgreSQL, y se agregan más en cada versión importante:

Tipos de datos

- Primitivas: Entero, Numérico, Cadena, Booleano.
- Estructurado: fecha / hora, matriz, rango / rango múltiple, UUID.
- Documento: JSON / JSONB, XML, valor clave (Hstore).
- Geometría: punto, línea, círculo, polígono.
- Personalizaciones: compuestos, tipos personalizados.

Integridad de los datos

- ÚNICO, NO NULO.
- Claves primarias.
- Llaves extranjeras.
- Restricciones de exclusión.
- Cerraduras explícitas, cerraduras de aviso.

Simultaneidad, rendimiento

- Indexación: árbol B, multicolumna, expresiones, parcial.
- Indexación avanzada: GiST, SP-Gist, KNN Gist, GIN, BRIN, índices de cobertura, filtros Bloom.

- Planificador / optimizador de consultas sofisticado, escaneos de solo índice, estadísticas de varias columnas.
- Transacciones, transacciones anidadas (a través de puntos de guardado).
- Control de concurrencia de múltiples versiones (MVCC).
- Paralelización de consultas de lectura y creación de índices de árbol B.
- Partición de la mesa.
- Todos los niveles de aislamiento de transacciones definidos en el estándar SQL, incluido Serializable.
- Compilación de expresiones Just-in-Time (JIT).

Fiabilidad, recuperación ante desastres

- Registro de escritura anticipada (WAL).
- Replicación: asíncrona, síncrona, lógica.
- Recuperación en un momento determinado (PITR), espera activa.
- Espacios de tabla.

Seguridad

- Autenticación: GSSAPI, SSPI, LDAP, SCRAM-SHA-256, Certificado y más.
- Sistema de control de acceso robusto.
- Seguridad a nivel de columnas y filas.
- Autenticación multifactor con certificados y un método adicional.

Extensibilidad

- Funciones y procedimientos almacenados.
- Lenguajes de procedimiento: PL / PGSQL, Perl, Python (y muchos más).
- Expresiones de ruta SQL / JSON.
- Envoltorios de datos externos: conéctese a otras bases de datos o flujos con una interfaz SQL estándar.
- Interfaz de almacenamiento personalizable para mesas.
- Muchas extensiones que brindan funcionalidad adicional, incluido PostGIS.

Internacionalización, búsqueda de texto

- Soporte para juegos de caracteres internacionales, por ejemplo, a través de intercalaciones de UCI.
- Intercalaciones que no distinguen entre mayúsculas y minúsculas ni acentos.

- Búsqueda de texto completo.

Hay muchas más características que puede descubrir en la documentación de PostgreSQL. Además, PostgreSQL es altamente extensible: muchas características, como los índices, tienen API definidas para que pueda construir con PostgreSQL para resolver sus desafíos.

Se ha demostrado que PostgreSQL es altamente escalable tanto en la gran cantidad de datos que puede administrar como en la cantidad de usuarios simultáneos que puede acomodar. Hay clústeres activos de PostgreSQL en entornos de producción que administran muchos terabytes de datos y sistemas especializados que administran petabytes.

#### 3.4.2.3. Maven

Maven tiene como objetivo recopilar los principios actuales para el desarrollo de mejores prácticas y facilitar la orientación de un proyecto en esa dirección.

Por ejemplo, la especificación, ejecución y generación de informes de pruebas unitarias son parte del ciclo de compilación normal con Maven. Las mejores prácticas actuales de pruebas unitarias se utilizaron como directrices:

Mantener el código fuente de prueba en un árbol fuente separado pero paralelo

Uso de convenciones de nomenclatura de casos de prueba para ubicar y ejecutar pruebas

Hacer que los casos de prueba configuren su entorno en lugar de personalizar la compilación para la preparación de la prueba

Maven también ayuda en el flujo de trabajo del proyecto, como la liberación y la gestión de problemas.

Maven también sugiere algunas pautas sobre cómo diseñar la estructura de directorios de su proyecto. Una vez que aprenda el diseño, puede navegar fácilmente por otros proyectos que usan Maven.

Si bien adopta un enfoque obstinado del diseño del proyecto, es posible que algunos proyectos no se ajusten a esta estructura por razones históricas. Si bien Maven está diseñado para ser flexible a las necesidades de diferentes proyectos, no puede satisfacer todas las situaciones sin comprometer sus objetivos.

Si su proyecto tiene una estructura de construcción inusual que no se puede reorganizar, es posible que deba renunciar a algunas funciones o al uso de Maven por completo.

La filosofía general de Maven es la estandarización de las construcciones generadas por seguir el principio de Convención sobre Configuración, a fin de utilizar modelos existentes en la producción de software.

Maven está construido alrededor de la idea de reutilización, y más específicamente, a la reutilización de la lógica de construcción. Como los proyectos generalmente se construyen en patrones similares, una elección lógica podría ser reutilizar los procesos de construcción. La principal idea es no reutilizar el código o funcionalidad (como Apache Ant), sino simplemente cambiar la configuración o también código escrito. Esa es la principal diferencia entre Apache Ant y Apache Maven: el primero es una librería de utilidades y funciones buenas y útiles, mientras que la otra es un framework configurable y altamente extensible.

#### 3.4.2.4. Spring Framework y Spring Boot

Spring Framework es un framework Open Source que facilita la creación de aplicaciones de todo tipo en Java, Kotlin y Groovy.

Si bien es cierto que, por lo que es más conocido es por la inyección de dependencias, Spring Framework está dividido en diversos módulos que podemos utilizar, ofreciéndonos muchas más funcionalidades:

- Core container: proporciona inyección de dependencias e inversión de control.
- Web: nos permite crear controladores Web, tanto de vistas MVC como aplicaciones REST.
- Acceso a datos: abstracciones sobre JDBC, ORMs como Hibernate, sistemas OXM (Object XML Mappers), JSM y transacciones.
- Programación orientada a Aspectos (AOP): ofrece el soporte para aspectos.
- Instrumentación: proporciona soporte para la instrumentación de clases.
- Pruebas de código: contiene un framework de testing, con soporte para JUnit y TestNG y todo lo necesario para probar los mecanismos de Spring.

Estos módulos son opcionales, por lo que podemos utilizar los que necesitemos sin tener que llenar nuestro classpath con clases que no vamos a usar.

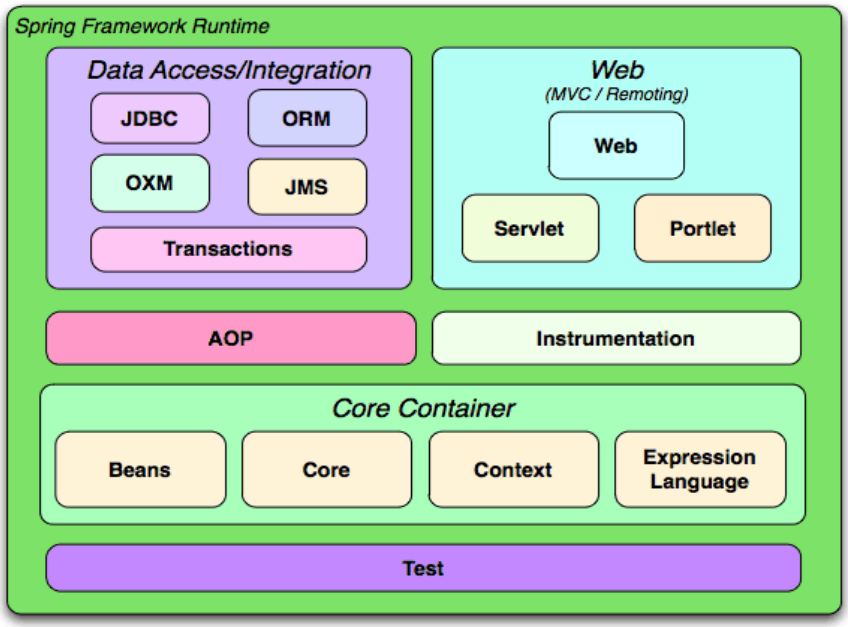



**Figura 3: Arquitectura de Sring Framework**

El uso de inyección de dependencias facilita la programación contra interfaz, permitiendo a los distintos componentes depender únicamente de interfaces y produciendo así un código menos acoplado.

Hay cientos de tecnologías que Spring permite integrar. Desde bibliotecas que implementan opentracing hasta las que nos generan métricas para nuestra aplicación, pasando por serialización/deserialización a JSON y XML, seguridad con OAuth2 o programación reactiva entre otras.

En general, Spring aumenta la productividad y reduce la fricción al ofrecernos abstracciones sobre implementaciones de tecnologías concretas. Un ejemplo claro es el de spring-data, que nos permite definir el acceso a base de datos con interfaces Java. Esto lo consigue parseando el nombre de los métodos y generando la consulta con la sintaxis específica para el driver que utilicemos. Por ejemplo, cambiar nuestra aplicación de MySQL a PostgreSQL es tan sencillo como cambiar el driver: Spring se encarga de la sintaxis de forma transparente.

Por lo general, Spring no obliga a implementar ni extender nada, lo que nos permite escribir código que es "agnóstico" del framework. De esta forma, desarrolladores con cero o muy poco conocimiento de Spring pueden realizar su trabajo sin mayores complicaciones.

A pesar de "la magia de Spring", como muchos lo llaman, Spring nos permite desactivar estos "comportamientos mágicos" en caso de ser necesario, por lo que podemos tomar el control cuando necesitemos más granularidad.

Si bien es cierto que Spring Framework es muy potente, la configuración inicial y la preparación de las aplicaciones para producción son tareas bastante tediosas. Spring Boot simplifica el proceso al máximo gracias a sus dos principales mecanismos.

Spring Boot es una de las tecnologías dentro del mundo de Spring, que nace con la idea de simplificar el proceso de construcción de aplicaciones con el framework Spring.

Generalmente el proceso de construcción consta de 3 pasos:

1. Seleccionar los jars con Maven
2. Crear la aplicación
3. Desplegar en el servidor

Fundamentalmente existen tres pasos a realizar. El primero es crear un proyecto Maven/Gradle y descargar las dependencias necesarias. En segundo lugar, desarrollamos la aplicación y en tercer lugar la desplegamos en un servidor. Si nos ponemos a pensar un poco a detalle en el tema, únicamente el paso dos es una tarea de desarrollo. Los otros pasos están más orientados a infraestructura. No deberíamos tener que estar eligiendo continuamente las dependencias y el servidor de despliegue.

SpringBoot nace con la intención de simplificar los pasos 1 y 3 y que nos podamos centrar en el desarrollo de nuestra aplicación.

#### 3.4.2.5. Heroku

Heroku es una plataforma en la nube como un servicio (PaaS), que permite a las empresas construir, entregar, supervisar aplicaciones y alojarlas en la nube.

El hecho de que sea una plataforma en la nube significa que como desarrolladores no nos tenemos que preocupar por la infraestructura, sino que solamente nos tenemos que centrar en el desarrollo de la aplicación, lo que nos evita todos los problemas que puede suponer llevar nuestra idea a la URL.

Heroku a diferencia de otras plataformas permite desarrollar prácticamente con cualquier lenguaje de programación: Ruby, Java, PHP, NodeJS, etc.

También permite desplegar versiones, hacer rollback, gestionar dependencias, entre otras acciones.

Dispone los denominados add ons, gracias a los que podemos añadir funcionalidad extra a nuestras aplicaciones de forma realmente sencilla, por ejemplo, memcached, redis, postgres, mongolab, etc.

Heroku utiliza contenedores de Linux independientes personalizados, que proveen capacidad de cómputo dentro de la plataforma. Estos componentes son denominados Dynos y son parte fundamental del modelo de arquitectura de Heroku.

Cada Dyno está aislado del resto, por lo que los comandos que se ejecutan y los archivos que se almacenan en un Dyno, no afectan a los otros. Además, cada Dyno provee el ambiente requerido por las aplicaciones para ser ejecutadas.

Los posibles comandos para ejecutar en los dynos incluyen procesos web, o cualquier otro tipo de proceso definido en el archivo Procfile de la aplicación. Este es un archivo de texto ubicado en el directorio raíz de la aplicación, y es el mecanismo provisto para la declaración de comandos que luego correrán los dynos. Básicamente, consiste en una lista de tipos de procesos de la aplicación. Cada tipo de procesos constituye una declaración de un comando.

Principales características:

- Heroku es gratuito para aplicaciones de poco consumo.
- Elasticidad y crecimiento. La cantidad de Dynos asignados a una aplicación se puede cambiar en cualquier momento a través de la línea de comandos o el dashboard.
- Tamaño. Heroku ofrece diferentes tipos de dynos, cada uno con diferentes capacidades de procesamiento y memoria.
- Routing. Internamente los routers realizan un seguimiento de la ubicación de los Dynos que estén corriendo, y redirigen el tráfico de acuerdo a la misma.
- Seguimiento. Existe un manejador de Dynos, el cual monitorea de forma continua los dynos que se estén ejecutando. En caso de una falla en un Dyno, este es eliminado y creado nuevamente.
- Distribución y redundancia. Los Dynos se encuentran aislados uno del otro. Esto implica que de existir fallos en la infraestructura interna de alguno de ellos, los otros dynos no se ven afectados, y consecuentemente tampoco la aplicación.

### 3.4.3. **Herramientas comunes para el desarrollo**

#### 3.4.3.1. Git

Git es un sistema de control de versiones distribuido gratuito y de código abierto diseñado para manejar todo, desde proyectos pequeños a muy grandes, con velocidad y eficiencia.

Git es fácil de aprender y ocupa poco espacio con un rendimiento increíblemente rápido. Supera a las herramientas SCM como Subversion, CVS, Perforce y ClearCase con características como ramificación local barata, áreas de preparación convenientes y múltiples flujos de trabajo.

La característica de Git que realmente lo distingue de casi todos los demás SCM es su modelo de ramificación.

Git te permite y te anima a tener múltiples ramas locales que pueden ser completamente independientes entre sí. La creación, fusión y eliminación de esas líneas de desarrollo lleva unos segundos.

En particular, cuando se empuja a un repositorio remoto, no tiene que empujar todas sus ramas. Puede optar por compartir solo una de sus sucursales, algunas de ellas o todas. Esto tiende a liberar a las personas para que prueben nuevas ideas sin preocuparse por tener que planificar cómo y cuándo van a fusionarlas o compartirlas con otros.

Git es rápido. Con Git, casi todas las operaciones se realizan localmente, lo que le da una gran ventaja de velocidad en sistemas centralizados que constantemente tienen que comunicarse con un servidor en algún lugar.

Una de las mejores características de cualquier SCM distribuido, incluido Git, es que está distribuido. Esto significa que en lugar de hacer un "checkout" del consejo actual del código fuente, haces un "clon" de todo el repositorio.

Esto significa que incluso si está utilizando un flujo de trabajo centralizado, cada usuario tiene esencialmente una copia de seguridad completa del servidor principal. Cada una de estas copias podría subirse para reemplazar el servidor principal en caso de una falla o corrupción. En efecto, no hay un solo punto de falla con Git a menos que solo haya una única copia del repositorio.

Debido a la naturaleza distribuida de Git y al excelente sistema de ramificación, se puede implementar una cantidad casi infinita de flujos de trabajo con relativa facilidad.

El modelo de datos que utiliza Git garantiza la integridad criptográfica de cada parte de su proyecto. Cada archivo y confirmación se suma de verificación y se recupera mediante su suma de verificación cuando se vuelve a verificar. Es imposible obtener algo de Git que no sean los bits exactos que ingresó.

A diferencia de los otros sistemas, Git tiene algo llamado "área de preparación" o "índice". Esta es un área intermedia donde las confirmaciones se pueden formatear y revisar antes de completar la confirmación.

Git se publica bajo la GNU General Public License versión 2.0, que es una licencia de código abierto. El proyecto Git eligió utilizar GPLv2 para garantizar su libertad de compartir y cambiar el software libre, para asegurarse de que el software sea gratuito para todos sus usuarios.

#### 3.4.3.2. Bitbucket

Bitbucket Cloud es una herramienta de colaboración y alojamiento de código basada en Git, creada para equipos. Proporciona un lugar para que un equipo colabore en el código desde el concepto hasta la nube, cree código de calidad a través de pruebas automatizadas e implemente código con confianza.

Permite aprobar revisiones del código de forma más eficiente mediante solicitudes de incorporación de cambios. Se puede crear una lista de comprobaciones de fusiones con aprobadores designados y mantener conversaciones directamente en el código fuente con comentarios en línea.

Las implementaciones de Bitbucket Pipelines te permite compilar, probar y realizar la implementación con CI/CD integradas.

Integra la seguridad directamente en todas las etapas del proceso de desarrollo. Consigue visibilidad en tiempo real sobre cualquier incidencia de seguridad en el código y los contenedores, determina correcciones de vulnerabilidades al principio del desarrollo y supervisa nuevos riesgos después de la implementación.

#### 3.4.3.3. Visual Studio Code

Visual Studio Code se presenta como un excelente aliado empresarial. Antes de definir qué es Visual Studio Code, es importante saber qué es un IDE. Sus siglas en inglés “Integrated Development Environment “hacen referencia a un entorno de desarrollo integrado o interactivo. Visual Studio es una aplicación informática cuya finalidad es simplemente hacer la vida de los desarrolladores y programadores mucho más fácil, al ofrecer servicios integrales que facilitan desarrollar el software de manera mucho más sencilla.

Lanzado en el año 2015, Visual Studio Code va un paso más allá de Visual Studio, presentándose como una de las herramientas más potentes del momento. Ante la pregunta de qué es Visual Studio Code, se podría decir que a grandes rasgos es un potente editor de código fuente para Windows, Linux y macOS, desarrollado por la marca Microsoft.

Es un editor de código fuente desarrollado por Microsoft para Windows, Linux y macOS. Incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código. También es personalizable, por lo que los usuarios pueden cambiar el tema del editor, los atajos de teclado y las preferencias. Es gratuito y de código abierto, aunque la descarga oficial está bajo software privativo e incluye características personalizadas por Microsoft.

Visual Studio Code se basa en Electron, un framework que se utiliza para implementar Chromium y Node.js como aplicaciones para escritorio, que se ejecuta en el motor de diseño Blink. Aunque utiliza el framework Electron, el software no usa Atom y en su lugar emplea el mismo componente editor (Monaco) utilizado en Visual Studio Team Services (anteriormente llamado Visual Studio Online).

#### 3.4.3.4. JSON

JSON, cuyo nombre corresponde a las siglas JavaScript Object Notation o Notación de Objetos de JavaScript, es un formato ligero de intercambio de datos, que resulta sencillo de leer y escribir para los programadores y simple de interpretar y generar para las máquinas.

JSON es un formato de texto completamente independiente de lenguaje, pero utiliza convenciones que son ampliamente conocidas por los programadores.

Dichas propiedades hacen de JSON un formato de intercambio de datos ideal para usar con API REST o AJAX. A menudo se usa en lugar de XML, debido a su estructura ligera y compacta.

Muchos lenguajes de programación proporcionan métodos para analizar una cadena de texto con este formato en un objeto nativo y viceversa.

Según la descripción de Stack Overflow, JSON "define seis tipos de valores: nulo, números, cadenas, booleanos, matrices y objetos".

Pese a su nombre, no es necesariamente parte de JavaScript, de hecho, es un estándar basado en texto plano para el intercambio de datos, por lo que se usa en muchos sistemas que requieren mostrar o enviar información para ser interpretada por otros sistemas.

Una de las características de JSON, al ser un formato que es independiente de cualquier lenguaje de programación, es que los servicios que comparten información por este método no necesitan hablar el mismo idioma, es decir, el emisor puede ser Java y el receptor Python, pues cada uno tiene su propia librería para codificar y decodificar cadenas en este formato.

Podemos concluir entonces en que JSON es un formato común para 'serializar' y 'deserializar' objetos en la mayoría de los idiomas.

Características de JSON:

- JSON es sólo un formato de datos.
- Requiere usar comillas dobles para las cadenas y los nombres de propiedades. Las comillas simples no son válidas.
- Una coma o dos puntos mal ubicados pueden producir que un archivo JSON no funcione.
- Puede tomar la forma de cualquier tipo de datos que sea válido para ser incluido en un JSON, no solo arreglos u objetos. Así, por ejemplo, una cadena o un número único podrían ser objetos JSON válidos.
- A diferencia del código JavaScript, en el que las propiedades del objeto pueden no estar entre comillas, en JSON solo las cadenas entre comillas pueden ser utilizadas como propiedades.

Utilizar JSON depende de las circunstancias y de las preferencias que en cada momento se determinen, pues cada uno tiene sus ventajas y desventajas. Aquí te mencionamos algunas:

Ventajas:

- Es autodescriptivo y fácil de entender.
- Su sencillez le ha permitido posicionarse como alternativa a XML.
- Es más rápido en cualquier navegador.
- Es más fácil de leer que XML.
- Es más ligero (bytes) en las transmisiones.
- Se parsea más rápido.
- Velocidad de procesamiento alta.
- Puede ser entendido de forma nativa por los analizadores de JavaScript.

Desventajas:

- Algunos desarrolladores encuentran su escueta notación algo confusa.
- No cuenta con una característica que posee XML: extensibilidad.
- No soporta grandes cargas, solo datos comunes.
- Para la seguridad se requiere de mecanismos externos como expresiones regulares.

## *3.5.* Diagrama de base de datos

Un diagrama de base de datos es un diagrama estático que describe las entidades relevantes del sistema de información, identificando sus atributos y relaciones. A continuación, se presenta el diagrama de clases que se definió.

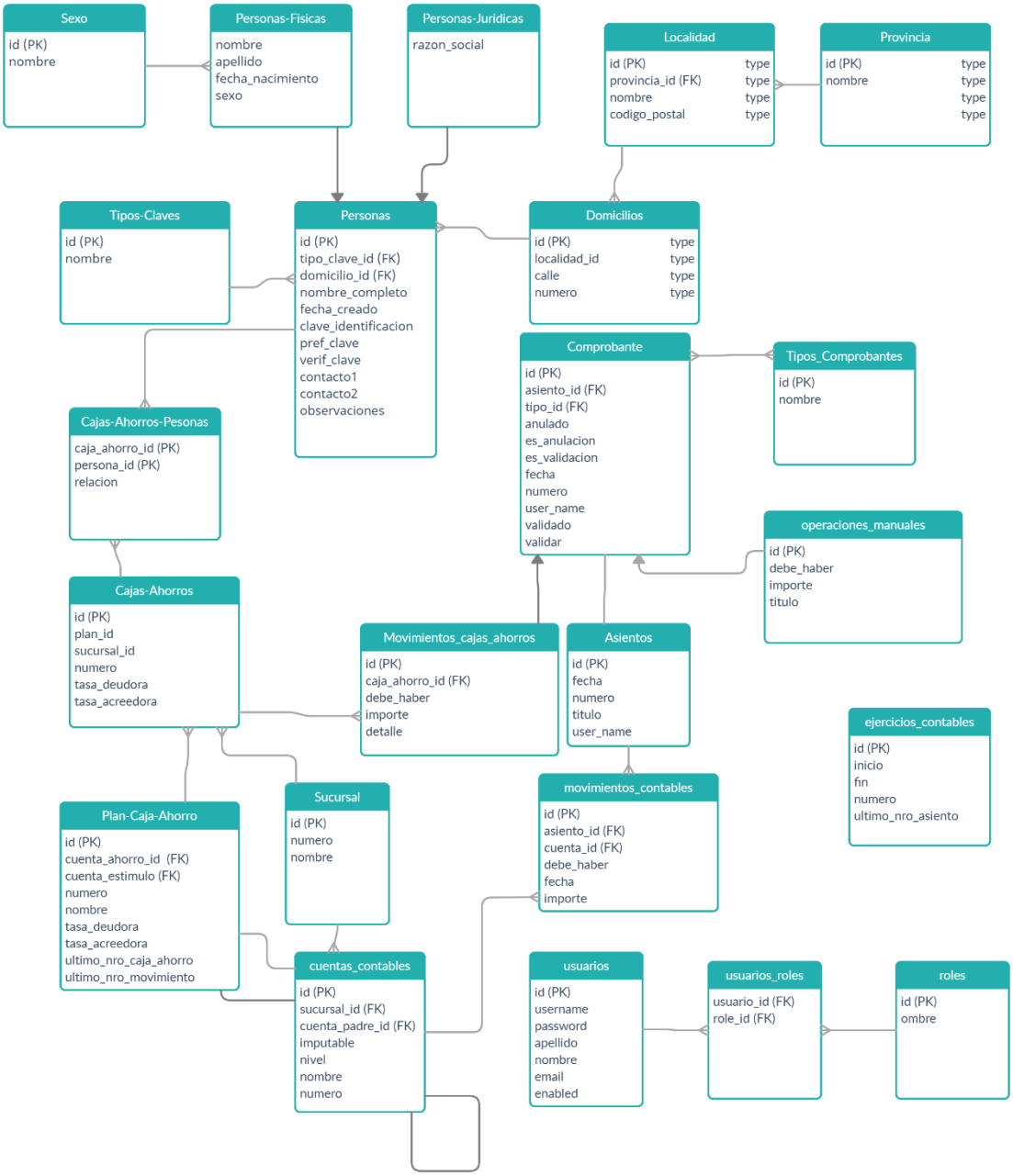



**Figura 4: Diagrama de base de datos.**

## *3.6.* Arquitectura del sistema

En este punto se explicará la arquitectura utilizada en el proyecto y el diseño de ésta.

La aplicación se ha creará utilizando el Framework de Spring Boot para el Backend y el Framework de Angular para el Frontend. Para almacenar la información se utilizará un servidor de bases de datos PostgreSQL.

Se ha seguido un modelo de tres capas, la capa de presentación, la capa de servicio y la capa de datos.

**Capa de presentación**: en esta capa se sitúa la representación de la información en interfaces de usuarios atractivas e intuitivas, para que los usuarios interactúen con el sistema y accedan a los servicios del backend. Para esta capa se utilizará el frameworks de Angular. Esta capa sería un cliente, que accedería a los servicios del servidor de aplicaciones respetando los estándares del protocolo REST. Se utilizarán los servicios de la nube de Firebase. De este modo, se logra la independencia entre el cliente y el servidor de aplicaciones, pudiendo en el futuro agregar nuevas tecnologías de clientes para smartphones, Smart TV, Notebooks, etc.

**Capa de servicio**: En esta capa se encuentra toda la lógica del negocio, la cual fue implementada utilizando el Framework de Spring Boot. Se utilizará los servicios de la nube de Heroku para desplegar el servidor de aplicaciones del backend. Los servicios del backend se expondrán a través de una API respetando los estándares del protocolo REST, a través de las cuales se enviarán y recibirán datos a los clientes, utilizando texto plano JSON. Para realizar el control de acceso y autorización a los servicios se utilizará SPRING SECURITY, con un mecanismo de tokens sin estado.

Este servidor será el responsable de ejecutar toda la lógica de negocio, y también de interactuar con el sistema de gestión de base de datos. Para la persistencia de datos, finalmente se utilizará el framework JPA que se explicará a continuación.

**Capa de datos:** En esta capa es donde se almacenan los datos. Se ha utilizado un sistema de gestión de base de datos PostgreSQL. Se utilizará el servicio de la nube de POSTGRESQL para administrar la base de datos de la aplicación ya que garantiza un sistema de seguridad y funcionamiento continuo sin fallos energéticos ni cortes de internet con estándares muy elevados.

En esta imagen ilustrativa de la estructura de 3 capas.

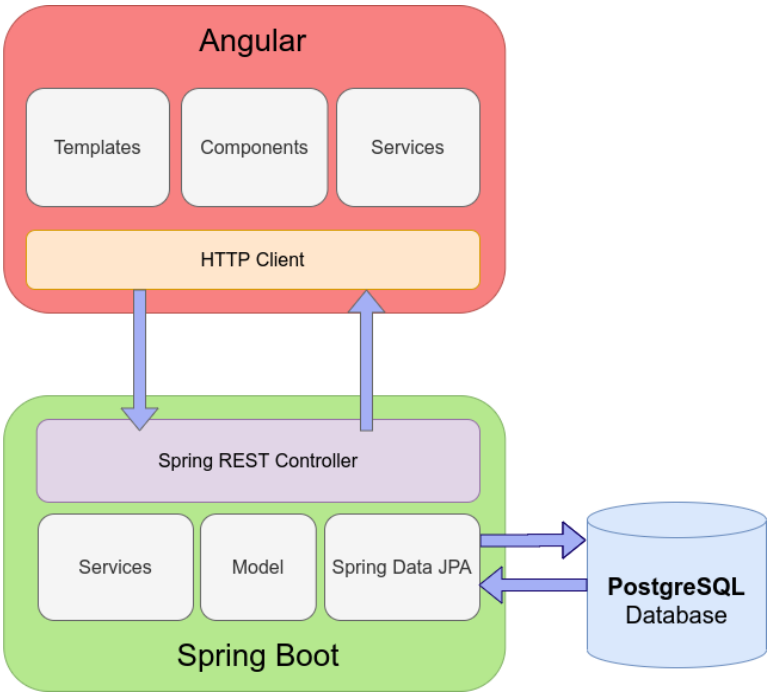



**Figura 5: Arquitectura propuesta.**

### *3.7.* Diseño Preliminar de Interfaces

En este ciclo, se refinará la especificación sobre el diseño de las interfaces del sistema, detallando la interacción que ocurrirán cuando los usuarios utilicen el sistema. En el Anexo "3: Diseño preliminar de interfaces" se encuentra el resultado de este trabajo.

## 4. Construcción

En esta etapa se lleva adelante la construcción del sistema a través de iteraciones. La construcción está guiada por el trabajo realizado en los ciclos anteriores, donde se ha realizado un análisis y diseño del sistema, lo que permite alcanzar el éxito del proyecto. La construcción de las funcionalidades de los casos de usos fue realizada, tomando cada caso de uso y codificando tanto la aplicación del cliente web, como el servidor, antes de pasar al siguiente caso de uso. De esta manera una vez que se terminaba con un caso de uso, se podía comenzar las pruebas para evaluar si existían fallos y si es lo que esperaban del sistema, antes de pasar al siguiente caso de uso. Esto permite tener una visión temprana del funcionamiento del sistema a los usuarios finales, para tener una retroalimentación y reducir ciertos riesgos, asociados a que los usuarios no acepten el sistema. También tiene el beneficio de que, al cierre de cada iteración, ya había parte del sistema funcionando y probado.

En la siguiente tabla, se muestran los plazos propuestos y funcionalidades esperadas dentro de dichos plazos, en la fase de construcción.

| Reléase | Iteración | Funcionalidad | Fecha Inicio | Fecha Fin |
|---|---|---|---|---|
| Reléase 1 | 1 | Se planteará en una primera iteración las funcionalidades básicas del módulo de socios, cajas de ahorros y la de autenticación para que se pueda comenzar a usar el sistema. En el mismo se podrá agregar socios nuevos, y buscarlos en el sistema. Agregar cajas de ahorros y buscarlas en el sistema, y realizar un depósito o extracción. Los casos de uso que incluye son:<br>CDU 01 – Realizar Autenticación<br>CDU 02 – Alta Socio<br>CDU 03 – Buscar Socio<br>CDU 07 – Alta Caja Ahorros<br>CDU 08 – Buscar Caja Ahorros<br>CDU 10 – Realizar Depósito/Extracción | 01/05/2011 | 30/06/2011 |
| | 2 | En esta iteración, se incorporarán funcionalidades para poder modificar tanto la información del socio, como de una caja de ahorros. También se podrá anular un movimiento de caja de ahorros y listar los movimientos de todas las cajas de ahorros. Los casos de usos son:<br>CDU 04 – Modificar Socio<br>CDU 09 – Modificar Caja Ahorros<br>CDU 11 – Anular Deposito / Extracción<br>CDU 14 – Listar Movimientos Caja de Ahorros | 01/07/2011 | 31/07/2011 |
| Reléase 2 | 1 | En esta iteración se incorporarán funcionalidades básicas del módulo cajero y Contabilidad. De este modo se podrá ver la información de caja del día y operar agregando un nuevo comprobante. También se podrá observar los saldos de las cuentas contables de la contabilidad Los casos de usos son:<br>CDU 15 – Mostrar resumen de caja del día<br>CDU 16 – Registrar Comprobante<br>CDU 20 – Mostrar Balance de Saldos de Cuentas Mensual | 01/08/2011 | 20/08/2011 |
| | 2 | En esta iteración se integrará mayor funcionalidad al módulo cajero de modo de poder anular un comprobante y también listar comprobantes entre dos fechas. Los casos de usos son:<br>CDU 17 – Anular Comprobante<br>CDU 18 – Listar Comprobantes | 21/08/2011 | 30/08/2011 |
| | 3 | Se realizarán los casos de uso que permiten modificar el plan de cuentas contable. Los casos de uso involucrados son: | 01/09/2011 | 31/09/2011 |

| | | | | |
|---|---|---|---|---|
| | | CDU 19 – Listar Cuentas Contables<br>CDU 21 – Alta de cuenta contable<br>CDU 22 – Modificar cuenta contable<br>CDU 23 – Eliminar Cuenta Contable | | |
| Reléase 3 | 1 | En la primera iteración del último reléase agregaremos funcionalidad del módulo de socios, de modo que los usuarios puedan hacer la baja de un socio, así como reincorporarlo. Los casos de uso a implementar son:<br>CDU 05 – Baja de Socio<br>CDU 06 – Reincorporar Socio | 01/10/2011 | 10/10/2011 |
| | 2 | En la segunda iteración del reléase, se incorporará la funcionalidad de los siguientes casos de usos:<br>CDU 12 – Habilitar Caja de Ahorros<br>CDU 13 – Deshabilitar Caja de Ahorros | 11/10/2011 | 20/10/2011 |

A continuación, se describe el trabajo realizado y los criterios tenidos en cuenta durante el desarrollo de los Releases propuestos en el Plan de Iteraciones de la Fase de Construcción. También se detallan las elecciones hechas al momento de resolver la implementación de las funcionalidades:

## *4.1.* RELEASE 1

En esta etapa, el objetivo fue lograr que los usuarios puedan autenticarse en el sistema, y a la vez que puedan ver las opciones disponibles de acuerdo con su rol. Para ello se tuvo que implementar la seguridad del sistema, que servirá para realizar el control de acceso. También se buscó, que los usuarios puedan comenzar a registrar a los socios de la mutual, mantener actualizada su información y poder consultarlos en el sistema cuando lo deseen. Por otro lado, en esta entrega se espera que el sistema permita para los socios ahorristas, dar de alta su caja de ahorros y comenzar a registrar depósitos y extracciones, ver sus resúmenes de cuenta y poder listar los movimientos de todas las cajas de ahorros existentes entre dos fechas determinadas por el usuario.

La implementación de las operaciones y consultas de cajero, como así también las consultas a la contabilidad se lo dejo para el próximo reléase.

Para esta primera etapa de construcción, se seleccionaron dos frameworks importantes para el Frontend y el Backend.

### 4.1.1. Framework para el desarrollo del Frontend

Para afrontar el desarrollo del frontend se eligió el framework de Angular. Esta decisión fue tomada a pesar de que, elegir un framework tenga algunos puntos en contra como la curva de aprendizaje que conlleva la primera utilización de este, o la dependencia futura hacia el desarrollo con dicho framework. A pesar de estos puntos en contra, las ventajas que pude encontrar, al utilizar un este framework fueron mayores en el contexto actual. Entre las ventajas que podemos nombrar, encontramos que a este framework lo están usando muchas empresas de la actualidad, por ende, aprender a usarlo nos brinda la posibilidad de insertarme más rápidamente en el mercado laboral actual. También a pesar de que tiene una curva de aprendizaje larga, también encontré que existe una amplia documentación y comunidad de desarrolladores, así como de piezas de software probadas y ya desarrolladas para reutilizarlas. Esto ayuda a que el aprendizaje sea más rápido, de este modo, una vez que los conocimientos se van consolidando, el desarrollo se acelera mucho.

En el desarrollo frontend es necesario javascript para aportar la dinámica de la visualización. En el caso de Angular, hace uso de TypeScript, que viene integrado y es recomendado porque brinda características de lenguajes orientados a objetos y tipado fuerte, sobre una sintaxis javascript mejorada.

Este framework facilita y automatiza el desarrollo de una aplicación de una sola página (PWA). Una PWA, tiene la ventaja de mejorar la velocidad de actualización de la interfaz de usuario, ya que la primera vez que se accede a la misma, el navegador obtiene del servidor los archivos necesarios para crear la vista principal de la aplicación. Luego, dichas interfaces quedan almacenadas en el cache del navegador y la dinámica de cambiar de interfaces es muy veloz y no necesita mucho ancho de banda. Esto lo hace internamente la aplicación a través del enrutamiento que posee el framework. Cada vez que se quiere acceder a una nueva interfaz que no se encuentra en caché el sistema la trae y la almacena una sola vez en el navegador. Además, el envío y recepción de datos desde el backend se realiza en formato JSON, lo cual no ocupa ancho de banda y la velocidad de comunicación es muy alta.

Está pensado para que el desarrollo siga una arquitectura Modelo – Vista – Controlador. Todo el código dentro del frontend se organiza de tal manera de separar la lógica de la vista y de la comunicación con el servidor de backend.

El framework de Angular no usa realmente el patrón Modelo-Vista-Controlador (MVC), se basa en componentes, y el modelo está más cercano a un Modelo-Vista-Vista-Modelo (MVVM). Esto es debido a una propiedad de data binding bidireccional que agiliza mucho el desarrollo de interfaces web, pero a su vez hace que se aleje del modelo clásico MVC.

Por último, esta aplicación se desplegará en el servidor de Firebase, debido a que es gratuito y tiene una API sencilla y muy rápida para desplegar aplicaciones, y es compatible con el framework de Angular, ya que tanto Firebase como Angular son de Google.

Ahora voy a explicar la estructura del proyecto:

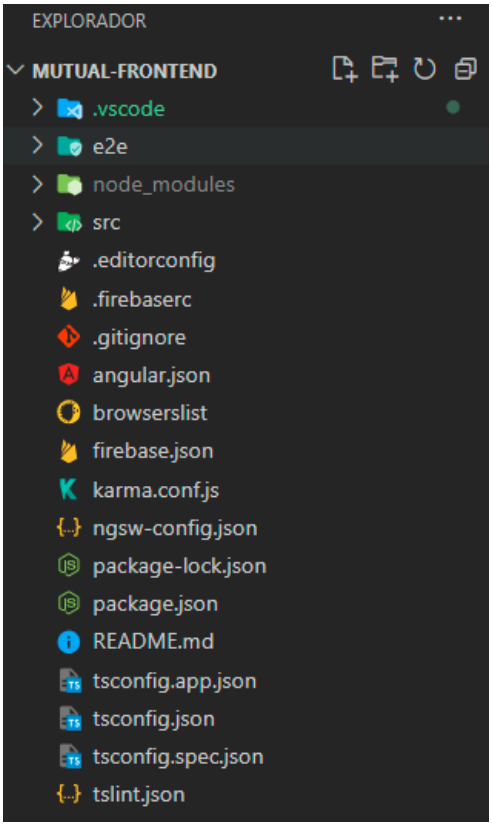



**Figura 6: Estructura del proyecto en el frontend.**

Como vemos en la imagen, vemos 3 carpetas:

- .vscode: esta carpeta se usa para hacer ciertas configuraciones del área de trabajo del entorno de desarrollo Visual Studio Code.
- e2e (e2e de “end to end”): contiene una serie de ficheros cuyo objetivo es realizar test automáticos, que simularán que un usuario real interactúa con nuestra app de Angular.
- Node_modules tiene los paquetes (también conocidos como dependencias) instaladas en nuestro proyecto con el instalador de paquetes (npm) de node.

- src: Carpeta en la que se realizará el desarrollo de la aplicación. Debido a que aquí se centra el código principal de la aplicación, lo voy a detallar en un punto aparte.

También podemos observar varios archivos con diferentes extensiones, en su mayoría son archivos de configuración del proyecto:

- .editorconfig: Configuraciones a la hora de desarrollar, identación y configuraciones del editor.
- .firebaserc: en este archivo se almacenan los alias del proyecto
- .gitignore: Archivo para que git ignore ciertas carpetas que no hace falta subir al repositorio, como node_modules, o archivos con las credenciales de las bases de datos o de los servidores que necesitamos que no se guarden en los sistemas de control de versiones.
- .gitignore: contiene un listado de ficheros y directorios que no queremos que se suban al repositorio de git (como contraseñas, o el paquete de node_modules, etc.)
- angular.json: este archivo contiene la configuración de Angular. Si nos metemos en el podemos ver que asocia y define muchos valores tales como: el nombre del proyecto, el HTML inicial que cargaremos, etc.
- .browserslist: es un listado sobre la que podemos definir la compatibilidad de los navegadores con nuestro proyecto de Angular.
- .firebase.json: El archivo es obligatorio para implementar recursos con Firebase CLI, ya que especifica qué archivos y qué configuración del directorio del proyecto se deben implementar en el proyecto de Firebase.
- karma.conf.js: es el archivo de configuración para realizar testing con Karma. Karma, ha sido desarrollado por el equipo de Angular, y viene preinstalada en Angular junto a Jasmine, con el fin de que nos centremos en realizar los test y para que sea lo más sencillo posible ya viene autoconfigurada para que nosotros nos centremos solamente en nuestros tests.
- ngsw-config.json: El archivo de configuración ngsw-config.json especifica qué archivos y URL de datos debe almacenar en caché el service worker de Angular y cómo debe actualizar los archivos y datos almacenados en caché.
- package.json y package-lock.json: el archivo package.json es un manifiesto (declaración pública) que nos informa sobre todas las dependencias que necesita nuestra aplicación de Angular para funcionar. El package.json , tiene 3 secciones a destacar:

  - Scripts: contiene un listado de métodos asociados a npm como start con el que ya hemos visto como arrancar una aplicación. También podemos añadir los nuestros.
  - Dependencies: contiene las dependencias necesarias para la ejecución de nuestra aplicación. Como, por ejemplo: angular/router, angular/core, angular/forms, etc.
  - DevDependencies: contiene las dependencias que vamos a usar durante el desarrollo de nuestra aplicación.
- README.md es el archivo que suele visualizarse en el repositorio a la hora de descargarlo. Suele ser una especie de guía que contiene los pasos o instrucciones de configuración de rutas, comandos de instalación, etc.
- tsconfig.app.json, tsconfig.json y tsconfig.spec.json: estos 3 ficheros nos permiten definir configuraciones para cuando realicemos la transpilación (traducción) de nuestro código. Este código no puede ser leído por el navegador por el momento ya que está en TypeScript (con extensión .TS) y una vez transpilado a JavaScript (con extensión .JS) y que ahora sí que puede ser leído por el browser.
- tslint.json: Configuración del linter de TypeScript (un linter nos sirve para que el editor nos muestre errores, sugerencias, etc).

**Carpeta src:**

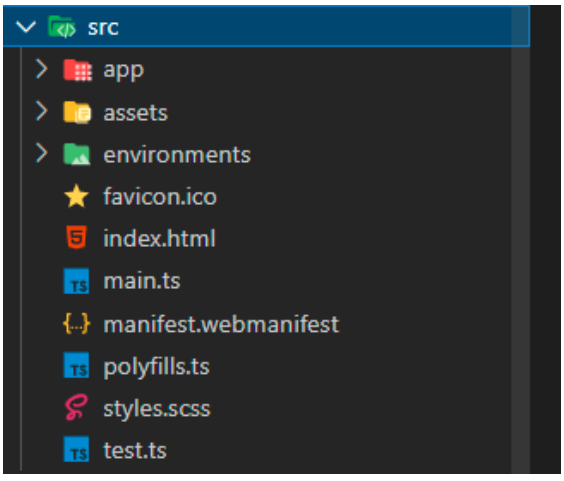



**Figura 7: Carpeta src en el frontend.**

En esta carpeta es donde se encuentra todo el código de nuestro proyecto (componentes, estilos, configuraciones de entornos, archivos de traducciones, etc.).

**Carpeta app**

Aquí se van a encontrar los componentes, vistas, y servicios de la app.

- Componentes: Son los diferentes elementos que van a componer la aplicación, por ejemplo, una tabla, una ventana modal o un formulario. Los componentes están formados por 3 tipos de archivos principales (un archivo HTML, un archivo SCSS y un archivo TypeScript). En el HTML se define cómo se estructura el componente. En el TypeScript de programa el comportamiento del componente. En el SCSS se definen los estilos propios del componente.
- Módulos: En estos módulos (archivos "module.ts") se importan los componentes que pertenecen a dicho módulo. Nos permiten organizar la aplicación y reutilizar componentes pertenecientes a otros módulos. Puede crearse un módulo por cada vista de la aplicación y adicionalmente se puede crear un módulo que englobe a los componentes comunes o compartidos por varias vistas. Además, disponer de diferentes módulos nos permite beneficiarnos de la carga perezosa (lazy loading) que aporta un mayor rendimiento en la aplicación debido a que sólo se cargarán aquellos módulos que sean necesarios en cada momento. Al crear una aplicación vendrá un único módulo por defecto: "app.module.ts".
- Archivos de rutas: Nos permiten indicar las rutas que tendrán las diferentes vistas de la aplicación. Las rutas se configuran dentro de los archivos llamados routing.module.ts, puede haber rutas principales y también rutas anidadas.

**Carpeta assets**

Esta carpeta está dedicada a guardar los diferentes recursos que necesitemos en nuestra aplicación como pueden ser imágenes, iconos, tipos de fuentes o archivos de traducciones si vamos a crear una aplicación con varios idiomas.

**Carpeta environments**

En esta carpeta guardaremos los archivos de configuración de los diferentes entornos que vayamos a utilizar.

Otros Archivos:

- favicon: El favicon o icono principal del sitio.
- index.html: Punto de entrada a nuestra web, este archivo se carga en todo el sitio, por lo que puedes poner código para que se incluya en todas las vistas.
- main.ts: Algunas configuraciones de Angular.
- mainfest.webmanifest: el archivo de manifiesto es una forma de orientar a los navegadores web, especialmente a los dispositivos móviles, sobre cómo mostrar una

aplicación. Es un archivo JSON donde podemos indicar sus diferentes propiedades con sus valores.

- polyfills.ts: Configuraciones y código que se ejecutará antes de que se inicie la app.
- styles.css: Estilos css globales que se aplicarán en todas las vistas de la página.
- test.ts

**Carpeta app**

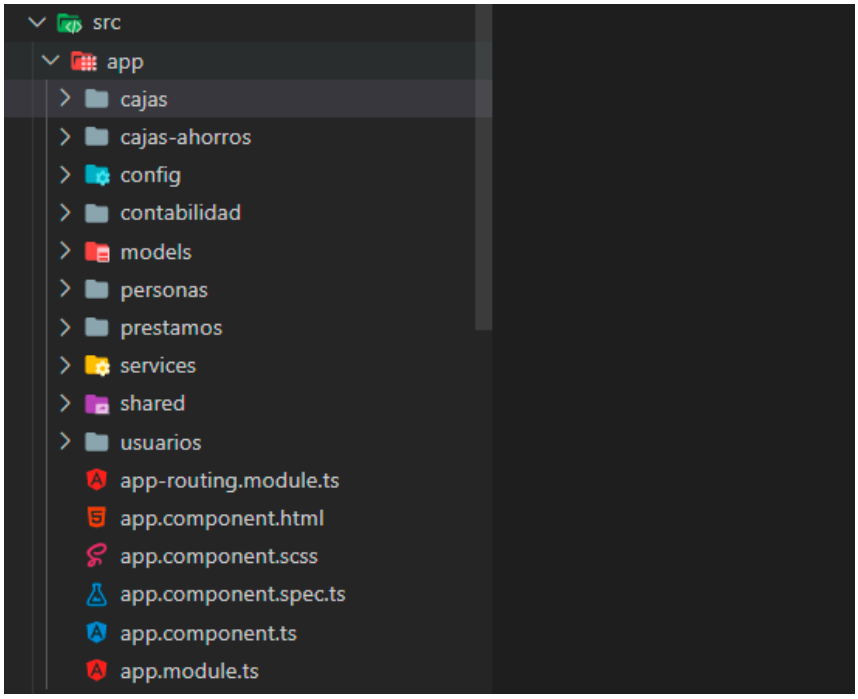



**Figura 8: Carpeta app en el frontend.**

En esta, se encuentran las carpetas de los módulos que componen la aplicación, y que se detalla más adelante.

Todos los módulos de la aplicación siguen la misma estructura de componentes, de modo que se utilizará el módulo de caja de ahorros para explicar en detalle cómo funciona.

**Módulo de caja de ahorros**

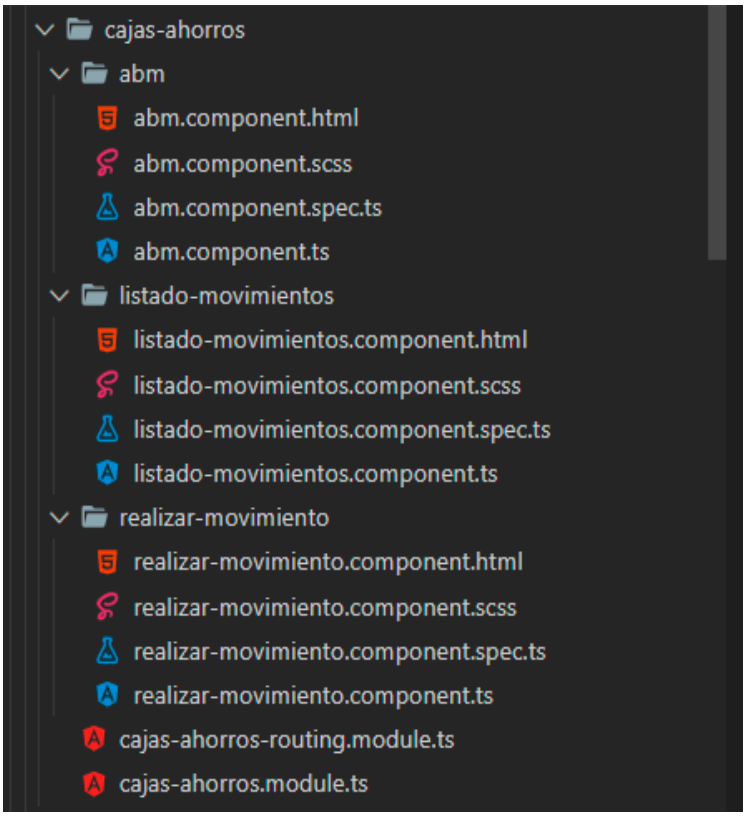



**Figura 9: Módulo de Caja de Ahorros en el frontend.**

Este módulo se encargará de toda la lógica de presentación de las funcionalidades de las cajas de ahorros. Este módulo está compuesto de tres componentes “listado-movimientos”, “abm” y “realizar-movimiento”. Con estos componentes se cubrirán los casos de usos “Alta de caja de ahorros”, “Modificar caja de ahorros”, “Buscar Caja de ahorros”, “Realizar depósito/extracción”, “Habilitar caja de ahorros”, “Inhabilitar caja de ahorros”, “Anular Movimiento”, “Listar movimientos de cajas de ahorros”.

Además, este módulo posee dos archivos necesarios para el funcionamiento. El archivo cajas-ahorros.routing.module.ts es el encargado de enrutar hacia el componente adecuado, cuando desde alguna parte de la aplicación se solicita una ruta que coincida con uno de los compontes de este módulo. Por otro lado el archivo “cajas-ahorros.module.ts” es el encargado de importar los componentes a reutilizar de otros módulos, y también de declarar los componentes de dicho modulo, para poder ser reutilizados en otros módulos. A continuación, se explica la estructura de archivos de un componente “listado-movimientos”

**Carpeta del Componente listado-movimientos**:

Este componente sirve para listar los movimientos de todas las cajas de ahorros entre dos fechas determinadas. Para desempeñar la funcionalidad, este componente está compuesto de tres archivos y uno extra para realizar test de la aplicación.

**listado-movimientos-component.html**

Este archivo HTML representa la vista del componente. En ella se encuentran los botones, tablas, títulos, formularios, y elementos que sirven para que el usuario interactúe con el sistema. Los campos del formulario son llenados por el controlador (el archivo con terminación ".ts"). Además, los eventos producidos en esta vista, son procesados por el controlador.

**listado-movimientos-component.scss**

Este archivo scss es una hoja de estilo para darle todos los formatos que embellecen la interfaz.

**listado-movimientos-component.ts**

Este archivo con terminación ".ts" es el controlador de la interfaz. Es el encargado de recibir los eventos generados en la página HTML y procesarlos. Además, será aquí donde se invoque a los servicios de la carpeta Service, para luego procesar nuevamente la información recibida desde el backend y servirla a la página HTML.

**listado-movimientos-component.spec.ts**

Este archivo es necesario para realizar los test del componente.

**Módulo de Socios**

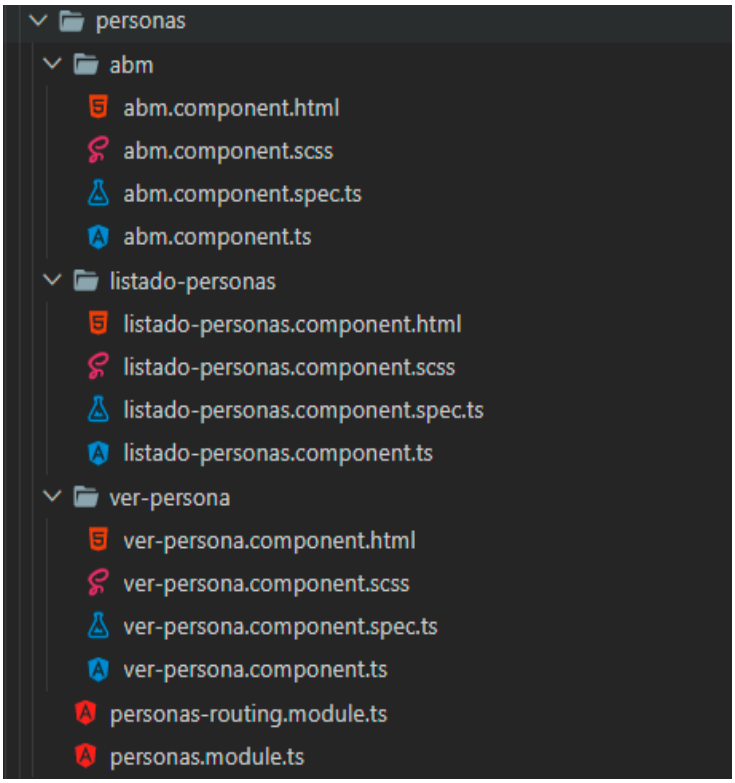



**Figura 10: Módulo de Socios en el frontend.**

Este módulo se encargará de toda la lógica de presentación de las funcionalidades de la Socios. Con estos componentes se cubrirán los casos de usos “Alta de Socio”, “Baja de Socio”, “Modificar Socio”, “Buscar Socio”, “Reincorporar Socio”.

**Módulo de Usuarios**

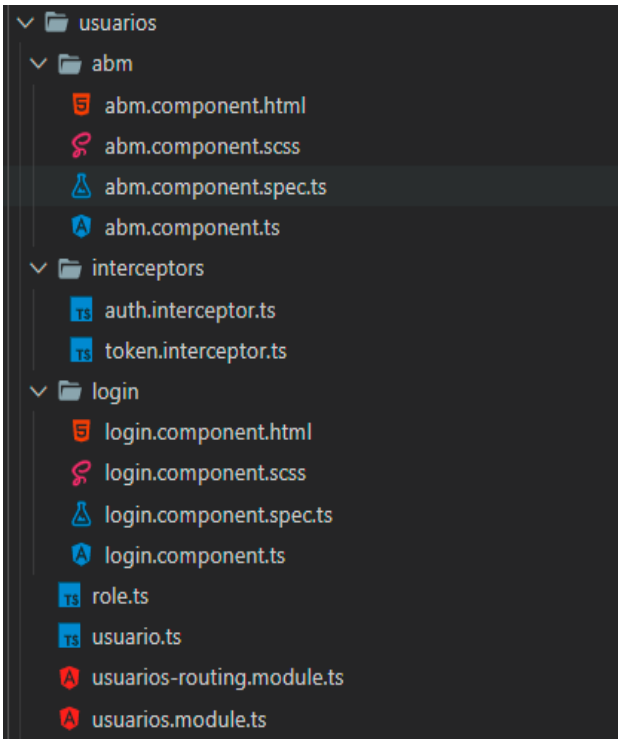



**Figura 11: Módulo de Usuarios en el frontend.**

Este módulo se encargará de toda la lógica de presentación de las funcionalidades de la Usuarios y del manejo de la seguridad de la aplicación. Con estos componentes se cubrirán los casos de usos “Realizar Autenticación”.

### Módulo Shared

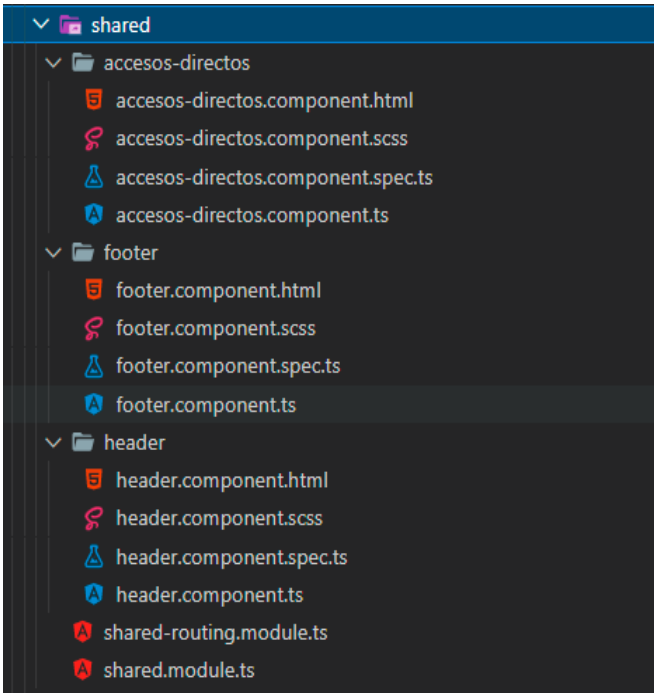



**Figura 12: Módulo de Shared en el frontend.**

En este módulo se encuentra la configuración de la pantalla principal de la aplicación. Entre los componentes encontramos el menú principal, el pie de página y todos los accesos a los demás componentes de la aplicación.

**Carpeta Service**

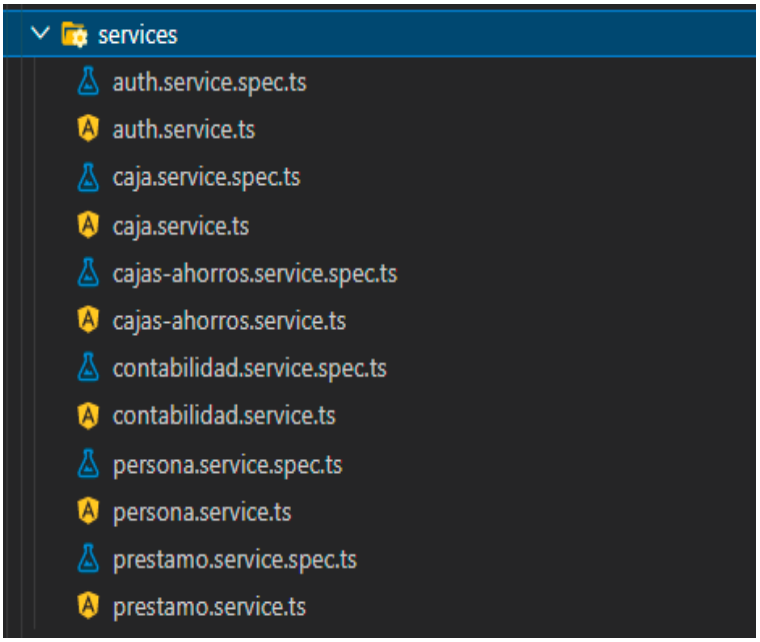



**Figura 13: Carpeta services en el frontend.**

En este módulo se encuentran todos los servicios que serán los que se comunicarán finalmente con las APIs del backend. Estos servicios son invocados por los controladores de los compontes, para llenar las vistas con datos y son también usados cuando se necesita enviar información para persistirse en el servidor.

**Carpeta Models**

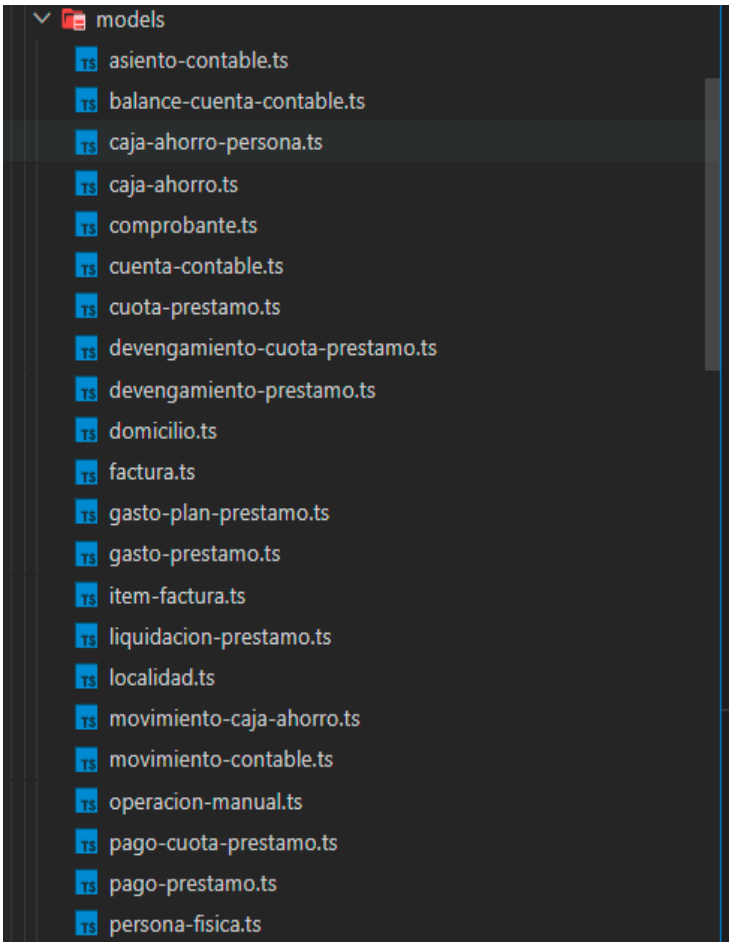



**Figura 14: Carpeta models en el frontend.**

Esta carpeta contiene todas las clases que representan el modelo de información desplegada en la vista, e intercambiada con las APIs del servidor.

### Carpeta Config


**Figura 15: Carpeta config en el frontend.**

Esta es usada para almacenar la dirección URL del servidor, que será utilizada por los servicios para acceder a las APIs. En etapas del desarrollo aquí se encuentra habilitada la URL del entorno de desarrollo para realizar pruebas. Pero en producción la URL resultante será la del servidor de aplicaciones de Heroku en la nube.

### 4.1.2. Framework para el desarrollo del backend

En esta fase se tiene el desarrollo del servidor, dado que en esta se desarrolló la lógica de la aplicación, se hizo uso de la arquitectura MVC en 3 niveles, la codificación se realizó en lenguaje Java.

Se utilizó el framework Spring Boot. Spring Boot me facilita la configuración del proyecto, ya que este se autoconfigura y también me facilita el empaquetado, haciendo que me concentre en los detalles de la lógica de negocio. También me ayuda a organizar el código siguiendo buenas prácticas de programación. Con Spring Boot puedo tener un Tomcat embebido dentro de mi aplicación.

Para securitizar la aplicación utilice Spring Security, donde se realiza el control de acceso y autorización sin estado de las API REST basado en JSON Web Tokens (JWT). De este modo, desde el backend no se necesita tener un registro de los tokens. Cada token es autónomo: contienen en sí mismos toda la data necesaria para confirmar su validez (así como también información puntual del usuario que ha iniciado sesión).

Para persistir el modelo de datos se utilizó Spring Data JPA. El objetivo es simplificar al desarrollador la persistencia de datos contra distintos repositorios de información. Es decir, me ayuda a concentrarme en que es lo que se quiere persistir o consultar de la base de datos abstrayendo los detalles del lenguaje de consulta a la misma.

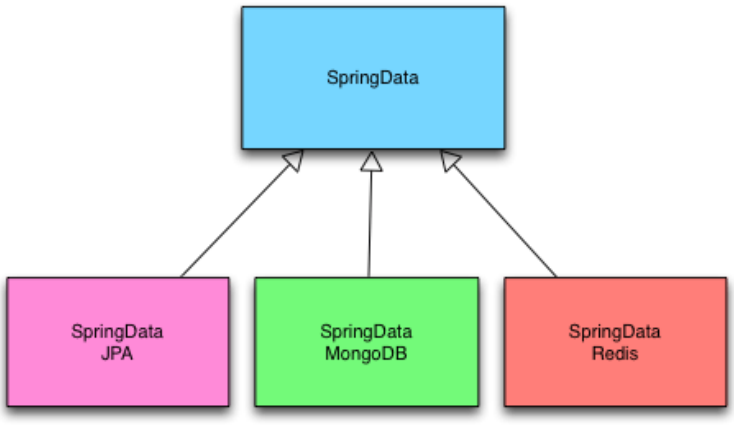



**Figura 16: Spring JPA.**

El sistema de gestión de bases de datos elegido es PostgreSQL.

Esta aplicación se desplegará en los servidores de Heroku en la nube. Elijo esta solución debido a la libertad de concentrarme más en el producto sin tener que preocuparme por la molestia de mantener el hardware, los servidores y la infraestructura.

Heroku además me ofrece acceso a una gran gama de complementos, incluido Heroku Postgres. PostgreSQL administrada de Heroku responderá sin problemas a patrones de cambio de tráfico y la ampliación y reducción sin problemas en línea con los requisitos de recursos de la base de datos de la aplicación. Al ser altamente administrado me ayuda a enfocarme más en las otras funciones centrales de la aplicación.

Entre otras ventajas de usar dicha solución encontramos,

- Heroku prioriza la seguridad de sus usuarios de acuerdo con los estándares de la industria desde cero para garantizar una protección de datos adecuada.
- Heroku Postgres ofrece alta disponibilidad de los datos. Heroku pondrá sus instancias en espera automáticamente si no están disponibles.
- Una migración fallida o una tabla eliminada no será un desastre para mi aplicación con Heroku Postgres porque la función de reversión de Heroku me permitirá crear una nueva instancia de base de datos con un solo comando para restaurar desde el punto antes de que ocurra cualquier problema.
- También me permite cifrar datos con mi clave para controlar su acceso. Con esta función, también puede bloquear el acceso desde cualquier entidad no deseada o incluso puede dar acceso completo a la entidad preferida con facilidad.

**Estructura del proyecto**

Estructura del proyecto utilizando el arquetipo web-app de Maven

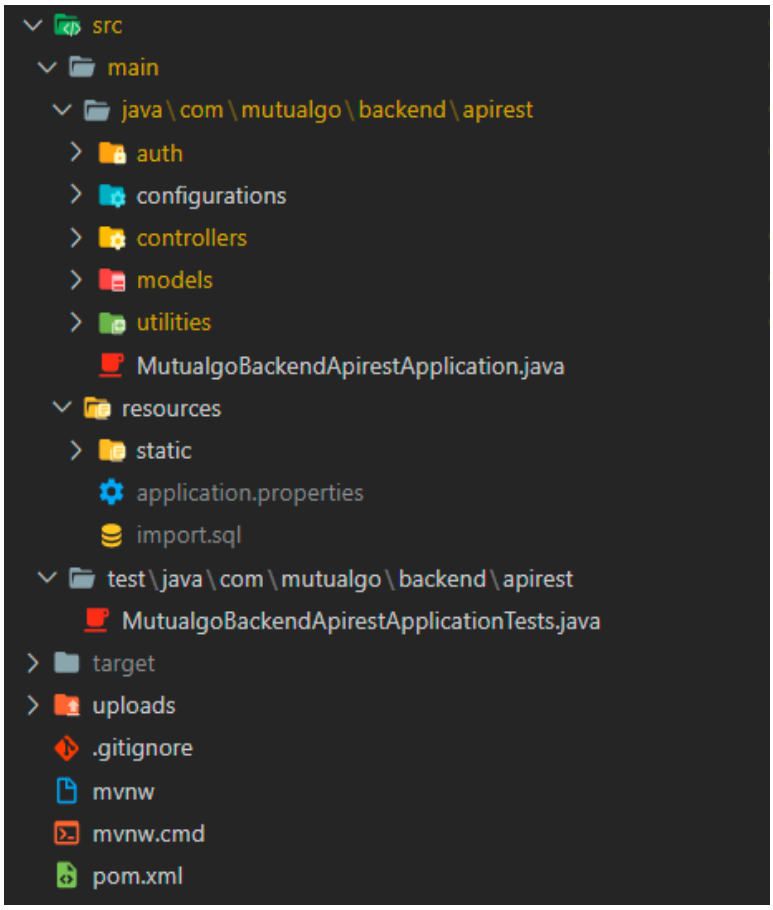



**Figura 17: Estructura del proyecto en el backend.**

Estructura del archivo pom.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.1.8.RELEASE</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.mutualgo.backend.apirest</groupId>
    <artifactId>mutualgo-backend-apirest</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>mutualgo-backend-apirest</name>
    <description>Demo project for Spring Boot</description>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
        <java.version>1.11</java.version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
            <scope>runtime</scope>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>

        <!-- https://mvnrepository.com/artifact/org.springframework.security.oauth/spring-security-oauth2 -->
        <dependency>
            <groupId>org.springframework.security.oauth</groupId>
            <artifactId>spring-security-oauth2</artifactId>
            <version>2.5.2.RELEASE</version>
        </dependency>
        <!-- https://mvnrepository.com/artifact/org.springframework.security/spring-security-jwt -->
        <dependency>
            <groupId>org.springframework.security</groupId>
            <artifactId>spring-security-jwt</artifactId>
            <version>1.0.10.RELEASE</version>
        </dependency>
        <dependency>
            <groupId>com.zaxxer</groupId>
            <artifactId>HikariCP</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-jdbc</artifactId>
        </dependency>
        <dependency>
            <groupId>org.postgresql</groupId>
            <artifactId>postgresql</artifactId>
        </dependency>

        <!-- https://mvnrepository.com/artifact/javax.xml.bind/jaxb-api -->
        <dependency>
            <groupId>javax.xml.bind</groupId>
            <artifactId>jaxb-api</artifactId>
        </dependency>
        <!-- https://mvnrepository.com/artifact/org.glassfish.jaxb/jaxb-runtime -->
        <dependency>
            <groupId>org.glassfish.jaxb</groupId>
            <artifactId>jaxb-runtime</artifactId>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>

        </plugins>
    </build>

</project>
```

**Figura 18: Archivo POM en el backend**

## Modelo

Se utilizan simples clases POJO (únicamente con sus constructores, getters y setters) para representar las entidades de la base de datos.

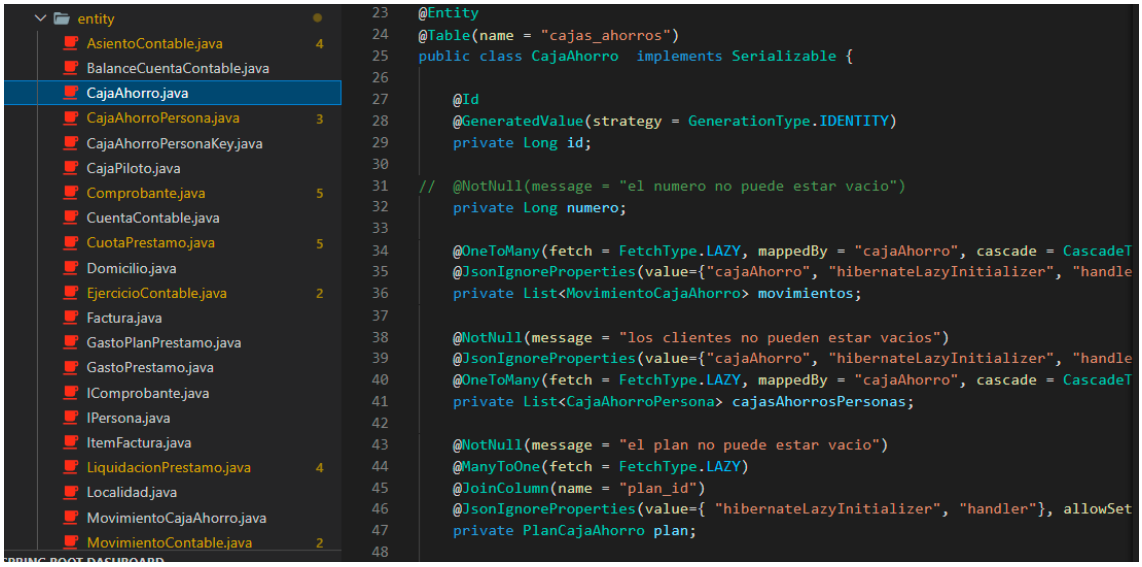



**Figura 19: Clase POJO en el backend.**

Cada entidad del modelo de datos a menudo se corresponderá con una tabla de la base de datos. En el caso del ejemplo mostrado de la clase CajaAhorro, se corresponde en la base de datos con la tabla caja_ahorros.

Explicación rápida de las anotaciones usadas:

Entity: marca nuestra entity como eso, una entity de base de datos, para que sea gestionada como tal.

Table: el nombre de la tabla de base de datos. Si no lo ponemos, tiene que ser igual al nombre de la clase.

Id: marcamos el atributo que será la primary key con esta anotación.

GeneratedValue: Como se genera la primary key.

## Capa DAO

La capa DAO es la capa más baja en el acceso al repositorio de datos.

Spring Data JPA realmente es una capa de abstracción más, montada sobre JPA, para evitarnos al máximo posible el código boilerplate necesario en cualquier aplicación que use

la especificación JPA para el acceso a base de datos. La herramienta principal son los repositorios (Repository). Spring nos provee de varias interfaces que simplemente debemos heredar (con una interfaz nuestra), y que contienen los métodos necesarios para el acceso a base de datos. Spring, por su parte, se encarga de implementar esa interfaz que hayamos creado.

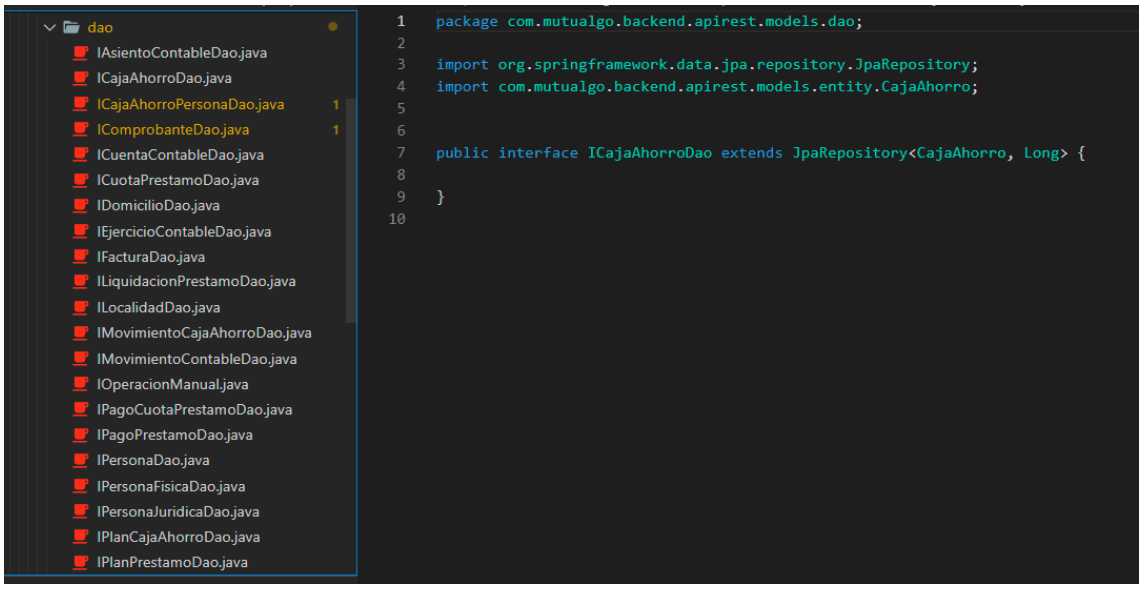



**Figura 20: Capa DAO en el backend**

Creamos un repositorio de Spring Data. Para ello simplemente creamos una interfaz que herede de JpaRepository (como parámetros genéricos, nuestro entity y Long por el tipo de la primary key). El código está correcto, es una interfaz vacía. Con esto es suficiente, ya tenemos implementada nuestra capa de base de datos. Al crear esa interfaz, es Spring quien se encarga de implementarla. Simplemente tenemos que usarla, obteniéndose como cualquier bean del contexto de Spring.

JpaRepository hereda, entre otras de CrudRepository, que tiene los métodos básicos para implementar un CRUD (getAll, save, delete, etc.).

### Capa de Servicios

Los servicios son clases que se encargan de la capa de negocio de la aplicación. Para ello, normalmente, acceden a los datos almacenados en la base de datos de la aplicación a través de los repositorios antes mencionados, hacen una serie de operaciones, y envían los datos al controlador. Los servicios (por ejemplo, el ya mencionado servicio de caja de ahorros) se caracterizan por:

Las clases que representan a los servicios están anotadas utilizando la etiqueta @Service, que es una de las anotaciones estándar de Spring para indicar que la clase es un bean de la capa de servicio. Además, hago uso de la anotación @Autowired que sirve para resolver las dependencias del bean de la capa DAO mediante un mecanismo de inyección de Spring.

Por cada servicio, implemento una interfaz fachada, como CajaAhorroService, que es llamada por una capa superior (controlador)

Dentro de cada servicio, los métodos suelen acceder a base de datos y por lo tanto tienen su transaccionalidad definida con la anotación @Transactional, diferenciando si realizan escrituras o únicamente lecturas, para garantizar la integridad de la información almacenada en casos que haya que realizar un rollback.

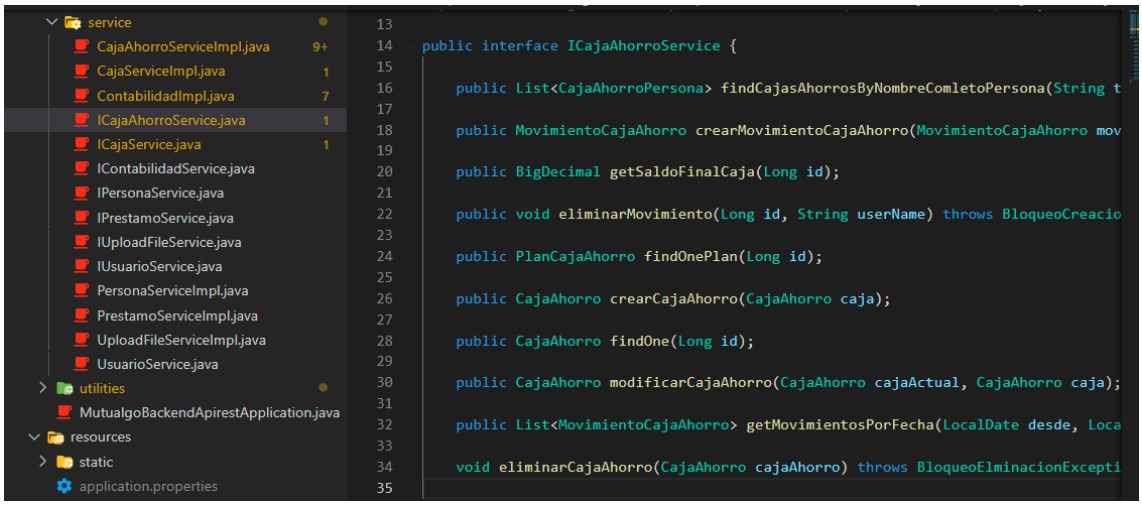
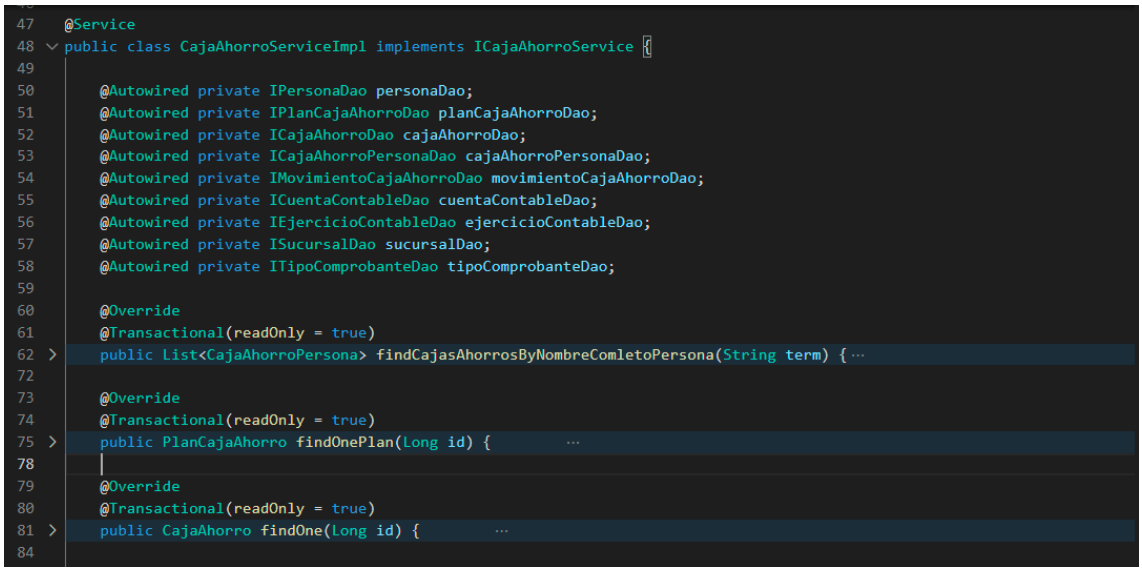



**Figura 21: Interfaz de Capa de Servicio en el backend**



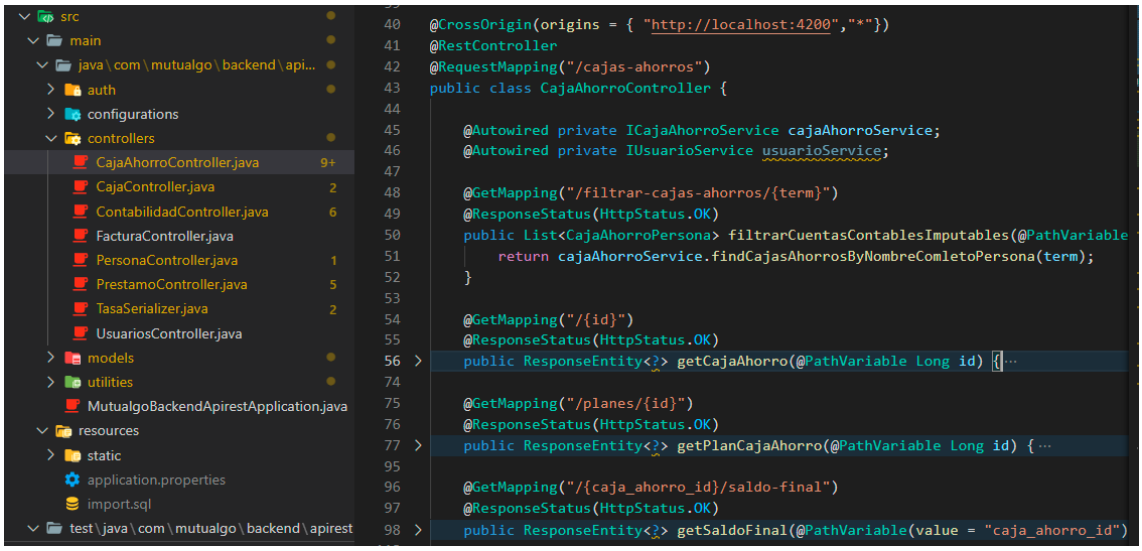
```java
@Service
public class CajaAhorroServiceImpl implements ICajaAhorroService {

    @Autowired private IPersonaDao personaDao;
    @Autowired private IPlanCajaAhorroDao planCajaAhorroDao;
    @Autowired private ICajaAhorroDao cajaAhorroDao;
    @Autowired private ICajaAhorroPersonaDao cajaAhorroPersonaDao;
    @Autowired private IMovimientoCajaAhorroDao movimientoCajaAhorroDao;
    @Autowired private ICuentaContableDao cuentaContableDao;
    @Autowired private IEjercicioContableDao ejercicioContableDao;
    @Autowired private ISucursalDao sucursalDao;
    @Autowired private ITipoComprobanteDao tipoComprobanteDao;

    @Override
    @Transactional(readOnly = true)
    public List<CajaAhorroPersona> findCajasAhorrosByNombreComletoPersona(String term) {···

    @Override
    @Transactional(readOnly = true)
    public PlanCajaAhorro findOnePlan(Long id) {        ···

    @Override
    @Transactional(readOnly = true)
    public CajaAhorro findOne(Long id) {        ···
```

**Figura 22: Clase que implementa interfaz en Capa de Servicio en el backend**

## Capa de Controladores

En esta capa se encuentran las clases de controladores, que son responsables de procesar las solicitudes de API REST entrantes, preparar un modelo y devolver la vista para que se muestre como respuesta.

El código se muestra a continuación, y se ubica en el paquete controllers:



```
src
main
java\com\mutualgo\backend\api...
auth
configurations
controllers
CajaAhorroController.java 9+
CajaController.java 2
ContabilidadController.java 6
FacturaController.java
PersonaController.java 1
PrestamoController.java 5
TasaSerializer.java 2
UsuariosController.java
models
utilities
MutualgoBackendApirestApplication.java
resources
static
application.properties
import.sql
test\java\com\mutualgo\backend\apirest
```

```java
@CrossOrigin(origins = { "http://localhost:4200","*"})
@RestController
@RequestMapping("/cajas-ahorros")
public class CajaAhorroController {

    @Autowired private ICajaAhorroService cajaAhorroService;
    @Autowired private IUsuarioService usuarioService;

    @GetMapping("/filtrar-cajas-ahorros/{term}")
    @ResponseStatus(HttpStatus.OK)
    public List<CajaAhorroPersona> filtrarCuentasContablesImputables(@PathVariable
        return cajaAhorroService.findCajasAhorrosByNombreComletoPersona(term);
    }

    @GetMapping("/{id}")
    @ResponseStatus(HttpStatus.OK)
    public ResponseEntity<?> getCajaAhorro(@PathVariable Long id) {···

    @GetMapping("/planes/{id}")
    @ResponseStatus(HttpStatus.OK)
    public ResponseEntity<?> getPlanCajaAhorro(@PathVariable Long id) {···

    @GetMapping("/{caja_ahorro_id}/saldo-final")
    @ResponseStatus(HttpStatus.OK)
    public ResponseEntity<?> getSaldoFinal(@PathVariable(value = "caja_ahorro_id")
```

**Figura 23: Clase que implementa el controlador en Capa de Controladores en el backend.**

En el fragmento de código anterior aparecen algunas anotaciones. Vamos a ver qué significa cada una de ellas:

@RestController: Con esta anotación Spring podrá detectar la clase CajaAhorroController cuando realice el escaneo de componentes.

@Autowired: A través de esta anotación Spring será capaz de llevar a cabo la inyección de dependencias sobre el atributo marcado. En este caso, estamos inyectando la capa de servicio, y por eso no tenemos que instanciarla.

@GetMapping: Con esta anotación especificamos la ruta de la API, desde la que escuchará el servicio, y define qué es un método GET de HTTP.

@ResponseStatus: sirve para indicar en la respuesta el estado HTTP resultando. Generalmente si sale todo bien por defecto se devuelve HttpStatus.OK

@ResponseEntity: Se usa para especificar los objetos genéricos de respuesta (org.springframework.http.ResponseEntity) que contiene, a su vez, más información propia de la petición.

@PathVariable: Sirve para indicar con qué variable de la url se relaciona el parámetro sobre el que se esté usando la anotación.

Además de las capas explicadas anteriormente correspondientes a la implementación del modelo-vista-controlador, tenemos otras carpetas con archivos importantes, que serán detalladas a continuación:

application.properties:

application.properties 9+

src > main > resources > application.properties

```
#spring.devtools.restart.enabled=true
spring.profiles.active=dev

#Ayuda a los acentos en la exportacion del archivo data.sql
spring.datasource.sqlScriptEncoding= UTF-8
spring.datasource.initialization-mode=always
spring.datasource.continue-on-error=true
spring.datasource.dbcp2.pool-prepared-statements=true
spring.jpa.show-sql=true

logging.level.org.hibernate.SQL=debug
spring.jackson.serialization.fail-on-empty-beans=false
spring.jackson.time-zone=America/Argentina/Buenos_Aires
spring.jackson.locale=es_AR
spring.servlet.multipart.max-file-size=10MB
spring.servlet.multipart.max-request-size=10MB
server.port=${PORT:8080}

##########################################################################################
spring.jpa.hibernate.ddl-auto=update
spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.PostgreSQLDialect
spring.jpa.properties.hibernate.jdbc.lob.non_contextual_creation=true
## default connection pool
#spring.datasource.hikari.connectionTimeout=20000
#spring.datasource.hikari.maximumPoolSize=5
spring.datasource.driverClassName=org.postgresql.Driver
spring.datasource.max-active=10
```

**Figura 24: Archivo de properties en el backend.**

A través de este archivo, se puede configurar, los mecanismos de conexión a la base de datos, como el driver utilizado, el dialecto, el tipo de codificación de utilizada en la base de datos, etc.

**import.sql**

src > main > resources > import.sql

```sql
/* Populate tabla personas */

INSERT INTO sexos ( id, nombre) VALUES ( 1, 'MASCULINO');
INSERT INTO sexos ( id, nombre) VALUES ( 2, 'FEMENINO');

INSERT INTO tipos_claves ( id, nombre) VALUES ( 1, 'CUIL');
INSERT INTO tipos_claves ( id, nombre) VALUES ( 2, 'CUIT');
INSERT INTO tipos_claves ( id, nombre) VALUES ( 3, 'CDI');

INSERT INTO provincias ( id, nombre) VALUES ( 1, 'Santa Fe');

INSERT INTO localidades ( id, nombre, codigo_postal, provincia_id) VALUES ( 1, 'Villa Ocampo', 3580, 1);
INSERT INTO localidades ( id, nombre, codigo_postal, provincia_id) VALUES ( 2, 'Las Toscas', 3583, 1);
INSERT INTO localidades ( id, nombre, codigo_postal, provincia_id) VALUES ( 3, 'Reconquista', 3500, 1);
INSERT INTO localidades ( id, nombre, codigo_postal, provincia_id) VALUES ( 4, 'Avellaneda', 3560, 1);

/* INSERT INTO domicilios ( id, calle, numero, localidad_id ) VALUES ( 1, 'San Martin'    , 1525, 1); */
/* INSERT INTO domicilios ( id, calle, numero, localidad_id ) VALUES ( 2, 'Caseros'           , 358 , 1); */
/* INSERT INTO domicilios ( id, calle, numero, localidad_id ) VALUES ( 3, 'Zona Urbana S/N', NULL, 1); */
/* INSERT INTO domicilios ( id, calle, numero, localidad_id ) VALUES ( 4, 'Zona Rural S/N' , NULL, 1); */

/* INSERT INTO personas ( id, nombre_completo, fecha_creado, tipo_clave_id, pref_clave, clave_identificacion, verif_cla
/* INSERT INTO personas ( id, nombre_completo, fecha_creado, tipo_clave_id, pref_clave, clave_identificacion, verif_cla
/* INSERT INTO personas ( id, nombre_completo, fecha_creado, tipo_clave_id, pref_clave, clave_identificacion, verif_cla
/* INSERT INTO personas ( id, nombre_completo, fecha_creado, tipo_clave_id, pref_clave, clave_identificacion, verif_cla
```

**Figura 25: Archivo import.sql en el backend**

Este archivo es utilizado para inicializar la base de datos. En tiempos de desarrollo, y según las propiedad spring.jpa.hibernate.ddl-auto establecida en el application.properties, cada vez que se corre el servidor, la base de datos se inicializará nuevamente con esta información. Pero en momentos de pasar a producción, se cambia la propiedad del application.properties para que la base de datos solo se inicialice la primera ejecución del servidor en Heroku y sirva para inicializar la base de datos, ya que luego serán los usuarios los que la actualizarán.

**Carpeta auth**

En esta carpeta, se almacenan los archivos de configuración de Spring Security.

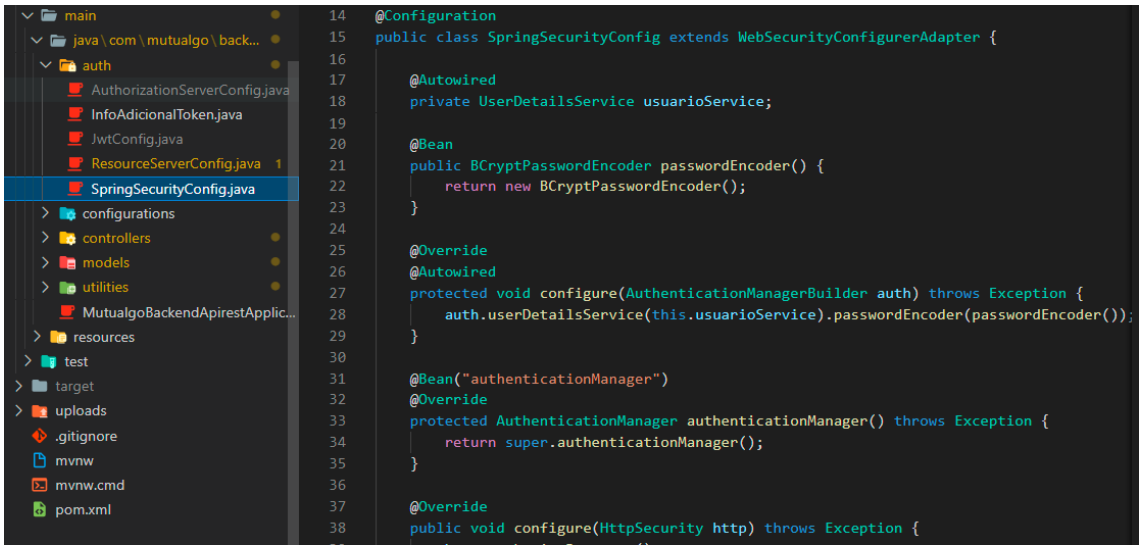



**Figura 26: clase SpringSecurityConfig.java en el backend.**

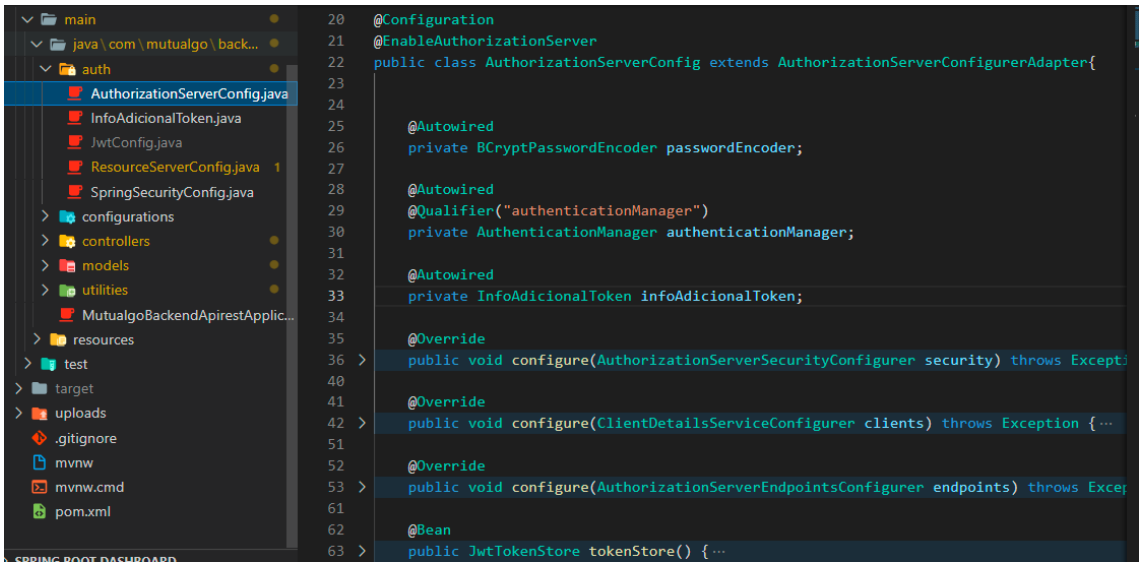



**Figura 27: clase AuthorizationServerConfig.java en el backend.**

**Figura 28: clase ResourceServerConfig.java en el backend.**

```
public class JwtConfig {
    public static final String LLAVE_SECRETA = "alguna.clave.secreta.12345678";

    public static final String RSA_PRIVADA = "-----BEGIN RSA PRIVATE KEY-----\r\n" +
            "MIIEogIBAAKCAQBi1VUUJfSiC0M5G/ldwcMXDop1v/uvtU3t2BjdI9uSsiKAKdlP\r\n" +
            "JvwsmR66CKwEK0DUoOL3RbHAV14wgqvvvLzfiJouDzID5KV4lLuk2rlfG8dnaoUG\r\n" +
            "VTyH3N/zWmgO6h6R92XW8JnCSMhHZzuagfmMXIOwJseaRBrPe5cjPKM7m711xaNl\r\n" +
            "+KcItlSOLet2X4HWSJKINBgVJYxKY/mCZeyZP+K0O+VyRo0jXjQHoijWqoqHmqh0\r\n" +
            "bZD6U+TCH9UtsRDNCUbW6b5mbMKiRksShJo3jAcOfC+JFG4lpED4p0ChOvxig8FN\r\n" +
            "wML7tEOeNK2fJRqz6kP+IjeOc8K7suNBSB6XAgMBAAECggEAGvYDTYnFIGdeITgm\r\n" +
            "YJM4+xM2ZkTHDfmxlo56yei8P1pk2v2PkG4cILZskWdINaLu/pIdGENk6AknjKg4\r\n" +
            "6x9bjsce+C2hATrggPJh9ejM5MsZLHb8buvVpWJ1nVcuR9NKH5uTpnzgFQXxzLi2\r\n" +
            "o8GNJ+Mc8XLCxKyLe8pFQpGKHfpYb9k2HO/bKlTQd6A8DMPTZLUsCLBYcIliIdEt\r\n" +
            "FRBqBps+A8opjSO5ac7NMnJv9Qf4F3p06Jv14QKzPqv4o2NAm4PGwaq/n3OOTV4A\r\n" +
            "irlWss6FewRyN/BigW6nZkWEE6IIpGnKG3nuNsgsNJt3Hu+9t5CXqlS0rVHXB5iD\r\n" +
            "dylOwQKBgQC5pU2wrT6COz/DZWdekwstYnKGRDcn9jVFNFMwYJ57THnmkXodmIph\r\n" +
            "DZMF/Wv/zBE/18Yn4W/0l+WxzLtGD/fS/5wTtzjExFCRY6OhAYnDk2Zm/RTWGyh0\r\n" +
            "XmvKdQfaHMW8yHhUzaj0/J4foYzoF4b5fHRL/KR7bcxTG4SJcTpWxwKBgQCIScuh\r\n" +
```

**Figura 29: clase JwtConfig.java en el backend.**

```
@Component
public class InfoAdicionalToken implements TokenEnhancer{

    @Autowired
    private IUsuarioService usuarioService;

    @Override
    public OAuth2AccessToken enhance(OAuth2AccessToken accessToken, OAuth2Authentication authentication) {

        Usuario usuario = usuarioService.findByUsername(authentication.getName());
        Map<String, Object> info = new HashMap<>();
        info.put("info_adicional", "Hola que tal!: ".concat(authentication.getName()));

        info.put("nombre", usuario.getNombre());
        info.put("apellido", usuario.getApellido());
        info.put("email", usuario.getEmail());

        ((DefaultOAuth2AccessToken) accessToken).setAdditionalInformation(info);

        return accessToken;
    }
```

**Figura 30: clase InfoAdicionalToken.java en el backend.**

## *4.2.* RELEASE 2

Para este reléase se implementaron los casos de uso como detalla la tabla de los pasos propuestos. Debajo detallaré los módulos implementados en el frontend.

**Módulo de cajas:**

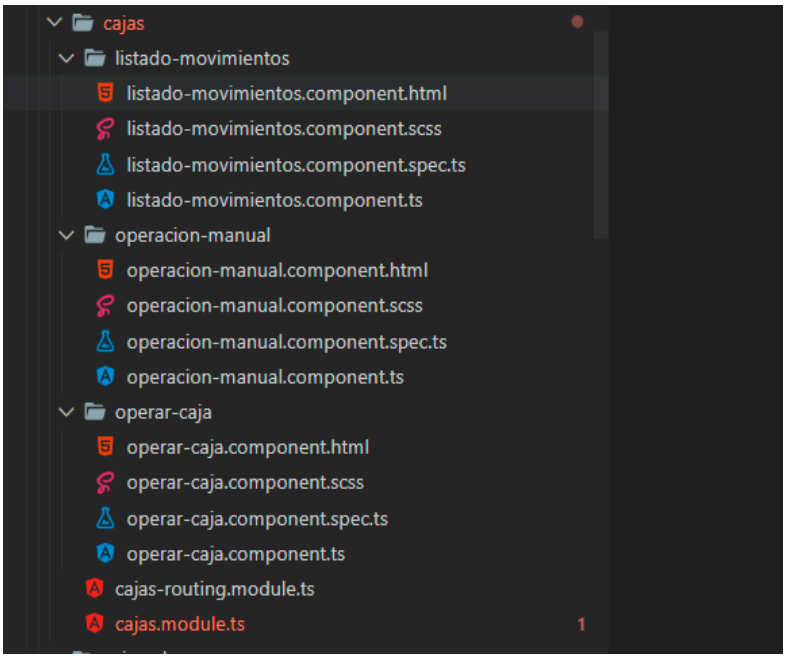



**Figura 31: Modulo de Cajas en el frontend**

En este módulo se organiza toda la funcionalidad de la caja. Este módulo se compone de tres componentes para cubrir los casos de uso “Registrar Comprobante”, “Mostrar Resumen de Caja del día”, “Anular Comprobante”, “Listar Comprobantes”.

El archivo cajas.module.ts se encarga de importar componentes de otros módulos para ser reutilizados dentro de este, así como también sirve para exportar cierta funcionalidad a ser reutilizada en otros módulos. Además, para organizar toda la funcionalidad relacionada dentro de la caja en un solo lugar.

El archivo cajas.routing.module.ts es el encargado de enrutar hacia el componente adecuado cuando desde alguna parte de la aplicación se solicita la ruta correspondiente que coincide con uno de los componentes del módulo.

**Componente listado-movimientos:**

Este componente sirve para listar los movimientos de caja entre dos fechas determinadas. Para desempeñar la funcionalidad, este componente está compuesto de tres archivos y uno extra para realizar test de la aplicación.

**listado-movimientos-component.html**

Este archivo html representa la vista del componente. En ella se encuentran los botones, tablas, títulos, formularios, etc… que sirven para que el usuario interactúe con el sistema.

Los campos del formulario son llenados por el controlador (el archivo .ts) y los eventos producidos en esta vista son procesados por el controlador.

**listado-movimientos-component.scss**

Este archivo scss es una hoja de estilo para darle todos los formatos que embellecen la interfaz.

**listado-movimientos-component.ts**

Este archivo con terminación ".ts" es el controlador de la interfaz. Es el encargado de recibir los eventos generados en la página HTML, procesarlos, invocar a los servicios necesarios para acceder a las APIs del servidor y luego procesar nuevamente la información recibida para ser servida a la página HTML.

**listado-movimientos-component.spec.ts**

Este archivo es necesario para realizar los test del componente.

**Módulo de Contabilidad**

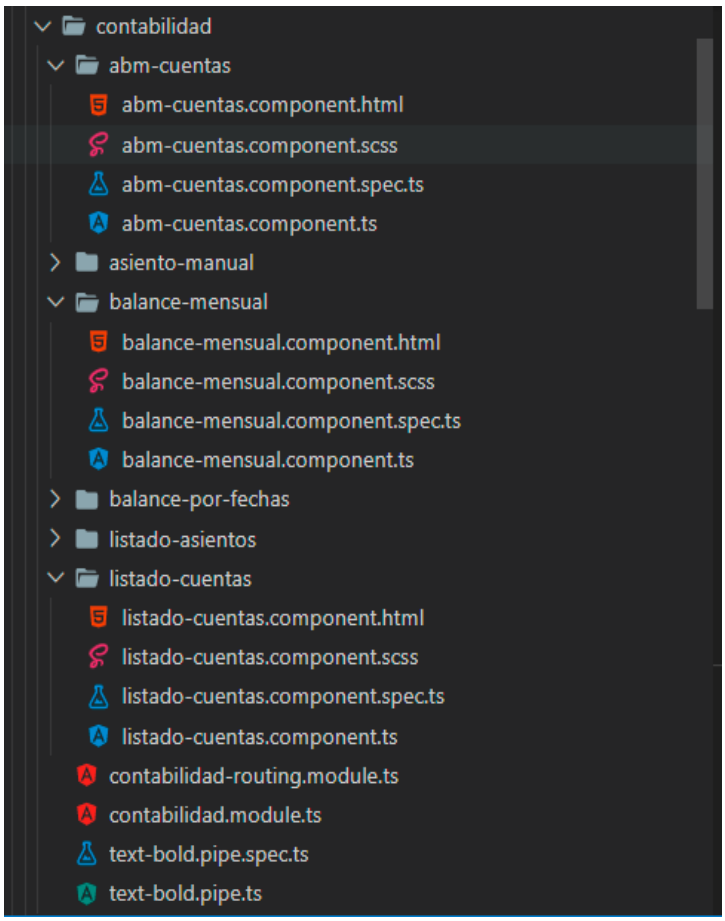



**Figura 32: Modulo de Contabilidad en el frontend.**

Este módulo se encargará de toda la lógica de presentación de las funcionalidades de la contabilidad. Con estos componentes se cubrirán los casos de usos “Alta de Cuenta Contable”, “Baja de Cuenta Contable”, “Modificación de Cuenta Contable”, “Listar Cuentas Contables”, “Mostrar Saldos de Cuentas Mensual”.

**Módulos implementados del backend:**

En el caso de los módulos del backend las imágenes obtenidas y dispuestas para el reléase 1 detallan los módulos implementados en este reléase.

### *4.3.* RELEASE 3

Al culminar la etapa de este reléase, se logró tener la versión completa del sistema planeado al inicio del proyecto. Se agregaron las funcionalidades previstas de los casos de usos CDU 05 – Baja de Socio, CDU 06 – Reincorporar Socio, CDU 12 – Habilitar Caja de Ahorros, CDU 13 – Deshabilitar Caja de Ahorros. Este release fue el que menos retraso tuvo respecto al cronograma de la iteración y los plazos previstos. Esto se debió a la amplia familiarización con las herramientas de desarrollo y ya los esfuerzos se centran en el desarrollo de las funcionalidades de la aplicación.

## 5. Conclusiones

El quinto y último capítulo del presente Proyecto Final de Carrera tiene por objetivo presentar las conclusiones a las que se arribó durante el desarrollo de éste.

### *5.1.* Conclusiones

El desarrollo de este proyecto fue un verdadero desafío, debido a la responsabilidad asumida, y a la importancia que el mismo tiene para la entidad que solicito mis servicios.

El trabajo se realizó a través del aporte de conocimientos y soluciones concretas, que incrementó mis conocimientos y afianzó mi vinculación con el ámbito profesional.

En el desarrollo de este proyecto, he hecho valer el conocimiento adquirido en la facultad. Desde la realización de entrevistas para la recolección de requerimientos y análisis del dominio, el diseño de la arquitectura del sistema y la implementación de este me ha permitido poner en práctica diversas cualidades indispensables para el desarrollo personal, como son la perseverancia para la superación de obstáculos, el entusiasmo con los clientes, etc.

Creo que es aquí donde la educación académica encuentra su objetivo: la formación sólida de profesionales capacitados para abordar proyectos, que brinden soluciones de software a las instituciones y comercios.

El proyecto del sistema de gestión para mutuales tiene posibilidades de ser ampliado, para ser una solución real de software y que las entidades mutuales, puedan utilizarlo para aprovechar todas las ventajas que se fueron mencionando a lo largo del documento con las tecnologías elegidas.

## *5.2.* Trabajos Futuros

Durante el desarrollo del sistema se descubrieron posibilidades para expandir las funcionalidades que el mismo soporta. A continuación, se detallan algunas de las que se propone desarrollar para mejorar el sistema:

**Módulo de Préstamos**: agregar toda la funcionalidad necesaria para que la mutual pueda registrar los préstamos que otorga, imprimir la documentación de los mismos, gestionar los préstamos en mora, dar soporte al cobro de los mismos, etc.

**Módulo de Servicios cuotizados:** agregar la funcionalidad necesaria para dar soporte a servicios que la mutual ofrece, por los cuales realiza una cobranza mensual.

**Soporte para impresiones de documentos:** agregar todo el soporte de software necesario para imprimir los formularios y comprobantes de caja que maneja la mutual.

## *5.3.* Referencias bibliográficas


- [1] Sitio Web oficial HTML: http://www.w3.org/html/
- [2] Sitio Web oficial CSS https://www.w3schools.com/css/
- [3] Wiki de JavaScript: http://en.wikipedia.org/wiki/JavaScript
- [4] Sitio Web de TypeScript https://www.typescriptlang.org/
- [5] Sitio Web oficial Spring Boot https://spring.io/projects/spring-boot
- [6] Sitio Web oficial Apache: http://www.apache.org/
- [7] Sitio Web oficial Angular https://angular.io/
- [8] Sitio Web oficial Angular Material https://material.angular.io/
- [9] Sitio Web oficial PostgreSQL https://www.postgresql.org/
- [10] Sitio Web oficial Visual Studio Code https://code.visualstudio.com/

- [11] Sitio Web oficial Heroku https://devcenter.heroku.com/
- [12] Wiki Java https://es.wikipedia.org/wiki/Java_(lenguaje_de_programacion)
- [13] Sitio Web oficial GIT: http://git-scm.com/
- [14] Sitio Web oficial https://firebase.google.com/
- [15] Wiki JSON: http://es.wikipedia.org/wiki/JSON
- [16] Sitio Web oficial de BitBucket: https://bitbucket.org/

# Anexos

## Anexo 1: Glosario de Términos

**Mutual:** Son asociaciones constituidas libremente sin fines de lucro por personas inspiradas en la solidaridad, con el objeto de brindarse ayuda recíproca frente a riesgos eventuales o de concurrir a su bienestar material y espiritual, mediante una contribución periódica.

**Mutualismo:** El Mutualismo es un sistema solidario de servicios mutuos, fundado en la asociación voluntaria de personas que se unen sobre la base de objetivos comunes de ayuda recíproca.

**La Mutual:** es la forma de referirse en el documento a la Mutual General Obligado.

**Directivo**: una asociación mutual tiene una estructura legal que cuenta con varios directivos, entre ellos un presidente, secretario, tesorero, vocales titulares y suplentes, etc.

**Socio**: también denominado asociado, es cualquier persona física o jurídica que haya solicitado ser un asociado en la asociación mutual, y que esta última, a través de sus directivos hayan aprobado dicha solicitud.

**Ayuda Económica**: también denominado préstamo, es una ayuda de dinero que la asociación mutual entrega a sus asociados, debiendo estos últimos devolver dicho dinero en un plazo pactado, más los intereses pactados, así como también, impuestos y gastos generados en la entrega de esta.

**Liquidación Ayuda Económica**: es el proceso por el cual se generan los documentos de la ayuda económica que firmaran el o los asociados y la cual producirá una transacción comercial, que deberá ser confirmada en la caja para que se produzca el egreso del dinero correspondiente de la dicha caja.

**Saldo impago de Ayuda Económica**: es el saldo aún le queda por devolver a los asociados responsables de la ayuda económica entregada por parte de la asociación mutual.

**Pago Ayuda Económica**: es el proceso de pago de una ayuda económica que genera una transacción comercial, que debe ser confirmada en caja para producirse el ingreso del dinero en la caja de la asociación mutual.

**Ahorro a Término:** son los ahorros que los asociados pueden depositar en la mutual a un plazo determinado (30, 60, 90, o más días). Los mismos devengan una tasa estímulo que es distinta por prestación, monto y plazo.

**Caja de Ahorros**: es una cuenta de ahorros para los socios de la mutual. En esas cuentas los socios pueden realizar depósitos o extracciones. Estas extracciones se pueden realizar a través de la caja en el edificio de la entidad, como así también por transferencia bancaria a una cuenta bancaria de dicho asociado.

**Depósito de Caja de ahorros**: es el proceso por el cual se acredita dinero en la caja de ahorros del socio, y que genera un comprobante en caja para producirse el ingreso del dinero correspondiente.

**Extracción de Caja de ahorros**: es el proceso por el cual se debita dinero de la caja de ahorros del socio, y que genera un comprobante en caja para producirse el egreso del dinero correspondiente.

**Responsable de Caja de ahorros**: es el socio que está autorizado a realizar depósitos o extracciones de una caja de ahorros.

**Tasa acreedora**: es el porcentaje mensual que cobrará el socio por el saldo a favor que posee de dinero en la caja correspondiente.

**Tasa deudora**: es el porcentaje mensual que pagará el socio por el saldo negativo o deudor que posee en la cuenta.

**Caja**: es un puesto de caja donde se registran todos los comprobantes de la mutual y donde se producen todos los ingresos y egresos de dinero.

**Cajero**: Es el empleado responsable de operar en la caja, abrirla, cerrarla, cargar comprobantes, recibir y entregar dinero.

**Comprobante:** Es un documento que emite la entidad mutual y que prueba una transacción o contratación. Normalmente, el comprobante es un justificante que sirve para

demostrar que se ha realizado un depósito, transferencia, gasto, pago de ayuda económica, etc.…

**Comprobante Manual:** es un comprobante que el cajero genera de manera manual en la caja, eligiendo las cuentas donde va a imputar los movimientos y los importes correspondientes.

# Anexo 2: Especificación de casos de uso

| Identificador | | CDU 01: Realizar Autenticación |
|---|---|---|
| Descripción | | Este caso de uso deberá permitir autenticar a un usuario en el sistema e identificar el rol del usuario como un Administrador, Cajero o Comercial. Las tareas que el usuario podrá realizar dentro del sistema dependerán de esta autenticación. |
| Precondición | | El usuario deberá poseer las credenciales correctas. |
| Postcondición | | Se deberá presentar en la pantalla un mensaje indicando que inició sesión con éxito.<br>Se deberá dar acceso al sistema, ofreciendo las opciones de sistema de acuerdo con el rol que posee el usuario. |
| Flujo Básico | | |
| Paso | Actor | Descripción |
| 1 | Usuario | Ingresa a la opción “Iniciar Sesión” en la pantalla “Principal”. |
| 2 | Sistema | Presenta la pantalla “Inicio de Sesión” que posee un campo para el nombre de usuario y otro campo para la contraseña. Además, posee la opción “Login”. |
| 3 | Usuario | Ingresa sus credenciales, que son el nombre de usuario y la contraseña y selecciona la opción “Login”. |
| 4 | Sistema | Valida que las credenciales coinciden con un usuario almacenado en la base de datos. Luego muestra un mensaje que el usuario ha iniciado sesión correctamente.<br>También muestra la pantalla principal con las opciones habilitadas de acuerdo con el rol de dicho usuario. |
| 7 | | Fin del caso de uso. |
| Flujo Alternativo | | 1.1 Si no hay conexión a Internet, se termina el caso de uso.<br>4.1 El sistema identifica que hay campos obligatorios que están vacíos, entonces resalta dichos campos. Se vuelve al paso 3.<br>4.2 El sistema no encuentra un usuario que coincida con el nombre de usuario ingresado, entonces muestra un mensaje indicando que el usuario no existe en el sistema. Se vuelve al paso 3.<br>4.3 El sistema encuentra un usuario que coincida con el nombre de usuario ingresado, pero la contraseña no coincide, entonces muestra un mensaje indicando que la contraseña es errónea. Se vuelve al paso 3. |
| Rendimiento | | El sistema deberá realizar las acciones descriptas entre 5 y 15 segundos. |
| Prioridad | | Imprescindible |

SOCIOS

| Identificador | | CDU 02: Alta de socio |
|---|---|---|
| Descripción | | Este caso de uso deberá permitir el registro de una persona como nuevo socio de la mutual. |
| Precondición | | Usuario inicio Sesión con el rol "Comercial o Administrador" |
| Postcondición | | Se deberá presentar en la pantalla un mensaje indicando la confirmación de que el socio fue registrado con éxito.<br>La información del nuevo socio deberá quedar almacenada. |
| Flujo Básico | | |
| Paso | Actor | Descripción |
| 1 | Comercial | Ingresa a la opción "Alta Socio" en la pantalla "Principal". |
| 2 | Sistema | Presenta la pantalla "Alta de Socios", de acuerdo con la estructura de datos definida en la sección datos de entrada y la opción "Confirmar" inhabilitada. Ver Datos de Entrada. |
| 3 | Comercial | Ingresa la información de la persona que se está dando de alta como socio, la cual se encuentra detallada en Datos de Entrada. |
| 4 | Sistema | Valida los datos de entrada de la persona y si están completos los campos obligatorios, entonces se habilita la opción "Confirmar". |
| 5 | Comercial | Selecciona la opción "Confirmar". |
| 6 | Sistema | Valida las siguientes opciones:<br>● Que el documento de la persona no exista entre los socios antes registrados.<br>Si la información registrada cumple con las validaciones entonces el sistema almacena la información y presenta un mensaje indicando que el registro fue exitoso. |
| 7 | | Fin del caso de uso. |
| Flujo Alternativo | | 1.1 Si no hay conexión a Internet, se termina el caso de uso.<br>4.1 El sistema identifica que hay campos obligatorios que están vacíos o presentan caracteres incompatibles, entonces resalta dichos campos. Se vuelve al paso 3.<br>6.1 El sistema identifica que el documento ya se encuentra registrado en el sistema con otro socio, por ende, se presenta un mensaje al usuario. Se vuelve al paso 3. |
| Rendimiento | | El sistema deberá realizar las acciones descriptas entre 5 y 15 segundos. |
| Prioridad | | Imprescindible |
| Datos de Entrada | | |
| Tipo Persona | | Corresponde a un campo tipo lista tendrá dos opciones seleccionables, Persona Física y Persona Jurídica. |
| Razón Social | | Corresponde a un campo editable donde se registrará la razón social de la persona jurídica que se está dando de alta como socio de la mutual. |
| Apellido | | Corresponde a un campo editable donde se registrará el apellido de la persona física que se está dando de alta como socio de la mutual. |
| Nombre | | Corresponde a un campo editable donde se registran los nombres de la persona física. |
| Tipo CUIT/CUIL | | Corresponde a un campo tipo lista que deberá contener los tipos de códigos de identificación tributaria admitidos en la República Argentina. |
| Número de CUIT | | Corresponde a un campo editable donde se registrará el número de CUIT de la persona. |
| Fecha de Nacimiento | | Corresponde a un campo editable donde se registrará la fecha de nacimiento de la persona física. |
| Sexo | | Corresponde a un campo tipo lista que deberá contener los tipos de documentos admitidos en la República Argentina. |
| Calle | | Corresponde a un campo editable donde se registrará la calle del domicilio. |

| | |
|---|---|
| Numero | Corresponde a un campo editable donde se registrará la altura de la calle del domicilio. |
| Localidad | Corresponde a un campo tipo lista que deberá contener los tipos de documentos admitidos en la República Argentina. |
| Número de contacto 1 | Corresponde a un campo editable donde se registrará un número de teléfono fijo o celular de contacto. |
| Número de contacto 2 | Corresponde a un campo editable donde se registrará un número de teléfono fijo o celular de contacto. |
| Observaciones | Corresponde a un campo editable donde se registrará cualquier otra información relacionada al socio. |

| Identificador | | CDU 03: Buscar socio |
|---|---|---|
| Descripción | | Este caso de uso deberá permitir encontrar un socio registrado y mostrar la información de este. |
| Precondición | | Usuario inicio Sesión con el rol "Comercial o Administrador" |
| Postcondición | | Se deberá presentar en la pantalla la información del socio buscado. |
| Flujo Básico | | |
| Paso | Actor | Descripción |
| 1 | Comercial | Ingresa a la opción "Buscar Socio" en la pantalla "Principal". |
| 2 | Sistema | Presenta la pantalla "Listar Socios", que posee un campo para filtrar a los socios, por apellidos y/o nombres. |
| 3 | Comercial | Ingresa los apellidos y/o nombres del socio y confirma la búsqueda. |
| 4 | Sistema | Busca los socios que posean nombres y apellidos que coincidan con los datos ingresados y devuelve una lista de socios. |
| 5 | Comercial | Selecciona el socio que estaba buscando. |
| 6 | Sistema | Presenta la pantalla "Información de Socio", donde se encuentran la información del socio, de acuerdo con "Datos de Entrada". |
| 7 | | Fin del caso de uso. |
| Flujo Alternativo | | 1.1 Si no hay conexión a Internet, se termina el caso de uso.<br>4.1 El campo con los datos ingresados se encuentra vacío entonces el sistema buscará todos los socios almacenados. Se va al paso 4.<br>4.2 El sistema busca los socios, pero el campo con los datos ingresados no coincide con algún socio, por lo tanto el resultado será una lista vacía. Se vuelve al paso 3. |
| Rendimiento | | El sistema deberá realizar las acciones descritas entre 1 y 4 segundos. |
| Prioridad | | Imprescindible |
| Datos de Entrada | | |
| Tipo Persona | | Corresponde a un campo automático no editable que deberá precargarse con el tipo de persona del socio consultado. |
| Razón Social | | Corresponde a un campo automático no editable que deberá precargarse con la razón social del socio consultado. |
| Apellido | | Corresponde a un campo automático no editable que deberá precargarse con el/los apellidos del socio consultado. |
| Nombre | | Corresponde a un campo automático no editable que deberá precargarse con el/los nombres del socio consultado. |
| Tipo de CUIT/CUIL | | Corresponde a un campo automático no editable que deberá precargarse con el tipo de CUIT del socio consultado. |
| Número de CUIT | | Corresponde a un campo automático no editable que deberá precargarse con el número CUIT del socio consultado. |
| Fecha de Nacimiento | | Corresponde a un campo automático no editable que deberá precargarse con la fecha de nacimiento del socio consultado. |
| Sexo | | Corresponde a un campo automático no editable que deberá precargarse con el sexo del socio consultado. |

| | |
|---|---|
| Calle | Corresponde a un campo automático no editable que deberá precargarse con el nombre de la calle del socio consultado. |
| Número | Corresponde a un campo automático no editable que deberá precargarse con el número de la altura de la calle del socio consultado. En caso de que no posea datos, permanecerá vacío. |
| Localidad | Corresponde a un campo automático no editable que deberá precargarse con el nombre de la localidad del socio consultado. |
| Número de contacto 1 | Corresponde a un campo automático no editable que deberá precargarse con el número de contacto 1 del socio consultado. |
| Número de contacto 2 | Corresponde a un campo automático no editable que deberá precargarse con número de contacto 2 del socio consultado. |
| Observaciones | Corresponde a un campo automático no editable que deberá precargarse con las observaciones del socio consultado. |

| Identificador | | CDU 04: Modificar socio |
|---|---|---|
| Descripción | | Este caso de uso deberá permitir modificar la información de un socio y que la misma quede almacenada. |
| Precondición | | Usuario inició Sesión con el rol “Comercial o Administrador” y se ejecutó el caso de uso “Buscar Socio”. |
| Postcondición | | Se deberá presentar en la pantalla un mensaje indicando la confirmación de que el socio fue registrado con éxito.<br>La información del nuevo socio deberá quedar almacenada. |
| Flujo Básico | | |
| Paso | Actor | Descripción |
| 1 | Comercial | Ingresa a la opción “Modificar Socio” en la pantalla de “Información de socio”. |
| 2 | Sistema | Presenta la pantalla “Modificación de Socio”, donde se encuentran la información del socio, de acuerdo con “Datos de Entrada”. |
| 3 | Comercial | Modifica la información del socio, de acuerdo con “Datos de Entrada”. |
| 4 | Sistema | Valida los datos y si están completos y correctos los campos obligatorios, entonces se habilita la opción “Confirmar”. |
| 5 | Comercial | Selecciona la opción “Confirmar”. |
| 6 | Sistema | Valida las siguientes opciones:<br>● Que el documento de la persona no corresponda a otro asociado antes registrado.<br>Si la información registrada cumple con las validaciones entonces el sistema almacena la información y presenta un mensaje indicando que el registro fue exitoso. |
| 7 | | Fin del caso de uso. |
| Flujo Alternativo | | 1.1 Si no hay conexión a Internet, se termina el caso de uso.<br>4.1 El sistema identifica que hay campos obligatorios que están vacíos o presentan caracteres incompatibles, entonces resalta dichos campos. Se vuelve al paso 3.<br>6.1 El sistema identifica que el documento ya se encuentra registrado en el sistema con otro socio, por ende, se presenta un mensaje al usuario. Se vuelve al paso 3. |
| Rendimiento | | El sistema deberá realizar las acciones descritas entre 5 y 15 segundos. |
| Prioridad | | Imprescindible |
| Datos de Entrada | | |
| Tipo de Persona | | Corresponde a un campo automático editable que deberá precargarse con el tipo de persona del socio consultado. |
| Razón Social | | Corresponde a un campo automático editable que deberá precargarse con la razón social del socio consultado. |

| | |
|---|---|
| Apellido | Corresponde a un campo automático editable que deberá precargarse con el/los apellidos del socio consultado. |
| Nombre | Corresponde a un campo automático editable que deberá precargarse con el/los nombres del socio consultado. |
| Tipo de CUIT | Corresponde a un campo automático tipo lista que deberá precargarse con el tipo de CUIT del socio consultado. |
| Número de CUIT | Corresponde a un campo automático editable que deberá precargarse con el número CUIT del socio consultado. |
| Fecha de Nacimiento | Corresponde a un campo automático editable que deberá precargarse con la fecha de nacimiento del socio consultado. |
| Sexo | Corresponde a un campo automático tipo lista que deberá precargarse con el sexo del socio consultado. |
| Calle | Corresponde a un campo automático editable que deberá precargarse con el nombre de la calle del socio consultado. |
| Número | Corresponde a un campo automático editable que deberá precargarse con el número de la altura de la calle del socio consultado. En caso de que no posea datos, permanecerá vacío. |
| Localidad | Corresponde a un campo automático editable que deberá precargarse con el nombre de la localidad del socio consultado. |
| Número de contacto 1 | Corresponde a un campo automático editable que deberá precargarse con el número de contacto 1 del socio consultado. |
| Número de contacto 2 | Corresponde a un campo automático editable que deberá precargarse con número de contacto 2 del socio consultado. |
| Observaciones | Corresponde a un campo automático editable que deberá precargarse con las observaciones del socio consultado. |

| Identificador | | CDU 05: Baja socio |
|---|---|---|
| Descripción | | Este caso de uso deberá permitir dar de baja un socio, debiendo quedar almacenado con el estado “Baja”, para que en un futuro pueda volver a reincorporarlo. |
| Precondición | | Usuario inició Sesión con el rol “Comercial o Administrador”.<br>Se ejecutó el caso de uso “Buscar Socio”.<br>El socio existe en el sistema y el mismo se encuentra en estado “Activo” |
| Postcondición | | Se deberá presentar en la pantalla un mensaje indicando que el socio fue dado de baja con éxito.<br>La información del nuevo socio deberá quedar almacenada. |
| Flujo Básico | | |
| Paso | Actor | Descripción |
| 1 | Comercial | Ingresa a la opción “Baja de Socio” en la pantalla de “Información de socio”. |
| 2 | Sistema | Presenta un mensaje ofreciendo la posibilidad de confirmar la baja a través de dos opciones “Confirmar” o “Cancelar” |
| 3 | Comercial | Selecciona la opción “Confirmar”. |
| 3 | Sistema | El sistema almacena la baja del socio y presenta un mensaje indicando que el socio fue dado de baja con éxito. |
| 7 | | Fin del caso de uso. |
| Flujo Alternativo | | 1.1 Si no hay conexión a Internet, se termina el caso de uso.<br>3.2 Se selecciona la opción “Cancelar”, y se termina el caso de uso. |
| Rendimiento | | El sistema deberá realizar las acciones descriptas entre 1 y 4 segundos. |
| Prioridad | | Imprescindible |

| Identificador | CDU 06: Reincorporar socio |
|---|---|

| | | |
|---|---|---|
| Descripción | | Este caso de uso deberá permitir reincorporar un socio, debiendo quedar almacenado como un socio activo en el sistema para que pueda volver a recibir los servicios de la mutual. |
| Precondición | | Usuario inicio Sesión con el rol “Comercial o Administrador”.<br>Se ejecutó el caso de uso “Buscar Socio”.<br>El socio existe en el sistema y el mismo se encuentra en estado “Baja” |
| Postcondición | | Se deberá presentar en la pantalla un mensaje indicando que el socio fue reincorporado con éxito.<br>La información del nuevo socio deberá quedar almacenada. |
| Flujo Básico | | |
| Paso | Actor | Descripción |
| 1 | Comercial | Ingresa a la opción “Reincorporar |
| 2 | Sistema | Presenta un mensaje ofreciendo la posibilidad de confirmar a través de dos opciones “Confirmar” o “Cancelar” |
| 3 | Comercial | Selecciona la opción “Confirmar”. |
| 3 | Sistema | El sistema almacena el nuevo estado del socio como activo y presenta un mensaje indicando que el socio fue reincorporado con éxito. |
| 7 | | Fin del caso de uso. |
| Flujo Alternativo | | 1.1 Si no hay conexión a Internet, se termina el caso de uso.<br>3.2 Se selecciona la opción “Cancelar”, y se termina el caso de uso. |
| Rendimiento | | El sistema deberá realizar las acciones descriptas entre 1 y 4 segundos. |
| Prioridad | | Imprescindible |

## CAJAS DE AHORROS

| Identificador | | CDU 07: Alta Caja de Ahorros |
|---|---|---|
| Descripción | | Este caso de uso deberá permitir el registro de una caja de ahorros para un socio de la mutual. |
| Precondición | | Usuario inició Sesión con el rol “Comercial o Administrador”.<br>Socios Responsable existe en el sistema y no se encuentra dado de baja. |
| Postcondición | | Se deberá presentar en la pantalla un mensaje indicando la confirmación de que la caja de ahorros fue registrada con éxito.<br>La información deberá quedar almacenada. |
| Flujo Básico | | |
| Paso | Actor | Descripción |
| 1 | Comercial | Ingresa a la opción “Abrir Caja Ahorros” en la pantalla “Principal”. |
| 2 | Sistema | Presenta la pantalla “Alta y Modificación de Caja de Ahorros”, de acuerdo con la estructura de datos definida en la sección datos de entrada y la opción “Confirmar” inhabilitada. Ver Datos de Entrada. |
| 3 | Comercial | Uno a la vez, ingresa los apellidos y/o nombres del o los socios que van a ser los responsables de la caja de ahorros y confirma la búsqueda. |
| 4 | Sistema | Busca los socios que posean nombres y apellidos que coincidan con los datos ingresados y devuelve una lista de socios. |
| 5 | Comercial | Selecciona el socio que estaba buscando. |
| 6 | Sistema | Agrega en la pantalla los apellidos, nombres y cuits de cada uno de los socios que se van seleccionando como responsables de la caja de ahorros. |
| 7 | Comercial | Ingresa la tasa deudora y la tasa acreedora. |
| 8 | Sistema | Valida que las tasas sean números y valida que haya al menos un responsable, entonces se habilita la opción “Confirmar”. |
| 9 | Comercial | Selecciona la opción “Confirmar”. |
| 10 | Sistema | Valida las siguientes opciones:<br>● Que el socio no esté dado de baja. |

| | | |
|---|---|---|
| | | • Que haya sido seleccionado como responsable al menos un socio y como máximo 3 socios.<br>Si la información registrada cumple con las validaciones entonces el sistema almacena la información y presenta un mensaje indicando que el registro fue exitoso. |
| 11 | | Fin del caso de uso. |
| Flujo Alternativo | | 1.1 Si no hay conexión a Internet, se termina el caso de uso.<br>8.1 El sistema identifica que hay campos obligatorios que están vacíos o presentan caracteres incompatibles, entonces resalta dichos campos. Se vuelve al paso 3. |
| Rendimiento | | El sistema deberá realizar las acciones descriptas entre 2 y 5 segundos. |
| Prioridad | | Imprescindible |
| Datos de Entrada | | |
| Apellido | | Corresponde a un campo editable donde se registrará el apellido de la persona que se está dando de alta como socio de la mutual. |
| Socio | | Corresponde a un campo automático tipo lista no editable donde se precargará con los asociados responsables de la caja de ahorros. Cada línea contendrá el rol del asociado con la caja de ahorros, apellido, nombre y CUIT del socio. |
| Tasa Deudora | | Corresponde a un campo editable donde se registrará la tasa que abonará el Socio cuando la caja de ahorros se encuentre con saldo negativo. |
| Tasa Acreedora | | Corresponde a un campo editable donde se registrará la tasa que se le abonará al Socio cuando la caja de ahorros se encuentre con saldo positivo. |
| Nro. de Caja de Ahorros | | Corresponde a un campo automático no editable donde el sistema rellenará con el número de caja de ahorros. |

| Identificador | | CDU 08: Buscar Caja de Ahorros |
|---|---|---|
| Descripción | | Este caso de uso deberá permitir encontrar una caja de ahorros registrada y mostrar la información de esta. |
| Precondición | | Usuario inicio Sesión con el rol “Comercial o Administrador” |
| Postcondición | | Se deberá presentar en la pantalla la información de la caja buscada. |
| Flujo Básico | | |
| Paso | Actor | Descripción |
| 1 | Comercial | Ingresa a la opción “Buscar Caja de Ahorros” en la pantalla “Principal”. |
| 2 | Sistema | Presenta la pantalla “Información de Caja de Ahorros”, que posee un campo para filtrar a los responsables, por apellidos y/o nombres. |
| 3 | Comercial | Ingresa los apellidos y/o nombres del responsable de la caja de ahorros que pretende buscar y confirma la búsqueda. |
| 4 | Sistema | Busca las cajas de ahorros, cuyos socios responsables posean nombres y apellidos que coincidan con los datos ingresados y devuelve una lista indicando número de caja, rol en la caja, apellidos, nombres y CUIT. |
| 5 | Comercial | Selecciona la caja que estaba buscando. |
| 6 | Sistema | Presenta en la pantalla la información obtenida de la caja de ahorros, de acuerdo con “Datos de Entrada”. |
| 7 | | Fin del caso de uso. |
| Flujo Alternativo | | 1.1 Si no hay conexión a Internet, se termina el caso de uso.<br>4.1 El sistema busca los socios, pero campo con los datos ingresados no coincide con algún socio, por lo tanto el resultado será una lista vacía. Se vuelve al paso 3. |
| Rendimiento | | El sistema deberá realizar las acciones descriptas entre 1 y 4 segundos. |
| Prioridad | | Imprescindible |
| Datos de Entrada | | |

| | |
|---|---|
| Apellido | Corresponde a un campo editable donde se registrará el apellido del socio responsable de la caja que se está buscando. |
| Responsables | Corresponde a un campo automático no editable tipo lista donde se precargará con los asociados responsables de la caja, Cada línea contendrá el rol del asociado con la caja, apellido, nombre y CUIT del socio. |
| Tasa Deudora | Corresponde a un campo automático no editable que se precargará con la tasa que abona el Socio cuando la caja de ahorros se encuentre con saldo negativo. |
| Tasa Acreedora | Corresponde a un campo automático no editable que se precargará con la tasa que se le abona al Socio cuando la caja de ahorros se encuentre con saldo positivo. |
| Nro. de Caja de Ahorros | Corresponde a un campo automático no editable donde el sistema rellenará con el número de caja de ahorros. |
| Saldo de Caja | Corresponde a un campo automático no editable donde el sistema rellenará con el saldo que posee la caja de ahorros. |
| Movimientos | Corresponde a un campo automático no editable tipo lista donde se precargará con los movimientos (depósitos/extracciones) ordenados de forma cronológica. Cada línea contendrá la fecha de realización, una referencia que puede ser depósito o extracción, un número de comprobante, un importe, y una opción para poder dar de baja el movimiento. |

| Identificador | | CDU 09 Modificar Caja Ahorros |
|---|---|---|
| Descripción | | Este caso de uso deberá permitir cambiar los responsables de una caja de ahorros como así también las tasas y que dichas modificaciones queden almacenadas. |
| Precondición | | Usuario inició Sesión con el rol “Comercial o Administrador” y se ejecutó el caso de uso “Buscar Caja Ahorros”. |
| Postcondición | | ● Se deberá presentar en la pantalla un mensaje indicando que la información fue almacenada con éxito.<br>● La información deberá quedar almacenada. |
| Flujo Básico | | |
| Paso | Actor | Descripción |
| 1 | Comercial | Ingresa a la opción “Modificar Caja Ahorros” en la pantalla de “Información de Caja Ahorros”. |
| 2 | Sistema | Presenta la pantalla “Modificación de Caja Ahorros”, de acuerdo con “Datos de Entrada”. |
| 3 | Comercial | Para agregar uno a la vez los nuevos responsables de la caja de ahorros, ingresa los apellidos y/o nombres de cada socio y confirma la búsqueda. |
| 4 | Sistema | Busca los socios que posean nombres y apellidos que coincidan con los datos ingresados y devuelve una lista. |
| 5 | Comercial | Selecciona el socio que estaba buscando. |
| 6 | Sistema | Por cada socio seleccionado, agrega en la pantalla los apellidos , nombres y cuits de cada uno de los socios. |
| 7 | Comercial | Ingresa la tasa deudora y la tasa acreedora. |
| 8 | Sistema | Valida que las tasas sean números y valida que haya al menos un responsable, entonces se habilita la opción “Confirmar”. |
| 9 | Comercial | Selecciona la opción “Confirmar”. |
| 10 | Sistema | Valida las siguientes opciones:<br>● Que el socio no esté dado de baja.<br>● Que haya sido seleccionado como responsable al menos un socio y como máximo 3 socios. |

| | | |
|---|---|---|
| | | Si la información registrada cumple con las validaciones entonces el sistema almacena la información y presenta un mensaje indicando que el registro fue exitoso. |
| 7 | | Fin del caso de uso. |
| Flujo Alternativo | | 1.1 Si no hay conexión a Internet, se termina el caso de uso.<br>3.1 El Comercial elige la opción "Eliminar" para el socio que desee quitar como responsable.<br>4.1 El sistema, quita la línea de la pantalla con la información del responsable eliminado, y además actualiza la información de los responsables.<br>8.1 El sistema identifica que hay campos obligatorios que están vacíos o presentan caracteres incompatibles, entonces resalta dichos campos. Se vuelve al paso 3. |
| Rendimiento | | El sistema deberá realizar las acciones descriptas entre 5 y 15 segundos. |
| Prioridad | | Imprescindible |
| Datos de Entrada | | |
| Apellido | | Corresponde a un campo editable donde se registrará el apellido del socio responsable de la caja de ahorros que se está buscando. |
| Responsables | | Corresponde a un campo automático tipo lista editable donde se precargará con los asociados responsables de la caja, Cada línea contendrá el rol del asociado con la caja de ahorros, apellido, nombre y CUIT del socio. |
| Tasa Deudora | | Corresponde a un campo editable automático que se precargará con la tasa que abona el Socio cuando la caja de ahorros se encuentre con saldo negativo. |
| Tasa Acreedora | | Corresponde a un campo editable automático que se precargará con la tasa que se le abona al Socio cuando la caja se encuentre con saldo positivo. |
| Nro. de Caja | | Corresponde a un campo automático no editable donde el sistema rellenará con el número de caja de ahorros. |

| Identificador | | CDU 10: Realizar Deposito / Extracción |
|---|---|---|
| Descripción | | Este caso de uso deberá permitir registrar un depósito o extracción, debiendo quedar almacenada tanto la información del movimiento como también actualizado el saldo de la caja de ahorros. |
| Precondición | | Usuario inició Sesión con el rol "Comercial o Administrador".<br>Se ejecutó el caso de uso "Buscar Caja de Ahorros".<br>La caja existe en el sistema y la misma se encuentra en estado "Activa" |
| Postcondición | | ● Se deberá presentar en la pantalla un mensaje indicando que el movimiento fue realizado con éxito.<br>● La información del movimiento y de la caja de ahorros deberán quedar almacenados. |
| Flujo Básico | | |
| Paso | Actor | Descripción |
| 1 | Comercial | Ingresa a la opción "Depósito " en la pantalla de "Información de caja". |
| 2 | Sistema | Presenta una interfaz con un campo donde se puede ingresar el importe del movimiento y las opciones "Cancelar" o "Confirmar". |
| 3 | Comercial | Ingresa el importe del movimiento correspondiente. |
| 4 | Sistema | Valida que el importe que sea un número positivo con decimales o entero. |
| 5 | Comercial | Selecciona la opción "Confirmar". |
| 6 | Sistema | Almacena el movimiento realizado y presenta un mensaje indicando que el movimiento fue realizado con éxito. |

| | | |
|---|---|---|
| 7 | Sistema | Agrega a la pantalla una línea con el nuevo movimiento y actualiza en pantalla el nuevo saldo de la caja de ahorros. |
| 8 | | Fin del caso de uso. |
| Flujo Alternativo | | 1.1 Si no hay conexión a Internet, se termina el caso de uso.<br>1.2 El Comercial ingresa la opción "Extracción" en la pantalla de "Información de caja".<br>4.1 El sistema identifica que el campo del importe se encuentra vacío o presenta caracteres incompatibles, entonces resalta dichos campos. Se vuelve al paso 3.<br>4.2 El sistema identifica que el saldo de la caja de ahorros queda negativo y entonces muestra un mensaje de error.<br>5.1 El comercial elige la opción "Cancelar" y el caso de uso se termina. |
| Rendimiento | | El sistema deberá realizar las acciones descriptas entre 1 y 4 segundos. |
| Prioridad | | Imprescindible |

| Identificador | | CDU 11: Anular Deposito / Extracción |
|---|---|---|
| Descripción | | Este caso de uso deberá permitir anular un movimiento, debiendo generarse un nuevo movimiento llamado "anulación" que compense el movimiento "anulado". También debe actualizarse el saldo de la caja de ahorros. |
| Precondición | | Usuario inició Sesión con el rol "Comercial o Administrador".<br>Se ejecutó el caso de uso "Buscar Caja de Ahorros".<br>La caja existe en el sistema y la misma se encuentra en estado "Activa", por ende, se muestra la información en pantalla de acuerdo con el caso de uso "Buscar Caja de Ahorros" |
| Postcondición | | Se deberá presentar en la pantalla un mensaje indicando que el movimiento fue anulado con éxito.<br>El movimiento anulado deberá quedar en estado "anulado" y debe indicarse en pantalla dicho estado.<br>Se debe generar un nuevo movimiento de anulación que tendrá un importe que compensará al movimiento anulado y debe indicarse en pantalla dicho estado.<br>El saldo de la caja quedará actualizado. |
| Flujo Básico | | |
| Paso | Actor | Descripción |
| 1 | Administrador | Elije el movimiento de la caja de ahorro que desee eliminar, y luego ingresa a la opción "Eliminar" en dicho movimiento. |
| 2 | Sistema | Presenta una interfaz, donde solicita que el usuario confirme la voluntad de anular el movimiento. De este modo, ofrece dos opciones "Cancelar" o "Confirmar". |
| 3 | Administrador | Selecciona la opción "Confirmar". |
| 4 | Sistema | Genera un nuevo movimiento de anulación de caja de ahorros, que compensará el movimiento anulado. También al movimiento anulado, lo cambia a estado "anulado".<br>Almacena el movimiento de anulación y el movimiento anulado. |
| 7 | Sistema | Agrega a la pantalla una línea con el nuevo movimiento de anulación y actualiza en pantalla el nuevo saldo de la caja de ahorros. |
| 8 | | Fin del caso de uso. |
| Flujo Alternativo | | 1.1 Si no hay conexión a Internet, se termina el caso de uso.<br>3.1 El Administrador ingresa la opción "Cancelar" en la pantalla de "Información de caja de ahorros" y el caso de uso termina. |
| Rendimiento | | El sistema deberá realizar las acciones descriptas entre 1 y 4 segundos. |
| Prioridad | | Imprescindible |

| Identificador | | CDU 12: Habilitar Caja de Ahorros |
|---|---|---|
| Descripción | | Este caso de uso deberá permitir habilitar una caja de ahorros, debiendo quedar almacenado con estado habilitada. |
| Precondición | | Usuario inició Sesión con el rol “Comercial o Administrador”.<br>Se ejecutó el caso de uso “Buscar Caja de Ahorros”.<br>La caja existe en el sistema y la misma se encuentra en estado “Deshabilitada” |
| Postcondición | | Se deberá presentar en la pantalla un mensaje indicando que la caja fue habilitada con éxito.<br>La información de la caja deberá quedar almacenada como habilitada. |
| Flujo Básico | | |
| Paso | Actor | Descripción |
| 1 | Comercial | Ingresa a la opción “Baja de Caja de Ahorros” en la pantalla de “Información de caja de ahorros”. |
| 2 | Sistema | Presenta un mensaje ofreciendo la posibilidad de confirmar la baja a través de dos opciones “Confirmar” o “Cancelar” |
| | | |
| 3 | Comercial | Selecciona la opción “Confirmar”. |
| 3 | Sistema | El sistema almacena la baja y presenta un mensaje indicando que la caja fue habilitada con éxito. |
| 7 | | Fin del caso de uso. |
| Flujo Alternativo | | 1.1 Si no hay conexión a Internet, se termina el caso de uso.<br>3.2 Se selecciona la opción “Cancelar”, y se termina el caso de uso. |
| Rendimiento | | El sistema deberá realizar las acciones descriptas entre 1 y 4 segundos. |
| Prioridad | | Imprescindible |

| Identificador | | CDU 13: Deshabilitar Caja de Ahorros |
|---|---|---|
| Descripción | | Este caso de uso deberá permitir deshabilitar una caja de ahorros, debiendo quedar almacenado con estado deshabilitada para que en un futuro pueda volver a habilitarla. |
| Precondición | | Usuario inició Sesión con el rol “Comercial o Administrador”.<br>Se ejecutó el caso de uso “Buscar Caja de Ahorros”.<br>La caja existe en el sistema y la misma se encuentra en estado “Habilitada” |
| Postcondición | | Se deberá presentar en la pantalla un mensaje indicando que la caja fue deshabilitada con éxito.<br>La información de la caja deberá quedar almacenada como Deshabilitada. |
| Flujo Básico | | |
| Paso | Actor | Descripción |
| 1 | Comercial | Ingresa a la opción “Baja de Caja de Ahorros” en la pantalla de “Información de caja”. |
| 2 | Sistema | Presenta un mensaje ofreciendo la posibilidad de confirmar la baja a través de dos opciones “Confirmar” o “Cancelar” |
| | | |
| 3 | Comercial | Selecciona la opción “Confirmar”. |
| 3 | Sistema | El sistema almacena la baja y presenta un mensaje indicando que la caja fue deshabilitada con éxito. |
| 7 | | Fin del caso de uso. |
| Flujo Alternativo | | 1.1 Si no hay conexión a Internet, se termina el caso de uso.<br>3.2 Se selecciona la opción “Cancelar”, y se termina el caso de uso. |
| Rendimiento | | El sistema deberá realizar las acciones descriptas entre 1 y 4 segundos. |

| | |
|---|---|
| Prioridad | Imprescindible |

| Identificador | | CDU 14: Listar Movimientos Caja de Ahorros |
|---|---|---|
| Descripción | | Este caso de uso deberá permitir mostrar los comprobantes de cajas de ahorros entre dos fechas determinadas. |
| Precondición | | Usuario inició Sesión con el rol “Administrador” |
| Postcondición | | Se deberá presentar en la pantalla un listado de los comprobantes de cajas de ahorros que fueron confirmados y almacenados entre las fechas especificadas. |
| Flujo Básico | | |
| Paso | Actor | Descripción |
| 1 | Administrador | Ingresa a la opción “Listado por Fecha” en la pantalla “Principal”. |
| 2 | Sistema | Presenta la pantalla “Listar Movimientos Cajas de Ahorros”, que posee dos campos para filtrar a los comprobantes entre dos fechas determinadas. |
| 3 | Administrador | Ingresa las dos fechas entre las cuales, desea ver los movimientos y confirma la búsqueda. |
| 4 | Sistema | Busca los movimientos de todas las cajas de ahorros que posean fechas de creación que se encuentren entre las fechas que eligió el usuario y devuelve una lista de los movimientos encontrados. Cada línea de la lista debe contener fecha de creación, una referencia, el número y el importe de este, y en caso de no ser un comprobante anulado o de anulación entonces se habilita una opción para anularlo. |
| 7 | | Fin del caso de uso. |
| Flujo Alternativo | | 1.1 Si no hay conexión a Internet, se termina el caso de uso.<br>4.1 El campo con una de las fechas no contiene una fecha válida o se encuentra vacío entonces el sistema resaltará dicho campo y se volverá al paso 3.<br>4.2 El sistema busca los comprobantes que coincidan con los datos ingresados, pero no existe ningún comprobante con esos datos, por los tanto el resultado será una lista vacía. Se vuelve al paso 2. |
| Rendimiento | | El sistema deberá realizar las acciones descriptas entre 1 y 4 segundos. |
| Prioridad | | Imprescindible |

## CAJA

| Identificador | | CDU 15: Mostrar resumen de caja del día |
|---|---|---|
| Descripción | | Este caso de uso deberá permitir mostrar los comprobantes confirmados en la caja en el día de la fecha. Además, deberá mostrar el saldo del inicio y final del día. |
| Precondición | | Usuario inicio Sesión con el rol “Cajero o Administrador” |
| Postcondición | | Se deberá presentar en la pantalla un listado de los comprobantes que fueron confirmados en el día de la fecha y los saldos de inicio y final de caja. |
| Flujo Básico | | |
| Paso | Actor | Descripción |
| 1 | Cajero | Ingresa a la opción “Resumen de caja” en la pantalla “Principal”. |
| 2 | Sistema | Presenta la pantalla “Resumen de caja del día”, que posee un saldo inicial con el que se comenzó el día, una lista de comprobantes confirmados en la caja y un saldo final luego de aplicarse al saldo final los comprobantes del día. |

| | | |
|---|---|---|
| | | Cada línea de la lista debe contener el número del comprobante, el título de referencia, la fecha y hora de creación, el importe del comprobante. |
| 7 | | Fin del caso de uso. |
| Rendimiento | | El sistema deberá realizar las acciones descriptas entre 1 y 4 segundos. |
| Prioridad | | Imprescindible |

| Identificador | | CDU 16: Registrar Comprobante Manual |
|---|---|---|
| Descripción | | Este caso de uso deberá permitir registrar un comprobante de caja, debiendo quedar almacenada tanto a la información del comprobante como también actualizado el saldo de caja. |
| Precondición | | Usuario inicio Sesión con el rol "Cajero o Administrador". |
| Postcondición | | Se deberá presentar en la pantalla un mensaje indicando que la operación fue realizada con éxito.<br>La información de la operación y el saldo de la caja deberán quedar almacenados. |
| Flujo Básico | | |
| Paso | Actor | Descripción |
| 1 | Cajero | Ingresa a la opción "Registrar Comprobante" en la pantalla de "Generar Comprobante". |
| 2 | Sistema | Presenta la pantalla de "Registrar Comprobante Manual "con un campo donde se debe ingresar el título del comprobante. También posee dos opciones "Ingreso" y "Egreso". |
| 3 | Cajero | Ingresa una descripción del título del comprobante.<br>Por cada uno de los movimientos del comprobante, elige la opción "Ingreso" o "Egreso". |
| 4 | Sistema | Por cada movimiento presenta una interfaz con un campo donde se puede ingresar el importe del movimiento, y otro elegir la cuenta contable en la que se imputará dicho movimiento. También presenta las opciones de "Confirmar" y "Cancelar", para agregar el movimiento al comprobante, o para cancelar el movimiento. |
| 5 | Cajero | Por cada movimiento, ingresa el importe del movimiento correspondiente y también elige la cuenta contable en la que se debe imputar el movimiento. Luego elije la opción "Confirmar" |
| 6 | Sistema | Por cada vez que el comercial confirma un movimiento, valida que el importe sea un número positivo con decimales o entero, y que se haya seleccionado una cuenta contable.<br>Luego agrega el movimiento a la lista de movimientos, del comprobante. |
| 7 | Cajero | Una vez ingresado todos los movimientos, selecciona la opción "Confirmar". |
| 8 | Sistema | Valida que el comprobante tenga completo el campo título y al menos un movimiento, luego almacena el comprobante realizado y presenta un mensaje indicando que el movimiento fue realizado con éxito. |
| 8 | | Fin del caso de uso. |
| Flujo Alternativo | | 1.1 Si no hay conexión a Internet, se termina el caso de uso.<br>3.1 El cajero elige eliminar un movimiento del comprobante, entonces el sistema lo quita de la lista de movimientos del comprobante, y actualiza el saldo del comprobante.<br>6.1 El sistema identifica que el campo del importe se encuentra vacío o presenta caracteres incompatibles, entonces resalta dichos campos. Se vuelve al paso 5.<br>6.1 El sistema identifica que el campo de la cuenta contable se encuentra vacío, entonces resalta dichos campos. Se vuelve al paso 5.<br>5.1 El comercial elige la opción "Cancelar" y el caso de uso se termina. |

| | |
|---|---|
| | 8.1 El campo de título se encuentra incompleto, entonces se resalta dicho campo y se vuelve al paso 3.<br>8.2 El comercial no confirmó ningún movimiento, entonces el sistema muestra un mensaje que indica que no hay movimientos, y se vuelve al paso 5. |
| Rendimiento | El sistema deberá realizar las acciones descriptas entre 1 y 4 segundos. |
| Prioridad | Imprescindible |
| Datos de Entrada | |
| Título del Comprobante | Corresponde a un campo editable donde se registrará un título que será una referencia del comprobante. |
| Lista de Movimientos | Corresponde a un campo automático no editable lista donde se precargará con los comprobantes de caja confirmados en el día de la fecha. Cada línea contendrá el nombre de la cuenta contable del movimiento, el importe y una opción para dar de baja el movimiento. |

| Identificador | | CDU 17: Anular Comprobante |
|---|---|---|
| Descripción | | Este caso de uso deberá permitir anular un comprobante, debiendo generarse un nuevo movimiento llamado “anulación” que compense el comprobante “anulado”. También debe actualizarse el saldo de la caja. |
| Precondición | | Usuario inició Sesión con el rol “Cajero o Administrador”.<br>El usuario, uso el caso de uso “Listar Comprobantes” o “Mostrar Resumen de caja del día”. |
| Postcondición | | Se deberá presentar en la pantalla un mensaje indicando que el comprobante fue anulado con éxito.El comprobante anulado deberá quedar en estado “anulado” y debe indicarse en pantalla dicho estado.<br>Se debe generar un nuevo comprobante de anulación que tendrá un importe que compensará al comprobante anulado y debe indicarse en pantalla dicho estado.<br>El saldo de la caja quedará actualizado. |
| Flujo Básico | | |
| Paso | Actor | Descripción |
| 1 | Administrador | Hace uso del caso de uso “Listar Comprobantes” entre dos fechas determinadas. |
| 2 | Sistema | Presenta una interfaz con una lista de comprobantes que fueron confirmados en la caja de acuerdo con el caso de uso “Listar Comprobantes”. |
| 3 | Administrador | Elige de la lista el comprobante que desea anular. |
| 4 | Sistema | El sistema, valida que el comprobante no esté en estado “anulado” o que sea un comprobante de “anulación”.<br>Presenta un mensaje con dos opciones “Confirmar” o “Cancelar” |
| 5 | Administrador | Selecciona la opción “Confirmar”. |
| 6 | Sistema | Cambia el comprobante al estado de anulado. Luego genera un nuevo comprobante de anulación cuyos movimientos compensan al comprobante anulado y lo registra en los comprobantes de la caja del día de la anulación, donde también actualiza el saldo de la caja.<br>Luego presenta en pantalla un mensaje indicando que la anulación fue exitosa. |
| 7 | | Fin del caso de uso. |
| Flujo Alternativo | | 1.1 Si no hay conexión a Internet, se termina el caso de uso.<br>4.1 El sistema identifica que el comprobante se encontraba en estado “anulado” o es un comprobante “anulación”, por ello, muestra un mensaje donde indica que no se puede anular dicho comprobante y se vuelve al paso 2. |

| | |
|---|---|
| | 5.1 El Administrador elige la opción “Cancelar” y se vuelve al paso 2. |
| Rendimiento | El sistema deberá realizar las acciones descriptas entre 1 y 4 segundos. |
| Prioridad | Imprescindible |

| Identificador | | CDU 18: Listar Comprobantes |
|---|---|---|
| Descripción | | Este caso de uso deberá permitir mostrar los comprobantes confirmados en la caja entre dos fechas determinadas. |
| Precondición | | Usuario inicio Sesión con el rol “Administrador” |
| Postcondición | | Se deberá presentar en la pantalla un listado de los comprobantes que fueron confirmados y almacenados entre las fechas especificadas. |
| Flujo Básico | | |
| Paso | Actor | Descripción |
| 1 | Administrador | Ingresa a la opción “Listar Comprobantes” en la pantalla “Principal”. |
| 2 | Sistema | Presenta la pantalla “Listar Comprobantes”, que posee dos campos para filtrar a los comprobantes entre dos fechas de creación determinadas. |
| 3 | Administrador | Ingresa las dos fechas entre las cuales, desea ver qué comprobantes hay confirmados en la caja y confirma la búsqueda. |
| 4 | Sistema | Busca los comprobantes que posean fechas de creación que se encuentren entre las fechas que eligió el usuario y devuelve una lista de los comprobantes encontrados. Cada línea de la lista debe contener el número del comprobante, el título de referencia, la fecha y hora de creación, el importe del comprobante, y en caso de no ser un comprobante anulado o de anulación entonces se habilita una opción para anularlo. |
| 7 | | Fin del caso de uso. |
| Flujo Alternativo | | 1.1 Si no hay conexión a Internet, se termina el caso de uso.<br>4.1 El campo con una de las fechas no contiene una fecha válida o se encuentra vacío entonces el sistema resaltará dicho campo y se volverá al paso 3.<br>4.2 El sistema busca los comprobantes que coincidan con los datos ingresados, pero no existe ningún comprobante con esos datos, por los tanto el resultado será una lista vacía. Se vuelve al paso 2. |
| Rendimiento | | El sistema deberá realizar las acciones descriptas entre 1 y 4 segundos. |
| Prioridad | | Imprescindible |

## CONTABILIDAD

| Identificador | | CDU 19: Listar Cuentas Contables |
|---|---|---|
| Descripción | | Este caso de uso deberá permitir mostrar una lista con las cuentas contables registradas en el sistema. |
| Precondición | | Usuario inicio Sesión con el rol “Administrador” |
| Postcondición | | • Se deberá presentar en la pantalla un listado de las cuentas contables registradas. |
| Flujo Básico | | |
| Paso | Actor | Descripción |
| 1 | Cajero | Ingresa a la opción “Listar Cuentas” en la pantalla “Principal”. |
| 2 | Sistema | Presenta la pantalla “Listar de Cuentas Contables”, que posee una lista de cuentas contables. Cada línea de la lista contendrá el número de cuenta, el nombre completo de la cuenta, una opción para editar la cuenta, y otra para agregar una cuenta dentro de su grupo de cuentas. |
| 7 | | Fin del caso de uso. |
| Flujo Alternativo | | 1.1 Si no hay conexión a Internet, se termina el caso de uso. |
| Rendimiento | | El sistema deberá realizar las acciones descriptas entre 1 y 4 segundos. |

| | |
|---|---|
| Prioridad | Imprescindible |

| Identificador | | CDU 20: Mostrar Saldos de Cuentas Mensual |
|---|---|---|
| Descripción | | Este caso de uso deberá permitir mostrar los totales de movimientos acreedores y deudores, y los saldos de los movimientos de las cuentas contables, generados en un mes determinado. |
| Precondición | | Usuario inicio Sesión con el rol "Administrador" |
| Postcondición | | Se deberá presentar en la pantalla un listado donde se identifique a cada una de las cuentas contables, y por cada una de ellas, detallar los totales de los movimientos acreedores y deudores, y el saldo correspondiente en el mes. |
| Flujo Básico | | |
| Paso | Actor | Descripción |
| 1 | Administrador | Ingresa a la opción "Balance Mensual" en la pantalla "Principal". |
| 2 | Sistema | Presenta la pantalla "Balance Mensual de Saldos de Cuentas Contables", que posee un campo para filtrar los movimientos correspondientes a un mes determinado. |
| 3 | Administrador | Ingresa el mes que desea consultar y selecciona la opción "Mostrar". |
| 4 | Sistema | Busca todas las cuentas contables, y por cada una de ellas, busca todos los movimientos de cuentas que se crearon durante el mes seleccionado. Para cada cuenta totaliza los movimientos acreedores, deudores y el saldo de la cuenta del mes. |
| 5 | Sistema | Presenta en pantalla una lista, donde cada línea corresponde a cada una de las cuentas contables registradas.<br>Las cuentas estarán ordenadas por número de cuenta.<br>Cada línea de la lista contendrá el número de cuenta, el nombre completo de la cuenta, el total de movimientos acreedores, el total de movimientos deudores y el saldo resultante de la resta de los dos totales antes mencionados. |
| 6 | | Fin del caso de uso. |
| Flujo Alternativo | | 1.1 Si no hay conexión a Internet, se termina el caso de uso.<br>4.1 El campo de fechas no contiene una fecha válida o se encuentra vacío entonces el sistema resaltará dicho campo y se volverá al paso 2. |
| Rendimiento | | El sistema deberá realizar las acciones descriptas entre 1 y 4 segundos. |
| Prioridad | | Imprescindible |

| Identificador | | CDU 21: Alta de cuenta contable |
|---|---|---|
| Descripción | | Este caso de uso deberá permitir el registro de una nueva cuenta contable. |
| Precondición | | Usuario inicio Sesión con el rol "Administrador" |
| Postcondición | | Se deberá presentar en la pantalla un mensaje indicando la confirmación de que la cuenta fue registrada con éxito.<br>La información de la nueva cuenta deberá quedar almacenada. |
| Flujo Básico | | |
| Paso | Actor | Descripción |
| 1 | Comercial | Ingresa a la opción "Alta Cuenta contable" en la línea de la cuenta contable más general en la jerarquía de cuentas de la pantalla "Listado de Cuentas Contables". |
| 2 | Sistema | Presenta la pantalla "Alta de Cuenta", de acuerdo con la estructura de datos definida en la sección datos de entrada y la opción "Confirmar" inhabilitada. Ver Datos de Entrada. |
| 3 | Comercial | Ingresa la información de la cuenta que se está dando de alta, la cual se encuentra detallada en Datos de Entrada. |

| | | |
|---|---|---|
| 4 | Sistema | Valida los datos de la cuenta y si están completos los campos obligatorios, entonces se habilita la opción “Confirmar”. |
| 5 | Comercial | Selecciona la opción “Confirmar”. |
| 6 | Sistema | Valida las siguientes opciones:<br>● Que el número de cuenta no exista entre los números de cuentas antes registrados.<br>● Que el número de cuenta siga la relación de la jerarquía de cuentas de acuerdo con las reglas contables de las mutuales.<br>● Que todos los campos estén completos y correctos<br>Si la información registrada cumple con las validaciones entonces el sistema almacena la información y presenta un mensaje indicando que el registro fue exitoso. |
| 7 | | Fin del caso de uso. |
| Flujo Alternativo | | 1.1 Si no hay conexión a Internet, se termina el caso de uso.<br>4.1 El sistema identifica que hay campos obligatorios que están vacíos o presentan caracteres incompatibles, entonces resalta dichos campos. Se vuelve al paso 3.<br>6.1 El sistema identifica que la cuenta ya se encuentra registrada en el sistema con otra cuenta, por ende, se presenta un mensaje al usuario. Se vuelve al paso 3. |
| Rendimiento | | El sistema deberá realizar las acciones descriptas entre 5 y 15 segundos. |
| Prioridad | | Imprescindible |
| Datos de Entrada | | |
| Número de cuenta | | Corresponde a un campo editable donde se registrará el número de cuenta. |
| Nombre de cuenta | | Corresponde a un campo editable donde se registrará el nombre de la cuenta. |
| Imputable | | Corresponde a un campo tipo lista que deberá contener las opciones IMPUTABLE o NO IMPUTABLE. |
| Nivel en la jerarquía | | Corresponde a un campo no editable donde se mostrará el nivel dentro de la jerarquía de cuentas en que se situará la cuenta. |

| Identificador | | CDU 22: Modificar cuenta contable |
|---|---|---|
| Descripción | | Este caso de uso deberá permitir modificar información de una cuenta contable. |
| Precondición | | Usuario inicio Sesión con el rol “Administrador” |
| Postcondición | | Se deberá presentar en la pantalla un mensaje indicando la confirmación de que la cuenta fue modificada con éxito.<br>La información de la cuenta deberá quedar almacenada. |
| Flujo Básico | | |
| Paso | Actor | Descripción |
| 1 | Comercial | Ingresa a la opción “Modificar Cuenta contable” en la línea de la cuenta contable de la pantalla “Listado de Cuentas Contables”. |
| 2 | Sistema | Presenta la pantalla “Modificar de Cuenta”, de acuerdo con la estructura de datos definida en la sección datos de entrada y la opción “Confirmar” inhabilitada. Ver Datos de Entrada. |
| 3 | Comercial | Modifica la información de la cuenta, la cual se encuentra detallada en Datos de Entrada. |
| 4 | Sistema | Valida los datos de la cuenta y si están completos los campos obligatorios, entonces se habilita la opción “Confirmar”. |
| 5 | Comercial | Selecciona la opción “Confirmar”. |
| 6 | Sistema | Valida las siguientes opciones:<br>● Que el número de cuenta no coincida con el número de otra cuenta antes registrada . |

| | | |
|---|---|---|
| | | ● Que el número de cuenta siga la relación de la jerarquía de cuentas de acuerdo con las reglas contables de las mutuales.<br>● Que todos los campos estén completos y correctos<br>Si la información registrada cumple con las validaciones entonces el sistema almacena la información y presenta un mensaje indicando que el registro fue exitoso. |
| 7 | | Fin del caso de uso. |
| Flujo Alternativo | | 1.1 Si no hay conexión a Internet, se termina el caso de uso.<br>4.1 El sistema identifica que hay campos obligatorios que están vacíos o presentan caracteres incompatibles, entonces resalta dichos campos. Se vuelve al paso 3.<br>6.1 El sistema identifica que la cuenta ya se encuentra registrada en el sistema con otra cuenta, por ende, se presenta un mensaje al usuario. Se vuelve al paso 3. |
| Rendimiento | | El sistema deberá realizar las acciones descriptas entre 5 y 15 segundos. |
| Prioridad | | Imprescindible |
| Datos de Entrada | | |
| Número de cuenta | | Corresponde a un campo editable donde se registrará el número de cuenta. |
| Nombre de cuenta | | Corresponde a un campo editable donde se registrará el nombre de la cuenta. |
| Imputable | | Corresponde a un campo tipo lista que deberá contener las opciones IMPUTABLE o NO IMPUTABLE. |
| Nivel en la jerarquía | | Corresponde a un campo no editable donde se mostrará el nivel dentro de la jerarquía de cuentas en que se situará la cuenta. |

| Identificador | | CDU 23: Eliminar Cuenta Contable |
|---|---|---|
| Descripción | | Este caso de uso deberá permitir eliminar una cuenta contable que no posee movimientos asociados. |
| Precondición | | Usuario inició Sesión con el rol “Administrador”.<br>El usuario, uso el caso de uso “Modificar Cuenta Contable”. |
| Postcondición | | Se deberá presentar en la pantalla un mensaje indicando que la cuenta fue eliminada con éxito. |
| Flujo Básico | | |
| Paso | Actor | Descripción |
| 1 | Administrador | Hace uso del caso de uso “Editar Cuenta Contable”. |
| 2 | Sistema | Presenta una interfaz con la información de la cuenta contable seleccionada. |
| 3 | Administrador | Elige la opción eliminar. |
| 4 | Sistema | El sistema, valida que la cuenta no posea movimientos de cuenta asociados. Presenta un mensaje con dos opciones “Confirmar” o “Cancelar” para volver a pedir la confirmación al usuario. |
| 5 | Administrador | Selecciona la opción “Confirmar”. |
| 6 | Sistema | Elimina la información registrada de dicha cuenta contable del sistema y vuelve al listado de cuentas actualizado donde ya no se encontrará la cuenta antes eliminada. |
| 7 | | Fin del caso de uso. |
| Flujo Alternativo | | 1.2 Si no hay conexión a Internet, se termina el caso de uso.<br>4.1 El sistema identifica que la cuenta posee movimientos asociados, por ello, muestra un mensaje donde indica que no se puede eliminar dicha cuenta y se vuelve al paso 2.<br>5.1 El Administrador elige la opción “Cancelar” y se vuelve al paso 2. |
| Rendimiento | | El sistema deberá realizar las acciones descriptas entre 1 y 4 segundos. |

| | |
|---|---|
| Prioridad | Imprescindible |

# Anexo 3: Diseño Preliminar de Interfaces de Usuario

En esta sección de detallaran las interfaces de graficas que utilizará el sistema, para interactuar con los usuarios en los casos de usos del sistema.

**Caso de uso 1: Autenticar usuario**

El objetivo de esta interfaz es permitir a un usuario autenticarse en el sistema. Para llevar a cabo este proceso, posee un campo de nombre de usuario, otro de password y un botón para enviar la información. El usuario deberá completar los campos y luego presionar Login. En caso de que los datos estén correctos, se le dará acceso al sistema con los permisos correspondientes. En caso contrario se indicarán cuáles campos poseen errores.

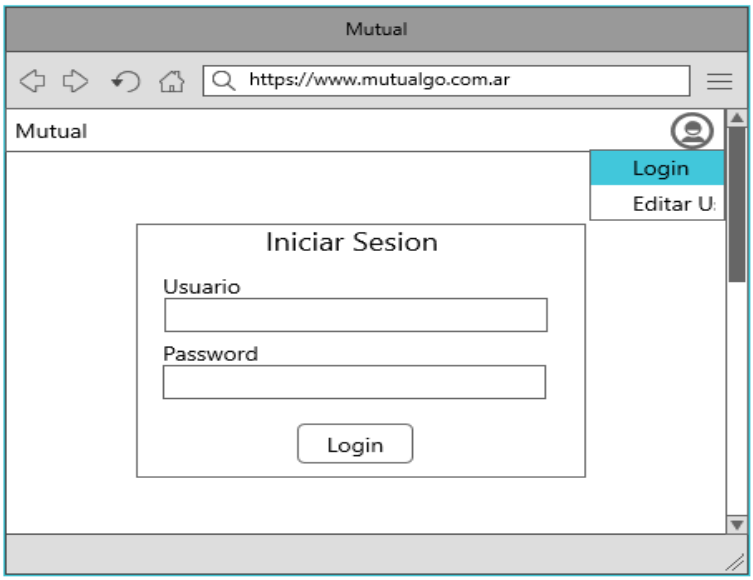



**Figura 33: Maqueta de pantalla de inicio de sesión.**

Luego se presenta la interfaz principal de la aplicación con las opciones disponibles de acuerdo con el rol que posea el usuario que inicio sesión.


**Figura 34: Maqueta de pantalla principal.**

### Caso de uso 2: Alta de socio

El objetivo de esta interfaz es permitir a un usuario de con rol Comercial o Administrador, registrar en el sistema a una persona como nuevo socio, para que la misma pueda comenzar a recibir los servicios como socio de la mutual. Para llevar a adelante dicho proceso, el usuario deberá completar el formulario primero eligiendo el tipo de persona. Luego si es una persona física los datos personales a ingresar son nombre, apellido, sexo y fecha de nacimiento, CUIT/CUIL, calle, número, localidad, números de contactos y observaciones. En cambio, si es una persona jurídica, los datos a ingresar son Razón Social, CUIT/CUIL, calle, número, localidad, números de contactos y observaciones. Luego de haber completado los datos, el sistema validará si los mismos se encuentran completos y correctos, y se habilitará la opción Confirmar para enviar la información al servidor. Si los datos poseen errores, o se encuentran incompletos el sistema indicará cuales son los campos que poseen errores. Luego de enviar los datos al servidor, el sistema volverá a analizar los datos y buscará que no exista una persona con el mismo CUIT. En caso de que los datos estén correctos, se registrará a la persona como socio en el sistema. En caso contrario se indicarán cuáles campos poseen errores.

Pantalla de alta de persona física

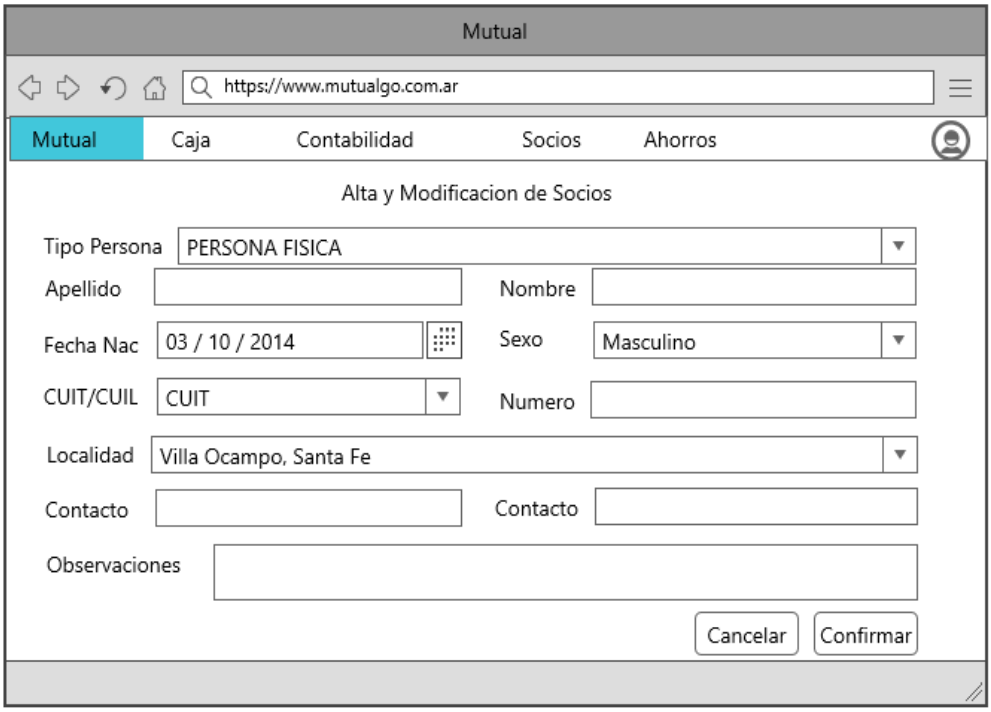



**Figura 35: Maqueta de pantalla alta de persona física.**

Pantalla de alta de persona jurídica

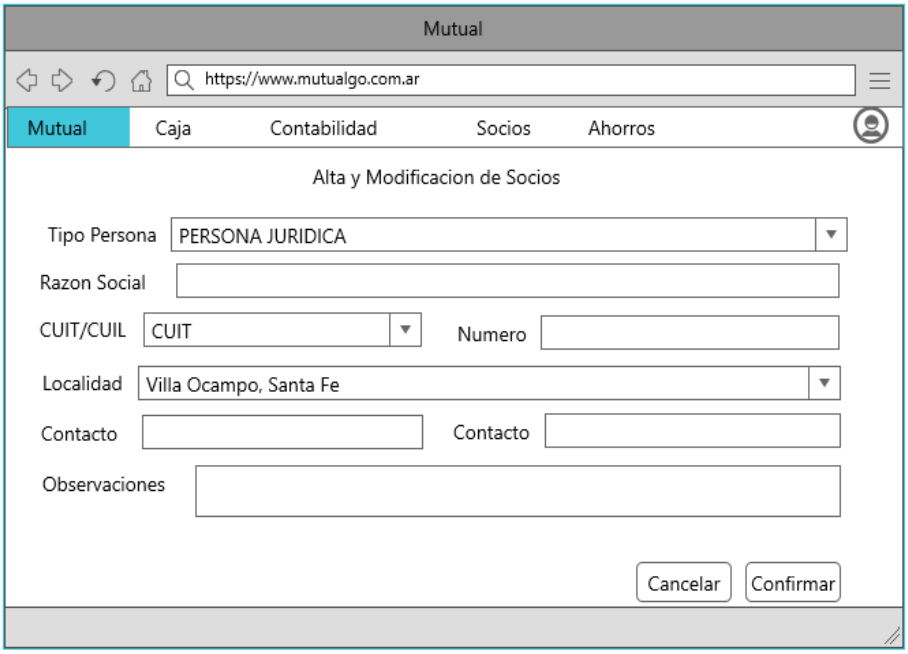



**Figura 36: Maqueta de pantalla alta de persona jurídica.**

**Caso de uso 3: Buscar Socio**

Para este caso de uso se presentan dos interfaces. El objetivo de estas interfaces es buscar un socio en el sistema, a través del ingreso del apellido o nombre (o razón social) y luego poder ver la información personal de este. El usuario deberá ingresar el apellido y/o nombre (o razón social) y el sistema mostrará una lista de socios, donde cada línea posee el número de socio, apellido, nombre (o razón social) y CUIT del socio. También cada línea posee habilitada una opción “dar de baja” en caso que esté activo el socio para acceder al caso Baja de socio, o una opción de “reincorporar” en caso que el socio se encuentre en estado dado de baja para acceder al caso de uso reincorporar socio. Luego el usuario seleccionará de la pantalla un socio, y el sistema mostrará la segunda interfaz. En esta segunda interfaz se mostrará el formulario con los datos personales del socio seleccionado. Todos esos campos se encontrarán deshabilitados. También en dicha interfaz se encuentra un botón Modificar, para acceder al caso de uso Modificar Socio.

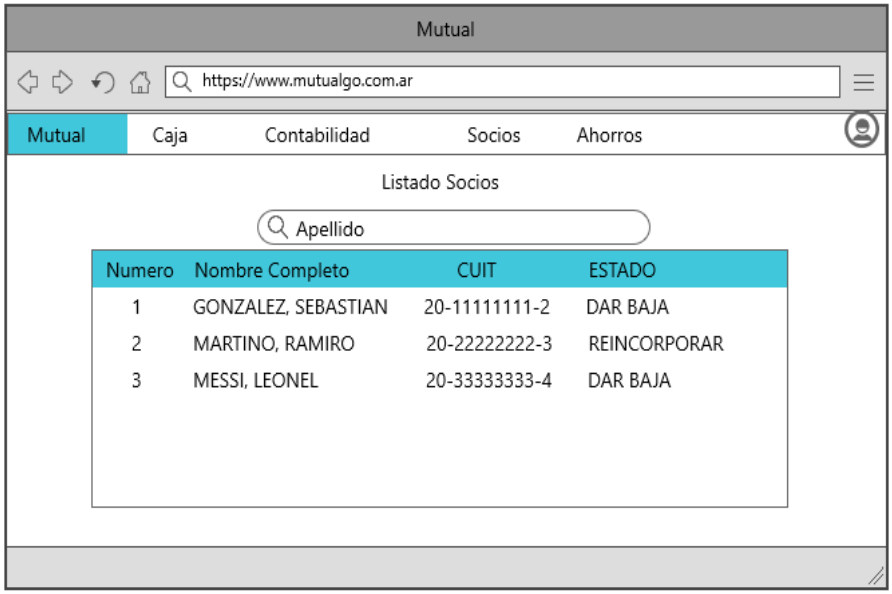



**Figura 37: Maqueta de pantalla de búsqueda de socios.**

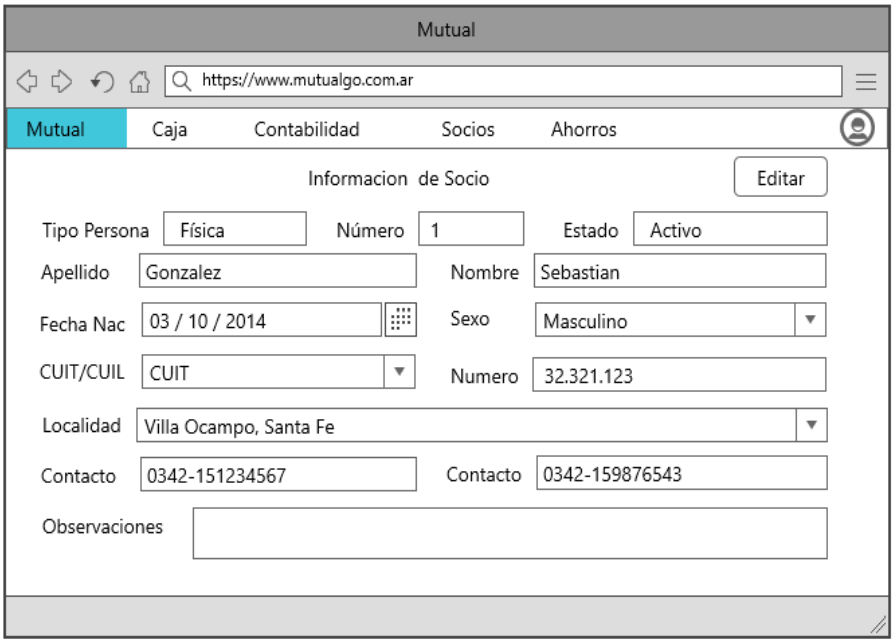



**Figura 38: Maqueta de pantalla de información de socio.**

### Caso de uso 4: Modificar Socio

El objetivo de esta interfaz es permitir a un usuario, modificar información personal, de domicilios y contacto del socio. Para llevar a cabo adelante dicho proceso, en la interfaz de información de socio, el usuario deberá seleccionar la opción editar. De este modo el sistema, habilitará la edición de los campos editables, y las opciones en los campos con múltiples opciones, y también los botones cancelar y confirmar. El usuario deberá modificar en el formulario el campo que desee. Entre los campos a modificar encontramos, el apellido, nombre, fecha nacimiento, sexo, Cuit/Cuil, número, Localidad, Contacto 1, Contracto 2, Observaciones. Luego el usuario deberá presionar Confirmar. El sistema en la parte cliente, chequeará si todos los datos son correctos y están completos antes de enviar la solicitud. En caso contrario, el sistema indicará cuáles campos poseen errores. Luego de enviarse los datos al servidor, el sistema volverá a analizar los datos. En caso de que los datos están correctos, se registrará los cambios en la información del socio. En caso contrario se indicarán cuáles campos poseen errores.


**Figura 39: Maqueta de pantalla de modificar información de socio.**

### Caso de Uso 5: Baja Socio

El objetivo de esta interfaz es permitir a un usuario, dar de baja un socio que se encuentre en estado activo. Para llevar a cabo este proceso, en la interfaz de Listado de Socios, el usuario deberá seleccionar la opción Dar de Baja. De este modo el sistema mostrará una interfaz donde pedirá al usuario que confirme la baja. El usuario presiona el botón confirmar y luego el sistema, dará de baja al socio, manteniendo la información de este, pero quedará inhabilitado a recibir cualquier servicio de la mutual. El sistema muestra un mensaje indicando que el socio fue dado de baja con éxito.

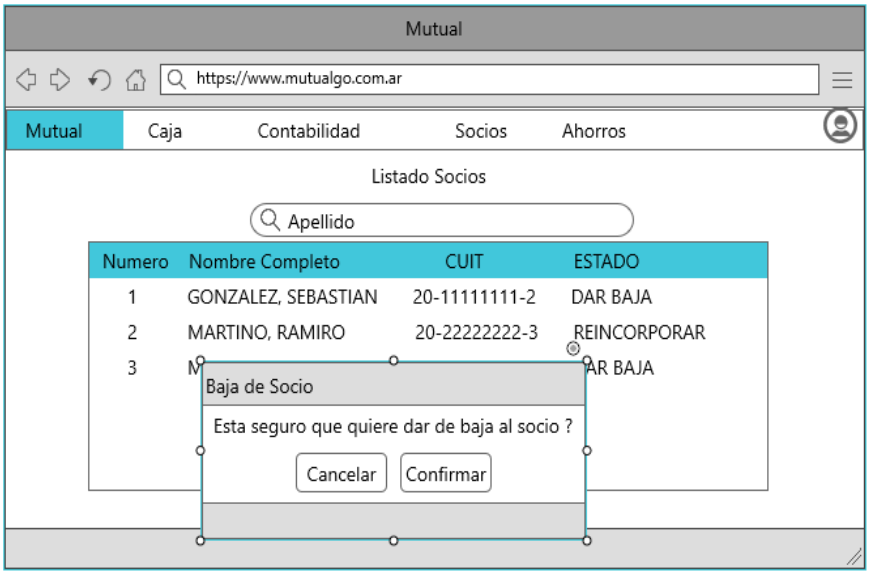


**Figura 40: Maqueta de pantalla de baja de socio.**

### Caso de Uso 6: Reincorporar Socio

El objetivo de esta interfaz es permitir a un usuario reincorporar un socio que se encuentre en estado de baja. Para llevar a cabo este proceso, en la interfaz de Listado de Socios, el usuario deberá seleccionar la opción Reincorporar. De este modo el sistema mostrará una interfaz donde pedirá al usuario que confirme la reincorporación del socio. El usuario presiona el botón confirmar y luego el sistema cambiará el estado del socio a activo nuevamente y volverá a estar habilitado a recibir cualquier servicio de la mutual. El sistema muestra un mensaje indicando que el socio fue reincorporado con éxito.

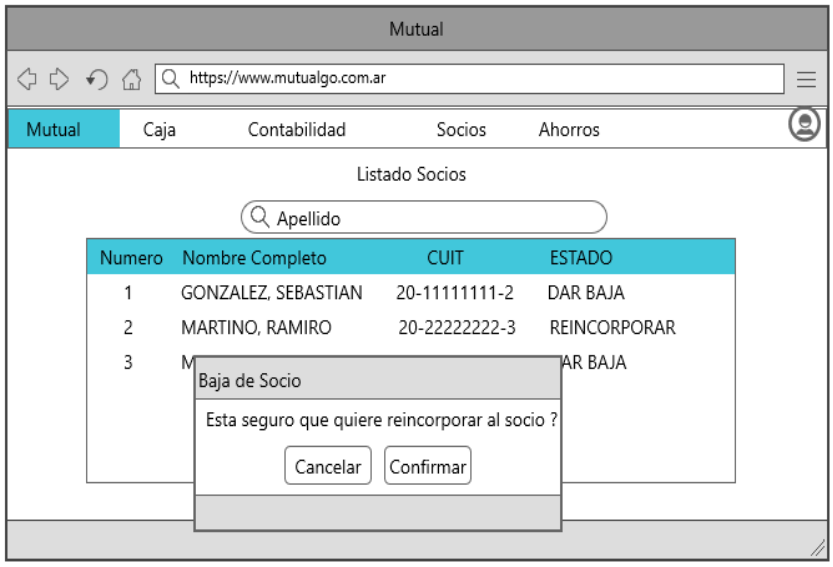



**Figura 41: Maqueta de pantalla de reincorporar socio.**

## CAJAS AHORROS

### Caso de Uso 7: Alta de Caja de Ahorros

El objetivo de esta interfaz es permitir a un usuario, que pueda dar de alta una nueva caja de ahorro a uno o varios socios. Para ello el sistema presenta una interfaz con un campo de búsqueda, donde el usuario debe ingresar el apellido del socio para que el sistema busque los socios que coinciden. El sistema lista los socios, mostrando el apellido, nombre y CUIT de cada uno. El usuario va a elegir a un socio que sea titular de la caja de ahorros y además puede buscar más socios que sean cotitulares. Luego el sistema posee campos para que el usuario cargue cuales van a ser la tasas deudoras y acreedoras y un campo para indicar cual va a ser el número de caja de ahorros. Por último, la interfaz posee dos botones. Un botón

de cancelar, para anular la operación de apertura de caja y otro botón Confirmar para confirmar la apertura de caja. El usuario deberá seleccionar la opción confirmar y el sistema guardará la nueva caja de ahorros en el sistema. Si no se ha seleccionado ningún socio responsable de la cuenta, o si las tasas acreedora o deudora se encuentran vacías o con caracteres inválidas el sistema mostrará un error y se deberá corregir antes de poder confirmar la apertura.

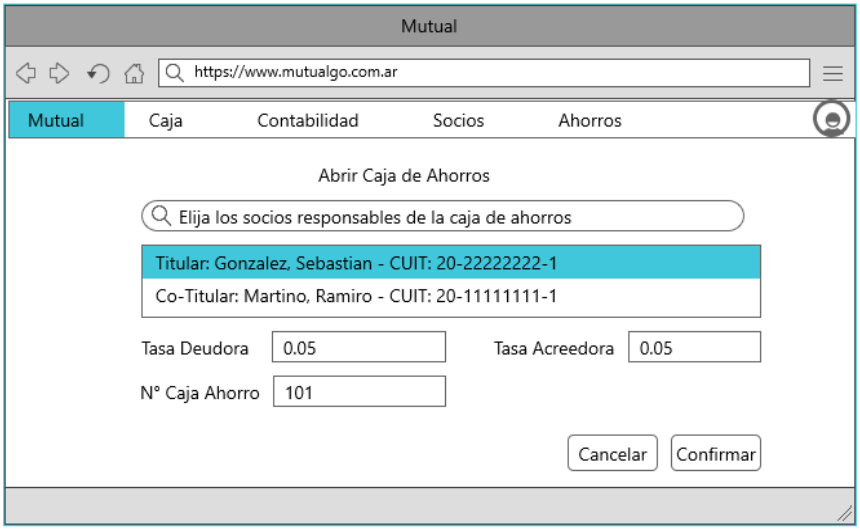



**Figura 42: Maqueta de pantalla de alta de caja de ahorro.**

### Caso de uso 8: Buscar Caja de ahorros

Para este caso de uso se presentan dos interfaces. El objetivo de estas interfaces es permitir a un usuario, buscar una caja de ahorros en el sistema, a través del ingreso del apellido o parte de este y luego poder ver la información almacenada en el sistema de esta. Una vez que el usuario ingresa parte del apellido, el sistema lista todas las cajas de ahorros que tengan como titular dicho apellido. Luego el usuario selecciona de la pantalla la caja de ahorros buscada, y el sistema mostrará la segunda interfaz. Entre los campos de esta segunda interfaz, encontramos el número de caja de ahorros, una lista con el apellido, nombre, cuit y rol de cada asociado en dicha cuenta. También, la interfaz posee un listado con los movimientos realizados y el saldo de la caja de ahorros. También posee varios botones. Un botón para salir de la interfaz donde se muestre información sobre esa caja de ahorros. También en dicha interfaz se encuentran dos botones habilitados según el estado de la caja de ahorros. El primero es el botón editar, para acceder al caso de uso Modificar caja de ahorros. El segundo botón será el botón inhabilitar, si la caja de ahorros está activa, y sirve para para dar de baja la caja a través del caso de uso inhabilitar de caja de ahorros. En

caso de que la caja se encuentre inhabilitada, entonces el botón activo será Habilitar, para volver a habilitar la caja de ahorros, a través del caso de uso Habilitar caja de ahorros. Por último, la interfaz presenta dos botones más, con las opciones de Realizar Depósito / Extracción, a través del caso de uso Realizar Depósito / Extracción.

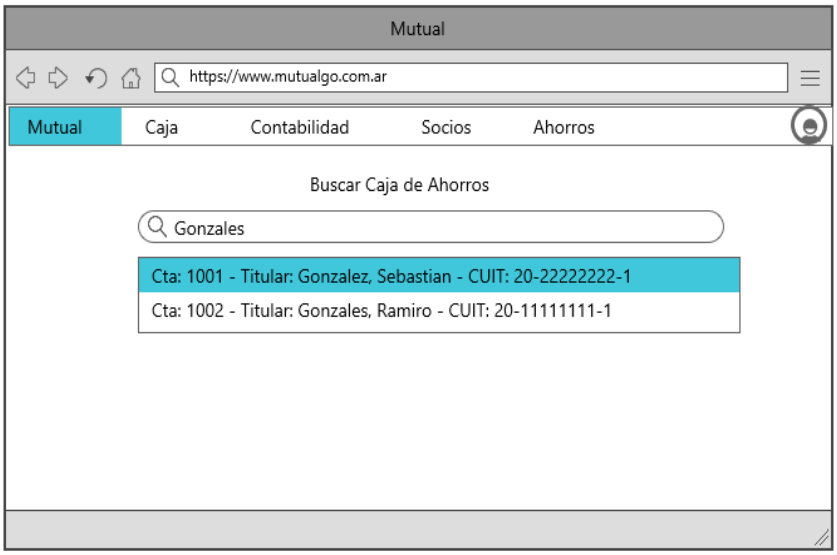



**Figura 43: Maqueta de pantalla de buscar caja de ahorros 1.**

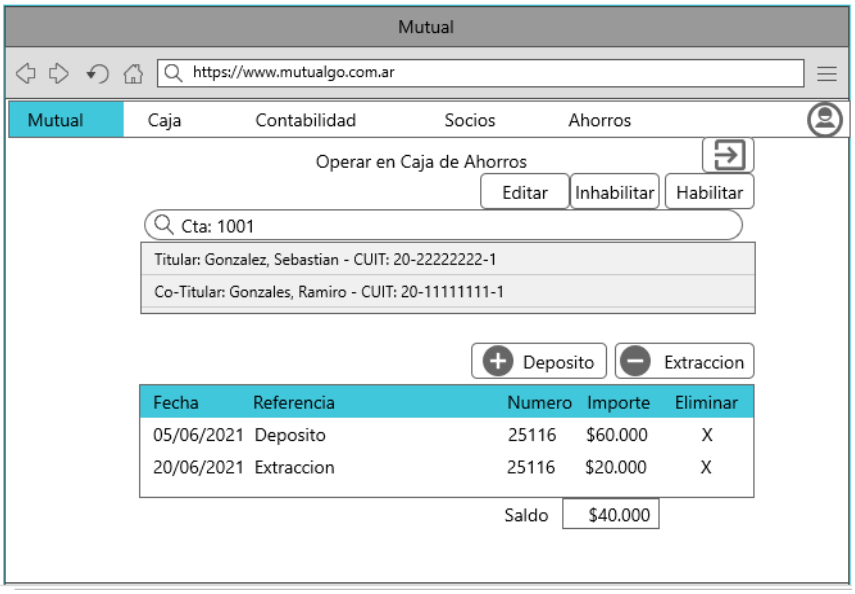



**Figura 44: Maqueta de pantalla de buscar caja de ahorros 2.**

### Caso de Uso 9: Modificar Caja de Ahorros

El objetivo de esta interfaz es permitir a un usuario, que pueda cambiar los responsables de una caja de ahorro, como así también cambiar las tasas acreedoras y deudoras. Para ello el sistema presenta una interfaz con un campo de búsqueda, donde el usuario debe ingresar

el apellido del socio que quiera agregar como nuevo responsable para que el sistema busque los socios que coinciden. El sistema lista los socios, mostrando el apellido, nombre y cuit de cada uno. El usuario va a elegir a un socio y el sistema lo agrega a la lista de responsables de la interfaz. En dicha lista el usuario puede eliminar al o los socios que ya no serán responsables. Luego el sistema posee campos para que el usuario modifique las tasas deudoras y acreedoras y un campo para indicar cual va a ser el número de caja de ahorros. Por último, la interfaz posee dos botones. Un botón de cancelar, para anular la operación de apertura de caja y otro botón Confirmar para confirmar la apertura de caja. El usuario deberá seleccionar la opción confirmar y el sistema guardará la nueva caja de ahorros en el sistema. Si no se ha seleccionado ningún socio responsable de la cuenta, o si las tasas acreedora o deudora se encuentran vacías o con caracteres inválidas el sistema mostrará un error y se deberá corregir antes de poder confirmar la apertura.

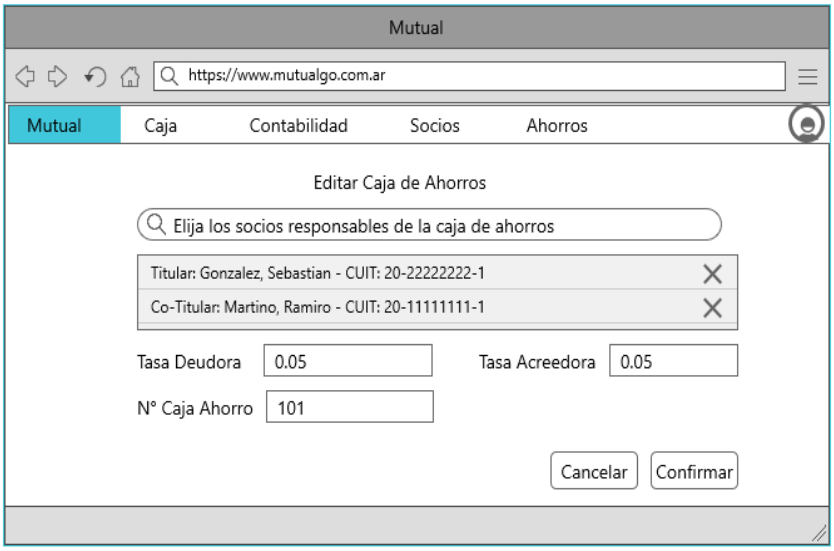



**Figura 45: Maqueta de pantalla de modificar caja de ahorros.**

### Caso de uso 10: Realizar Depósito / Extracción

El objetivo de esta interfaz es permitir a un usuario, que pueda realizar un depósito o extracción de una caja de ahorro de la cual se consultó su información. Para ello el usuario deberá seleccionar la opción depósito / extracción en la interfaz de información de caja de ahorros. Luego el sistema desplegará una interfaz con un campo de importe, donde el usuario ingresará el importe del movimiento. También posee dos botones, uno para cancelar la operación y otro para confirmar. El usuario seleccionará la opción confirmar. Si es una extracción y el importe para retirar es superior al saldo de la caja de ahorros entonces el

sistema emitirá un error indicando dicha situación. También si el importe contiene caracteres que no conforman un número entonces emitirá un error. De otro modo si la validación es correcta entonces el sistema pedirá que el usuario confirme la operación y luego de hacerlo el sistema registrará el movimiento correspondiente.

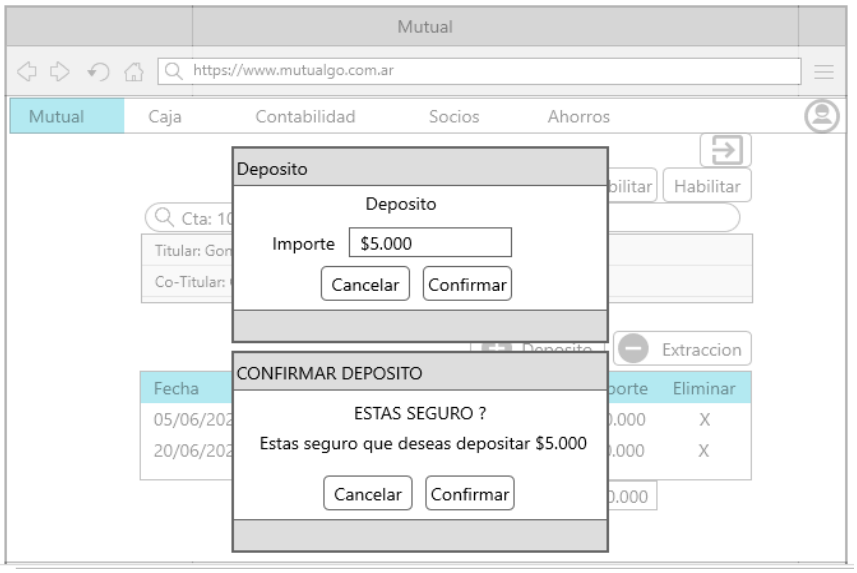



**Figura 46: Maqueta de pantalla de realizar depósito en caja de ahorros.**

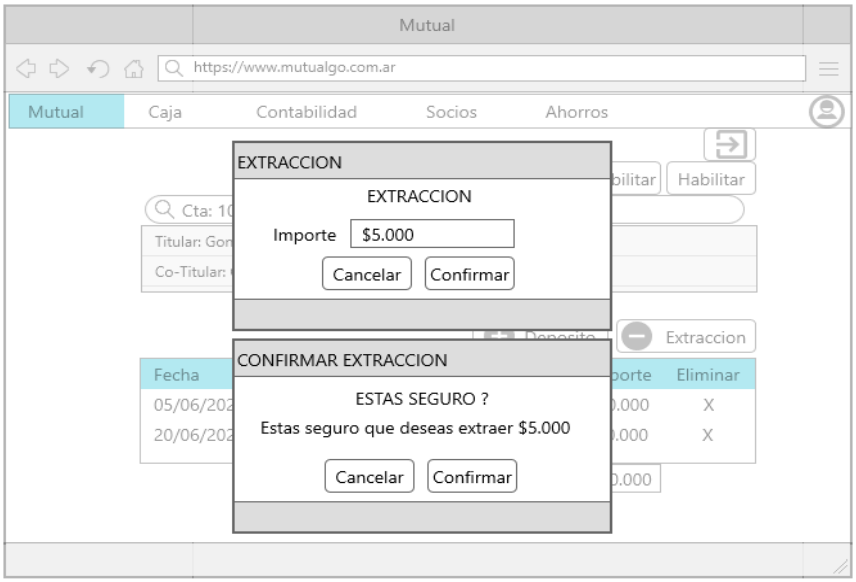



**Figura 47: Maqueta de pantalla de realizar extracción de caja de ahorros.**

### Caso de uso 11: Anular Deposito/Extracción Caja Ahorros

El objetivo de esta interfaz es permitir a un usuario, anular un movimiento de caja de ahorros. El usuario seleccionará la opción de anular en los movimientos de caja de ahorro del socio correspondientes y el sistema anulará el movimiento. Este movimiento

permanecerá almacenado en el sistema, pero en estado de anulado y generará un nuevo movimiento de anulación que compensará el movimiento anulado.

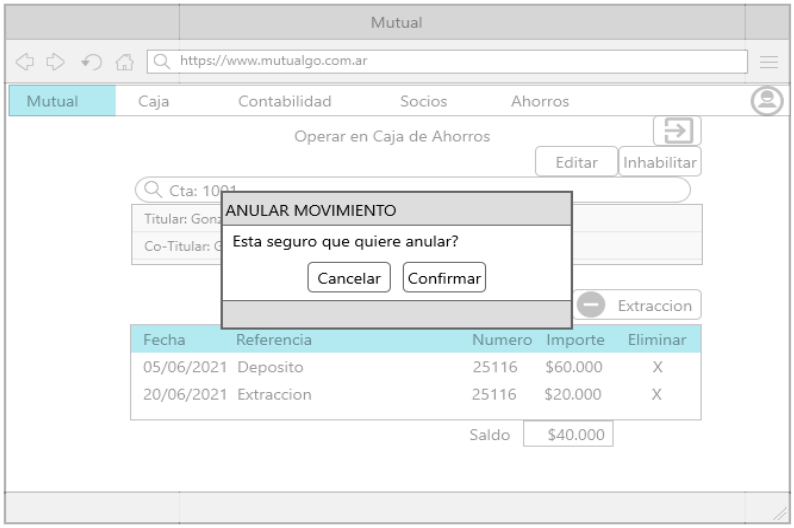



**Figura 48: Maqueta de pantalla de anular depósito/extracción de caja de ahorros.**

### Caso de uso 12: Habilitar Caja de Ahorro

El objetivo de esta interfaz es permitir a un usuario, habilitar una caja de ahorros que esté deshabilitada. Para llevar a cabo este proceso, en la interfaz de información de caja de ahorros, el usuario deberá seleccionar la opción habilitar. De este modo el sistema mostrará una interfaz donde pedirá al usuario que confirme la habilitación. El usuario presiona el botón confirmar y luego el sistema, habilitará nuevamente la caja de ahorros. El sistema muestra un mensaje indicando que la caja de ahorros fue habilitada con éxito.

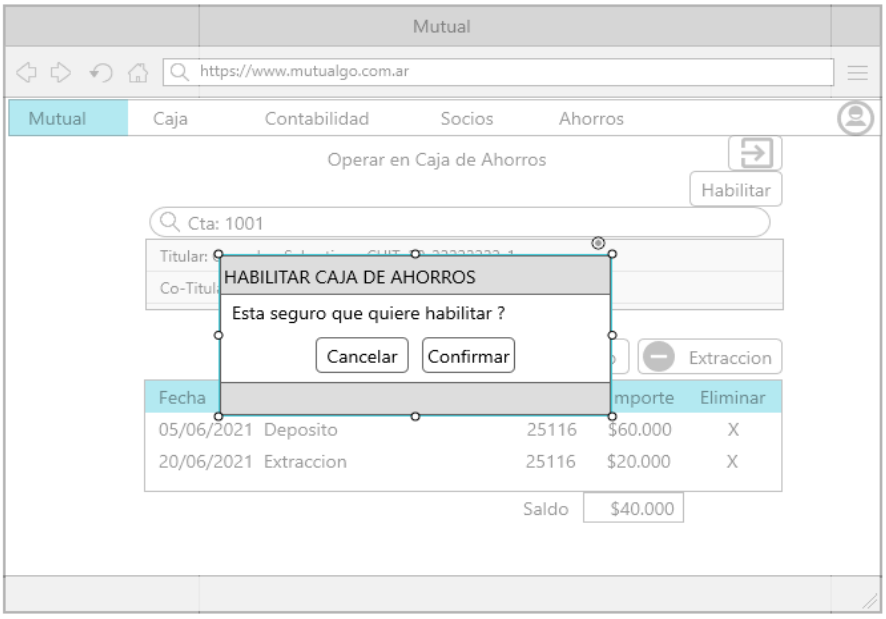


**Figura 49: Maqueta de pantalla de habilitar caja de ahorros.**

### Caso de uso 13: Deshabilitar Caja de Ahorro

El objetivo de esta interfaz es permitir a un usuario, deshabilitar una caja de ahorros que esté habilitada. Para llevar a cabo adelante dicho proceso, en la interfaz de información de caja de ahorros, el usuario deberá seleccionar la opción deshabilitar. De este modo el sistema mostrará una interfaz donde pedirá al usuario que confirme. El usuario presiona el botón confirmar y luego el sistema, deshabilitará la caja de ahorros haciendo que la misma no pueda recibir depósitos ni extracciones. El sistema muestra un mensaje indicando que la caja de ahorros fue deshabilitada con éxito.

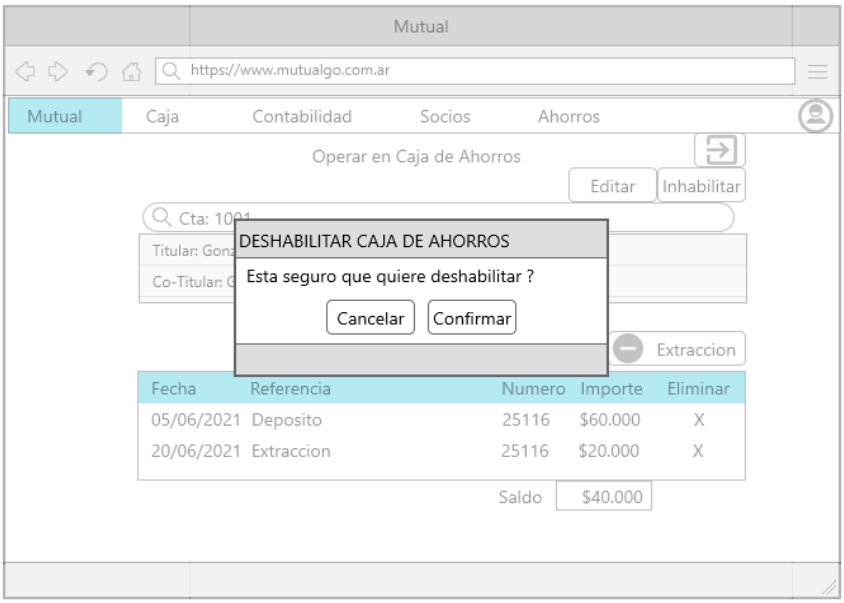



**Figura 50: Maqueta de pantalla de deshabilitar caja de ahorros.**

### Caso de uso 14: Listar Movimientos Cajas de ahorros

El objetivo de esta interfaz es permitir a un usuario, listar los movimientos de todas las cajas de ahorros, filtrando los mismos a través de un período de fechas elegido por el usuario. El usuario, por ende, ingresará las fechas que desea filtrar, luego presionará la opción buscar y el sistema mostrará el listado de los movimientos. Por cada movimiento el sistema mostrará la fecha, una referencia, el número y el importe de este.

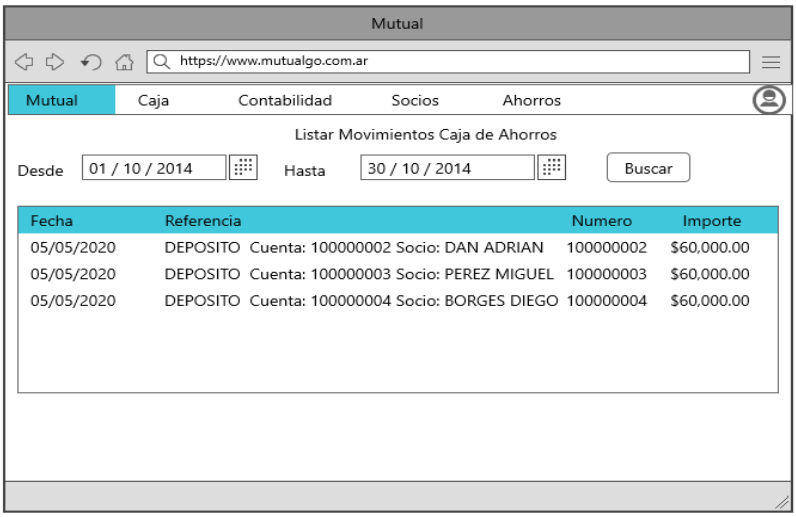



**Figura 51: Maqueta de pantalla de listar comprobantes caja de ahorros.**

## CAJA

### Caso de uso 15: Mostrar Resumen de Caja del día

El objetivo de esta interfaz es permitir a un usuario, que pueda ver el resumen de los comprobantes registrados en la caja del día, así como el saldo con el que se inició la caja y el saldo final de la caja. También tiene las opciones de anular un comprobante y generar un comprobante nuevo.

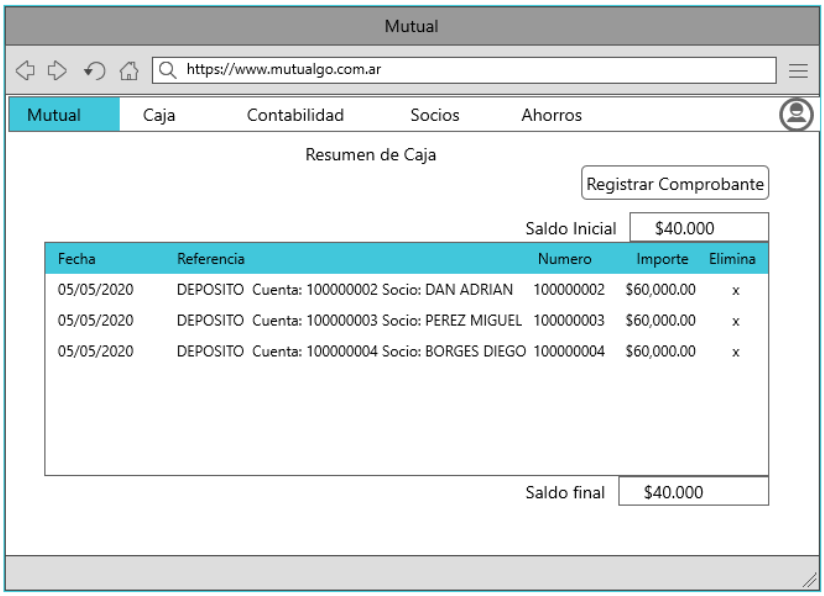



**Figura 52: Maqueta de pantalla de mostrar resumen de caja del día.**

### Caso de uso 16: Registrar Comprobante Manual

El objetivo de esta interfaz es permitir a un usuario, registrar un comprobante manual. Para ello el usuario deberá ingresar un título al comprobante, y luego debe ir agregando los renglones al comprobante, donde cada renglón va a representar un movimiento que impactará en una cuenta contable. Cuando un usuario selecciona Egreso/Ingreso, el sistema muestra una interfaz con un campo para que el usuario ingrese el nombre de la cuenta contable a imputar y otro campo para que ingrese el importe. Una vez que el usuario confirma el ingreso/egreso, el sistema agrega el renglón al comprobante, y permite que el usuario siga agregando renglones al comprobante. Cuando termina de agregar los renglones necesarios, presiona la opción confirmar y el sistema guarda el comprobante y lo agrega a la interfaz de resumen de caja. Si el comprobante no cumple con la regla que la suma de los movimientos de ingresos debe coincidir con la suma de los egresos, entonces el sistema emite un error para que el usuario corrija la información del comprobante.

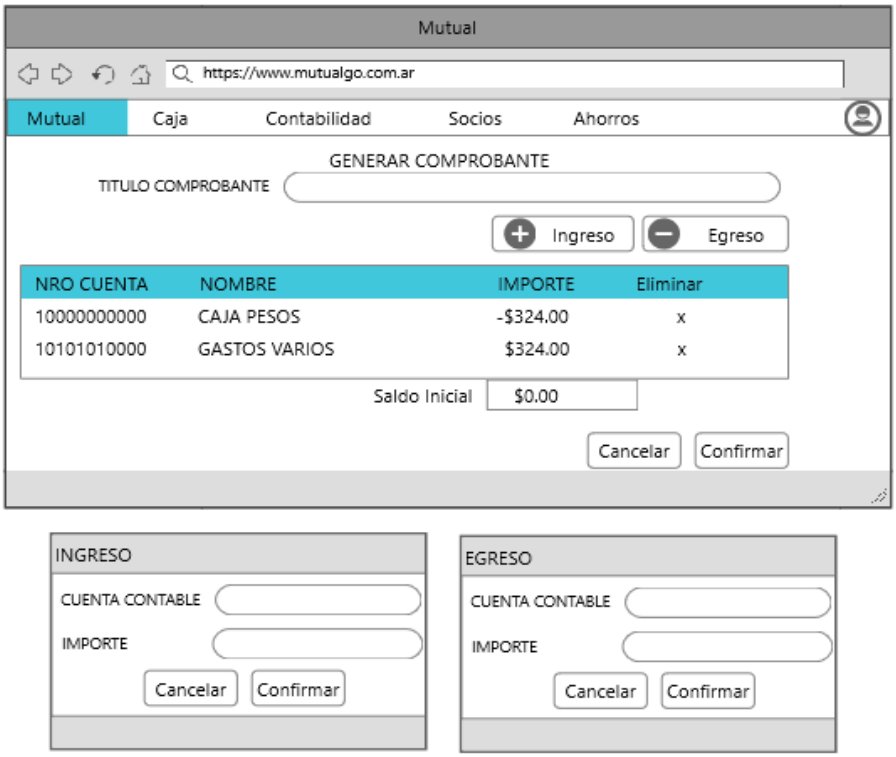



**Figura 53: Maqueta de pantalla de registrar comprobante manual en caja.**

### Caso de uso 17: Anular Comprobante

El objetivo de esta interfaz es permitir a un usuario, anular un comprobante. El usuario deberá seleccionar un comprobante que quiera anular, y el sistema pedirá una confirmación. Luego que el usuario confirma, el sistema registrará ese comprobante como anulado, y generará un nuevo comprobante de anulación que tendrá los movimientos invertidos de tal

manera que se compensen los movimientos del comprobante anulado, y los saldos de las cuentas contables queden igual que antes de cargar los comprobantes.

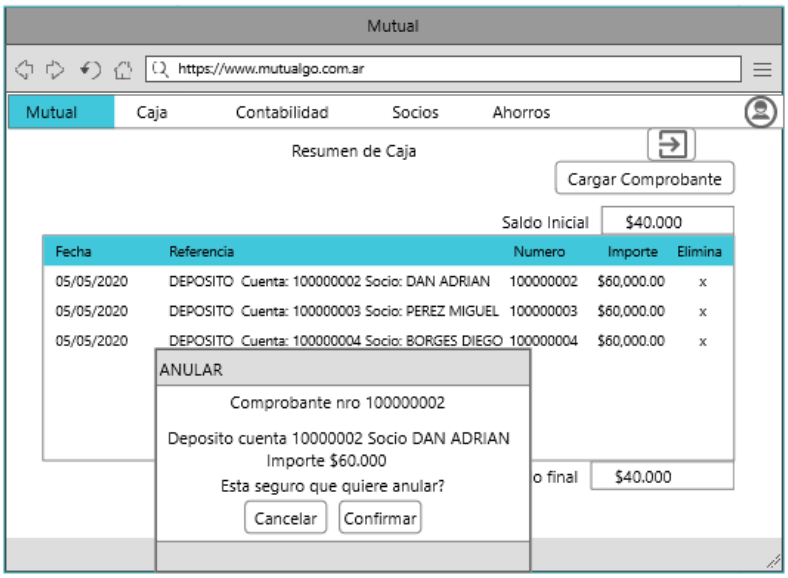



**Figura 54: Maqueta de pantalla de anular comprobante en caja.**

### Caso de uso 18: Listar Comprobantes

El objetivo de esta interfaz es permitir a un usuario, listar comprobantes, filtrando los mismos a través de un período de fechas elegido por el usuario. El usuario, por ende, ingresará las fechas que desea filtrar, luego presionará la opción buscar y el sistema mostrará el listado de los comprobantes. Por cada comprobante el sistema mostrará la fecha, una referencia, el número y el importe de este.

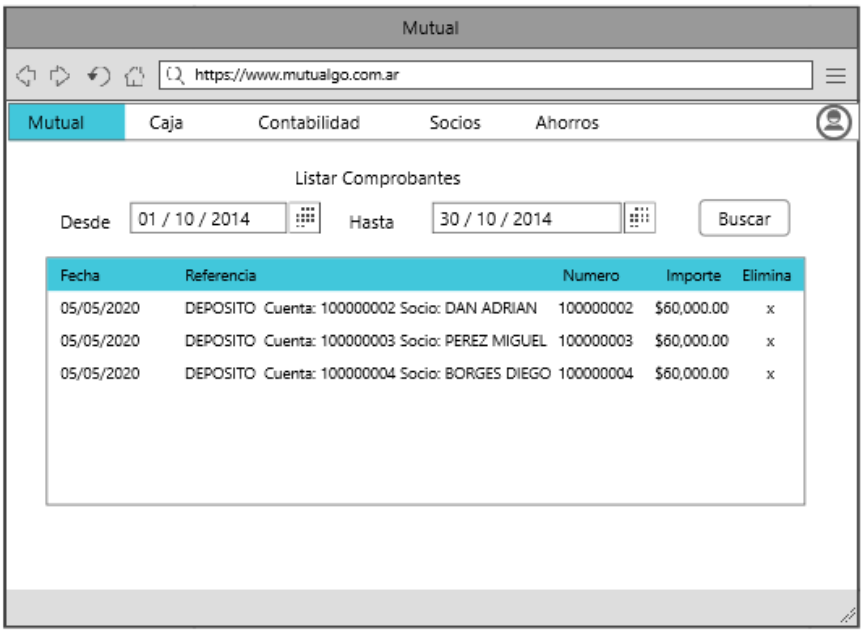


**Figura 55: Maqueta de pantalla de listar comprobantes de caja.**

## CONTABILIDAD

### Caso de uso 19: Listar Cuentas Contables

El objetivo de esta interfaz es permitir a un usuario, poder listar cada una de las cuentas existentes en el sistema. Por cada línea, el sistema mostrará el número de cuenta, el nombre de cuenta. Además, por cada línea se encontrará habilitada la opción de editar cuenta, para modificar información de esta. Por último, si es una cuenta no imputable, entonces en dicho renglón se encontrará habilitado un botón para poder agregar una cuenta imputable más específica en la jerarquía de cuentas a través del caso de uso alta de cuenta contable.


**Figura 56: Maqueta de pantalla de listar cuentas contables.**

### Caso de uso 20: Mostrar Saldos de Cuentas Mensual

El objetivo de esta interfaz es permitir a un usuario, poder listar todas las cuentas contables, junto con los importes totales de movimientos deudores y acreedores y el saldo correspondiente de cada cuenta en el mes seleccionado. Para ello el sistema presentará una interfaz con un campo seleccionable de mes para que el usuario elija el mes que desea consultar y un botón que el usuario deberá presionar para mostrar los movimientos antes mencionados. Además, una vez que el usuario presiona mostrar el sistema desplegará en la interfaz una lista donde cada renglón estará compuesto del número y nombre de cuenta, total de importes que se movieron en el haber y debe en el mes correspondiente y por último el saldo final de la cuenta contable de dicho mes.

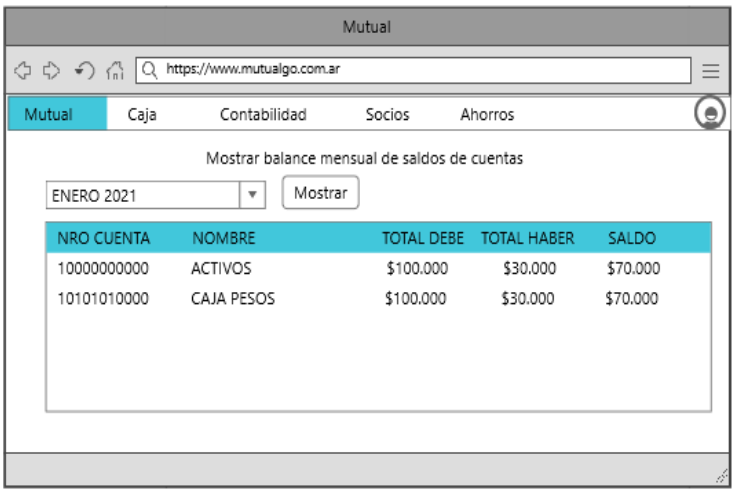



**Figura 57: Maqueta de pantalla de mostrar saldo de cuentas contables.**

### Caso de uso 21: Alta Cuenta Contable

El objetivo de esta interfaz es permitir a un usuario, agregar una cuenta contable nueva. Para ello, dentro del listado de cuentas contables, el usuario buscará la cuenta no imputable de orden superior a la cuenta a crear y seleccionará en dicha cuenta la opción agregar cuenta contable. Luego el sistema desplegará una interfaz donde el usuario deberá completar con el nombre de cuenta, su número y la imputabilidad y enviar la información. El sistema validará la información y en caso de ser correcta registrará la nueva cuenta y notificará al usuario que se registró con éxito la cuenta. Caso contrario el sistema mostrará un mensaje de error.

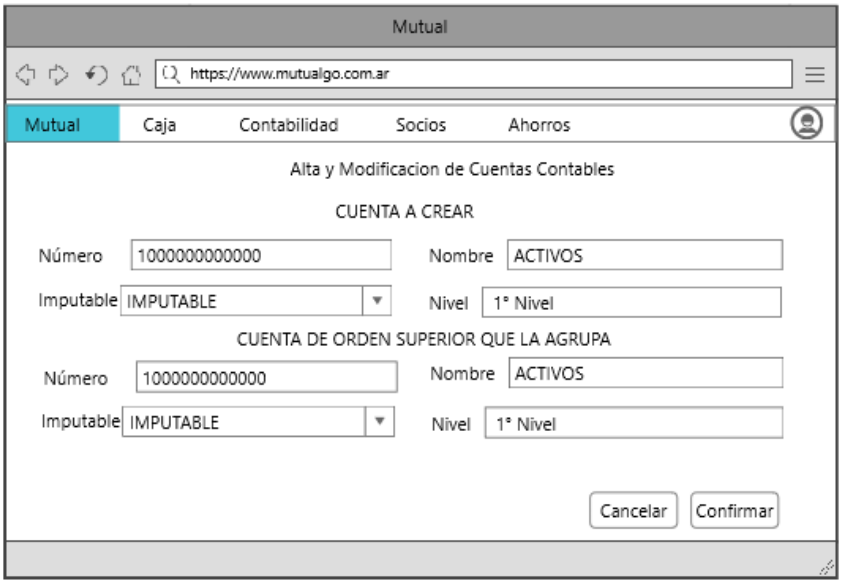



**Figura 58: Maqueta de pantalla de alta de cuenta contable.**

### Caso de uso 22: Modificar Cuenta Contable

El objetivo de esta interfaz es permitir a un usuario, editar la información de una cuenta contable como es el nombre de la cuenta, su número y si es imputable o no. También en la misma interfaz se encuentra disponible la opción eliminar a través del caso de uso eliminar cuenta contable.

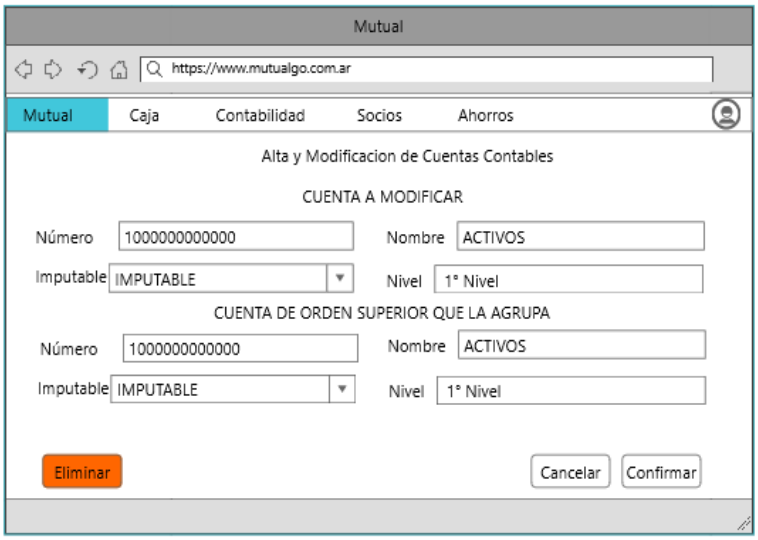



**Figura 59: Maqueta de pantalla de modificar cuenta contable.**

### Caso de uso 23: Eliminar Cuenta Contable

El objetivo de esta interfaz es permitir a un usuario, eliminar una cuenta contable, solo si la misma nunca tuvo movimientos contables asociados. El usuario deberá seleccionar la opción eliminar cuenta y el sistema pedirá que se confirme la operación. Luego el sistema buscará movimientos en la cuenta contable, y si no encuentra ninguno eliminará la cuenta. Caso contrario mostrará un mensaje de error y no permitirá que se la elimine.

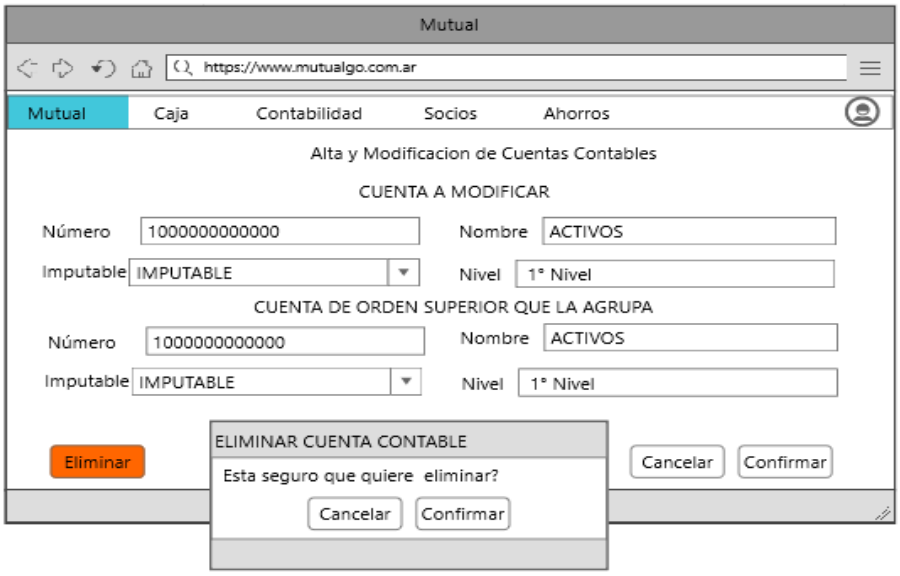



**Figura 60: Maqueta de pantalla de eliminar cuenta contable.**

# Anexo 4: Manual de Usuario

Para acceder al sistema remotamente es necesaria una conexión de internet.

Por defecto se ingresa a la pantalla principal, donde la única opción disponible es la que se encuentra en la esquina superior derecha para identificarse en el sistema con credenciales validas de un usuario registrado. Para ello deberá seleccionar la opción “Login”.

### Pantalla de inicio de sesión

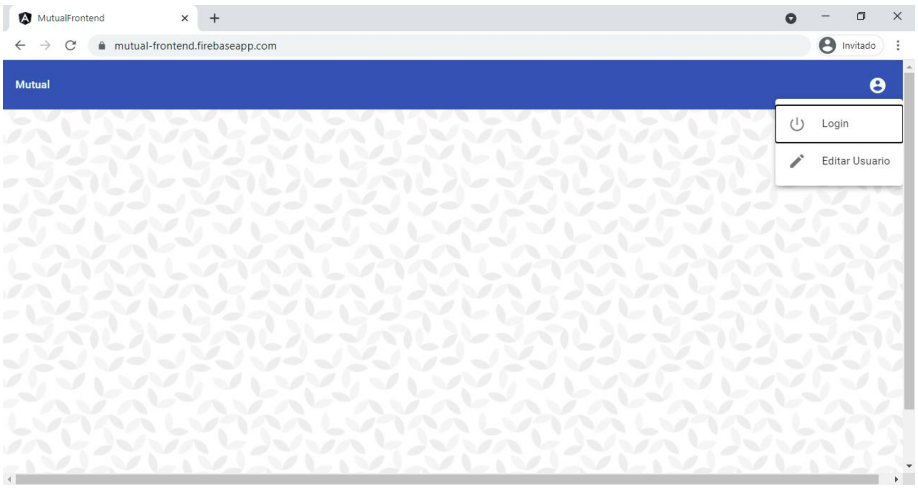



**Figura 61: Pantalla de inicio de sesión 1.**

El sistema le solicitará usuario y contraseña, que el usuario deberá completar y luego enviar para identificarse. Luego el sistema validará las credenciales y si existen en el sistema, se brindará acceso a la pantalla principal con las opciones habilitadas, de acuerdo con el rol del usuario correspondiente.

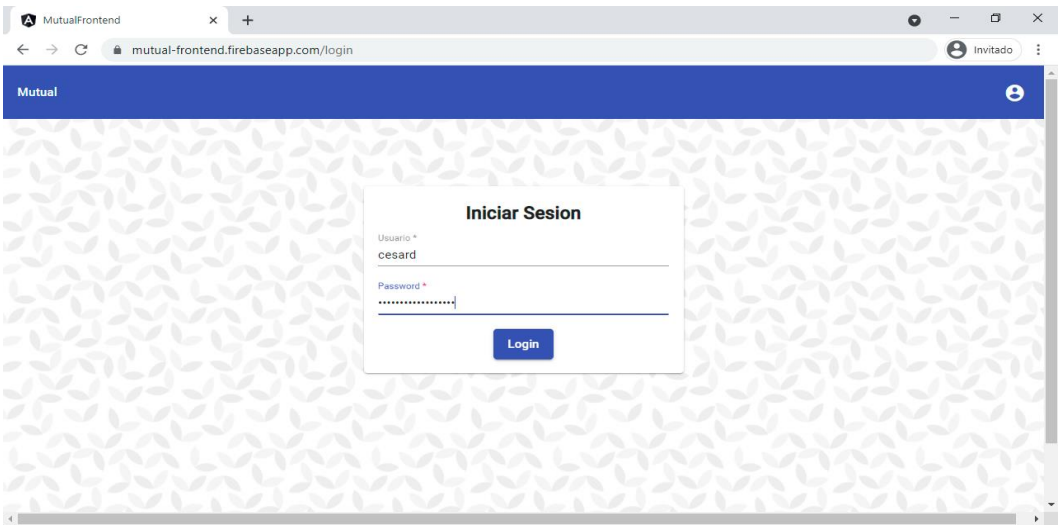



**Figura 62: Pantalla de inicio de sesión 2.**

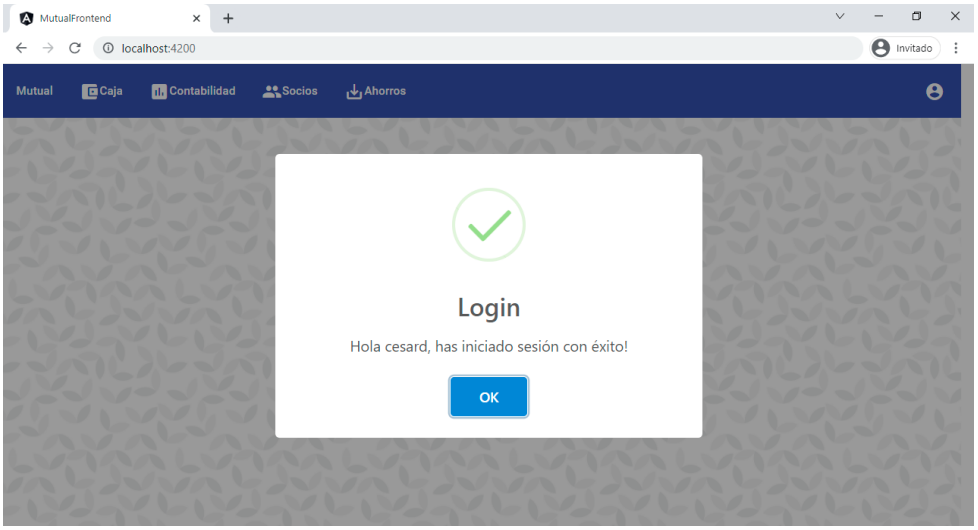



**Figura 63: Pantalla de inicio de sesión 3.**

## Pantalla Principal

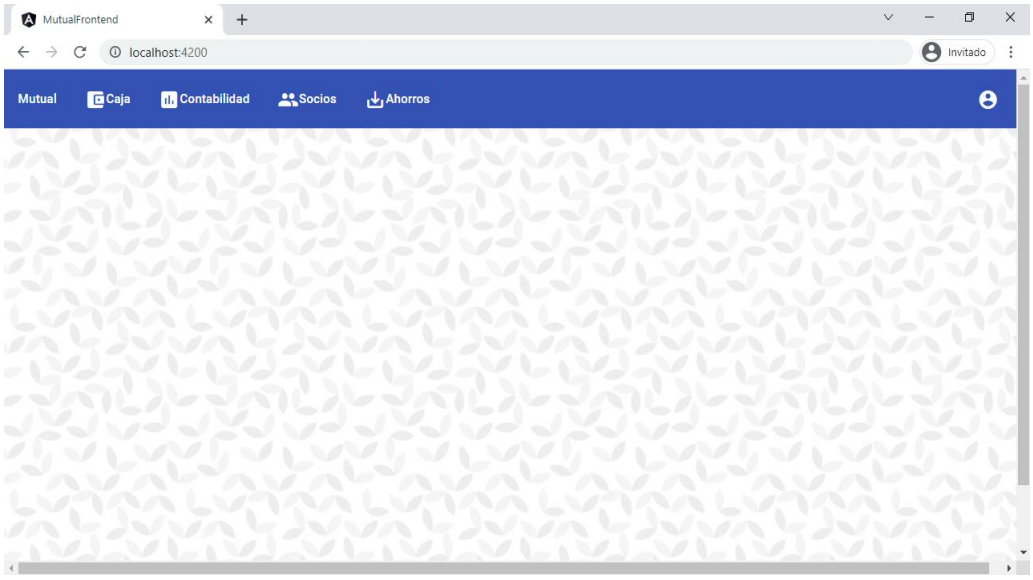



**Figura 64: Pantalla principal.**

En la pantalla principal encontramos en un menú principal de opciones para ingresar a los módulos que posee el sistema.

Hay tres roles de usuarios habilitados actualmente y que detallaremos debajo:

- **Administrador**: Tiene acceso a todas las opciones disponibles del sistema

- **Comercial**: Tiene acceso a las opciones de los módulos de Socios y Cajas de **Ahorros**, pero no tiene acceso a las opciones del módulo de Contabilidad y Caja.
- **Cajero**: Tiene acceso a las opciones del módulo de Caja y Socios, pero no tiene acceso a las de los módulos de Cajas de ahorros y Contabilidad.

## Módulo de Socios

### Alta de Socio

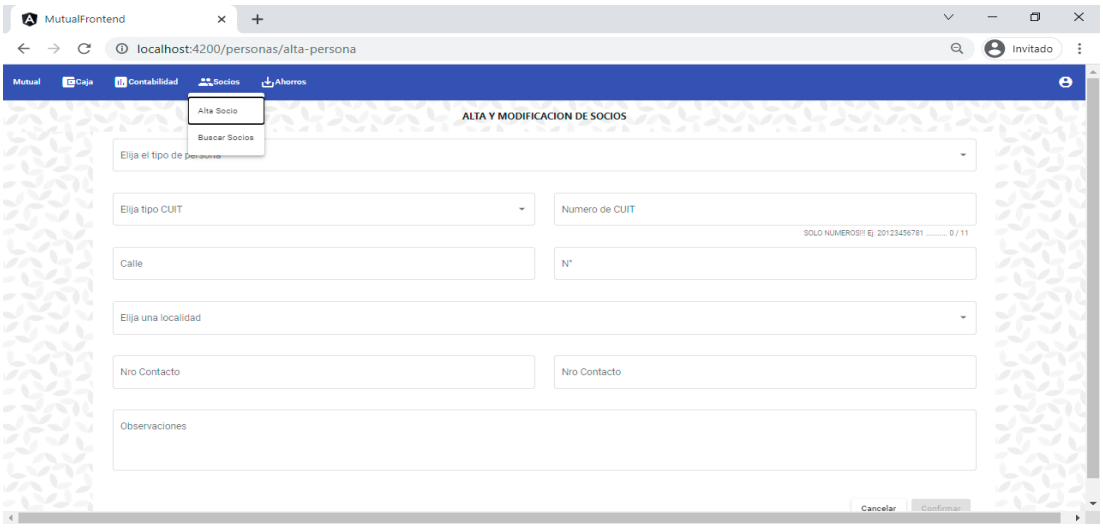



**Figura 65: Pantalla de alta de socio.**

Se debe seleccionar en el menú principal la opción Alta Socio y se desplegará la pantalla anterior.

El usuario deberá completar los campos con la información de la persona que se quiere dar de alta como nuevo socio de la mutual. Existen campos dispuestos en pantalla que son obligatorios y deben estar completos y correctos, para que el sistema habilite el botón de Guardar. Una vez que el usuario completa los campos y presiona guardar, el sistema emitirá un mensaje solicitando confirmar el registro del nuevo socio. El usuario deberá confirmar y el sistema validará la información. Si la misma es válida, el sistema registrará un nuevo socio y emitirá un mensaje para informar al usuario que se registró con éxito. De lo contrario se notificará que existen errores, y para ello es necesario que el usuario los corrija.

### Buscar Socios

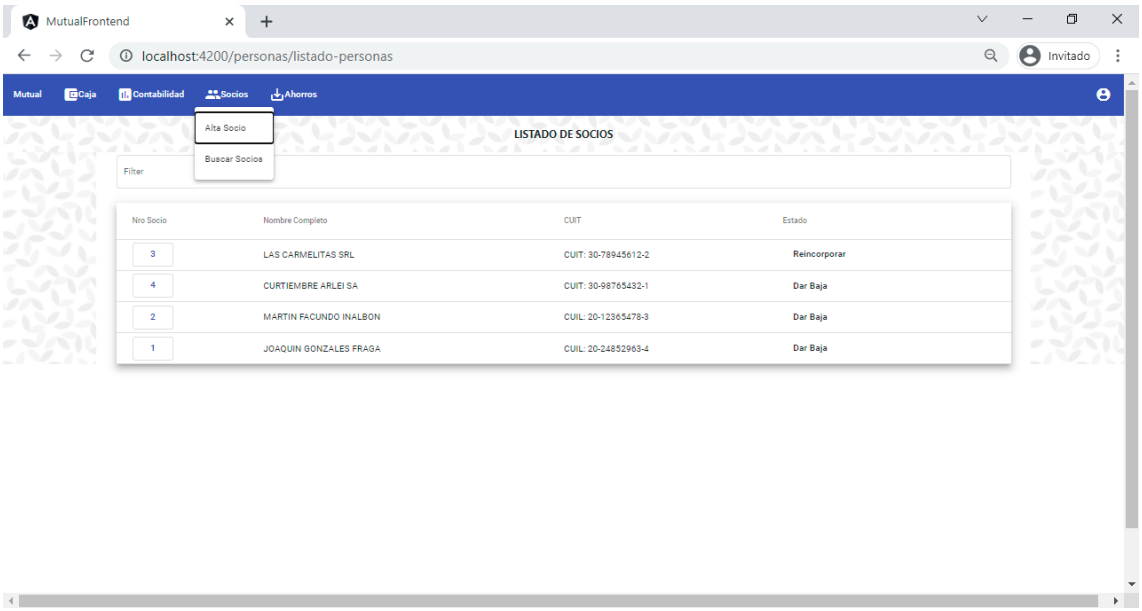



**Figura 66: Pantalla de buscar socios.**

Se debe seleccionar en el menú principal la opción Buscar Socios y se desplegará la pantalla anterior.

El usuario deberá completar el campo del filtro, con el nombre y/o apellido del socio que quiere buscar en el sistema. El sistema desplegará una lista de los socios existentes que coinciden con el filtro. Cada línea posee un botón Ver para que se pueda visualizar la información personal del socio. Una vez que el usuario lo hace el sistema mostrará la siguiente ventana.

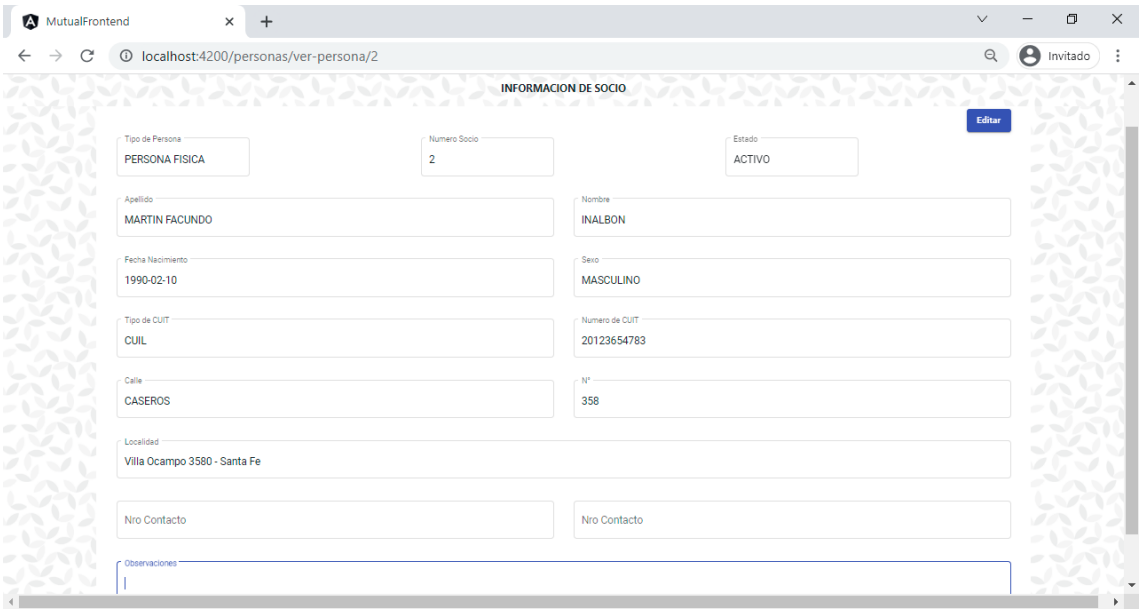



**Figura 67: Pantalla de información de socio.**

En esta ventana se muestra la información registrada del socio. También se encuentra en la esquina superior derecha un botón Editar.

**Editar Información de Socio:**

Si el usuario presiona el botón editar se abre la misma ventana de Alta de Socio, pero con la información registrada del mismo socio. De este modo el usuario puede modificar los campos que considere necesario, y cuando culmine con las modificaciones debe presionar la opción guardar (que solo estará disponible si los campos obligatorios están completos y son válidos). Una vez que se guardan los cambios, el sistema volverá a la interfaz donde se muestra la información actualizada del socio, pero con los campos deshabilitados para que no se puedan modificar.

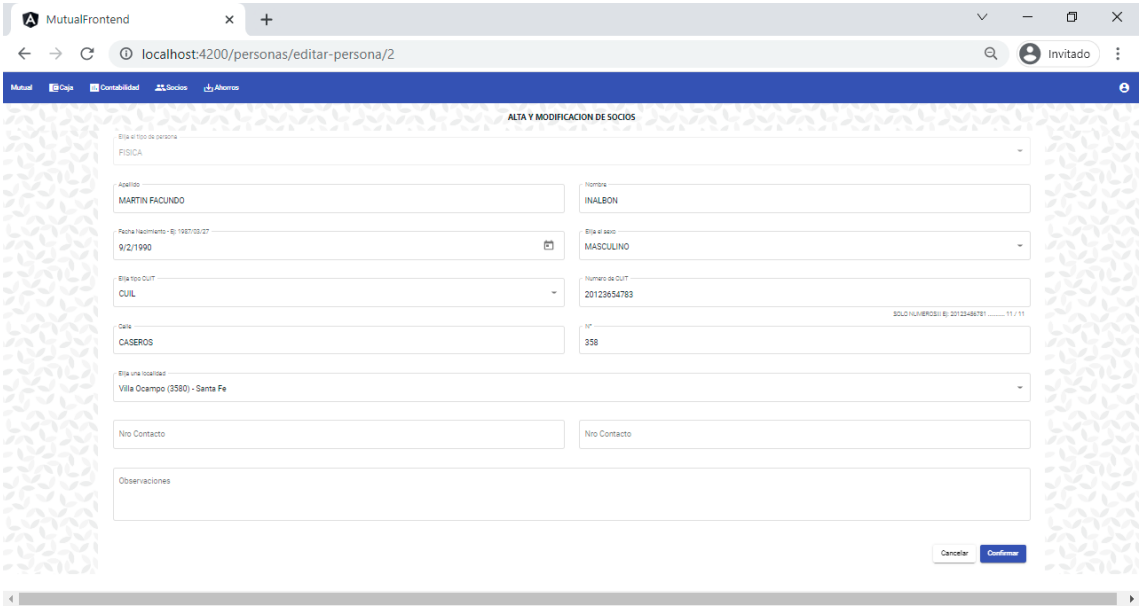



**Figura 68: Pantalla de editar información de socio.**

## Módulo de Caja

**Resumen de Caja del Dia**

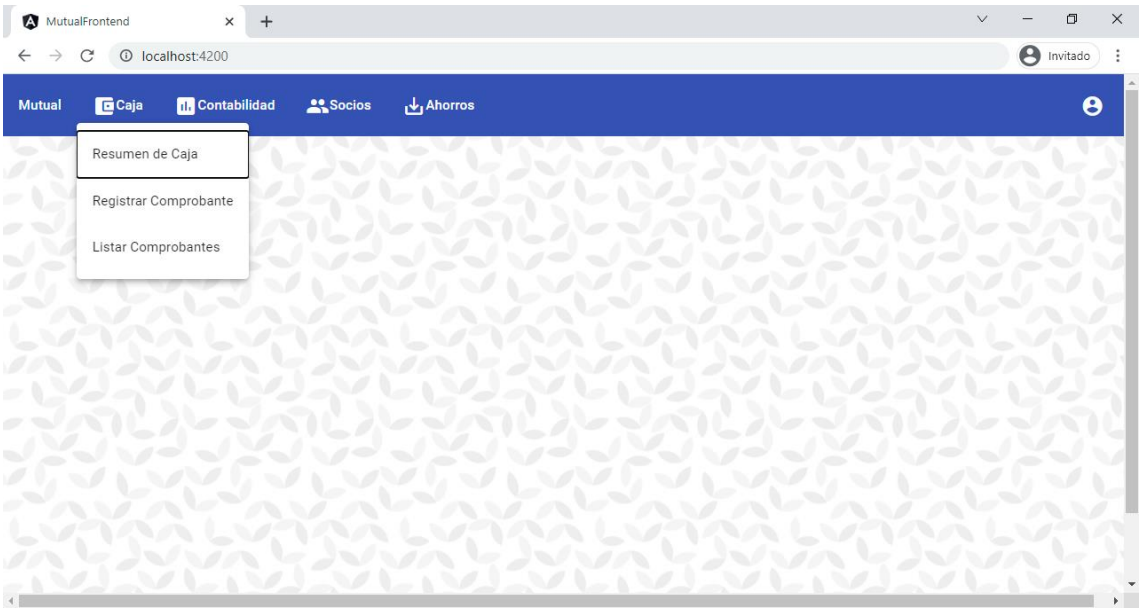



**Figura 69: Pantalla de caja 1.**

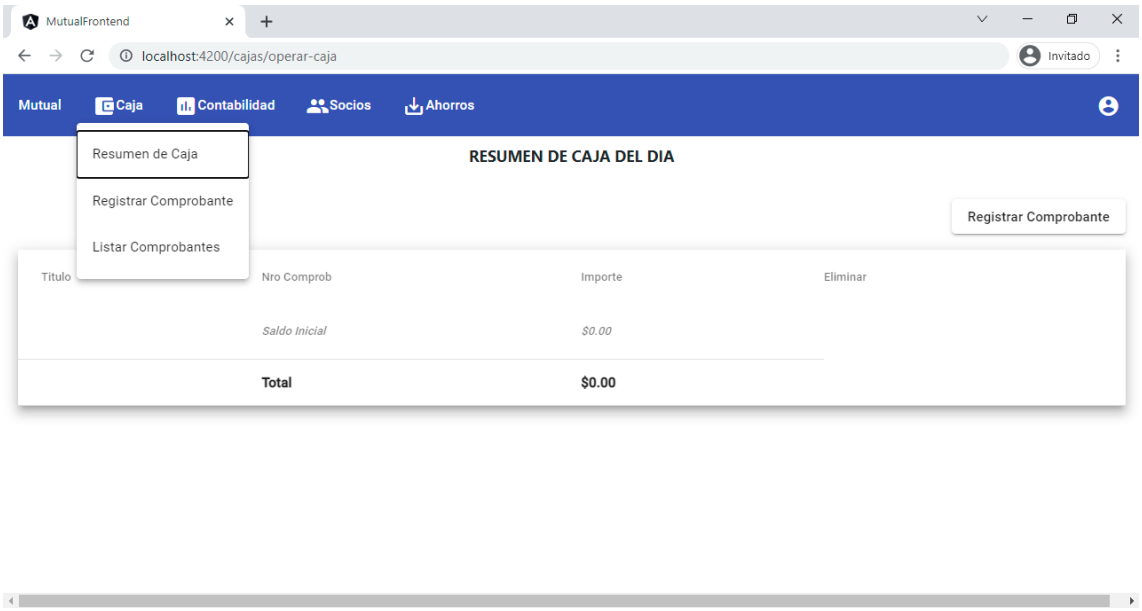



**Figura 70: Pantalla de caja 2.**

Se debe seleccionar en el menú principal la opción Resumen de Caja del Día y se desplegará la pantalla anterior.

En la parte superior derecha, encontramos un botón para poder ingresar un nuevo comprobante, como puede ser el caso de un gasto, una venta, un interés pagado, etc.

En la pantalla también encontramos un campo con el saldo del inicio del día y con el saldo de fin del día, además de una tabla donde se listan todos los comprobantes generados en el día.

### Registrar Comprobante

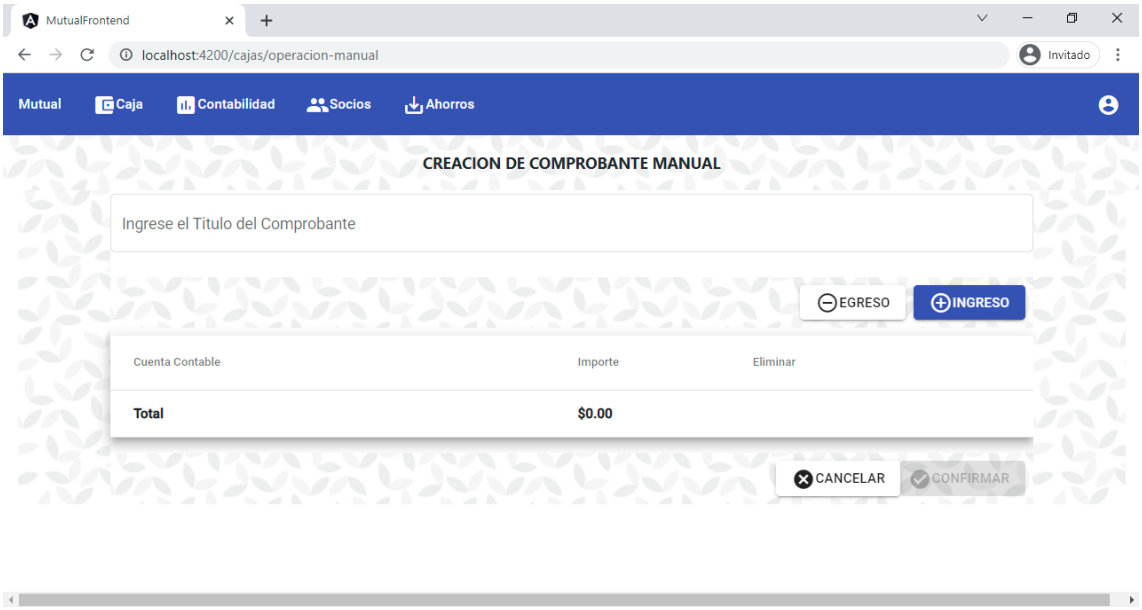



**Figura 71: Pantalla de registrar comprobante en caja 1.**

Desde el menú principal de la aplicación, o desde la pantalla de Resumen de Caja del Dia, se puede acceder al Registro de un nuevo comprobante. Esta interfaz sirve para poder ingresar comprobantes de manera manual, es decir, que no provengan del módulo de Cajas de Ahorros, como pueden ser depósitos o extracciones de dinero.

Se debe ingresar el título que llevará el comprobante. Y luego seleccionar la opción ingreso o egreso dependiendo del movimiento de caja que se quiera realizar.

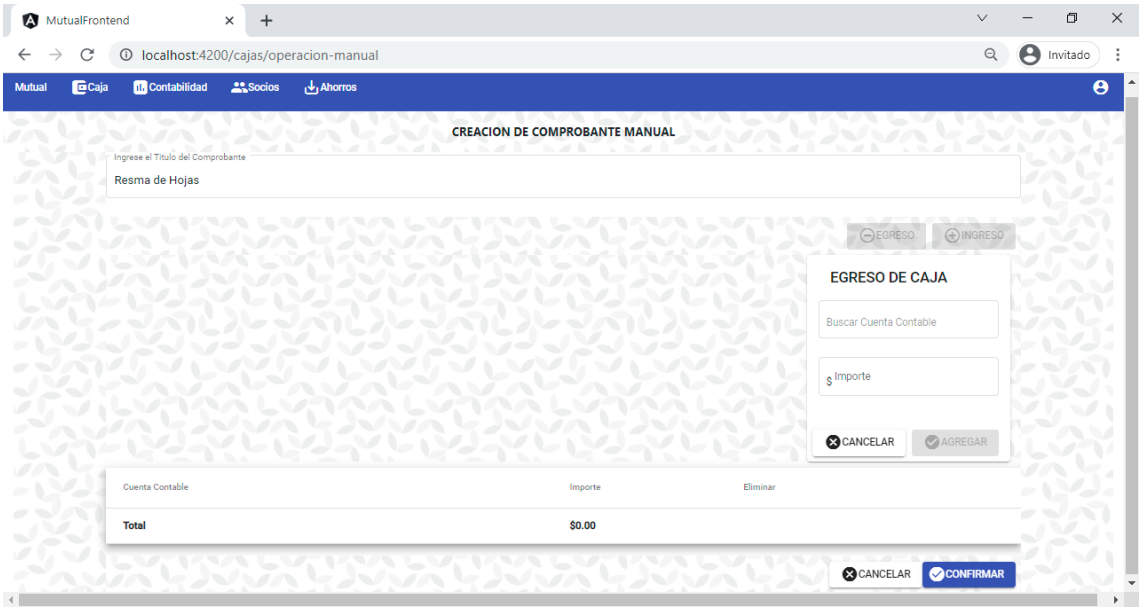



**Figura 72: Pantalla de registrar comprobante en caja 2.**

Una vez presionado el botón de egreso, el sistema desplegará una interfaz donde se debe ingresar el nombre de la cuenta contable y el importe del movimiento que impactará en dicha cuenta. Luego se debe confirmar, y el sistema desplegará el movimiento creado, en la lista de movimientos que tendrá el comprobante.

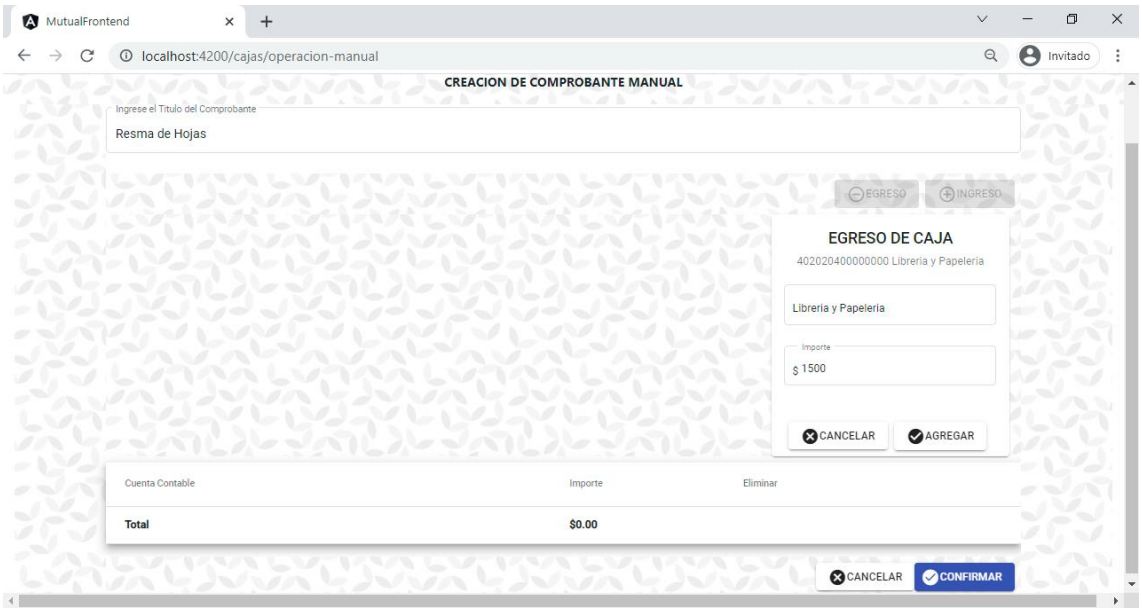



**Figura 73: Pantalla de registrar comprobante en caja 3.**

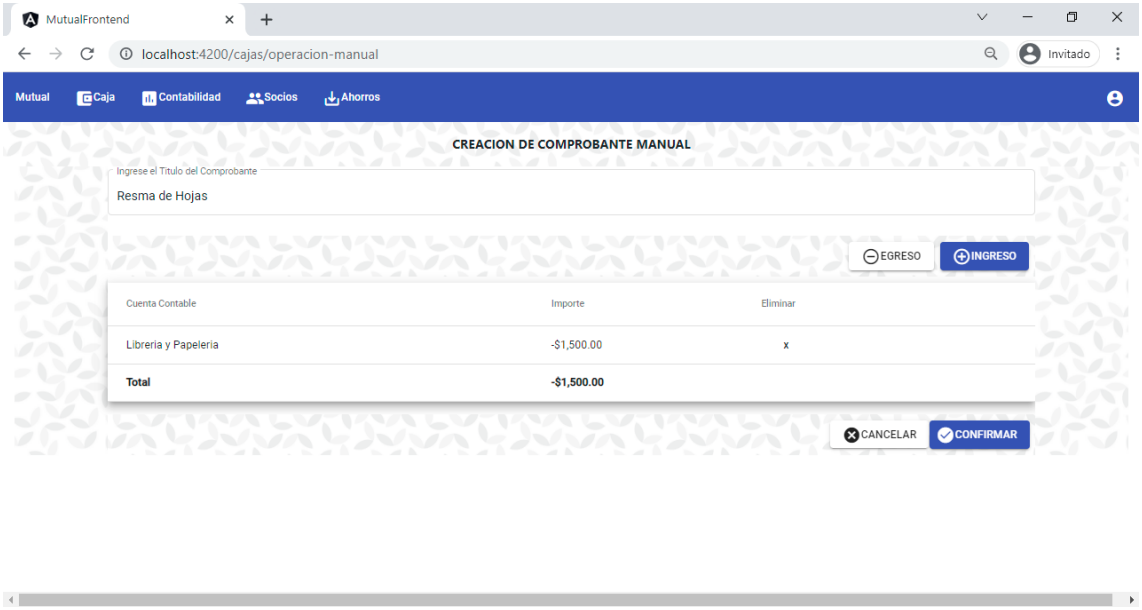



**Figura 74: Pantalla de registrar comprobante en caja 4.**

Se podrán agregar todos los movimientos de cuentas necesarios.

El sistema, habilitará la opción confirmar, una vez que el comprobante sea válido, es decir, que tenga un título y al menos un movimiento.

Por último, se deberá presionar la opción confirmar, para enviar al servidor el nuevo comprobante, y el sistema lo validará y guardará. Luego se volverá a la interfaz de Resumen de Caja del Día donde el sistema, agregará dicho comprobante, a la tabla de los comprobantes del día y se actualizará en pantalla el saldo final de caja.

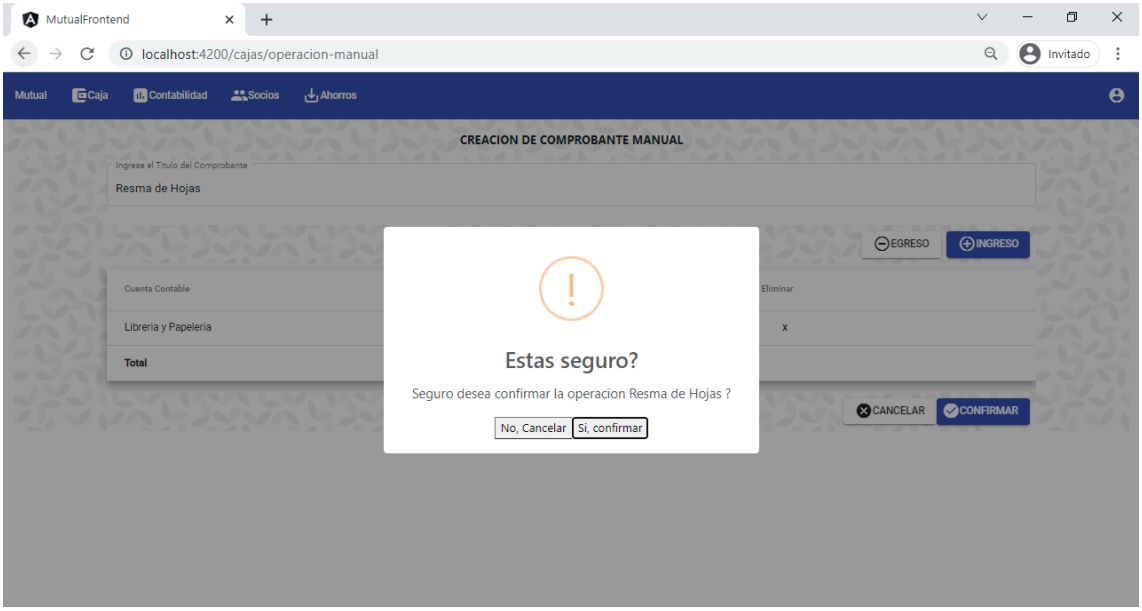



**Figura 75: Pantalla de registrar comprobante en caja 5.**

### Anular Comprobante Cajero

Desde la interfaz de Resumen de Caja del Dia, se puede anular un movimiento manual que haya sido cargado por el socio en el día de la fecha. Para realizar dicha acción, se debe dirigir al listado de comprobantes y encontrar el movimiento que busca. Luego en dicha línea, elegir la opción eliminar (que sólo se encontrará disponible si antes no fue anulado). El sistema pedirá confirmación. Luego el sistema actualizará la información del comprobante para indicar que fue anulado, y agrega un nuevo comprobante, que compense al comprobante anulado.

No se podrán anular comprobantes de Depósitos o Extracciones. Para anular dichos comprobantes, se deberá ir al módulo de Caja de Ahorros.

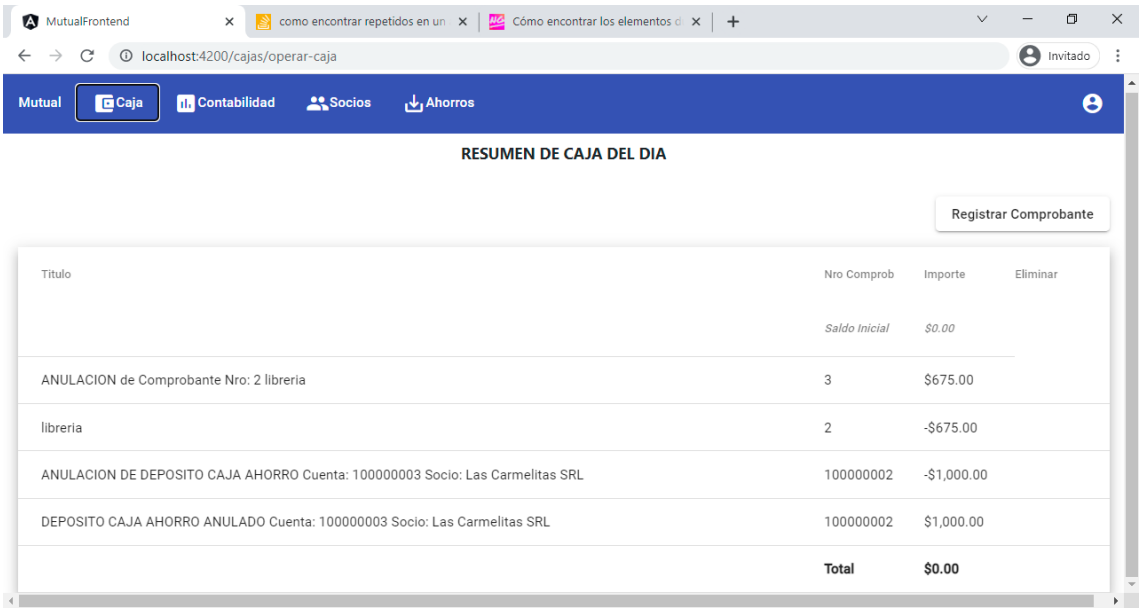



**Figura 76: Pantalla de anular comprobante.**

## Listado de Comprobantes

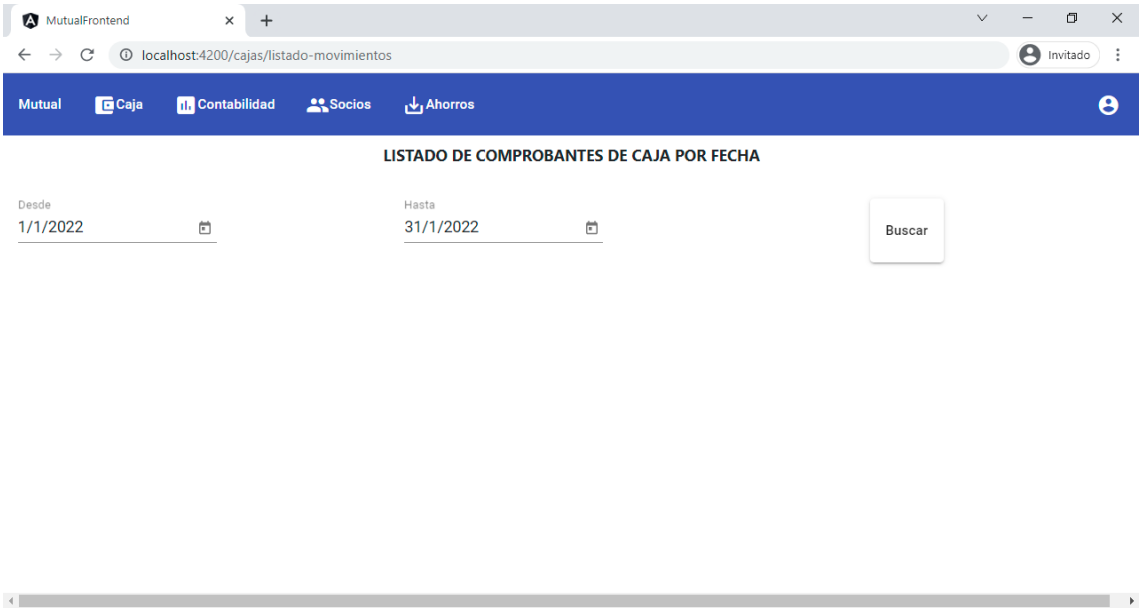



**Figura 77: Pantalla de listar comprobantes de caja 1.**

Desde el menú principal de la aplicación, se puede acceder al Listado de comprobantes.

En la parte superior de esta pantalla, hay dos campos de fechas para filtrar los comprobantes que queremos ver, así como también un botón para confirmar la búsqueda.

Debajo se desplegará una tabla, con un resumen de información de los comprobantes que se generaron dentro de las fechas seleccionadas. En esta tabla, cada línea representa un comprobante, y posee un botón, para anular el dicho comprobante.

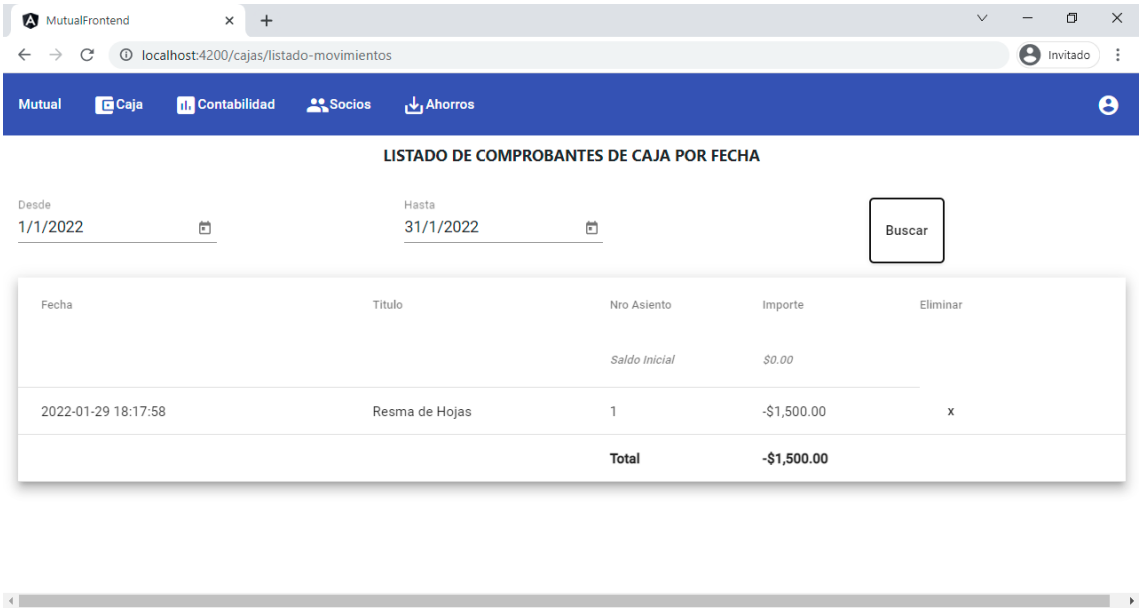



**Figura 78: Pantalla de listar comprobantes de caja 2.**

### Anular Comprobante Administrador

El usuario con rol de Administrador puede anular un comprobante manual que haya sido cargado en cualquier fecha. Para realizar dicha acción, desde la interfaz de Listado de Comprobantes, debe filtrar los comprobantes por fecha, como se explicó en el punto anterior. Luego de encontrar la línea del comprobante deseado, elegir la opción eliminar (que sólo se encontrará disponible si antes no fue anulado). El sistema pedirá confirmación. Luego el sistema actualizará la información del comprobante para indicar que fue anulado, y agrega un nuevo comprobante, que compense al comprobante anulado.

## Módulo de Cajas de Ahorros

### Alta Caja de Ahorros

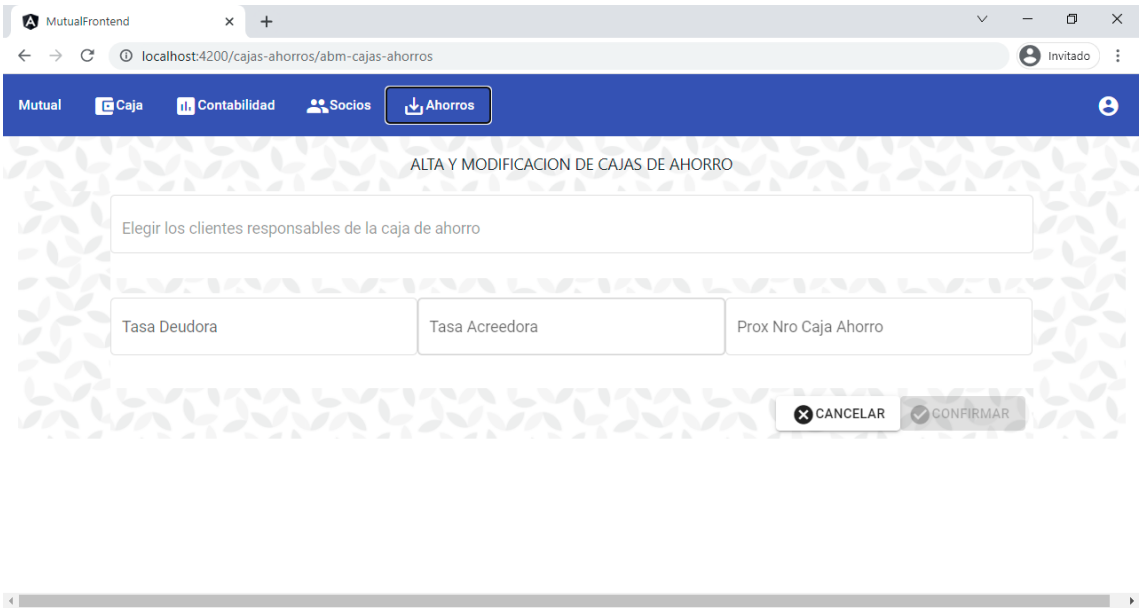



**Figura 79: Pantalla de alta de caja de ahorros 1.**

Se debe seleccionar en el menú principal la opción Alta Caja de Ahorros y se desplegará la pantalla anterior. Luego en el campo de búsqueda, se deberá completar con el apellido, o nombre del socio, y el sistema, listará a los socios que coinciden. El usuario seleccionará un titular y también puede agregar cotitulares a la cuenta.

También hay dos campos editables para configurar las tasas de la caja de ahorros.

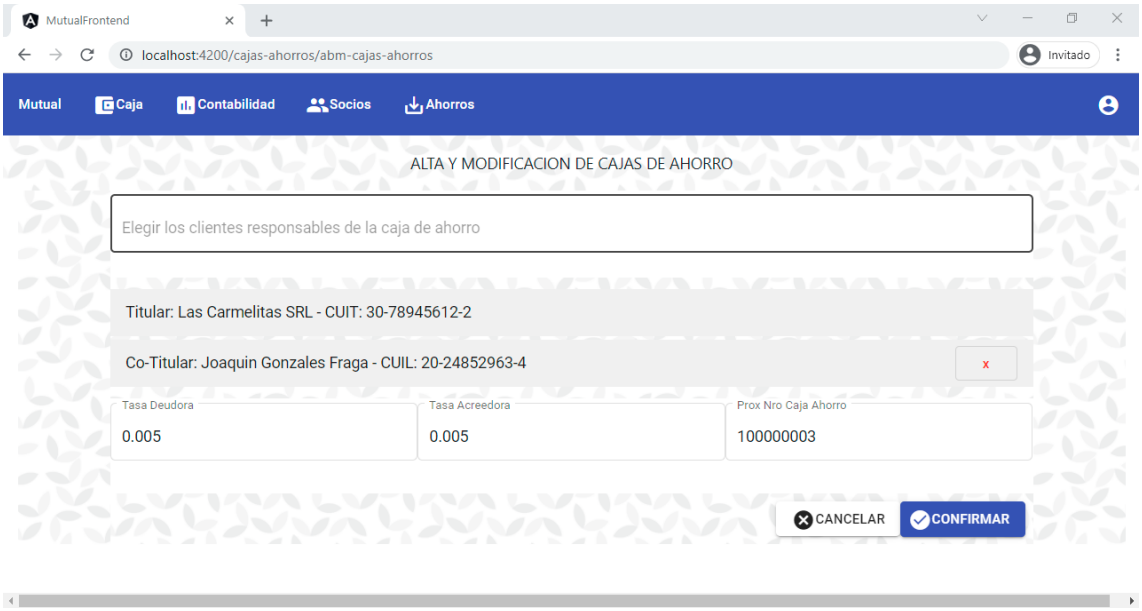



**Figura 80: Pantalla de alta de caja de ahorros 2.**

Una vez que completa la información se deberá confirmar para que el sistema proceda a registrar la nueva caja de ahorros.

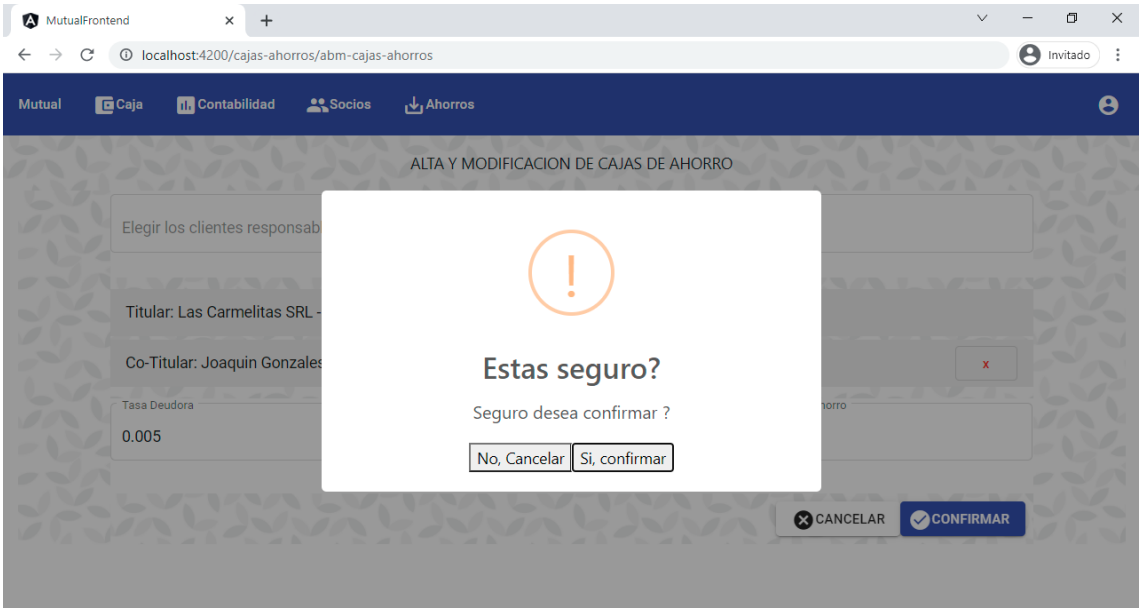



**Figura 81: Pantalla de alta de caja de ahorros 3.**

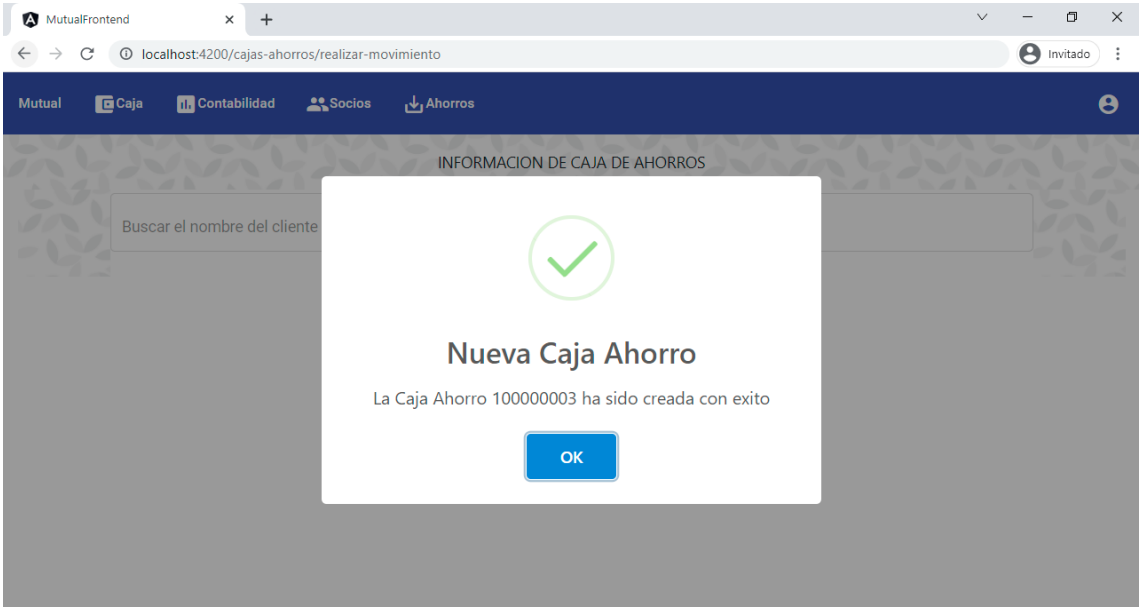



**Figura 82: Pantalla de alta de caja de ahorros 4.**

### Realizar Depósito/Extracción

Desde el menú principal de la aplicación, seleccionar la opción Buscar Caja de Ahorros. En dicha interfaz se debe ingresar el apellido y nombre o razón social del socio responsable

de una caja de ahorros buscado. El sistema desplegará una lista de socios que son responsables de cajas de ahorros y que coinciden con la información de búsqueda suministrada por el usuario.

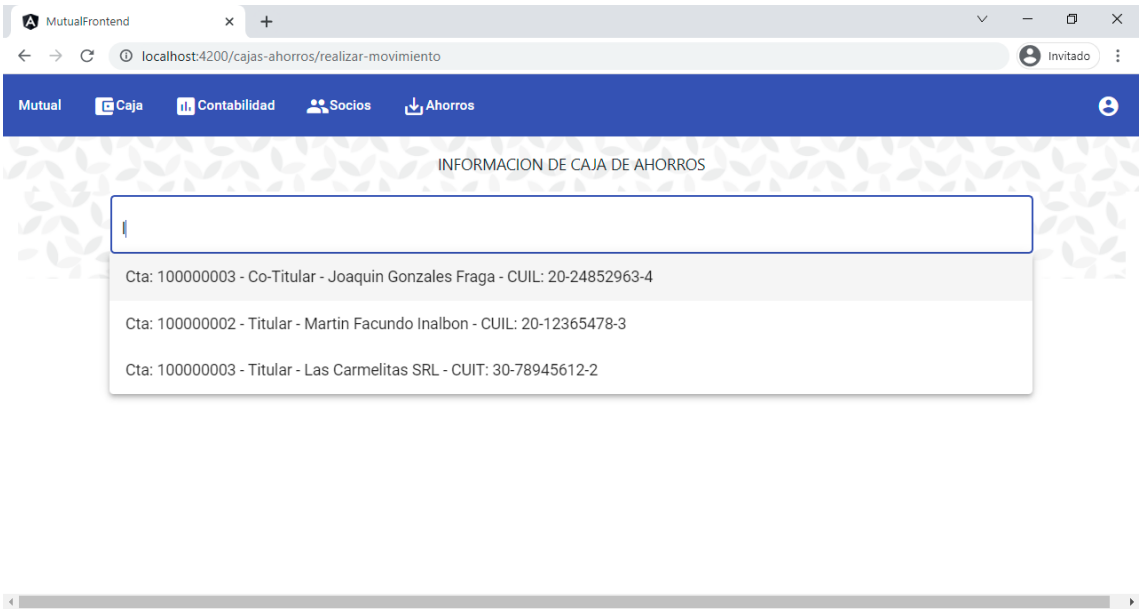



**Figura 83: Pantalla de registro de depósito/extracción 1.**

Una vez que el usuario selecciona un socio de la lista, el sistema desplegará la siguiente interfaz, donde se podrá ver un resumen del saldo de la caja de ahorros y sus movimientos. También se encontrará disponible las opciones de realizar depósito o extracciones, deshabilitar la cuenta y también modificarla.

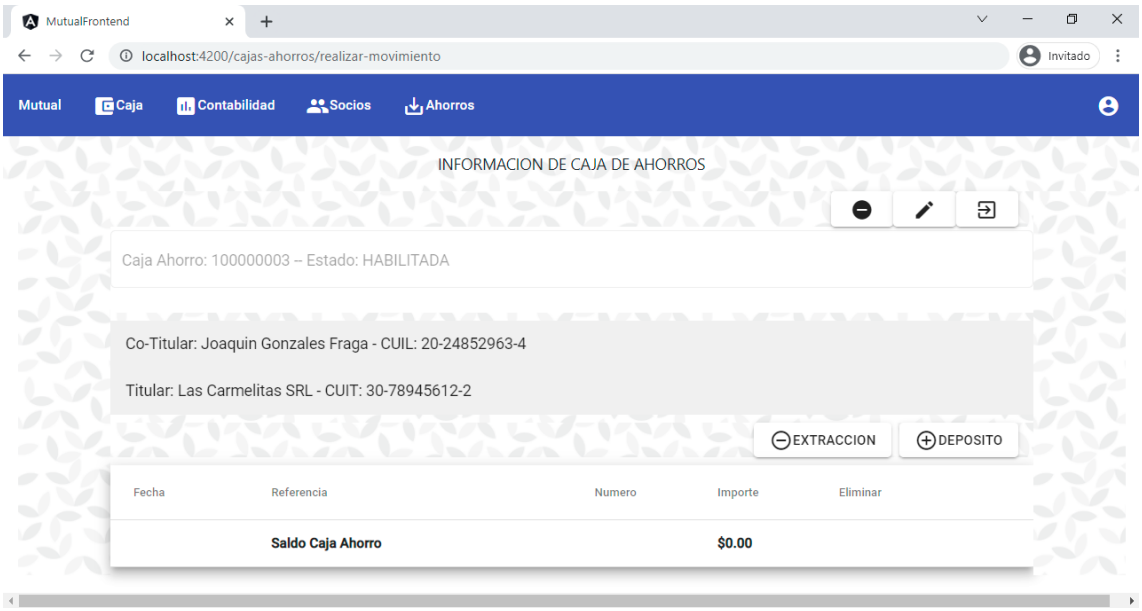



**Figura 84: Pantalla de registro de depósito/extracción 2.**

Ahora veremos cómo realizar un depósito. Para ello, el usuario selecciona la opción "depósito", y el sistema abre un dialogo donde el usuario deberá completar con el importe a depositar en la cuenta, y luego deberá confirmar.

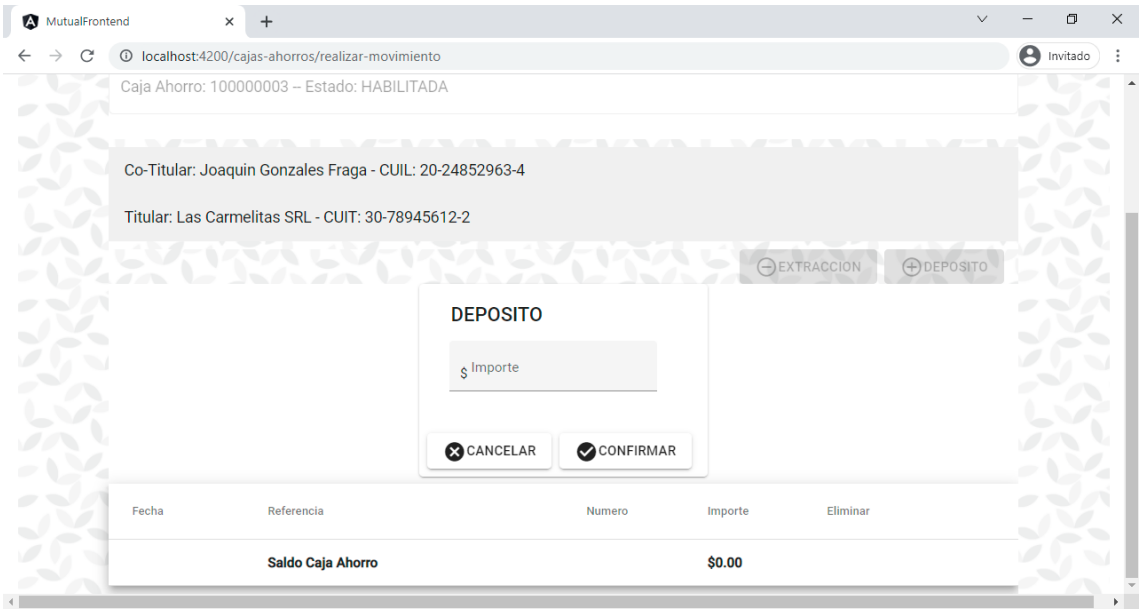



**Figura 85: Pantalla de registro de depósito/extracción 3.**

Una vez confirmado el movimiento, el sistema, actualizará el saldo de la caja de ahorros en pantalla, y agregará al listado de movimientos, este último generado.

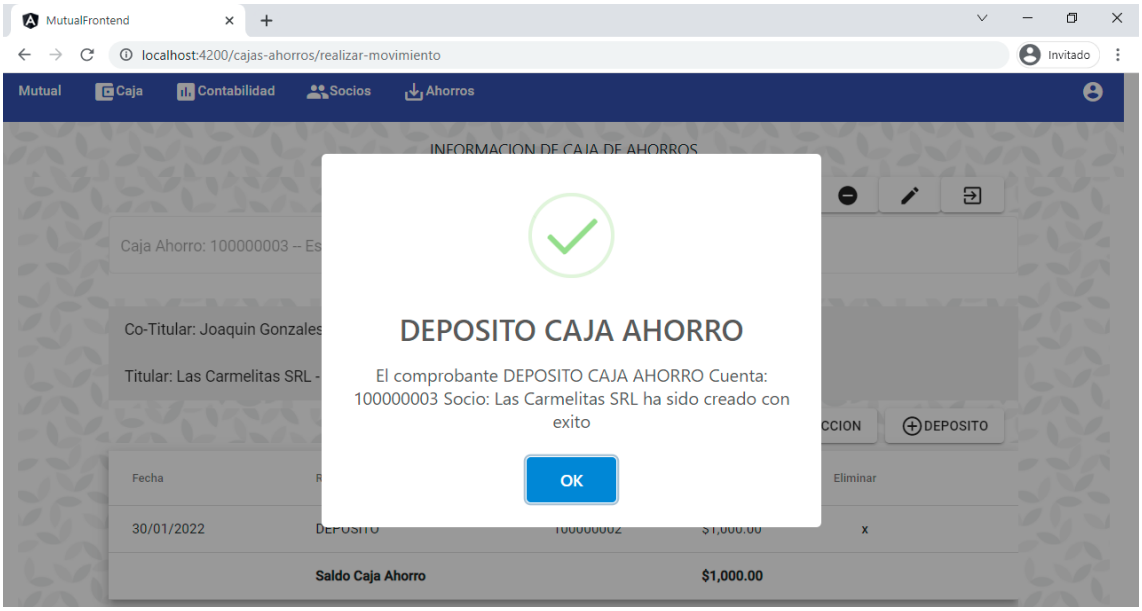



**Figura 86: Pantalla de registro de depósito/extracción 4.**

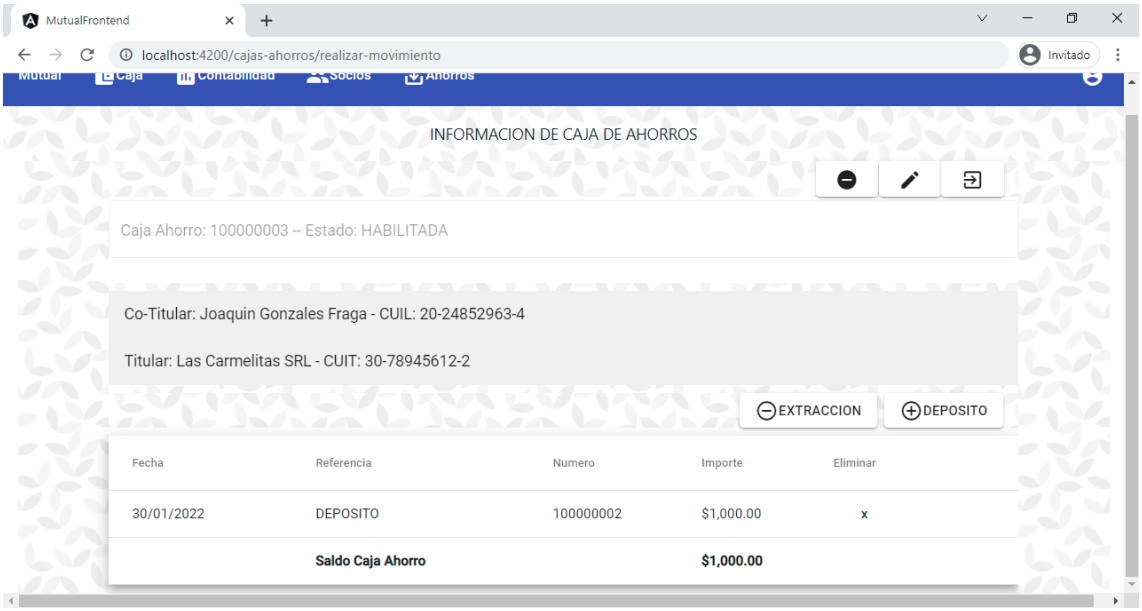



**Figura 87: Pantalla de registro de depósito/extracción 5.**

En la lista de movimientos de la cuenta, cada línea representa un movimiento, y posee un botón, para anular el dicho movimiento.

**Anular Deposito/Extracción**

Desde la interfaz de información de caja de ahorros, el usuario puede anular un depósito o extracción. Para realizar dicha acción, se debe dirigir al listado de movimientos de la caja de ahorros para encontrar el movimiento que busca. Luego en dicha línea, elegir la opción eliminar (que sólo se encontrará disponible si antes no fue anulado). El sistema pedirá confirmación. Luego el sistema actualizará la información del movimiento para indicar que fue anulado, y agrega a la lista de movimientos uno nuevo que compense al movimiento anulado.

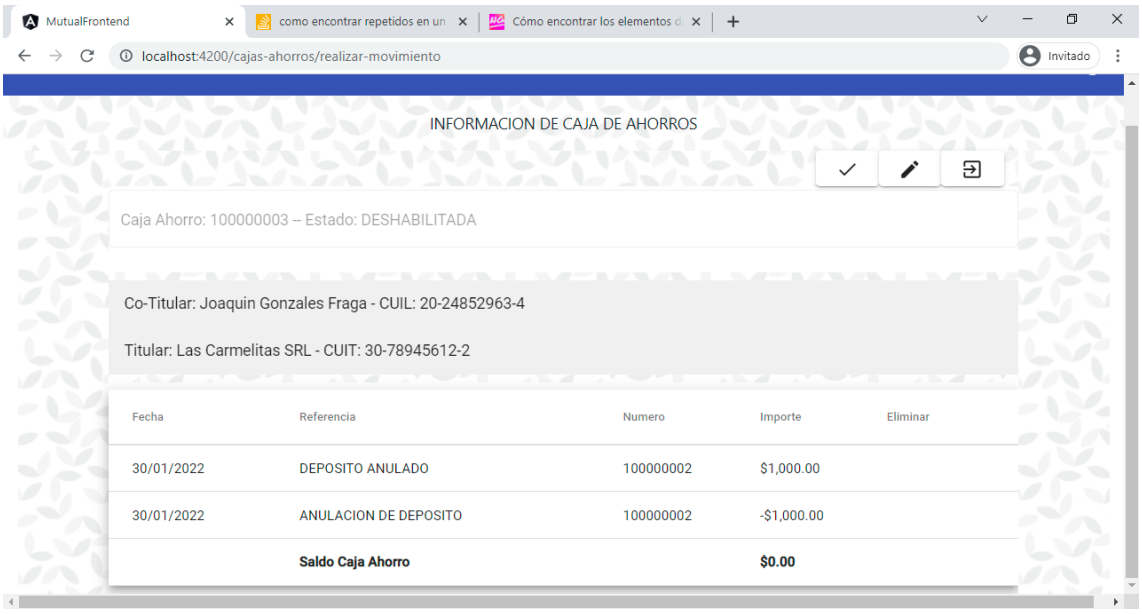



**Figura 88: Pantalla de anular de depósito/extracción.**

### Listado Por Fechas

Desde el menú principal de la aplicación, en ahorros, se puede acceder al submenú Listar movimientos.

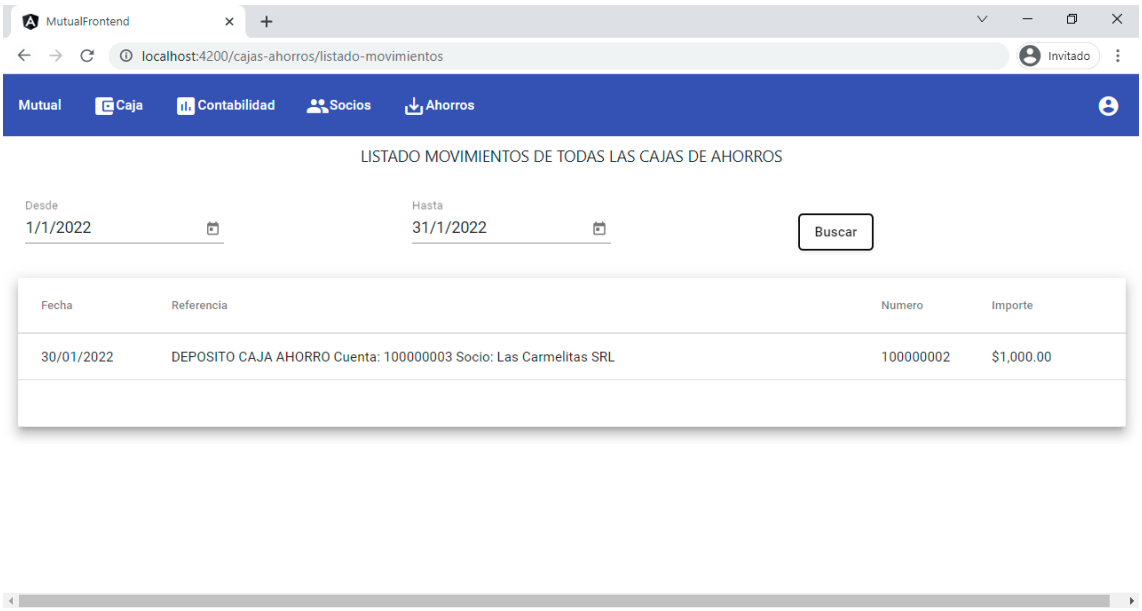



**Figura 89: Pantalla de listado de movimientos de cajas de ahorros por fecha.**

En la parte superior de esta pantalla, hay dos campos de fechas para filtrar los movimientos que queremos ver, así como también un botón para confirmar la búsqueda.

Debajo se desplegará una tabla, con un resumen de información de los movimientos que se generaron dentro de las fechas seleccionadas.

### Modificar Caja de Ahorros

Desde la interfaz de Información de Caja de Ahorros, del módulo de cajas de ahorros, se podrá ingresar en la opción editar, para modificar la información de la caja de ahorros. Se puede quitar o agregar cotitulares, también se puede cambiar las tasas deudora y acreedora. Para cambiar las tasas solo se deben agregar los porcentajes nuevos. Para agregar nuevos cotitulares, se deben buscar a través del campo de búsqueda que posee la interfaz, y seleccionar los correspondientes. Para eliminar un cotitular, se debe seleccionar en el botón (x) del listado de cotitulares de la cuenta.

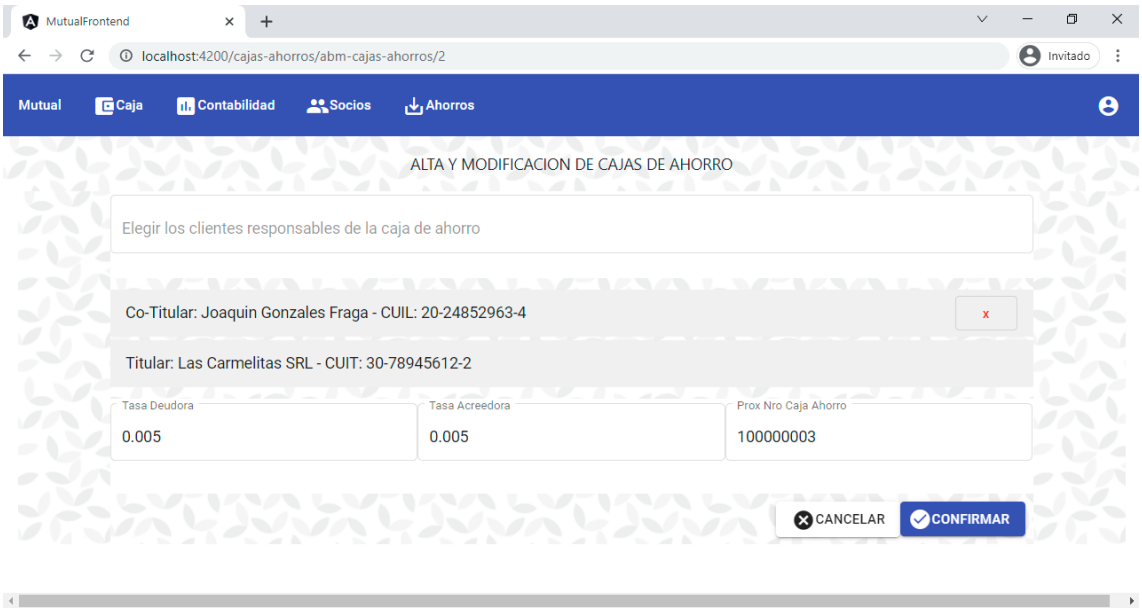



**Figura 90: Pantalla de modificar caja de ahorros.**

### Deshabilitar o Rehabilitar Caja de Ahorros

Desde la interfaz de Información de caja de ahorros, se puede dehabilitar o rehabilitar una caja de ahorros. Si la caja de ahorros se encuentra habilitada, entonces el botón disponible va a ser el de deshabilitar. Si el usuario desea deshabilitar una caja de ahorros puede seleccionar dicha opción y el sistema le pedirá confirmación.

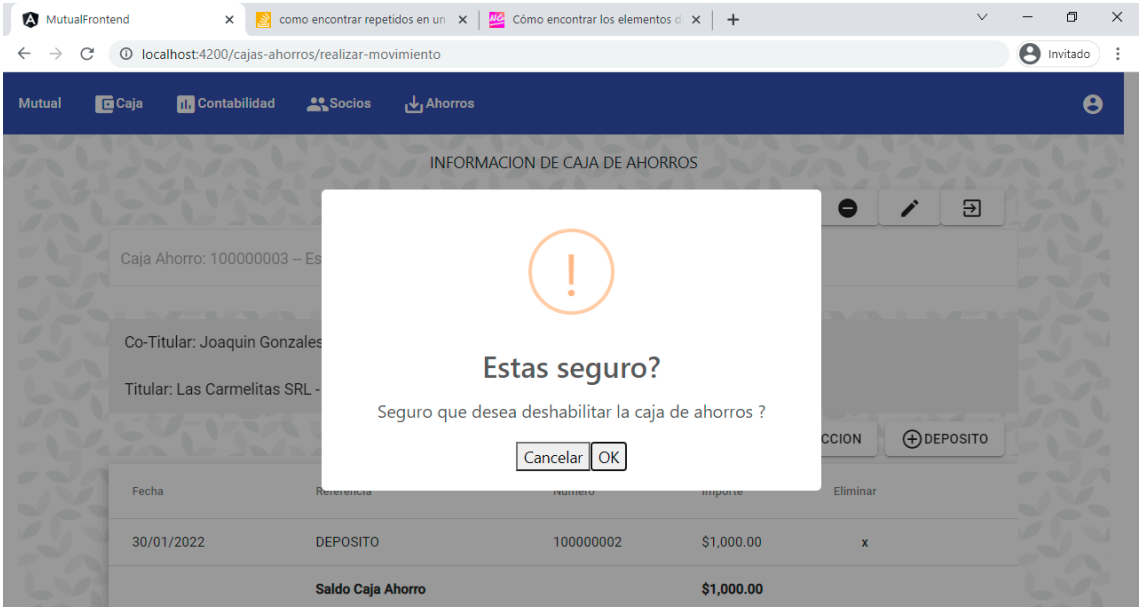



**Figura 91: Pantalla de rehabilitar/deshabilitar caja de ahorros 1.**

Luego que el usuario confirme, la caja de ahorros pasará a estar deshabilitada y no se podrá realizar depósitos o extracciones en la misma. Como se puede ver en las siguientes imágenes, las opciones de depósito o extracción desaparecen.

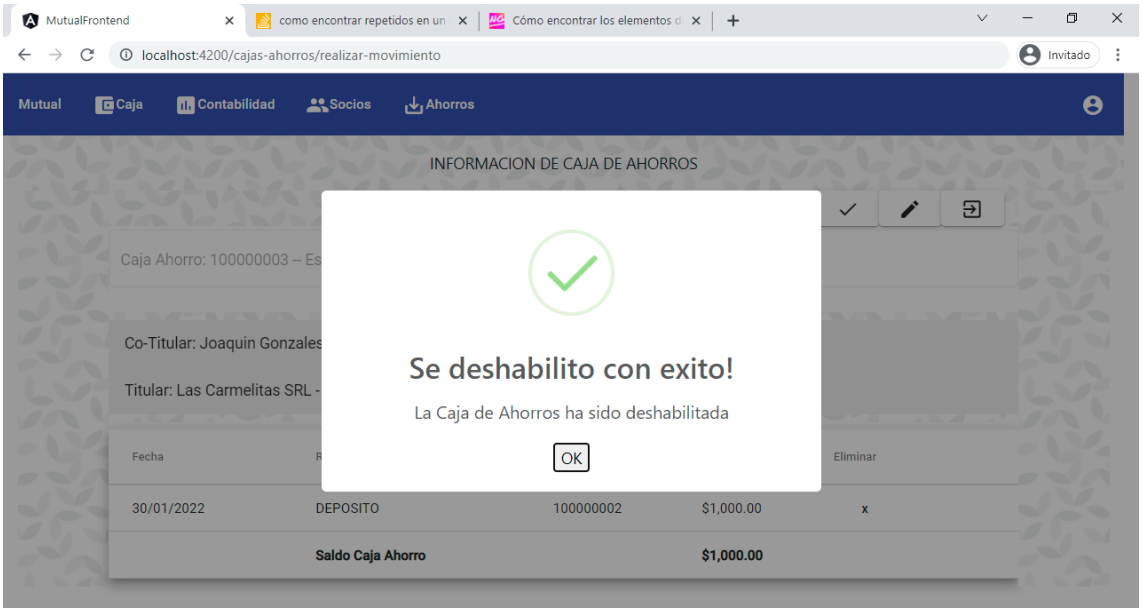



**Figura 92: Pantalla de rehabilitar/deshabilitar caja de ahorros 2.**

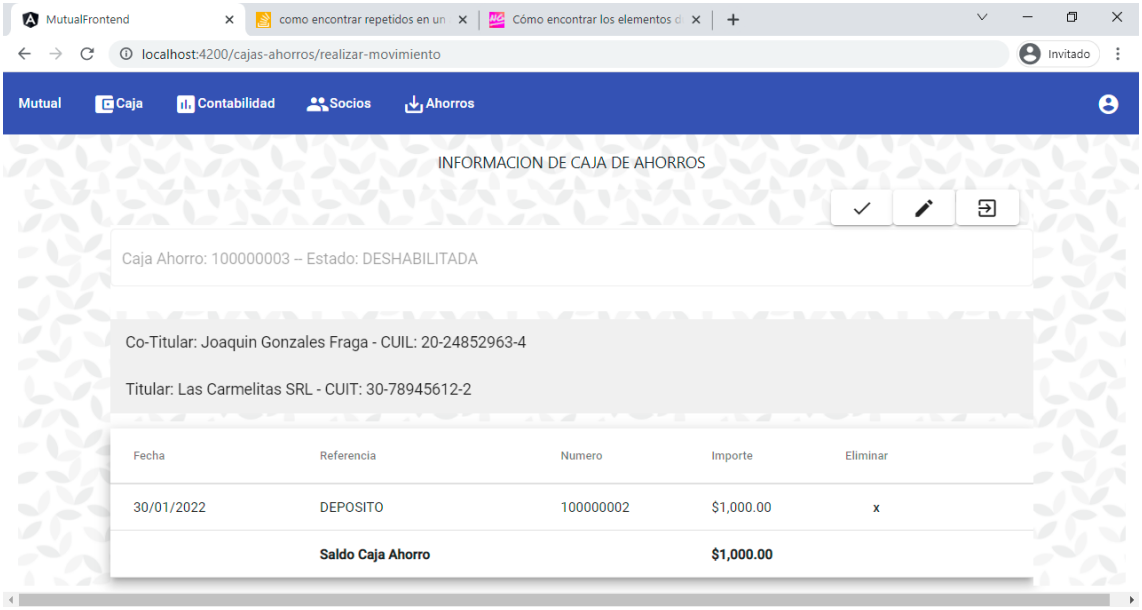



**Figura 93: Pantalla de rehabilitar/deshabilitar caja de ahorros 3.**

## Módulo de Contabilidad

### Listado de Cuentas Contables

Desde el menú principal de la aplicación, en el módulo de contabilidad se debe seleccionar Listar Cuentas. El sistema desplegará un listado de todas las cuentas contables, ordenadas por números de cuenta y agrupadas por rubros y sub-rubros contables.

Cada línea, representa una cuenta contable, y en cada una ellas poseen un botón para editar su información, como así también otro botón para agregar una cuenta dentro de su rubro.

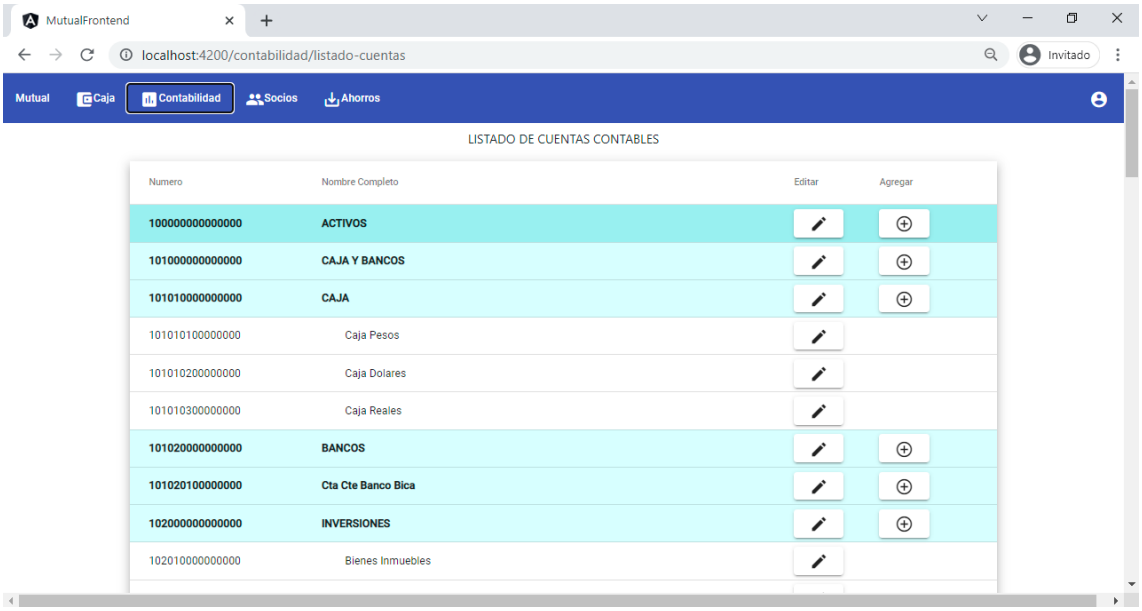



**Figura 94: Pantalla de listado de cuentas contables.**

### Alta de Cuenta Contable

Desde la interfaz de Listado Cuentas, se debe ingresar en la opción (+) dentro de la línea de la cuenta donde se desea agregar una nueva cuenta contable dentro del rubro de dicha cuenta.

Una vez que el usuario presiona dicho botón, el sistema desplegará la interfaz que se detalla en la siguiente imagen. En esta podemos ver que están disponibles para completar los campos de nombre de cuenta, número de cuenta, imputable/no imputable y viene completado el nivel de la cuenta dentro de la jerarquía de cuentas.

También en dicha interfaz vemos que el sistema detalla cuál es la cuenta superior en la jerarquía de cuentas.

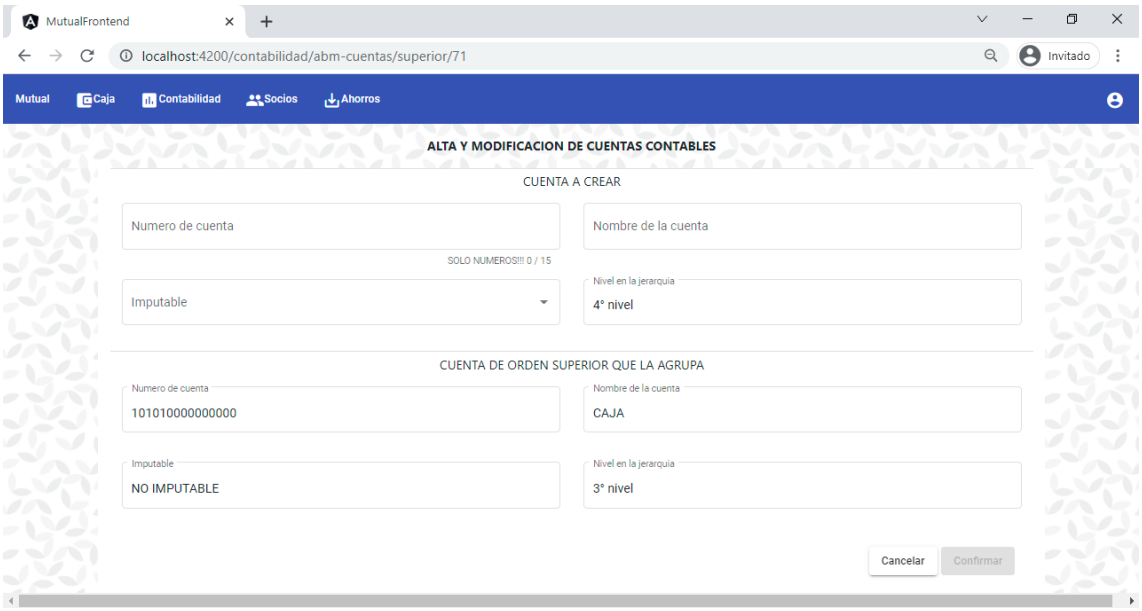



**Figura 95: Pantalla de alta de cuenta contable.**

**Modificar Cuenta Contable y Eliminar Cuenta**.

Desde la interfaz de Listado Cuentas, se debe ingresar en la opción editar dentro de la línea de la cuenta, de la cual se quiere modificar su información.

Una vez que el usuario presiona dicho botón, el sistema desplegará la interfaz que se detalla en la siguiente imagen. En esta podemos ver que están disponibles para editar los campos de nombre de cuenta, número de cuenta, imputable/no imputable.

También en dicha interfaz vemos que el sistema detalla cuál es la cuenta superior en la jerarquía de cuentas.

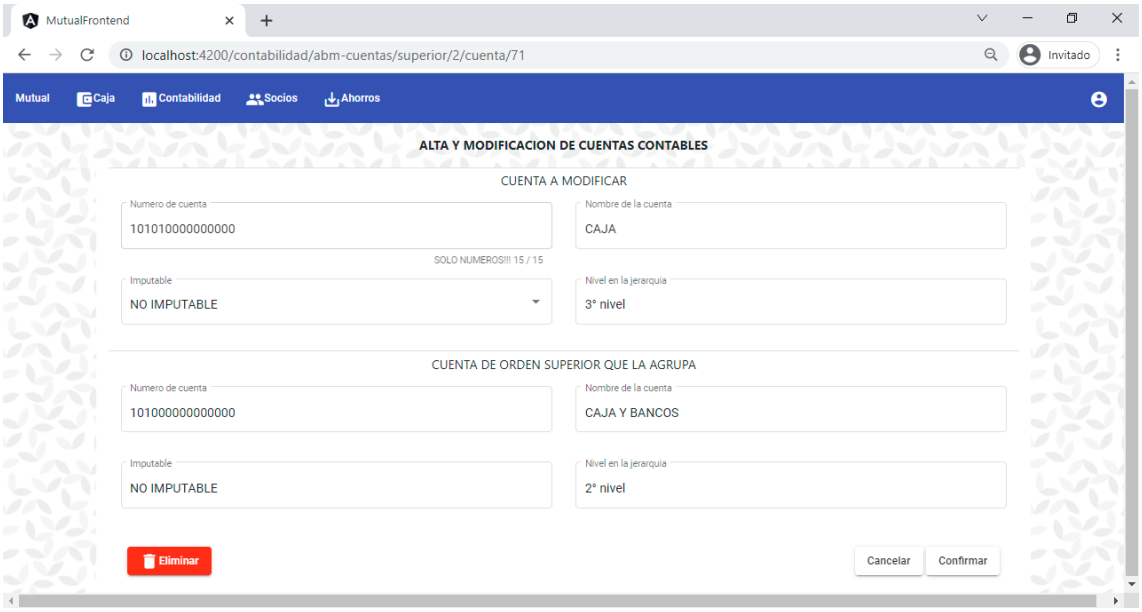


**Figura 96: Pantalla de modificar cuenta contable.**

En la parte inferior podemos ver que existe un botón “Eliminar”, para eliminar una cuenta contable. El usuario deberá presionar dicho botón, y el sistema pedirá que se confirme la acción, pero el sistema solo permitirá que se elimine una cuenta contable que no posee movimientos asociados.

**Balance de Saldos de Cuentas Contables del Mes**

Desde el menú principal de la aplicación, en el módulo de contabilidad se debe seleccionar Balance Mensual. Luego elegir el mes, del que se necesita conocer los saldos de las cuentas contables. El sistema desplegará un listado de todas las cuentas contables, ordenadas por números de cuenta y agrupadas por rubros y sub-rubros contables.

Cada línea, representa una cuenta contable, y el sistema detalla el total de movimientos deudores del mes, el total de movimientos acreedores del mes y un saldo que surge de la resta entre los movimientos deudores con los acreedores.

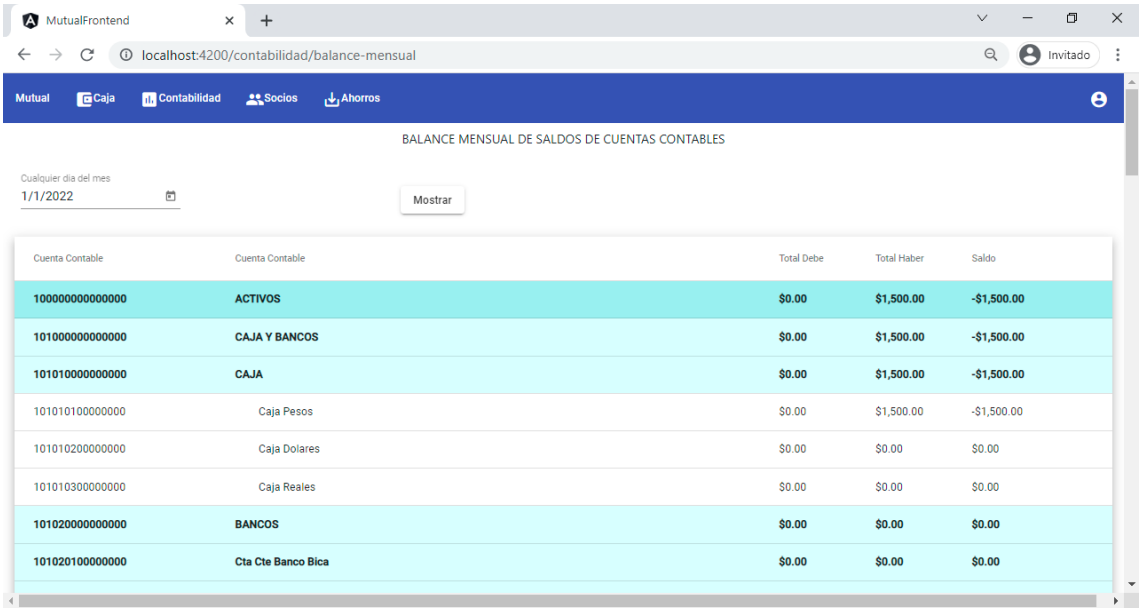



**Figura 97: Pantalla de balance de saldos de cuentas contables del mes.**