1. INTRODUCCION

La “Universidad Autónoma Juan Misael Saracho” como institución de educación superior orienta su accionar a un proceso de educación continua, abierta y permanente, acorde con el avance científico-tecnológico. Busca nuevas formas y métodos de enseñanza y aprendizaje que involucre todos los postulados de la nueva tecnología informática.

El rápido desarrollo de la tecnología y de la informática a proporcionando herramientas revolucionarias en todos los campos de la ciencia tecnológica. En este sentido vemos que los principales beneficiados de estos avances son las personas y empresas, un ejemplo de estos avances son los dispositivos móviles como los smartphones y tablets son el principal ejemplo de tendencia tecnológica. Hoy en día las empresas y negocios locales se adaptan al avance tecnológico, ya que se ven obligadas a disponer de herramientas que le ayuden a expandir su funcionamiento y al mismo tiempo publicidad

La principal herramienta de los dispositivos móviles son las aplicaciones móviles por aquello en los últimos años se ha observado su proliferación, éstas cumplen funciones dentro de nuestro teléfono, ya sea para comunicarnos, entretenernos o ayudarnos, asimismo permiten a empresas ofrecer nuevos servicios a sus clientes y generar expectativas permitiendo que las organizaciones conduzcan su negocio de forma más práctica y ofrecer nuevos beneficios y servicios. Un concepto erróneo que tienen inicialmente los emprendedores es que las aplicaciones son para grandes empresas, pero las apps son importantes en cualquier modelo de negocio sea para emprendedores o startup. Con base en el crecimiento de las apps en el mercado móvil, según un estudio de la firma de investigación Yankee Group, las instalaciones de App seguirán creciendo en todo el mundo, de 134,150 millones de instalaciones en 2014 a 341,890 instalaciones en 2018.

El presente trabajo de titulación está orientado al desarrollo de una aplicación móvil para un entorno Android, como medio que facilite la información en el módulo administrativo el registro de los diferentes platos, registro de las personas pensionadas, registro de la ubicación

del local y permita la programación del menú para la visualización del menú en el módulo cliente que sería los consumidores del establecimiento Sabores de Campo en la ciudad de Tarija.

1.2. Planteamiento del Problema

Hoy en día sigue existiendo la manera tradicional de ordenar un pedido, esto sobrelleva a realizar carga operativa al empleado, no satisface la calidad del servicio que provee a sus clientes. Se presenta lentitud en el proceso de atención al cliente al realizar sus pedidos en el establecimiento, el inconveniente se da porque el cliente no tiene conocimiento de los productos, las bebidas disponibles y precios de los mismos agregan valor al tiempo que lleva el cliente dentro del establecimiento hasta que tome la decisión de ordenar, cancelar y retirar su orden por la falta de una propuesta tecnológica de una aplicación móvil para pedidos y reservas de comida en la empresa "Sabores de Campo".

1.3. Objetivo general

Desarrollar una aplicación móvil para el entorno Android, que permita la gestión de pedidos y reservas y sirva como herramienta tecnológica viable, para mejorar el proceso de atención al cliente en la Empresa Sabores de Campo.

1.4. Objetivos específicos

- Diseñar un interfaz amigable e interactiva para su fácil identificación.
- Definir los requerimientos funcionales para el desarrollo de la aplicación móvil.
- Facilitar la información actualizada del menú y el precio a los clientes.
- Diseñar una base de datos específica para el llenado de la información.
- Utilizar la metodología Scrum para el desarrollo de la aplicación y utilizar UML para el modelado.

1.5. Justificación

El desarrollo de aplicaciones móviles es una herramienta tecnológica de gran utilidad, las empresas las utilizan como un canal de comunicación rápida, de esta manera los usuarios acceden a información desde cualquier lugar de manera efectiva con una mínima conexión a internet.

1.5.1. Justificación Social

A través de la Ampliación Móvil nos ahorra tiempo y esfuerzo porque tiene el acceso directo y la información específica sobre el restaurante donde el usuario podrá obtener la información del menú del día y hacer su consulta o reserva, esta aplicación está disponible para todas las personas ya que es de fácil instalación y manipulación.

1.6. ALCANCES Y LIMITACIONES

1.6.1. Alcances

- La App contará con una base de datos de acuerdo a requerimiento de la información de la empresa.
- La App contará con una interfaz de módulo administrativa y módulo cliente.
- La App del módulo administrativa se ingresará con usuario y clave.
- La App del módulo cliente tendrá acceso libre.
- La App registrará la programación del menú de la semana.
- La App del cliente mostrará la programación del menú.
- La App del cliente podrán ver los diferentes menús que ofrece la empresa.
- La App del cliente podrá comunicarse sólo con una llamada directa para hacer su consulta y con mensajes mediante el acceso directo al WhatsApp.
- La aplicación cliente mostrará la ubicación de la empresa.

1.6.2. Limitaciones

- La App estará disponible para celulares que tengan Android 7.0 para adelante.
- La App será implementado en el idioma español.
- El módulo administrativo soportará un solo usuario.
- La Aplicación administrativa ingresará con usuario y clave.
- El módulo cliente de la aplicación funcionará si tiene el celular conectado a internet.
- En el módulo cliente de la aplicación sólo podrá comunicarse con llamadas y con mensajes mediante el acceso directo al WhatsApp.

2. MARCO TEÓRICO

2.1. Aplicaciones móviles

Las aplicaciones móviles son programas diseñados para ser ejecutados en teléfonos, Tablet y otros dispositivos móviles, que permiten al usuario realizar actividades profesionales, acceder a servicios, mantenerse informado, entre otro universo de posibilidades llamamos feature phones, en contraposición a los smartphones, más actuales.

Es un software se instala en un dispositivo móvil de gama alta ya sea teléfono o tableta y que se puede integrar a las características del equipo, ampliando sus funciones. Se pueden ejecutar con o sin conexión a internet (Alegsa, 2017).

Tener una app móvil hoy en día para nuestra empresa es vital:

Fortalecimiento de marca: Tu empresa se destacará de la competencia al tener una aplicación móvil mediante la cual tus usuarios puedan interactuar contigo de una manera que no puedes conseguir mediante otros canales.

Mayor visibilidad: Una aplicación móvil, al estar hospedada en las principales tiendas de aplicaciones como Google Play Store y App Store de Apple, estará disponible para miles de usuarios interesados en tu producto.

Otro canal de venta: Al desarrollar una app para tu negocio, no sólo estarás consiguiendo fortalecer tu marca, sino que también estarás abriendo un nuevo canal de venta desde donde los usuarios podrán realizar compras de tus productos como lo harían desde una sucursal física o una tienda en línea.

Velocidad: A diferencia de una aplicación web donde la velocidad de carga es de 2 a 5 segundos una app móvil funciona mucho más rápido, al haberse descargado en los dispositivos de tus usuarios.

Usabilidad: Tu aplicación móvil podrá ser como tú quieras, con un diseño orientado a tener una gran experiencia de usuario y obtener los máximos resultados de venta para tu negocio.

Notificaciones: Podrás crear una relación más fuerte con tus usuarios que la que podrías conseguir utilizando otros medios, ya que estarás tan cerca de ellos como ellos lo están de sus dispositivos. Hablamos de que podrás enviarle notificaciones con promociones, descuentos, ofertas, encuestas y mucho más.

2.2. Tipos de aplicaciones

A nivel de programación, existen varias formas de desarrollar una aplicación. Cada una de ellas tiene diferentes características y limitaciones, especialmente desde el punto de vista técnico. Existen tres tipos de Apps que son las siguientes.

2.2.1. Aplicaciones Nativas

Las aplicaciones nativas son aquellas que han sido desarrolladas con el software que ofrece cada sistema operativo, llamado genéricamente Software Development Kit o SDK. Android, iOS y Windows Phone tienen una aplicación diferente y las aplicaciones nativas se diseñan y programan específicamente para cada plataforma, en el lenguaje utilizado por el SDK, cuando hablamos de desarrollo móvil casi siempre nos estamos refiriendo a aplicaciones nativas (Ramirez, 2014).

Por ejemplo:

- Las apps para iOS se desarrollan con lenguaje Objective-C
- Las apps para Android se desarrollan con lenguaje Java
- Las apps en Windows Phone se desarrollan en .Net

Las aplicaciones nativas se actualizan continuamente y el usuario debe volver a descargarlas para obtener la última versión, que a veces corrige errores o añade mejoras. Una característica generalmente menospreciada es que pueden hacer uso de las notificaciones del S.O. para mostrar avisos importantes al usuario, aun cuando no se esté usando la aplicación, como los mensajes de WhatsApp.

Ventajas	Inconvenientes
• Acceso completo al dispositivo. • Mejor experiencia del usuario. • Visibilidad en APP Store. • Envío de notificaciones o avisos a los usuarios.	• Diferentes habilidades/ idiomas/ herramientas para cada plataforma de destino. • Tienden a ser más caras de desarrollar.

• La actualización de las apps es constante.	• El código del cliente no es reutilizable entre las diferentes plataformas.

Fuente: Elaboración propia

2.2.2. Aplicaciones Web App

Es la desarrollada con lenguajes muy conocidos por los programadores, como es el HTML, Javascript y CSS. La principal ventaja con respecto a la nativa es la posibilidad de programar independiente de la aplicación operativo en el que se usará la aplicación. De esta forma se pueden ejecutar en diferentes dispositivos sin tener que crear varias aplicaciones.

Las aplicaciones web se ejecutan dentro del propio navegador web del dispositivo a través de una URL. Por ejemplo, en Safari, si se trata de la plataforma iOS. El contenido se adapta a la pantalla adquiriendo un aspecto de navegación APP. (eumed.net, 2016)

¿Puede considerarse esto una APP? En realidad, la gran diferencia con una aplicación nativa (además de los inconvenientes que se muestran en la tabla) es que no necesita instalación por lo que no pueden estar visibles en app store y la promoción y comercialización debe realizarse de forma independiente. De todas formas, se puede crear un acceso directo que sería como "instalar" la aplicación en el dispositivo.

Las apps web móviles son siempre una buena opción si nuestro objetivo es adaptar la web a formato móvil.

Ventajas	Inconvenientes
• El mismo código base reutilizable en múltiples plataformas. • Proceso de desarrollo más sencillo y económico. • No necesitan ninguna aprobación externa para publicarse.	• Requiere conexión a internet. • Acceso muy limitado a los elementos y características del hardware del dispositivo. • La experiencia del usuario (navegación, interacción) y el

• El usuario siempre dispone de la última versión.	tiempo de respuesta es menor que una app nativa. • Requiere de mayor esfuerzo en promoción y visibilidad.

Fuente: Elaboración propia

2.2.3. Aplicaciones Híbridas

Este tipo de aplicaciones es una combinación entre las dos anteriores. La forma de desarrollarlas es parecida a la de una aplicación web, usando HTML, CSS y JavaScript, y una vez que la aplicación está terminada, se compila o empaqueta de tal forma, que el resultado final es como si se tratara de una aplicación nativa. Por aquello también son denominadas Web App nativa. Esto permite agrupar los códigos, obtener diferentes aplicaciones y permite su uso en cada plataforma y distribuirlas en cada una de sus tiendas (Ramirez, 2014).

A diferencia de las aplicaciones web, éstas permiten acceder, usando librerías, a las capacidades del teléfono, tal como lo haría una app nativa.

Ventajas	Inconvenientes
• Es posible distribuirla en las tiendas de IOS y Android. • Instalación nativa pero construida con JavaScript, HTML y CSS. • El mismo código base para múltiples plataformas. • Acceso a parte del hardware del dispositivo.	• Experiencia del usuario más propia de la aplicación web que de la app nativa. • Diseño visual no siempre relacionado con la aplicación operativo en el que se muestre.

Fuente: Elaboración propia

2.3. SISTEMA OPERATIVOS PARA MÓVILES.

Un sistema operativo (SO) móvil controla un dispositivo móvil al igual que los PC utilizan Windows o Linux entre otros. Sin embargo, los sistemas operativos móviles son mucho más simples y están más orientados a la conectividad inalámbrica, los formatos. El sistema Operativo (SO) móvil de un teléfono o tableta realiza la interacción real con lo que podemos hacer a partir de las capacidades del hardware que conforman un equipo. A manera de traductor, esta plataforma interpreta lo que el usuario quiere que la terminal realice y cada vez, lo ejecuta con mayor inteligencia. Una de las cualidades más atractivas de un sistema operativo móvil es la rapidez con la que en general se desempeña (Rosa, 2016).

"A medida que los teléfonos móviles crecen en popularidad, los sistemas operativos con los que funcionan adquieren mayor importancia y tenemos los siguientes tipos:

- Android "84,7%
- iOS 11,7%
- Windows Phone 2,5%
- BlackBerry OS 0,5%
- Otros 0,6%

2.3.1. Android

"Android es una solución completa de software de código libre (GNU Linux7) para teléfonos y dispositivos móviles. Es un paquete que engloban un sistema operativo, un "Runtime8" de ejecución basado en Java9, un conjunto de librerías de bajo y medio nivel y un conjunto inicial de aplicaciones destinadas al usuario final. Android se distribuye bajo una licencia libre que permite la integración con soluciones de código propietario".

2.3.2. Node js.

Es un entorno de ejecución de JavaScript orientado a eventos asíncronos, Node está diseñado para construir aplicaciones en red escalables. En la siguiente aplicación de ejemplo "hola mundo", se pueden manejar muchas conexiones concurrentes.

Esto contrasta con el modelo de concurrencia más común hoy en día, donde se usan hilos del sistema Operativo. Las operaciones de redes basadas en hilos son relativamente ineficientes y son muy difíciles de usar. Además, los usuarios de Node están libres de preocupaciones sobre el bloqueo del proceso, ya que no existe. Casi ninguna función en Node realiza I/O directamente, así que el proceso nunca se bloquea. Debido a que no hay bloqueo es muy razonable desarrollar sistemas escalables en Node (Dafl, 2019).

2.3.3. Ionic

Es un framework que se está haciendo muy popular últimamente. Es una herramienta que los programadores pueden utilizar totalmente gratis, para desarrollar apps basadas en HTML5, CSS y JavaScript. Está construido con Sass y optimizado para AngularJS. Además, es libre y de código abierto, open source (Varela, 2016).

Ventajas:

- Animaciones Aceleradas por Hardware
- Mínima manipulación del DOM
- No usa JQuery.
- Fácil adaptación del diseño
- Soporte de Phonegap integrado

Desventajas:

- Hace falta conocimientos medios-avanzados para programar las aplicaciones en JavaScript
- Se tienen que tener conocimientos de AngularJS

Integra Córdova directamente, lo que permite que con una sola instalación ya se puede compilar directamente la aplicación a la plataforma que se desee o instalar los plugins de Córdova para utilizarlos en Ionic.

2.3.4. Angular

Es un framework de JavaScript de código abierto, mantenido por Google, que se utiliza para crear y mantener aplicaciones web de una sola página. Su objetivo es aumentar las aplicaciones

basadas en navegador con capacidad de Modelo Vista Controlador (MVC), en un esfuerzo para hacer que el desarrollo y las pruebas sean más fáciles. Angular lleva servicios tradicionales del lado del servidor, tales como controladores dependientes de la vista, a las aplicaciones web del lado del cliente. En consecuencia, gran parte de la carga en el backend se reduce, lo que conlleva a aplicaciones web mucho más ligeras.

AngularJS se puede combinar con el entorno en tiempo de ejecución Node.js, el framework para servidor Express.js y la base de datos firebase.

Velocidad y rendimiento

Generación de código: Angular convierte tus plantillas en código altamente optimizado para las máquinas virtuales de JavaScript de hoy en día, ofreciéndote todas las ventajas del código escrito a mano con la productividad de un framework.

Universal: Ejecuta la primera vista de tu aplicación en node.js, .NET, PHP, y otros servidores para renderizado de forma casi instantánea obteniendo sólo HTML y CSS. También abre posibilidades para la optimización del SEO del sitio.

División del código: Las aplicaciones de Angular se cargan rápidamente gracias al nuevo enrutador de componentes. Éste ofrece una división automática de códigos para que los usuarios sólo carguen el código necesario para procesar la vista que solicitan (Platzi, 2014).

Las principales ventajas son:

1. Es de fácil mantenimiento.
2. Una documentación consistente: toda la sintaxis y la manera de desarrollar es la misma, esto añade coherencia a la información y a la forma de leer el código.
3. TypeScript proporciona autocompletado avanzado, navegación y refactorización. Tener tales herramientas es casi un requisito para grandes proyectos. Sin ellos, el temor de cambiar el código coloca al código base en un estado de semi-solo lectura y hace que las refactorizaciones a gran escala sean muy riesgosas y costosas.
4. Tiene una interfaz de usuario declarativa: Para la definición del interfaz de usuario, Angular utiliza HTML y dado que ésta es un lenguaje declarativo se considera que es más intuitivo que utilizar javascript para definir la interfaz de manera manual.

5. Angular es, un proyecto totalmente open source “código abierto”, se encuentra publicado en la página de Github y con constantes colaboraciones por parte de su comunidad.

2.4. Córdova

Apache Córdova es un marco de desarrollo móvil de código abierto. Permite utilizar las tecnologías estándar web como HTML5, CSS3 y JavaScript para desarrollo multiplataforma, evitando el lenguaje de desarrollo nativo cada plataforma móvile. Aplicaciones ejecutan dentro de envolturas para cada plataforma y dependen de enlaces estándares API para acceder a cada dispositivo.

Apache Córdova se graduó en octubre de 2012 como un proyecto de nivel superior dentro de la Apache Software Foundation (ASF). A través del ASF, futuro desarrollo Córdova asegurará administración abierta del proyecto. Siempre permanecerá libre y de código abierto bajo la licencia Apache (cordova, 2016).

2.5. TypeScript

Es un lenguaje de programación de código abierto (de uso libre y gratuito) desarrollado por Microsoft. Se dice que es un superconjunto de JavaScript, lo que viene a significar que es un lenguaje “montado” sobre otro lenguaje.

Digamos que el propósito de TypeScript es cubrir las carencias o deficiencias del lenguaje JavaScript, haciéndolo un lenguaje más completo, escalable, seguro e incluso más fácil de utilizar. Tiene un gran parecido al nuevo estándar de JavaScript, ECMAScript6, el cual en la actualidad aún no está completamente implementado, así que de cierta manera TypeScript nos acerca al futuro de JavaScript.

Sus principales características frente al JavaScript es el tipado estático (o sea, declaración del tipo de dato que debe alojar cada variable) y la programación orientada a objetos basado en clases, lo cual, aunque de alguna manera era posible con JavaScript, resultaba complicado ya que el lenguaje no está pensado para ello.

Aunque Angular no te obliga a usar TypeScript, el equipo del core de Angular sí que lo ha adoptado y en la documentación sugiere usar TypeScript por defecto. Esto implica que los

ejemplos relacionados y los proyectos de código abierto parezcan más familiares y consistentes. Angular ya ofrece ejemplos claros que enseñan cómo usar el compilador TypeScript (tutorialesenpdf, 2016)

2.6. PROCESO DE DISEÑO Y DESARROLLO DE UNA APP

Según (Cuello & Vittone , 2013) en su libro explican brevemente sobre el proceso de diseño y desarrollo de una app, donde detalla desde la concepción de la idea hasta el análisis posterior a su publicación en las tiendas.

Indicando que durante las diferentes etapas, diseñadores y desarrolladores trabajan la mayor parte del tiempo de manera simultánea y coordinada.

Figura 1: Etapas de la aplicación

2.6.1. Conceptualización

El resultado de esta etapa es una idea de aplicación, que tiene en cuenta las necesidades y problemas de los usuarios. La idea responde a una investigación preliminar y a la posterior comprobación de la viabilidad del concepto. En este proceso se da paso a la idea, investigación y formalización de la misma.

2.6.2. Definición

En este paso del proceso se describe con detalle a los usuarios para quienes se diseñará la aplicación, usando metodologías como «Personas» y «Viaje del usuario». También aquí se sientan las bases de la funcionalidad, lo cual determinará el alcance del proyecto y la

complejidad de diseño y programación de la app. En esta etapa se define usuarios y requerimiento funcional.

2.6.3. Diseño

En la etapa de diseño se llevan a un plano tangible los conceptos y definiciones anteriores, primero en forma de wireframes, que permiten crear los primeros prototipos para ser probados con usuarios, y posteriormente, en un diseño visual acabado que será provisto al desarrollador, en forma de archivos separados y pantallas modelo para la programación del código. En síntesis se crean los prototipos de pantalla, se crea un diseño visual para realizar un test con los usuarios.

2.6.4. Desarrollo

El programador se encarga de dar vida a los diseños y crear la estructura sobre la cual se apoyará el funcionamiento de la aplicación. Una vez que existe la versión inicial, dedica gran parte del tiempo a corregir errores funcionales para asegurar el correcto desempeño de la app y la prepara para su aprobación en las tiendas.

2.6.5. Publicación

La aplicación finalmente es puesta a disposición de los usuarios en las tiendas. Luego de este paso importante se realiza un seguimiento a través de analíticas, estadísticas y comentarios de usuarios, para evaluar el comportamiento y desempeño de la app, corregir errores, realizar mejoras y actualizarla en futuras versiones.

2.7. Firebase

Firebase es un conjunto de herramientas orientadas a la creación de aplicaciones de alta calidad. Es la nueva y mejorada plataforma de desarrollo móvil en la nube de Google. Se trata de una plataforma disponible para diferentes plataformas (Android, iOS, web), Con la base en tiempo real de Firebase podrás guardar todos los datos que requiera tu aplicación. Se lleva muy bien con React y su patrón reactivo que permite actualizar los datos en los componentes automáticamente. Los datos se almacenan en formato JSON y se pueden agregar reglas para permitir requests con token o sólo desde una URL. (Zamora, 2016).En conclusión, es una plataforma en la nube disponible para todos.

Imagen2: Firebase

Sus características fundamentales están divididas en varios grupos, las cuales podemos agrupar en:

•**Analíticas**: Provee una solución gratuita para tener todo tipo de medidas (hasta 500 tipos de eventos), para gestionarlo todo desde un único panel.

•**Desarrollo**: Permite construir mejores apps, permitiendo delegar determinadas operaciones en Firebase, para poder ahorrar tiempo, evitar bugs y obtener un aceptable nivel de calidad. Entre sus características destacan el almacenamiento, testeo, configuración remota, mensajería en la nube o autenticación, entre otras.

•**Crecimiento:** Permite gestionar los usuarios de las aplicaciones, pudiendo además captar nuevos. Para ello dispondremos de funcionalidades como las de invitaciones, indexación o notificaciones.

Esta herramienta se centra en ofrecer servicios a diferentes niveles.

Desarrollo, Crecimiento y Ganancias constituyen las principales áreas de Firebase y para cada una de ellas ofrece múltiples funcionalidades

- Las funcionalidades ofrecidas dentro de la categoría de desarrollo están enfocadas a facilitar el desarrollo rápido de apps manteniendo una alta calidad.
- Las funcionalidades ofrecidas dentro la categoría de crecimiento tienen como objetivo permitir aumentar la base de usuarios sin esfuerzo ni dificultad, nos aportan escalabilidad.

- Las funcionalidades ofrecidas dentro de la categoría de ganancias están orientadas a monetizar la app, a obtener beneficios a través de la misma.

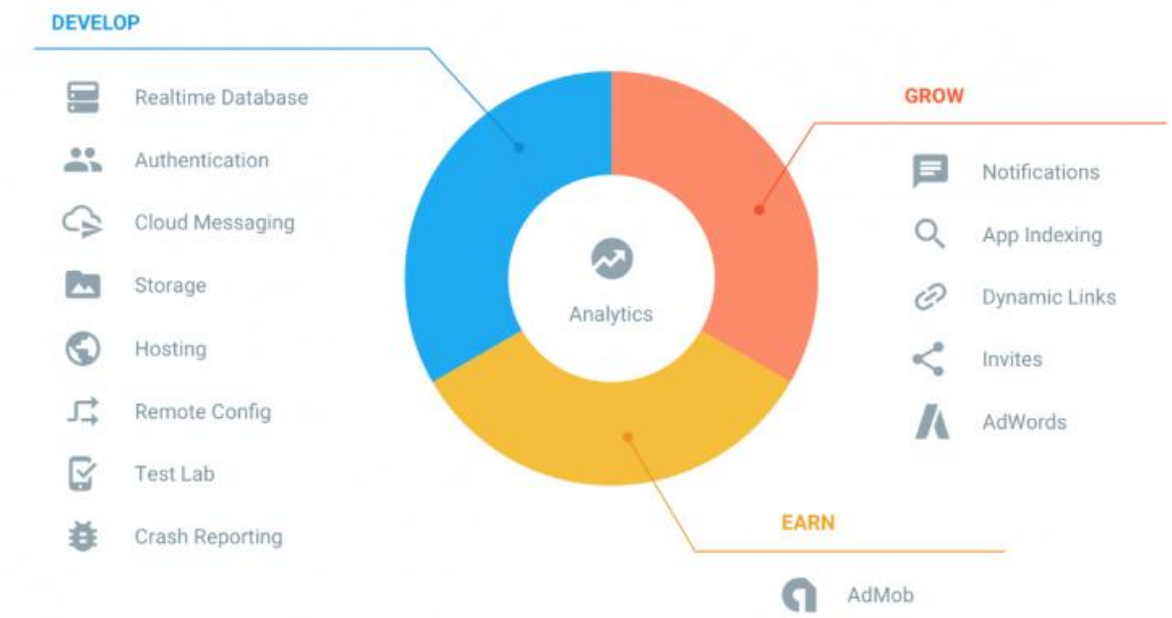


Ilustración 1 - Servicios Firebase

¿**Por qué firebase en nuestro proyecto**?

Uno de los pilares básicos sobre los que construir aplicaciones móviles hoy en día es la escalabilidad, a menudo vemos ejemplos de aplicaciones que se viralizan y terminan rehaciendo su aplicación o presentando otra versión de la misma debido a que la inicial no era escalable y no es capaz de afrontar la demanda de usuarios.

Encontrar el éxito y la viralización de una aplicación hoy en día es un tremendo reto en un mercado tan saturado, pero si encima ésta no es escalable, nuestra aplicación habrá muerto antes de ver la luz.

Firebase pertenece a Google, no obstante, no sólo está disponible para Android, sino que también para iOS, web y hoy en día también está disponible para plataformas como Unity. Además, la herramienta ofrece la confianza y madurez necesaria contando con el apoyo de un gigante como Google

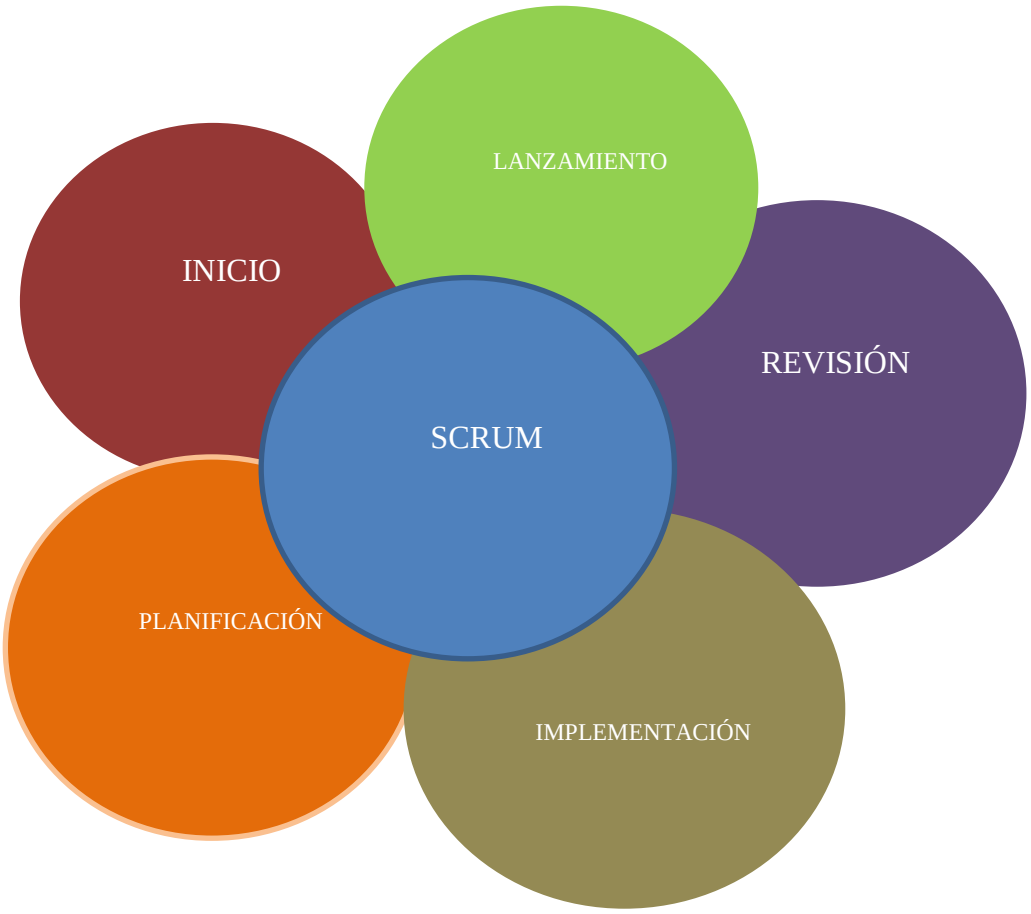
2.8. METODOLOGÍA SCRUM

Scrum es un proceso, marco de trabajo o framework, usado en equipos que trabajan en proyectos complejos con el fin de trabajar en equipo y obtener el mejor resultado para un proyecto; una metodología de trabajo ágil que tiene como finalidad la entrega de valor en períodos cortos de tiempo, basada tres pilares: la transparencia, inspección y adaptación.

Según el portal Proyectos Ágiles, en Scrum se realizan entregas parciales y regulares del producto final, priorizadas por el beneficio que aportan al receptor del proyecto. Scrum es útil para desarrollar proyectos en entornos complejos, que requieren resultados en el muy corto plazo y donde los requisitos cambian o están poco definidos. Es decir, proyectos donde son fundamentales aspectos como la innovación, la competitividad, la flexibilidad y la productividad.

Proyectos Ágiles menciona que también se utiliza Scrum para encontrar soluciones a problemas diversos como cuando no se entrega al cliente lo que necesita; cuando las entregas se alargan demasiado; cuando los costos se elevan considerablemente; cuando la calidad no es aceptable; cuando se necesita capacidad de reacción ante la competencia; cuando la moral de los equipos es baja y la rotación es alta; cuando es necesario identificar y solucionar ineficiencias sistemáticamente; y cuando se quiere trabajar utilizando un proceso especializado en el desarrollo de producto. (bahit, 2014).

2.8.1. Fases del scrum



Fuente: Elaboración propia

Inicio: En esta fase se crea la Visión del Proyecto que sirve de enfoque y dirección del mismo. Se crean e identifican roles claves del proyecto como el Scrum Master, Product Owner, interesados, equipo del proyecto. Así mismo, se define la lista de prioridades o el Product Backlog la cual sirve de base para la elaboración del plan de lanzamiento y tamaño de cada Sprints.

Planificación: Aquí se definen y aterrizan en los Sprints las historias de usuarios, se alinean a todo lo que genera valor a la organización y se hacen las estimaciones de tiempo y esfuerzo para cumplirlas, los cuales se traducen en listas de tareas cuyos tiempos de desarrollo se definen en reuniones de equipo correspondientes, así como el proceso de definición del Sprint Backlog que contiene todas las tareas que deben completarse en el Sprint.

Implementación: En esta fase se trabaja en las tareas del Sprint Backlog para crear Sprint Deliverables, para ello se utiliza a menudo un Scrumboard para realizar el seguimiento del trabajo y de actividades que se llevan a cabo. También, los inconvenientes o problemas que enfrenta el Equipo Scrum se actualizan en un Impediment Log. Durante esta fase se realizan las llamadas Daily Standup Meeting que son reuniones cortas y eficientes en tiempo donde el equipo da el estatus de sus actividades diarias y manifiesta cualquier inconveniente que pueda tener. Igualmente se actualiza o revisa la lista de prioridades de pendientes del producto.

Revisión: Para proyectos grandes que involucran varios equipos Scrum, se realiza en esta etapa, reuniones que permitan juntar a estos equipos y discutir y revisar avances, dependencias e impedimentos en el desarrollo del proyecto. También en esta etapa se lleva a cabo el proceso donde el Equipo Scrum lo demuestra el Sprint Deliverable al Propietario del producto y a los Socios relevantes en un Sprint Review Meeting. Igualmente, el Scrum Master y el Equipo Scrum se reúnen para discutir las lecciones aprendidas a lo largo del Sprint, información que se documenta como las lecciones aprendidas que pueden aplicarse a los futuros Sprints.

Lanzamiento: Finalmente, ésta es la fase más esperada por los interesados o socios del proyecto, así como del Scrum Master y Equipo Scrum. En esta fase se la entrega del producto a los Socios relevantes. Un acuerdo formal llamado Working Deliverables Agreement documenta la finalización con éxito del Sprint. Del mismo modo, se realizan actividades de restrospectiva que permite identificar mejoras y lecciones aprendidas del proyecto.

2.9. Uml.

El Lenguaje Unificado de Modelado (UML) fue creado para forjar un lenguaje de modelado visual común y semántica y sintácticamente rico para la arquitectura, el diseño y la implementación de aplicaciónes de software complejos, tanto en estructura como en comportamiento. UML tiene aplicaciones más allá del desarrollo de software, p. ej., en el flujo de procesos en la fabricación.

Es comparable a los planos usados en otros campos y consiste en diferentes tipos de diagramas. En general, los diagramas UML describen los límites, la estructura y el comportamiento de la aplicación y los objetos que contiene.

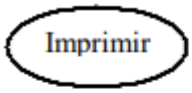
UML no es un lenguaje de programación, pero existen herramientas que se pueden usar para generar código en diversos lenguajes usando los diagramas UML. UML guarda una relación directa con el análisis y el diseño orientados a objetos. (Anacleto, 2006).

A continuación, se describirán los diagramas más comunes del UML y los conceptos que representan:

- Diagrama de Casos de Uso
- Diagrama de Actividades
- Diagrama de Secuencias
- Diagrama de colaboración

Diagrama de casos de uso

Un caso de uso es una descripción de las acciones de la aplicación desde el punto de vista del usuario. Es una herramienta valiosa dado que es una técnica de aciertos y errores para obtener los requerimientos de la aplicación, justamente desde el punto de vista del usuario. Los diagramas de caso de uso modelan la funcionalidad de la aplicación usando actores y casos de uso. Los casos de uso son servicios o funciones provistas por la aplicación para sus usuarios.



Casos de Uso

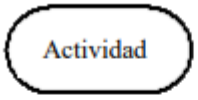
Se representan con óvalos. La etiqueta en el óvalo indica la función del sistema.

Actores

Los *actores* son los usuarios de un sistema.

Diagrama de actividad: Un diagrama de actividades ilustra la naturaleza dinámica de una aplicación mediante el modelado del flujo ocurrente de actividad en actividad. Una actividad representa una operación en alguna clase de la aplicación y que resulta en un cambio en el estado

de la aplicación. Típicamente, los diagramas de actividad son utilizados para modelar el flujo de trabajo interno de una operación.



Estados de Acción

Los *estados de acción* representan las acciones no interrumpidas de los *objetos*.

Flujo de la Acción

Los *flujos de acción*, representados con flechas, ilustran las relaciones entre los *estados de acción*.

Estado Inicial

Estado inicial de un estado de *acción*.

Final State

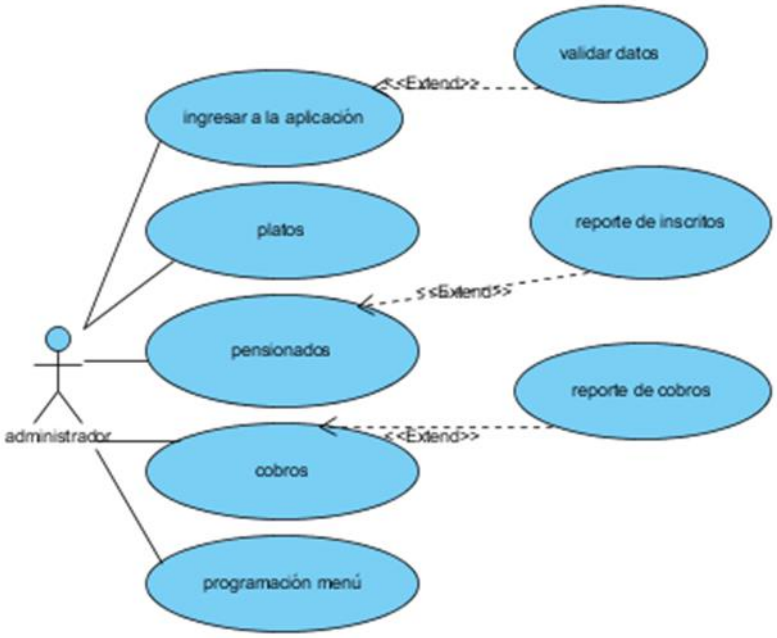
Estado final de un estado de *acción*.

Diagrama de secuencia: El diagrama de secuencia es un tipo de diagrama de interacción cuyo objetivo es describir el comportamiento dinámico de la aplicación de información haciendo énfasis en la secuencia de los mensajes intercambiados por los objetos.

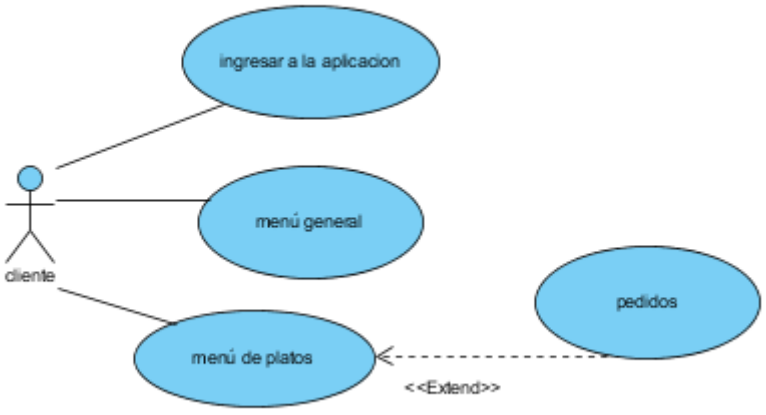
Diagrama de colaboración: Es esencialmente un diagrama que muestra interacciones organizadas alrededor de los roles. A diferencia de los diagramas de secuencia, los diagramas de colaboración, también llamados diagramas de comunicación, muestran explícitamente las relaciones de los roles. Por otra parte, un diagrama de comunicación no muestra el tiempo como una dimensión aparte, por lo que resulta necesario etiquetar con números de secuencia tanto la secuencia de mensajes como los hilos concurrentes.

3. ANÁLISIS

3.1. Caso de uso para el rol administrador



3.2. Caso de uso para el rol cliente



3.3. Descripción de los casos de uso mediante el sprint

	Caso de uso	Tareas	Díaz	Horas
Sprint 1	Registro Personal	Modelar el Análisis. prototipo de Interfaz. Actualizar arquitectura. Implementar funcionalidad. Probar registro de datos. Implementar backend restante de la funcionalidad.	20	160
	Registro platos	•Modelar el Análisis. prototipo de Interfaz. •Actualizar arquitectura. •Implementar funcionalidad. •Probar registro de datos. •Implementar backend restante de la funcionalidad.	25	200
Sprint 2	Programación de menú	•Modelar el Análisis. prototipo de Interfaz. •Actualizar arquitectura. •Implementar funcionalidad. •Probar registro de datos. •Implementar backend restante de la funcionalidad.	29	232
	Búsqueda de platos	•Modelar el Análisis. prototipo de Interfaz. •Actualizar arquitectura. •Implementar funcionalidad. •Probar registro de datos. •Implementar backend restante de la funcionalidad.	15	120
Sprint 3	Registro cobros	•Modelar el Análisis. prototipo de Interfaz. •Actualizar arquitectura. •Implementar funcionalidad. •Probar registro de datos. •Implementar backend restante de la funcionalidad.	18	144

	Búsqueda de pensionados	•Modelar el Análisis. prototipo de Interfaz. •Actualizar arquitectura. •Implementar funcionalidad. •Probar registro de datos. •Implementar backend restante de la funcionalidad.	5	40
Sprint 4	Registro Pensionados	•Modelar el Análisis. prototipo de Interfaz. •Actualizar arquitectura. •Implementar funcionalidad. •Probar registro de datos. •Implementar backend restante de la funcionalidad.	8	64
Sprint 5	Repostes de cobros y pensionados	•Modelar el Análisis. Prototipo de Interfaz. •Actualizar arquitectura. •Implementar funcionalidad. •Probar registro de datos. •Implementar backend restante de la funcionalidad.	30	240
Sprints 6	Generar salida de datos para el cliente.	•Modelar el Análisis. Prototipo de Interfaz para el cliente. •Actualizar arquitectura. •Implementar funcionalidad. •Probar navegabilidad de datos. •Implementar frontend restante de la funcionalidad.	27	216

3.4. Matriz de requerimientos

Proceso de negocio	Actividad de negocio	Actor del negocio	Cod refe función	Requerimiento funcional	Código caso de uso	Caso de uso	Actor de la aplicación
Registro	Registrar personal	Administrador de la app	P01	La app deberá registrar personal que manejará los datos de la aplicación móvil	C01	Personal	Trabajador
	Registro de platos	Administrador	P02	La app permitirá registrar los nombres de los platos que permitirá hacer la programación del menú	C02	platos	Trabajador
	Registro de pensionados	Administrador	P03	La app permitirá registrar a los pensionados	C04	Pensionados	Trabajador
Cobros	Registra de cobros	Administrador	P04	La app permitirá realizar el cobro del mes a los pensionados.	C05	administrativo	Trabajador
	Reporte de	Administrador	P05	La aplicación móvil	C06	Reportes inscritos	Trabajador

	pensionados			permite realizar el reporte de inscritos en la pensión			
Reporte de coros	Registro de reporte de cobros a pensionados	Administrador	P06	La aplicación permitirá generar el reporte de cobros del mes de los pensionados	C07	Administrador	Trabajador
Programación menú	Registro de la programación del menú del día y la semana	Administrador	P07	La aplicación permitirá realizar la el registro de la programación del menú del día y de toda la semana	C08	Administrador	Trabajador
Menú de la semana	Menú de la semana	Cliente	P08	El cliente podrá ingresar a la aplicación y ver el menú del restaurante del día y de toda la semana	C09	Menú general	Cliente

3.5. Diagrama de actividades rol administrador

Diagrama de actividad caso de uso platos

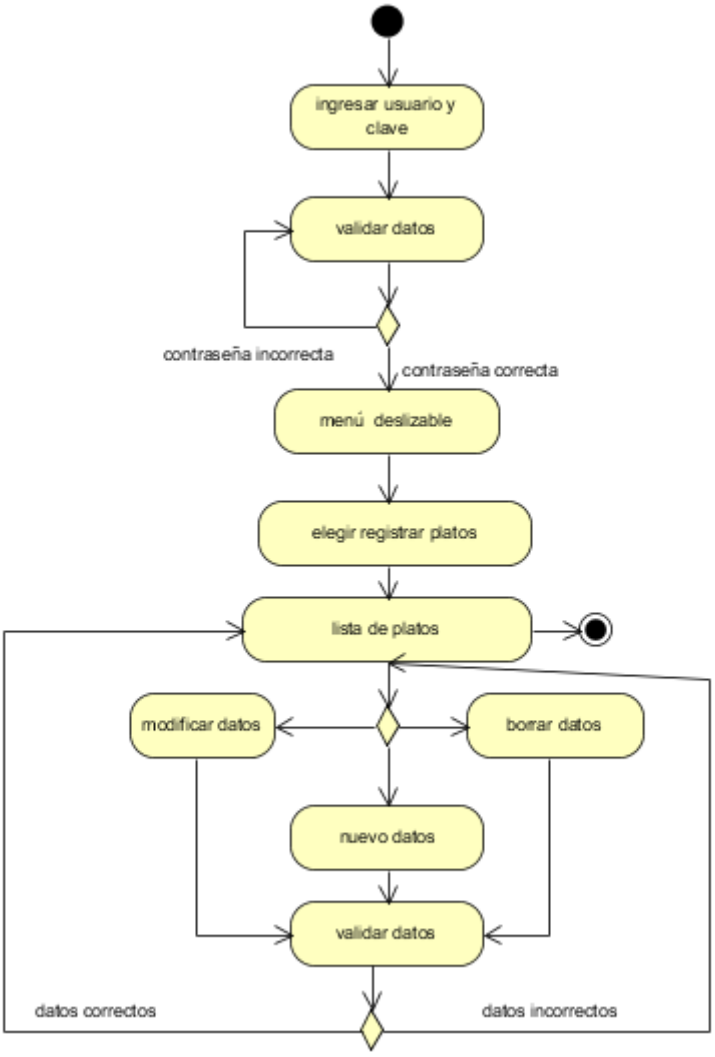


Diagrama de actividad caso de uso pensionado

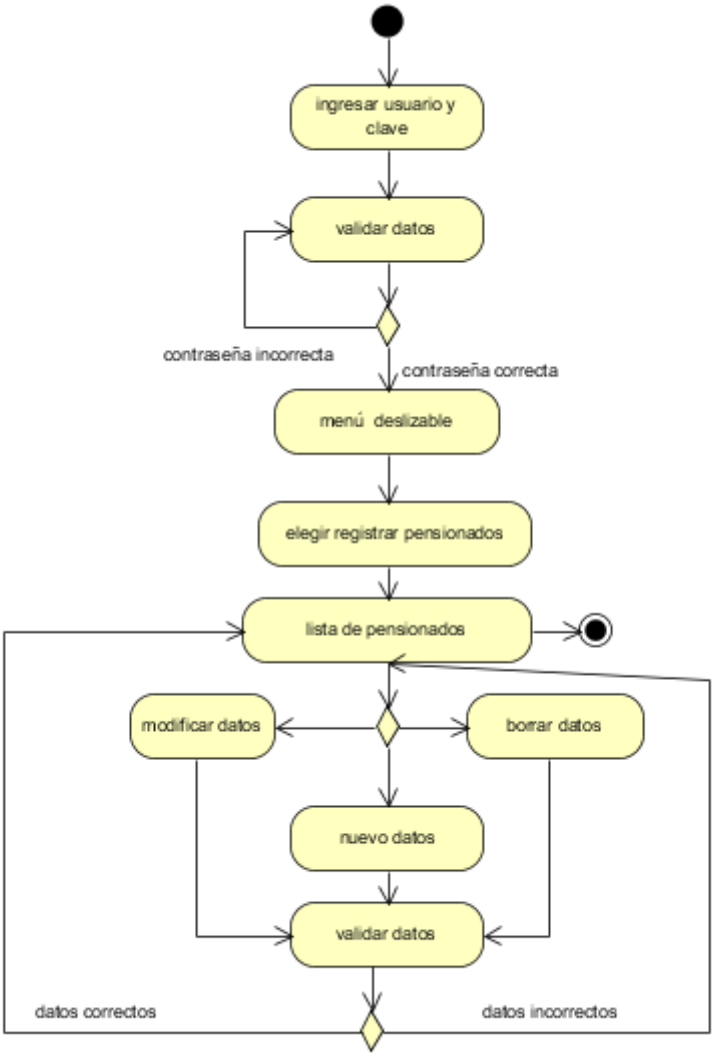


Diagrama de actividad caso de cobros

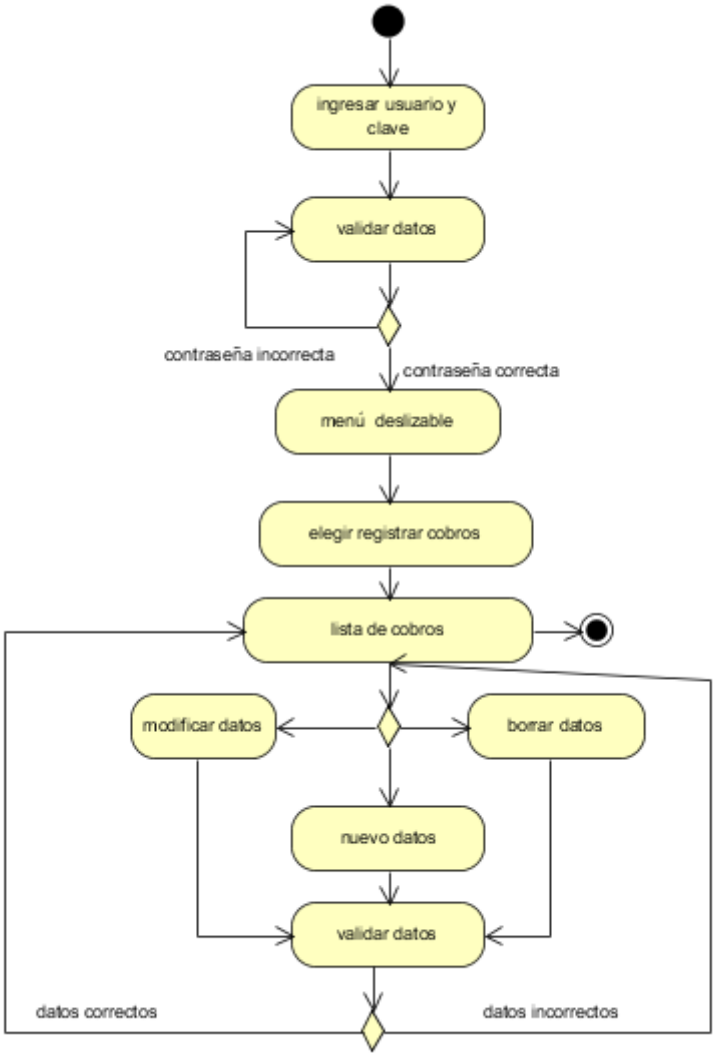
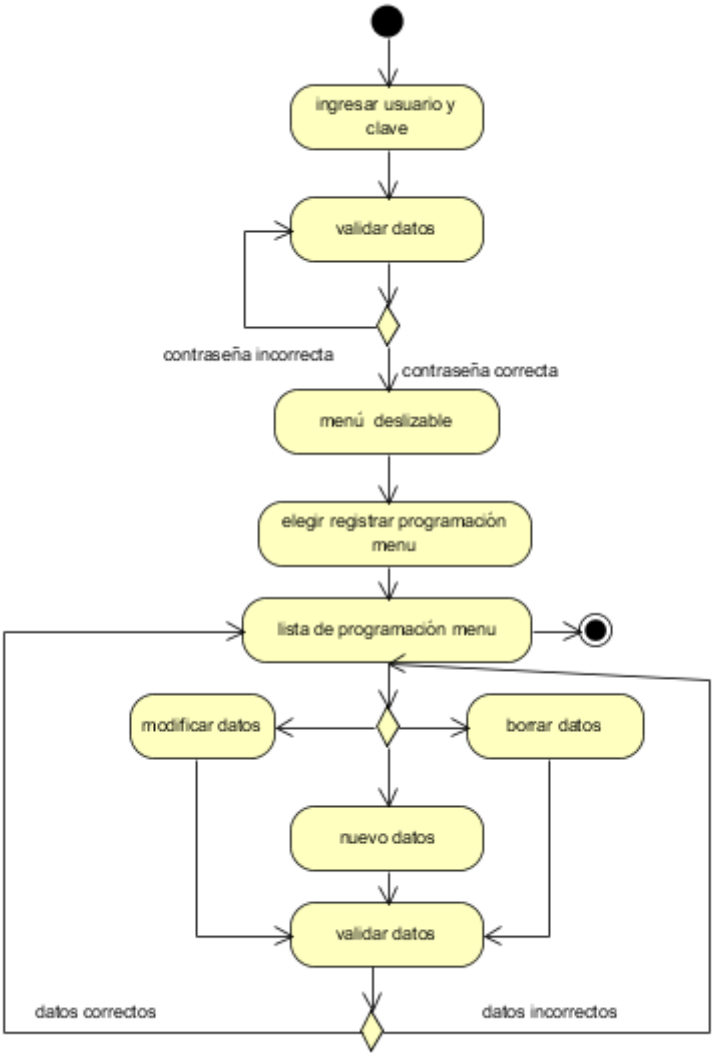
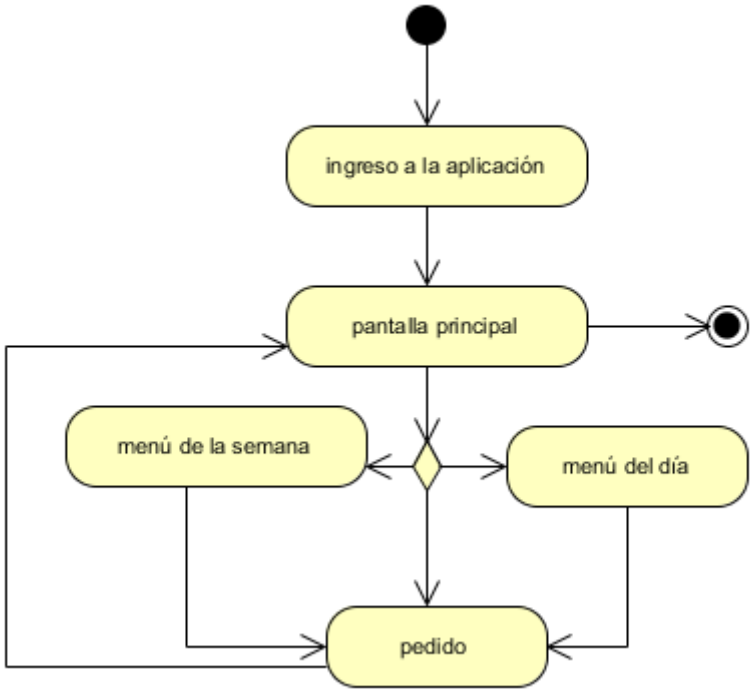


Diagrama de actividad caso de programación



3.6. Diagrama de actividad rol cliente

Diagrama de actividad caso de uso pedido



3.7. Diagrama de secuencia para el rol administrador

Diagrama de secuencia para el caso de uso platos

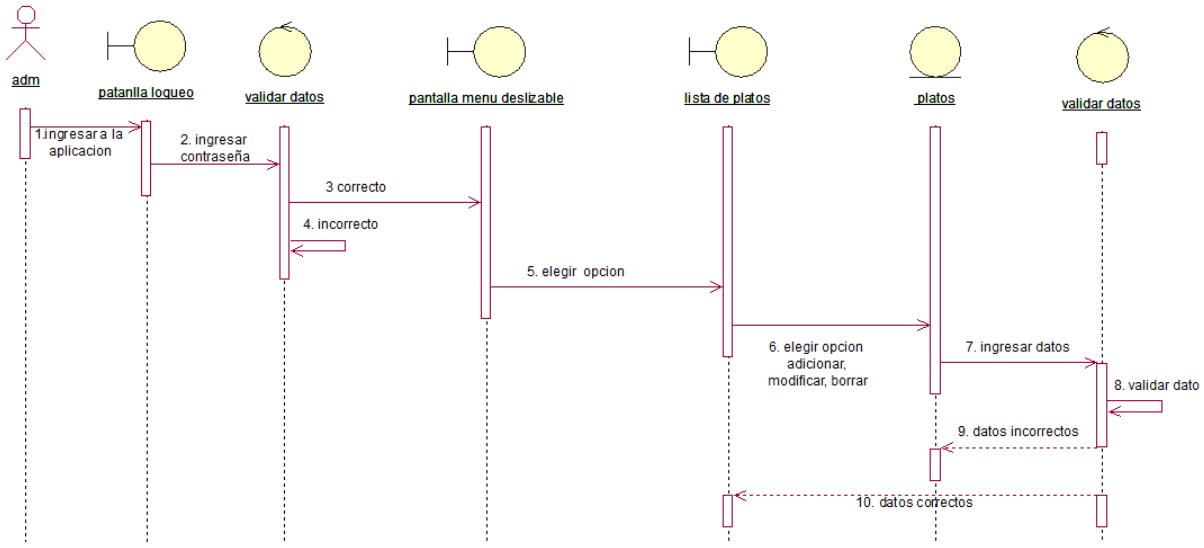


Diagrama de secuencia para el caso de uso pensionados

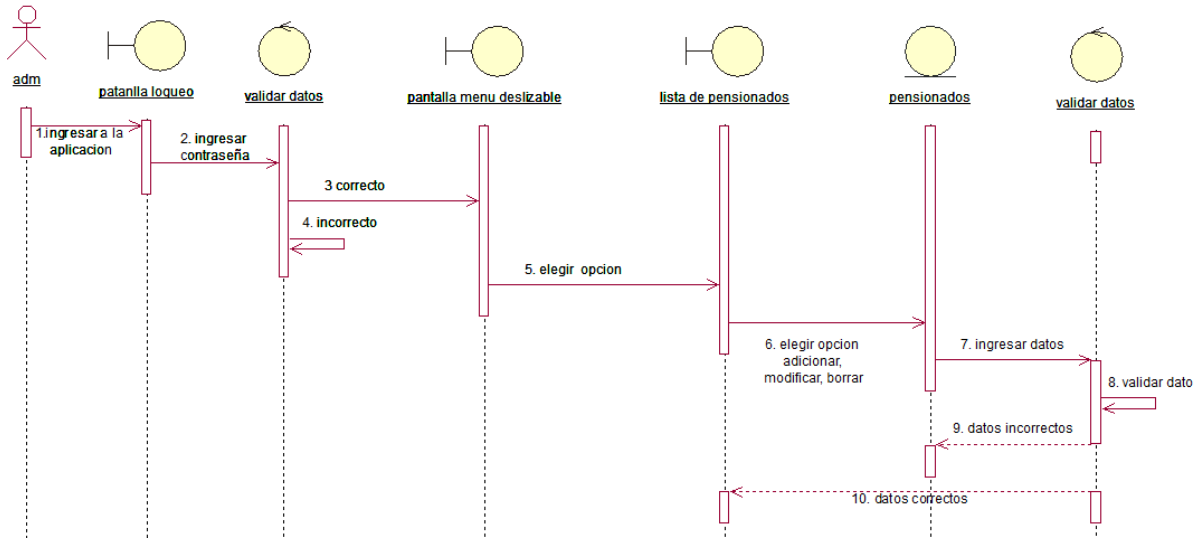


Diagrama de secuencia para el caso de uso cobros

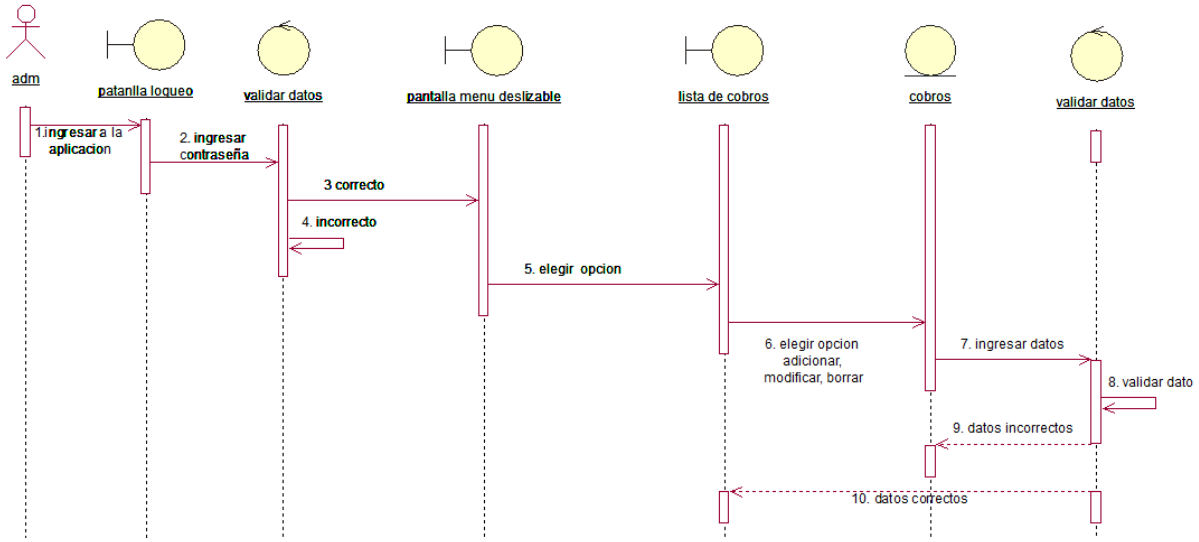
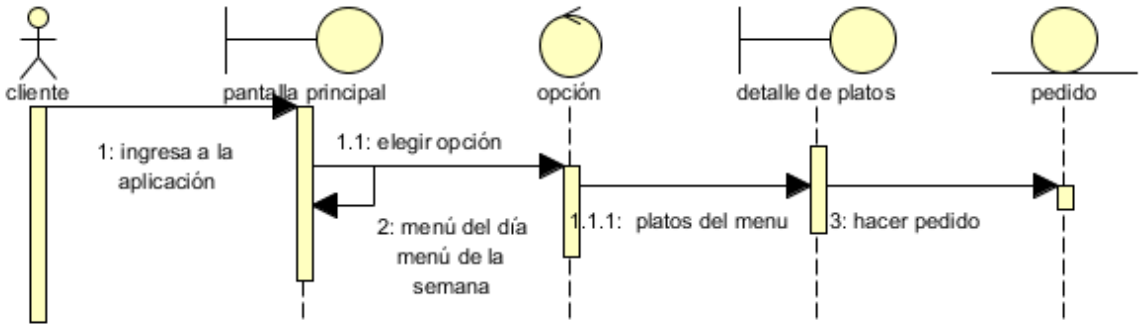


Diagrama de secuencia para el caso de uso programación menú

3.8. Diagrama de secuencia rol cliente

Diagrama de secuencia caso de uso pedido



3.9. Diagrama de colaboración rol administrador

Diagrama de colaboración caso de uso platos

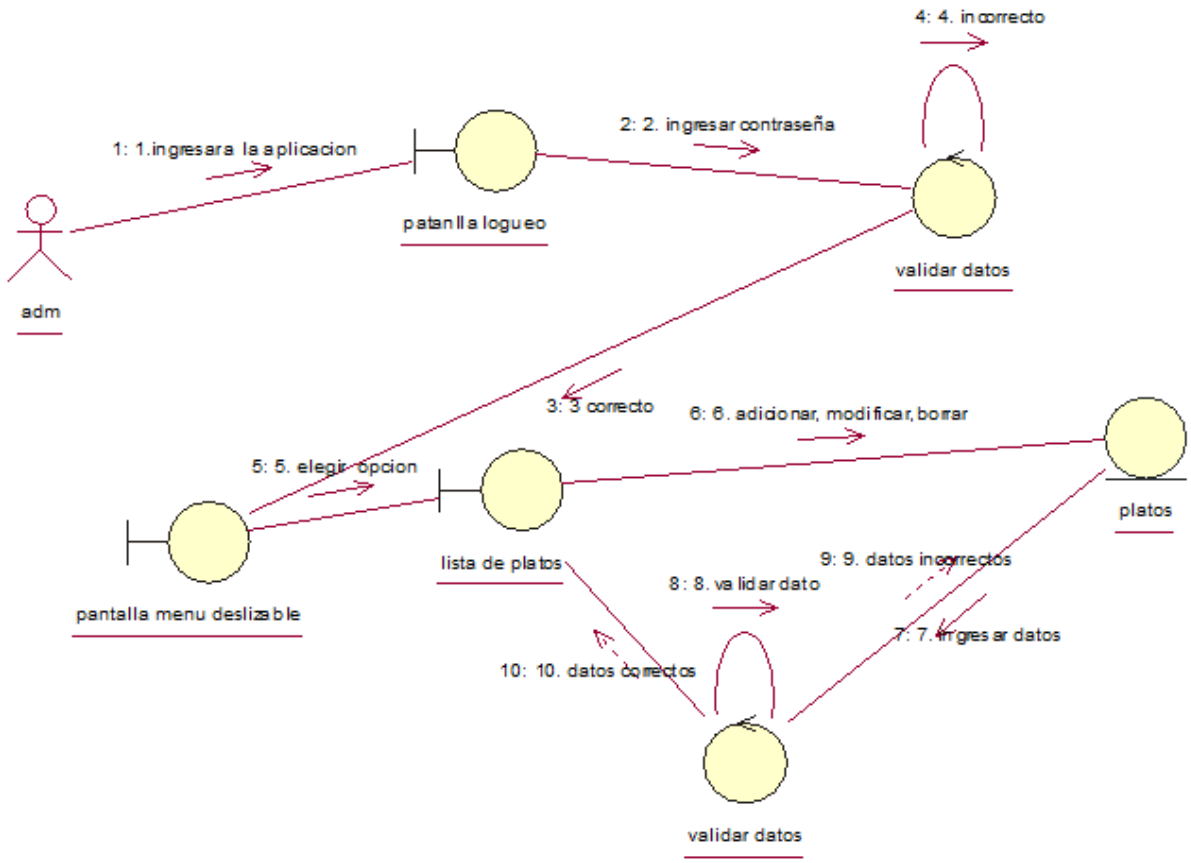


Diagrama de colaboración caso de uso cobros

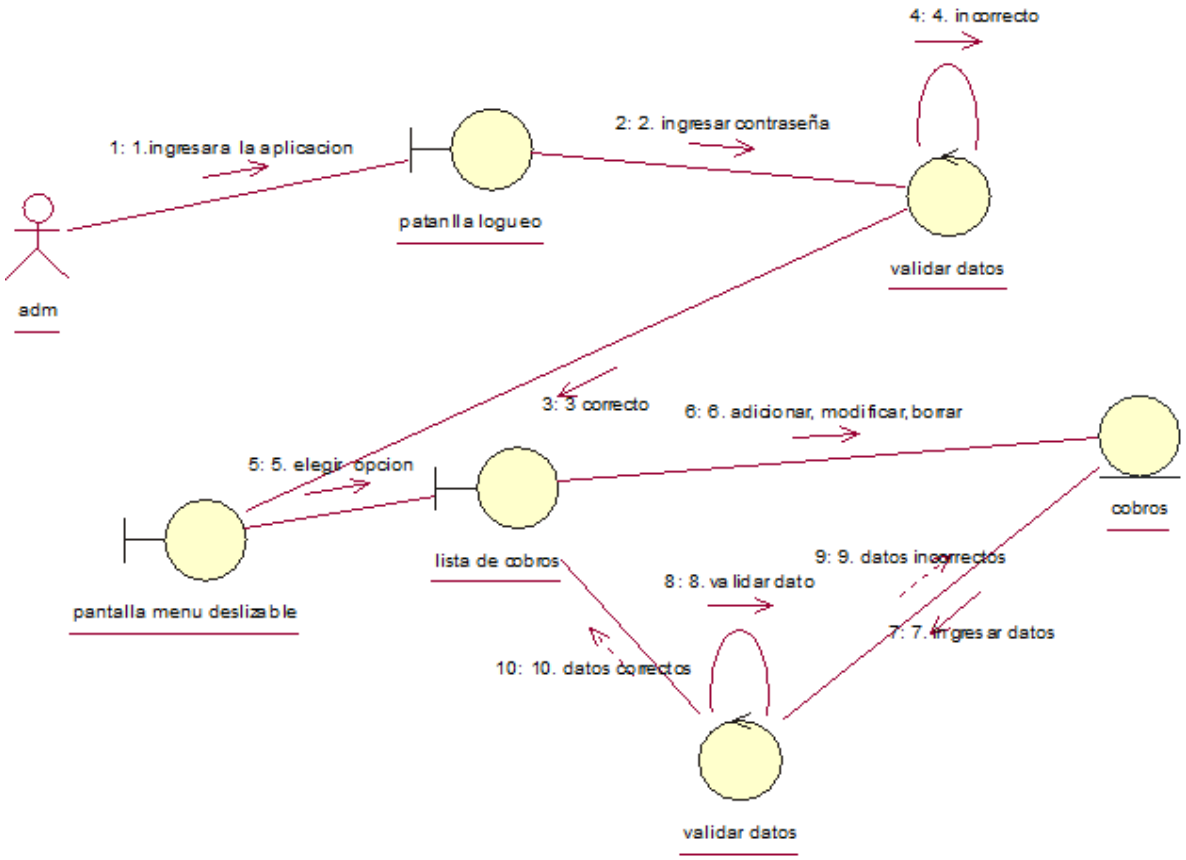


Diagrama de colaboración caso de uso Pensionado

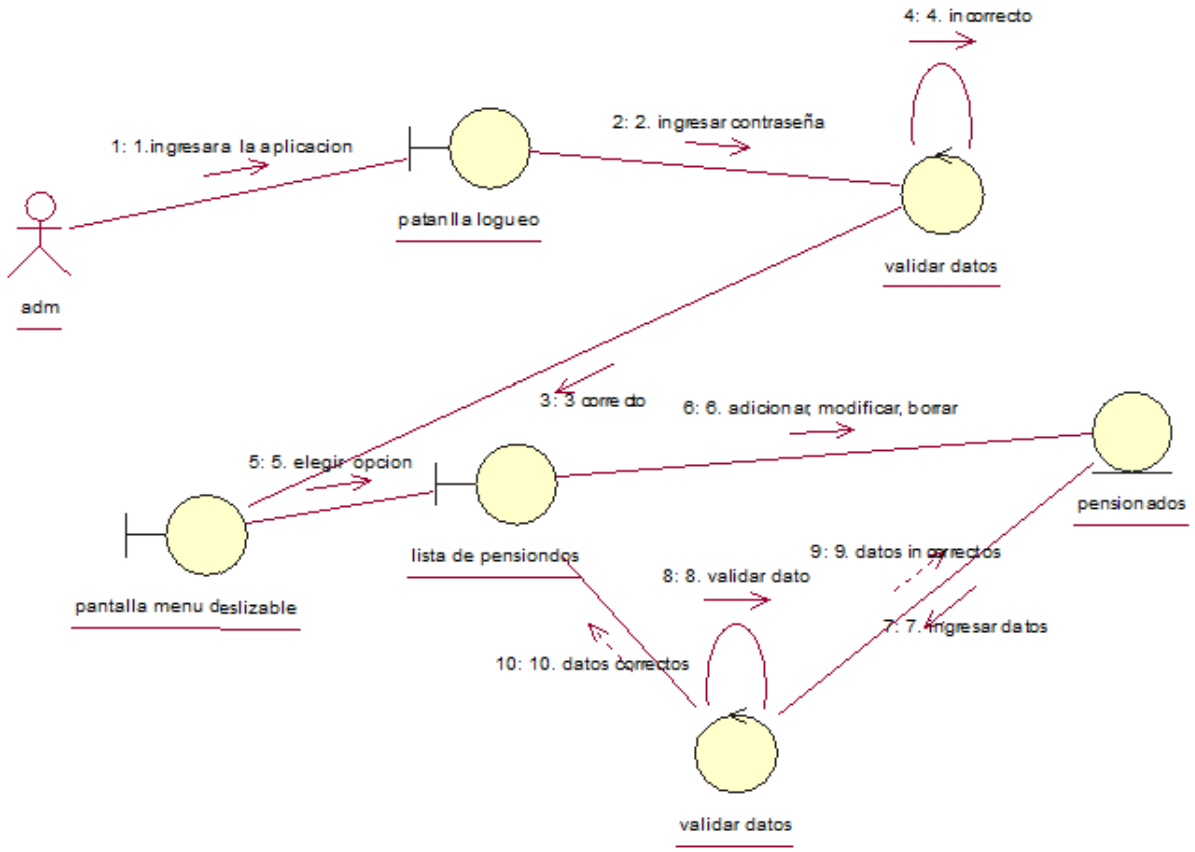
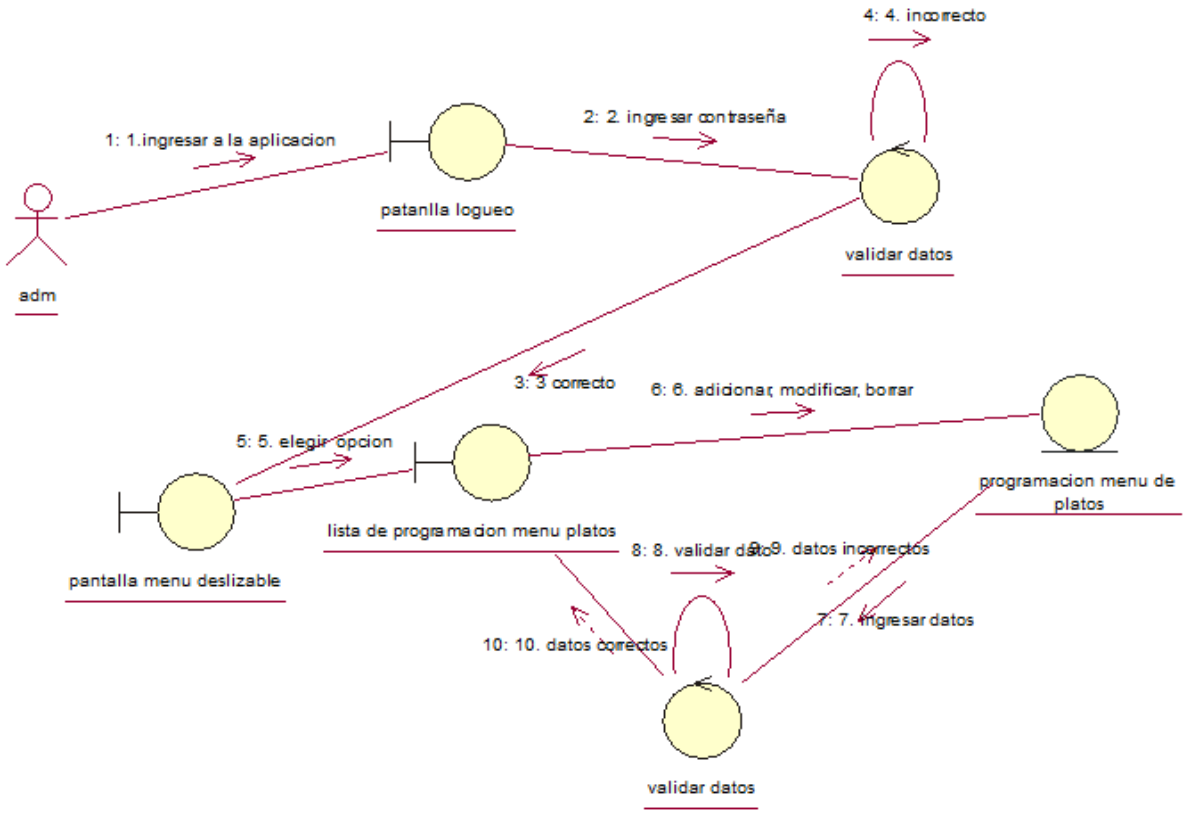
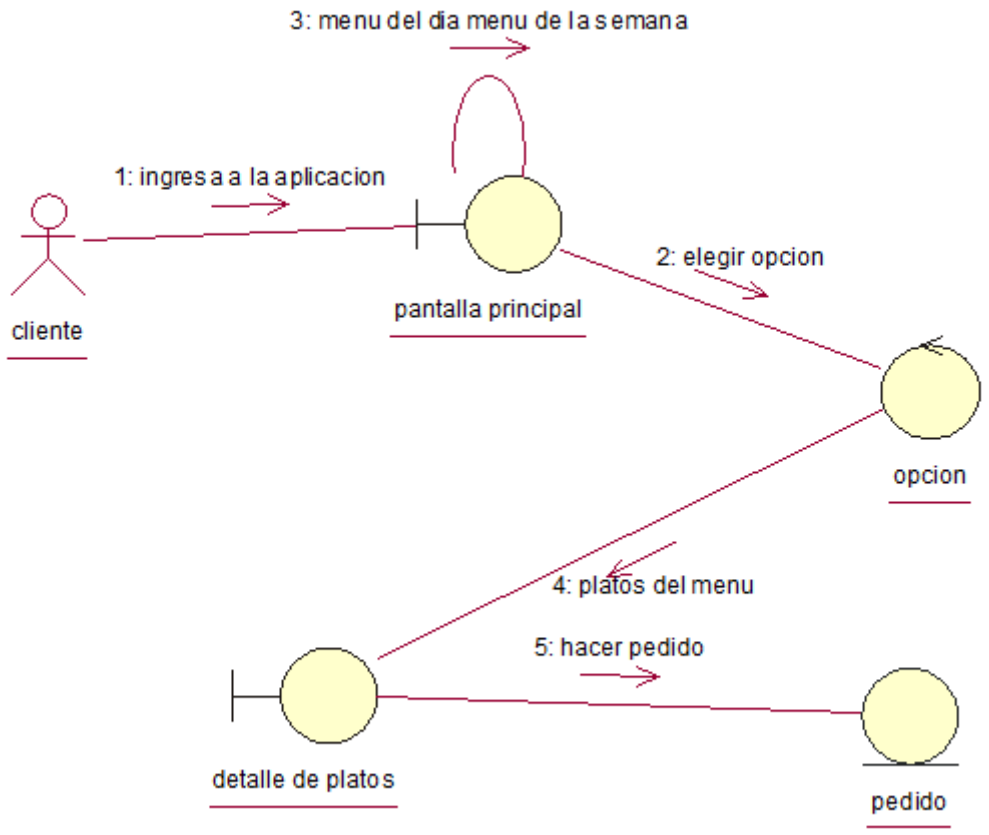


Diagrama de colaboración caso de uso programación de menú



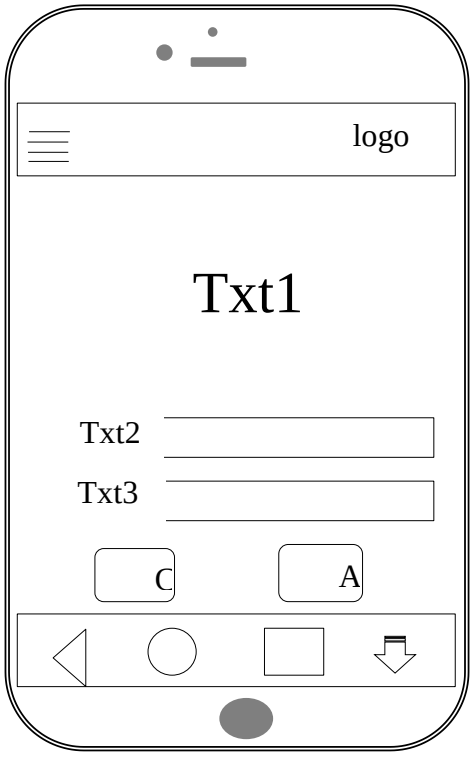
3.10. Diagrama de colaboración rol cliente

Diagrama de colaboración caso de uso pedido

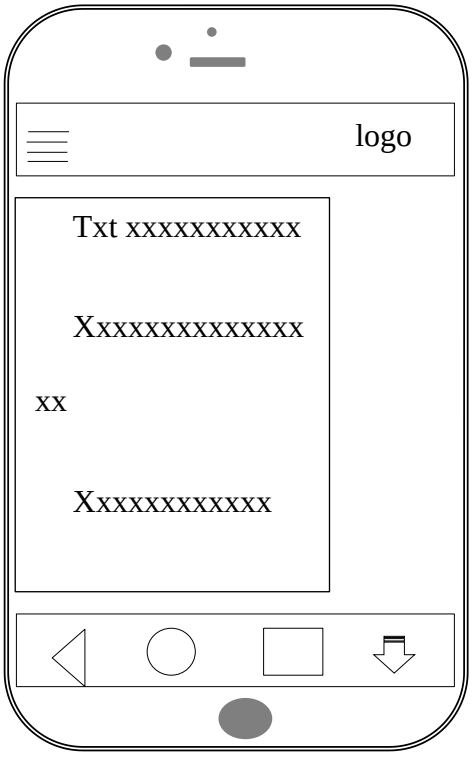


3.11. Estructura de Pantalla

Estructura del logueo

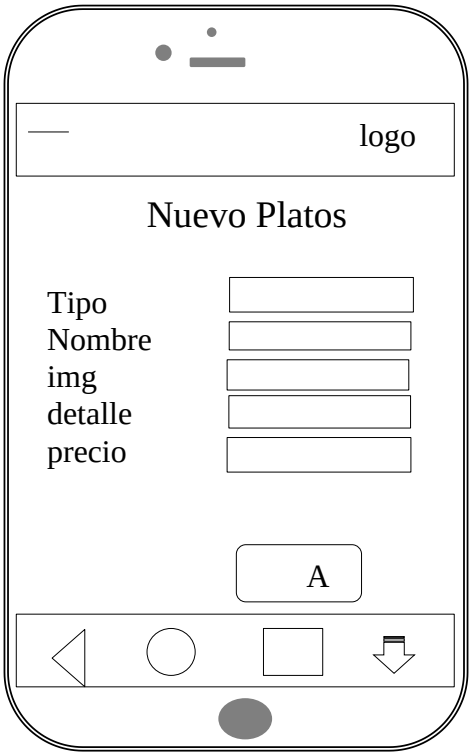


Pantalla menú general

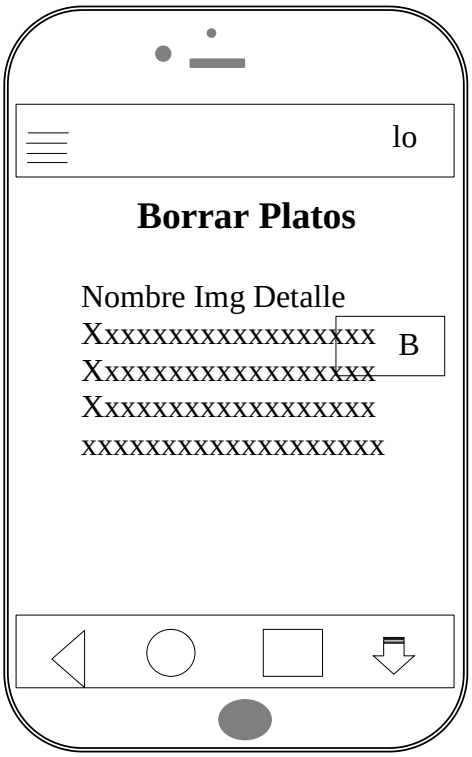


Pantalla listar platos

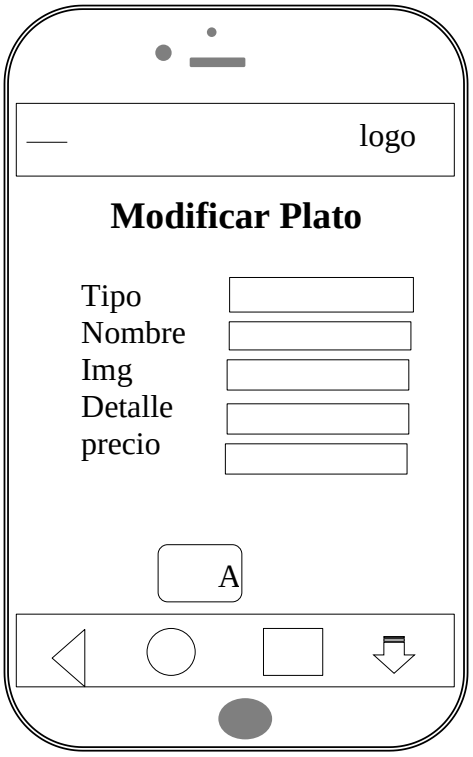
Pantalla Registrar Nuevo Platos



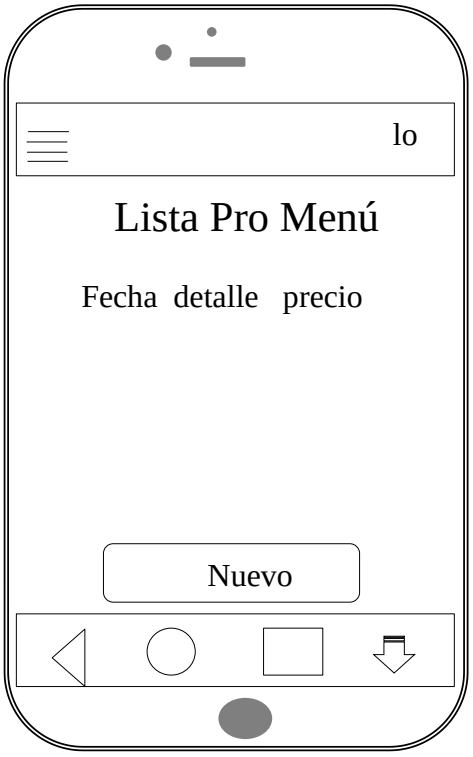
Pantalla Borrar platos



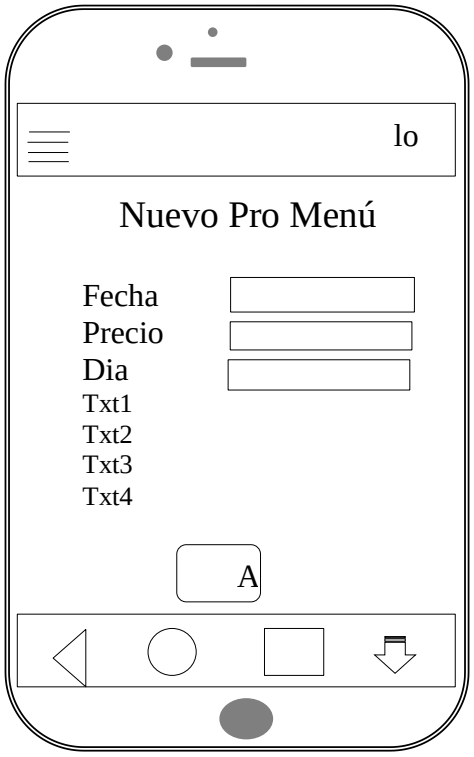
Pantalla Modificar Plato



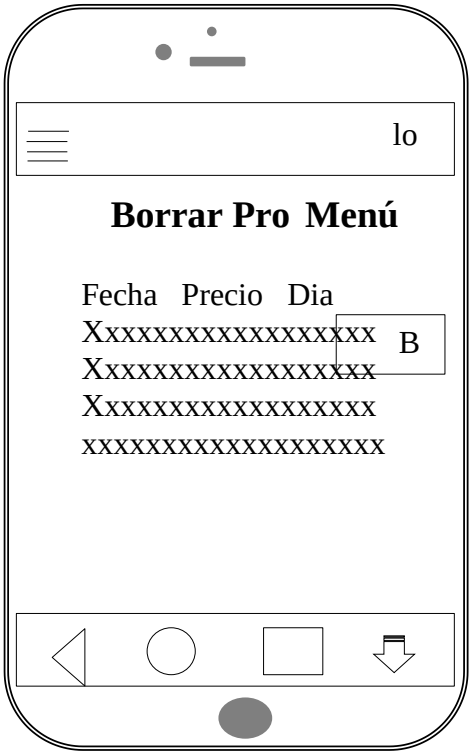
Pantalla programar menú



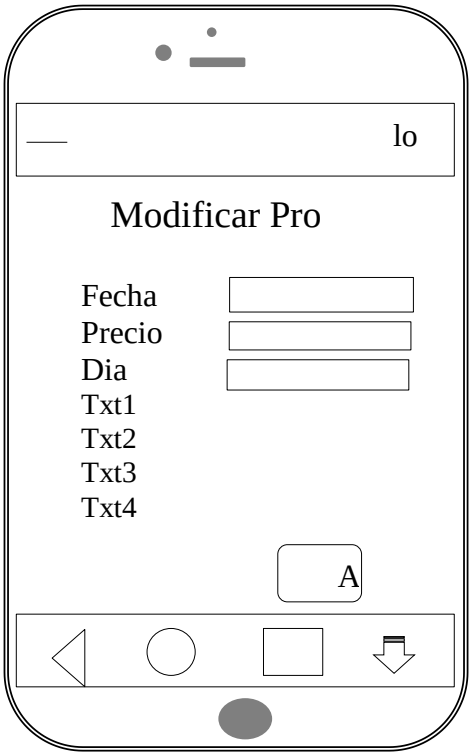
Pantalla nueva programación de menú



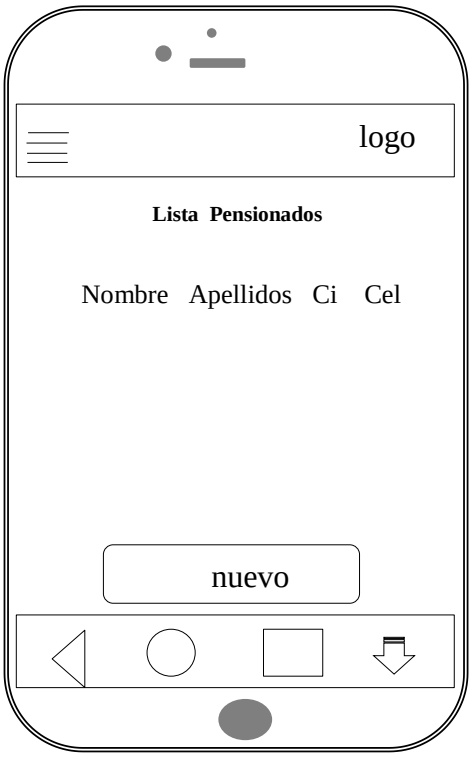
Pantalla Borrar Programación Menú



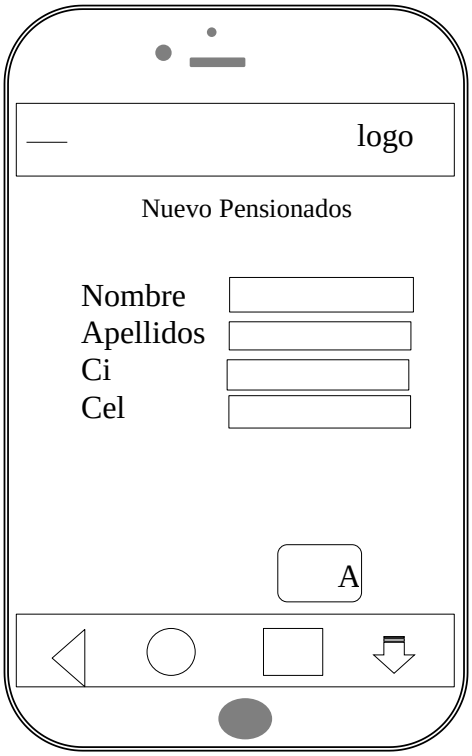
Pantalla Modificar Programación Menú



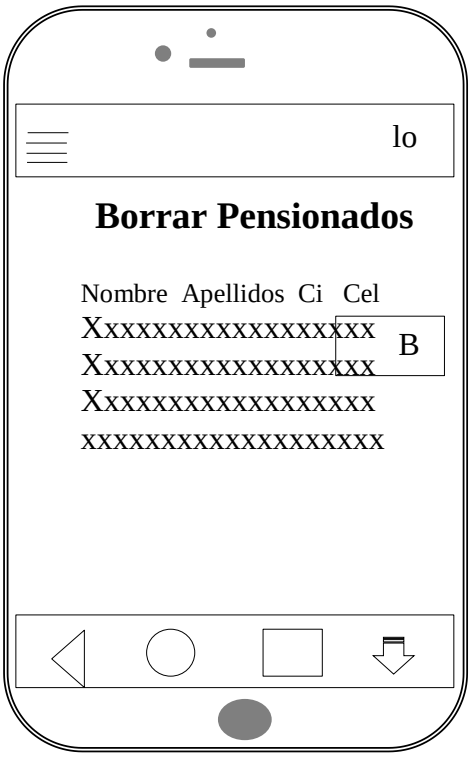
Pantalla lista pensionados



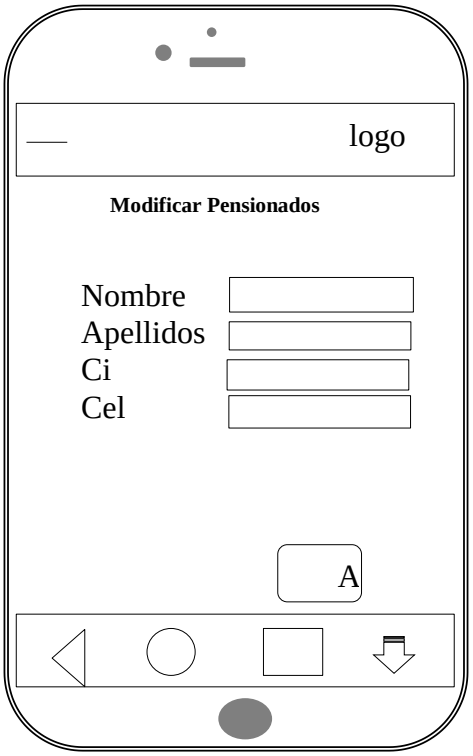
Pantalla Registro Nuevo Pensionados



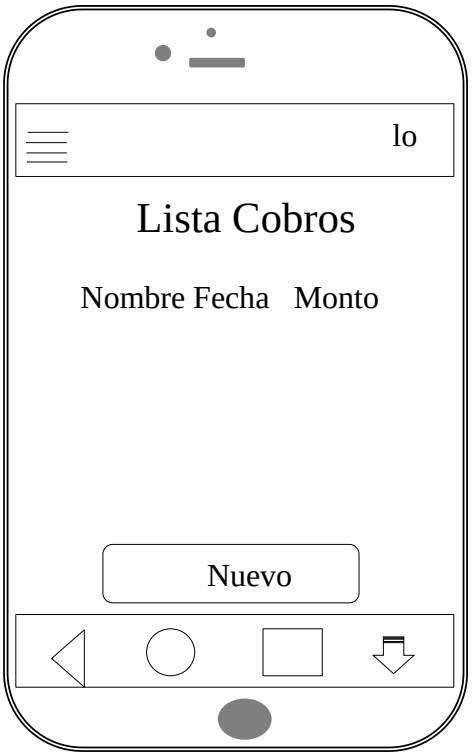
Pantalla Borrar Pensionados



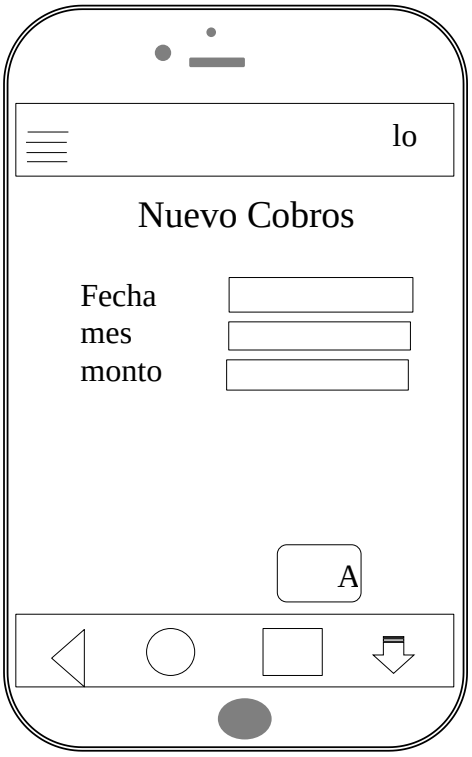
Pantalla Modificar Pensionados



Pantalla Listar Cobros

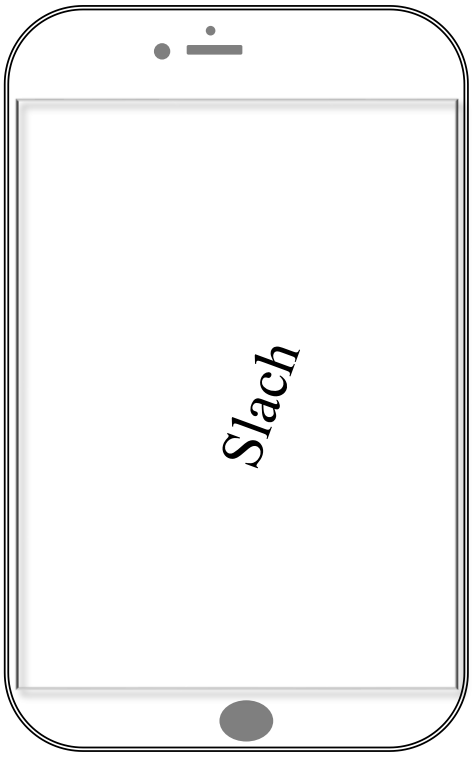


Pantalla Registrar Nuevos Cobros

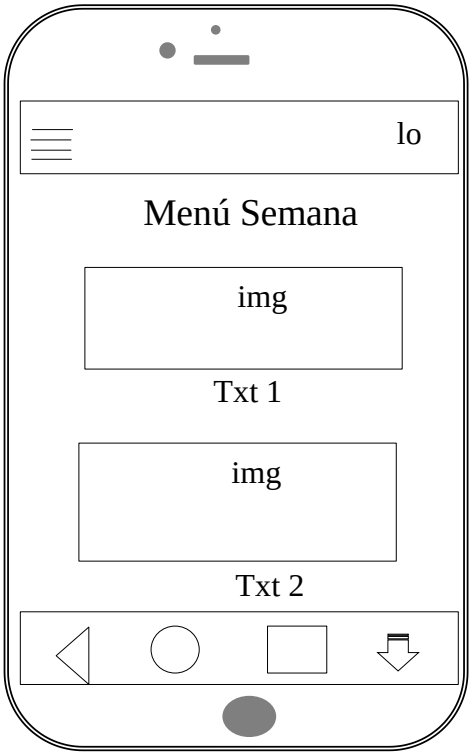


Pantallas de salida para el cliente

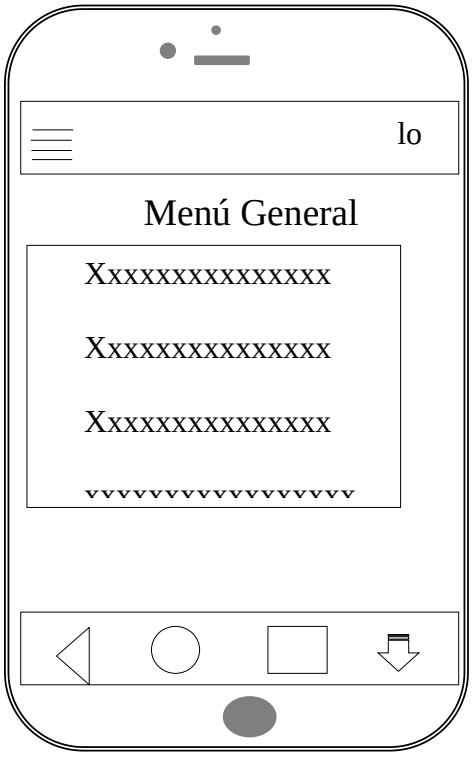
Pantalla de Introducción



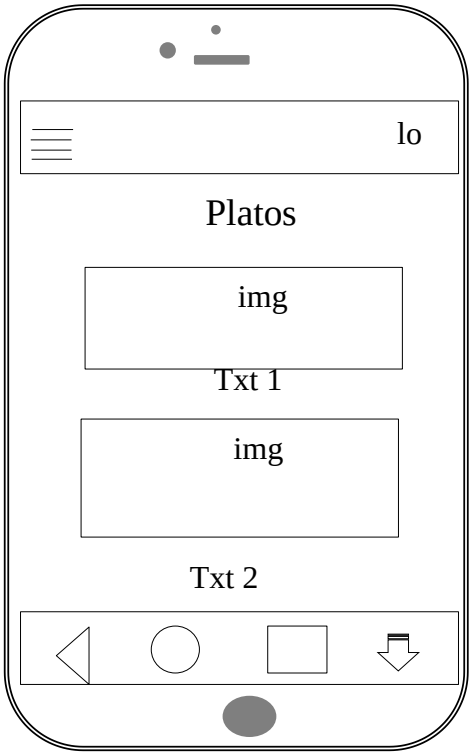
Pantalla principal



Pantalla General deslizable



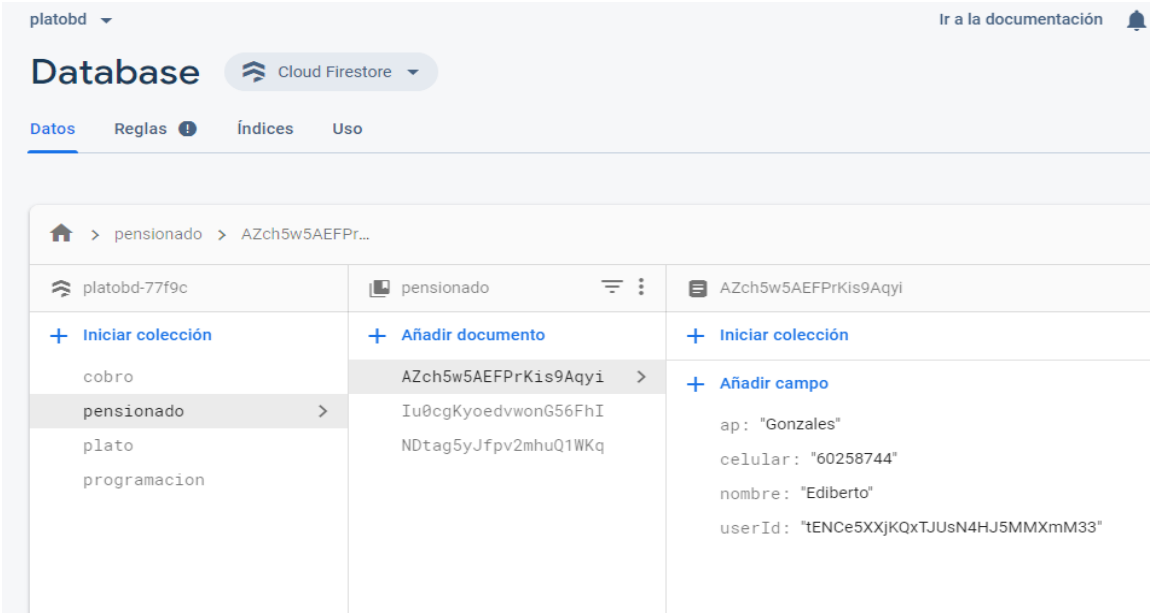
Pantalla de Platos



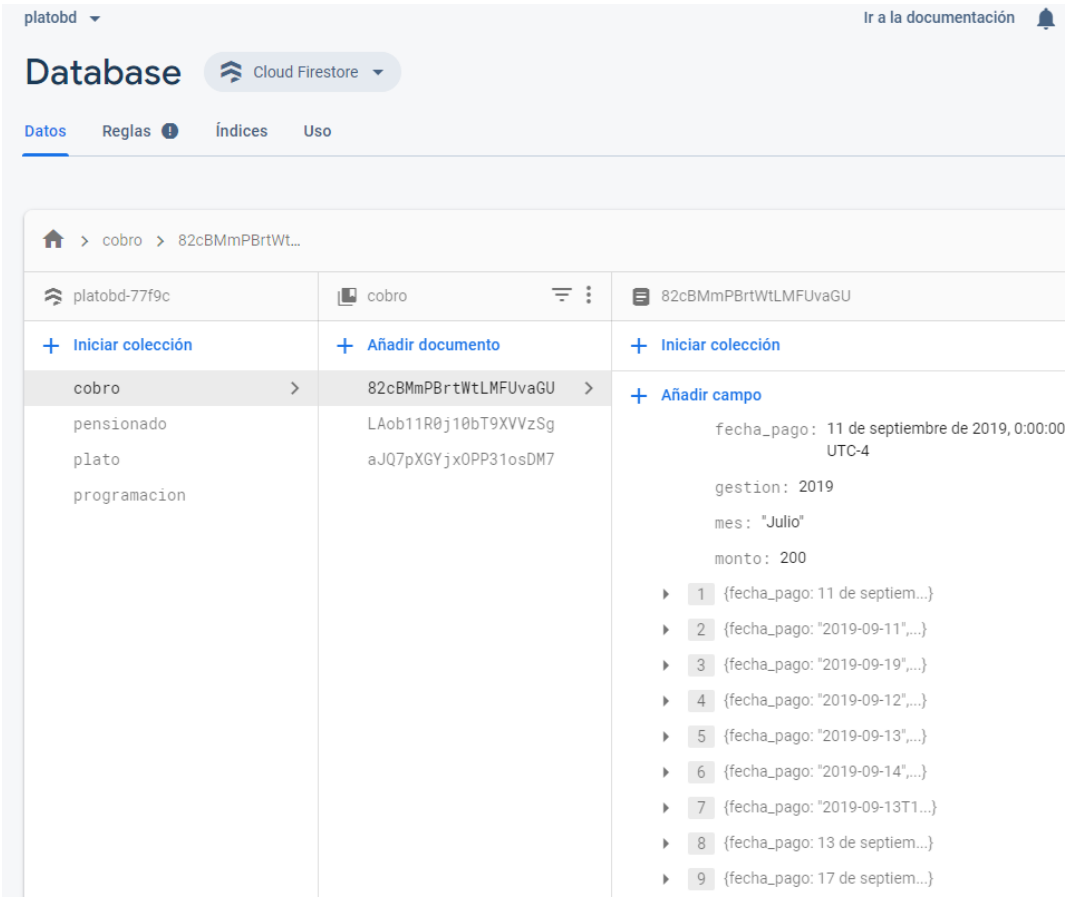
4. DISEÑO

4.1. BASE DE DATOS EN FIREBASE

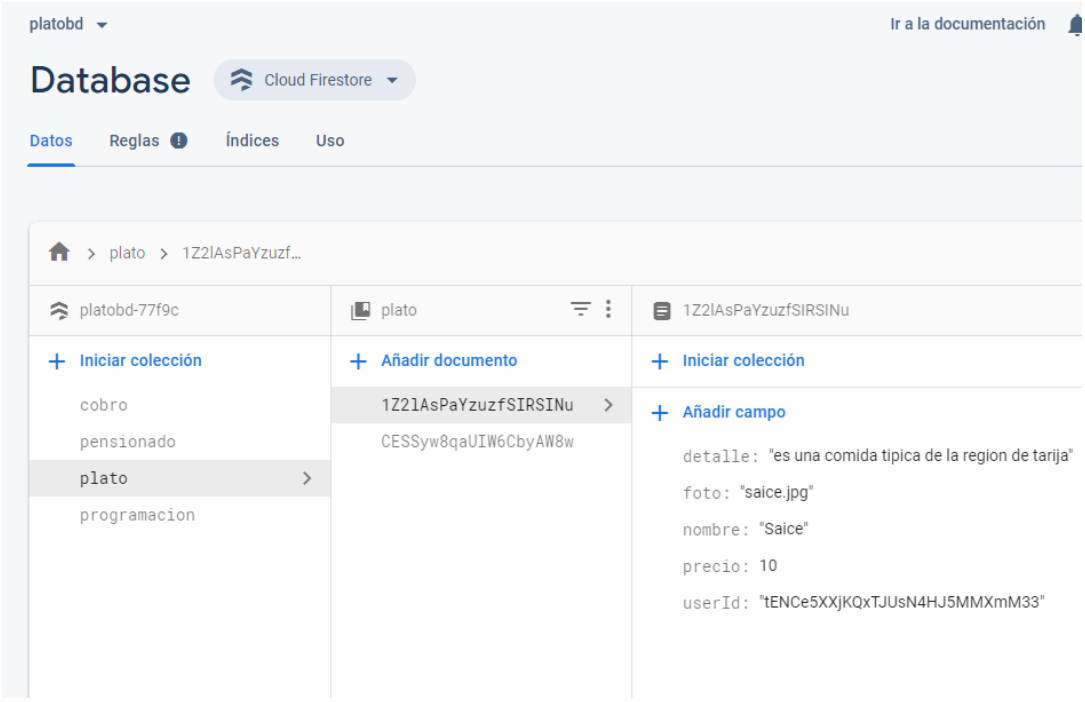
4.1.1. Base de datos tabla pensionado



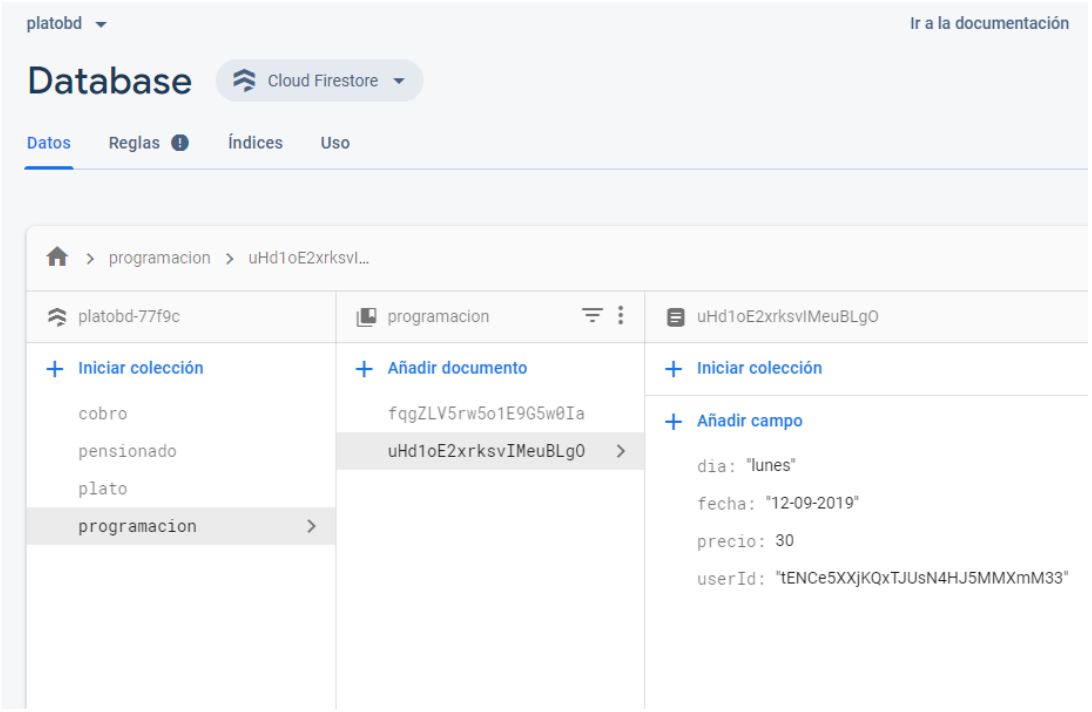
4.1.2. Base de datos tabla cobros



4.1.3. Base de datos tabla plato



4.1.4. Base de datos tabla programación



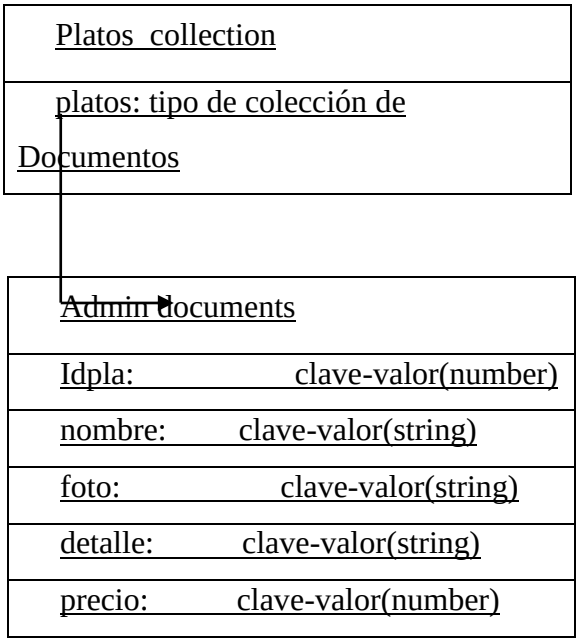
Creación de tablas para la nube

```
export interface Platos {
    idpla?: number;
    nombre?: string;
    foto?: string;
    detalle?:string;
    precio?: number;
    }
export interface Programacion {
    idpro?: number;
    fecha?: string;
    precio?: number;
    dia?:string;
    }
```

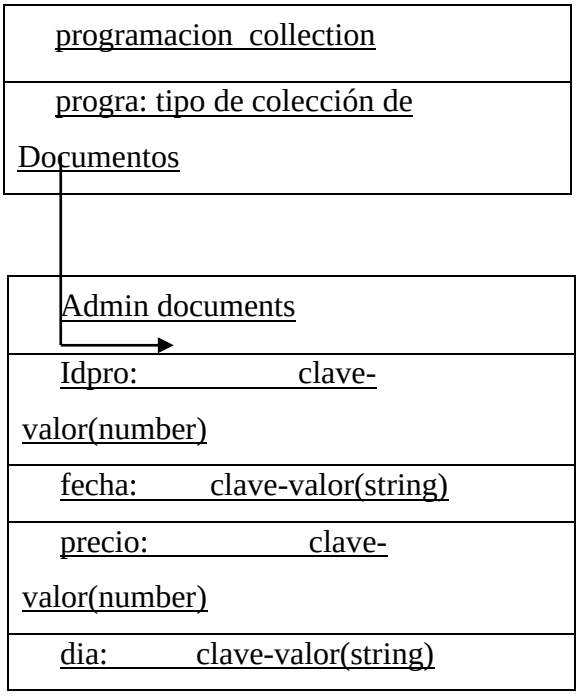
```
export interface Pensionados {

   idpen?: number;
   nombre?: string;
   apellidos?: string;
   ci?:number;
   celular?: number;
   }
export interface Cobros {
   idco?: number;
   fecha?: string;
   monto?: number;
      }
```

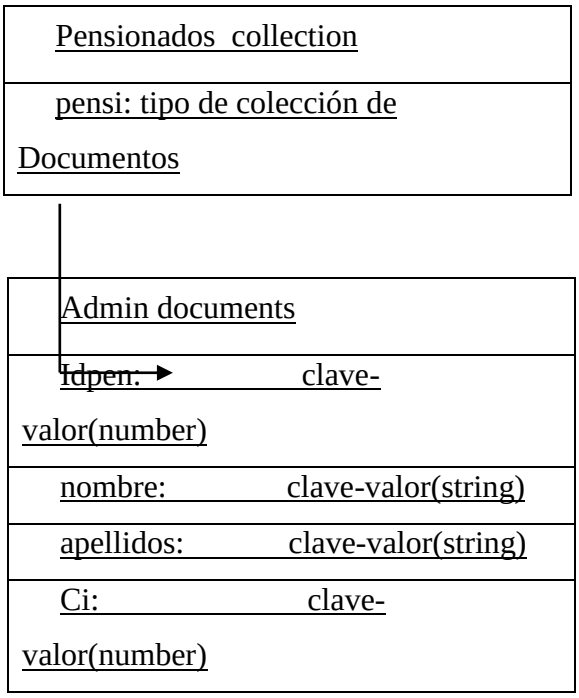
Diccionario de datos para la tabla platos



Diccionario de datos de la tabla programación



Diccionario de datos para la tabla pensionados

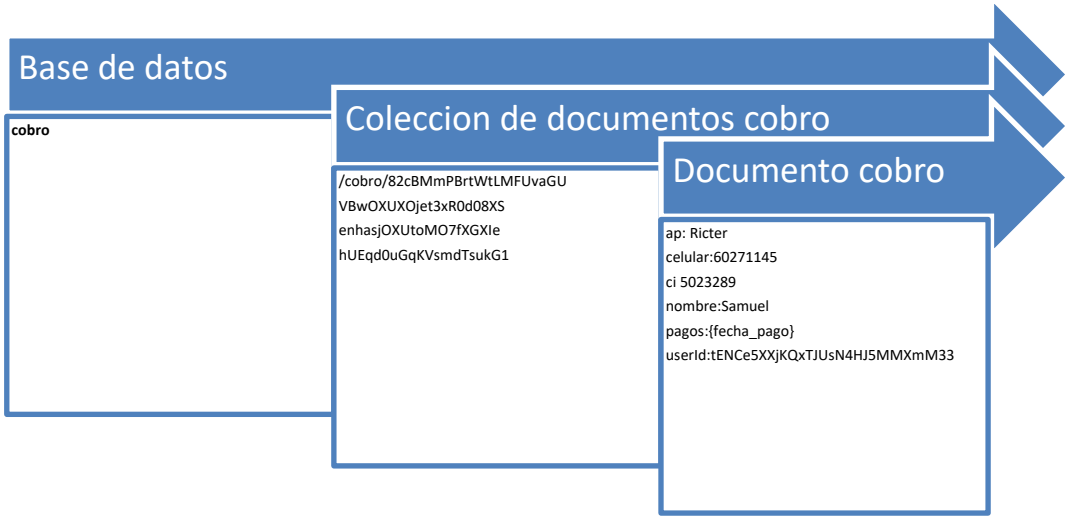


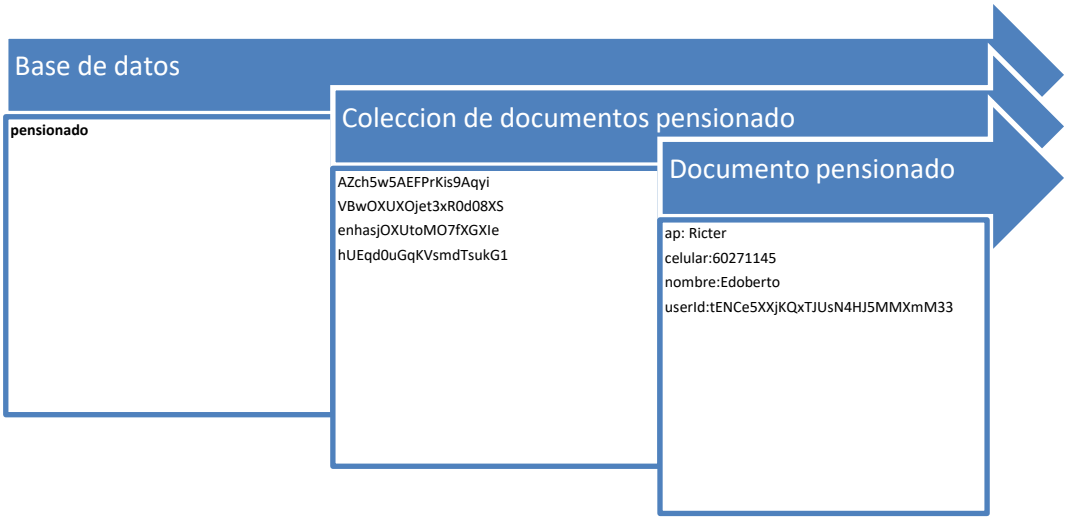
Celular: clave-valor(number)

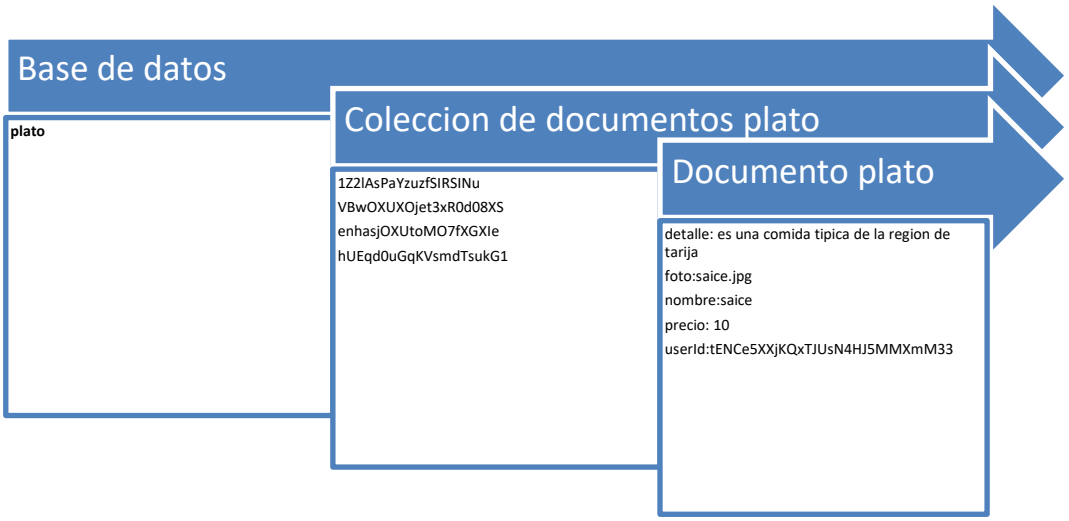
Diccionario de datos para la tabla cobros

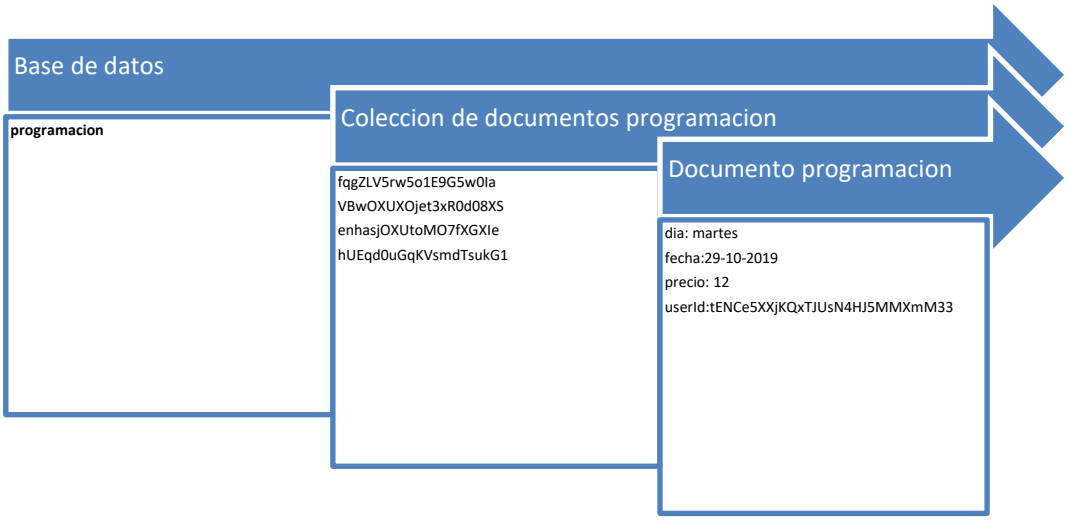
Cobros collection
cobro: tipo de colección de Documentos

Admin documents
Idco: clave-valor(number)
fecha: clave-valor(string)
monto: clave-valor(number)



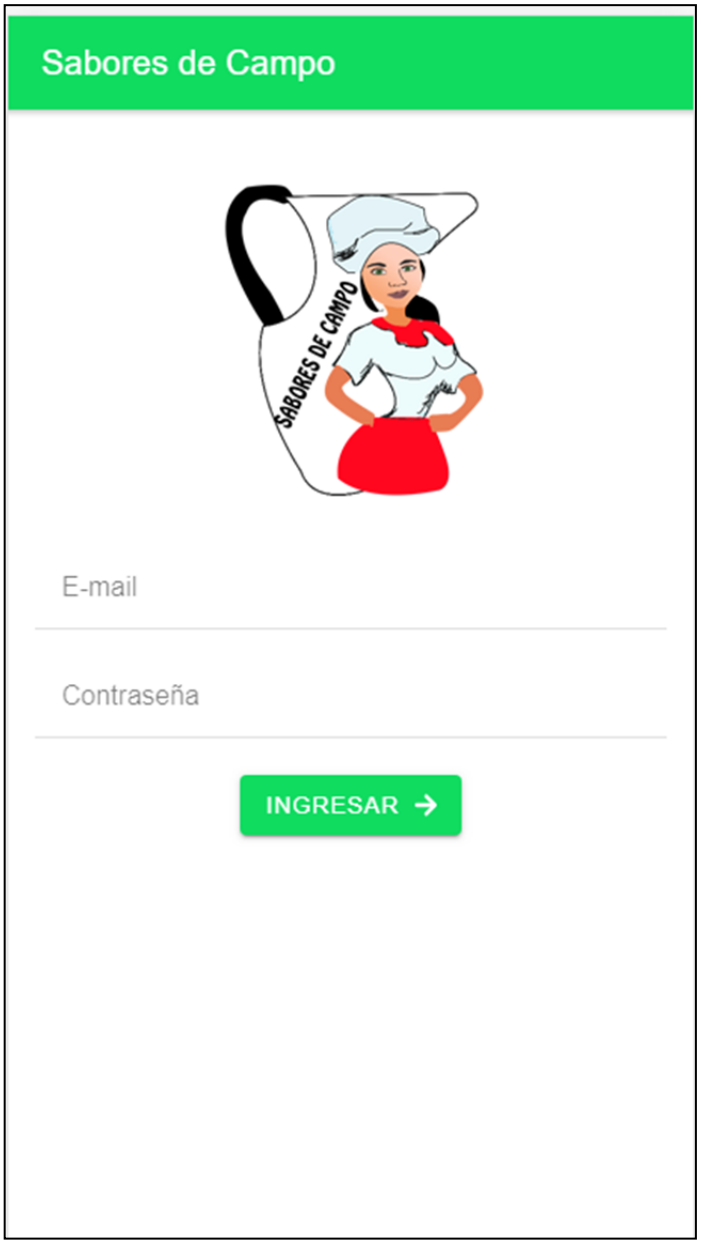

Base de datos
pensionado
Coleccion de documentos pensionado
AZch5w5AEFPrKis9Aqyi
VBwOXUXOjet3xR0d08XS
enhasjOXUtoMO7fXGXIe
hUEqd0uGqKVsmdTsukG1
Documento pensionado
ap: Ricter
celular:60271145
nombre:Edoberto
userId:tENCe5XXjKQxTJUsN4HJ5MMXmM33


Base de datos
plato
Coleccion de documentos plato
1Z2lAsPaYzuzfSIRSINu
VBwOXUXOjet3xR0d08XS
enhasjOXUtoMO7fXGXIe
hUEqd0uGqKVsmdTsukG1
Documento plato
detalle: es una comida tipica de la region de tarija
foto:saice.jpg
nombre:saice
precio: 10
userId:tENCe5XXjKQxTJUsN4HJ5MMXmM33


Base de datos
programacion
Coleccion de documentos programacion
fqgZLV5rw5o1E9G5w0Ia
VBwOXUXOjet3xR0d08XS
enhasjOXUtoMO7fXGXIe
hUEqd0uGqKVsmdTsukG1
Documento programacion
dia: martes
fecha:29-10-2019
precio: 12
userId:tENCe5XXjKQxTJUsN4HJ5MMXmM33

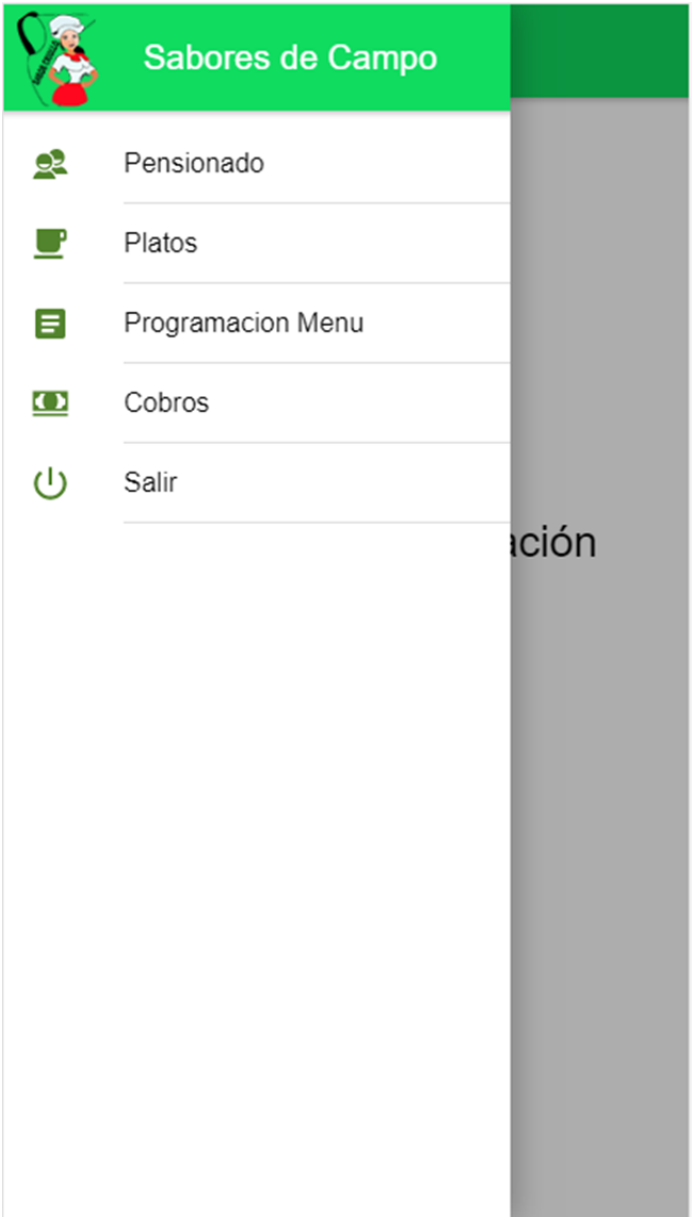
4.2. Pantallas de la aplicación móvil

Pantalla 1: Logueo



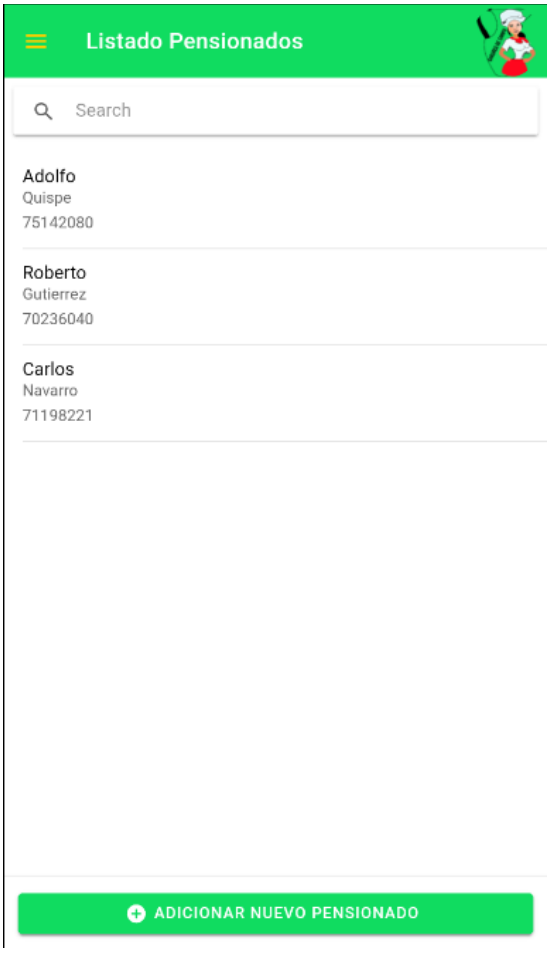
En esta pantalla permitirá ingresar sus respectivas contraseñas para poder ingresar a la aplicación móvil.

Pantalla 2: Menú principal



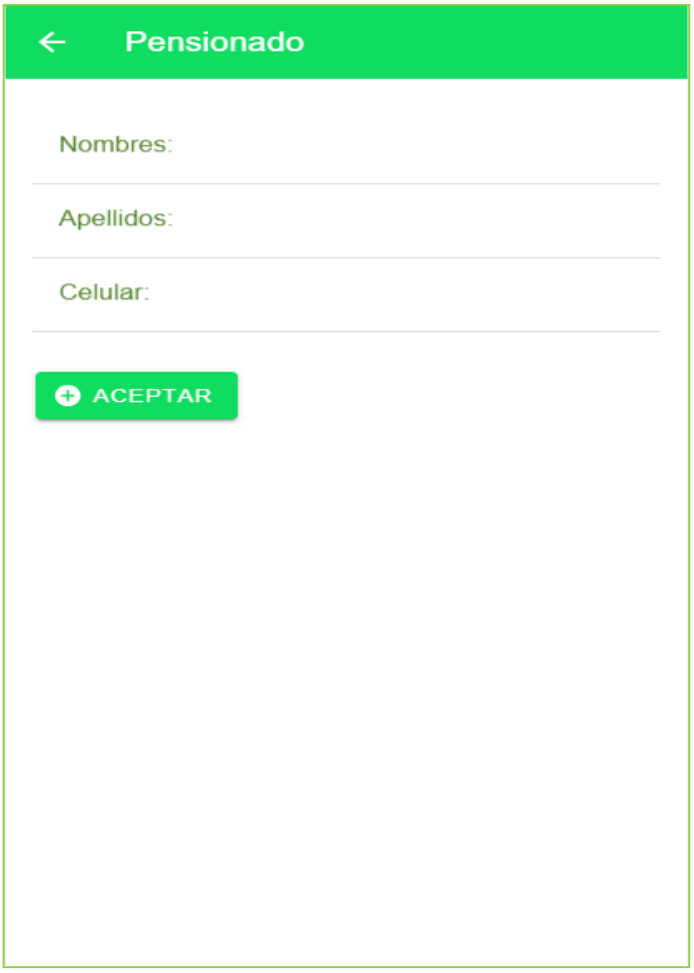
Esta pantalla visualiza el menú general de la aplicación móvil haciendo click en el menú permitirá ingresar cualquier de las opciones.

Pantalla 3: Lista de pensionados



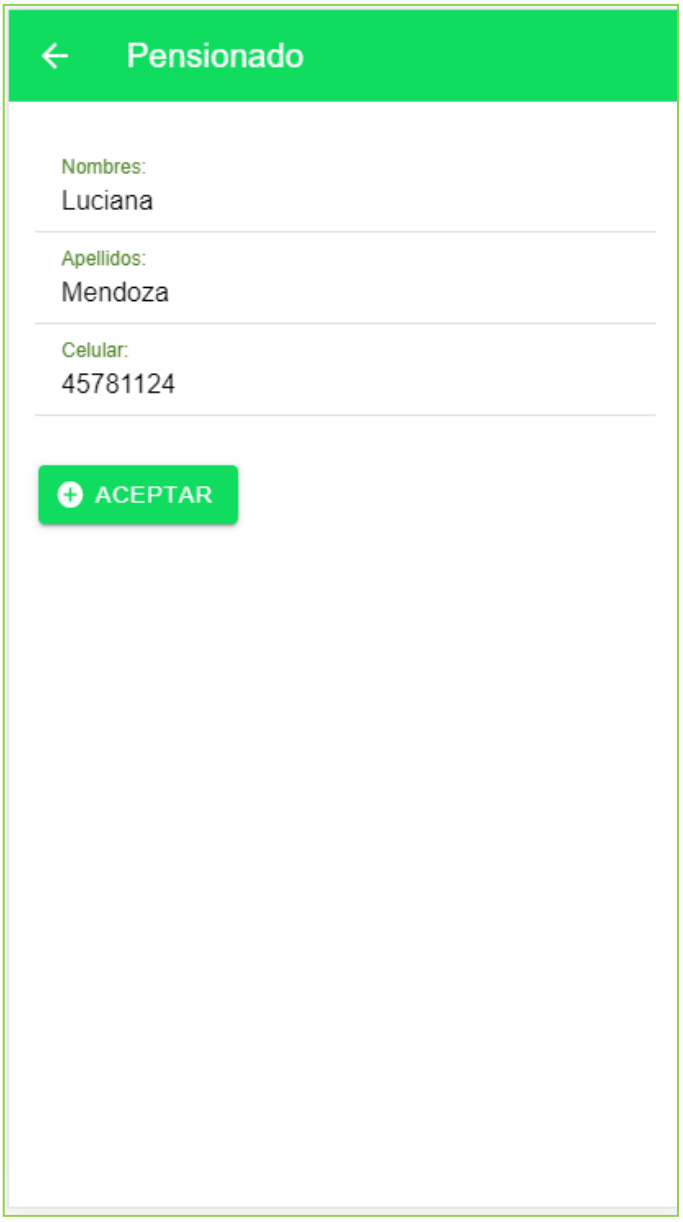
En esta pantalla permite visualizar el listado de los pensionados ya registrados en la aplicación móvil

Pantalla 4: Nuevo pensionado



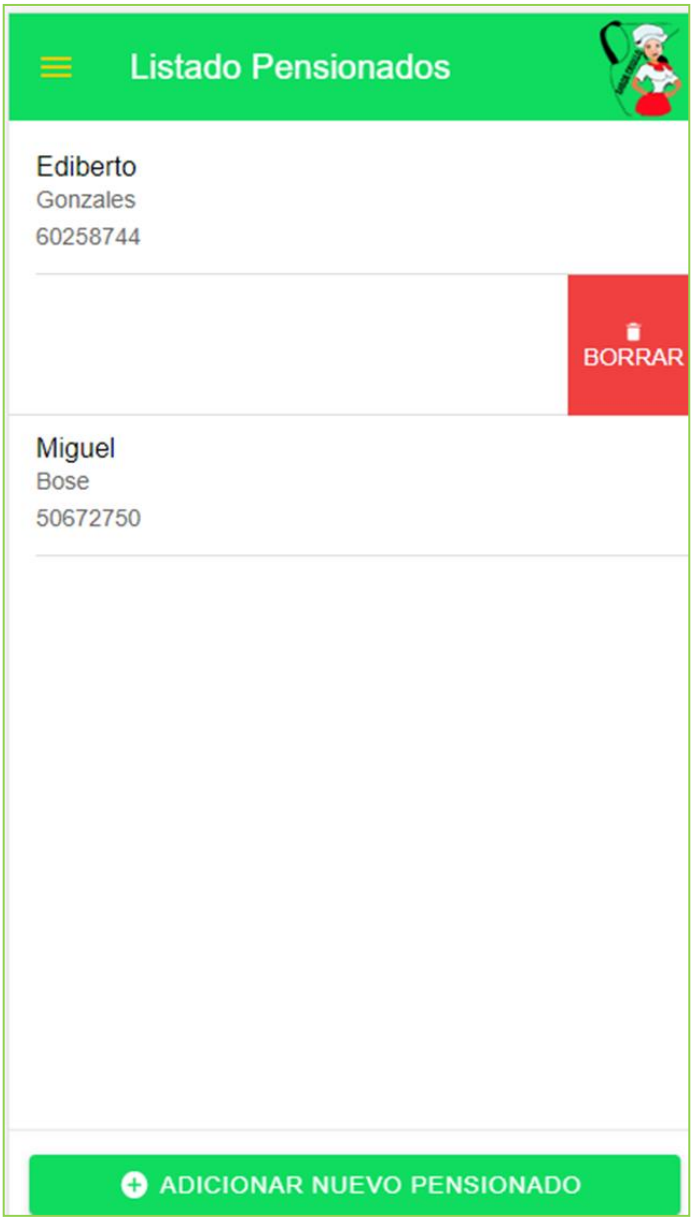
En esta pantalla permite introducir nuevos datos del pensionado llenado todos los campos y presionar el botón aceptar.

Pantalla 5: Modificar pensionado



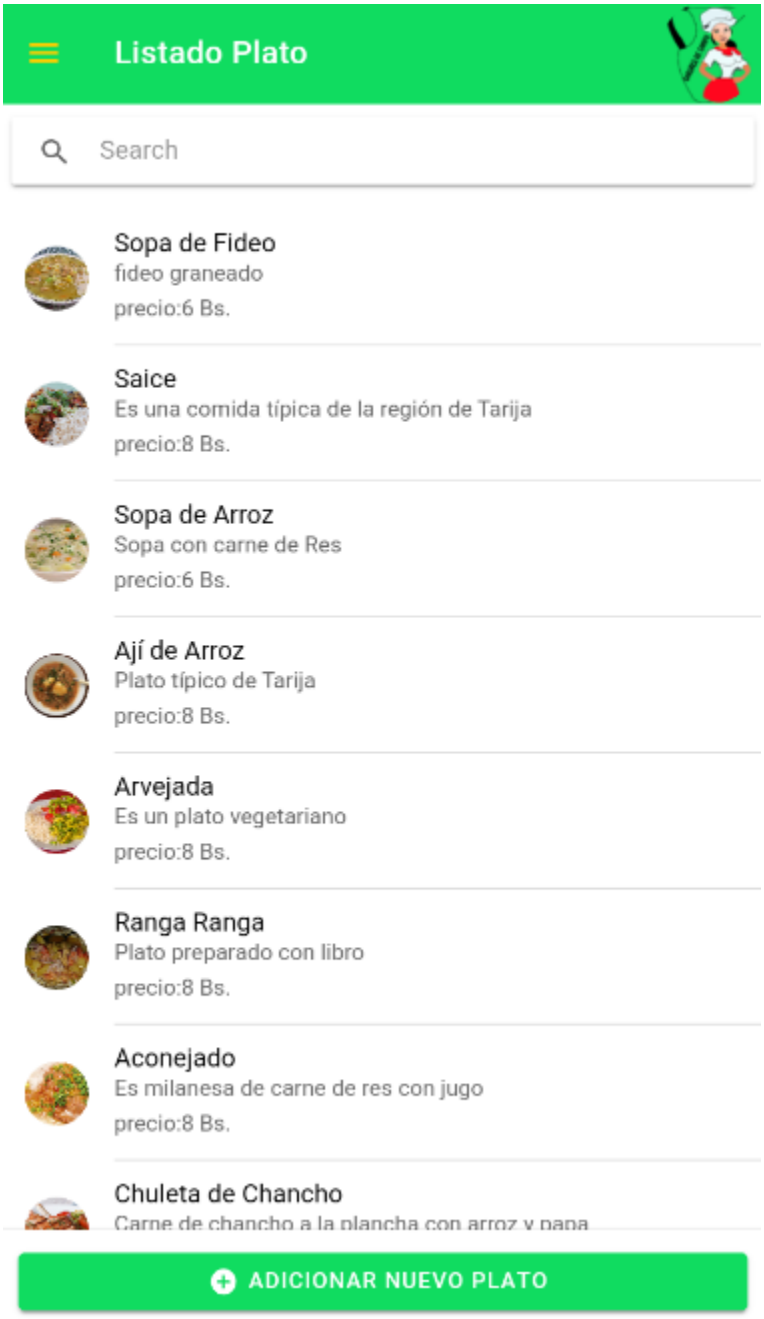
En esta pantalla muestra los campos y los datos que desea modificar y luego presiona en el botón aceptar.

Pantalla 6: Eliminar pensionado



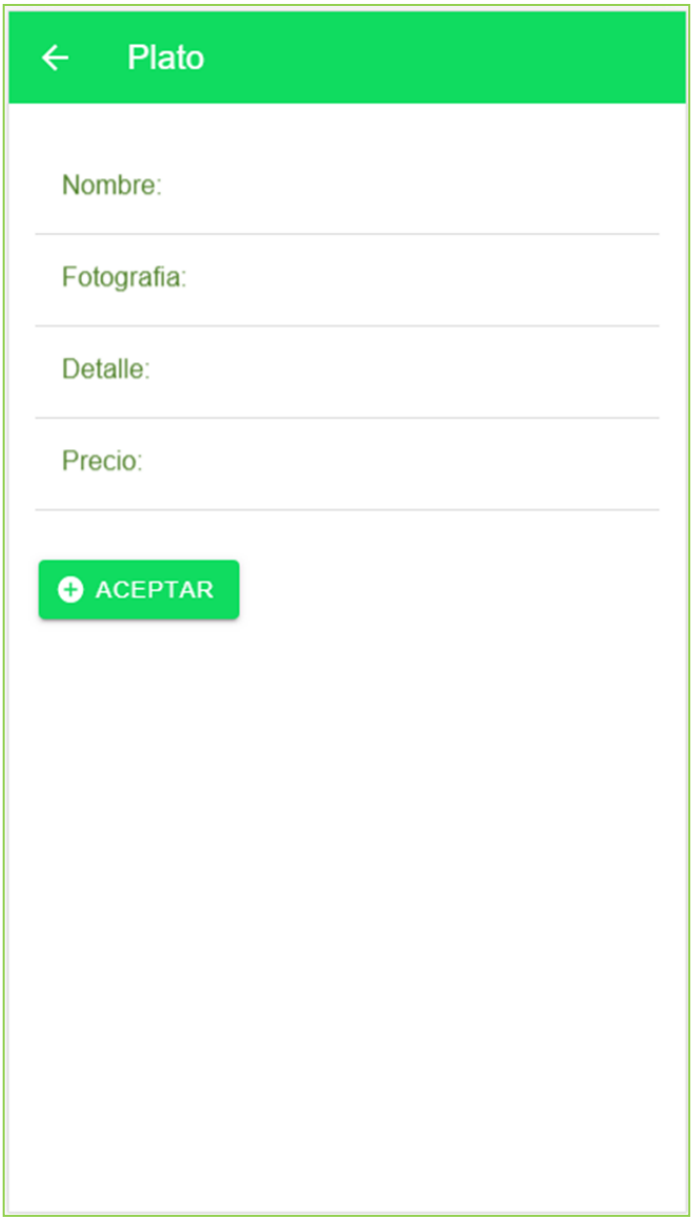
En esta pantalla muestra la lista de los pensionados registrados en la aplicación móvil y haciendo click y arrastrando hacia el lado izquierdo sobre los nombres de la lista aparece un botón borrar que permitirá eliminar los datos de la lista.

Pantalla 7: Listado de platos



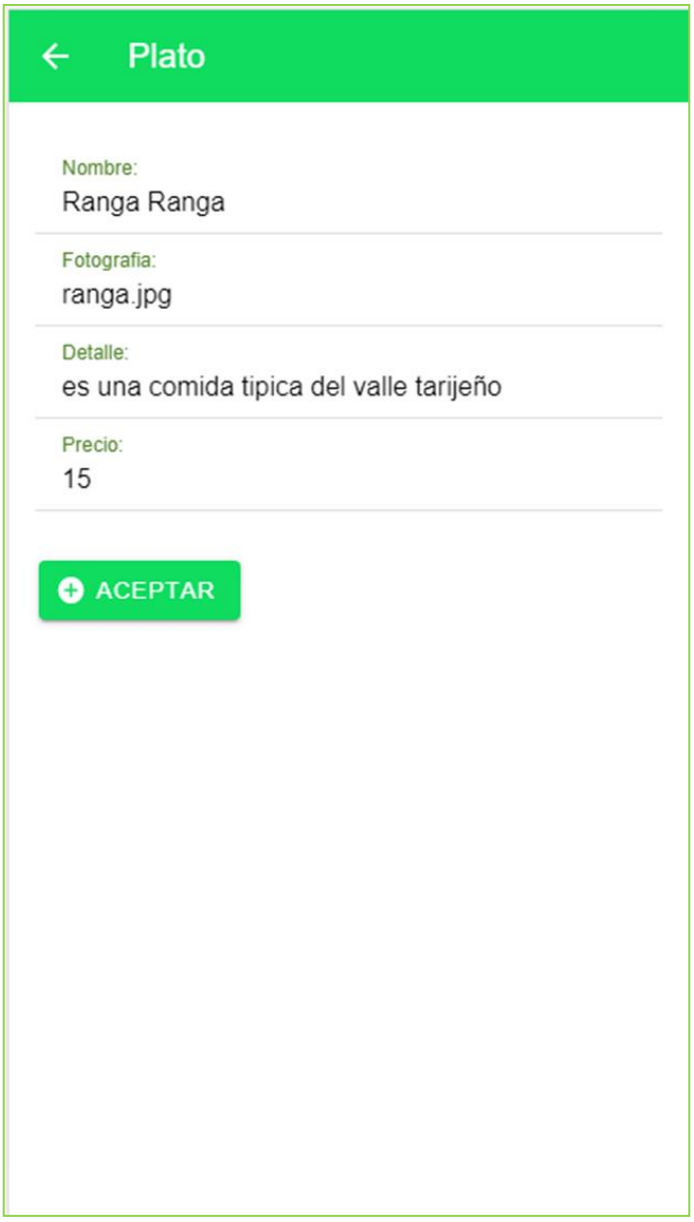
En esta pantalla permite visualizar el listado de los pensionados ya registrados en la aplicación móvil

Pantalla 8: Nuevo Platos



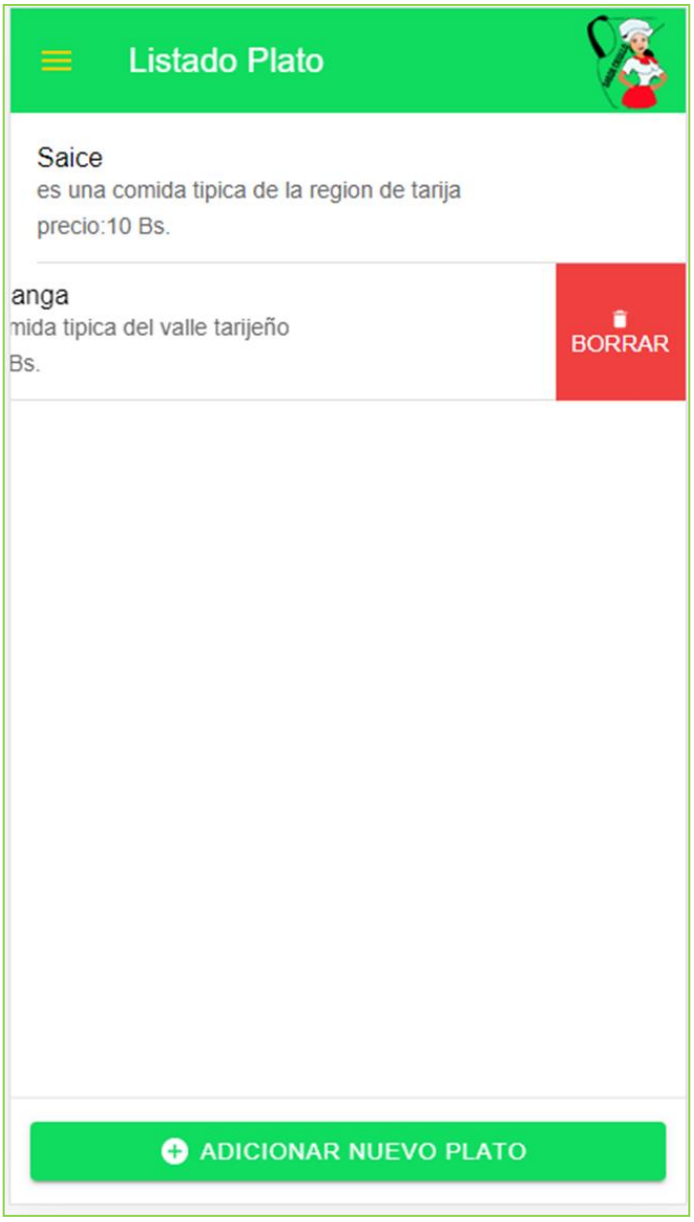
En esta pantalla permite introducir nuevos datos del plato, llenado todos los campos y presionar el botón aceptar.

Pantalla 9: Modificar Datos Platos



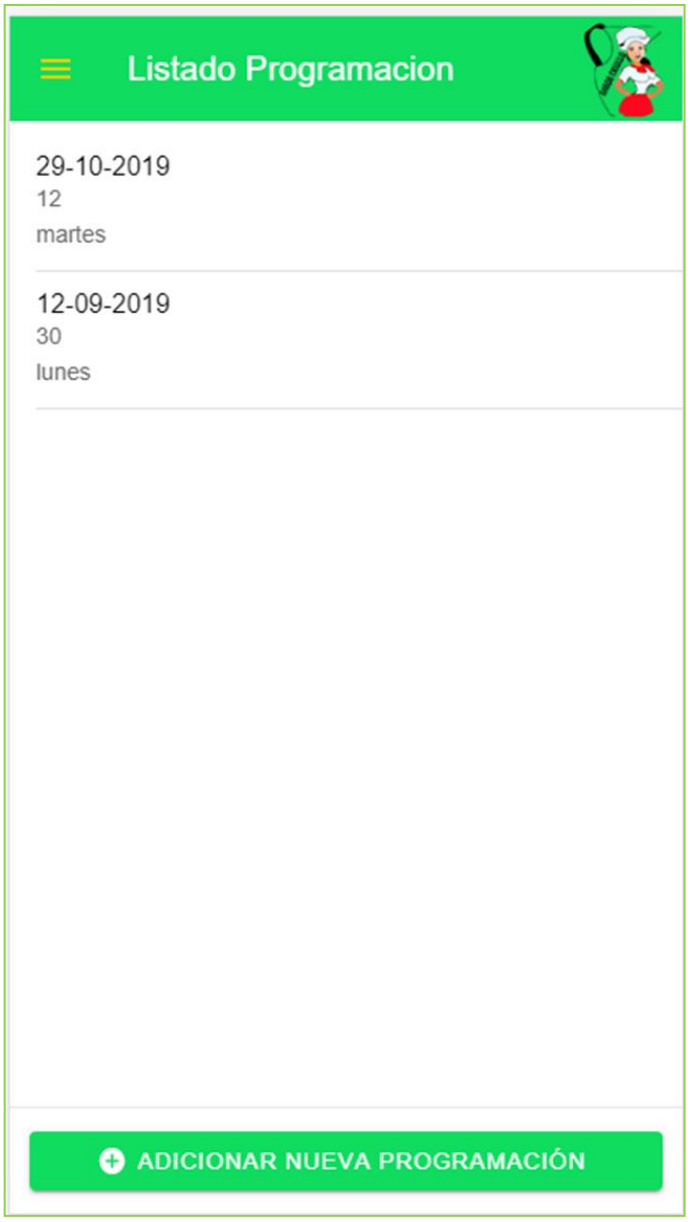
En esta pantalla muestra los campos y los datos que desea modificar y luego presiona en el botón aceptar.

Pantalla 10: Eliminar datos de platos



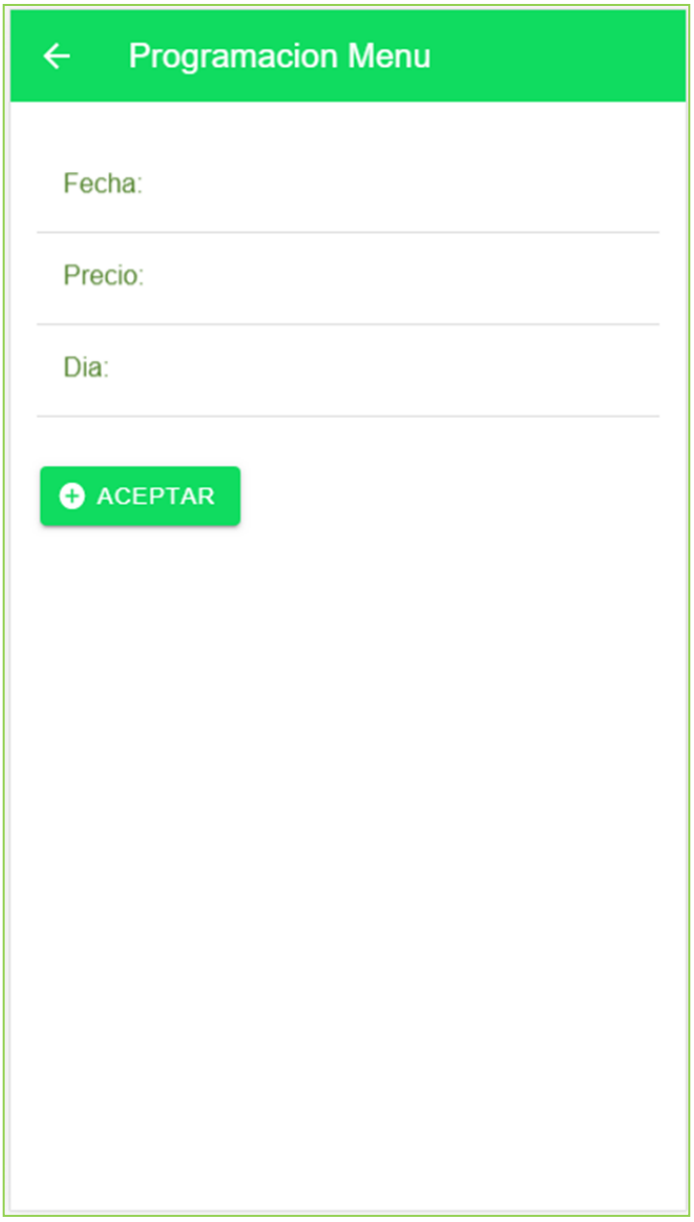
En esta pantalla muestra la lista de los platos registrados en la aplicación móvil y haciendo click y arrastrando hacia el lado izquierdo sobre los nombres de la lista aparece un botón borrar que permitirá eliminar los datos de la lista.

Pantalla 11: Listado de Programación Menú



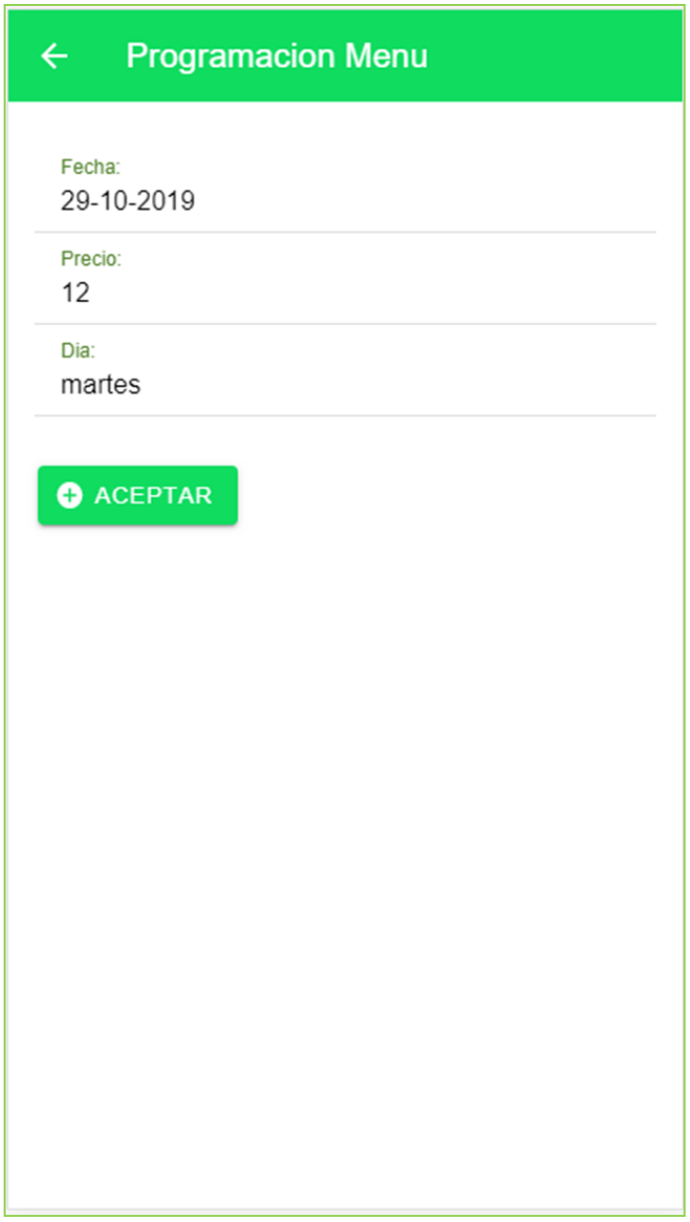
En esta pantalla permite visualizar el listado de los pensionados ya registrados en la aplicación móvil.

Pantalla 12: Adicionar Nueva Programación



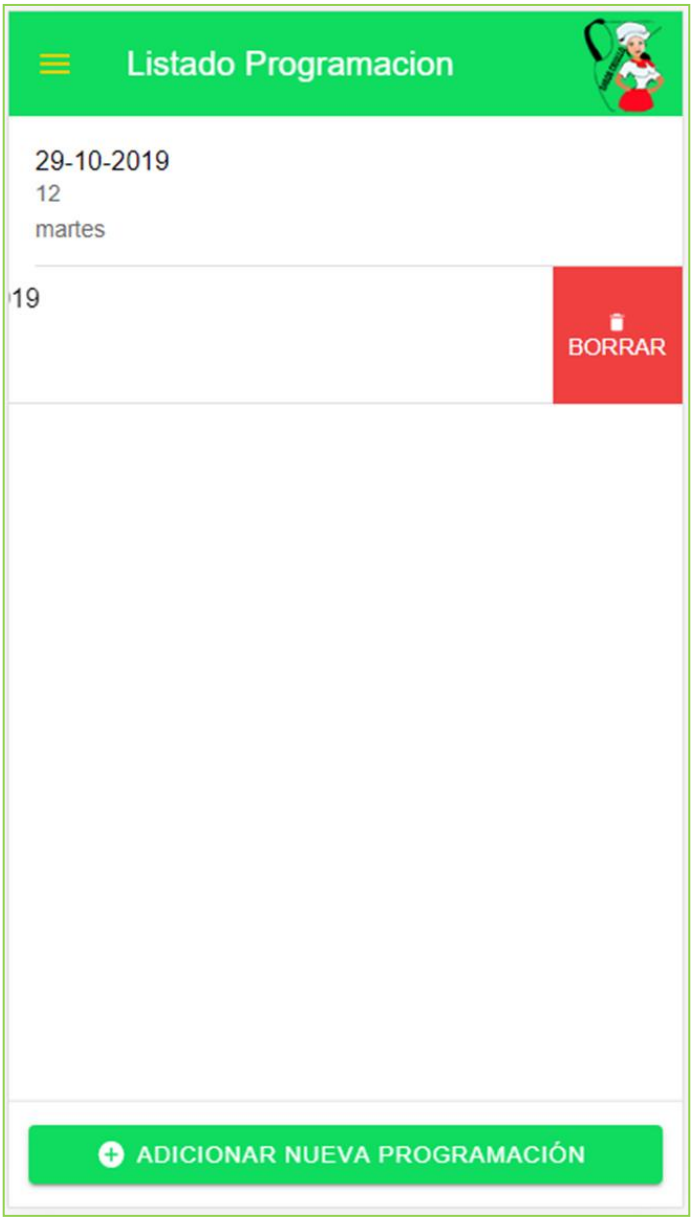
En esta pantalla permite realizar la programación del menú del día y de toda la semana y presiona el botón aceptar.

Pantalla 13: Modificar Programación



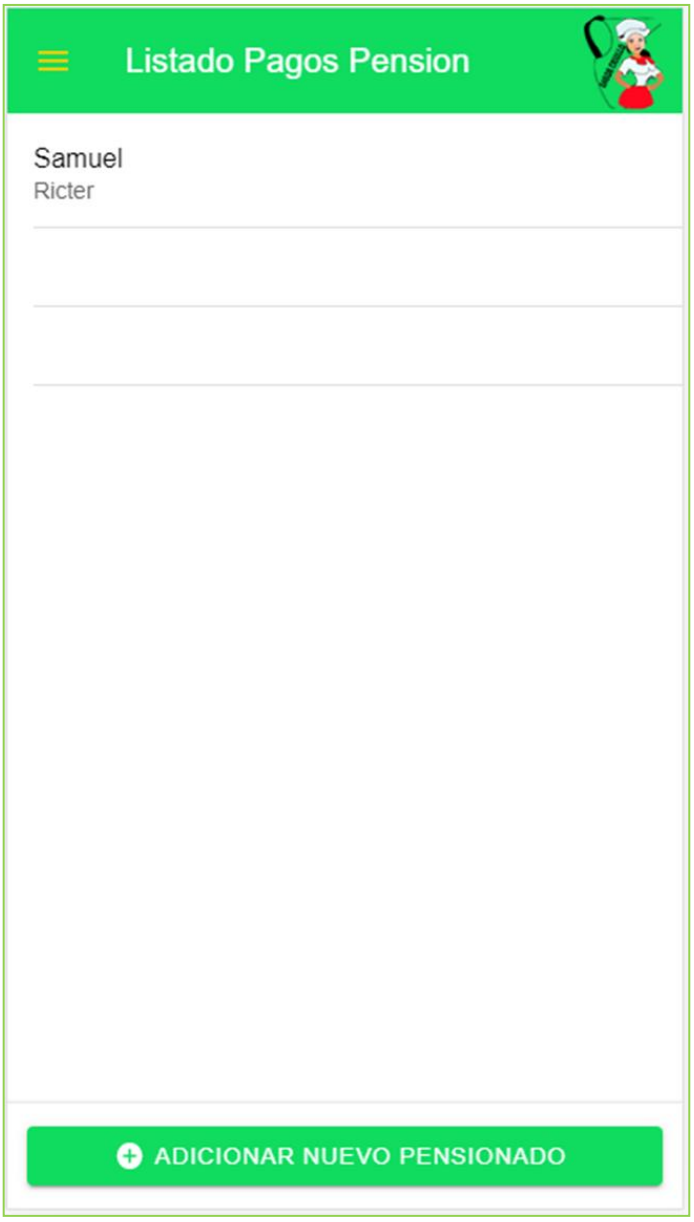
En esta pantalla muestra los campos y los datos que desea modificar y luego presiona en el botón aceptar.

Pantalla 14: Eliminar Programación



En esta pantalla muestra la lista del menú programados en la aplicación móvil y haciendo click y arrastrando hacia el lado izquierdo sobre los nombres de la lista aparece un botón borrar que permitirá eliminar los datos de la lista.

Pantalla 15: Listado de Cobros



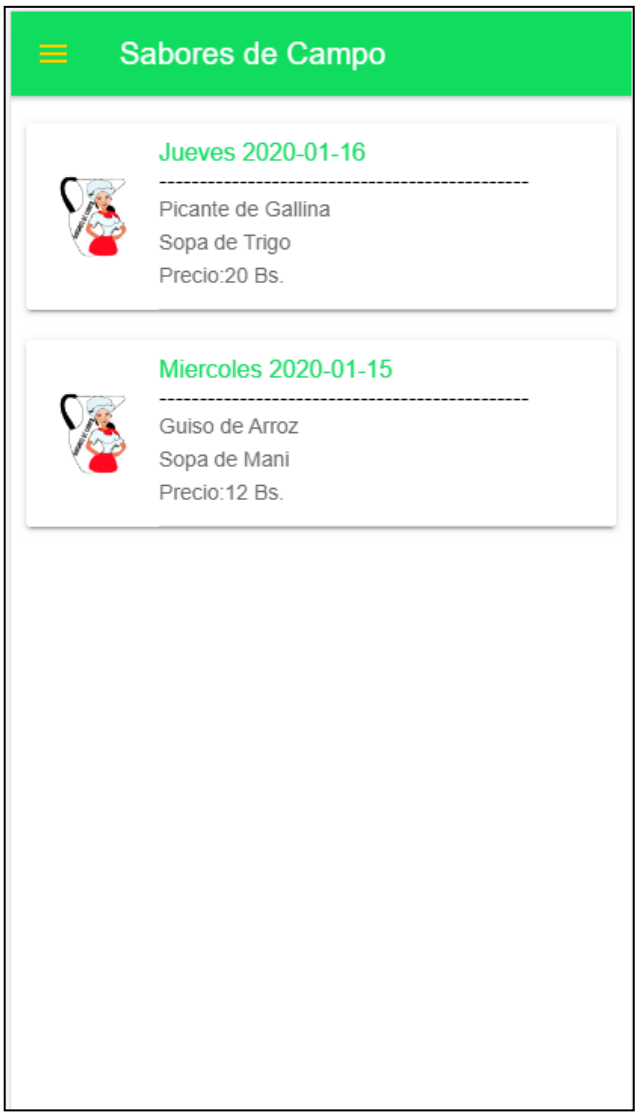
En esta pantalla permite visualizar el listado de los cobros ya registrados en la aplicación móvil

Pantalla 16: Nuevo Cobros

En esta pantalla permite registrar los nuevos cobros del pensionado llenado todos los campos y presionar el botón aceptar.

Pantallas Rol cliente

Pantalla 1: Pantalla principal

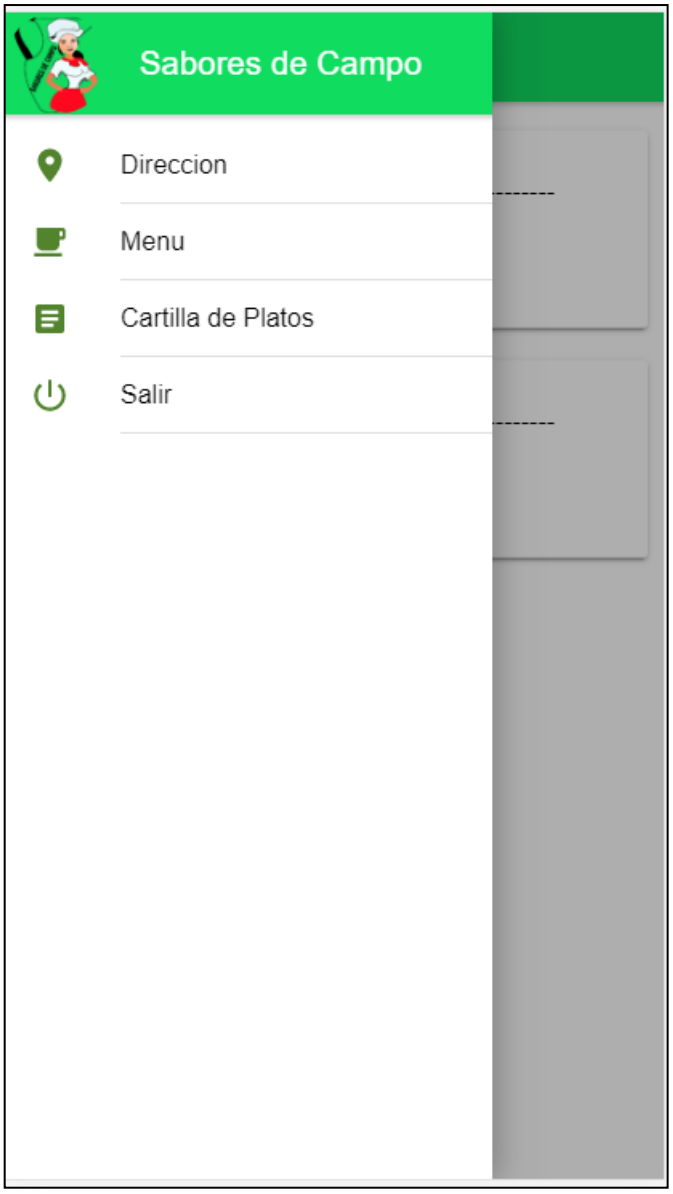


En esta pantalla visualiza la información del menú programado de los diferentes días.

Pantalla 2: Pantalla de Comunicación

En esta pantalla permite ver el menú a detalle y platos extras del día y opta por dos botones uno de llamada directa y otro hace referencia por whasApp.

Pantalla 3: Menú general

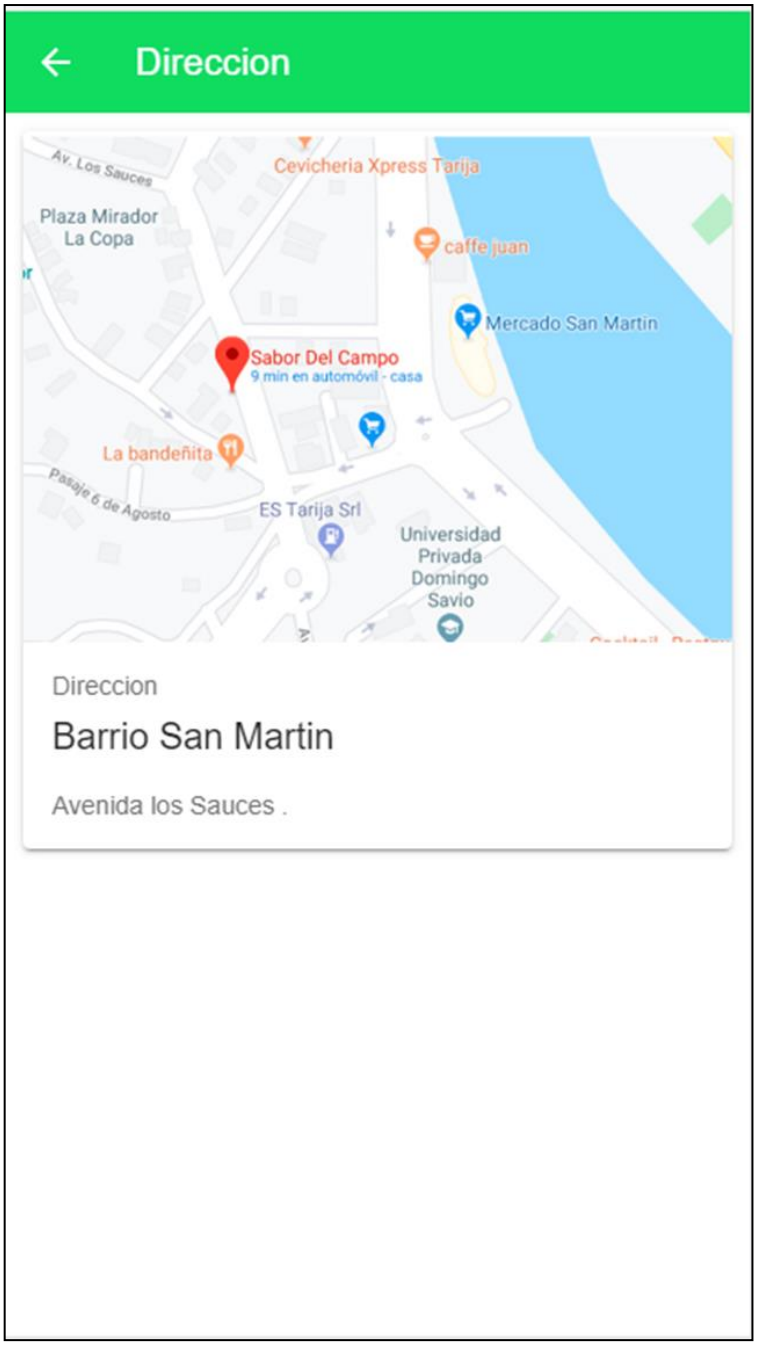


En esta pantalla permite la visualización de menú general donde el cliente puede ingresar y permite el salir de la apliación.

Pantalla 4: Cartilla de platos

En esta pantalla permite visualizar la cartilla que presenta el restaurante.

Pantalla 5: Dirección



En esta pantalla permite registrar la dirección de la empresa

5. IMPLEMENTACIÓN

Código de logueo de la aplicación móvil

```
export class LoginPage implements OnInit {
  //@ViewChild(IonSlides) slides: IonSlides;
  public userLogin: User = {};
  public userRegister: User = {};
  private loading: any;
  constructor(private loadingCtrl: LoadingController,
    private toastCtrl: ToastController,
    private authService: AuthService
  ) { }
  ngOnInit() {
  }
  async login() {
    await this.presentLoading();
    try {
      await this.authService.login(this.userLogin);
    } catch (error) {
        console.error(error);
        this.presentToast(error.message);
    } finally {
      this.loading.dismiss();
    }
  }

  async register() {
    await this.presentLoading();
    try {
      await this.authService.register(this.userRegister);
    } catch (error) {
        console.error(error);
        this.presentToast(error.message);
    } finally {
```

```
      this.loading.dismiss();
    }
  }//fin register
  //funcion para loading
  async presentLoading() {
    this.loading = await this.loadingCtrl.create({ message: 'Por favor, Aguarde...' });
    return this.loading.present();
  }
  async presentToast(message:string) {
    const toast = await this.toastCtrl.create({message,duration:2000});
    toast.present();
  }
}
```

Código menú principal

```
 export class MenuprincipalPage implements OnInit {
constructor(private authService: AuthService, private menuControl: MenuController) { }
  ngOnInit() {
  }
  async logout() {
    try {
      await this.authService.logout();
    } catch (error) {
      console.log(error);
    }
  }
  abriMenu() {
    this.menuControl.toggle();
  }
}
```

Código listado de pensionados

```
export class ListapensionPage implements OnInit {
```

```
  private pensionados = new Array<Pensionado>();
  private pensionadosSubscription: Subscription;
  private loading: any;
  constructor(
    private pensionadosService: PensionadoService,
    private authService: AuthService,
    private loadingCtrl:LoadingController,
    private toastCtrl:ToastController,
    public alertCtrl:AlertController
  ) {
 this.pensionadosSubscription = this.pensionadosService.getPensionados().subscribe(data =
> {
      this.pensionados = data;
    });
  }
  ngOnInit() { }
  ngOnDestroy() {
    this.pensionadosSubscription.unsubscribe();
  }
  async logout() {
    try {
      await this.authService.logout();
    } catch (error) {
      console.log(error);
    }
  }
  async deletePensionado(id:string){
    if(!this.presentAlert()){
        try{
        await this.pensionadosService.deletePensionado(id);
```

```
      }catch(error){
        this.presentToast('Error al intertar guardar');
      }
    }
}
//funcion para loading
async presentLoading() {
  this.loading = await this.loadingCtrl.create({ message: 'Por favor, Aguarde...' });
  return this.loading.present();
}
async presentToast(message: string) {
  const toast = await this.toastCtrl.create({ message, duration: 2000 });
  toast.present();
}
async presentAlert(){
  const alert=await this.alertCtrl.create({
    header:'Alert',
    subHeader:'Subtitle',
    message:'this is an alert message.',
    //buttons:['OK']
    buttons:[
      {
        text:'Cancel',
        role:'cancel',
        cssClass:'secondary',
        handler:(blah)=>{
```

```
          //return false;
          console.log('Cancelar');
        }
      },
      {
        text:'Ok',
        handler:(blah)=>{
          //return true;
          console.log('Boton Ok');
        }
      }
    ]
  });
  await alert.present();
 }
}
```

Código nuevo pensionado

```
await this.pensionadoService.addPensionado(this.pensionado);
await this.loading.dismiss();
this.navCtrl.navigateBack('/listapension');
```

Código modificar pensionado

```
await this.pensionadoService.updatePensionado(this.pensionadoId, this.pensionado);
await this.loading.dismiss();
this.navCtrl.navigateBack('/listapension');
} catch (error) {
this.presentToast("Error al intentar Grabar");
this.loading.dismiss();
}
```

código eliminar pensionado

```
deletePensionado(id:string) {
  return this.pensionadosCollection.doc(id).delete();
}
```

Código listado de platos

```
export class ListaplatoPage implements OnInit {
  private platos = new Array<Plato>();
  private platosSubscription: Subscription;
  private loading: any;
  constructor(
    private platosService: PlatoService,
    private authService: AuthService,
    private loadingCtrl:LoadingController,
    private toastCtrl:ToastController,
```

```
  public alertCtrl:AlertController
) {
  this.platosSubscription = this.platosService.getPlatos().subscribe(data => {
    this.platos = data;
  });
}
ngOnInit() { }
ngOnDestroy() {
  this.platosSubscription.unsubscribe();
}
async logout() {
  try {
    await this.authService.logout();
  } catch (error) {
    console.log(error);
  }
}
async deletePlato(id:string){

  if(!this.presentAlert()){
      try{
```

```
      await this.platosService.deletePlato(id);
        }catch(error){
        this.presentToast('Error al intertar guardar');
      }
    }
}
//funcion para loading
async presentLoading() {
  this.loading = await this.loadingCtrl.create({ message: 'Por favor, Aguarde...' });
  return this.loading.present();
}
async presentToast(message: string) {
  const toast = await this.toastCtrl.create({ message, duration: 2000 });
  toast.present();
}
async presentAlert(){
  const alert=await this.alertCtrl.create({
    header:'Alert',
    subHeader:'Subtitle',
    message:'this is an alert message.',
    //buttons:['OK']
```

```
    buttons:[
      {
        text:'Cancel',
        role:'cancel',
        cssClass:'secondary',
        handler:(blah)=>{
          return false;
          //console.log('Cancelar');
        }
      },
      {
        text:'Ok',
        handler:(blah)=>{
          return true;
          //console.log('Boton Ok');
        }
      }
    ]
  });
  await alert.present();
}
```

Código nuevo platos

```
await this.platoService.addPlato(this.plato);

await this.loading.dismiss();

this.navCtrl.navigateBack('/listaplato');
```

Código modificar platos

```
 await this.platoService.updatePlato(this.platoId, this.plato);

await this.loading.dismiss();

this.navCtrl.navigateBack('/listaplato');
```

Código eliminar datos de platos

```
deletePlato(id:string) {

  return this.platosCollection.doc(id).delete();

}
```

Código listado de programación de menú

```
export class ListaprogramaPage implements OnInit {

private programacions = new Array<Programacion>();

private programacionsSubscription: Subscription;

private loading: any;

constructor(

private programacionsService: ProgramacionService,

private authService: AuthService,
```

```
    private loadingCtrl:LoadingController,
    private toastCtrl:ToastController,
    public alertCtrl:AlertController
    ) {
    this.programacionsSubscription = this.programacionsService.getProgramacions().subscribe
(data => {
    this.programacions = data;
    });
    }
    ngOnInit() { }
    ngOnDestroy() {
    this.programacionsSubscription.unsubscribe();
    }
    async logout() {
    try {
    await this.authService.logout();
    } catch (error) {
    console.log(error);
    }
    }
    async deletePrograma(id:string){
```

```
if(!this.presentAlert()){
try{
await this.programacionsService.deleteProgramacion(id);
}catch(error){
this.presentToast('Error al intertar guardar');
}
}
}
//funcion para loading
async presentLoading() {
this.loading = await this.loadingCtrl.create({ message: 'Por favor, Aguarde...' });
return this.loading.present();
}
async presentToast(message: string) {
const toast = await this.toastCtrl.create({ message, duration: 2000 });
toast.present();
}
async presentAlert(){
const alert=await this.alertCtrl.create({
header:'Alert',
subHeader:'Subtitle',
```

```
message:'this is an alert message.',
//buttons:['OK']
buttons:[
{
text:'Cancel',
role:'cancel',
cssClass:'secondary',
handler:(blah)=>{
return false;
//console.log('Cancelar');
}
},
{
text:'Ok',
handler:(blah)=>{
return true;
//console.log('Boton Ok');
}
}
]
});
```

```
await alert.present();
}
}
```

Código adicionar programación de menú

```
await this.programacionService.addProgramacion(this.programacion);
await this.loading.dismiss();
this.navCtrl.navigateBack('/listaprograma');
```

Código modificar programación de menú

```
await this.programacionService.updateProgramacion(this.programacionId, this.programaci
on);
await this.loading.dismiss();
this.navCtrl.navigateBack('/listaprograma');
```

código eliminar programación de menú

```
deleteProgramacion(id:string) {
  return this.programacionsCollection.doc(id).delete();
}
```

Código listado de cobros de pensionados

```
export class ListapensionadosPage implements OnInit {
private cobros = new Array<Cobro>();
//private pensionados = new Array<Pensionado>();
```

```
private pensionadosSubscription: Subscription;

private loading: any;

constructor(

//private pensionadosService: PensionadoService,

private pensionadosService: CobroService,

private authService: AuthService,

private loadingCtrl:LoadingController,

private toastCtrl:ToastController,

public alertCtrl:AlertController

) {

this.pensionadosSubscription = this.pensionadosService.getCobros().subscribe(data => {

this.cobros = data;

});

}

ngOnInit() { }

ngOnDestroy() {

this.pensionadosSubscription.unsubscribe();

}

async logout() {

try {

await this.authService.logout();
```

```
} catch (error) {
console.log(error);
}
}
async deletePensionado(id:string){
if(!this.presentAlert()){
try{
await this.pensionadosService.deleteCobro(id);
}catch(error){
this.presentToast('Error al intertar guardar');
}
}
}
//funcion para loading
async presentLoading() {
this.loading = await this.loadingCtrl.create({ message: 'Por favor, Aguarde...' });
return this.loading.present();
}
async presentToast(message: string) {
const toast = await this.toastCtrl.create({ message, duration: 2000 });
toast.present();
```

```
}

async presentAlert(){

const alert=await this.alertCtrl.create({

header:'Alert',

subHeader:'Subtitle',

message:'this is an alert message.',

//buttons:['OK']

buttons:[

{

text:'Cancel',

role:'cancel',

cssClass:'secondary',

handler:(blah)=>{

//return false;

console.log('Cancelar');

}

},

{

text:'Ok',

handler:(blah)=>{

//return true;
```

```
console.log('Boton Ok');

}

}

]

});

await alert.present();

}

}
```

Código adicionar cobros de pensionados

```
export class CobroPage implements OnInit {

public fecha:Date=new Date();

//public fecha1:Date=new Date();

public pagos=[];

public pago:Pagos={};

public variable:any;

private pensionado: Pensionado = {};

private pensionadoId: string = null;

private pensionadoSubscription: Subscription;

private cobro: Pensionado = {};

private loading: any;

private cobroId: string = null;
```

```
private cobroSubscription: Subscription;

constructor(

private loadingCtrl: LoadingController,

private toastCtrl: ToastController,

private authService: AuthService,

private activeRoute: ActivatedRoute,

private cobroService: CobroService,

private pensionadoService: PensionadoService,

private navCtrl: NavController,

) {

//this.pensionadoId = this.activeRoute.snapshot.params['id'];

this.cobroId = this.activeRoute.snapshot.params['id'];

//if (this.pensionadoId) this.loadPensionado();

if (this.cobroId) this.loadCobro();

}

ngOnInit() {

/*this.cobroService.getCobro(this.cobroId).subscribe(datos=>{

console.log(datos);

this.variable=datos;

});

*/
```

```
    }
    ngOnDestroy(){
    if(this.pensionadoSubscription) this.pensionadoSubscription.unsubscribe();
    if(this.cobroSubscription) this.cobroSubscription.unsubscribe();
    }
    sendPago(){
    //this.pagos.push(this.pago);
    //this.pago={};
    this.pago.fecha_pago= new Date();
    const pago:Pagos={
    fecha_pago:this.pago.fecha_pago,
    monto:this.pago.monto,
    mes:this.pago.mes,
    gestion:this.pago.gestion
    }
    this.cobroService.sendToFirebird(this.pago,this.cobroId);
    this.pago={};
    }
    loadPensionado() {
    this.pensionadoSubscription = this.pensionadoService.getPensionado(this.pensionadoId).su
bscribe(data => {
```

```
this.cobro = data;
})
}
loadCobro() {
this.cobroSubscription = this.cobroService.getCobro(this.cobroId).subscribe(data => {
this.cobro = data;
})
}
async saveCobro() {
await this.presentLoading();
this.cobro.userId = this.authService.getAuth().currentUser.uid;
if (this.cobroId) {
try {
await this.cobroService.updateCobro(this.cobroId, this.cobro);
await this.loading.dismiss();
this.navCtrl.navigateBack('/listacobro');
} catch (error) {
this.presentToast("Error al intentar Grabar");
this.loading.dismiss();
}
} else {
```

```
//this.admin.createdAt = new Date().getTime();
try {
await this.cobroService.addCobro(this.cobro);
await this.loading.dismiss();
this.navCtrl.navigateBack('/listacobro');
} catch (error) {
this.presentToast("Error al intentar Grabar");
this.loading.dismiss();
}
}
}
//funcion para loading
async presentLoading() {
this.loading = await this.loadingCtrl.create({ message: 'Por favor, Aguarde...' });
return this.loading.present();
}
async presentToast(message: string) {
const toast = await this.toastCtrl.create({ message, duration: 2000 });
toast.present();
}
}
```

CÓDIGO ROL CLIENTE

Visualización del menú

```
import { Component, OnInit, ViewChild } from '@angular/core';
import { AuthService } from 'src/app/services/auth.service';
import { MenuController, IonSegment } from '@ionic/angular';
import { Observable, Subscription } from 'rxjs';
import { Programacion } from 'src/app/interfaces/programacion';
import { ProgramacionService } from 'src/app/services/programacion.service';
@Component({
  selector: 'app-menuprincipal',
  templateUrl: './menuprincipal.page.html',
  styleUrls: ['./menuprincipal.page.scss'],
})
export class MenuprincipalPage implements OnInit {
  fechactual: Date;
  fechactual2 = "";
  private programacions = new Array<Programacion>();
  public programacion: Programacion = {};
  private programacionsSubscription: Subscription;
  private programacionSubscription: Subscription;
  private loading: any;
```

```
    @ViewChild(IonSegment) segment: IonSegment;
    programenu: Observable<any>;
    constructor(private authService: AuthService,
      private menuControl: MenuController,
      private programacionsService: ProgramacionService,
      private programacionService: ProgramacionService
    ) {
      this.programacionsSubscription =
this.programacionsService.getProgramacions().subscribe(data => {
        this.programacions = data;
      });
      //this.segment.value='lunes';
      this.fechactual = new Date();
      if (this.fechactual.getDate() > 1 && (this.fechactual.getDate() <= 9)) {
        this.fechactual2 = this.fechactual.getFullYear() + "-0" + (this.fechactual.getMonth() +
1) + "-0" + this.fechactual.getDate()
      }
      if (this.fechactual.getMonth() < 9 && this.fechactual.getDate() > 9) {
        this.fechactual2 = this.fechactual.getFullYear() + "-0" + (this.fechactual.getMonth() +
1) + "-" + this.fechactual.getDate()
      }
```

```
      //console.log(this.fechactual.getFullYear() + "/" + (this.fechactual.getMonth() + 1) + "/"
+ this.fechactual.getDate());

      //this.fechactual2 = this.fechactual.getFullYear() + "-" + (this.fechactual.getMonth() + 1)
+ "-" + this.fechactual.getDate()

       console.log(this.fechactual2.substring(0, 4));

       console.log(this.fechactual2.substring(5, 7));

       console.log(this.fechactual2.substring(8, 10));

    }

    ngOnInit() {

    }

    ngOnDestroy() {

      this.programacionsSubscription.unsubscribe();

      //this.programacionSubscription.unsubscribe();

    }

    async logout() {

      try {

        await this.authService.logout();

      } catch (error) {

        console.log(error);

      }

    }

    abriMenu() {
```

```
    this.menuControl.toggle();
  }
  segmentChanged(event) {
    const valorSegmento = event.detail.value;
    console.log(valorSegmento);
  }
}
```

Código visualización detalle menú del día

```
import { Component, OnInit } from '@angular/core';
import { Programacion } from 'src/app/interfaces/programacion';
import { Subscription } from 'rxjs';
import { LoadingController, ToastController, NavController } from '@ionic/angular';
import { AuthService } from 'src/app/services/auth.service';
import { ProgramacionService } from 'src/app/services/programacion.service';
import { ActivatedRoute } from '@angular/router';
import { Plato } from 'src/app/interfaces/plato';
import { PlatoService } from 'src/app/services/plato.service';
import { SocialSharing } from '@ionic-native/social-sharing/ngx';
import { storage } from 'firebase';
import { CallNumber } from '@ionic-native/call-number/ngx';
@Component({
```

```
  selector: 'app-programacion',
  templateUrl: './programacion.page.html',
  styleUrls: ['./programacion.page.scss'],
})
export class ProgramacionPage implements OnInit {
  text:string="contactate con nosotros presionando este link" ;
  image:string=null;
  url:string="https://api.whatsapp.com/send?phone=59175142080";
  telefono:string="59175142080";
  private platos = new Array<Plato>();
  private platosSubscription: Subscription;
  private programacion: Programacion = {};
  private loading: any;
  private programacionId: string = null;
  private programacionSubscription: Subscription;
  constructor(
    private loadingCtrl: LoadingController,
    private toastCtrl: ToastController,
    private authService: AuthService,
    private activeRoute: ActivatedRoute,
    private programacionService: ProgramacionService,
```

```
    private navCtrl: NavController,
    private platoService:PlatoService,
    private socialSharing: SocialSharing,
    private callNumber:CallNumber
  ) {
    this.programacionId = this.activeRoute.snapshot.params['id'];
    if (this.programacionId) this.loadProgramacion();
    this.platosSubscription = this.platoService.getPlatos().subscribe(data => {
      this.platos = data;
      console.log(this.platos);
    });
  }
llamar(){
  this.callNumber.callNumber("75142080", true)
  .then(res => console.log('Llamando!', res))
  .catch(err => console.log('Error launching dialer', err));
}
  ngOnInit() {}
  ngOnDestroy(){
    if(this.programacionSubscription) this.programacionSubscription.unsubscribe();
    this.platosSubscription.unsubscribe();
```

```
  }

 whatsap(){

  console.log("funciona envia");

  //this.socialSharing.shareViaWhatsApp(this.text, this.image, this.url).then((res) => {

   this.socialSharing.shareViaWhatsAppToReceiver(this.telefono,this.text, this.image, this.url).then((res) => {

   // Success

    }).catch((e) => {

   // Error!

  });

 }

  loadProgramacion() {

   this.programacionSubscription = this.programacionService.getProgramacion(this.programacionId).subscribe(data => {

     this.programacion = data;

   })

  }

 }
```

Código menú general

```
<ion-menu side="start" contentId="principal" menuId="primerMenu">

  <ion-header>

   <ion-toolbar color="success">
```

```
    <ion-avatar slot="start">
      <img src="/assets/img/logo3.png">
    </ion-avatar>
    <ion-title>Sabores de Campo</ion-title>
  </ion-toolbar>
</ion-header>
<ion-content>
  <ion-list>
    <ion-menu-toggle>
      <ion-item routerLink="/direccion">
        <ion-icon color="primary" slot="start" name="pin"></ion-icon>
        Direccion
      </ion-item>
      <ion-item routerLink="/menuprincipal">
        <ion-icon color="primary" slot="start" name="cafe"></ion-icon>
        Menu
      </ion-item>
      <ion-item routerLink="//listaplato">
        <ion-icon color="primary" slot="start" name="list-box"></ion-icon>
        Código Cartilla de Platos
      </ion-item>
```

```
        <ion-item (click)="logout()">
          <ion-icon color="primary" slot="start" name="power"></ion-icon>
          Salir
        </ion-item>
      </ion-menu-toggle>
    </ion-list>
  </ion-content>
</ion-menu>
```

Código lista de cartilla

```
import { Component, OnInit } from '@angular/core';
import { Plato } from 'src/app/interfaces/plato';
import { Subscription } from 'rxjs';
import { PlatoService } from 'src/app/services/plato.service';
import { AuthService } from 'src/app/services/auth.service';
import { LoadingController, ToastController, AlertController } from '@ionic/angular';
import { CallNumber } from '@ionic-native/call-number/ngx';
@Component({
  selector: 'app-listaplato',
  templateUrl: './listaplato.page.html',
  styleUrls: ['./listaplato.page.scss'],
})
```

```
export class ListaplatoPage implements OnInit {
  private platos = new Array<Plato>();
  private platosSubscription: Subscription;
  private loading: any;
  constructor(
    private platosService: PlatoService,
    private authService: AuthService,
    private loadingCtrl:LoadingController,
    private toastCtrl:ToastController,
    public alertCtrl:AlertController,
    private callNumber:CallNumber
  ) {
    this.platosSubscription = this.platosService.getPlatos().subscribe(data => {
      this.platos = data;
    });
  }
  ngOnInit() { }
  ngOnDestroy() {
    this.platosSubscription.unsubscribe();
  }
  llamar(){
```

```
    this.callNumber.callNumber("75142080", true)
    .then(res => console.log('Llamando!', res))
    .catch(err => console.log('Error launching dialer', err));
    }
  async logout() {
    try {
      await this.authService.logout();
    } catch (error) {
      console.log(error);
    }
  }
}
```

6. CONCLUSIONES Y RECOMENDACIONES

6.1. Conclusiones

- Se realizó el diseño del interfaz amigable e interactiva para el módulo administrador y módulo cliente.
- Se registró la información Administrativa a través del Registro Automatizado de registro de platos, pensionados, cobros y programación de menú de una forma más ágil y eficaz.
- Se diseñó la base de datos específica para el llenado de la información y se utilizó Firebase para la implementación.
- Se utilizó la metodología Scrum para el desarrollo de la aplicación y también se utilizó UML para el modelado.

6.2. Recomendaciones

- La aplicación móvil deberá subirlo en el play story o podrá adquirirlo de forma directa en el restaurante.
- La aplicación móvil podrá ser instalado en celulares con Android versión 7 para adelante para su mejor funcionamiento.
- La aplicación móvil hace uso sólo 200 registros gratuitos en la base de datos de la nube si se llega pasar este rango tendré que pagar para el aumento de espacio y tener una óptima funcionalidad.
- Si se desea ampliar otro módulo en el administrador se debe tener en cuenta que sea los más específico posible según la necesidad.