Desarrollo de Aplicaciones Multiplataforma Online

GENERALITAT VALENCIANA
CONSELLERIA D'EDUCACIÓ, INVESTIGACIÓ, CULTURA I ESPORT

Florida Universitària

Proyecto Final de Ciclo de Desarrollo de Aplicaciones Multiplataforma Online

Título: YummyGo

Apellidos y nombre del autor/a: Víctor González Devesa

Curso Académico: 2024/2025

Fecha Entrega: 21/05/2025

ÍNDICE

1. RESUMEN DEL PROYECTO. 4
2. Justificación y objetivos del proyecto 5
2.1. Justificación..... 5
2.2. Objetivos: 6
3. DESARROLLO DEL PROYECTO..... 7
3.1. Análisis del mercado y posible modelo de negocio.. 7
3.3. Metodologías utilizadas.. 8
3.4. Descripción de los componentes de la aplicación.. 8
3.5. Problemas/dificultades encontradas en el desarrollo del PFC y soluciones adoptadas: 18
3.6. Resultados obtenidos..... 19
4. CONCLUSIONES..... 21
5. LÍNEAS FUTURAS DE TRABAJO 21
6. BIBLIOGRAFIA 23

Tabla de ilustraciones

Ilustración 1 - Pantalla inicial(Login, Registro, Recuperar contraseña) 11
Ilustración 2 - Pantalla principal(Perfil y listado de recetas buscadas) 12
Ilustración 3 - Pantalla favoritos (Recetas y Productos) 13
Ilustración 4 - Pantallas extendidas de la recetas y productos 14
Ilustración 5 - Pantalla de escaneo o buscador de productos 15
Ilustración 6 - Ejemplo de datos de usuario 16
Ilustración 7 - Ejemplo de receta guardada 17
Ilustración 8 - Ejemplo de producto guardado 17

1. RESUMEN DEL PROYECTO

Proyecto de Aplicación de recetas y escaneo de productos.

YummyGo es una aplicación móvil que he creado para ayudar a encontrar recetas a partir de los ingredientes que ya hay por casa. La idea surge de una situación muy común: tengo hambre, hay comida en la nevera, pero no me apetece nada en concreto o no sé qué preparar.
Con esta aplicación, los usuarios pueden escribir los ingredientes con los que cuentan y YummyGo les sugiere recetas que pueden hacer con eso, sin necesidad de salir a comprar más cosas. El objetivo principal es facilitar el día a día de la gente que tiene poco tiempo para pensar qué cocinar y, además, ayudar a aprovechar lo que ya se tiene para desperdiciar menos alimentos.
Además, la aplicación incluye una funcionalidad extra que permite escanear productos mediante su código de barras. Esta opción resulta especialmente útil a la hora de hacer la compra, ya que muchas veces las etiquetas de los productos están llenas de información poco clara y, en el caso de los menos saludables, ni siquiera aparece el Nutri-Score, una herramienta muy útil para ver rápidamente el valor nutricional general del alimento.
YummyGo está pensada para ser práctica, rápida y fácil de utilizar. No hace falta ser un experto en cocina ni tener mucho tiempo.

Recipe and Product Scanning App Project

YummyGo is a mobile application I created to help people find recipes based on the ingredients they already have at home. The idea comes from a very common situation: I'm hungry, there's food in the fridge, but I don't feel like eating anything in particular or I simply don't know what to cook.
With this app, users can type in the ingredients they have, and YummyGo suggests recipes they can prepare using those items, without needing to go out and buy more. The main goal is to make everyday cooking easier for people who don't have much time to think about what to cook, while also encouraging them to make the most of what they already have and reduce food waste.

In addition, the app includes an extra feature that allows users to scan products using their barcode. This is especially useful when shopping, since product labels are often filled with unclear or overwhelming information, and in many cases, especially with less healthy items, the Nutri-Score isn't even shown — which is a very helpful way to quickly understand the product's overall nutritional value.
YummyGo is designed to be practical, fast, and easy to use. You don't need to be a cooking expert or have much time.

2. JUSTIFICACIÓN Y OBJETIVOS DEL PROYECTO

En este apartado explico porqué decidí desarrollar una aplicación de este tipo, qué necesidad busco cubrir con la aplicación y que objetivos concretos me marqué al empezar. La idea no surge solo de la necesidad de hacer el proyecto final, sino de una situación real y cotidiana que me parecía fácil de identificar.

2.1. Justificación:

YummiGo nace como respuesta a una situación que todo el mundo ha vivido alguna vez. Tienes comida en casa, pero ni sabes qué hacer con ella ni tienes tiempo para pensar en como combinarlos o puedes tener tiempo, pero no ganas para pensar en que hacer. Esto lleva a lo fácil, pedir comida, comer fuera de casa o incluso a tirar comida porque se acaba estropeando.
Quería hacer una aplicación que solucionara eso de forma rápida, sin complicaciones y pensada para gente que no quiere perder el tiempo ni complicarse demasiado. La idea es que con solo introducir algunos de los ingredientes, la aplicación sugiera platos que se pueden preparar con lo que ya se tiene. De esta forma, se fomenta el aprovechamiento de la comida y se reduce el desperdicio, que es algo que también me parece muy importante.
Además, decidí añadir una funcionalidad extra que permite escanear productos durante la compra. Muchas veces los envases están llenos de letra pequeña, información poco clara, y especialmente en los productos menos saludables, no aparece el **Nutri-Score.** Me parecía útil que con una misma app pudiera ayudar a tomar decisiones rápidas y más informadas.

En resumen, la aplicación no solo pretende ser práctica a nivel funcional, sino también contribuir a hábitos de consumo y cocina más conscientes, sin necesidad de que el usuario tenga conocimientos avanzados ni se complique la vida.

2.2. Objetivos:

Ofrecer ideas de recetas según los ingredientes disponibles
Permitir al usuario escribir lo que tiene por casa y obtener recetas que pueda preparar sin necesidad de comprar más cosas.

Hacer más fácil el momento de decidir qué cocinar
Evitar que el usuario pierda tiempo buscando o pensando recetas al azar. La app debe agilizar ese momento.

Ayudar a aprovechar mejor los alimentos que ya se tienen
Fomentar el uso de ingredientes antes de que se estropeen, reduciendo así, el desperdicio de comida en casa.

Permitir guardar recetas favoritas
Dar la opción de marcar recetas para tenerlas a mano y volver a consultarlas fácilmente.

Crear una experiencia de uso simple, rápida y sin complicaciones
Diseñar una interfaz que sea amable con el usuario y no requiera de una curva elevada de aprendizaje.

Consultar información nutricional de productos de forma rápida
Permitir al usuario escanear productos en el supermercado y acceder fácilmente a datos relevantes como el Nutri-Score, ayudando a tomar decisiones de compra más conscientes.

Tener todo en una sola app
Reunir en una única herramienta tanto la búsqueda de recetas, como el

escaneo de productos y la gestión de favoritos, evitando tener que usar múltiples aplicaciones o páginas web.

Aumentar la autonomía del usuario en la cocina

Dar apoyo a personas que no tienen conocimientos de cocina o poca creatividad culinaria, para que puedan cocinar por su cuenta sin depender de ayuda externa.

3. DESARROLLO DEL PROYECTO

3.1. Análisis del mercado y posible modelo de negocio.

Aunque existen numerosas aplicaciones de recetas, muchas se centran en ofrecer contenido general sin adaptarse realmente a lo que el usuario tiene disponible. Con YummyGo quería crear una app más práctica y rápida, centrada en solucionar ese problema diario.

Una función que diferencia a YummyGo es la posibilidad de escanear productos para consultar información básica como el nombre, ingredientes o Nutri-Score. Esta funcionalidad usa la API de Open Food Facts, una base de datos abierta pero muy general, lo que hace que en muchos casos no aparezcan productos típicos del mercado español.

Por ese motivo, una posible evolución dentro del modelo de negocio sería construir una base de datos propia alimentada automáticamente a partir de los escaneos realizados por los usuarios. Así, se iría generando un repositorio real de productos disponibles en España, mejorando la calidad de la información y haciendo que la app sea más útil a largo plazo.

En cuanto al modelo de negocio en sí, si el proyecto creciera, se podría plantear una versión “free”, donde las funciones básicas sean gratuitas y algunas funciones avanzadas, como el historial de recetas, el escaneo ilimitado o sugerencias más detalladas, requieran una pequeña suscripción. También podría incluirse publicidad relacionada o colaboraciones con supermercados para integrar listas de la compra o incluso añadiendo un asistente de IA que te ayude o aconseje con las recetas o la lista de la compra, pero siempre cuidando

que no afecten a la experiencia de uso.

3.2. Metodologías utilizadas.

Al inicio del desarrollo de YummyGo me marqué una serie de objetivos claros y traté de visualizar las partes más importantes que debía tener la aplicación para que fuera útil desde el principio. Durante el desarrollo no seguí una metodología formal ni utilicé herramientas específicas de organización. En lugar de eso, fui marcándome pequeños objetivos y tareas a medida que avanzaba, dividiendo el proyecto por partes según la lógica de la aplicación: primero el backend, luego la integración con APIs, y finalmente el frontend.

Me organicé de forma personal, apuntando lo que quería implementar en cada momento y cumpliendo esos objetivos uno a uno. Aunque no seguí ningún sistema ágil como Scrum, Jira o Trello, intenté mantener un orden y una progresión clara, priorizando siempre las funcionalidades más importantes, y una vez terminadas y comprobadas, pasaba a la siguiente tarea.

3.3. Descripción de los componentes de la aplicación.

3.3.1 Arquitectura del sistema

Esta aplicación sigue un patrón de arquitectura cliente-servidor clásica. El frontend está desarrollado con React Native, esta se encarga de interactuar con el usuario, mientras que la parte del backend, está desarrollada con Spring Boot. Expone una API REST que recibe y procesa las peticiones del front para comunicarse con las API s externas además de gestionar también la base de datos.

La parte móvil se encarga de realizar peticiones HTTP al backend, este las procesa y dependiendo del endpoint al que se haya hecho la llamada, puede consulta la base de datos, tanto para agregar, modificar o borrar datos o puede pedir información a las APIs externas, exponiendo la información devuelta modificada por DTOs.

He utilizado una arquitectura por capas tradicional, que ha sido suficiente para el tamaño y alcance, de momento, del proyecto. Sin embargo, si en el futuro la aplicación creciera tomando un rumbo profesional y se quisiera facilitar la

escalabilidad, el testeo o se quisiera incorporar nuevas funcionalidades, el backend se migraría hacia una arquitectura hexagonal, donde las dependencias estarían mejor aisladas y la lógica quedaría completamente separada de los controladores, servicios externos y bases de datos.

3.3.2 Backend y APIs utilizadas

Para el Backend se ha utilizado el lenguaje Java con Spring Boot, toda la lógica está organizada por capas, controladores, servicios y DTOs para facilitar la gestión de las clases y para reducir la cantidad de código he utilizado Lombok, que es una herramienta que permite generar automáticamente los getters, setters, constructores y otros métodos más comunes. Para el entorno en vez de utilizar el eclipse standard, he utilizado Spring Tool Suit 4(STS4), que ofrece uan integración más cómoda al entorno de Spring.
La aplicación funciona a través de una API REST, que recibe las peticiones del front, las procesa y responde con un formato JSON con la información necesarios.
Uno de los problemas del backend ha sido integrar varias APIs externas, las cuales, son esenciales para el funcionamiento de la aplicación. Las APIs utilizadas son las siguientes:

- **Spoonacular API:** Esta se encarga de devolver las recetas en función de los alimentos que le pase el usuario. Las respuestas vienen en inglés por lo que opté por utilizar la API de Google Translate para poder mostrar la información en español.
- **Open Food Facts API:** Con esta Api puedo escanear los códigos de barras y recibir información del producto com por ejemplo: nombre, marca, ingredientes… Lo bueno de esta es que es una API muy completa y grande pero tiene la desventaja de que en algunos productos, muchos campos vienen vacíos o la información de estos está incompleta.
- **Google Translate API:** Esta se ha utilizado para traducir automáticamente al español la información recibida tanto de Spoonacular como de Open Food, lo cual mejora la experiencia de uso de los usuarios y hace que la aplicación sea más accesible.

3.3.3 Base de datos

Para el almacenamiento de los usuarios y los favoritos he utilizado MongoDB Atlas, una base de datos no relacional en la nube que permite que pueda trabajar con los documentos en formato JSON. Como es una aplicación donde los datos que necesito utilizar son variables, MongoDB me ha dado mucha flexibilidad para poder adaptar las colecciones a los usuarios, recetas y productos.

Desde el backend se gestionan las diferentes operaciones como por ejemplo guardar nuevos usuarios, actualizar información de estos, guardar o eliminad de favoritos recetas y productos. Para organizar los datos he utilizado una colección de Usuarios donde cada usuario tiene sus apartados de favoritos donde guardar las recetas y los productos que más les gusten y así tenerlos siempre a mano.

Toda la conexión con MongoDB se realiza desde Spring Boot, utilizando los repositorios de Spring para realizar operaciones CRUD de una forma más sencilla.

3.3.4 Interfaz de usuario y navegación

La interfaz se ha desarrollado utilizando React Native con Expo. El objetivo principal desde el principio fue crear una aplicación intuitiva y fácil de utilizar, sin menús complicados ni procesos de registro largos y que así cualquier persona pueda entenderla a la primera sin necesidad de haber utilizado muchas mas aplicaciones de este estilo.

La navegación entre pantallas la he construido utilizando React Navigation que es un sistema flexible para la integración, apoyado por renderizados condicionales. El flujo de navegación principal es el siguiente:

- Pantallas de inicio:
-

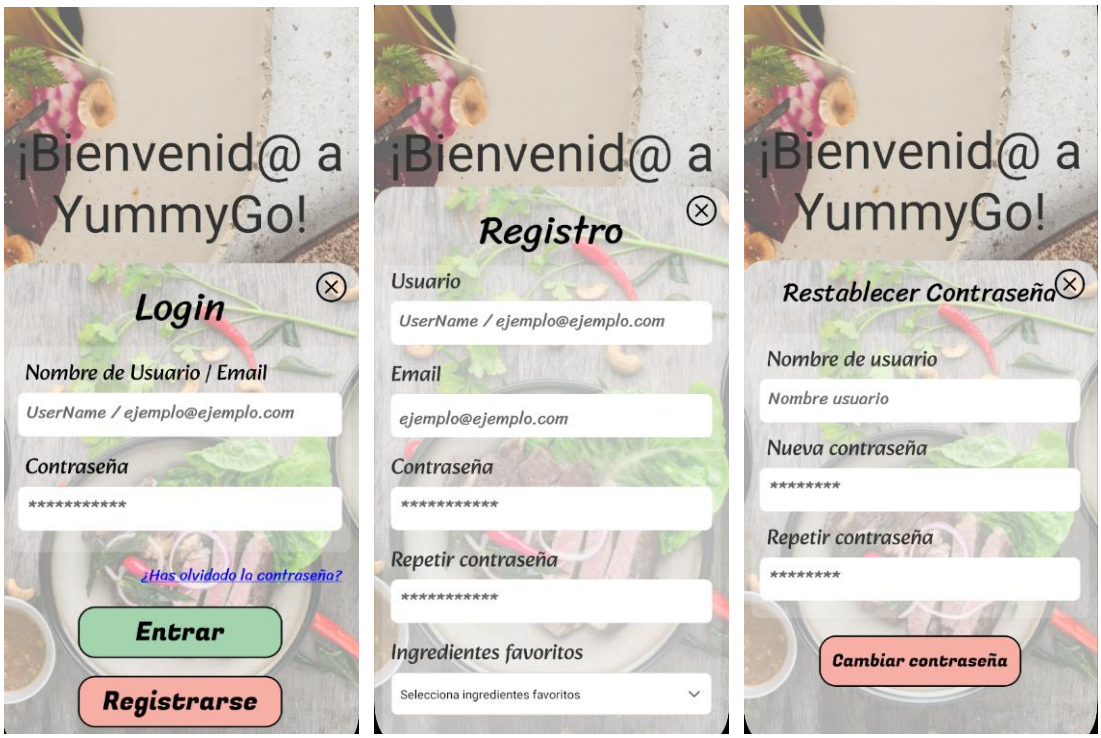


Ilustración 1 - Pantalla inicial (Login, Registro, Recuperar contraseña)

En la pantalla de inicio que se puede ver, está constituida por 4 botones:

1. Iniciar sesión: Accedes a la pantalla donde te puedes loguear e iniciar

sesión, también puedes acceder desde aquí a “recuperar contraseña”.

2. Registrarse: Accedes a la pagina de registro, donde introduciendo usuario, correo electrónico, contraseña y al menos, un alimento favorito puedes registrarte y disfrutar de la aplicación.
3. Boton de Spoonacular(Izquierda): este te lleva a la página principal de Spoonacular.
4. Boton de Open Foon Facts(Derecha): Con este si se accede desde el ordenador, te redirige a su pagina principal y si es en móvil te lleva a su aplicación oficial de la que me he inspirado.

- Pantalla principal:

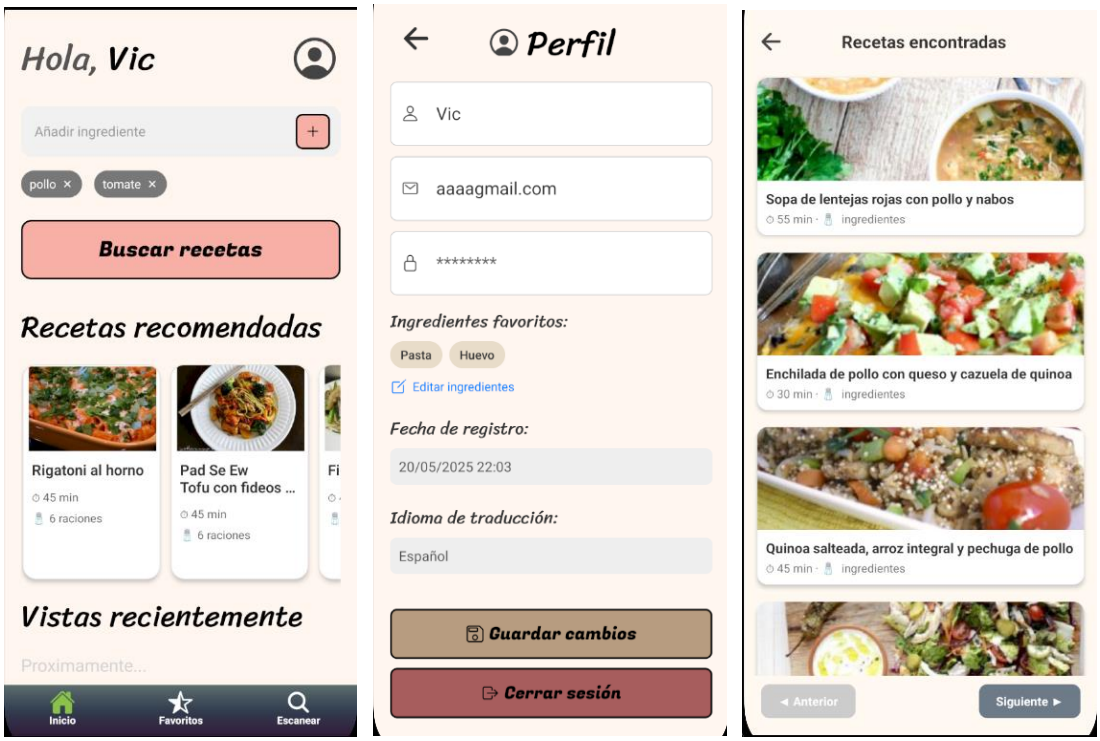


Ilustración 2 - Pantalla principal (Perfil y listado de recetas buscadas)

En esta pantalla puedes acceder al perfil pulsando en el icono del perfil, acceder a alguna de las recetas recomendadas que vienen dadas según los alimentos que elegiste al registrarte, puedes buscar recetas añadiendo ingredientes en el input de la parte superior, en la parte inferior de la pantalla se puede ver un “Tab” desde el que se puede navegar libremente entre pantallas.

- Favoritos:

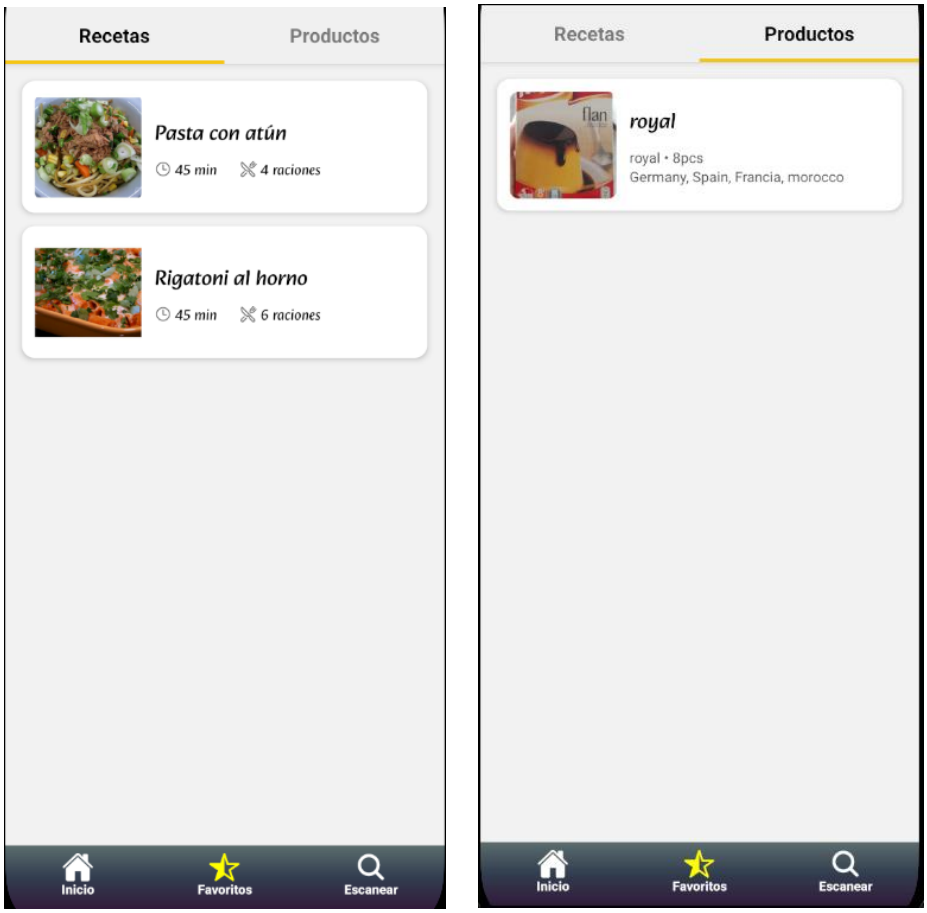


Ilustración 3 - Pantalla favoritos (Recetas y Productos)

En esta pantalla se ven las recetas y los productos que se han agregado a favoritos con un poco de información, desde ahí al pulsar sobre una tarjeta te abre la receta o la información completas del producto.

- Pantallas de receta y producto:

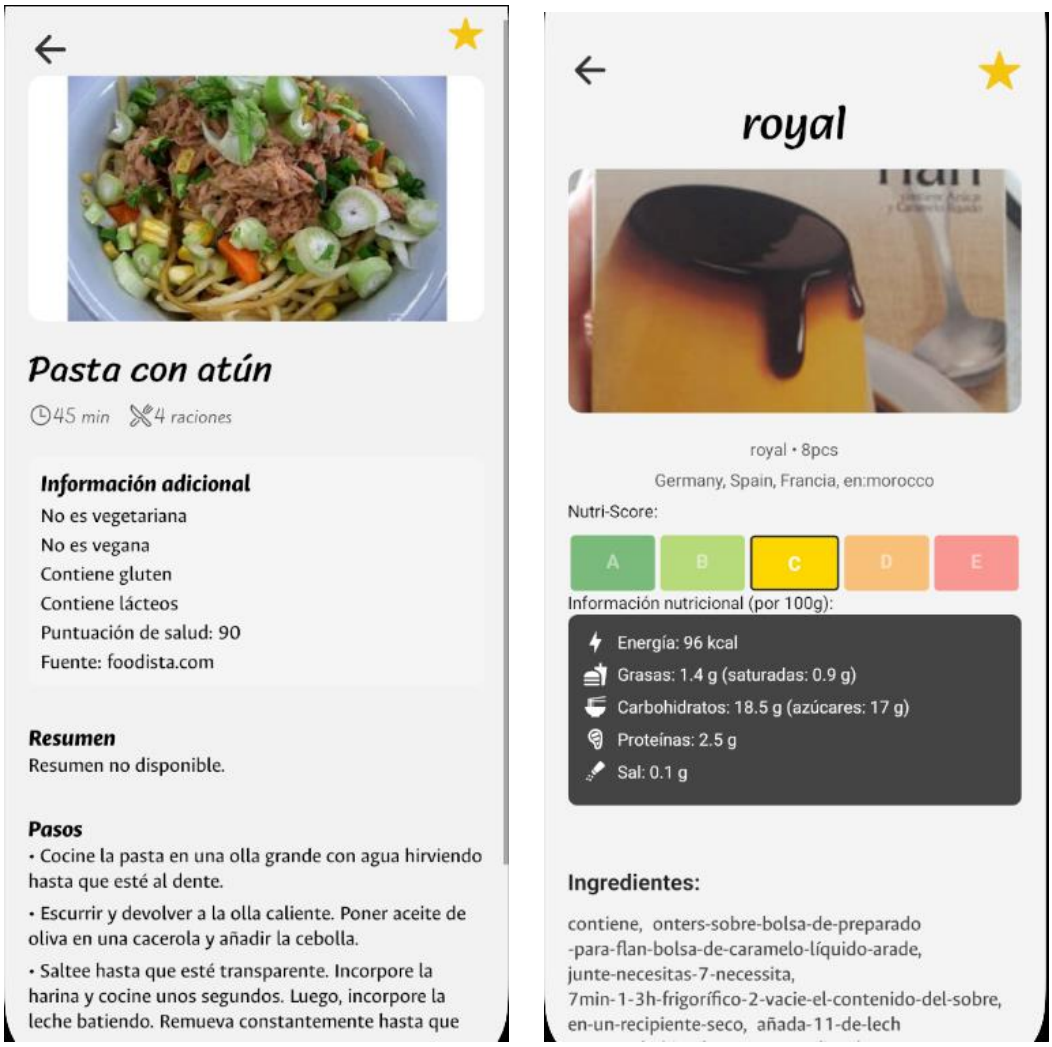


Ilustración 4 - Pantallas extendidas de la recetas y productos

En estas dos pantallas se puede ver la información completa de una receta (izquierda) y la información de un producto (derecha). La estrella de la parte superior derecha está iluminada porque está en favoritos, de no estarlo aparece gris.

- Pantalla de escaneo

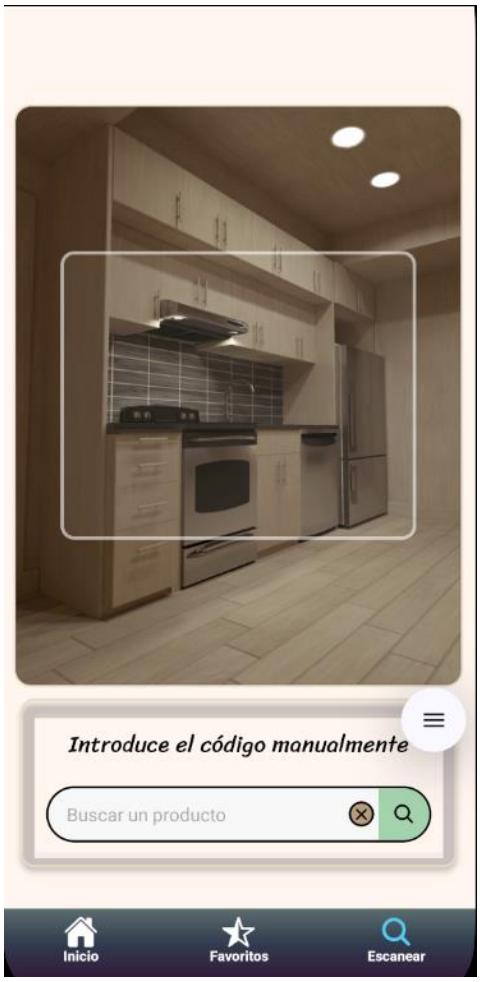


Ilustración 5 - Pantalla de escaneo o buscador de productos

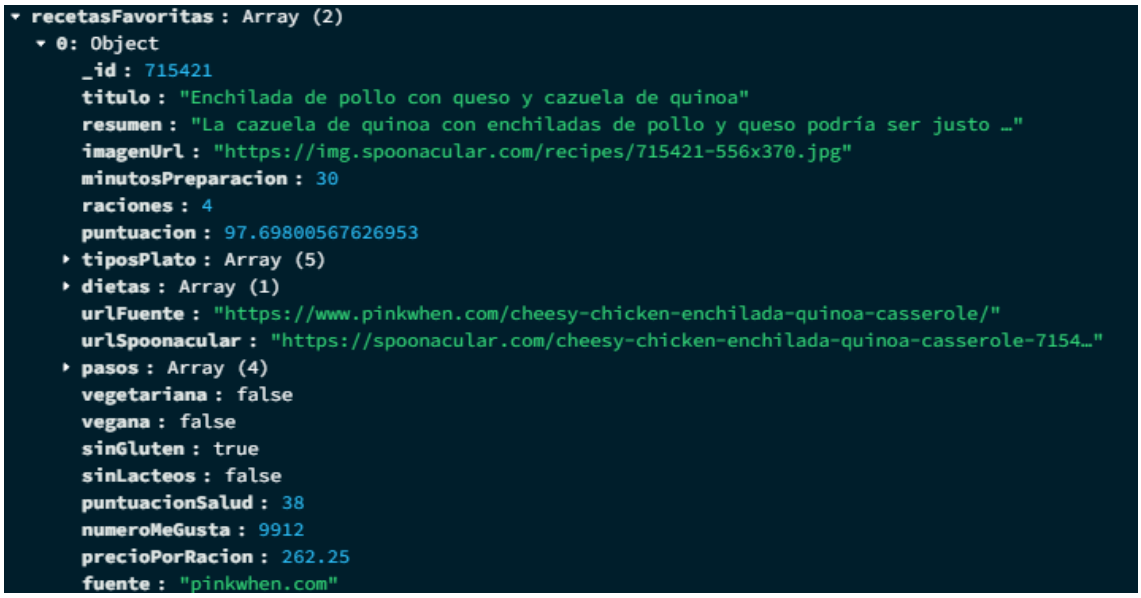
En esta pantalla consta de una parte superior donde está la cámara y recoge el código de barras y la parte inferior donde se puede buscar introduciendo dicho código. Cuando se busca tanto manual como por la cámara una vez detectado el código, este te lleva a la página de información del producto (ilustración 4 derecha).

3.4.5 Ejemplos de respuesta JSON y DTO

En este punto voy a mostrar un ejemplo de como guardo a los usuarios en la base de datos representado como un documento JSON por Mongo.

Ilustración 6 - Ejemplo de datos de usuario

Aquí se puede ver que entre todos los tatos, se guarda la contraseña hasheada con BCrypt que es una buena forma de proteger las contraseñas y tres listas donde la de recetas contiene toda la información que recojo de la receta y la de productos escaneados igual pero con los escaneos. Por último se ve otra lista de ingredientes, los cuales son lo que se eligen en la pantalla de registro y así poder mostrar recomendaciones.



```
recetasFavoritas : Array (2)
  0: Object
      _id : 715421
      titulo : "Enchilada de pollo con queso y cazuela de quinoa"
      resumen : "La cazuela de quinoa con enchiladas de pollo y queso podría ser justo …"
      imagenUrl : "https://img.spoonacular.com/recipes/715421-556x370.jpg"
      minutosPreparacion : 30
      raciones : 4
      puntuacion : 97.69800567626953
    ▸ tiposPlato : Array (5)
    ▸ dietas : Array (1)
      urlFuente : "https://www.pinkwhen.com/cheesy-chicken-enchilada-quinoa-casserole/"
      urlSpoonacular : "https://spoonacular.com/cheesy-chicken-enchilada-quinoa-casserole-7154…"
    ▸ pasos : Array (4)
      vegetariana : false
      vegana : false
      sinGluten : true
      sinLacteos : false
      puntuacionSalud : 38
      numeroMeGusta : 9912
      precioPorRacion : 262.25
      fuente : "pinkwhen.com"
```


Ilustración 7 - Ejemplo de receta guardada

```
productosEscaneadosFavoritos : Array (1)
  0: Object
      nombre : "royal"
      nombreGenerico : ""
      marca : "royal"
      cantidad : "8pcs"
      categoria : "ca:postres"
      paises : "Germany, Spain, Francia, en:morocco"
      ingredientes : "ar:contiene, ar:onters-sobre-bolsa-de-preparado-para-flan-bolsa-de-car…"
      imagen : "https://images.openfoodfacts.org/images/products/841/000/000/5349/fron…"
      energiaKcal : 96
      grasas : 1.4
      grasasSaturadas : 0.9
      carbohidratos : 18.5
      azucares : 17
      proteinas : 2.5
      sal : 0.1
      nutriscore : "c"
      idiomaOriginal : "en"
```


Ilustración 8 - Ejemplo de producto guardado

Con estos dos ejemplos se puede ver que datos recojo de cada cosa mediante sus respectivos DTOs, para mostrar la información en las recetas y productos que no he guardado utilizo la misma estructura para mostrar en el front, así que los ejemplos de llamada serían igual a estos.

3.5. Problemas/dificultades encontradas en el desarrollo del PFC y soluciones adoptadas:

- Gestión del tiempo.

Una de las principales dificultades con las que he tenido que lidiar ha sido con la gestión del tiempo, sobre todo por el comienzo tardío y la dificultad para llevar una constancia en el trabajo al tenerlo que compaginar con las practicas. Para esto, a partir del comienzo en el que por suerte le pude dedicar bastantes horas seguidas, me marqué objetivos semanales realistas y traté de centrarme en terminar completamente una funcionalidad, con pruebas incluidas, antes de pasar a lo siguiente. Este enfoque me ayudó a mantener el foco y evitar distraerme innecesariamente.

- Integración con APIs externas

Uno de los puntos más importantes para el proyecto era integrar las APIs externas: **Spoonacular** para obtener recetas, **Open Food Facts** para el escaneo de productos y **Google Translator** para traducir automáticamente las respuestas. Antes de comenza, en teoría, parecía que este apartado iba a ser relativamente sencillo, pero en la practica me encontré con respuestas muy largas con estructuras bastante complejas, datos desorganizados e incluso campos importantes vacíos.

Antes de empezar a trabajar con esta, estuve mirando APIs más especializadas o profesionales, pero eran de pago o tenían unas limitaciones muy estrictas en las versiones gratuitas y las descarté para el proyecto. Aunque las APIs elegidas no son las más perfectas, pero si ofrecen una buena funcionalidad y me permitieron avanzar en el desarrollo.

Como las respuestas estaban desorganizada y eran demasiado largas decidí adaptar esta información creando DTOs personalizados y que se gestionen en el backend. De esta forma, puedo procesar, más fácilmente las respuestas y filtrar lo que realmente quiero utilizar en el frontend.

- Escaneo de productos y falta de información

La funcionalidad de escaneo es un extra de la aplicación, pero a su vez, es una de las partes más diferenciadoras, pero también una de las que más problemas me dio. Aunque la API de Open Food Facts es útil, muchos de los productos

comunes de los supermercados españoles no aparecen o aparecen, pero con una falta muy grande de información (falta el nombre, la marca…).
A parte de lo anterior, durante el desarrollo me encontré con problemas técnicos al probar el escáner en el móvil. La versión de Expo que utilizaba (SDK 53) había sido actualizado recientemente y no era compatible con el módulo de escaneo, lo que me impedía que funcionara. La solución para esto fue bajar a la versión SDK 52, que si era compatible porque estuve trabajando con ella anteriormente y con esto pude continuar con el desarrollo.

- Validaciones y pruebas

En algunos momentos del desarrollo, especialmente cuando empezaban a juntarse varias funcionalidades, surgían pequeños errores que no eran evidentes al momento de implementarlos. En la mayoría de los casos, estos fallos se debían a que el sistema esperaba una respuesta de un tipo y acababa recibiendo otra, lo que generaba errores inesperados en la comunicación entre el backend y el frontend.
La solución fue hacer pruebas en el backend con bruno comprobando que mis endpoints funcionaban y mis DTOs estaban bien mapeados. También me apoye mucho en la consola de SprinBoot donde iba debugueando paso a paso para poder detectar donde fallaba el flujo. Siguiendo estos pasos pude ajustar bien las respuestas que quería recibir y a su vez me facilitó mucho la implementación de esta en los componentes de mi front.

3.6. Resultados obtenidos

Al finalizar el desarrollo de la aplicación puedo decir que se han cumplido gran parte de los objetivos que me propuse al principio del proyecto, si bien es verdad que algunas partes que pensaba incluir no las he podido terminar, como por ejemplo, las recetas vistas recientes (no registraba bien todas las recetas) y prefiero no ponerlo hasta terminar su desarrollo completo el balance generan ha sido muy positivo.

Dentro de los objetivos principales obtenidos están:

- Se pueden buscar recetas mediante ingredientes sueltos que se van

introduciendo manualmente de una forma visualmente atractiva. La integración con Sponacular funciona bien y no aparecen errores, las recetas las recojo con mi DTO como quiero y se traduce correctamente.

- He podido implementas la funcionalidad del escaneo de códigos de barras y utilizar la API de Open Food Facts de una forma más llamativa y divertida que escribiendo el código manualmente, además de que he podido recoger e implementar el Nutri-Score, una forma fácil de ver a primera vista si el producto es saludable.
- En el apartado de la base de datos he podido recoger e implementar todos los servicios necesarios para loguear, registrar y modificar datos del perfil correctamente.
- En la parte de backend, la que más problemas me ha dado, he podido implementar los DTO que necesitaba para mapear la información de la respuesta de Open que era demasiado extensa y muchos apartados venían vacíos y filtrarla a mi conveniencia para después consumir esos datos en el front de una forma mucho más ágil y cómoda.
- La implementación de JWT(JSON Web Token) ha sido un éxito, cosa que era impensable para mi al inicio del proyecto, tan solo era una idea más que un requerimiento que debería tener la aplicación, pero gracias a la experiencia que he adquirido en mis prácticas.
- Por último el poder haber implementado la API de Google Transalate ha hecho que la aplicación tenga un valor añadido en cuanto a funcionalidad interna, la primera idea era recibir la respuesta traducida de la API (según la documentación devolvía respuestas en Español) pero tras intentarlo y no conseguirlo, descartando la idea te traducir mediante el uso de una IA integrada debido al coste que conllevaría decidí utilizar esta API la cual me ha parecido muy útil y enriquecedora.

4. CONCLUSIONES

Cuando empecé con este proyecto tenía muy claro qué quería desarrollar y, mentalmente, los pasos que iba a seguir. Sin embargo, como suele ocurrir en los proyectos reales, surgieron imprevistos, decisiones que tuve que tomar sobre la marcha y varios ajustes de lo que tenía pensado al inicio. A medida que avanzaba, tuve que adaptarme a los problemas que iban apareciendo y reorganizar prioridades para mantener el ritmo de desarrollo.

A pesar de las dificultades, he logrado sacar adelante una aplicación funcional, práctica y enfocada en resolver una necesidad concreta, facilitar la toma de decisiones a la hora de cocinar con lo que ya se tiene en casa. También he podido incluir funcionalidades adicionales, como el escaneo de productos, que aportan valor y diferencian la aplicación de otras similares.

Este proyecto me ha servido para consolidar conocimientos del desarrollo de software, tanto en backend como en frontend, y para aprender a integrar APIs externas, trabajar con estructuras de datos más complejas. Me ha ayudado a reforzar una forma de trabajar más ordenada y orientada a objetivos.

En definitiva, ha sido un proyecto completo y exigente, con el que he podido aplicar de forma práctica lo aprendido tanto en el curso como en la empresa. Termino el proyecto con una aplicación real, funcional y útil, y con una base sólida sobre la que podría seguir desarrollando más adelante.

5. LÍNEAS FUTURAS DE TRABAJO

- Validación del correo electrónico

Implementar la validación por correo al registrarse o para recuperar la contraseña, reforzando así la seguridad de las cuentas y se garantizaría que los correos utilizados en el registro existen y son reales.

- Asistente con inteligencia artificial

Incorporar un pequeño asistente dentro de la aplicación, basado en IA que ayude al usuario si este mismo lo quisiera mediante prompts con sugerencias de recetas, consejos o dudas relacionadas con ingredientes o que lista de la compra

hacer.

- Mejora visual de la aplicación

Mejorar el diseño visual actualizándolo para que sea más moderna y atractiva para los usuarios. Mejorando la navegabilidad, la presentación de la información y el uso general al introducir las mejoras o implementaciones futuras.

- Base de datos propia de productos

Crear una base de datos interna con los productos que el usuario va escaneando, incluyendo en esta, los que ya están en la API de Open Food Facts con imágenes e información actualizada, Esta información se introduciría a través de un formulario con los datos nutricionales, nombre, marca y foto del producto.

- Misiones y recompensas

Para fomentar la participación de los usuarios y de esta forma rellenar la base de datos propia, incluiría un sistema de “misiones”, donde los usuarios reciban recompensas por escanear y subir cierto número de productos.

- Recetas vistas recientemente

En un futuro se terminará de implementar el apartado de vistas recientes, esto consistirá en una lista igual a las recomendadas pero que solo contendrá unas recetas que se visitaron y estarán ordenadas de más actual a menos.

6. BIBLIOGRAFIA

- Tecnologías:

- Spring Boot – https://spring.io/projects/spring-boot
- MongoDB Atlas – https://www.mongodb.com/cloud/atlas
- React Native – https://reactnative.dev
- Expo – https://docs.expo.dev

- APIs

- Spoonacular API – https://spoonacular.com/food-api
- Open Food Facts – https://world.openfoodfacts.org
- Google Cloud Translation – https://cloud.google.com/translate

- Herramientas de desarrollo

- Spring Tool Suite 4 (STS4) – https://spring.io/tools
- Visual Studio Code – https://code.visualstudio.com
- Bruno (API client) – https://www.usebruno.com
- Android Studio – https://developer.android.com/studio