

CasualChef



Unión Europea

Fondo Social Europeo
"El FSE invierte en tu futuro"



INSTITUTO DE EDUCACIÓN SECUNDARIA SERPIS

CasualChef

Proyecto de Desarrollo de Aplicaciones Multiplataforma

Ciclo Formativo Desarrollo de Aplicaciones Multiplataforma

Autor: Jose Enrique Sanchis Hueso

Tutor: Maria Angeles, Chover Miñana

Curso: 2022/2023



Resumen

La aplicación CasualChef es una herramienta de gestión de recetas que se conecta a una base de datos Firebase en línea para obtener información sobre las recetas, las preferencias de la aplicación y los datos de los usuarios. Se les permite a los usuarios crear, borrar y modificar recetas, también personalizar su perfil y cambiar los colores de los contenidos de las recetas para dar total libertad al usuario. Es importante destacar que sólo los usuarios registrados tienen acceso a crear recetas, mientras que los usuarios anónimos sólo pueden visualizarlas. CasualChef cuenta con una función de filtros para buscar recetas por diferentes criterios, como ingredientes, dificultad, etiquetas, descripción y condiciones especiales. Esta función es útil para los usuarios con restricciones dietéticas o que buscan recetas específicas basadas en los ingredientes disponibles. Además, la función de filtros también puede ser beneficiosa para aquellos que buscan recetas adecuadas para su nivel de habilidad culinaria. En resumen, CasualChef ofrece una amplia variedad de funcionalidades para la gestión de recetas, proporcionando una experiencia de usuario personalizada y fácil de usar. Con su función de filtros, los usuarios pueden encontrar rápidamente las recetas que se ajustan a sus preferencias y necesidades alimentarias. En general, es una excelente opción para aquellos que buscan una aplicación de gestión de recetas en línea.

Abstract

CasualChef is an innovative software program that facilitates the organization of recipes by connecting to an online database for recipe details, app settings, and user data. The software incorporates advanced features for registration, login, and logout purposes, while offering users the ability to create, modify, and delete recipes. The primary screen showcases recipes grouped by the creator and the general public, and it allows for recipe searches based on multiple criteria. Additionally, users have the freedom to configure their profiles and change the colors of the software interface, though the updates will solely be noticeable in the recipe contents. It is worth mentioning that recipe creation is limited to registered users, and anonymous users are solely allowed to view recipes. The Firebase database contains three collections - recipes, app preferences, and user data, while storage is used to safeguard images of each recipe and user profiles. One of the unique features of CasualChef is its cutting-edge filter function, which enables users to explore recipe options based on name, tags, description, ingredients, difficulty, and special dietary requirements like low calorie or vegetarian. The filter feature is especially handy for people with dietary limitations or those seeking specific recipes based on the ingredients they have. Additionally, the filter function is also useful for those who want recipes that match their cooking skills. To sum up, CasualChef is a sophisticated recipe management tool that provides a variety of innovative features for its users. It offers personalized profile settings, recipe creation and modification, and multiple criteria recipe searches, all in an easy-to-use interface. With its advanced filtering, users can quickly find recipes that meet their needs and preferences. Overall, CasualChef is an excellent choice for anyone in need of an effective online recipe management solution.

Resum

L'aplicació CasualChef és una eina de gestió de receptes que es connecta a una base de dades Firebase en línia per a obtenir informació sobre les receptes, les preferències de l'aplicació i les dades dels usuaris. Es permet als usuaris crear, esborrar i modificar receptes, així com personalitzar el seu perfil i canviar els colors del contingut de les receptes per donar total llibertat a l'usuari. És important destacar que només els usuaris registrats tenen accés a crear receptes, mentre que els usuaris anònims només poden visualitzar-les. CasualChef compta amb una funció de filtres per a buscar receptes per diferents criteris, com ara ingredients, dificultat, etiquetes, descripció i condicions especials. Aquesta funció és útil per als usuaris amb restriccions dietètiques o que busquen receptes específiques basades en els ingredients disponibles. A més, la funció de filtres també pot ser beneficiosa per a aquells que busquen receptes adequades per al seu nivell d'habilitat culinària. En resum, CasualChef ofereix una àmplia varietat de funcionalitats per a la gestió de receptes, proporcionant una experiència d'usuari personalitzada i fàcil d'utilitzar. Amb la seva funció de filtres, els usuaris poden trobar ràpidament les receptes que s'ajusten a les seves preferències i necessitats alimentàries. En general, és una excel·lent opció per a aquells que busquen una aplicació de gestió de receptes en línia.

Índice

Resumen	2
Índice	5
Índice de imágenes	6
Justificación	7
Objetivos a cumplir	8
Gestión del proyecto	9
Metodología	10
Herramientas utilizadas	10
Descripción del proyecto	11
Análisis	12
Requisitos funcionales	12
Diagrama de casos de uso (UML)	13
Requisitos no funcionales	22
Diseño	23
Diagrama Entidad-Relación (E-R)	30
Diagramas de clases (UML)	31
Implementación	32
Dependencias	32
Desarrollo	34
Pruebas	43
Documentación	45
Trabajos futuros	45
Conclusiones	46
Bibliografía y webgrafía	46
Anexos	46

Índice de imágenes

Imagen 1: Esquema ganntt -	Página 9
Imagen 2: Commit 18 -	Página 10
Imagen 3: Diagrama de casos de uso -	Página 13
Imagen 4: Diagrama de casos de uso Iniciar sesión -	Página 14
Imagen 5: Diagrama de casos de uso Registrarse -	Página 15
Imagen 6: Diagrama de casos de uso Editar información -	Página 16
Imagen 7: Diagrama de casos de uso Buscar por filtros -	Página 17
Imagen 8: Diagrama de casos de uso Editar receta -	Página 18
Imagen 9: Diagrama de casos de uso Borrar receta -	Página 19
Imagen 10: Diagrama de casos de uso Crear receta -	Página 20
Imagen 11: Diagrama de casos de uso Cambiar colores -	Página 21
Imagen 12: Interfaz de Registro -	Página 23
Imagen 13: Interfaz de Login -	Página 23
Imagen 14: Interfaz de Menú Principal -	Página 24
Imagen 15: Interfaz de Menú Principal, Menú hamburguesa -	Página 24
Imagen 16: Interfaz Filtros -	Página 25
Imagen 17: Interfaz Detalle -	Página 25
Imagen 18: Interfaz Crear -	Página 26
Imagen 19: Interfaz Editar -	Página 26
Imagen 20: Interfaz Editor Perfil -	Página 27
Imagen 21: Interfaz Perfil -	Página 27
Imagen 22: Interfaz Editor Preferencias Color -	Página 28
Imagen 23: ColorPicker -	Página 28
Imagen 24: Interfaz de Menú Principal Anónimo -	Página 29
Imagen 25: Interfaz de Menú Principal Anónimo Menú hamburguesa -	Página 29
Imagen 26: Diagrama Entidad-Relación -	Página 30
Imagen 27: Diagramas de clases -	Página 31
Imagen 28: Boceto básico del proyecto -	Página 34
Imagen 29: AppBar Menú Principal -	Página 34
Imagen 30: Forma de evadir email/contraseña Login y Registro -	Página 35
Imagen 31: Método para cargar recetas desde json Receta -	Página 35
Imagen 32: Imágenes en storage de firebase -	Página 37
Imagen 33: Código para poner los filtros Filtro -	Página 37
Imagen 34: Procesado de los filtros aplicados RecetasFragmentFiltro -	Página 37
Imagen 35: ColorPicker enseñado al editar Preferencias_Interfaz -	Página 38
Imagen 36: Forma de enseñar el color al usuario Preferencias_Interfaz -	Página 38
Imagen 37: Botones de editar y borrar de la receta del usuario Detalle -	Página 39
Imagen 38: Datos extensos de una receta Detalle -	Página 39

Imagen 39: Receta al haber apretado editar | EditarReceta - **Página 39**

Imagen 40: Listener usado para utilizar 2 digitos | EditarReceta - **Página 40**

Imagen 41: Código usado al apretar el botón| EditarReceta - **Página 40**

Imagen 42: Detección de uso de imagen | EditarReceta - **Página 40**

Imagen 43: Menú principal anónimo con el menú hamburguesa abierto - **Página 41**

Imagen 44: Menú de filtros sin conexión a internet - **Página 41**

Imagen 45: Detalle de sin conexión a internet - **Página 41**

Imagen 46: Datos usuario sin conexión a internet - **Página 42**

Imagen 47: Código de glide en la búsqueda de imágenes | Utils - **Página 42**

Imagen 48: Código de Glide para buscar | MyRecyclerViewAdapter - **Página 43**

Justificación

La gastronomía es una parte importante de la cultura y la sociedad actual, y cada vez son más las personas interesadas en el mundo de la cocina. A pesar de que existen numerosos programas de televisión, revistas y blogs dedicados a la gastronomía, no existe una plataforma que permita a los usuarios compartir sus propias creaciones de forma fácil y accesible.

Es por ello que se ha desarrollado esta aplicación de creación de recetas, que tiene como objetivo fomentar la comunidad de amantes de la cocina y permitir que los usuarios compartan sus creaciones de forma intuitiva y eficiente. Gracias a tecnologías actuales como Kotlin y Firebase, se ha podido crear una aplicación fácil de usar y con múltiples herramientas para crear y compartir recetas.

El objetivo principal de la aplicación es ofrecer un acceso rápido y sencillo a los usuarios, eliminando la necesidad de registro y permitiendo el acceso en modo anónimo para mejorar la experiencia de uso. Esto hace que la aplicación sea más fácil de manejar y más accesible, lo que puede ampliar su público objetivo y aumentar su popularidad y retención de usuarios.

Ahora justificaré el uso de las tecnologías utilizadas en mi proyecto:

Kotlin: Es un lenguaje de programación con múltiples aplicaciones, y su uso permite aplicar los conocimientos adquiridos en clase para llevar a cabo este proyecto con mejor eficiencia.

Firebase: Es una plataforma en la nube que tiene a disposición un número de herramientas que de las cuales se han utilizado las suficientes para poder guardar imágenes, guardar datos y autenticar cada usuario que quiera registrarse o iniciar sesión en la aplicación.

Se ha utilizado esta base de datos en específico debido a que es una base de datos NoSQL y por lo tanto es más rápida al enviar datos ya que mi aplicación necesita obtenerlos con rapidez.

Objetivos a cumplir

1. Se podrá registrar un usuario y podrá iniciar sesión con esa cuenta.
2. Se podrá iniciar sesión como anónimo para todos aquellos que no tengan interés en crear nuevas recetas.
3. Todos los usuarios podrán acceder al menú de filtros para poder buscar las recetas que quieran específicamente.
4. Se podrán crear recetas por cualquier usuario que haya iniciado sesión.
5. Se podrán editar recetas por cualquier usuario que haya iniciado sesión.
6. Se podrán borrar recetas por cualquier usuario que haya iniciado sesión.
7. Cualquier usuario que haya iniciado sesión podrá tener acceso a su propio menú de configuración de interfaz del detalle de las recetas.
8. Cualquier usuario que haya iniciado sesión podrá tener acceso a su propio menú de configuración de perfil para modificar y para que lo puedan ver el resto de usuarios.

Gestión del proyecto

La gestión y planificación del proyecto se ha creado como un elemento clave para garantizar el éxito del desarrollo de la aplicación.

Se ha considerado primordial llevar a cabo una organización rigurosa y anticipar posibles desafíos para abordarlos de manera eficaz.

Con este propósito, se ha efectuado una valoración estimativa de la temporalización y la planificación del proyecto, con el fin de establecer los plazos y objetivos a cumplir. En este proceso, se han tomado en consideración los recursos disponibles, las funcionalidades a implementar y las limitaciones del proyecto para dar forma a una planificación inicial coherente.

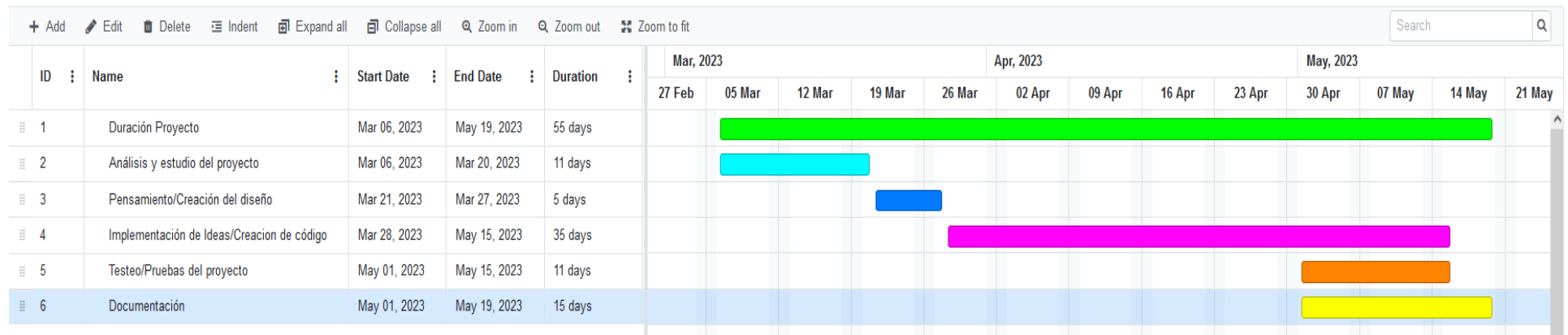


Imagen 1: Esquema gantt

Metodología

Se han ido guardando todos los avances del proyecto en Github

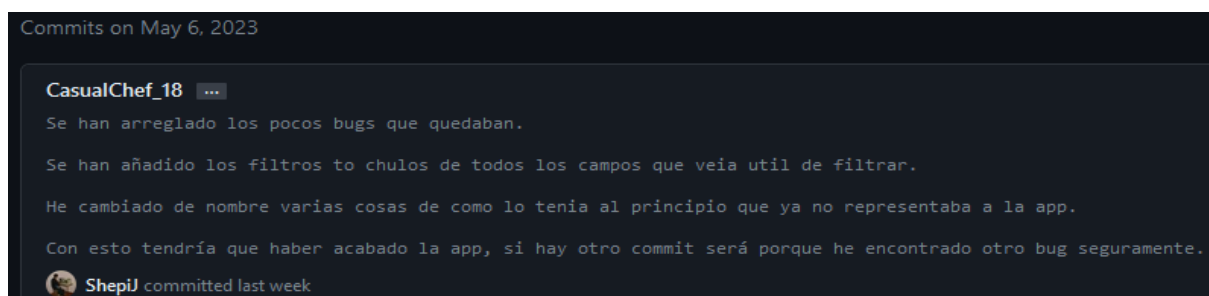
Enlace al proyecto: <https://github.com/ShepiJ/CasualChef>

Se ha utilizado de manera de que a partir de haber hecho la base del proyecto con un recyclerview que funcionara y que tuviera conexión a firebase aunque el único ejemplo fuera el inicio de sesión había comenzado a poner versiones en el git.

El 90% de los commits son funcionales, y se pueden ejecutar desde un sistema propio ya que se había decidido hacer commits cuando la aplicación fuera estable para evitar problemas relacionados con eso.

La mayoría de los commits han sido escritos con buenas descripciones excepto raros casos.

Imagen 2: Commit 18



Herramientas utilizadas

Se han utilizado las siguientes tecnologías:

Github: Se ha utilizado github para la gestión de versiones y para guardar los datos del proyecto en la nube.

Enlace al proyecto: <https://github.com/ShepiJ/CasualChef>

Android Studio: Se ha utilizado android studio debido a que es la herramienta que se ha utilizado siempre para tratar con aplicaciones de android.

Drawio.io: Se ha utilizado Draw.io debido a que se ha utilizado con anterioridad y tiene una amplia extensión de herramientas para hacer esquemas.

Firebase: Se ha utilizado Firebase para guardar los datos de cada usuario en la base de datos para poder recibirla en cualquier momento.

Descripción del proyecto

CasualChef es una aplicación para la gestión de recetas que requiere estar conectada a internet para acceder a una base de datos que proporciona información sobre las recetas, las preferencias de la aplicación y los datos personales de cada usuario.

La aplicación incluye funcionalidades para iniciar sesión, registrarse y cerrar sesión. La pantalla principal muestra las recetas clasificadas por el usuario que las ha creado y las que han sido creadas por otros usuarios. Además, ofrece la posibilidad de crear, borrar y modificar recetas (esta última funcionalidad era opcional), y de acceder a la información completa de cada receta, incluyendo ingredientes, etiquetas, nombre y fotografía. La búsqueda de recetas por nombre, etiquetas y creador también está disponible.

Los usuarios pueden configurar su perfil y cambiar los colores de la aplicación aunque solo se verá el cambio en el contenido de las recetas. Es importante destacar que solo los usuarios registrados pueden crear recetas, mientras que los usuarios anónimos sólo pueden visualizarlas.

La base de datos Firebase incluye tres colecciones: recetas, preferencias de la aplicación y datos de los usuarios. Además, se utiliza el almacenamiento para guardar las imágenes de cada receta y los perfiles de los usuarios.

Además de las funcionalidades mencionadas, CasualChef también cuenta con una función de filtros que permite a los usuarios buscar recetas por nombre, etiquetas, descripción, ingredientes, dificultad y condiciones especiales, como pocas calorías o vegetariano. Esto permite a los usuarios encontrar rápidamente las recetas que se ajustan a sus preferencias y necesidades alimentarias.

La función de filtros puede ser especialmente útil para aquellos usuarios que tienen restricciones dietéticas, ya que pueden filtrar las recetas según sus necesidades específicas. También puede ser útil para aquellos que buscan recetas basadas en los ingredientes que tienen a mano o que están buscando recetas de dificultad adecuada para su nivel de habilidad culinaria.

Análisis

Requisitos funcionales

1. Iniciar Sesión

Los usuarios registrados podrán iniciar sesión y en el proceso si es su primera vez se les crearán datos predefinidos para su perfil y colores de recetas que ya podrán editar en cualquier momento.

También se puede iniciar sesión como anónimo para evitar crear una cuenta o para acceder sin internet.

2. Registro

Los usuarios se podrán registrar en firebase sin problemas mientras que el nombre de usuario y la contraseña sean válidas para el authentication de firebase.

3. Buscar recetas por filtros.

Los usuarios podrán buscar las recetas que quieran con este filtro que buscará entre todas las que haya y ya el usuario podrá cambiar cada campo del filtrador para encontrar lo que desee.

Se puede filtrar por Nombre de campo:

(tags,nombre,autor,ingredientes,descripción)

Y por nivel de dificultad de la receta y por condiciones como que una receta sea vegetariana o que sea apta para diabéticos.

4. Editar información del perfil del usuario.

Aquí cada usuario comienza con todos los campos en default excepto el usuario pero todo se puede cambiar. Hay que mencionar que todos los usuarios pueden ver tu perfil mientras que hayas creado una receta por lo menos ya que hay que darle click al usuario dentro de la receta para poder acceder al perfil no editable.

5. Editar receta propia.

Con una receta ya creada el usuario puede acceder a editarla y en ese menú todo se podrá cambiar.

Además por comodidad se rellenan todos los campos con los que tenía la receta que se quiere editar para no perder tiempo.

6. Crear receta propia.

Aquí cada usuario puede crear una nueva receta, la única condición para poder guardarla es rellenar todos los campos o sino no se permitirá.

7. Borrar receta propia.

En la información de una receta si la receta es del usuario que ha iniciado sesión entonces podrá darle al botón de borrar, una vez se le da a borrar no se puede recuperar la receta.

8. Cambiar configuración de colores de los detalles de las recetas

Aquí podemos customizar el color de la información de todas las recetas pero es solo para el usuario que ha iniciado sesión. Todos los campos aquí también comienzan siendo los predeterminados y se dictarán cuales son dependiendo si el usuario tiene el móvil en modo oscuro o no.

Diagrama de casos de uso (UML)

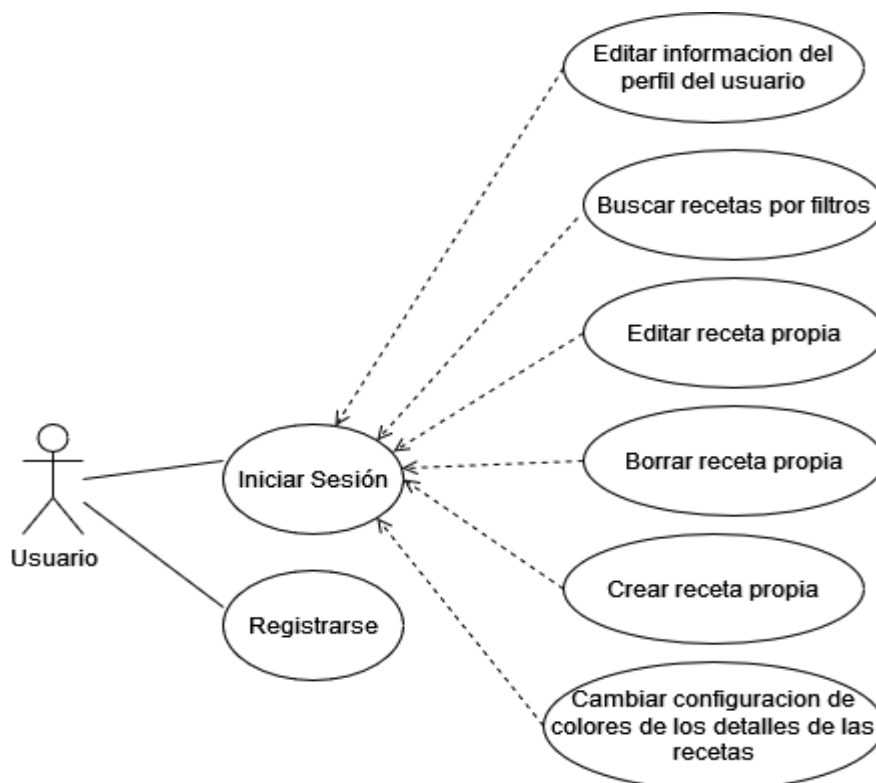


Imagen 3: Diagrama de casos de uso

1. Iniciar sesión

Caso de Uso	Iniciar Sesión		
Actores	Usuario		
Tipo	Principal		
Referencias	Nombre de usuario y contraseña		
Precondición	Aplicación en funcionamiento		
Postcondición	Se inicia la sesión		
Autor	Jose Enrique Sanchis Hueso		
Propósito			
El usuario inicia sesión			
Resumen			
El usuario pone su usuario y su contraseña y si existen en el servidor accederá a su cuenta			
Curso Normal			
1	El usuario pone su usuario y contraseña	1	
2		2	Se comprueba si el usuario existe
3		3	Si el usuario no tiene datos de usuario o de interfaz se crean nuevos en la base de datos
4	Se permite acceder a la cuenta del usuario	4	
5		5	
6		6	
7		7	
8		8	
9		9	
10		10	
Cursos Alternos			
1a	El usuario pone su contraseña o usuario mal		
2a	El usuario no tiene acceso a internet y no se puede buscar si el usuario existe		
3a			
Otros Datos			
Frecuencia esperada	Siempre que se inicie la aplicación	Rendimiento	Adecuado
Importancia	Alta	Urgencia	Alta
Estado	Hecho	Estabilidad	Buena
Comentarios			
Aparte de poder iniciar sesión con usuario y contraseña puedes iniciar sesión como anónimo pero no podrás crear recetas			
Tiene una función también de mantenerse conectado para no tener que poner el usuario y la contraseña siempre, solo cuando cierres sesión			

Imagen 4: Diagrama de casos de uso | Iniciar sesión

2. Registrarse

Caso de Uso	Registrarse		
Actores	Usuario		
Tipo	Principal		
Referencias	Nombre de usuario y contraseña		
Precondición	Aplicación en funcionamiento		
Postcondición	Se registra el usuario y inicia la sesión		
Autor	Jose Enrique Sanchis Hueso		
Propósito			
El usuario se registra			
Resumen			
El usuario se registra poniendo su usuario y contraseña deseados y la base de datos mira si existe ya ese usuario o no			
Curso Normal			
1	El usuario pone su usuario y contraseña	1	
2		2	Se comprueba si el usuario existe y si el nuevo usuario cumple todas las normativas para crear
3	Se crea la cuenta y te manda a la actividad de Iniciar sesión	3	
4		4	
5		5	
6		6	
7		7	
8		8	
9		9	
10		10	
Cursos Alternos			
1a	El usuario que se intentaba poner ya existía		
2a	El usuario no tiene acceso a internet y no se puede crear el nuevo usuario		
3a	El usuario ha puesto una contraseña que no cumple las normas de la base de datos		
Otros Datos			
Frecuencia esperada	Una vez por usuario	Rendimiento	Adecuado
Importancia	Alta	Urgencia	Alta
Estado	Hecho	Estabilidad	Buena
Comentarios			

Imagen 5: Diagrama de casos de uso | Registrarse

3. Editar información del perfil del usuario

Caso de Uso	Editar informacion del perfil del usuario		
Actores	Usuario		
Tipo	Principal		
Referencias	Los datos que se quieran modificar		
Precondición	Se ha iniciado sesión correctamente		
Postcondición	Se ha hecho el cambio al perfil correctamente		
Autor	Jose Enrique Sanchis Hueso		
Propósito			
El usuario cambia la informacion que tenia en su perfil			
Resumen			
El usuario cambia la informacion que tenia en su perfil como su foto de perfil, nombre y apellidos reales, gmail, etc.			
Curso Normal			
1	El usuario cambia los datos que desea cambiar	1	Se guardan los nuevos datos en la base de datos para poder verlos por ti y por otros
2		2	
3	El usuario puede ver como los datos se han cambiado	3	
4		4	
5		5	
6		6	
7		7	
8		8	
9		9	
10		10	
Cursos Alternos			
1a	El usuario no tiene acceso a internet y no se pueden guardar los cambios		
2a			
3a			
Otros Datos			
Frecuencia esperada	No muy a menudo	Rendimiento	Adecuado
Importancia	Media	Urgencia	Media
Estado	Pendiente	Estabilidad	Buena
Comentarios			
También aparte de ver tu configuración de perfil puedes ver la de los demás apretando el nombre del creador de la receta en rojo en detalles aunque no puedes editarlas de esa forma			

Imagen 6: Diagrama de casos de uso | Editar información

4. Buscar recetas por filtros

Caso de Uso	Buscar recetas por filtros		
Actores	Usuario		
Tipo	Principal		
Referencias	Por que se quiere filtrar y el valor de lo que se quiere mirar		
Precondición	Se ha iniciado sesión correctamente		
Postcondición	Se puede navegar por los resultados de filtros que se han seleccionado		
Autor	Jose Enrique Sanchis Hueso		
Propósito			
El usuario puede buscar recetas por los filtros que se quieran			
Resumen			
El usuario pone por que se va a buscar como nombre, creador, tag, ingredientes, dificultad, condiciones de la receta y luego van a salir por pantalla las recetas que cumplan los requerimientos			
Curso Normal			
1	El usuario selecciona los 2 valores necesarios para filtrar	1	
2		2	Se filtra el json de las recetas solo enseñando lo deseado
3	El usuario puede ver las recetas con los filtros aplicados	3	
4		4	
5		5	
6		6	
7		7	
8		8	
9		9	
10		10	
Cursos Alternos			
1a	El usuario ha puesto un valor que no se ha encontrado en el json por lo que la lista se queda vacía		
2a			
3a			
Otros Datos			
Frecuencia esperada	A menudo	Rendimiento	Adecuado
Importancia	Alta	Urgencia	Alta
Estado	Por acabar	Estabilidad	Buena
Comentarios			
Opino que es lo que mejor y lo más complicado que he hecho en el proyecto porque la forma en la que hago toda la búsqueda es un poco rara con sharedPrefs			

Imagen 7: Diagrama de casos de uso | Buscar por filtros

5. Editar receta propia

Caso de Uso	Editar receta propia		
Actores	Usuario		
Tipo	Principal		
Referencias	Los datos que se quieran modificar de la receta		
Precondición	Se ha iniciado sesión correctamente		
Postcondición	La receta se ha modificado correctamente		
Autor	Jose Enrique Sanchis Hueso		
Propósito			
El usuario puede modificar/editar una receta que ya ha creado			
Resumen			
El usuario tiene la posibilidad de editar la receta que había creado con el proposito de arreglar errores en la receta o por si el usuario simplemente quiere cambiar algo			
Curso Normal			
1	El usuario elige editar una receta y le da al botón de editar	1	
2		2	La aplicación pasa a un menú de edicion
3		3	Se rellenan todos los campos con la informacion que tenia en ellos para no escribir la receta de nuevo
4	El usuario cambia los datos que desea cambiar	4	
5		5	Se guardan los nuevos datos en la base de datos
6	El usuario puede ver los cambios hechos la receta	6	
7		7	
8		8	
9		9	
10		10	
Cursos Alternos			
1a	El usuario no tiene acceso a internet y no se pueden guardar los cambios		
2a			
3a			
Otros Datos			
Frecuencia esperada	A menudo	Rendimiento	Adecuado
Importancia	Alta	Urgencia	Alta
Estado	Pendiente	Estabilidad	Buena
Comentarios			
El usuario solo podrán modificar/editar las recetas que le pertenezcan a su cuenta			

Imagen 8: Diagrama de casos de uso | Editar receta

6. Borrar receta propia

Caso de Uso	Borrar receta propia		
Actores	Usuario		
Tipo	Principal		
Referencias	Que receta se desea borrar		
Precondición	Se ha iniciado sesión correctamente		
Postcondición	La receta se ha borrado correctamente		
Autor	Jose Enrique Sanchis Hueso		
Propósito			
El usuario puede borrar una receta			
Resumen			
Si un usuario decide borrar alguna receta por alguna razón le damos la opción de poder hacerlo			
Curso Normal			
1	El usuario elige la receta que desea borrar y le da al botón de borrar	1	
2		2	Se guardan los nuevos datos en la base de datos
3	El usuario puede ver que la receta ya no está	3	
4		4	
5		5	
6		6	
7		7	
8		8	
9		9	
10		10	
Cursos Alternos			
1a	El usuario no tiene acceso a internet y no se pueden guardar los cambios		
2a			
3a			
Otros Datos			
Frecuencia esperada	Pocas veces	Rendimiento	Adecuado
Importancia	Alta	Urgencia	Alta
Estado	Pendiente	Estabilidad	Buena

Imagen 9: Diagrama de casos de uso | Borrar receta

7. Crear receta propia

Caso de Uso	Crear receta propia		
Actores	Usuario		
Tipo	Principal		
Referencias	Los datos que se quieran aplicar en la receta		
Precondición	Se ha iniciado sesión correctamente		
Postcondición	Se ha creado una nueva receta		
Autor	Jose Enrique Sanchis Hueso		
Propósito			
El usuario puede crear una nueva receta			
Resumen			
Esta aplicación trata de guardar recetas y poder visualizarlas, las tuyas y las de los demás por lo que es un elemento clave			
Curso Normal			
1	El usuario le da al botón de añadir receta	1	
2		2	La aplicación pasa a un menú de añadir receta
3	El usuario rellena todos los campos y le da a guardar la receta	3	
4		4	Se guardan los nuevos datos en la base de datos
5	El usuario puede ver que se ha creado la nueva receta	5	
6		6	
7		7	
8		8	
9		9	
10		10	
Cursos Alternos			
1a	El usuario no tiene acceso a internet y no se pueden guardar los cambios		
2a	El usuario no ha rellenado todos los campos y por lo tanto no permite guardar los datos		
3a			
Otros Datos			
Frecuencia esperada	Muy a menudo	Rendimiento	Adecuado
Importancia	Alta	Urgencia	Alta
Estado	Hecho	Estabilidad	Buena
Comentarios			
Da igual si algunas recetas tienen el mismo nombre ya que eso va por ids y no se van a repetir			

Imagen 10: Diagrama de casos de uso | Crear receta

8. Cambiar configuración de colores de los colores de las recetas

Caso de Uso	Cambiar configuración de colores de los colores de las recetas		
Actores	Usuario		
Tipo	Principal		
Referencias	Que colores seleccionar para cada apartado		
Precondición	Se ha iniciado sesión correctamente		
Postcondición	Los cambios se aplican		
Autor	Jose Enrique Sanchis Hueso		
Propósito			
El usuario puede modificar la interfaz gráfica de la aplicación a su querer			
Resumen			
El usuario puede seleccionar de una lista de colores cada color de cada componente en la interfaz gráfica			
Curso Normal			
1	El usuario cambia los colores que desee cambiar	1	
2		2	Se guardan los nuevos datos en la base de datos
3	Me meto en el contenido de una receta	3	
4		4	Nada más cargar obtiene los colores de la base de datos
5	Puedo ver los colores que había configurado para que salieran	5	
6		6	
7		7	
8		8	
9		9	
10		10	
Cursos Alternos			
1a	El usuario no tiene acceso a internet y no se pueden guardar los cambios		
2a			
3a			
Otros Datos			
Frecuencia esperada	Muy poco	Rendimiento	Adecuado
Importancia	Baja	Urgencia	Media
Estado	Pendiente	Estabilidad	Buena
Comentarios			
Al principio había intentado que fuera un cambio de color de todo el proyecto pero era muy complicado o simplemente imposible sin rehacer el proyecto con muchos plugins tal vez			
Al final he optado por solo cambiar el interior de la receta ya que era lo más cercano a lo que quería			
Aunque no se pueda cambiar todo he aplicado un modo oscuro/claro dependiendo de que modo tengas en tu móvil la interfaz de toda la app cambiará			

Imagen 11: Diagrama de casos de uso | Cambiar colores

Requisitos no funcionales

1. Seguridad

La aplicación debe de tener seguridad entre usuarios para proteger cada cuenta, y utilizo la función authentication de firebase para cumplir un mínimo de seguridad de contraseña.

2. Usabilidad

Se había pensado que a lo mejor el usuario no tenía interés en crear recetas y no quería forzar a nadie a crearse una cuenta y acordarse de su nombre de cuenta/contraseña por lo que he implementado la posibilidad de poder iniciar sesión como anónimo para poder ver las recetas.

3. Rendimiento

Gracias a firebase siendo una base de datos no relacional permite que las búsquedas a la base de datos sean más rápidas y por lo tanto tarde menos al descargarla en la app en su inicio. Además como

4. Customización

La aplicación tiene una gran cantidad de opciones para cambiar tu perfil/colores/recetas por lo que el usuario desee personalizar.

Diseño

1. Registro



The registration screen features a header with the text "No se olvidará de esta experiencia!". Below this is the CasualChef logo, which consists of a circular emblem with a spiral and the text "CASUALCHEF" repeated twice. The form includes two input fields: "Usuario" and "Contraseña", each with a horizontal line below the label. At the bottom, there is a grey button labeled "REGISTRARSE".

En esta pantalla el usuario podrá registrar una cuenta para la aplicación.

Estos datos se enviarán a la base de datos mediante autenticación para que el usuario pueda iniciar sesión con su usuario.

También hay un usuario especial:

Anónimo es el usuario que se utiliza para entrar de forma anónima y no se puede registrar nadie como ese nombre.

Imagen 12: Interfaz de Registro

2. Iniciar Sesión



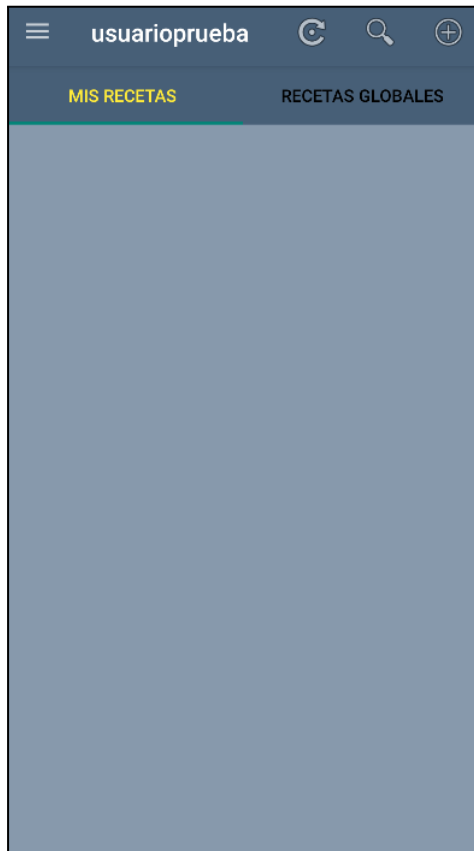
The login screen features a header with a chef hat icon, the text "Bienvenido", and a checkbox labeled "Recordar". Below this is the CasualChef logo. The form includes two input fields: "Usuario" and "Contraseña", each with a horizontal line below the label. At the bottom, there are two grey buttons: "INICIAR SESIÓN" and "REGISTRARSE".

En esta ventana el usuario podrá iniciar sesión con sus credenciales o de modo anónimo haciendo click sobre el icono de la esquina superior izquierda. Además dispondrá de una casilla checkbox para recordar la sesión en el siguiente inicio de la aplicación.

Si el usuario no tiene cuenta, tendrá acceso a un botón en donde navegará hacia la ventana de registro para crear un usuario.

Imagen 13: Interfaz de Login

3. Menú Principal

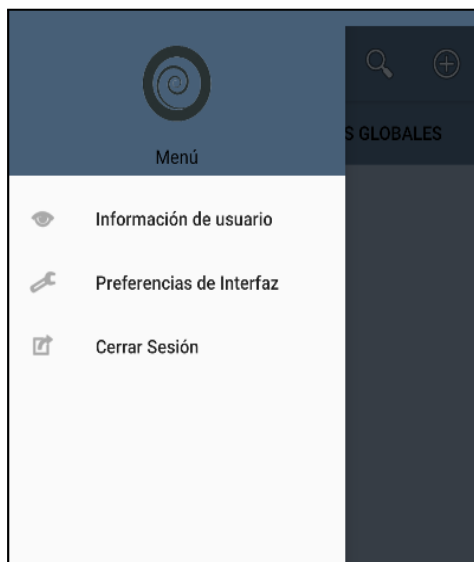


En esta ventana el usuario podrá ver las recetas que ha creado él y el resto de usuarios y solo se puede acceder aquí si se ha iniciado sesión.

También podrán acceder al detalle de las recetas al hacer click sobre la receta deseada.

El menú de opciones en la parte superior derecha permite acceder al menú de filtros, añadir una receta y poder reiniciar la lista de recetas.

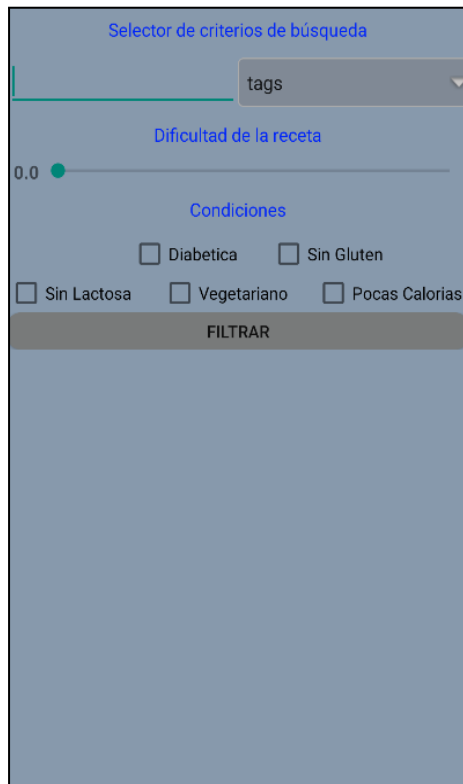
Imagen 14: Interfaz de Menú Principal



Al apretar en el menú hamburguesa podremos acceder a poder cambiar el aspecto del perfil de usuario, el aspecto del detalle de la receta y también se puede cerrar sesión para poder cambiar de usuario.

Imagen 15: Interfaz de Menú Principal, Menú hamburguesa

4. Filtros



Filtros es la ventana que utilizamos para filtrar todas las recetas guardadas en la base de datos, podemos filtrar por los siguientes campos.



Y que contengan el texto puesto en el campo elegido.

También se puede filtrar por nivel de dificultad y condiciones varias.

Imagen 16: Interfaz Filtros

5. Detalle Receta



Al acceder en esta ventana la aplicación carga los datos de esa receta con su dato del JSON y descarga su imagen.

Si la receta es del usuario que ha iniciado sesión en el dispositivo aparecerán los botones de editar y borrar y si no será visible y no se podrá interactuar con ellos.

Aquí podemos ver los datos de la receta en su totalidad y además le podemos dar click al nombre del autor para poder ver su perfil

También si el usuario ha iniciado sesión entonces los colores seleccionados en Editor de preferencias de color se cargaran nada más entrar.

Imagen 17: Interfaz Detalle

6. Creador de recetas

En esta ventana podemos crear nuestras recetas y tenemos que rellenar todos los campos para poder enviarla.

Los campos son:

1. Nombre de receta
2. La imagen de la receta
3. Ingredientes de la receta
4. Tags de la receta
5. Condiciones
7. Tiempo de preparación
8. Dificultad de la receta.

Al rellenarlo todo podemos darle a crear subimos los nuevos datos a la base de datos y descargamos el nuevo JSON con la receta incluida

Imagen 18: Interfaz Crear

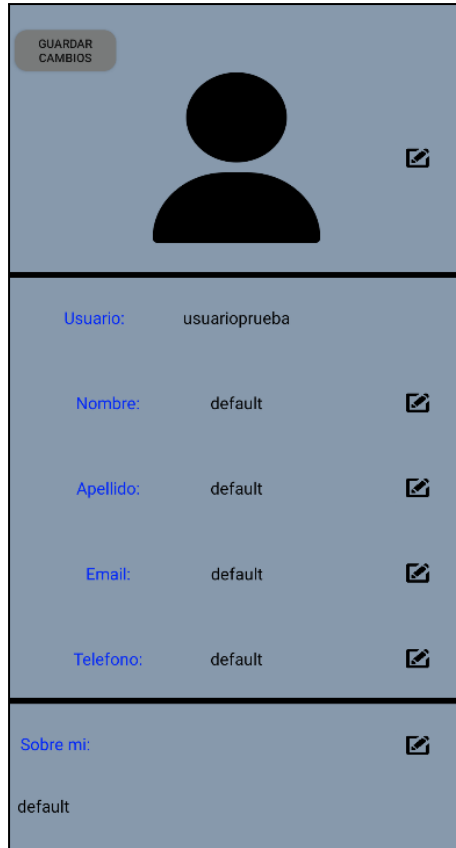
7. Editor de recetas

Esta ventana es idéntica a la de crear recetas pero se accede al darle click en detalle a editar en detalle.

Lo especial de esta ventana es que al entrar se rellenarán todos los campos incluido la imagen con los datos que tenga la receta seleccionada para proveer al usuario una experiencia más cómoda para cambiar los campos que se quieran cambiar.

Imagen 19: Interfaz Editar

8. Editor de Perfil



GUARDAR CAMBIOS

Usuario: usuarioprueba

Nombre: default

Apellido: default

Email: default

Telefono: default

Sobre mi:

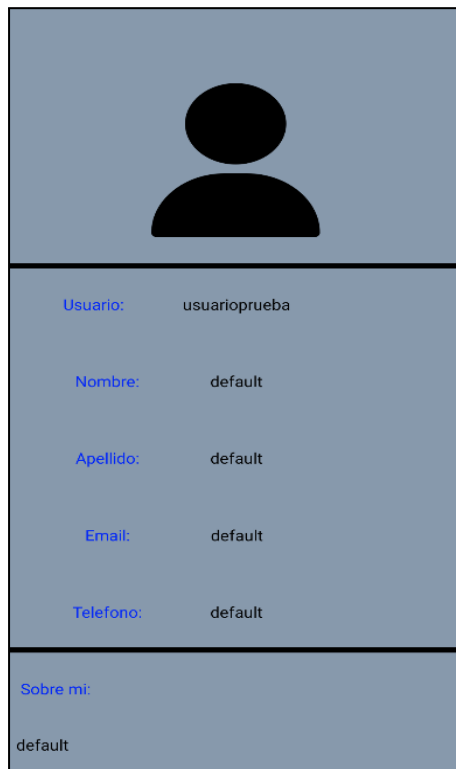
default

En esta ventana nada más entrar se cargan los datos del usuario con el que hayamos iniciado sesión.

La primera vez que se inicie sesión se pone todo como se ve en la captura, todos los campos están rellenos con default excepto el usuario y está hecho con el propósito de cambiarlo todo a el gusto de cada usuario.

Imagen 20: Interfaz Editor Perfil

9. Perfil Usuario



Usuario: usuarioprueba

Nombre: default

Apellido: default

Email: default

Telefono: default

Sobre mi:

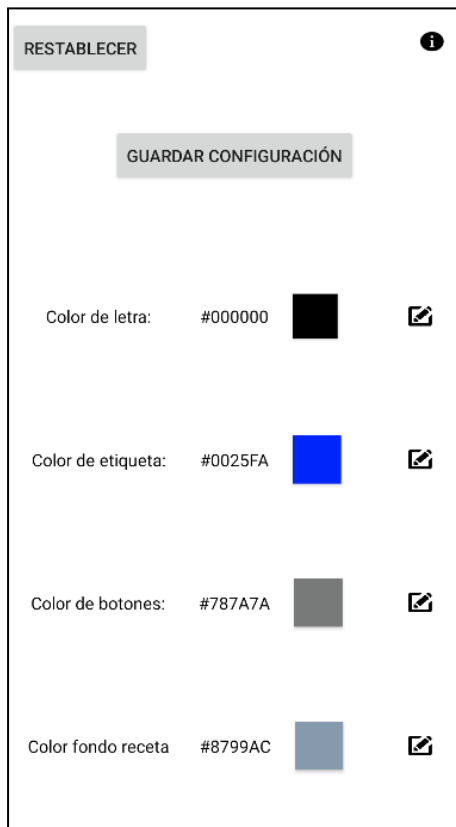
default

Esta venta es idéntica a la del editor de perfil pero solo se puede acceder al hacerle click al nombre del autor en detalle.

Como Editor de Perfil nada más entrar se cargan los datos del usuario en cuestión con la única diferencia de que no se puede editar nada.

Imagen 21: Interfaz Perfil

10. Editor de preferencias de color



En esta pantalla podemos seleccionar los colores que queremos ver en las recetas nuestras y las de los demás.

Podemos cambiar el color al que queramos con la ayuda del ColorPicker que he implementado en mi proyecto y cuando se seleccione se enseñará el valor hexadecimal del color en su zona y se cambiará el color de al lado se cambiará de color también.

También igual que como muchas ventanas de esta aplicación al entrar se cargarán todos los colores seleccionados del usuario.

Imagen 22: Interfaz Editor Preferencias Color

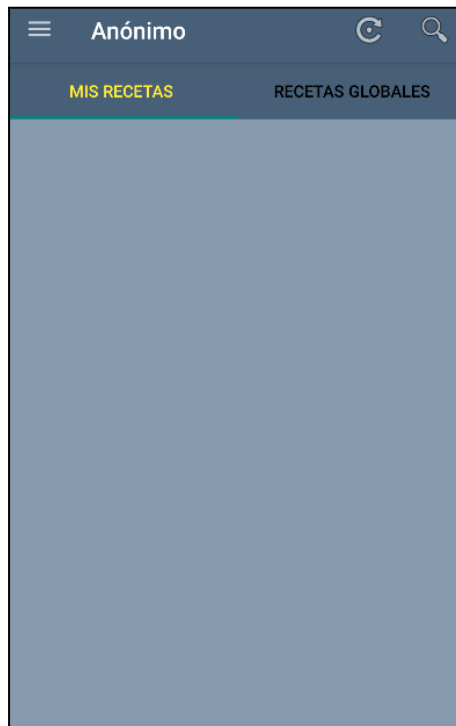


Se ha tenido en cuenta cuando el usuario ha editado mucho el programa hasta el punto de ser difícil identificar cada campo por lo que dependiendo de si el usuario está en modo oscuro o no se pondrán unos colores predeterminados que quedan bien con la interfaz general.

Igual que cuando al iniciar sesión se creaban datos predeterminados para cada usuario también se crean datos predeterminados para los colores dependiendo si el dispositivo está en modo oscuro o no.

Imagen 23: ColorPicker

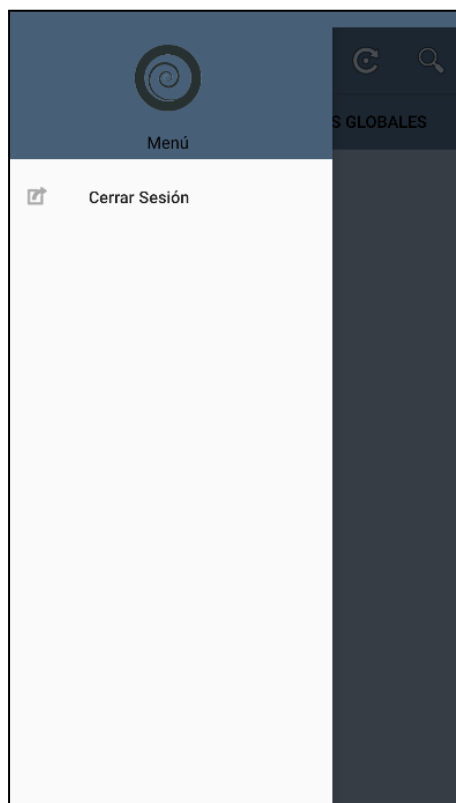
11. Menú Principal Anónimo



Esta ventana igual que el menú principal de los usuarios que han iniciado sesión es la versión exclusiva de los usuarios que no quieren iniciar sesión y han decidido entrar como usuarios anónimos pero tiene un par de cambios.

La función de filtro y la de recarga de lista sigue existiendo pero no está la opción de crear recetas.

Imagen 24: Interfaz de Menú Principal Anónimo



También el menú hamburguesa está cambiado para no solo tener la opción de cerrar sesión.

Imagen 25: Interfaz de Menú Principal Anónimo |Menú hamburguesa

Diagrama Entidad-Relación (E-R)

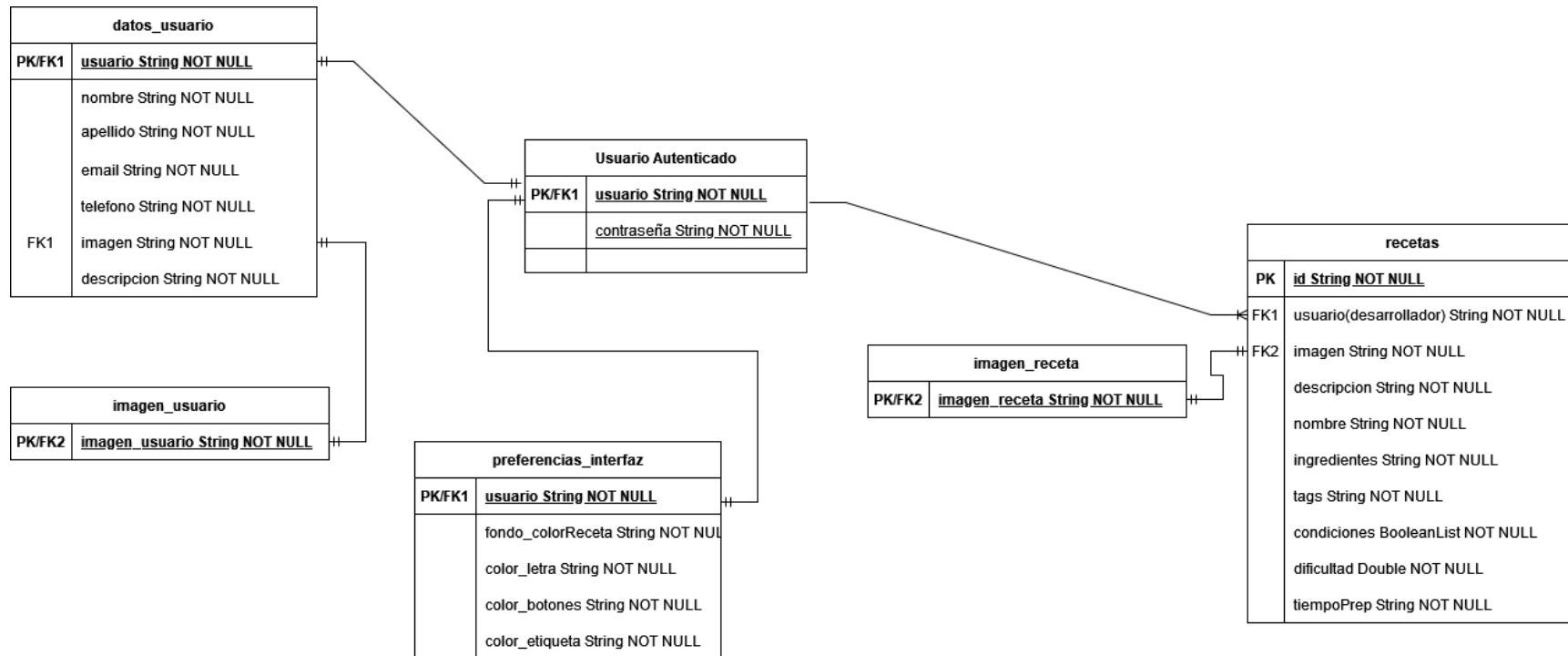


Imagen 26: Diagrama Entidad-Relación

Diagramas de clases (UML)

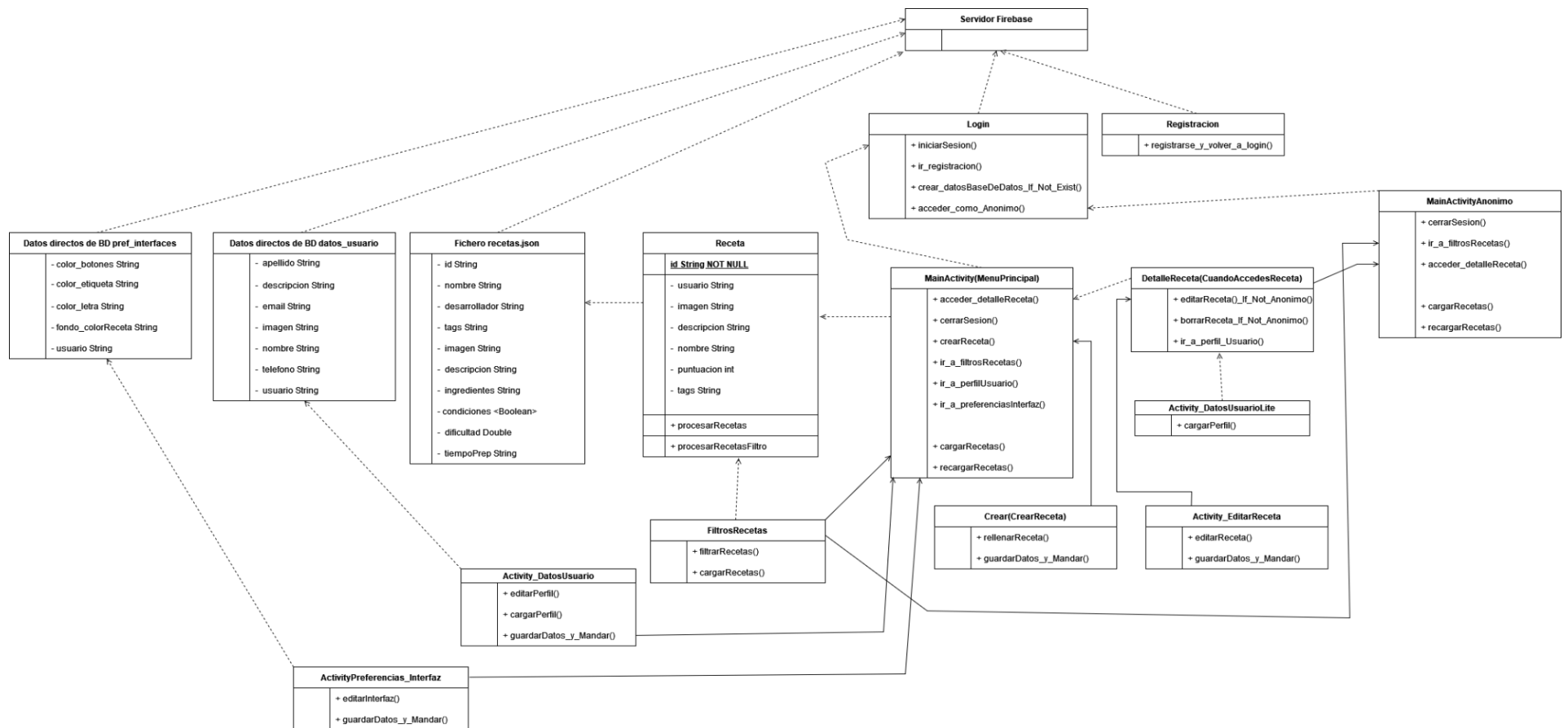


Imagen 27: Diagramas de clases

Implementación

En el inicio del desarrollo del proyecto CasualChef, fue necesario instalar algunos programas y dependencias para poder trabajar en él. Uno de los programas requeridos fue el IDE Android Studio, el cual fue utilizado como herramienta principal para crear el proyecto. La elección de este IDE se debió a que siempre se ha utilizado en clase y su manejo resulta sencillo y práctico.

Además, se emplearon algunas dependencias externas a Android Studio para mejorar algunos aspectos de la aplicación, tales como la optimización y el aspecto gráfico. Si bien estos aspectos podrían haberse logrado sin estas dependencias, su utilización permitió obtener una mayor calidad en la aplicación final.

Dependencias

Glide: Es una librería Android de código abierto que permite cargar imágenes, videos y gifs.

Se puede utilizar con servidores remotos o en el sistema local de archivos

Líneas que aplicar en el gradle:

```
{  
  
'com.github.skydoves:colorpickerview:2.2.4'  
  
}
```

Enlace: <https://github.com/bumptech/glide>

ColorPicker: Es una librería Android de código abierto que permite utilizar un selector de colores con una interfaz agradable.

Líneas que aplicar en el gradle:

```
{  
  
'com.github.bumptech.glide:glide:4.12.0'  
  
'com.github.bumptech.glide:compiler:4.12.0'  
  
'com.github.bumptech.glide:glide:4.12.0'  
  
}
```

Enlace: <https://github.com/skydoves/ColorPickerView>

CasualChef

Firebase: Es la base de datos del proyecto y como tal la aplicación necesita conectarse a ella con las herramientas que provee Firebase.

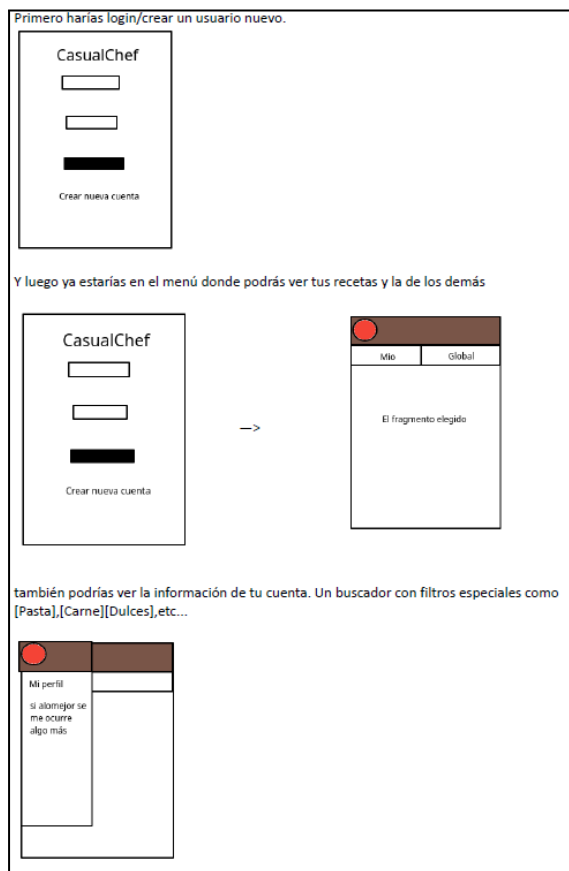
Líneas que aplicar en el gradle:

```
{  
  
'com.google.firebase:firebase-analytics-ktx'  
  
platform('com.google.firebase:firebase-bom:31.2.3')  
  
'com.google.firebase:firebase-firestore-ktx:24.1.0'  
  
'com.google.firebase:firebase-auth-ktx:21.0.3'  
  
'com.google.firebase:firebase-storage-ktx:20.1.0'  
  
}
```

Enlace: <https://firebase.google.com>

Una vez se haya acabado Instalado/Implementado todo podremos crear el programa

Desarrollo



Al principio se comenzó a crear el diseño de CasualChef con la inspiración de los bocetos que se hicieron en la entrega de propuesta de proyectos, en vez de hacer un boceto detallado de todas las ventanas de la aplicación simplemente se crearon todas las ventanas en los xml del proyecto que se iban a comenzar poco a poco, empezando primero por lo más básico como poner TextViews, hasta el momento de poner componentes más complicados como hacer que un recyclerview funcionara en una ventana.

Imagen 28: Boceto básico del proyecto

Lo primero que se creó de la aplicación fue el Menú principal y no se acabó con esa actividad hasta que funcionase el RecyclerView, enseñando los datos más básicos que pudiera y que al apretar a la receta se pudiera acceder a la información

completa de la misma receta que había apretado. Luego se tuvo que cambiar parte de la estructura para poder utilizar los datos de firebase. Una vez completada esta tarea se decidió crear otras Activities como Login y Registro que fueron las siguientes adiciones en el proyecto.

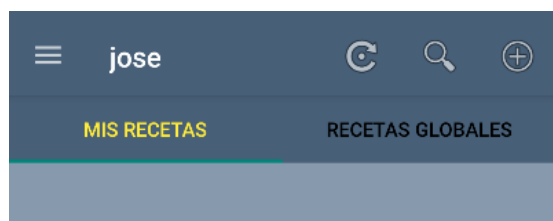


Imagen 29: AppBar | Menú Principal

Mientras se crearon las actividades mencionadas se aprovechó para aplicar todas las dependencias de firebase para poder iniciar la investigación sobre cómo

funcionaba, primero lo más básico era crear un usuario y al iniciar sesión con el poder pasar al menú principal y después de haber probado alrededor de 2 días de revisar la documentación de firebase al final se pudo hacer la tarea.

Luego al comprobar que Login funcionaba bien aunque le faltaran muchas cosas que aplicarle aún se hizo la actividad Registro para poder crear usuarios sin tener que crearlos desde la base de datos directamente.

CasualChef

Era bastante fácil debido a que se comprenden mejor los contenidos de firebase por lo que no costó mucho, pero se tuvo que pensar un tiempo sobre cómo se podrían evitar ciertos problemas que daba firebase como por ejemplo que Authentication (El componente de firebase que permite guardar los usuarios).

Firebase no permitía utilizar lo que se quería hacer que era registrar/iniciar sesión como usuario/contraseña pero solo permitía correo/contraseña y se hizo de manera de que en la interfaz solo pusieras un nombre sin caracteres especiales y al mandarlo a firebase el string tendría sumado un "@gmail.com" por lo que se daría como válido.

```
binding.button2.setOnClickListener { it: View!
    val email = binding.cogerUsuario.text.toString()+"@gmail.com"
    val password = binding.cogerContraseA.text.toString()
```

Imagen 30: Forma de evadir email/contraseña | Login y Registro

Luego en el menú principal se implementó un Menú tab que funcionaba de manera que enseñaba las recetas del usuario que había hecho login en un tab y el resto de usuarios en el otro, fue una tarea que consumió mucho tiempo, podría ser la tarea que más tiempo se tardó en crear, pero al final se llegó a crear gracias a que en el proceso de iniciar sesión, ahora creaba un sharedPref del nombre del usuario y se hacía de manera que solo salieran las recetas que coincidieran con el nombre del autor. El funcionamiento de este proceso está dividido entre varias clases incluyendo la Data Class Receta.

```
fun getReceta(context: Context, desarrollador: String = ""): List<Receta> {
    val recetaList: MutableList<Receta> = mutableListOf()

    val sharedPrefs = context.getSharedPreferences( name: "login", Context.MODE_PRIVATE)
    val username = sharedPrefs.getString( key: "username", defValue: "")

    val file = File(context.filesDir, child: "recetas.json")
    val jsonString = file.readText(Charset.defaultCharset())

    val listType: Type = object : TypeToken<MutableList<Receta?>>() {}.type
    val gson = Gson()
    recetaList.addAll(gson.fromJson(jsonString, listType))

    return when (desarrollador) {
        username -> recetaList.filter { articulo -> articulo.desarrollador == desarrollador }
        else -> recetaList.filter { articulo -> articulo.desarrollador != username }
    }
}
```

Imagen 31: Método para cargar recetas desde json | Receta

CasualChef

En este punto Login, Registro y el Menú principal estaban estables por lo que se podía continuar con otras partes del proyecto. Por estos momentos se habían añadido todos los campos que se habían pensado en un principio para que tuviera cada receta:

Id, Nombre, Autor, Tags, Imagen, Descripción e Ingredientes.

Luego en el desarrollo a petición de mi tutora se añadieron los campos:

Condiciones, Dificultad y TiempoPrep.

Al hacer el cambio se tuvieron que borrar todas recetas que habían en la base de datos y crear otras para que no hubieran incongruencias debido a que no tenían los nuevos datos.

Luego después de haber hecho el cambio se puso en marcha la actividad Crear para poder subir recetas a la base de datos, costó un poco crear el diseño de la nueva actividad ya que pedía demasiados campos como para caber en toda la pantalla por lo que se había hecho de manera que se pudiera bajar y subir en esta actividad para poder verlo todo.

Lo más complicado de subir una receta es que todo era fácil de subir ya que la mayoría de los campos era texto excepto las imágenes, primero había que hacer un método que permitiera acceder a la galería y al usuario, para poder poner la imagen de la receta que eligiese el usuario, y luego dependiendo del tamaño de la imagen de la aplicación, está podría crashear por lo que se tenía que hacer 2 métodos más. El primer método cambiaba la resolución a la imagen seleccionada y el segundo método hacía que la imagen se guardará en caché y hacer que esta parte fuera lo que se guardará en la base de datos.

Gracias a estos 2 últimos métodos aparte de poder evitar un crasheo se había podido extender la vida de la base de datos ya que firebase solo permite un uso de guardado de contenidos de solo 1 GB si no se está pagando nada por lo que de poder haber llenado el plazo con 100 fotos se ha podido extender a más de 800 por la mayoría de las imágenes mandadas a la nube son de alrededor a 1 MB.

gs://casualchef.appspot.com > images








<input type="checkbox"/>	Nombre	Tamaño	Tipo
<input type="checkbox"/>	 02ba4a9c-372f-4198-b07b-d79adb8c08ec.png	54.84 KB	image/jpeg
<input type="checkbox"/>	 030c9d04-269e-4cdd-ac19-5cb3637e06b9.png	54.84 KB	image/jpeg
<input type="checkbox"/>	 04ccc275-7b63-48a6-bfd2-ae4ea1c4113d.png	21.11 KB	image/jpeg
<input type="checkbox"/>	 06ddb246-b71c-49ef-8030-56293cd53ccc.png	255.74 KB	image/jpeg
<input type="checkbox"/>	 097864ad-717a-42ac-a1ec-9c40c9ce8e02.png	567.28 KB	image/jpeg
<input type="checkbox"/>	 0a52fc9d-0402-40da-8ea6-dd28bc33ebb2.png	254.38 KB	image/jpeg
<input type="checkbox"/>	 0ababb8c-449e-4cde-9c16-5095ece50f00.png	312.44 KB	image/jpeg

Imagen 32: Imágenes en storage de firebase

A continuación se creó la actividad Filtros, y para completarla se pudo reciclar bastante código del menú principal debido a que es muy parecido en interfaz y en su función de filtración, además la filtración ha sido extendida para poder acomodar todos los campos que se podrán utilizar para filtrar.

```
var filtroPreferencia =
    this?.getSharedPreferences(
        name: "filtro",
        MODE_PRIVATE
    )
filtroPreferencia?.edit()
    ?.putString("filtroClase", spinner.selectedItem.toString())
    ?.putString("valor", binding.textoFiltrar.text.toString())
    ?.putString("dificultad", binding.dificultadSlider.progress.toDouble().toString())
    ?.putString("bool1", binding.bool1.isChecked.toString())
    ?.putString("bool2", binding.bool2.isChecked.toString())
    ?.putString("bool3", binding.bool3.isChecked.toString())
    ?.putString("bool4", binding.bool4.isChecked.toString())
    ?.putString("bool5", binding.bool5.isChecked.toString())
    ?.apply()
```

Imagen 33: Código para poner los filtros | Filtro

El proceso de filtración funciona de manera que al apretar el botón de filtrar con los campos seleccionados, se crea un sharedPreferences llamado filtro y rellena todos los campos con todo lo que se ha puesto para filtrar. Lo más difícil de entender en la imagen es filtroClase, este es el valor que se obtiene del spinner por lo que es estático y buscará el tipo de filtrado por texto

```
when (prefFiltro) {
    "tags" -> it.tags.contains(prefValor.toString(), ignoreCase = true) ^filter
    "nombre" -> it.nombre.contains(prefValor.toString(), ignoreCase = true) ^filter
}
```

Imagen 34: Procesado de los filtros aplicados | RecetasFragmentFiltro

CasualChef

Aquí tenemos un ejemplo de código que dependiendo del prefValor, por ejemplo, “tags” mirará si la receta contiene en tags el valor que ha escrito el usuario y después de todo eso se reiniciará el recicleviewer y enseñará las recetas que cumplen los filtros.

Al acabar la función de los filtros, se comenzó con la creación de la actividad DatosUsuario y DatosUsuarioLite. Las dos actividades utilizan todo lo que hay en la base de datos para la información de perfil de cada usuario. Los campos que se componen un usuario son:

Usuario, Nombre, Apellido, Email, Teléfono, Imagen de usuario y Descripción

Debido al desarrollo de la creación de la actividad Crear, subir estos datos del usuario era bastante fácil. ya que todo era texto y la imagen se había subido a la base de datos de igual manera que en la actividad Crear. El único imprevisto es que al iniciar cada usuario por primera vez se tiene que crear un perfil para dicho usuario, si no la aplicación crasheaba.

El siguiente paso era crear la actividad Preferencias_Interfaz y su creación en general también ha sido bastante fácil pero han habido imprevistos que habían demorado la creación de esta actividad por un día entero.

Hasta este momento todas o la mayoría de los campos del proyecto tenían su color predefinido en el fichero Colors.xml, como color de letra o color de fondo y hasta el momento de crear esta actividad el plan era cambiar los valores de esos colores desde código, pero por lo que se descubrió después de numerosas pruebas se llegó a la conclusión de que la tarea en cuestión resulta inviable, y por lo tanto, no se podría cambiar los colores de todo el programa sin tener que obtener los colores usuario y cambiarlos en cada recoveco del programa con código.

Entonces se había decidido por simplemente cambiar los colores de cada campo en la información detallada de las recetas para no descartar esta actividad.



El código manda la información a firebase generando los códigos hexadecimales de los colores con la herramienta ColorPicker, que permite al usuario seleccionar el color que desee de este menú y al hacerlo se cambia el color hexadecimal que había, y también se cambia el color del cubo posicionado a la derecha.

Color de botones: #787A7A



Imagen 35: ColorPicker enseñado al editar | Preferencias_Interfaz

Imagen 36: Forma de enseñar el color al usuario | Preferencias_Interfaz

Los datos que se pasan a la base de datos son los más fáciles de obtener de todo el programa ya que son todo texto.

Igual que en la actividad DatosUsuario si el usuario no tiene ya una configuración de colores, se le proveerá la suya propia en la base de datos y esos colores dependerán de si el usuario tiene el dispositivo en modo oscuro o no.

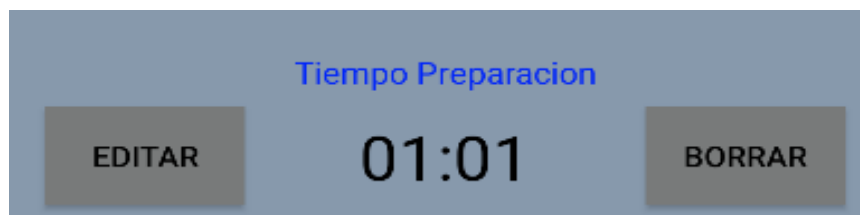


Imagen 37: Botones de editar y borrar de la receta del usuario | Detalle

La siguiente actividad que se ha hecho es EditarRecetas y se puede acceder a ella de la misma manera que para borrar una receta. Al entrar se muestra la información detallada de una receta que pertenezca al usuario que ha iniciado sesión y dándole a editar podrá modificar los campos.



El contenido de la actividad

EditarRecetas es la misma actividad que Crear, pero en esta debido a que se tiene que editar algo que ya existe, al cargar la interfaz se carga cada campo con la información debida de la receta que se va a editar.

Imagen 38: Datos extensos de una receta | Detalle

Imagen 39: Receta al haber apretado editar | EditarReceta

CasualChef

Lo más complicado en la creación este código ha sido pensar alguna manera de evitar errores como por ejemplo al poner las horas y los minutos, si no habian 2 dígitos en cada apartado la aplicación crasheaba por lo que se ha hecho un arreglo con un listener para cada campo que detecte si se ha dejado el campo sin 2 dígitos para rellenarlo con un cero a la izquierda cuando salga del campo.

```
horasEditText.setOnFocusChangeListener { _, hasFocus ->
    if (!hasFocus) {
        val text = horasEditText.text.toString().padStart( length: 2, padChar: '0')
        horasEditText.setText(text)
    }
}
```

Imagen 40: Listener usado para utilizar 2 digitos | EditarReceta

Y para añadir una medida más al hacer clic en editar para enviar los datos se rellenan los 2 dígitos si no se han rellenado antes porque el usuario no ha salido del campo al apretar el botón.

```
binding.btnMandar.setOnClickListener { it: View!
    if (horasEditText.toString().length == 1) {
        val text = horasEditText.text.toString().padStart( length: 2, padChar: '0')
        horasEditText.setText(text)
    }

    if (minutosEditText.toString().length == 1) {
        val text = minutosEditText.text.toString().padStart( length: 2, padChar: '0')
        minutosEditText.setText(text)
    }
}
```

Imagen 41: Código usado al apretar el botón | EditarReceta

```
if (imagenClicked==true) {
    var imagenID: String = UUID.randomUUID().toString() + ".png"

    val storageRef =
        FirebaseStorage.getInstance().reference.child( pathString: "images/${imagenID}")

    val uploadTask =
        imageUri?.let { cambiarResolucionImagen_ComprimirGuardar(it) }?.let { storageRef.putFile(it) }
    uploadTask?.continueWithTask { task ->
        if (!task.isSuccessful) {
            task.exception?.let { it: Exception
                throw it
            }
        }

        storageRef.downloadUrl ^continueWithTask
    }
    paqueteReceta["imagen"] = imagenID
} else {
    paqueteReceta["imagen"] = nombreImagen
}
```

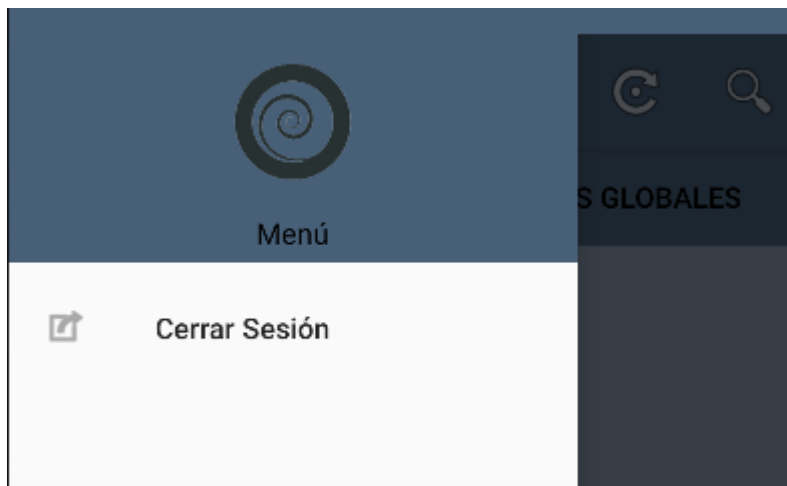
Imagen 42: Detección de uso de imagen | EditarReceta

O también si el usuario quiere editar la receta y no quiere añadir foto se tenía que hacer un modo de ver si se tiene que utilizar alguna imagen usada o simplemente poner el mismo código de imagen que tenía antes.

CasualChef

Luego como última adición se añadió la actividad MainActivity_Anonimo que es la pestaña a la que manda el botón de anónimo en la actividad Login.

No tiene muchas diferencias comparado a la actividad MainActivity normal pero está hecha exclusivamente para la gente que no desea crearse otra cuenta debido a que tendría que preocuparse de no olvidarse de su usuario/contraseña ya que la aplicación no tiene ninguna forma de recordarla o por otros motivos del usuario.

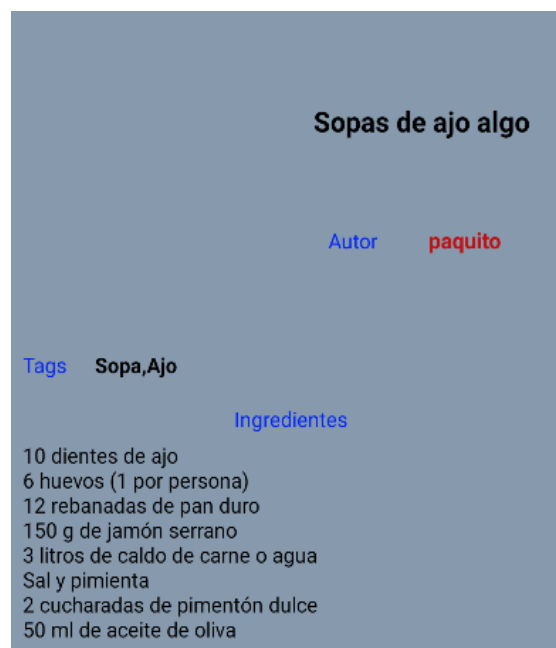


Ya que no es un usuario, por primero no se puede crear un usuario con el mismo nombre ya que no está permitido y segundo no tiene acceso ni a configurar su perfil ni a configurar su interfaz ni a poder crear recetas.

Imagen 43: Menú principal anónimo con el menú hamburguesa abierto

También está diseñada para cuando no tenga acceso a internet ya que no hace login en la base de datos en ningún momento y puedes utilizar la aplicación sin conexión mientras que se haya descargado un json antes de la pérdida de conexión.

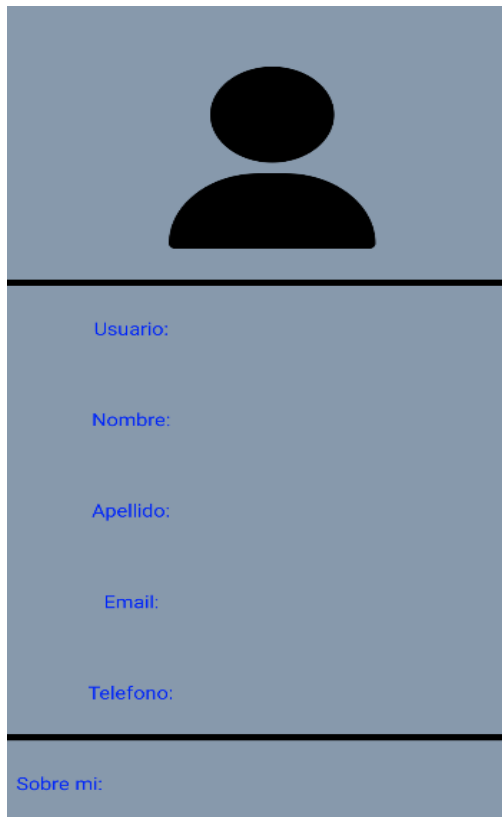
Tratando del tema sin conexión, Firebase está preparado para no poder acceder a la red ya que al iniciar sesión el usuario descarga un json con todas las recetas y eso una vez se quede sin conexión lo seguirá pudiendo ver con el unico inconveniente que las fotos no se pueden ver.



CasualChef

Imagen 44: Menú de filtros sin conexión a internet

Imagen 45: Detalle de sin conexión a internet



Los usuarios a los que se les quiera ver el perfil tampoco podrán verse sin conexión ya que es para cargar sus datos se llama a firebase para obtenerlos por lo que solo saldría un perfil así de vacío.

También al no tener un menú de configuración de interfaz el usuario anónimo siempre utiliza los colores de interfaz predeterminados del sistema que cambiarán dependiendo si el usuario tiene el dispositivo en modo oscuro o no.

Imagen 46: Datos usuario sin conexión a internet

Luego al haber acabado todo lo que tendría que tener el proyecto se hicieron algunos cambios para mejorar la optimización de la aplicación, para hacer el código más legible cambiando el nombre de variables y funciones a otros más correspondientes a su función y casos similares.

Uno de esos casos era Glide, la aplicación siempre obtenía las imágenes desde el login y tenía que esperar a que todas las imágenes que se utilizaban en recetas se descargan y eso hacia la carga significativamente más larga y tediosa por lo que se buscaron alternativas y se encontró el repositorio de Glide

```
fun String.ponerImagen(context: Context, imageUrl: String, imageView: ImageView){  
    Glide.with(context).requestManager  
        .load(imageUrl).requestBuilder<Drawable!>  
        .diskCacheStrategy(DiskCacheStrategy.ALL)  
        .placeholder(R.drawable.casualchef)  
        .into(imageView)  
}
```

Imagen 47: Código de glide en la búsqueda de imágenes | Utils

CasualChef

En vez de estar utilizando imágenes fijas en la cache una vez se hayan descargado todas glide va descargando poco a poco cada imagen de cada receta por lo que no hay esperas, lo único que empeora comparado a descargarlo todo de una vez es que dependiendo de la velocidad de internet del usuario alomejor la imagen tarda más o menos en cargar.

Además se ha intentado de toda manera intentar que Glide guarde todas las imágenes que descargar en caché pero no enseña si no hay internet aunque al ver los datos de la aplicación en el emulador enseñe de que si que se guardan.

```
val storageRef = FirebaseStorage.getInstance().reference.child( pathString: "images/${articulo.imagen}")
storageRef.downloadUrl.addOnCompleteListener { task ->
    if (task.isSuccessful) {
        val imageUrl = task.result.toString()
        articulo.imagen.ponerImagen(holder.ivArticulo.context, imageUrl, holder.ivArticulo)
    } else {
        holder.ivArticulo.setImageResource(R.drawable.casualchef)
    }
}
```

Imagen 48: Código de Glide para buscar | MyRecyclerViewAdapter

Pruebas

Estas pruebas van desde el principio del proyecto hasta el final

Prueba 1. Debido a que firebase no permite usuario/contraseña sino correo/contraseña así que pido al usuario un nombre de usuario en registro pero en si le estoy metiendo un “@gmail.com” al final para que firebase lo de como válido.

Prueba 2. Se estaban descargando todas las imágenes desde firebase cuando se iniciaba sesión y por lo consecuente tardaba bastante en cargar cuando habían más de 1 o 2 fotos por lo que se había cambiado a utilizar Glide para poder iniciar sesión sin esperas y se podía dejar que Glide fuera descargando y colocando cada imagen donde le tocaba aunque dependiendo de la conexión a internet a veces tardaba más y a veces menos.

Prueba 3. Al principio se había pensado que se necesitaba una clase para cada colección de firebase para poderlas cargar de alguna manera pero al final solo se necesitaba la de Recetas ya que se necesitaba para que el recycleviewer utilizara el json para cargar todas las recetas, el resto de colecciones se cargaban con una simple query a firebase que pidiera los datos de algo específicamente.

Prueba 4. Cuando se había comenzado a crear la actividad Filtros se estaba pensando en poder hacer como en el Menú principal y poder filtrar en un tab por lo que has buscado y en otro por lo que no pero por desgracia parecía mucho más difícil filtrar por tal cosa ya que el menú principal solo filtraba si la receta tenía el nombre del usuario que ha iniciado sesión o no y en este caso tal vez se podría hacer obteniendo todo el json de las recetas y crear otro exclusivamente para la actividad filtros que dependiendo de los filtros elegidos que se ponga a cada receta un boolean y si es True que se pusiera en Filtrado y si no en otros pero eso habría sido mucho tiempo de desarrollo, al final se había tenido que dejar ver solo lo que se quería filtrar.

Prueba 5. Se estaba intentado optimizar la aplicación y se había podido hacer para el Options Menu pero no para el Drawer ya que tengo 3 navigations en el mismo Activity no funcionaba muy bien que al apretar algo que hiciera algo desde el MainActivity y solo podía hacer lo que quería cambiando de cada drawer invisible a un drawer que te envía a otro Activity dependiendo de cual hayas decidido utilizar y luego al volver al MainActivity hacer que el drawer se vuelva a poner el primero que no hace nada y está vacío.

Prueba 6. Cuando se estaban haciendo los colores estáticos para todo en colors.xml esperaba poder cambiar los valores de algún modo con código y después de estar probando un día al final se había decidido solo cambiar los colores de cada campo de las recetas para no descartar la idea de los colores por completo. Pero a cambio de haber hecho eso he podido añadir un modo oscuro y un modo claro dependiendo de que tenga el dispositivo en el que esté ejecutándose la aplicación.

Prueba 7. Se había intentado dejar algún tipo de indicador de calidad de la receta que pudieran controlar los usuarios en forma de Likes/Puntuación pero sería una función que costaría mucho de implementar debido a que habría que controlar que cada usuario solo pudiera dar una puntuación a cada receta y se tendría que hacer todo en tiempo real para no evitar problemas.

Documentación

Este proyecto de TFG sobre la aplicación móvil CasualChef se complementa con los siguientes documentos:

- **Instrucciones de instalación de CasualChef app:** Este documento proporciona instrucciones detalladas para instalar y configurar la aplicación CasualChef en un dispositivo móvil Android.

Los contenidos en soporte digital son:

- **Archivo del proyecto comprimido en formato zip:** Este archivo contiene el código fuente de la aplicación CasualChef con los archivos de configuración necesarios para su funcionamiento. El archivo zip se puede descomprimir y utilizar para acceder a todo el contenido del proyecto.
- **Instalador de entorno de desarrollo:** Se incluye el instalador del entorno de desarrollo utilizado para desarrollar la aplicación CasualChef, Android Studio.

Estos documentos y archivos ofrecen una visión integral del proyecto, desde la documentación escrita hasta los recursos digitales indispensables para compilar y ejecutar la aplicación móvil CasualChef.

Trabajos futuros

Implementaciones futuras:

1. Un chat entre usuarios a tiempo real
2. Poder poner comentarios en las recetas
3. La función de poder valorar con likes/puntuación de 0 a 5 cada receta
4. Poder ver en el perfil cuantas recetas tiene el usuario y listarlos
5. Poder guardar recetas en accesos rápidos
6. Poder ocultar recetas que no le gusten al usuario
7. Poder hacer que las fotos se pudieran ver offline

Conclusiones

Con todo lo que se ha tardado se puede ver el porque desarrolladores de juegos y cualquier tipo de software pueden tardar tanto en arreglar bugs y añadir nuevo contenido.

Para aplicar nuevo contenido siempre son entre 1 día hasta 5 porque a lo mejor mientras se está creando sale una nueva idea mejor y hay que comenzar desde 0 o simplemente que añadir ciertas cosas resulta en mucho tiempo.

Y para encontrar bugs hay que hacer bastantes tests seguidos por cada versión una vez ya todo está más o menos estable para poder ver los problemas pequeños como deshabilitar un botón al presionarlo o que un texto tiene fallos ortográficos.

En general ha sido una gran oportunidad para aprender y acostumbrarse a este tipo de vida.

Bibliografía y webgrafía

<https://firebase.google.com/docs?hl=es-419>

<https://developer.android.com/docs>

<https://github.com/skydoves/ColorPickerView>

<https://github.com/bumptech/glide>

<https://stackoverflow.com/>

<https://www.youtube.com/>

Anexos

Adjuntar cosas que se han utilizado en el proyecto o algo?