



# Recetas saludables a tu medida: descubre nuestra app.



Come saludablemente, vive  
saludablemente

The smartphone screen displays the GreenChef app's user interface. At the top, the GreenChef logo is centered, featuring a stylized green leaf shape with the letters 'GC' inside and a green head of lettuce below it. Below the logo, the brand name 'GreenChef' is written in a dark green, sans-serif font. The main screen area is divided into two teal-colored rectangular buttons. The top button contains the text 'Recetas Personalizadas' in white. The bottom button contains the text 'Supermercados Cercanos' in white. The background of the phone screen is a light gray color with a subtle grid pattern.

## ÍNDICE

<b>1. ESTUDIO PRELIMINAR</b>	<b>2</b>
1.1 Descripción de la aplicación	2
1.2 Alcance	4
1.3 Valoración de alternativas existentes	5
1.4 Stack Tecnológico	6
1.5 Objetivos Generales	6
<b>2. ANÁLISIS DE LA APLICACIÓN</b>	<b>7</b>
2.1 Requisitos funcionales	7
2.2 Requisitos no funcionales	7
2.4 Casos de Uso	8
2.4.1 Descripción de los casos de uso	8
<b>3. MANUAL DE USUARIO</b>	<b>10</b>
3.1 Objetivo	10
3.2 Requerimientos	10
3.3 Funcionalidades de la Aplicación	10
1. Registrar un usuario	11
2. Iniciar sesión	12
3. Ver supermercados cercanos	13
4. Ver recetas	19
<b>4. PROTOTIPO</b>	<b>26</b>
4.1 Libro de Estilos	26
4.2 Login and Register	27
4.3 Home	27
4.4 Mapa, Recetas y Perfil	28
4.5 Recetas	28
4.6 Administrador	29
<b>5. BASE DE DATOS</b>	<b>30</b>
<b>6. MANUAL TÉCNICO</b>	<b>31</b>
6.1 Conexión con la base de datos MongoDB	31
6.2 Conexión a nivel de código	34
6.3 Operaciones CRUD	35
6.3.1 Búsqueda de datos	36
6.3.2 Inserción de datos	37
6.3.3 Actualización de datos	38
6.3.4 Eliminación de datos	38

## 1. ESTUDIO PRELIMINAR

### 1.1 Descripción de la aplicación

La tecnología y la innovación han revolucionado la forma en que las personas abordan su salud y nutrición. La aplicación móvil consiste en una plataforma diseñada para facilitar la búsqueda y la preparación de recetas saludables que satisfagan las preferencias alimentarias y las necesidades dietéticas de los usuarios.

Al abrir la aplicación, los usuarios son recibidos por una interfaz de usuario intuitiva que les permite registrarse e iniciar sesión. Una vez que se han autenticado, la aplicación mostrará una ventana principal donde nos encontraremos con 3 botones para las diferentes funcionalidades de la aplicación.

La aplicación también incluye una barra de navegación que permite a los usuarios acceder a otras funciones importantes de la aplicación. Por ejemplo, el mapa utiliza la geolocalización del usuario para mostrar los supermercados cercanos, lo que permite al usuario seleccionar un supermercado específico y navegar por el inventario de productos saludables que otros usuarios han introducido. Al seleccionar los elementos que consideren saludables.

Cuando un usuario selecciona un supermercado específico en el mapa de la aplicación, se mostrará una ventana donde se podrá observar información detallada sobre el supermercado seleccionado. En la parte superior de la ventana aparecerá el nombre del supermercado y su ubicación exacta en el mapa. Debajo de esto, habrá más información para que el usuario pueda explorar diferentes aspectos del supermercado.

El primer botón será "Productos", que mostrará una lista de los productos saludables que se pueden encontrar en el supermercado. Los productos se presentarán en una lista, con una foto y una breve descripción de cada uno. Al hacer clic en un producto en particular, se mostrará una ventana emergente con información sobre el producto, como su precio y su disponibilidad en el supermercado.

El siguiente botón consiste en añadir productos a dicho supermercado. Una vez el usuario pulse un supermercado en el mapa contará con la posibilidad de introducir un producto que considere saludable en dicho supermercado, añadiendo información de este como el precio, kcal...

Por otro lado tenemos la ventana de recetas, donde en primer lugar, aparecerá varias opciones de tipo de comida para que el usuario pueda elegir cual prefiere, ya sea proteína, verdura...hasta postres saludables.

Una vez pulse en el tipo de receta que desea, le aparecerá una lista de las diferentes recetas basadas en el tipo de alimento que ha seleccionado el usuario. Cuando pulse una receta, se le enviará a otra ventana donde podrá ver información detallada de dicha receta como, una descripción breve, tiempo de preparación, los ingredientes y los pasos de preparación.

Asimismo, se dispondrá de una sección denominada "Perfil de Usuario" que exhibirá datos relacionados con el usuario, además de brindar la opción de efectuar modificaciones en dicha información, con el propósito de permitir al usuario ampliar los detalles de su cuenta.

Una de las principales ventajas de esta aplicación es la capacidad de personalización. La aplicación recopila información detallada sobre las preferencias alimentarias y las restricciones dietéticas de los usuarios, lo que permite ofrecer una lista de recetas adaptadas a sus necesidades. Por ejemplo, si el usuario es vegetariano o vegano, la aplicación puede proporcionar recetas específicas que eviten el uso de productos de origen animal. Además, si el usuario tiene alergias o intolerancias alimentarias, la aplicación también puede adaptarse para ofrecer opciones que eviten los ingredientes que puedan causar problemas de salud. La aplicación también proporciona información detallada sobre los ingredientes necesarios para cada receta, así como una lista de compras sugerida para que el usuario pueda adquirir los ingredientes necesarios en el supermercado.

Otra característica destacable de esta aplicación es la facilidad de uso. La interfaz de usuario es intuitiva y fácil de navegar, lo que permite a los usuarios encontrar rápidamente lo que están buscando. Además, la aplicación proporciona información detallada sobre los ingredientes necesarios para cada receta, lo que facilita la preparación de las comidas. Incluso se proporciona una lista de compras sugerida que incluye los ingredientes necesarios para cada receta, lo que facilita la planificación de las compras en el supermercado.

En resumen, esta aplicación móvil es una herramienta valiosa para las personas que buscan una forma fácil y conveniente de encontrar y preparar recetas saludables que se ajusten a sus preferencias y necesidades dietéticas. Con una interfaz de usuario intuitiva, una lista de recetas personalizadas y una lista de compras sugerida, los usuarios pueden estar seguros de que tienen todo lo que necesitan para preparar comidas saludables en su hogar. Además, la capacidad de buscar productos saludables en supermercados cercanos también es una característica valiosa para aquellos que buscan opciones de alimentos saludables y convenientes mientras están en movimiento.

## 1.2 Alcance

Las aplicaciones móviles que ofrecen recetas personalizadas saludables se han convertido en una herramienta muy popular entre los usuarios preocupados por su alimentación y estilo de vida saludable. Esta aplicación puede proporcionar una amplia gama de beneficios, desde mejorar la calidad de vida hasta promover la pérdida de peso y prevenir enfermedades crónicas.

El alcance de dicha aplicación puede llegar a un público muy amplio. Al ofrecer recetas personalizadas, la aplicación puede adaptarse a las preferencias y necesidades específicas de cada usuario.

GreenChef puede ofrecer recetas que se adapten a una gran variedad de planes alimentarios. Además, las recetas pueden ser adaptadas para cumplir con diferentes necesidades de nutrición.

Otro beneficio importante es que puede fomentar la educación nutricional y la conciencia de lo que se está comiendo. Al ofrecer información detallada sobre los ingredientes de cada receta, los usuarios pueden aprender sobre la nutrición y tomar decisiones más informadas sobre su alimentación. Además, puede ayudar a los usuarios a planificar sus comidas con anticipación y a controlar su consumo de calorías.

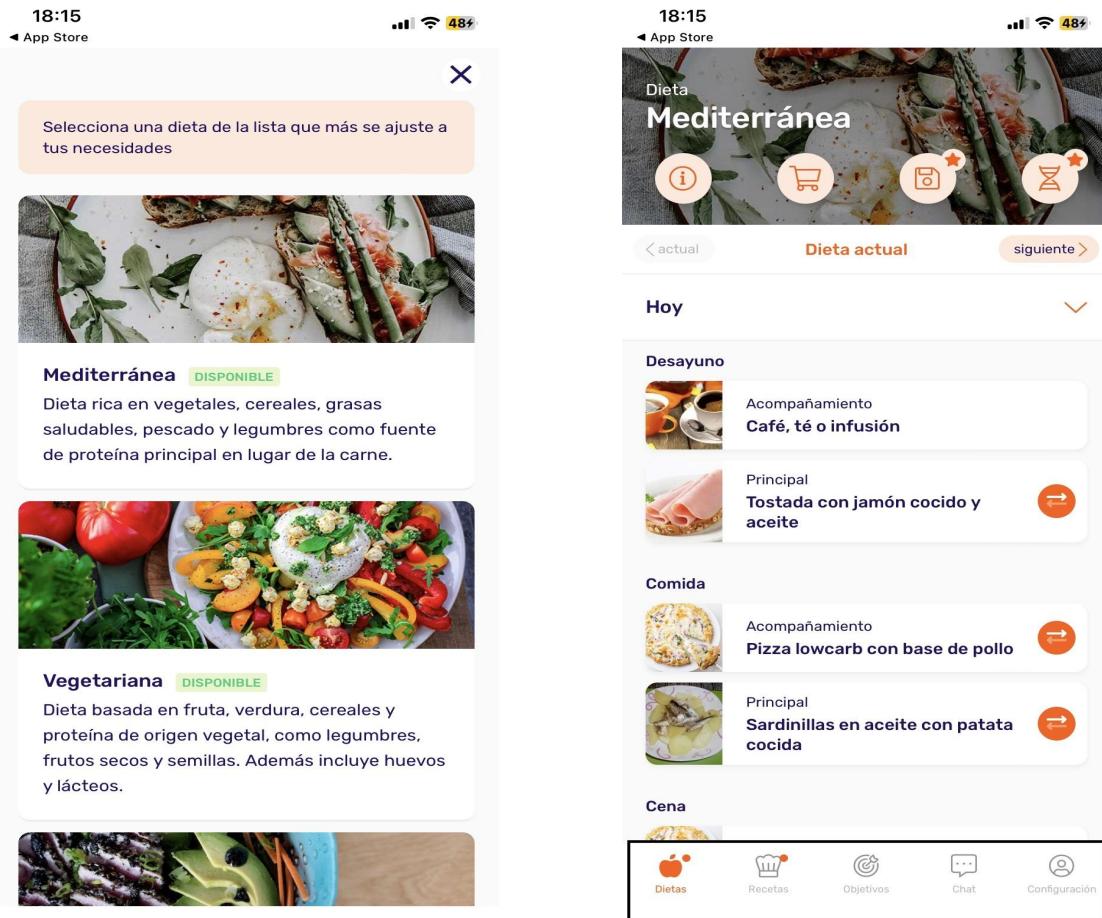
GreenChef tiene un alcance amplio y puede ser beneficiosa para una amplia gama de usuarios preocupados por su alimentación y su estilo de vida saludable. Puede adaptarse a las necesidades y preferencias individuales de cada usuario, fomentar la educación nutricional, y ayudar a los usuarios a planificar y controlar su ingesta de alimentos. En definitiva, esta aplicación puede ser una herramienta poderosa para mejorar la salud y el bienestar de sus usuarios.

### 1.3 Valoración de alternativas existentes

En un mercado cada vez más competitivo en términos de aplicaciones, es común encontrar diversas alternativas que ofrecen soluciones similares. En el caso de la aplicación mencionada, es esencial tener conocimiento de las opciones disponibles en el mercado y compararlas con la aplicación en cuestión.

Al evaluar las diferentes aplicaciones encontradas, se puede observar que en las aplicaciones que ofrecen recetas saludables, usualmente se presentan varias recetas preestablecidas al ingresar a la aplicación. En contraste, la aplicación GreenChef utiliza un enfoque distinto, al preguntar al usuario sobre sus preferencias alimenticias para generar una receta saludable personalizada.

Otra ventaja distintiva de la aplicación GreenChef en comparación con otras aplicaciones es su función de mapa de supermercados cercanos. Esta característica resulta particularmente útil, ya que no se ha encontrado otra aplicación que ofrezca un mapa que especifique los supermercados cercanos que ofrecen los ingredientes saludables necesarios para la receta seleccionada.



#### 1.4 Stack Tecnológico

- Lenguaje de programación: Java
- Base de datos: Base de Datos REALM
- Framework de desarrollo: Android Studio

#### 1.5 Objetivos Generales

1. Proporcionar recetas saludables y personalizadas: El objetivo principal de la aplicación es ofrecer a los usuarios una selección de recetas personalizadas y saludables según sus preferencias y necesidades alimenticias.
2. Fomentar un estilo de vida saludable: La aplicación tiene como objetivo motivar a los usuarios a adoptar hábitos de alimentación saludable y ayudarlos a mantenerse en forma y saludables.
3. Mejorar la accesibilidad a la información nutricional: La aplicación proporciona a los usuarios información nutricional detallada sobre los ingredientes y las recetas para que puedan tomar decisiones más informadas sobre su alimentación.
4. Facilitar la planificación de comidas: La aplicación ayuda a los usuarios a planificar sus comidas de manera más efectiva, proporcionándoles sugerencias de recetas saludables.
5. Ofrecer una experiencia de usuario intuitiva y satisfactoria: La aplicación ofrece una experiencia de usuario fluida y atractiva, con una interfaz intuitiva y fácil de usar, para que los usuarios puedan disfrutar de la aplicación y encontrar rápidamente lo que buscan.

## 2. ANÁLISIS DE LA APLICACIÓN

### 2.1 Requisitos funcionales

- La aplicación debe permitir a los usuarios registrarse en la plataforma proporcionando información personal como nombre, dirección de correo electrónico y una contraseña.
- Por otro lado debe permitir a los usuarios poder acceder con sus credenciales si ya se encuentran registrados en la base de datos.
- La aplicación debe permitir a los usuarios seleccionar los diferentes alimentos que desea para después realizar recetas personalizadas con dichos alimentos. Deberá permitir ver la información de dichas recetas.
- La aplicación debe permitir a los usuarios registrar los alimentos que consideren más saludables en los supermercados más cercanos.
- La aplicación debe enviar notificaciones a los usuarios cuando se inserte un nuevo alimento saludable en su supermercado más cercano.
- Como administrador de la aplicación, se cuenta con la funcionalidad de eliminar productos que hayan sido añadidos por error al supermercado correspondiente. Esta funcionalidad permitirá corregir errores en la base de datos de productos, eliminando aquellos que no deban estar disponibles en la aplicación.

### 2.2 Requisitos no funcionales

- Usabilidad: La aplicación será fácil de usar y navegar para los usuarios, incluso aquellos que no tienen experiencia en la cocina o en el uso de aplicaciones.
- Seguridad: La aplicación garantiza la seguridad de la información personal y financiera de los usuarios, así como la privacidad de sus preferencias alimenticias y recetas guardadas.
- Fiabilidad: Confiable y disponible para los usuarios en todo momento, sin interrupciones o fallos.
- Escalabilidad: Escalable para manejar un aumento en la cantidad de usuarios y datos de la aplicación.
- Interoperabilidad: La aplicación será compatible con otras aplicaciones y dispositivos que los usuarios puedan tener.
- Diseño atractivo: Tendrá un diseño atractivo y moderno para atraer a los usuarios y fomentar la participación en la aplicación.

### 2.3 Requisitos de interfaz

Cuando inicies la aplicación, se observarán tres botones diferentes. Uno es para iniciar sesión y tiene dos campos de texto para que ingreses tus credenciales. Los otros dos son para registrarse, y si los presionas, aparecerá una interfaz similar pero con campos de texto diferentes para que puedas registrar tus credenciales.

La interfaz principal de la aplicación tiene varios botones:

- Generar recetas según tus preferencias.
- Mostrar en un mapa los supermercados cercanos a tu ubicación.
- Acceder a la interfaz de usuario para modificar tus datos.

## 2.4 Casos de Uso

### 2.4.1 Descripción de los casos de uso

<b>Caso de uso : Registro e Inicio de sesión de usuario</b>		
<b>Precondición</b>	El usuario debe estar en la página de registro de la aplicación.	
<b>Descripción</b>	El usuario puede registrarse en la aplicación con sus datos personales y de contacto para poder realizar reservas y acceder a su historial de reservas.	
<b>Secuencia Normal</b>	<b>Paso</b>	<b>Acción</b>
	1	El usuario hace clic en el botón "registrarse".
	2	La aplicación muestra un formulario de registro
	3	El usuario completa el formulario con sus datos personales y de contacto.
	4	El usuario hace clic en el botón "enviar"
<b>Postcondición</b>	5	La aplicación valida la información del formulario y crea una cuenta de usuario para el usuario
	El usuario tiene una cuenta en la aplicación y puede acceder a las funciones de reserva y consulta de historial de reservas.	
<b>Excepciones</b>	<b>Paso</b>	<b>Acción</b>
	1	Si el usuario ya tiene una cuenta registrada, la aplicación muestra un mensaje de error y solicita que inicie sesión en lugar de registrarse de nuevo.
	E.1	Si el usuario proporciona información incompleta o incorrecta en el formulario, la aplicación muestra un mensaje de error y solicita que corrija la información antes de enviarla.

<b>Caso de uso : Búsqueda de Supermercado</b>		
<b>Precondición</b>	El usuario debe haber iniciado sesión en la aplicación.	
<b>Descripción</b>	El usuario puede buscar diferentes supermercados cercanos a través del mapa que proporciona la aplicación	
<b>Secuencia Normal</b>	<b>Paso</b>	<b>Acción</b>
	<b>1</b>	El usuario hace clic en el botón situado en la parte superior izquierda.
	<b>2</b>	La aplicación muestra un lateral desplegable.
	<b>3</b>	El usuario hace clic en el botón de "mapa".
	<b>4</b>	La aplicación muestra un mapa con los diferentes supermercados cercanos marcados.
	<b>5</b>	El usuarios podrá seleccionar el supermercado que más deseé.
<b>Postcondición</b>	El usuario puede ver la información de los diferentes supermercados junto con sus productos más saludables.	
<b>Excepciones</b>	<b>Paso</b>	<b>Acción</b>
	<b>1</b>	Si no existen supermercados cercanos al usuario, el mapa no mostrará ningún supermercado marcado.
	<b>E.1</b>	Si el usuario no ha iniciado sesión en la aplicación, la aplicación muestra un mensaje indicando que debe iniciar sesión para buscar supermercados cercanos.

### 3. MANUAL DE USUARIO

#### 3.1 Objetivo

El objetivo de la aplicación móvil es facilitar la búsqueda y preparación de recetas saludables, adaptadas a las preferencias alimentarias y necesidades dietéticas de los usuarios, aprovechando las posibilidades que brindan la tecnología y la innovación en el ámbito de la salud y nutrición.

#### 3.2 Requerimientos

1. Sistema operativo: Android Studio requiere un sistema operativo compatible, como Windows, macOS o Linux.
2. Hardware: Se recomienda tener un equipo con suficiente potencia para ejecutar Android Studio sin problemas. Esto incluye un procesador de al menos 2 GHz, 8 GB de RAM y un disco duro con espacio suficiente para la instalación y almacenamiento de archivos de proyecto.
3. Java Development Kit (JDK): Android Studio utiliza el lenguaje de programación Java para desarrollar aplicaciones de Android, por lo que necesitarás tener instalado el JDK de Java en tu sistema.
4. Android SDK: Android Studio viene con el Android Software Development Kit (SDK) integrado. Sin embargo, debes asegurarte de tener las versiones adecuadas del SDK y las herramientas de compilación instaladas para el nivel de API objetivo que deseas desarrollar.
5. Conexión a Internet: Para descargar las dependencias y las actualizaciones del SDK, necesitarás una conexión a Internet estable.

#### 3.3 Funcionalidades de la Aplicación

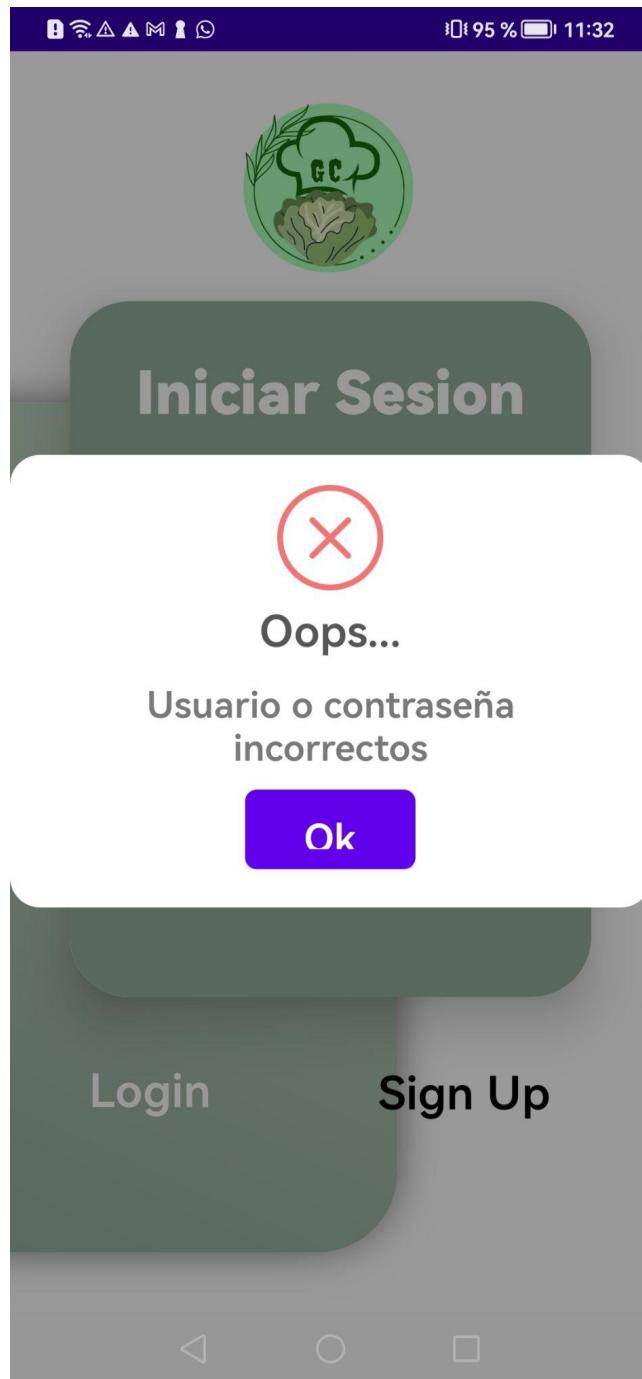
Las funcionalidades que permite esta aplicación son las siguientes:

1. Registrar un usuario.
2. Iniciar Sesión.
3. Ver supermercados cercanos.
  - a. Ver información sobre el supermercado.
  - b. Añadir productos.
  - c. Ver productos.
4. Ver recetas.
  - a. Seleccionar el tipo de receta.
  - b. Seleccionar dicha receta.
  - c. Ver información para la elaboración de la receta.
5. Ver perfil de usuario.

## 1. Registrar un usuario

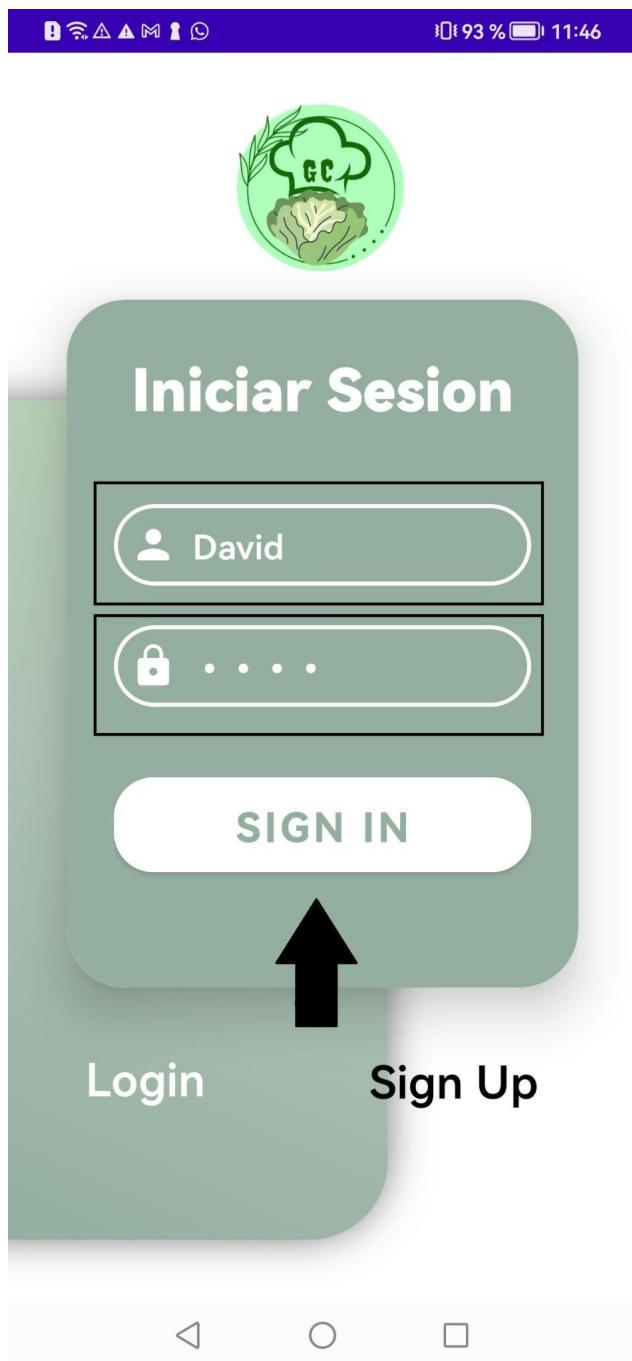
Cuando un usuario que no se encuentra registrado en la aplicación y desea iniciar sesión, le saldrá una alerta diciendo que el usuario no se encuentra registrado por lo que tendrá que dirigirse a la pestaña de registro para poder iniciar sesión posteriormente.

Una vez en la ventana de registro, tendremos que introducir nuestros datos, como el nick, el correo electrónico y la contraseña. Si todo está correcto nos saldrá un mensaje de registro exitoso y nos enviará automáticamente a la venta de inicio de sesión.



## 2. Iniciar sesión

Una vez el usuario se haya registrado, podrá iniciar sesión con las credenciales que ha decidido introducir. Una vez dentro aparecerá la ventana principal de la aplicación.



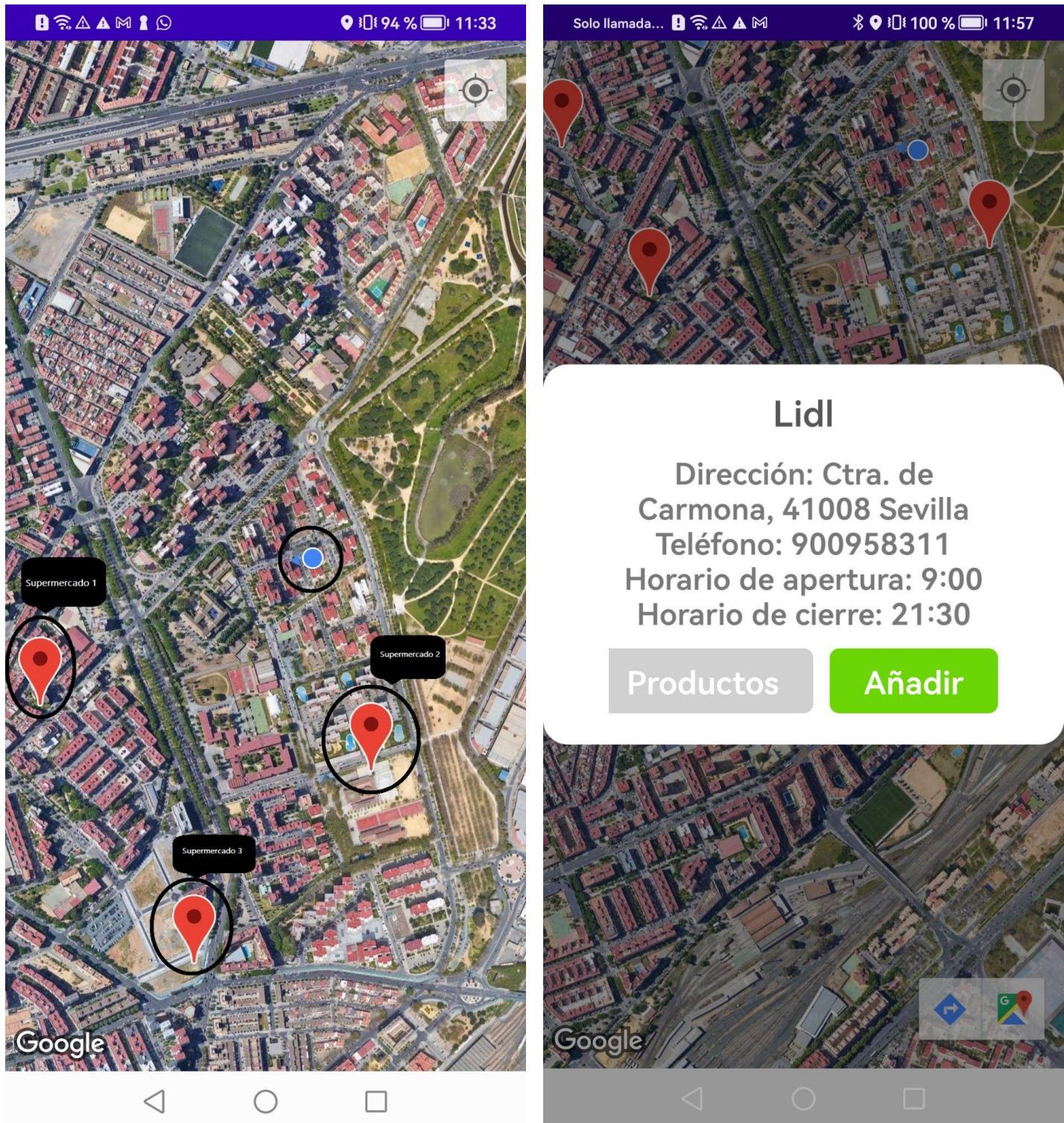
### 3. Ver supermercados cercanos

En caso de que el usuario opte por visualizar los distintos supermercados en las proximidades, tan solo deberá hacer clic en el botón denominado "Mapa", lo que dará lugar a la apertura de la interfaz de mapa que mostrará su ubicación geográfica destacada, así como la presencia de los diversos supermercados en la zona.



### a. Ver información del supermercado

Cuando el usuario pulse sobre uno de los marcadores, le aparecerá una tarjeta con información del supermercado que ha seleccionado.



### b. Añadir producto

Cuando el usuario desee añadir un producto a un supermercado específico, solo tendrá que seleccionar dicho supermercado en el mapa y se aparecerá, además de la información de este, dos botones el cual tendrá que seleccionar “Añadir”.

Una vez pulsado el botón se abrirá una nueva ventana, donde podrá poner un nombre, un precio y una foto del dicho producto

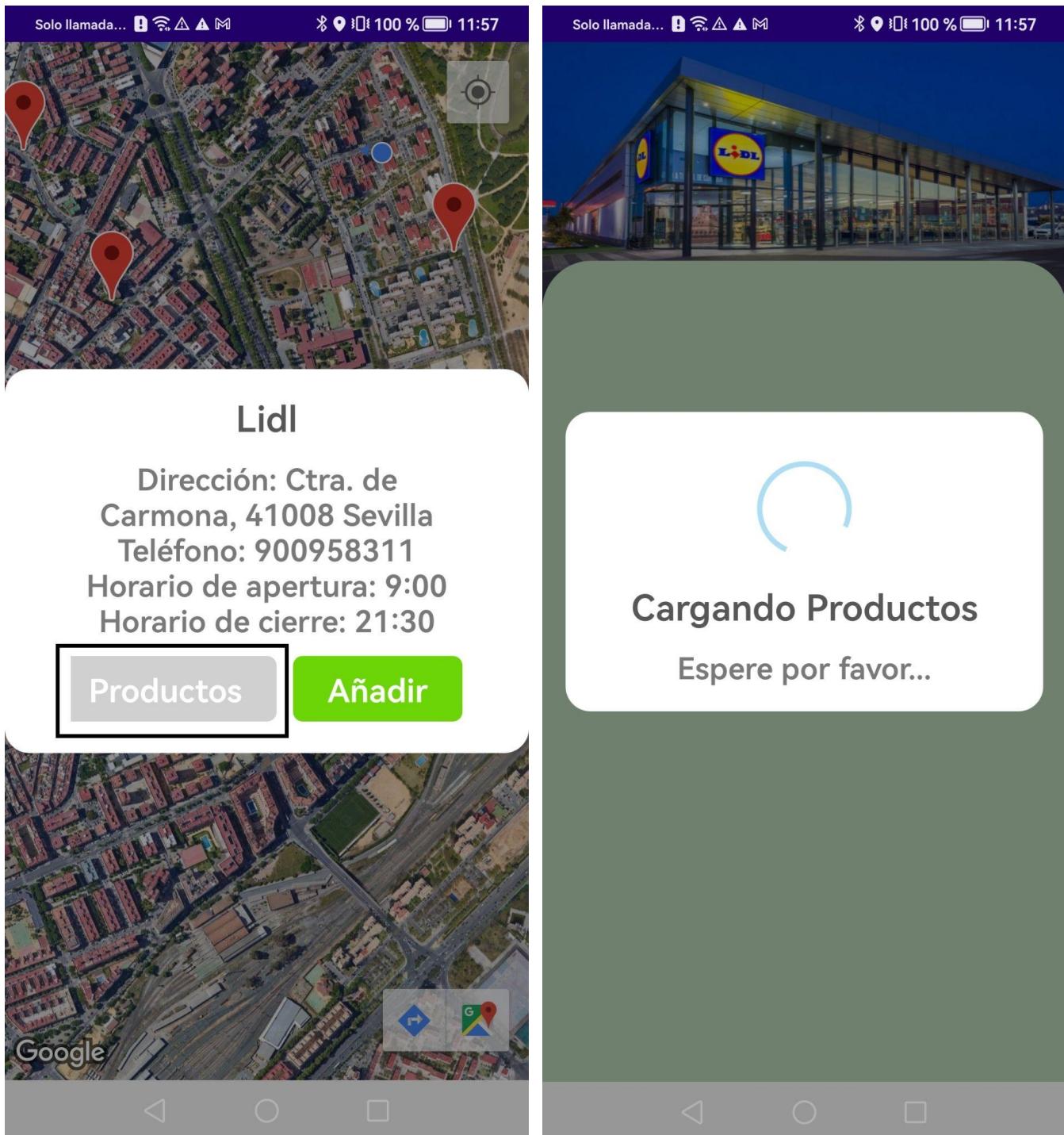




### c. Ver productos

Cuando el usuario desee ver los diferentes productos de un supermercado específico, solo tendrá que seleccionar dicho supermercado en el mapa y se aparecerá, además de la información de este, dos botones el cual tendrá que seleccionar "Productos".

Una vez pulsado el botón se abrirá una nueva ventana, donde podrá ver un listado con los diferentes productos.



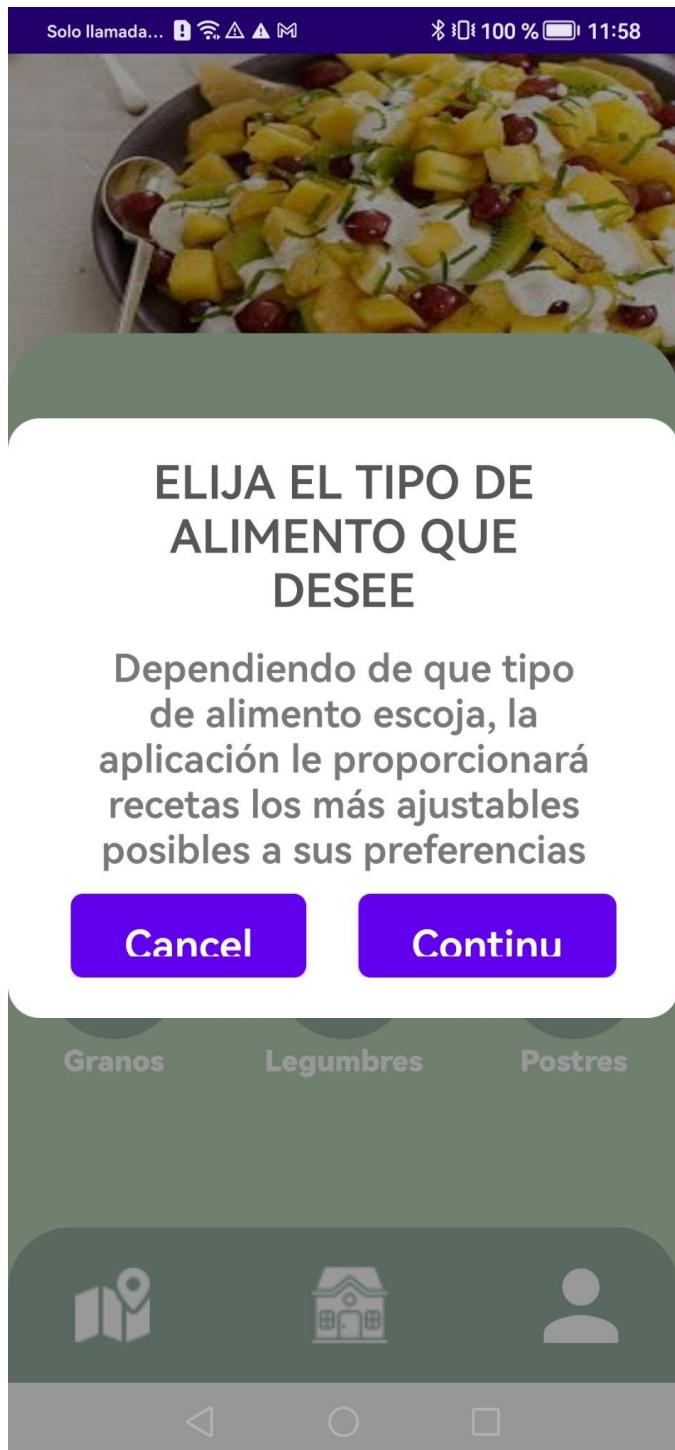


#### 4. Ver recetas

En caso de que el usuario opte por visualizar las distintas recetas que tiene la aplicación, tan solo deberá hacer clic en el botón denominado "Recetas", lo que dará lugar a la apertura de una primera interfaz de elegir el tipo de receta que desea el usuario.

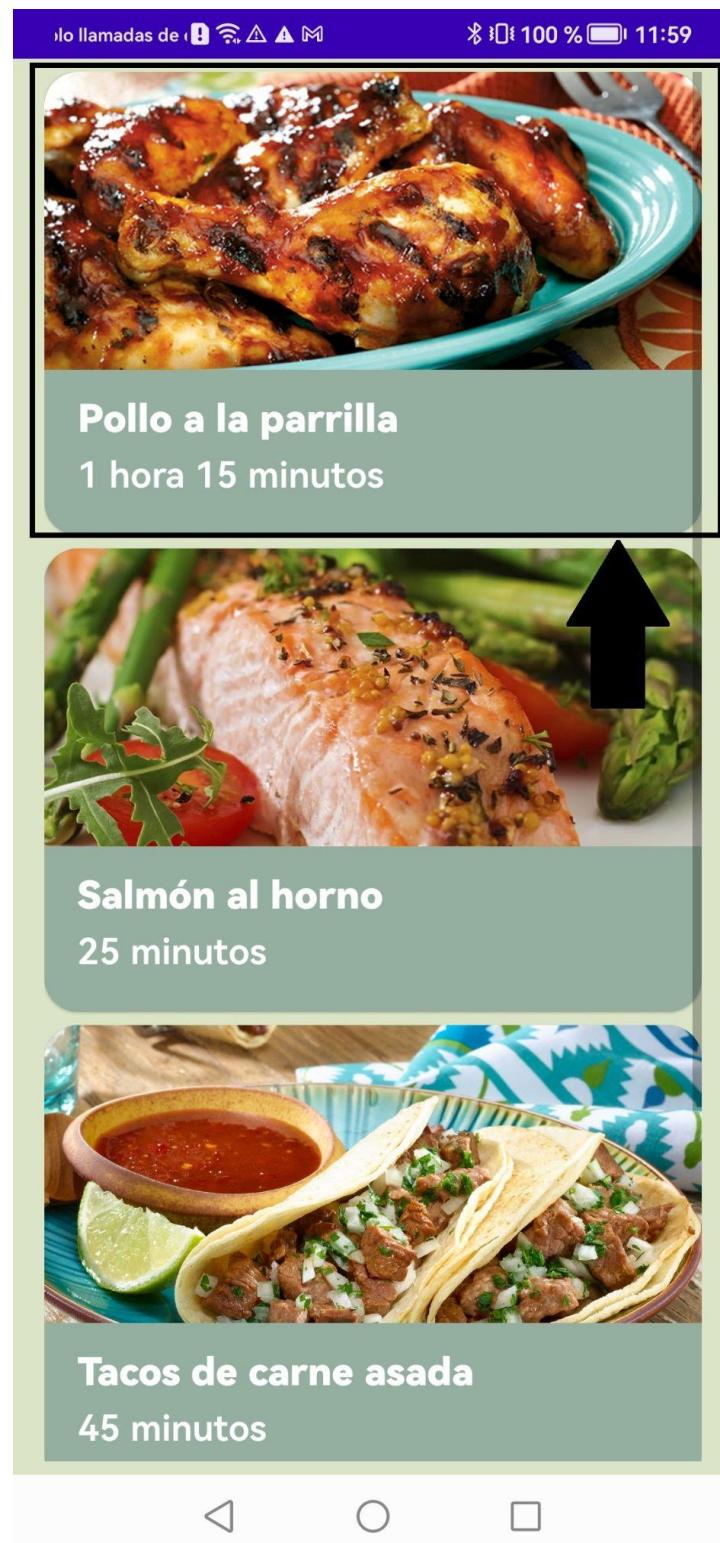


a. Seleccionar el tipo de receta.



Una vez el usuario haya elegido el tipo de receta que desea, le aparecerá una lista con las diferentes recetas.

- b. Seleccionar dicha receta.



c. Ver información para la elaboración de la receta.

**Pollo a la parrilla**

Una deliciosa receta de pollo a la parrilla con hierbas y especias.

Tiempo de preparación  
**1 hora 15 minutos**

Porciones: 4

Ingredientes

- 4 pechugas de pollo
- 1/4 taza de aceite de oliva
- 2 cucharadas de perejil fresco picado
- 1 cucharada de tomillo fresco picado
- 1 cucharada de romero fresco picado
- 2 dientes de ajo picados
- Sal y pimienta al gusto

Solo llamadas de **...** 100 % 11:59

- Sal y pimienta al gusto

## Preparacion

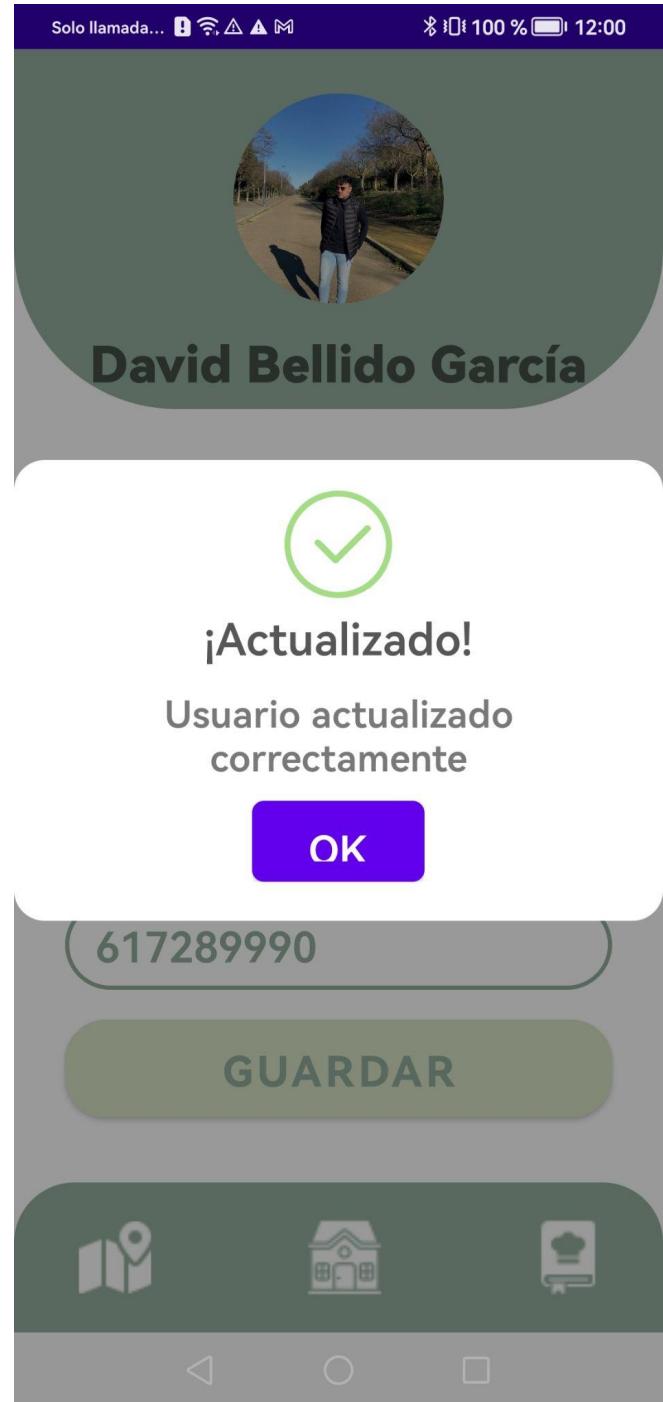
- 1 Mezcla el aceite de oliva, perejil, tomillo, romero, ajo, sal y pimienta en un tazón
- 2 Coloca las pechugas de pollo en la mezcla y asegúrate de cubrirlas completamente
- 3 Marinar en la nevera durante al menos una hora
- 4 Precalentar la parrilla a fuego medio-alto
- 5 Retirar las pechugas de pollo de la marinada y colocarlas en la parrilla
- 6 Cocinar durante 6-8 minutos por lado o hasta que el pollo esté cocido
- 7 Dejar reposar el pollo durante 5 minutos antes de servir.

##### 5. Ver perfil de usuario.

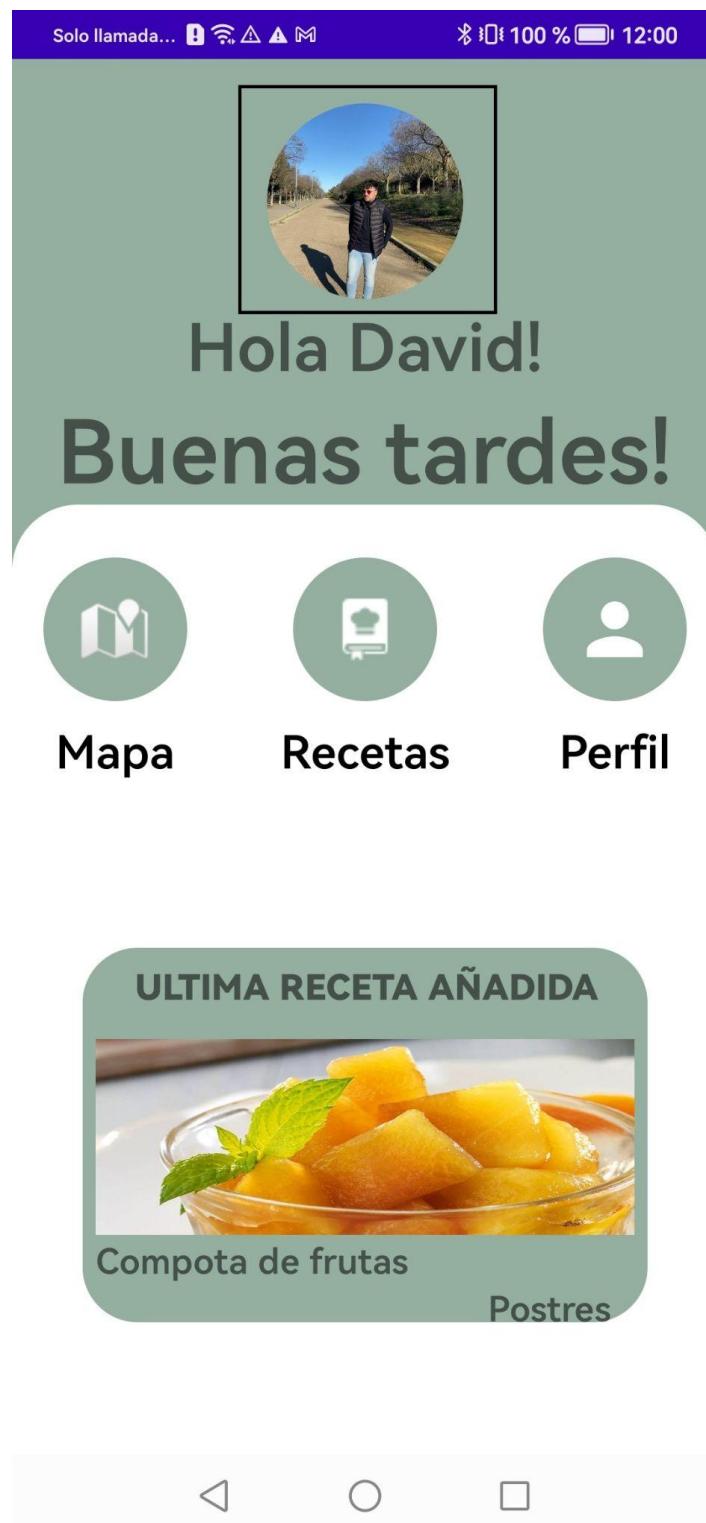
Si el usuario desea ver su perfil o modificarlo, simplemente deberá hacer clic en el botón de “Perfil” de la ventana de “Home” o si se encuentra en otra ventana como receta, pulsando el botón con icono de perfil le llevará a esta ventana donde aparecerá toda su información.

Si el usuario desea modificar su perfil simplemente tendrá que modificar los campos que se muestran y pulsar el botón de guardar.





Ahora cada vez que el usuario entre en la aplicación, aparecerá con su foto de perfil, que anteriormente guardó.

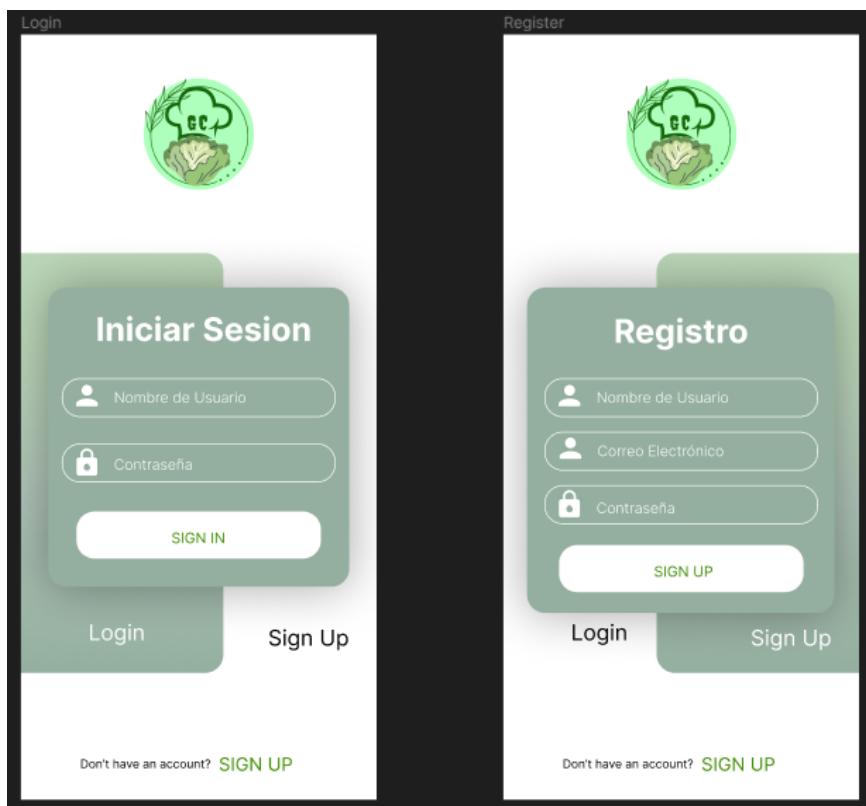


## 4. PROTOTIPO

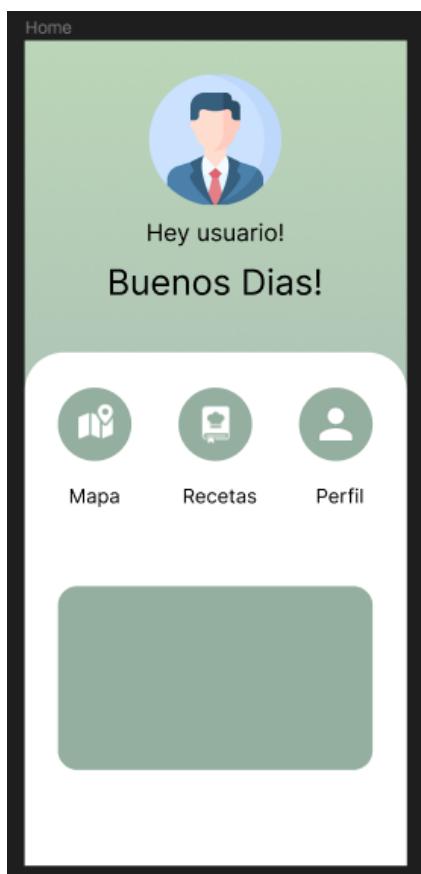
### 4.1 Libro de Estilos



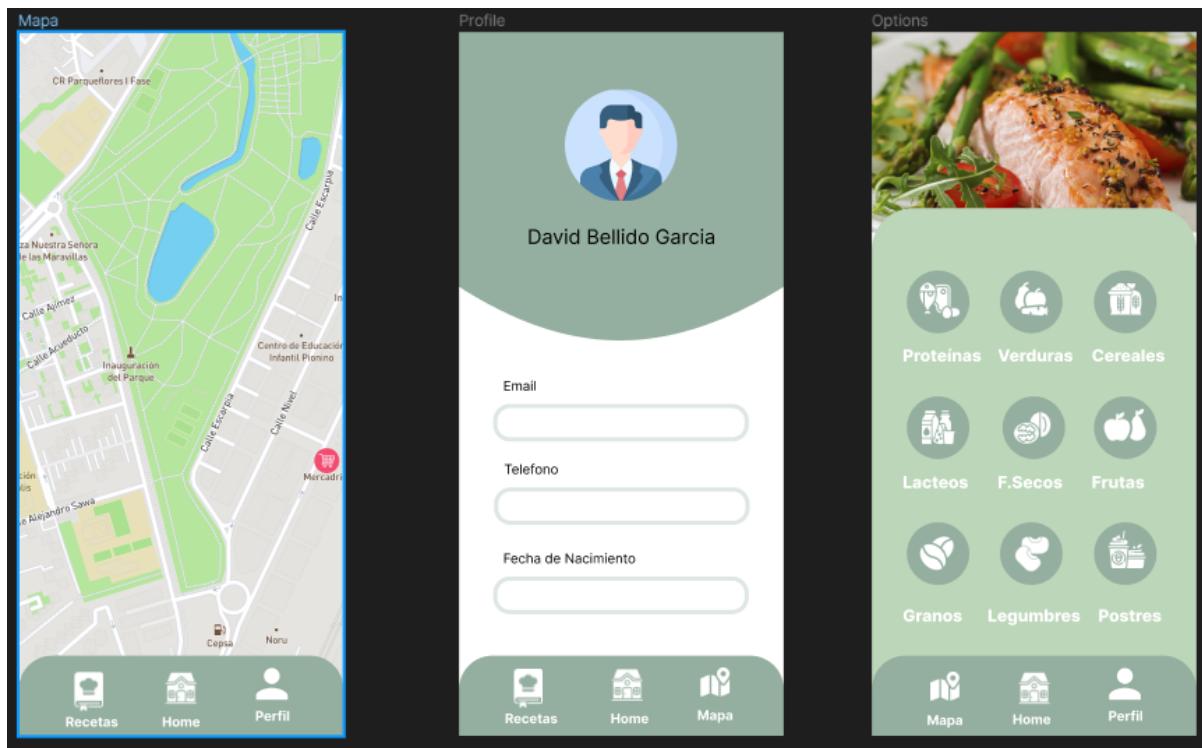
#### 4.2 Login and Register



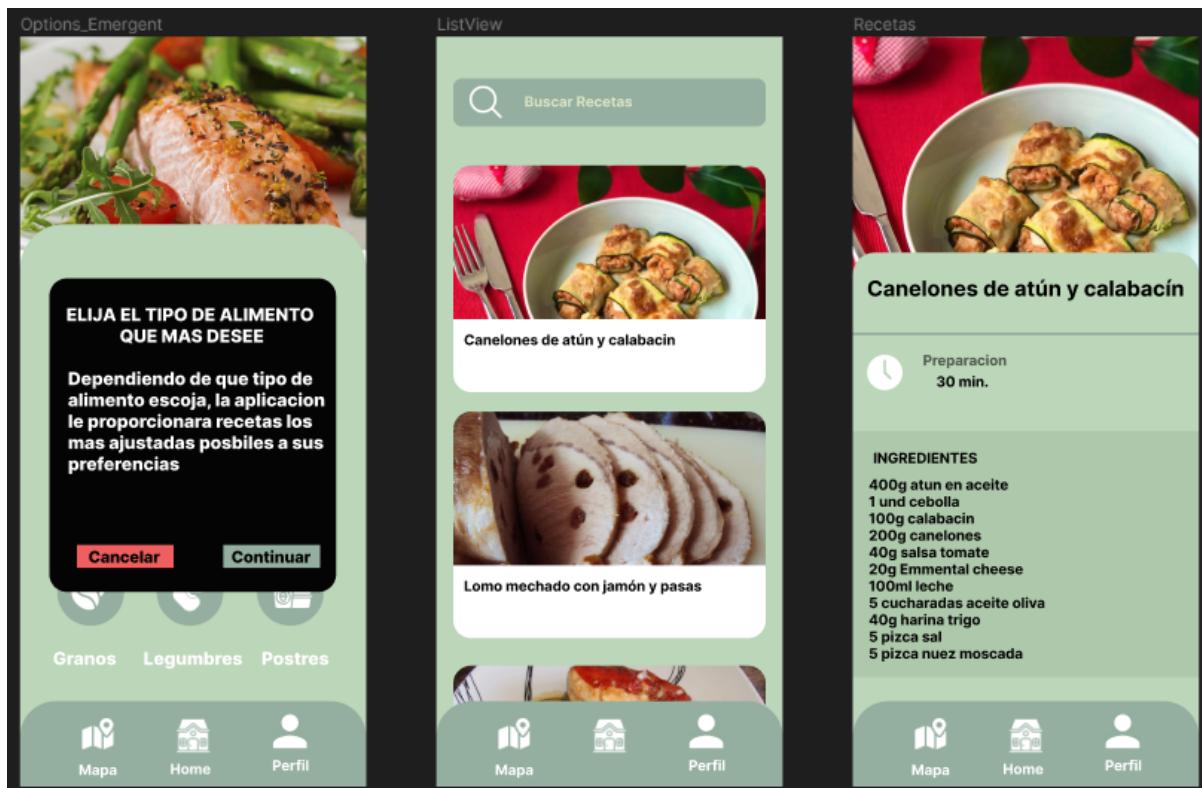
#### 4.3 Home



#### 4.4 Mapa, Recetas y Perfil



#### 4.5 Recetas



## 4.6 Administrador

The image displays three screenshots of a mobile application interface for an administrator, showing the product management screen and a product editing modal.

**Screenshot 1: Product Management Screen**

This screen shows a list of products under the heading "Products". Each item includes a thumbnail image, the product name ("Pollo"), the number of users ("Usuario: 1"), the price ("11.0€"), and the supermarket information ("Supermercado: 1").

Thumbnail	Product	User(s)	Price	Supermarket
	Pollo	Usuario: 1	11.0€	Supermercado: 1
	Pollo	Usuario: 1	11.0€	Supermercado: 1
	Pollo	Usuario: 1	11.0€	Supermercado: 1
	Pollo	Usuario: 1	11.0€	Supermercado: 1

**Screenshot 2: Product Edit Modal**

A modal window titled "Editar Producto" is open, showing the current product details for a selected item. The fields are pre-filled with the values from the first screenshot.

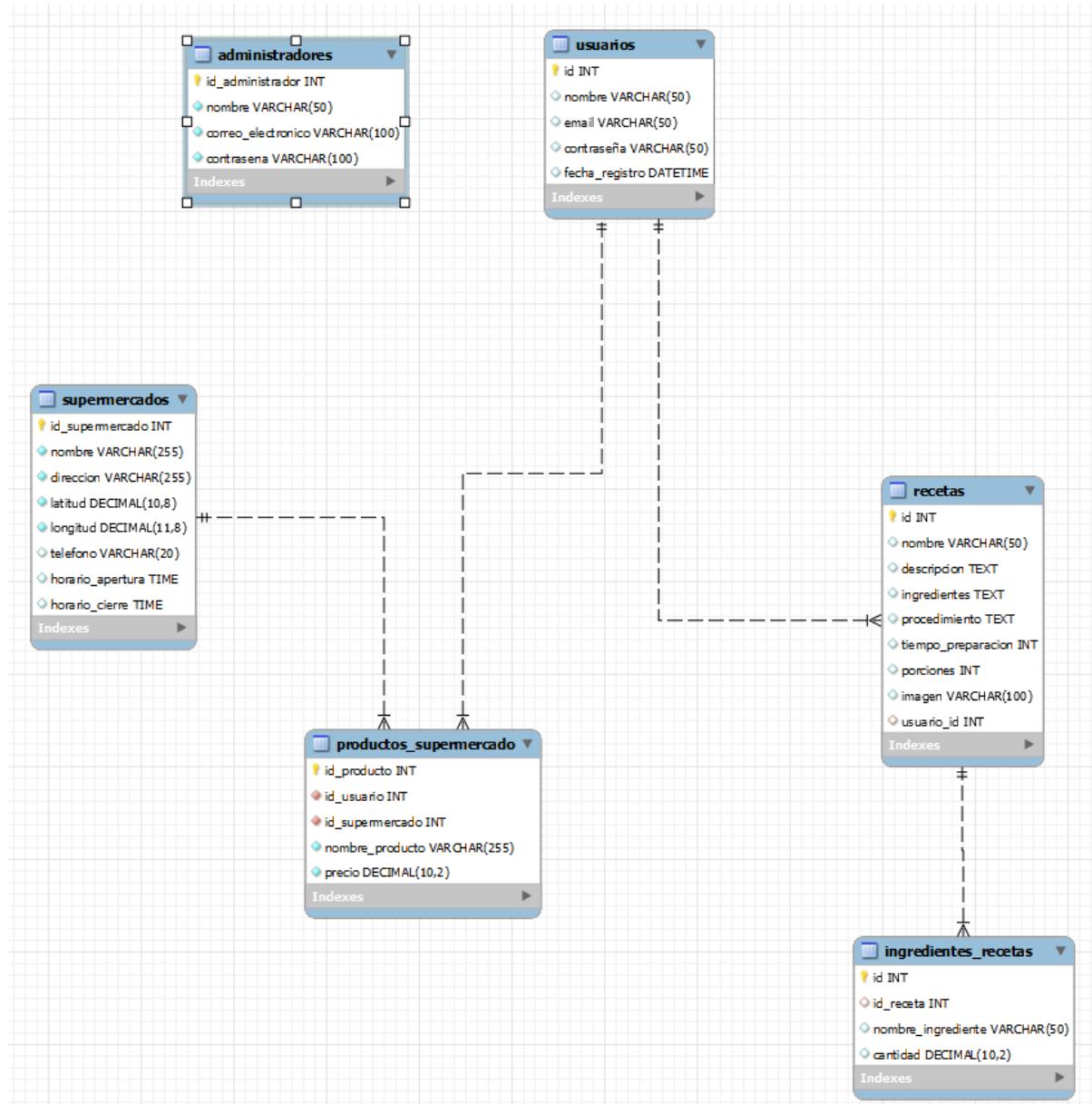
Field	Value
Nombre	Pollo
Cantidad	1
Precio	11.0

**Screenshot 3: Product List (After Edit)**

The product list now reflects the changes made in the edit screen. The fourth item in the list has been updated to show the new quantity (1) and price (11.0€).

Thumbnail	Product	User(s)	Price	Supermarket
	Pollo	Usuario: 1	11.0€	Supermercado: 1
	Pollo	Usuario: 1	11.0€	Supermercado: 1
	Pollo	Usuario: 1	11.0€	Supermercado: 1
	Pollo	Usuario: 1	11.0€	Supermercado: 1

## 5. BASE DE DATOS



## 6. MANUAL TÉCNICO

### 6.1 Conexión con la base de datos MongoDB

Para acceder a la base de datos de MongoDB, el primer paso consiste en dirigirse a la página oficial de la plataforma. Una vez allí, es necesario navegar hasta la sección "Browse Collections" y seleccionar la opción "Create Database" para proceder con la creación de nuestra propia base de datos.

The screenshot shows the MongoDB Atlas interface. On the left, there's a sidebar with various service icons and tabs like Deployment, Database, Services, Security, and Data Lake. The main area is titled 'Database Deployments'. It features a 'Load sample datasets to AtlasCluster.' button and a 'Visualize Your Data' section with charts for R: 0, W: 0, Connections: 0, and network traffic. Below these are tabs for Overview, Real Time, Metrics, and Collections (which is highlighted with a red box). At the bottom, there's a table with cluster details: Version 6.0.6, Region AWS / Ireland (eu-west-1), Cluster Tier M0 Sandbox (General), Type Replica Set - 3 nodes, Backups Inactive, Linked App Services Multiple applications linked, and Atlas SQL and Atlas Search options. The footer includes system status, copyright information, and a feedback icon.

This screenshot shows the same MongoDB Atlas interface as the previous one, but with a different focus. The 'Collections' tab is highlighted in the main navigation bar. The right-hand panel displays a database named 'GreenChef.Admin' with a single document listed: '\_id: ObjectId('6454e6d6ed9cbdc6659fb18'), id: 'admin', log: 'admin', password: 'admin''. Other tabs visible include Overview, Real Time, Metrics, and a 'Find' search bar.

Una vez que la base de datos ha sido creada, es necesario generar un "App Service" para establecer la conexión con nuestra aplicación móvil.

The screenshot shows the MongoDB Atlas interface under the 'App Services' tab. It displays two app services: 'PruebaProyecto' and 'Application-0'. Each service has its own dashboard with metrics like requests, data transfer, and compute runtime. A large blue arrow points upwards from the bottom of the page towards the 'App Services' tab, highlighting it. A callout box at the top left says: 'App Services are free until you exceed the free tier. You can now see the state of your project's total usage as well as the specific usage of each of your apps.' A green button labeled 'Create a New App' is visible on the right.

Al encontrarnos en el "App Service" previamente creado, prestamos atención al valor del "App Id" que nos es proporcionado, ya que resulta crucial para establecer la conexión entre nuestro proyecto y la base de datos.

This screenshot shows the 'PruebaProyecto' app service dashboard. It includes sections for 'DATA ACCESS', 'BUILD', 'MANAGE', and 'HELP'. The 'Metrics' section shows request statistics: Total Requests (0), Success (0/s), and Fail (0/s). A callout box highlights the 'App ID: pruebabaproyecto-umix' field. A green arrow points upwards from the bottom of the page towards the 'App ID' field.

Sin embargo, para permitir la conexión entre nuestro proyecto y la base de datos, es necesario realizar un paso adicional. Nos dirigimos a la sección "Data Access" > "Rules" donde se mostrarán las bases de datos existentes en MongoDB, junto con sus respectivas tablas. En este punto, debemos asignar los permisos adecuados a cada tabla de la base de datos, permitiendo las operaciones de inserción, eliminación y búsqueda, así como la capacidad de lectura y escritura en dichas tablas.

The screenshot shows the MongoDB Atlas interface under the 'App Services' tab, specifically the 'Rules' section for the 'Prueba' app. On the left sidebar, 'DATA ACCESS' is selected, and 'Rules' is highlighted. The main panel displays the 'Collections' section with 1 cluster, 1 database, and 5 collections. A search bar and a 'Show collections without rules' button are present. Below this, the 'mongodb-atlas' database is expanded, showing the 'GreenChef' collection with sub-collections: Admin, Ingredients, Recipes, SuperMarket, SupermarketProducts, and Users. In the center, a 'Roles' section is shown for the 'mongodb-atlas:GreenChef:Admin' role. It includes tabs for 'VIEW' and 'Filters'. Under 'VIEW', it says 'Learn more about Roles | Docs'. A '+ Add role' button is available. A detailed view of the '0. readAndWriteAll' role is shown, with 'Apply When' set to 'Always'. Under 'Document Permissions', 'Insert', 'Delete', and 'Search' are checked. Under 'Field Permissions', 'Read: All' and 'Write: All' are listed. A '... More' button is at the bottom right of the role card.

Este paso es muy importante que lo realicemos, ya que sin él no podremos trabajar en nuestro proyecto aun poniendo el id que nos proporciona.

## 6.2 Conexión a nivel de código

Para poder conectar nuestro proyecto con nuestra base de datos de MongoDB, lo primero de todo es declarar tres variables importantes:

```
public class MainActivity extends AppCompatActivity {
    private TextView txtSignUp;
    private EditText username;
    private EditText password;
    private Button btnInicioSesion;
    private Bundle bundle;
    String AppId = "pruebaproyecto-urnlx";
    MongoDB mongoDatabase;
    MongoClient mongoClient;
```

“AppId” = Id comentado anteriormente de nuestro “AppService” creado.

“mongoDatabase” = atributo para obtener la base de datos

“mongoClient” = atributo para poder obtener el cliente de MongoDB

Para establecer la conexión con la base de datos y, posteriormente, acceder a la colección deseada, es necesario llevar a cabo dos pasos iniciales. En primer lugar, debemos inicializar Realm (AppService) y, a continuación, configurar la aplicación de MongoDB.

```
// Inicialización de Realm y la aplicación de MongoDB
Realm.init(context);
App app = new App(new AppConfigBuilder(AppId).build());

// Inicio de sesión anónimo en MongoDB
Credentials credentials = Credentials.anonymous();
app.loginAsync(credentials, new App.Callback<User>() {
    @Override
    public void onResult(App.Result<User> result) {
        if (result.isSuccess()) {
            // Obtención del cliente y la base de datos de MongoDB
            User user = app.currentUser();
            mongoClient = user.getMongoClient(serviceName: "mongodb-atlas");
            mongoDatabase = mongoClient.getDatabase(databaseName: "GreenChef");
            MongoCollection<Document> mongoCollection = mongoDatabase.getCollection(collectionName: "Users");
```

Como se puede observar, utilizaremos el “AppId” para indicarle cual sera nuestro “AppService” que hemos creado.

A continuación, procederemos a iniciar sesión utilizando las credenciales específicas. En este caso, se utilizará una sesión de inicio anónima para tener acceso completo. Es importante destacar que es posible iniciar sesión con un usuario registrado, aunque esta opción puede limitar algunas funcionalidades. Una vez establecidas las credenciales, se realizará la función app.loginAsync utilizando las credenciales recién creadas, lo cual nos permitirá conectarnos a MongoDB.

Una vez establecida la conexión con MongoDB, el siguiente paso consiste en especificar el cliente, la base de datos y la colección a la cual deseamos conectarnos. Con estos sencillos pasos, lograremos establecer la conexión con nuestra base de datos y la colección deseada.

```
// Inicialización de Realm y la aplicación de MongoDB
Realm.init(context);
App app = new App(new AppConfigurations.Builder(AppId).build());

// Inicio de sesión anónimo en MongoDB
Credentials credentials = Credentials.anonymous();
app.loginAsync(credentials, new App.Callback<User>() {
    @Override
    public void onResult(App.Result<User> result) {
        if (result.isSuccess()) {
            // Obtención del cliente y la base de datos de MongoDB
            User user = app.currentUser();
            mongoClient = user.getMongoClient(serviceName: "mongodb-atlas");
            mongoDatabase = mongoClient.getDatabase(databaseName: "GreenChef");
            MongoCollection<Document> mongoCollection = mongoDatabase.getCollection(collectionName: "Users");
        }
    }
});
```

### 6.3 Operaciones CRUD

Una vez establecida la conexión con la base de datos y la colección correspondiente, para llevar a cabo estas operaciones, se debe realizar una consulta utilizando el método "Document", el cual pertenece a la biblioteca de MongoDB.

```
// Consulta para buscar al usuario en la base de datos
Document query = new Document("nick", usuario).append("password", contrasenia);
```

### 6.3.1 Búsqueda de datos

Para realizar una búsqueda de datos en una colección específica, una vez completados los pasos previos de conexión a la colección y creación de la consulta, se debe utilizar el método "findOne". Este método devuelve un documento que corresponde al primer registro que cumple con los criterios de la consulta.

```
// Consulta para buscar al usuario en la base de datos
Document query = new Document("nick", usuario).append("password", contraseña); // Buscar al usuario por su nick y contraseña
mongoCollection.findOne(query).getAsync(result1 -> {
    if (result1.isSuccess()) {
        Document usuario = result1.get();
        if (usuario != null) {
            // Si el usuario existe, se inicia la actividad OptionsActivity
            Intent i = new Intent( packageContext: MainActivity.this, OptionsActivity.class);
            i.putExtras(bundle);
            MainActivity.this.startActivity(i);
        } else {
            // Si el usuario no existe, se muestra un mensaje de error
            new SweetAlertDialog( context: MainActivity.this, SweetAlertDialog.ERROR_TYPE)
                .setTitleText("Oops...")
                .setContentText("Usuario o contraseña incorrectos")
                .show();
        }
    } else {
        // Si hay un error en la búsqueda del usuario, se muestra un mensaje de error
        new SweetAlertDialog( context: MainActivity.this, SweetAlertDialog.ERROR_TYPE)
            .setTitleText("Oops...")
            .setContentText("Error al buscar usuario")
            .show();
    }
});
```

### 6.3.2 Inserción de datos

Para realizar la inserción de datos en nuestra colección, el procedimiento es similar a la búsqueda de datos, pero con algunas diferencias. Antes de insertar los datos, es necesario realizar una consulta donde se especifican los atributos de la colección y los valores que se desean introducir.

```
Document usuario = new Document();
usuario.append("id_usuario", id+1)
    .append("nick", nick)
    .append("email", email)
    .append("password", contrasenia)
    .append("fecharegistro", fechaRegistro)
    .append("nombre", "")
    .append("apellidos", "")
    .append("telefono", "");
```

Una vez creada la estructura de inserción de datos, llevaremos a cabo una operación similar a la búsqueda, pero en lugar de utilizar el método "findOne", emplearemos el método "insertOne".

```
mongoCollection.insertOne(usuario).getAsync(result1 -> {
    if (result1.isSuccess()) {
        new SweetAlertDialog(RegisterActivity.this, SweetAlertDialog.SUCCESS_TYPE)
            .setTitleText("¡Registro exitoso!")
            .setContentText("¡Bienvenido a GreenChef!")
            .setConfirmText("Volver")
            .setConfirmClickListener(new SweetAlertDialog.OnSweetClickListener() {
                @Override
                public void onClick(SweetAlertDialog sDialog) {
                    RegisterActivity.this.finish();
                }
            })
            .show();
    }
});
```

### 6.3.3 Actualización de datos

Para realizar la actualización de datos en nuestra colección, el procedimiento es igual que para la inserción de datos. Antes de actualizar los datos, es necesario realizar una consulta donde se especifican los atributos de la colección y los valores que se desean introducir. En lugar de utilizar “insertOne” utilizaremos “updateOne”

```
// Crear el objeto con los campos que se van a actualizar
Document actualizaciones = new Document()
    .append("id_usuario", id)
    .append("nick", txtNick.getText().toString())
    .append("email", txtEmail.getText().toString())
    .append("password", password)
    .append("fecharegistro", fecharegistro)
    .append("nombre", txtNombre.getText().toString())
    .append("apellidos", txtApellido.getText().toString())
    .append("telefono", txtTelefono.getText().toString())
    .append("img", imagen);

// Actualizar el documento en MongoDB
mongoCollection.updateOne(filtro, actualizaciones).getAsync(resul -> {
    if (resul.isSuccess()) {
        // Mostrar un mensaje de éxito
        final SweetAlertDialog dialogo = new SweetAlertDialog(context: UserActivity.this, SweetAlertDialog.PROGRESS_TYPE)
            .setTitleText("Actualizando")
            .setContentText("Espere por favor...");
    }
});
```

### 6.3.4 Eliminación de datos

Para realizar la eliminación de datos es mucho más sencillo que todos los anteriores. Simplemente primero tendremos que realizar una búsqueda de datos para encontrar el objeto que deseamos eliminar y posteriormente utilizaremos “deleteOne”.

```
// Crear el filtro para buscar el producto por su ID
Document filtro = new Document().append("id", productId);
RealmResultTask<MongoCursor<Document>> findTask = mongoCollection.find(filtro).iterator();

findTask.getAsync(task -> {
    if (task.isSuccess()) {
        MongoCursor<Document> results = task.get();
        if (results.hasNext()) {

            // Eliminar el documento de MongoDB
            mongoCollection.deleteOne(filtro).getAsync( result -> {
                if (result.isSuccess()) {
                    Log.d( tag: "ListProductActivity", msg: "Producto eliminado de MongoDB");
                } else {
                    Log.d( tag: "ListProductActivity", msg: "Producto no eliminado de MongoDB");
                }
            });
        }
    }
});
```