Red social culinaria orientada a los procesadores de alimentos

Memoria de Proyecto Final de Máster

Máster Universitario en Desarrollo de Sitios y Aplicaciones Web

Área de Informática, Multimedia y Telecomunicación

Autor: Aníbal Santos Gómez

Consultor: Miguel Calvo Matalobos

Profesor: César Pablo Córcoles Briongos

Junio 2022

Créditos/Copyright

Licencias recomendadas

Abstract

Este proyecto busca crear una comunidad de usuarios que tengan en común el gusto por la cocina y el uso de procesadores de alimentos para elaborar recetas. Se pretende recoger las recetas que vayan creando los usuarios según el robot utilizado, categorizar dichas recetas y ofrecer la posibilidad de crear un listado de ingredientes con el objetivo de tener una lista de la compra exacta a las recetas que se vayan a preparar. Se podrá dar feedback a las aportaciones y se incentivará el uso de la aplicación mediante diversas estrategias como la creación de secciones o una pequeña página de producto con un blog. Esta pequeña comunidad conseguirá concentrar a personas que utilizan diariamente los procesadores de alimentos. Para su desarrollo utilizaremos un stack MERN[1] con un diseño basado en la técnica Mobile First y servicios escalables.

Palabras clave: cocina, recetas, alimentos, procesadores de alimentos, robots de cocina, red social, aplicación web.

[1] Mongodb, Express.js, React y Node.js

Abstract (english version)

This project seeks to build a community of users who have in common a love for cooking and using food processors to prepare recipes. It is intended to store the recipes created by users according to the processor used, categorize these recipes and offer the possibility of adding a list of ingredients in order to have an exact shopping list for the recipes to be prepared.

It will be possible to give feedback to the contributions and the use of the application will be encouraged through various strategies such as the creation of sections or a small product page with a blog. This small community will bring together people who use food processors on a daily basis. For its development we will use a MERN[2] stack with a design based on the Mobile First technique and scalable services.

Keywords: cooking, recipes, food, food processors, food robots, social network, web application.

[2] Mongodb, Express.js, React and Node.js

Agradecimientos

Gracias a mis **padres** (**Isabel** y **Silvio**) que me han apoyado siempre en mi camino y en este apasionante cambio de planes. A **Marta**, que soporta y admira día tras día mi pasión por seguir creciendo. A **Miguel**, **Paula**, **Alejandra**, **Vera** y **Carlos**, por el apoyo durante estos años de transformación constante. A **Sergio**, **Manuel** y a **David**, porque juntos nos empujamos.

Gracias a **Luis** y a **Javi** por sus horas de clases.

Gracias también a todo el equipo de **Iberdrola Comercial** (muy especialmente a **Laura**, **Nuria**, **Sacris**, **Sesi**, **Miguel**, **Victor** y **Edu**) la oportunidad que me dieron y con la que empecé a volar, estaré eternamente agradecido por ello. Y por supuesto no pueden faltar en estas líneas mis agradecimientos a la **comunidad Open Source**.

Y sobre todo a mi fiel compañero, **Congo**.

“Hay que creer en la posibilidad de la felicidad para ser feliz.” (Lev Tolstoi, Guerra y Paz, 1867).

Índice

Figuras y tablas

Índice de figuras

Índice de tablas

1. Introducción

En pleno 2022, nos encontramos con ante un escenario digital donde abunda la creación de contenido. Podemos ver como plataformas como Youtube, Twitch, Instagram o Linkedin, están llenas de usuarios que generan grandes cantidades de contenido. Más de 500 horas de video son subidas a YouTube cada minuto (Kinsta.com), 500 millones de personas usan Instagram diariamente (Oberlo.es).

No cabe duda de que el contenido es el gran protagonista en las plataformas y de que están abandonando los canales de comunicación tradicionales como la radio o la televisión; en virtud de las plataformas, que de alguna manera generan un contenido que atrae a otros usuarios.

Por otro lado, estamos asistiendo a un proceso de domotización de nuestros hogares, donde se están integrando sistemas de automatización de tareas y nuevos medios de comunicación y servicios. El objetivo de este proceso, no es otro que, mejorar nuestra calidad de vida, el ahorro de energía y el tiempo.

La cocina está sufriendo también una revolución, los procesadores de alimentos, comúnmente llamados, robots de cocina empezaron a ocupar un lugar en los hogares y con los años han ido tomando protagonismo no sólo a nivel doméstico sino también profesional.

Este tipo de herramientas han sido susceptibles a los cambios y las mejoras tecnológicas. Nos ahorran tiempo y concentran en una herramienta funcionalidades que se encuentran repartidas en multitud de utensilios culinarios.

Con esta pequeña situación, podemos situarnos en el contexto de que existen infinidad de personas generando contenido, en campos como el culinario. Y el principal problema que suelen tener los robots de cocina, es la comunidad que tienen detrás. Normalmente esta comunidad de usuarios se centra en el modelo de robot que tienen y en las recetas disponibles que les proporciona la marca o distribuidor, mediante su plataforma de contenido, que normalmente es privativa.

Este proyecto tratará de abrir un camino a que los usuarios de internet puedan crear libremente recetas de cocina adaptadas para sus procesadores, y tengan un sitio de referencia donde encontrar nuevas creaciones compartidas por otros usuarios. Gracias a esta comunidad, podremos beneficiarnos de recetas relativas al modelo de robot que utilicemos sin pasar largas horas buscando en internet.

Se pretende generar una comunidad viva, participativa, y además una API de datos que puede ser explotada posteriormente con otro tipo de negocios.

2. Descripción

Con este Trabajo Final de Máster se plantea la creación de una aplicación web que responda, **en primer lugar** a la problemática que se comentaba en el punto anterior; crear una comunidad de usuarios bajo una red social, que permita a cada uno de ellos crear recetas de cocina adaptadas a su robot y compartirla con el resto de miembros, agregar recetas a su colección, que les parezcan atractivas y generar una lista de la compra con la totalidad de ingredientes ya calculada.

En segundo lugar un sistema de creación de producto digital ágil, que nos permita desarrollar un ecosistema satélite a nuestro producto principal, mediante páginas de promoción, blog con contenido, y otra serie de servicios. Esta aplicación será pública y podrá acceder al contenido sin registro previo. El usuario podrá revisar las recetas, sus detalles, elaboraciones, ingredientes, compartirlas en otras redes sociales o por correo electrónico.

Los **usuarios registrados**, además podrán crear, editar y eliminar sus propias recetas, añadir a colecciones aquellas recetas que quieran cocinar o a favoritos aquellas que más les hayan gustado. Podrán consultar la lista total de ingredientes de las recetas en colecciones para obtener una lista de la compra de todo aquello que desean preparar. También podrán generar **interacciones** de algún tipo con otros usuarios, mediante el feedback directo a cada receta, comentando, siguiendo a otras personas.

De forma general podemos listar las secciones y funcionalidades que tendrá la aplicación:

- **Página de inicio**: donde encontraremos un buscador y diversas secciones destacadas, como secciones categorizadas, sección por robot, algunas categorías y las doce últimas publicaciones.

- **Página de recetas**: donde encontraremos un listado de todas las recetas ordenadas por la fecha de creación más reciente.

- **Páginas de categorías y robots**: donde podremos seleccionar una categoría para ver las recetas correspondientes a la misma.

- **Página de categoría y robot**: tendremos varios listados de recetas, las cuatro más populares, las más comentadas y las últimas publicadas.

- **Páginas de tendencias y robousers**: en estas páginas podremos ver un podium de las recetas que más han gustado a los usuarios, o un listado de los usuarios que más recetas han creado.

- **Página de detalle de receta**: aquí podremos consultar la información de la receta, ingredientes, tiempo, fotografía, comentarios, interacciones, el usuario que la ha creado. Sólo mostraremos a los usuarios registrados los pasos de creación de la receta así como las acciones de interacción sobre la receta (comentar, agregar a favoritos o guardar en colecciones).

- **Página de perfil de usuario**: aquí podremos ver las recetas que ha publicado un usuario, seguirle/dejar de seguirle.

- **Dashboard**: para usuarios registrados, donde tendremos una vista, con las opciones disponibles:

    - **Mis recetas:** donde el usuario registrado podrá editar y/o eliminar la receta que ha creado.

    - **Crear una receta:** donde el usuario registrado podrá rellenar un formulario con toda la información de la receta.

    - **Perfil**: para editar la información del usuario.

    - **Favoritos**: las recetas favoritas que vaya añadiendo el usuario.

    - **Colecciones**: recetas organizadas por colecciones, con la opción de calcular el total de todos los ingredientes por colección.

Para el desarrollo del proyecto se ha elegido un **MERN stack**, basado en Mongodb, Expres.js React y Node.js, se integrará bajo el framework **Next.js**, basado en **React** en la capa Frontend y en la capa Backend en **Express.js**, para la capa de datos, se trabajará con el cluster de **Atlas** basado en **Mongodb**.

- **Next.js**, basado en **React** para el Frontend y **Express.js** para el Backend.
- **Typescript**, para el tipado de Javascript.
- **Tailwindcss**, como framework CSS principal.
- **DaisyUI**, como librería de componentes basada en **Tailwindcss**.
- **Redux Toolkit**, para la gestión del estado de la aplicación a nivel de Frontend.
- **Cloudinary**, para la transformación y almacenamiento estático de las imágenes.
- **Vercel**, como plataforma de despliegue e integración continua.
- **Github**, como plataforma para el alojamiento del repositorio.
- **Atlas Mongodb**, como servicio cloud para la base de datos en **Mongodb**.

3. Objetivos

El presente trabajo tiene como objetivo principal, el desarrollo de una aplicación web, para su utilización como red social que permita a los usuarios la creación y gestión de recetas orientada a procesadores de alimentos. Crear una comunidad de usuarios y promover la publicación y consumo de un contenido accesible tanto por usuarios tanto registrados como no registrados.

Concretamente podemos realizar una división entre **objetivos principales**, aquellos que fundamentan la sostenibilidad del actual proyecto, y **secundarios**, aquellos que nos permiten la consecución de los primeros.

3.1 Principales

- Diseñar, desarrollar e implementar una aplicación web que nos permita construir una comunidad de usuarios y una API propia, a través de una red social

- Poner en práctica los conocimientos adquiridos a lo largo de la realización de cada una de las asignaturas del presente Máster.

- Crear un sistema de desarrollo eficaz para la elaboración de productos digitales.

3.2 Secundarios

- Determinar los requisitos para el desarrollo de la aplicación web, mediante un análisis integral del producto y las necesidades de los usuarios.

- Diseñar una aplicación responsive bajo el enfoque Mobile First, utilizando los requisitos obtenidos durante el análisis, aplicación de guías de diseño, usabilidad y experiencia.

- Desarrollo de un ecosistema de desarrollo integral orientado a la producción de bloques de código reutilizables que nos permitan ampliar el producto digital, mediante un marco de trabajo efectivo, que nos permita reducir el tiempo de producción.

- Utilizar herramientas modernas de despliegue e integración contínua, para centrarnos en el desarrollo del producto.

- Utilizar algún patrón de diseño de ingeniería del software, que nos permita minimizar los efectos secundarios en la lógica de negocio.

4. Marco teórico/Escenario

4.1. Los medios de comunicación

En los medios clásicos de comunicación, el receptor de esa comunicación es un receptor parcialmente pasivo. Llamamos **nuevos medios de comunicación** a aquéllos que se ponen al servicio del cliente y que permiten la interacción con el mismo, por medio de un canal que está a disposición tanto de la empresa como del cliente.

Understanding Media (McLuhan, 1964) sienta las bases para el estudio y la comprensión de los medios de comunicación de masas. McLuhan habla de que todos los medios de comunicación:

Son una prolongación humana, física o psíquica. Entendiendo medio en su aceptación más amplia, no sólo las creaciones tecnológicas sino las facultades humanas: la palabra hablada, la escritura, la mano, el puño, el pie y la piel. Y todo medio mecánico o tecnológico es, a su vez, extensión de las facultades humanas.

El medio es el que condiciona el mensaje. Se trata de poner énfasis en el efecto de la forma, más que en el efecto del contenido. Medio y mensaje funcionan simbióticamente puesto que uno suele contener una representación del otro. El contenido de la televisión es la electricidad, que contiene, a su vez, la imagen. El contenido del cine es la fotografía, que contiene la pintura, etc. [3]

Conocer las características del medio y la forma en que se transmite el mensaje, al mismo tiempo que las características físicas que potencia el ser humano y la implicación cognitiva en el uso de los nuevos medios de comunicación, se convertirá en el germen justificativo de este proyecto.

El uso del teléfono ha evolucionado hasta convertirlo en un objeto del que se depende constantemente: para comunicar que se está llegando, para descargarse sonidos personalizados, para narrar la vida en el momento presente.

El uso de Internet por las corporaciones ha sido espectacular y los motivos son muy diversos. Por ejemplo, muchas empresas creen que Internet es la respuesta para conquistar el mundo y llegar a segmentos de clientes que no eran posibles para una organización.

[3] Sempere, P. *McLuhan en la era de Google: Memorias y profecías de la Aldea Global*. Madrid: Editorial Popular, 2007.

El conocimiento del ser humano en cuanto al uso de Internet y de las nuevas tecnologías se convirtió en el mantra de muchas de estas nuevas empresas nacidas en Internet, mientras que, para las empresas tradicionales, no se innovó internamente en cuanto a la comprensión del cliente.

4.2. El marketing social

En los últimos años, el marketing social se ha aplicado al ahorro de energía, a dejar de fumar, a una conducción de vehículos más segura. Muchas veces se trata de convencer a la gente de que adopte hábitos nuevos. El proceso de difusión queda definido por Rogers como:

Un proceso en el que una innovación se comunica a través de determinados canales de comunicación durante un determinado tiempo a los miembros de un sistema social.[4]

El proceso de decisión sobre una innovación es una actividad que representa una búsqueda y procesado de información, en la que un individuo se encuentra motivado para reducir su incertidumbre sobre las ventajas y desventajas de dicha innovación. Las innovaciones que proporcionan mayores ventajas, que son compatibles, que se pueden comprobar, observar y que son menos complejas se adoptarán más rápidamente que otras.

La comunicación es un proceso por el cual los participantes crean y comparten información entre ellos, para alcanzar una comprensión mutua.

4.3. Las redes sociales

En primer lugar deberíamos señalar que, la tecnología *parece haber enraizado profundamente en nuestro sentido común y actualmente es fácil detectar su presencia implícita o explícita en la mayor parte de los discursos u opiniones de los medios de comunicación*[5].

Asumiendo este discurso y los puntos anteriores en nuestro marco teórico, respecto a los canales de comunicación y su respectiva evolución, no cabe duda, que podemos resaltar, como las herramientas tecnológicas han cambiado la forma en la que nos comunicamos y recibimos la información. Han entrado en esfera nuevos agentes y canales de comunicación, y la proliferación de los mismos ha generado una pluralidad a nivel exponencial tanto de emisores como receptores.

[4] Rogers, E. M. *Diffusion of Innovations*. New York: Free Press, 2003.

[5] Aibar, E. *La visión constructivista de la innovación tecnológica. Una introducción al modelo SCOT.* Barcelona: UOC, 2006.

Las redes sociales propician la interacción de miles de personas en tiempo real. Las redes sociales no son otra cosa que máquinas sociales diseñadas para fabricar situaciones, relaciones y conflictos. No hay duda de que suponen un nuevo agente comunicativo.

Definidos claramente estas bases y antecedentes, podemos determinar que la herramienta red social, sirve como canal de comunicación en una aplicación cuya misión es estar viva, crecer compartir conocimiento y tener una autonomía interna, controlada, pero menos privativa que las plataformas de empresas ligadas a los procesadores de alimentos.

Las plataformas donde proliferan amplias cantidades de recetas creadas por usuarios, son **Instagram** y **Youtube**, donde podemos ver influencers que se dedican a promocionar recetas de cocina con algún tipo de recompensa comercial. Fuera de estas encontramos algunos ejemplos más centrados en la cocina en sí, como serían:

- **Funcook,** es una red social donde se pueden conseguir cientos de recetas que la gente comparte. Los usuarios explican paso a paso los ingredientes y utensilios que han utilizado para crearlas. Además, estas recetas están divididas en secciones según del tipo o categoría que sean: ensaladas, postres, pastas, carne etc.

  Se pueden puntuar, valorar y comentar los platos de los demás. Permite ver las recetas mejor puntuadas y los usuarios mejor valorados, puedes seguirles, guardar sus mejores recetas y preguntarles dudas. También puedes elegir los ingredientes con los que quieres cocinar o no, para filtrar las recetas que más se adapten a tus gustos. Esta red social incluye una lista de la compra para que no olvides ningún ingrediente en tu próxima compra.

- **Cookpad,** es una aplicación utilizada por personas de todo el mundo que comparten el gusto por cocinar. Existen unas 60,000 recetas de miles de chefs y cocineros aficionados. Esta aplicación se desarrolló por primera vez en Japón y presentaba una variedad de recetas de Cookpad en japonés. Se extendió a Occidente a lo largo de los años, aunque tiene pocos seguidores en Estados Unidos. Está dirigida a todos los chefs caseros, sin importar en qué parte del mundo se encuentren. Podemos buscar recetas, publicarlas y conectarnos con otros usuarios.

Respecto a las comunidades que giran entorno a los procesadores de alimentos, están han ido generando en mucha medida por el tipo o marca, como es el claro ejemplo en:

- **Recetario Thermomix**, es la comunidad que rodea al procesador Thermomix, donde encontramos recetas creadas por la marca y un foro que congrega a todos los usuarios que poseen un robot de cocina Thermomix.

- **Recetas Mambo**, esta plataforma es creada por Cecotec, propietaria del procesador Mambo, aqui podemos ver las recetas que crea la propia marca, categorizadas y en cada detalle la preparación de la receta y comentarios que pueden dejar usuarios mediante el envío de un formulario sencillo sin registro previo.

4.4. Procesadores de alimentos

A lo largo del presente proyecto nominamos al robot de cocina o procesador de alimentos innumerables veces. ¿Pero qué es exactamente un procesador de alimentos? ¿Qué lo hace tan atractivo para multitud de usuarios?

El procesador de alimentos es un electrodoméstico usado en multitud de tareas repetitivas durante el proceso de preparación de la comida. Son parecidos a las licuadoras en muchos aspectos, pero difieren en las cuchillas y discos intercambiables en lugar de una cuchilla fija, lo que los hace instrumentos versátiles para elaborar multitud de platos diferentes.

El origen de estos electrodomésticos tiene su origen en un procesador inventado por Pierre Verdun, un chef francés que al ver como sus trabajadores pasaban horas y horas cortando, rallando y mezclando ingredientes decidió inventar el **Robot-Coupe** en 1961, fue el primer robot de cocina.

Posteriormente **Vorwerk** desarrollaría el famoso procesador de alimentos **Thermomix**, capaz de cortar, mezclar, amasar, cocer a fuego lento, hervir y cocer al vapor. Este conocido procesador de alimentos cuenta ya con diez generaciones a sus espaldas, desde 1961 hasta 2019 se han desarrollado diez versiones renovando dicho electrodoméstico y adaptándolo a mejoras técnicas incluyendo su digitalización.

Por supuesto este electrodoméstico no ha sido siempre para todos los públicos, y la última década ha supuesto el inicio de la “democratización” de este tipo de productos. Actualmente el mercado ha sido inundado por diferentes marcas que han ido sacando versiones más económicas.

Las principales marcas en este panorama, son **Vorwerk** con Thermomix, **Moulinex** con Maxichef, **Taurus** con Mycook, **Kenwood** con KCook, **Cecotec** con Mambo, y hasta la conocida empresa de supermercados **Lidl** con su Monsieur Cuisine.

La mayor parte de estos aparatos poseen funciones muy parecidas, y destacan en detalles como la conectividad y la transformación digital que van adquiriendo versión tras versión. Normalmente suelen abarcar las funciones de cocer al vapor, hornear, freír, amasar masa, trocear, preparar purés y sopas, realizar recetas y menús elaborados de forma ágil, integración de báscula y temporizadores. Y actualmente presentan conectividad WiFi, sistemas de aplicaciones desarrolladas por cada fabricante y un consumo energético eficiente.

Así como otros electrodomésticos nos ahorran realizar tareas repetitivas, es muy probable que el robot de cocina en la próxima década sea un electrodoméstico de lo más común y esencial dentro de nuestros hogares. Es por eso que es uno de los principales pilares y motores en el escenario de la revolución tecnológica de los hogares y de este proyecto.

5. Contenidos

A continuación se citan los contenidos de nuestra aplicación, será un conjunto de elementos visuales sobre los que trabajaremos a lo largo de la usabilidad, y nos permitirán crear un marco estandarizado sobre el cual evolucionar el producto. Estos elementos serán los siguientes:

- **Navbar**: nos mostrará una barra común de navegación de la aplicación a las diferentes rutas o secciones.

Figura 1: Contenidos - Navbar

- **Footer**: puede utilizarse como una forma de intentar convencer a los usuarios para que sigan navegando en la aplicación.

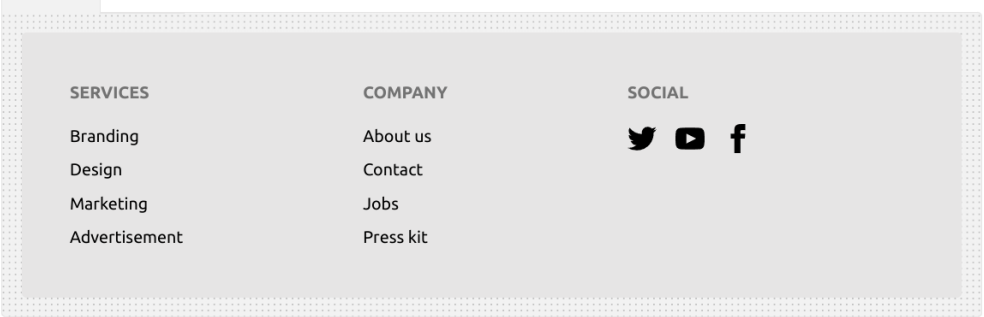


Figura 2: Contenidos - Footer

- **Card**: se utilizará para mostrar entradas de datos e información a usuarios en múltiples formas y contextos, como por ejemplo, perfiles de usuario o recetas.

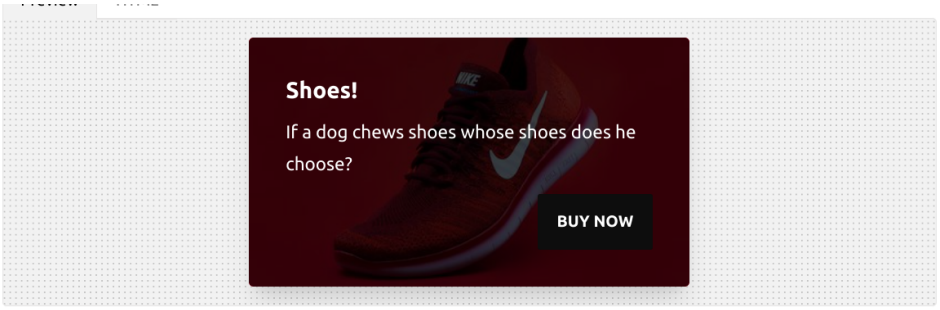


Figura 3: Contenidos - Card 1

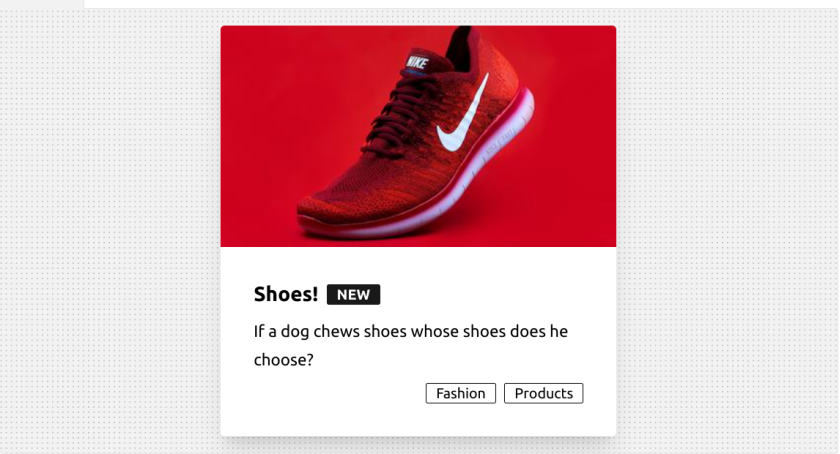


Figura 4: Contenidos - Card 2

- **Formularios**: los utilizaremos para recopilar información de los usuarios utilizando inputs, checkboxes, radiobuttons o text areas.

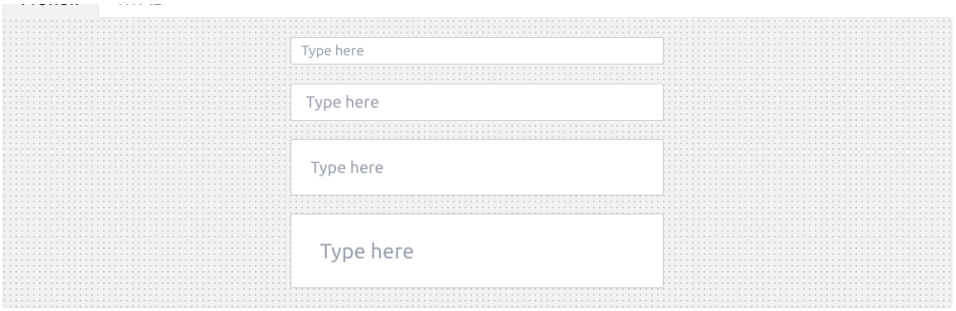


Figura 5: Contenidos - Input

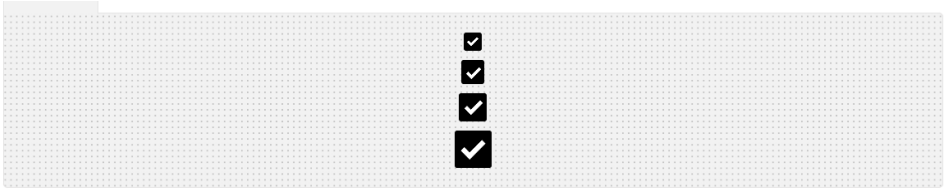


Figura 6: Contenidos - Checkbox

- **Avatar**: se utilizan para mostrar una representación en miniatura de una persona.

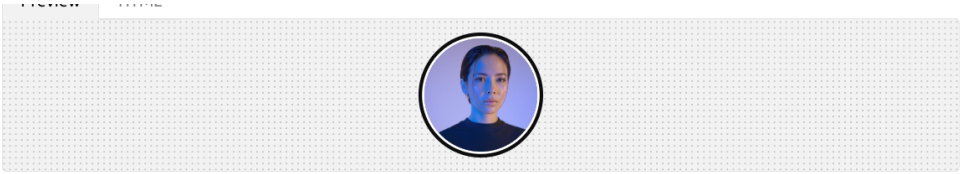


Figura 7: Contenidos - Avatar

- **Badge**: se utilizan para informar al usuario del estado de determinados datos. En nuestro caso podremos utilizarlos para mostrar las categorías de una receta por ejemplo.

Figura 8: Contenidos - Badge

- **Tooltip**: mostrar contenido adicional al pasar el ratón por encima de un elemento. Los utilizaremos para añadir información útil o explicativa para el usuario.

Figura 9: Contenidos - Tooltips

- **Dropdown**: nos sirven para mostrar un menú al hacer clic en un elemento como un botón.

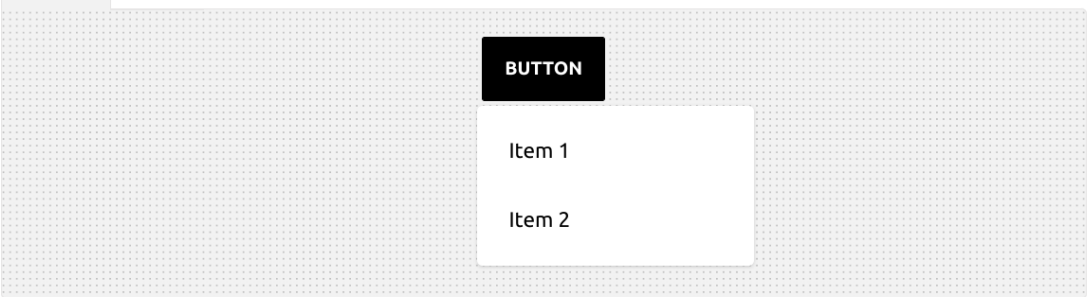


Figura 10: Contenidos - Dropdown

- **Drawer**: será una barra de navegación lateral que utilizaremos en la vista de gestión del usuario.

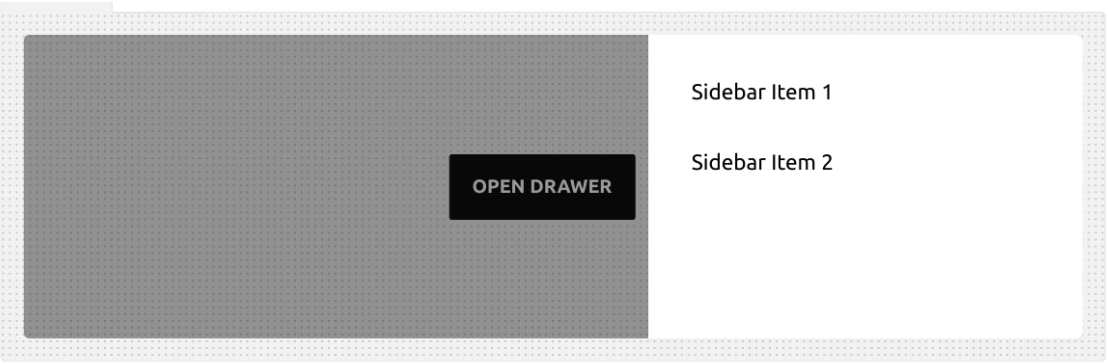


Figura 11: Contenidos - Drawer

- **Collapse**: se utiliza para mostrar y ocultar el contenido, en situaciones en las cuales el contenido es muy extenso, como por ejemplo a la hora de crear una publicación, donde tendremos formularios un poco extensos.

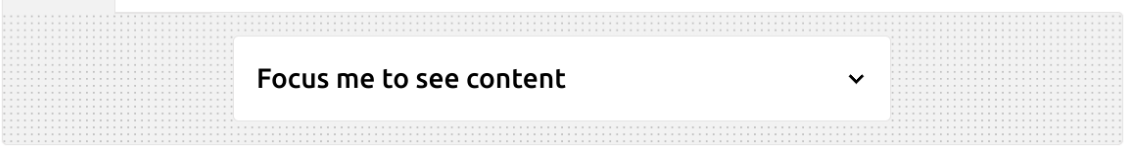


Figura 12: Contenidos - Collapse

- **Breadcrumbs**: nos ayudarán a navegar en la aplicación cuando la jerarquía sea más profunda.

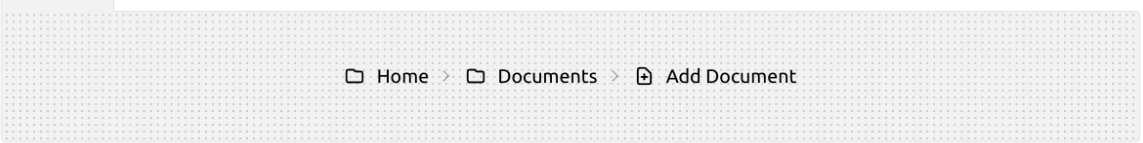


Figura 13: Contenidos - Breadcrumb

- **Alertas**: muestra información contextual a sus usuarios.

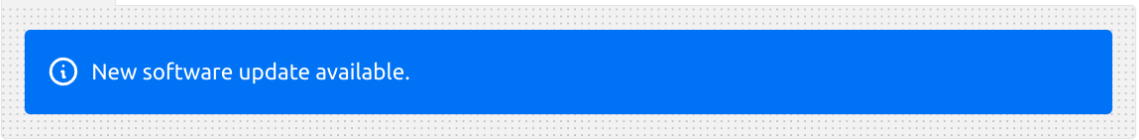


Figura 14: Contenidos - Alerta

- **Notificaciones**: las utilizaremos para que el usuario conozca si el estado de una acción que ha realizado es exitoso o fallido.

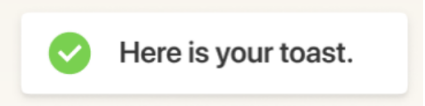


Figura 15: Contenidos - Notificaciones

- **Tablas**: mostraremos una lista de datos en un formato de tabla.

	NAME	JOB	FAVORITE COLOR
1	Cy Ganderton	Quality Control Specialist	Blue
2	Hart Hagerty	Desktop Support Technician	Purple
3	Brice Swyre	Tax Accountant	Red

Figura 16: Contenidos - Tablas

- **Tabs**: nos permitirán navegar en una misma página entre vistas diferentes.

Figura 17: Contenidos - Tabs

6. Metodología

Respecto a la forma de trabajar o metodología de trabajo, este proyecto de fin de Máster, se basará en la implementación de Agile o metodologías ágiles, que describiremos a continuación.

El desarrollo ágil de software envuelve un enfoque para la toma de decisiones en los proyectos de software, que se refiere a métodos de ingeniería del software basados en el desarrollo iterativo e incremental, donde los requisitos y soluciones evolucionan con el tiempo según la necesidad del proyecto.[6]

Ahora bien, las metodologías ágiles nos permiten adaptar la forma de trabajo a las condiciones del proyecto, consiguiendo flexibilidad en la respuesta para amoldar el proyecto a las circunstancias del entorno.

Las ventajas que nos proporciona este tipo de forma de trabajar, son:

- **Mejor calidad del producto**: se fomenta la integración, comprobación y mejora continúa de las propiedades del producto.

- **Satisfacción del cliente**: mediante varias demostraciones y entregas, el cliente vive a tiempo real las mejoras introducidas en el proceso.

- **Motivación en el trabajo**:a través del trabajo autogestionado se facilita el desarrollo de la capacidad creativa y la innovación.

- **Trabajo colaborativo**: la división del trabajo, permite una mejor organización del mismo.

- **Utilización de métricas relevantes**: gracias a la división en pequeñas y fases podemos ser más conscientes de lo que está sucediendo.

- **Control y capacidad de predicción**: La oportunidad de revisar y adaptar el producto a lo largo del proceso nos permite ejercer un mayor control sobre el trabajo y esto permite mejorar la capacidad de predicción en tiempo y costes.

- **Reducción de costes**: se elimina prácticamente la posibilidad de fracaso absoluto en el proyecto, porque los errores se van identificando a lo largo del desarrollo.

[6] Wikipedia, *Desarrollo ágil de software*, 2022

Existen diversas metodologías ágiles y en este proyecto utilizaremos dos para establecer el siguiente punto de esta memoria. Nos centraremos en:

- **Kanban**: es una estrategia que consiste en la elaboración de un cuadro o diagrama en el que se reflejan tres columnas de tareas; pendientes, en proceso o terminadas. En este cuadro estableceremos todas las tareas que integrarán las diversas fases del desarrollo. Con ello evitaremos la repetición de las tareas. Sus ventajas fundamentales son: planificación de las tareas, la mejora en el rendimiento del trabajo, métricas visuales y plazos de entrega continuos.

- **Scrum**: se basa en una estructura de desarrollo incremental o iterativo; cualquier ciclo de desarrollo del proyecto se desgrana en pequeños proyectos divididos en distintas etapas: análisis, desarrollo y testing. En la etapa de desarrollo encontramos las iteraciones o sprints, que son, entregas regulares y parciales del producto final. Scrum nos permite abordar proyectos complejos que exigen flexibilidad, es probable que podamos fallar en algún sprint, en este caso analizaremos el porqué y buscaremos otra solución flexible que nos permita adaptarnos a las circunstancias y no bloquear el flujo de trabajo.

En los dos próximos puntos (**planificación** y **proceso de trabajo**), daremos forma a los hitos y fases de desarrollo del proyecto, mediante el uso de estas dos técnicas. Para ello catalogaremos las tareas en un **roadmap** u hoja de ruta, con plazos de entrega donde tendremos los siguientes procesos:

- **Epics:** son una forma de agrupar tareas y planificar a un nivel global, incluyendo las fases de análisis, desarrollo y pruebas.

- **Sprints**: son empujones con límite de tiempo para completar un conjunto de tareas, en las fases de desarrollo.

- **Tareas**: son las acciones que conforman los epics y los sprints

Las tareas, son los átomos que compondrán nuestros sprints en las fases de desarrollo y los epics a nivel más global, incluyendo otras fases que no conforman el desarrollo, como el análisis, las pruebas o la agrupación de bugs o errores o mantenimiento y/o soporte. Tendrán como mínimo las siguientes propiedades:

- Título.
- Descripción.
- Fecha de creación.

- Tipo: Epic o tarea.
- Estado: Sin empezar, en progreso, completada, o bloqueada.
- Sprint: sprint al que pertenece.
- Prioridad: Alta, media, baja.
- Timeline: inicio y fin del periodo de entrega.

7. Planificación

La planificación del Trabajo Final de Máster, como comentábamos en el punto anterior está dividida en cuatro **epics**, cada **epic** corresponde a las diferentes PECs establecidas en el calendario académico del semestre, donde constan los **sprints** (si los hubiese) y el conjunto de tareas entregables y sus respectivas fechas.

7.1. Epics

Epic	Título	Timeline
PEC 1	Propuesta y definición formal del proyecto	16/02/2022 - 01/03/2022
PEC 2	Fundamentación y análisis del software	02/03/2022 - 30/03/2022
PEC 3	Desarrollo, POC y MVP	31/03/2022 - 08/05/2022
PEC 4	Entrega final del proyecto, documentación y publicación	09/05/2022 - 06/06/2022

Tabla 1: Resumen de entregas del Trabajo de Fin de Máster

7.2. Sprints y tareas

En la siguiente tabla se presentan los sprints de desarrollo, así como las tareas que no se engloban en dichos sprints, pero que sí forman parte de los **epics** anteriormente mencionados:

Epic	Título	Sprint	Timeline
PEC 1	Elección del tema	-	16/02/2022 - 17/02/2022
	Propuesta formal y elección de tecnologías	-	18/02/2022 - 18/02/2022
	Fundamentación teórica: introducción, descripción, objetivos, marco teórico	-	19/02/2022 - 23/02/2022
	Fundamentación teórica: contenidos, metodología, planificación, proceso de trabajo	-	24/02/2022 - 28/02/2022
PEC 2	Definición de la arquitectura y plataforma de desarrollo	-	01/03/2022 - 07/03/2022
	Diseño del modelo y entidades de datos	1	07/03/2022 - 12/03/2022
	Usabilidad: Conceptualización	2	12/02/2022 - 15/03/2022
	Usabilidad: diseño de la experiencia	3	15/03/2022 - 20/03/2022
	Usabilidad: diseño de la interacción	4	20/02/2022 - 28/03/2022
	Desarrollo: creación de repositorios de la aplicación, primeros componentes base.	5	28/03/2022
	Despliegues: Entorno de desarrollo e integración continua	5	28/03/2022
	Documentación de la memoria del trabajo	-	28/03/2022 - 31/03/2022

PEC 3	Backend: Autenticación y registro de usuarios	6	31/03/2022 - 04/04/2022
	Backend: Creación de capa API Rest y servicios crud	7	04/04/2022 - 11/04/2022
	Frontend: Estructuración inicial y primeras integraciones con backend	8	11/04/2022 - 16/04/2022
	Frontend: Estructuración de rutas, vistas y guards	9	13/04/2022 - 16/04/2022
	Frontend: integración de funcionalidades	10	16/04/2022 - 27/04/2022
	Desarrollo: Revisión y refactorización de Frontend y Backend	11	27/05/2022 - 05/05/2022
	MVP: Despliegue y pruebas de usuario	12	05/05/2022 - 08/05/2022
	Documentación de la memoria del trabajo	-	05/05/2022 - 08/05/2022
PEC 4	Deuda técnica y refactorización	13	09/05/2022 - 25/05/2022
	Documentación de la memoria del trabajo	-	21/05/2022 - 25/05/2022
	Presentación académica	-	26/05/2022 - 28/05/2022
	Presentación pública	-	29/05/2022 - 31/05/2022
	Video de presentación	-	01/06/2022 - 03/06/2022
	Autoinforme de evaluación	-	04/06/2022 - 06/06/2022

Tabla 2: Sprints, tareas y planificación del Trabajo de fin de Máster

7.3. Diagrama de Gantt

En el siguiente diagrama de Gantt podemos ver la planificación mencionada en el punto anterior:

Figura 18: Diagrama de Gantt - Planificación

8. Proceso de trabajo

8.1 MVP[7]

El producto mínimo viable será una aplicación web que contendrá las siguientes funcionalidades:

- Autenticación y registro de usuarios.
- Edición y eliminación del usuario previamente registrado.
- Navegación entre las diferentes vistas.
- Creación, edición y eliminación de recetas de los usuarios.
- Búsqueda por nombre, ingrediente o categoría de la receta.
- Interacciones sociales “like” sobre la receta de un usuario.
- Comentarios sobre una receta.
- Seguir y dejar de seguir a un usuario.
- Crear colecciones de recetas.
- Calcular el total de ingredientes de las recetas añadidas previamente a una colección.

8.2. Sprints

Las fases del proceso de desarrollo estarán contenidas por los trece sprints, mencionados en el punto anterior de la planificación:

1. Diseño del modelo y entidades de datos.
2. Usabilidad: Conceptualización.
3. Usabilidad: diseño de la experiencia.
4. Usabilidad: diseño de la interacción.
5. Desarrollo: creación de repositorios de la aplicación, primeros componentes base. Despliegues: Entorno de desarrollo e integración continua.
6. Backend: Autenticación y registro de usuarios.
7. Backend: Creación de capa API Rest y servicios crud.
8. Frontend: Estructuración inicial y primeras integraciones con backend.
9. Frontend: Estructuración de rutas, vistas y guards.
10. Frontend: integración de funcionalidades.
11. Desarrollo: Revisión y refactorización de Frontend y Backend.
12. MVP: Despliegue y pruebas de usuario.
13. Deuda técnica y refactorización.

[7]Producto mínimo viable

9. Arquitectura de la aplicación

En este capítulo justificamos la arquitectura de la aplicación atendiendo a dos puntos muy importantes, los **patrones de diseño** y las **tecnologías** que utilizaremos para llevar a cabo nuestro desarrollo.

9.1. Patrones de diseño

En este punto trataremos de definir y resumir la adopción de un patrón de diseño para nuestra aplicación. Si bien existen infinidad de patrones, casi como problemas a resolver, existe uno que nos permite encapsular la lógica de negocio, aislandola de dependencias externas. Estamos hablando del patrón **Clean Architecture**. Realmente no es un patrón, sino más bien un conjunto de principios, como explica *Jorge Sánchez Fernández* en su artículo acerca de Clean Architecture:

[...] cuya finalidad principal es ocultar los detalles de implementación a la lógica de dominio de la aplicación. De esta manera mantenemos aislada la lógica, consiguiendo tener una lógica mucho más mantenible y escalable en el tiempo.[8]

Resumidamente podemos hablar de las siguientes capas en **Clean Architecture**[9]:

- Entidades: donde se encuentra la lógica de negocio empresarial que existiría aunque no tengamos una aplicación para automatizar procesos.

- Casos de uso: representan la lógica de aplicación, que existe principalmente debido a la automatización de procesos mediante la aplicación y es inherente a cada aplicación.

- Adaptadores: se van a encargar de transformar la información como se entiende y es representada en los detalles de implementación o frameworks, drivers a como la entiende el dominio (entidades y casos de uso).

- Frameworks y drivers: serían los detalles de implementación, todos aquellos frameworks, librerías que se suelen utilizar en una aplicación para mostrar o almacenar información (como por ejemplo React, Vue, etc…).

En la siguiente figura podemos observar las capas mencionadas y cómo éstas no pueden depender de los exteriores, pero los exteriores tienen que poder comunicarse entre ellos.

[8] J.S. Fernández. *¿Por qué utilizo Clean Architecture?*, XurxoDev, 2016.
[9] R. C. Martin. *Clean Architecture*, The Clean Code Blog, 2022.

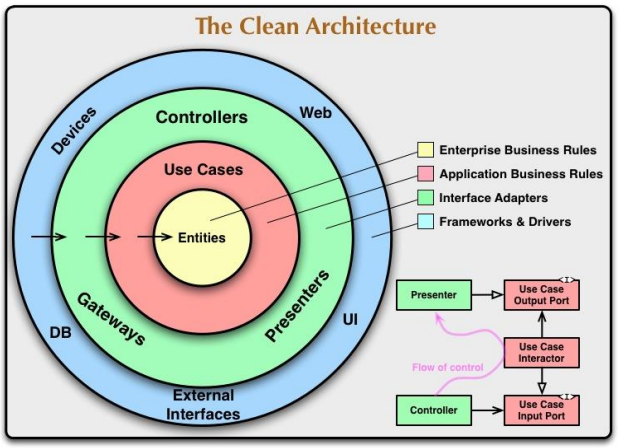


Figura 19: Clean Architecture

9.2. Tecnologías

En este punto se definirá la arquitectura respecto a las tecnologías, donde se detalla la comunicación entre las mismas y el papel que tienen en los diferentes procesos. Para poder llevar a cabo el desarrollo completo de este proyecto, utilizaremos las herramientas:

Figura 20: Nextjs

- **Next.js**, es el core de nuestra aplicación, Nextjs nos proporciona un ecosistema fullstack para el desarrollo frontend y backend.

  Por un lado tendremos las diferentes pantallas que se desarrollarán bajo **React** en la parte frontend; y por otro lado tendremos todo el sistema de servicios API Rest, que se desarrolla internamente bajo **Expressjs** en **Node** dentro del propio Next.js.

  Esto nos permite ahorrar costes para el almacenamiento de nuestro proyecto y desarrollar ágilmente toda la lógica fullstack bajo esta herramienta, conocida como *The Web SDK*.

Figura 21: Typescript

- **Typescript**, como realizaremos un desarrollo fullstack en Javascript, introduciremos Typescript, para poder tener tipado de datos y modelización de objetos. Esta herramienta nos permitirá tener un desarrollo más estable y evitar multitud de errores por la falta de tipado en la que Javascript es propenso.

Figura 22: Tailwindcss

- **Tailwindcss**, es un framework de CSS orientado a listas de clases, lo cual nos permite individualizar la maquetación de los componentes que integrarán la interfaz.

Figura 23: DaisyUI

- **DaisyUI**, esta librería de componentes basada en **Tailwindcss**, nos permite montar componentes de forma ágil para ir construyendo nuestra interfaz, y como está basada en Tailwindcss podremos y/o customizar cada componente de la manera que más nos convenga. De esta manera podremos partir de unos componentes ya diseñados y acomodarlos a nuestras exigencias del proyecto.

Figura 24: Redux Toolkit

- **Redux Toolkit**, nos permite mantener la organización del estado de la aplicación a nivel de desarrollo Frontend y poder organizar mejor la información que utilizaremos entre componentes. Esto nos permitirá compartir información a lo largo de toda la aplicación y mantener el código de forma más clara.

Figura 25: Cloudinary

- **Cloudinary**, es una servicio que nos permitirá transformar y almacenar las imágenes que los usuarios utilicen en nuestra aplicación. Existen diversas alternativas para almacenar archivos estáticos, pero Cloudinary nos ofrece la transformación de imágenes lo cual nos resuelve el problema de optimización de este tipo de archivos.

Figura 26: Vercel

- **Vercel**, es una plataforma que nos permite desplegar aplicaciones orientadas a proyectos Frontend. Tenemos plena integración con Github y podemos configurar ramas para el despliegue automático tanto para producción como para versionado. Vercel es la creadora de Next.js por lo tanto prescindiremos de tener un servidor dedicado para Node.js, teniendo en una misma plataforma la integración completa de nuestra aplicación.

Figura 27: Github

- **Github:** es la plataforma donde albergamos los diferentes repositorios y que conectaremos con **Vercel** para poder desplegar nuestro producto. Utilizamos Git y Gitflow como control de versionado del código.

Figura 28: Atlas MongoDB

- **Atlas MongoDB:** para la persistencia de los datos de nuestra aplicación utilizaremos **MongoDB** y para servir la base de datos utilizaremos **Atlas Cloud** como servicio, y así no tener servidores propios y perder tiempo en configuraciones innecesarias.

9.3. Arquitectura

Como comentábamos en el punto de patrones de diseño, tendremos una arquitectura basada en capas, para evitar la dependencia entre ambas y poder desligar en la medida de lo posible la lógica de negocio.

Por un lado tendremos las capas de la aplicación, y por otro los servicios que sirven o mantienen a la misma.

Atlas y **Cloudinary** están comunicados con nuestra aplicación. El primero es el cluster de **Mongodb**, mientras que Cloudinary, nos sirve como almacenamiento de estáticos. **Github** alojará el repositorio de nuestra aplicación de forma remota, y **Vercel** será el encargado de la integración continúa, los despliegues, y la publicación.

En el lado de las capas tendremos:

- **Base de datos**: queries y conectores con Mongodb para las operaciones de persistencia de los datos de nuestra aplicación.

- **API**: esta capa conectará nuestra capa de base de datos con la capa de servicios de la aplicación. Está basada en Express.js, pero en nuestro proyecto se encuentra integrada plenamente en Next.js, ya que éste framework nos provee de un servidor Express plenamente integrado. El router del servidor se genera automáticamente añadiendo las dependencias y ficheros de forma dinámica, por lo que básicamente programaremos los controladores que serán los encargados de realizar las operaciones necesarias, como validaciones, comunicaciones con la base de datos, subida de estáticos o envío de correos.

- **Servicios**: en la siguiente capa simplemente tendremos un conjunto de funciones estructurado según las funcionalidades de la lógica de negocio. Estos servicios comunican los endpoints de la API, con el store del Frontend. Están basados en fetch, por lo que trabajaremos con promesas a lo largo de toda la aplicación. Lo normal es que los conectemos al store, pero en casos puntuales podemos consumirlos de forma independiente en algún componente.

- **Store**: en esta capa gestionaremos el estado global de la aplicación, para ello utilizaremos **Redux Toolkit**. Será la encargada de conectar la mayor parte de los servicios con la interfaz, enriqueciendo de datos la misma. El diseño del store está basado en features orientadas a la lógica de negocio, al igual que los servicios. Definiremos pequeños estados que nos darán la información clave que luego consumirán los componentes de la interfaz.

- **UI**: aquí tendremos la capa de la interfaz gráfica, que será la encargada de mostrar al usuario todo aquello que sucede en nuestra plataforma. Está basada en componentes de **React.** Por norma general tendremos componentes Página que Next.js gestiona y con los que establecerá un índice para crear de forma automática el router de la aplicación. Estos componentes Página, estarán compuestos de componentes comunes y específicos. En la arquitectura frontend, tendemos a atomizar lo máximo posible todos los componentes, para no sobrecargarlos de lógica.

En la siguiente figura podemos ver la estructuración completa de la aplicación y los servicios que orbitan entorno a ella:

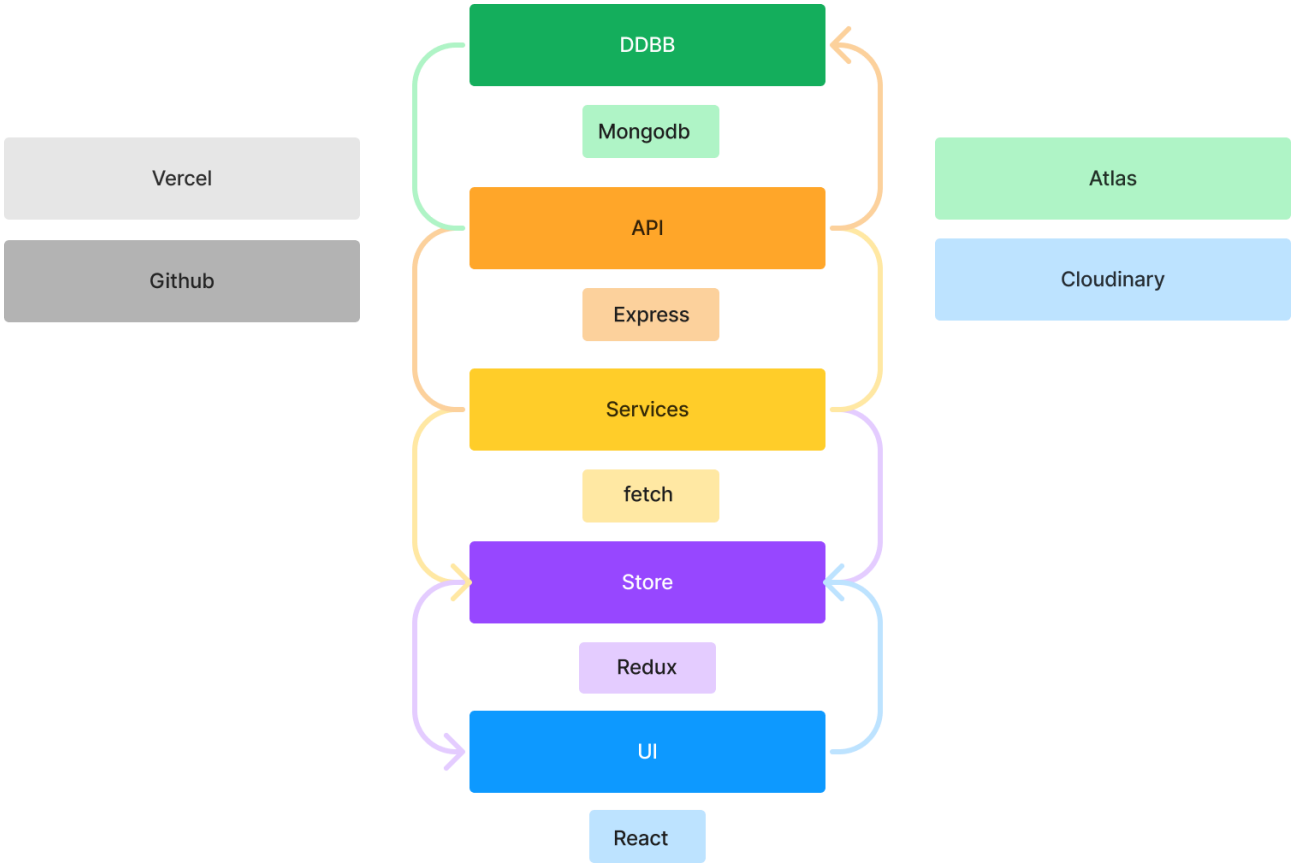


Figura 29: Arquitectura

A continuación desgranamos de forma ejemplificativa la arquitectura de la base de datos, el backend y el frontend, como bloques más genéricos:

- **Base de datos**: el modelo de base de datos estará compuesto por colecciones de **Mongodb**:
    - **Accounts**: es la colección de la cuenta de usuario, donde almacenaremos la información personal del usuario.
    - **Blenders**: es la colección de robots de cocina.
    - **Bookmarks**: es la colección de marcadores de recetas de un usuario, contendrá todas las recetas que vaya marcado el usuario, y todas las colecciones de recetas que vaya creando el usuario, que serán almacenadas en la colección collections.
    - **Categories**: es la colección de las categorías que puede tener una receta.
    - **Collections**: es la colección de las recetas que un usuario crea, tiene un Bookmark asignado que pertenece a una cuenta de usuario.
    - **Comments**: son los comentarios que recibe una receta, contienen el autor de la misma, el id de la receta y el bloque de texto.
    - **Recipes**: son las recetas creadas por los usuarios y los bloques principales de información. Tienen un id de cuenta de usuario asignado para poder recuperar la pertenencia de la misma.
    - **Users**: son las colecciones de usuarios de la aplicación, en ellas se almacenan, el nombre de usuario, la contraseña cifrada, el correo electrónico, y el id de cuenta.

En la siguiente figura podemos ver la relación del modelado de la base de datos:

Figura 30: Arquitectura de la base de datos

- **Backend**: como ya comentamos en el punto explicativo de las capas, en este grupo tendremos, los conectores con base de datos, la api rest y la capa de los servicios, a nivel ejemplificativo, tendremos las siguientes dependencias del proyecto que engloba el bloque del backend:

    - **lib/api/db**: son los registros con la base de datos, organizados por features.

    - **lib/api/auth**: instancia de la librería **Passport.js** para la gestión de la autenticación.

    - **lib/api/issues**: conector con **Notion,** para el sistema de gestión de incidencias.

    - **lib/api/mail**: conector con **Nodemailer** para el sistema de correos.

    - **lib/api/middlewares**: manejadores y conectores de auth, base de datos y sesiones.

    - **lib/api/nc**: manejador de errores para los controladores.

    - **lib/api/schemas**: esquemas de validación para los controladores.

    - **lib/services**: capa de servicios conectores.

    - **lib/models**: interfaces utilizadas para tipar los objetos de la aplicación.

    - **pages/api**: API rest, son los controladores decorados con las librerías y conectores. Integrados en el sistema de Next.js con Express. (Esta parte no puede ser extraída del sistema de organización del framework).

- **Frontend:** respecto al bloque que contiene lo relativo al Frontend, tendremos el store, y componentes basados en React, los componentes podrán ser de dos tipos, componentes página, que representarán todas aquellas vistas de la aplicación; y componentes bloque, que serán todos aquellos que representan pequeñas piezas de la interfaz gráfica, como botones, barras de navegación, modales, avatares, entre otros. En cuanto a las dependencias del proyecto podemos seguir la siguiente estructura:

    - **pages/***: a excepción de **/pages/api**, el resto de dependencias harán referencia a todas los componentes de tipo página enrutados por Next.js. Estos nos permitirán crear todas las vistas con su correspondiente router.

- **components/common**, aquí encontraremos todos los componentes comunes reutilizados a lo largo de toda la aplicación, tendremos botones, alertas, avatares, migas de pan, tarjetas de receta, navegaciones, footers, componentes genéricos de formulario, iconografía, portales y colapsables.

- **components/dashboard:** tendremos componentes que serán utilizados en el backoffice, además estarán subdivididos por su utilización en cada página del dashboard.
- **components/layout**: los componentes layout son los envoltorios que generarán la disposición de la parte pública de la aplicación y la disposición del backoffice. Serán interfaces diferentes conformadas por componentes comunes.

- **components/pages**: aquí encontraremos todos aquellos componentes que conforman las vistas públicas.

- **components/skeletons**, los skeletons serán componentes utilizados para mostrar al usuario cuando los datos se están cargando desde un servicio.

- **lib/store**, aquí encontraremos todas las dependencias de **Redux**, estará organizado por features, que estarán orientadas a la lógica de negocio.

10. Plataforma de desarrollo

10.1 Entorno de desarrollo

En el siguiente apartado hablaremos de las herramientas que conforman el entorno de desarrollo, destacar que además de editores de código y otro tipo de software orientado a la producción de codificación, se incluyen además las herramientas orientadas al diseño y conceptualización y organización de las tareas.

- **Ubuntu**, como sistema operativo utilizaremos Linux bajo la distro Ubuntu, que está basada en Debian. Es de las distros más utilizadas de Linux por su bajo nivel de complejidad a la hora de configuraciones y nos permitirá ahorrar recursos y tiempo.

- **VSCode**, editor de código fuente desarrollado por Microsoft para Windows, Linux, macOS y Web. Incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código. También es personalizable, mediante la instalación de plugins, lo que lo convierte en casi un IDE. Es gratuito y de código abierto.

- **Git**, es un software de control de versiones de aplicaciones. Su propósito es llevar registro de los cambios en los diferentes archivos del proyecto, incluyendo coordinar el trabajo que varias personas realizan sobre archivos compartidos en un repositorio de código.

- **npm**, ya que nuestro proyecto es full stack Javascript, utilizaremos **npm** como sistema de gestión de paquetes por defecto, para la instalación de todas las librerías que necesitaremos en nuestro proyecto.

- **Node.js**, será el entorno de ejecución que utilizaremos por defecto. Basado en Javascript construido con el motor V8 de Chrome. Estaremos trabajando la versión 16.14.LTS, estable y de larga duración.

- **Postman**, utilizaremos esta herramienta para hacer peticiones a APIs y generar colecciones de peticiones que nos permitan probarlas de una manera rápida y sencilla.

- **Mongodb**, será nuestro sistema de base de datos NoSQL, orientado a documentos y de código abierto. Trabajaremos remotamente conectados al cluster de **Atlas Cloud** de **Mongodb**. Para acceder al mismo, utilizaremos la extensión creada por **Mongodb** para **VSCode.**

- **Notion**, es un software de gestión de proyectos y toma de notas, aquí organizaremos todas las tareas referentes al desarrollo y organización del proyecto. Como Kanban, Pecs, los Epics, las Tasks y las incidencias observadas por los usuarios durante las pruebas con usuarios reales. Eso nos permitirá tener todo tipo de registro de datos durante el proceso de conceptualización, desarrollo y pruebas.

- **Figma**, es un editor de gráficos vectorial y una herramienta de generación de prototipos. En nuestro caso ya que utilizaremos **Ubuntu**, utilizaremos la versión web ya que no tenemos versión de cliente dedicada. Utilizaremos Figma para la generación de prototipos de componentes y pantallas y la conceptualización de los flujos de navegación.

10.2 Hardware

Para llevar a cabo el desarrollo de este proyecto se ha utilizado el siguiente hardware que conforma el equipo de desarrollo:

- Corsair iCUE 4000X RGB Cristal Templado USB 3.0 Blanca, como caja.
- MSI MAG Core Liquid 240R V2 Kit, como refrigeración principal del procesador.
- Nfortec Alcyon 512GB SSD M.2 NVMe, como disco duro principal.
- Gigabyte B660 GAMING X DDR4, como placa base.
- Corsair RMx White Series RM750x 750W 80 Plus Gold Full Modular, como fuente de alimentación.
- Corsair Vengeance RGB Pro DDR4 3200 PC4-25600 32GB 2x16GB CL16, como memoria RAM.
- Intel Core i5-12400 4.4 GHz, como procesador.
- LG 27UL550-W 27" LED IPS Ultra HD 4K FreeSync x2, como monitores.
- Keychron K6, como teclado.
- Logitech Pro Wireless Mouse, como ratón.

11. Estudio de viabilidad

Llegados a este punto realizaremos un análisis del mercado basado en el estudio publicado por Silestone Institute, Global Kitchen, *The home Kitchen in the globalization era*[10]

- **La función de la cocina en el hogar.**

  Casi nueve de cada diez profesionales (87%) confirman la tendencia de que la cocina va a adquirir más importancia en el hogar, frente a un 9% que cree que que la cocina conservará la misma importancia y un 3,5% que considera que otras estancias que otras estancias serán más importantes.

  Esta opinión es más pronunciada en Brasil, Portugal y Reino Unido (por encima del 90%), mientras que Australia parece más reacia a reconocer este nuevo papel (74%).

  Debido a la creciente relevancia de la cocina en el hogar, el 82,8% de los profesionales creen que el papel de esta estancia debería tener más importancia en la forma de planificar y construir los espacios colectivos de la vivienda.

- **¿Qué actividades tendrán lugar en las cocinas en el futuro?**

  La cocina se consolidará en los próximos años como espacio de encuentro y ocio para los miembros de la familia. Además de la manipulación de alimentos y la cocción, la cocina será el centro de otras actividades que tradicionalmente se han llevado a cabo en otras zonas del hogar.

  Actividades como, estar con nuestros amigos o familiares, comer, ver la televisión, navegar en internet o trabajar o estudiar, están empezando a formar parte de nuestros hábitos en esta estancia de nuestro hogar.

[10] G.Kitchen, *The home kitchen in the globalization era*, Silestone, 2019

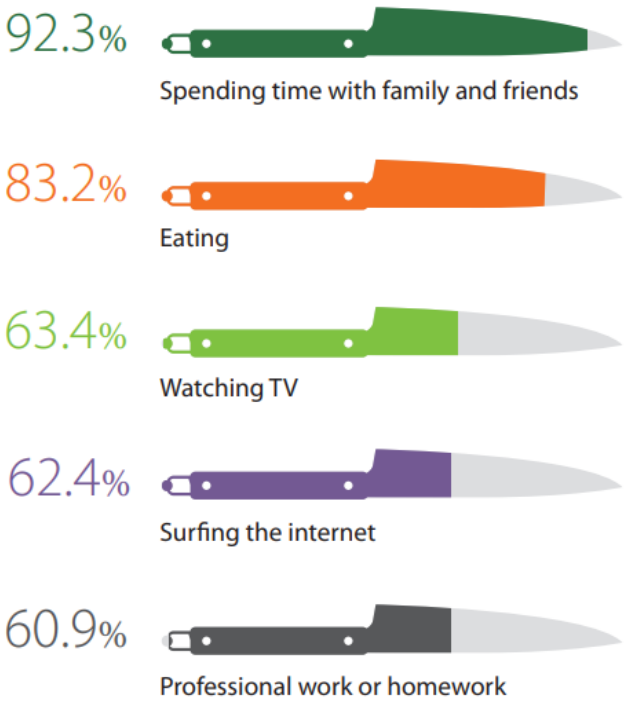


Figura 31: Las actividades en la cocina del futuro

- **¿Cuáles son los principales cambios tecnológicos que afectarán a la cocina en el futuro?**

La conexión de la cocina a Internet y a los dispositivos y los electrodomésticos inteligentes destacan como las principales innovaciones tecnológicas a corto y medio plazo. Según los profesionales del sector, las encimeras del futuro deberían permitir a los usuarios cocinar directamente en la superficie, además de incorporar conectividad y actuar como un panel de control.

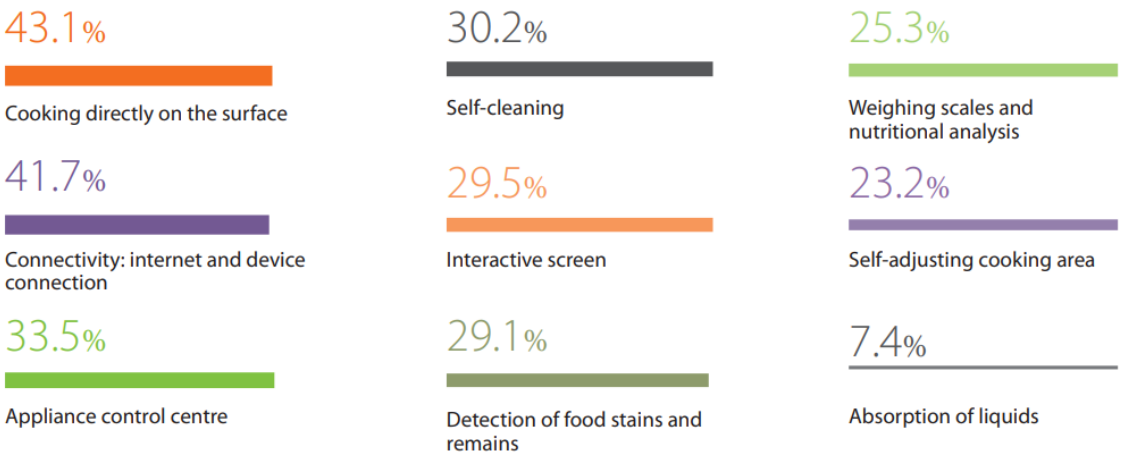


Figura 32: Cambios tecnológicos en las cocinas.

Bajo las premisas anteriormente expuestas, parece plausible, que la interconexión entre cocinas o usuarios que las manejan se digitalizará y que realizaremos en ellas otro tipo de actividades que además de incluir la cocina, nos permitirán socializar de alguna manera, tal es así que podremos seguramente, intercambiar opiniones con personas bien de forma remota (mediante alguna plataforma) o físicamente.

Como ya comentamos en el **Marco Teórico** del presente trabajo, existen diversas alternativas que si bien en su conjunto, cubren la necesidad de tener una comunidad orientada a los robots de cocina, no lo hacen de forma aislada.

Si bien **Cookpad** o **Funcook** nos permiten intercambiar recetas entre usuarios, utilizar estas plataformas orientadas a procesadores puede suponer una gran pérdida de tiempo para encontrar el contenido deseado. Esto es suplido por los foros y plataformas de las marcas de cada robot, pero obviamente no recogen las recetas de la competencia.

Este proyecto justifica su viabilidad en el nicho de mercado existente tras la revolución que supone la digitalización de los espacios comunes en los hogares, la aparición de los procesadores de alimentos como multiherramientas que concentran el foco culinario y la masificación de los creadores de contenido a lo largo de multitud de plataformas en internet.

Gracias a este conjunto de circunstancias, podremos concentrar todas estas necesidades en la virtualización de un espacio que dé a los usuarios una herramienta intuitiva, sencilla y que les permita encontrar lo que buscan, pero además crear todo aquello que necesiten. Permitiendo además ahorrar tiempos de búsqueda, herramientas nativas pesadas o privativas y agrupando todas aquellas publicaciones que sean de su interés.

Los usuarios no solo contarán con una aplicación que les facilite encontrar recetas orientadas al procesador de alimentos que tienen, sino que además podrán crear libremente aquellas que deseen sin estar supeditados a plataformas privativas de cada marca; y compartir con otros usuarios dichas publicaciones.

Esto conseguirá generar una comunidad atractiva y libre que aportará un valor de negocio mediante la aportación de datos a través del contenido. Estos datos serán muy valiosos en el futuro, ya que podremos ir modelando un sistema de alimentación explotable no solo a nivel culinario sino a nivel dietético y podremos comercializar su venta y/o distribución, o explotación por terceros mediante servicios de suscripción, etc…

Por lo tanto podemos afirmar la viabilidad del proyecto en varios puntos clave:

- Comunidad de usuarios.
- Democratización del contenido orientado a procesadores de alimentos.
- Sistema de API para explotación de servicios, como suscripciones, información nutricional, y/o explotación del mismo a terceros distribuidores.

12. Diseño y usabilidad

12.1 Conceptualización

Según el estudio Global Kitchen, la cocina se perfila como un espacio multifuncional e hiperconectado, en el que aparatos inteligentes no sólo facilitarán el cocinado sino también la organización y compra de alimentos en el hogar.

Si bien el cambio tecnológico de cocineros robots, hornos inteligentes, encimeras donde ver vídeos y proyectar recetas está aún por llegar. Lo que sí es ya es hoy una realidad es el uso de las redes sociales para la cocina.

Como sucede en muchos otros ámbitos de la vida cotidiana, las redes sociales se utilizan habitualmente para recabar información y compartir experiencias sobre nuevos platos e ingredientes, conocer nuevas recetas o introducir cambios en la dieta.

Las redes sociales no solo nos conectan con aquellos familiares y amigos que hoy, debido al confinamiento, los tenemos a la distancia, sino que también nos ofrecen soluciones y novedades para variar en comidas ricas, originales y dignas de cualquier chef.

En este contexto, tendremos por un lado usuarios habituados al uso de las redes sociales, consumidores y creadores de contenido, familiarizados con algún procesador de alimentos y otro tipo de dispositivos electrónicos y tecnológicos. Y por otro lado tendremos personas con aficiones o gustos orientados a la gastronomía, críticos de cocina o profesionales de la misma.

Para visualizar de una manera más clara las necesidades y expectativas del público objetivo, a continuación definiremos las siguientes **User Personas**:

<table>
<tr><td> 

<u>Ana Sánchez</u>

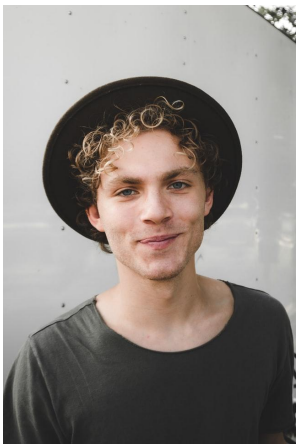
Edad: 28 años
Ocupación: Youtuber
Localidad: Barcelona
Educación: Universidad</td><td>Bio

Ana es una joven barcelonesa apasionada por la cocina, trabaja como influencer mediante la plataforma Youtube, a través de la cual distribuye contenido acerca de consejos para la cocina, promociona productos recomendados y ofrece a sus suscriptores listas de recetas de cocina.

Está siempre conectada ya sea por medio del teléfono o el ordenador, y gracias al patrocinio de varias marcas recibe nuevas versiones de diferentes robots de cocina para hacer reviews de los mismos y compartir con sus suscriptores la experiencia de uso de estos.

Le encanta categorizar todo su contenido para facilitar su posicionamiento y así garantizar a sus seguidores que encuentran aquello que buscan en su canal.</td></tr>
<tr><td>Comportamiento</td><td>● Utiliza las herramientas digitales de forma plena
● Es detallista y procura ofrecer una información real y sin sesgo en su contenido.
● Busca la sencillez en su forma de comunicar e informar
● Comparte mucho contenido visual
● Está actualizada con cada novedad tecnológica</td></tr>
<tr><td>Necesidades</td><td>● Acceso intuitivo y moderno.
● Encontrar de forma sencilla lo que busca.
● Compartir el contenido que crea.
● Recibir buenas valoraciones de sus creaciones.</td></tr>
<tr><td>Pain points</td><td>● Categorización ineficaz.
● Mapa web demasiado amplio.
● Aplicación web no adaptada a dispositivos móviles.
● Capacidad para compartir según tendencia de redes compleja.</td></tr>
</table>

Tabla 3: User persona - Ana Sánchez

<table>
<tr><td rowspan="2"> 

<u>Carlos Hernández</u>

Edad: 33 años
Ocupación: Periodista
Localidad: Vigo
Educación: Universidad</td><td>Bio</td></tr>
<tr><td>Carlos es un joven vigués, amante de la cocina, de los buenos productos gastronómicos y de salir a comer. Le encanta hacerle fotos a todo lo que come y cocina y sobre todo compartirlas con la gente. Es un “cocinillas”, siempre sorprende a sus amigos en las cenas con alguna receta nueva.

Le encanta la tecnología, ha empezado a domotizar su casa recientemente y la cocina ha recibido un nuevo procesador de alimentos. Pasa muchas horas delante de las pantallas buscando ingredientes y recetas exóticas.

Cocina de forma tradicional, pero gracias a tener un nuevo procesador de alimentos está mejorando su forma en la que cocina y sus conocidos no paran de repetirlo.

Suele participar en foros, seguir a influencers del mundo de la cocina y a prestigiosos chefs</td></tr>
<tr><td>Comportamiento</td><td>● Busca recetas en todo tipo de plataformas.
● Atiende al detalle y sabe que es la clave del éxito.
● Está siempre actualizado tecnológicamente.
● Sus aficiones son el vehículo para socializar.</td></tr>
<tr><td>Necesidades</td><td>● Acceso optimizado desde dispositivos móviles.
● Encontrar la información de manera precisa.
● Compartir el contenido que crea.
● Buscar personas afines a sus gustos.</td></tr>
<tr><td>Pain points</td><td>● Demasiada información.
● Mapa web demasiado amplio.
● Interacciones complejas.</td></tr>
</table>

Tabla 4: User persona - Carlos Hernández

Las redes sociales, son nuestro principal punto de entrada. A través de ellas penetramos en nuestro público objetivo. Cómo hacerlo, a través de anuncios publicitarios, influencers a los que patrocinamos o cualquier

otra campaña en redes sociales que mueva a nuestro principal público objetivo. Como principal estrategia para el **User Engagement**:

- Cuidaremos la estética, que promoverá la atención focalizada y mantendrá una asociación de imagen de marca, en nuestro sitio y en cualquier otro medio publicitario. Tendremos en cuenta principios de simetría, equilibrio, prominencia y sencillez.

- Promoveremos la navegación para retener al usuario, creando una comunidad,que permita a los usuarios construir, y recibir de otros usuarios aportaciones.

- En cuanto a la usabilidad, haremos un espacio simple, dinámico y sencillo donde los pain points de los usuarios como Ana y Carlos, se reduzcan a la mínima expresión. Evitando la sobreinformación, y un mapa web muy amplio.

12.2 Diseño de la experiencia

En este apartado analizaremos los requisitos desde un punto de vista conceptual y funcional para los tipos o modelos de usuario. Además de los requisitos propuestos, desarrollaremos los flujos de usuario a lo largo de toda la aplicación, así como el sitemap de nuestro espacio.

12.2.1 Requisitos

En este punto expondremos los requisitos funcionales de la aplicación categorizados por las páginas principales de la misma. Cada requisito describe la expectativa de las funcionalidades esperadas por los user persona descritos en el punto anterior:

Categoría	Requisto Funcional
Registro	Registro de usuario mediante correo y contraseña.
Inicio de sesión	Autenticación de usuario mediante correo y contraseña.
Recuperación de cuenta	Recuperación de cuenta por contraseña perdida.
Verificación de email	Verificación de email a través del perfil de usuario y confirmación mediante el correo electrónico y página de confirmación
Home	Visualización de página de inicio con recetas y categorías.
Home	Selección de contenido para cualquier tipo de usuario.
Home	Búsqueda de receta según palabra para cualquier usuario.

Home	Visualización listado de recetas de usuarios seguidos para usuarios autenticados.
Recetas	Visualización listado de recetas ordenadas por fecha de creación.
Recetas	Visualización listado de las recetas más cocinadas.
Categoría	Posibilidad de ver las recetas pertenecientes a una categoría por todos los usuarios
Detalle Receta	Visualización de información de la receta para todos los usuarios.
Detalle Receta	Visualización de pasos a seguir para usuarios autenticados.
Detalle Receta	Interacción social, like, comentario y guardar en colecciones para usuarios autenticados.
Editar Receta	Posibilidad de modificar una receta o eliminarla por un usuario autenticado al que pertenezca.
Añadir Receta	Creación de receta por un usuario autenticado mediante un formulario.
Perfil	Visualización del contenido creado por todos los usuarios.
Perfil	Acceso a edición de perfil por el propio usuario autenticado.
Perfil	Posibilidad de seguir o dejar de seguir o enviar mensaje privado por parte de otro usuario autenticado.
Cuenta / Editar perfil	Posibilidad de modificar los datos del perfil o su eliminación por un usuario autenticado
Cuenta / Colecciones	Visualización de colecciones de recetas agregadas por un usuario autenticado.
Cuenta / Colecciones	Visualización del total de ingredientes de una colección
Cuenta / Colecciones	Edición o eliminación de una colección de recetas
Cuenta / Favoritos	Visualización de recetas añadidas a favoritos
Cuenta / Favoritos	Eliminación de recetas de favoritos

Tabla 5: Requisitos Funcionales

Categoría	Requisitos no funcionales
Frontend	El frontend debe estar desarrollado en React 18 bajo el Framework Nextjs 12.1
Frontend	Componentes UI desarrollados en React con DaisyUI y Tailwindcss
Frontend	El store deberá estar basado en Redux Toolkit con gestión de peticiones asíncronas mediante Thunks
Frontend	La aplicación deberá ser responsive
Frontend	La aplicación deberá ser compatible con la mayoría de navegadores.
Backend	El backend debe de estar desarrollado con Express en Nextjs 12.1
Backend	Backend protegido con Passport.js
Base de datos	La base de datos de ser en MongoDB desplegada en Atlas Cloud
Servicios	Comunicaciones entre frontend y backend serán ser HTTPS
Servicios	Comunicaciones entre frontend y backend en formato JSON

Tabla 6: Requisitos no funcionales

12.2.2 Sitemap y user flows

En este subapartado se recoge el mapa de navegación de nuestra aplicación en un sentido amplio, es decir, para usuarios autenticados y también no autenticados, que son los que podrían acceder a todas las vistas disponibles.

Posteriormente detallaremos en los user flows los accesos a los procesos que tienen los usuarios autenticados y no autenticados, así como la creación, edición o eliminación de una receta, que son las funcionalidades básicas de la aplicación.

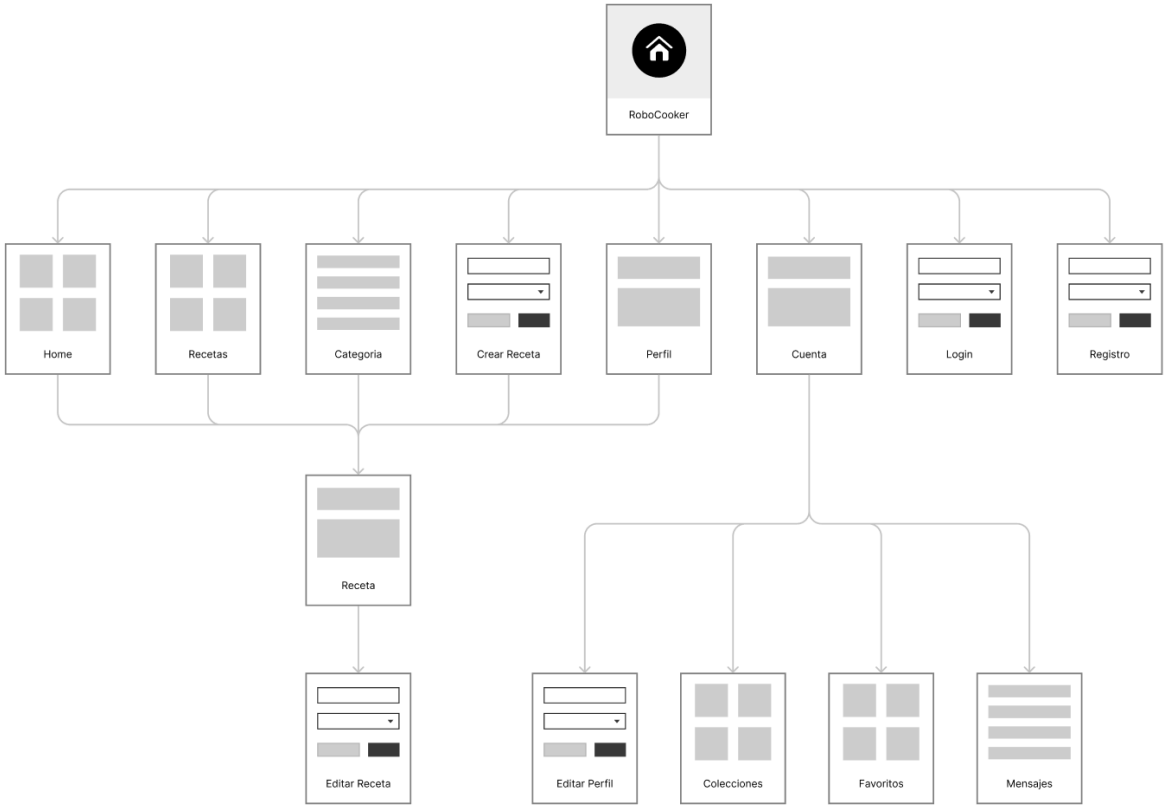


Figura 33: Sitemap

Como podemos ver accederemos a una vista Home y podremos navegar entre las diferentes vistas, Recetas, Categoria, Crear Receta o Perfil, ambas nos conducen a la vista en detalle de una receta seleccionada y su posible edición o eliminación en otra vista independiente.

En el apartado cuenta podremos editar el perfil del usuario, ver las colecciones que haya generado el usuario o los favoritos añadidos, así como los mensajes. Si el usuario no está autenticado podremos acceder a las vistas de Inicio de sesión o registro.

En la siguiente figura podremos ver el itinerario que puede realizar un usuario que no ha iniciado sesión en la aplicación. En este caso la figura describe que podrá acceder desde la Home a diferentes vistas utilizando el menú de navegación, el cual le llevará a la opción deseada y ésta acabará en una página de receta.

Podrá acceder en última instancia al perfil del usuario que haya creado esa receta:

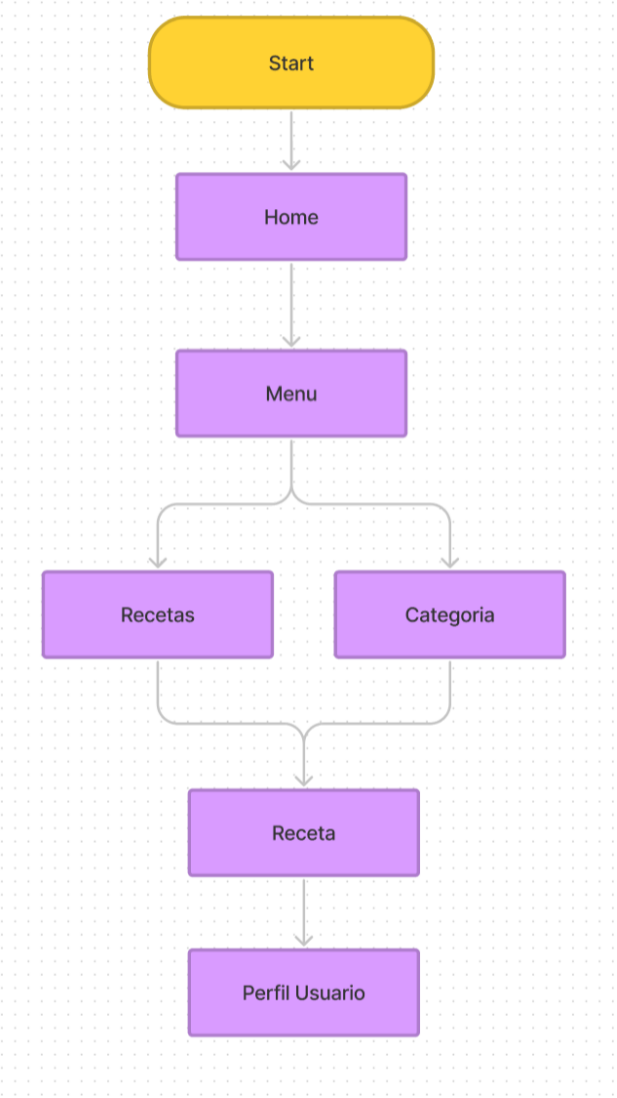


Figura 34: Flujo de interacción sin autenticación

En la siguiente figura podremos ver todas las acciones que puede realizar un usuario autenticado en la plataforma, incluyen las de la figura anterior, y además podrá salir de la sesión, acceder a su cuenta, colecciones, favoritos, mensajes, y editar su perfil.

Podrá también comentar una receta, darle a like o unlike, y añadir a su colección la receta. Si dicha receta la ha creado él, podrá además editarla o eliminarla.

Los diferentes bloques en morado constituyen las vistas sobre las cuales un usuario podrá navegar si está autenticado, y los bloques azules serán las determinadas acciones que podrá realizar en una página de receta, que coinciden con las interacciones sociales propuestas como aplicativo:

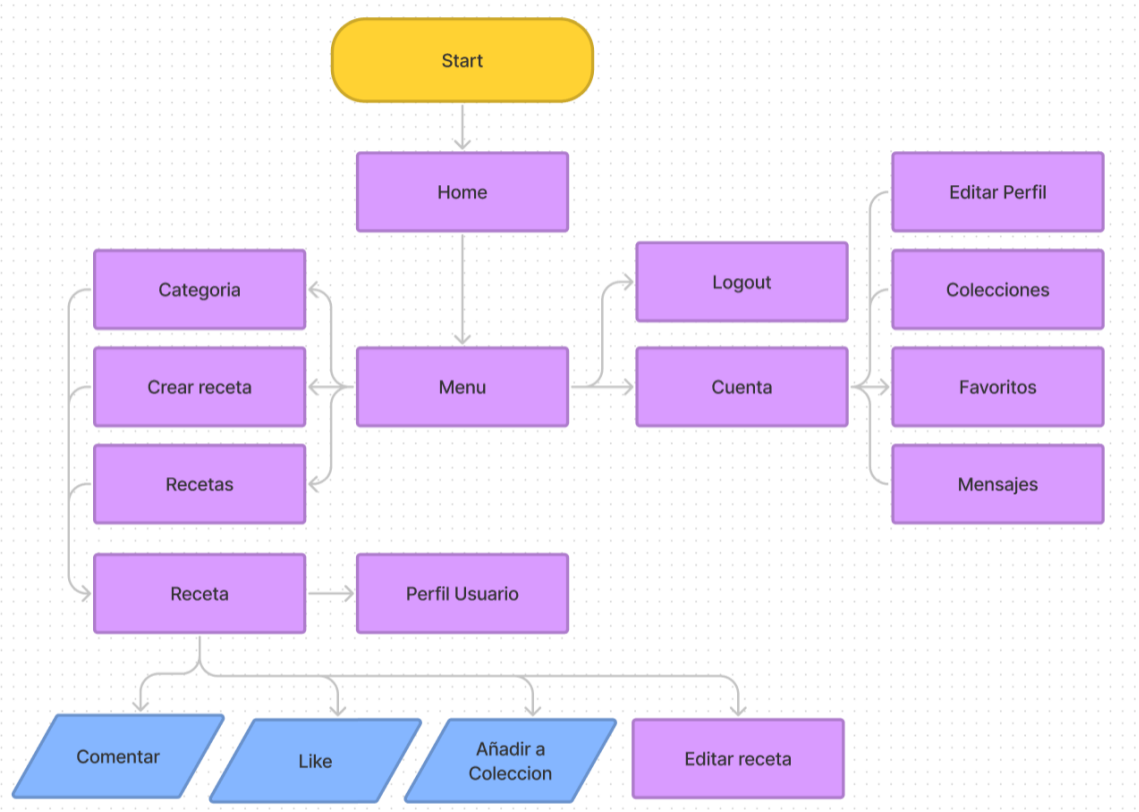


Figura 35: Flujo de interacción con autenticación

En el siguiente diagrama podemos ver como un usuario **creará una receta**, el usuario podrá acceder a la creación de receta si ha iniciado sesión, la aplicación comprobará si está logueado o no.

Si está autenticado entonces podrá crear la receta rellenando un formulario, al crear la receta y guardarse en base de datos el usuario será redirigido la página de su publicación para ver el resultado final:

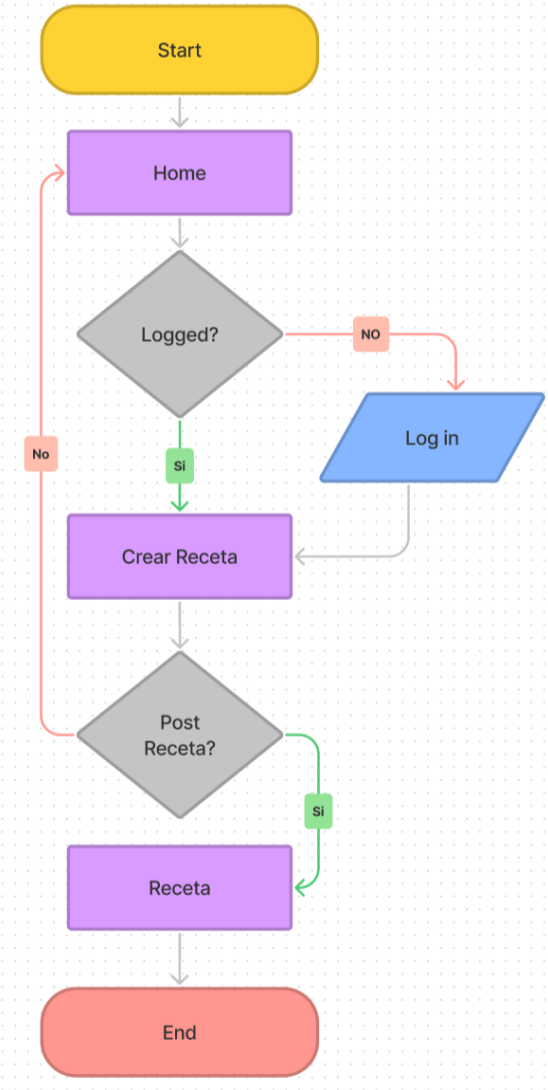


Figura 36: Flujo de creación de una receta

Un usuario autenticado podrá también **editar o eliminar una receta** desde la vista de Editar Receta.

En el supuesto tambien de **añadir una colección, editarla o eliminarla** será muy similar y podrá hacerlo en la subvista Colecciones, como queda reflejado en las siguientes figuras:

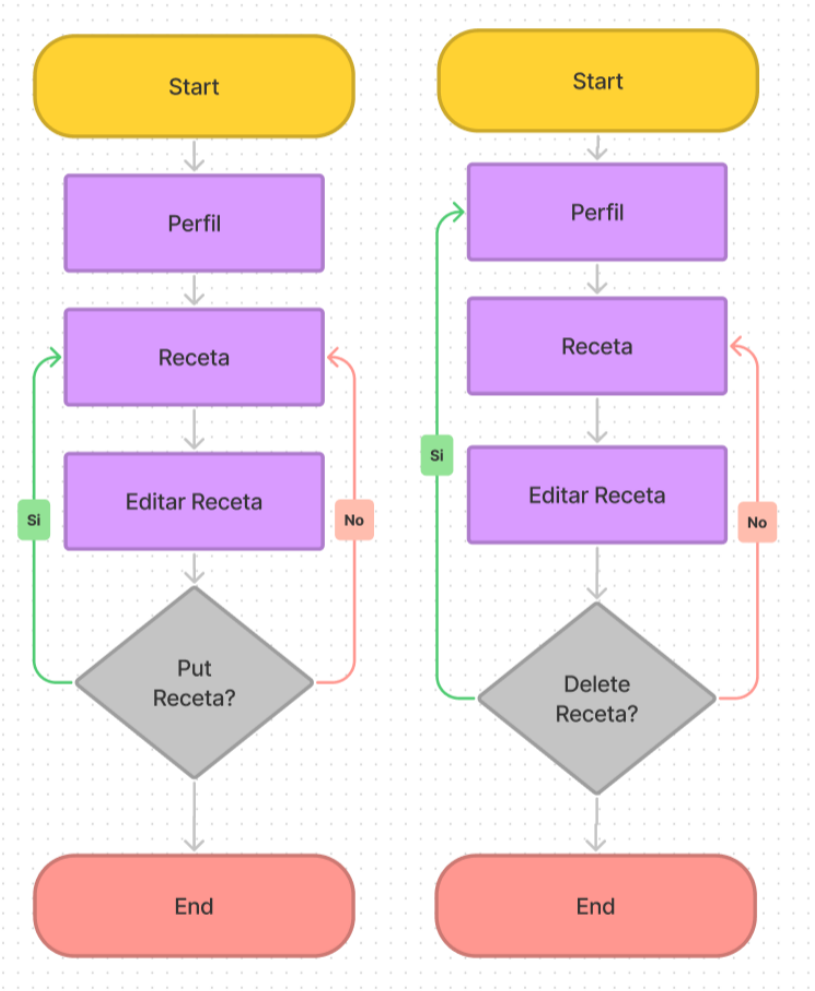


Figura 37: Flujos de edición y eliminación de una receta

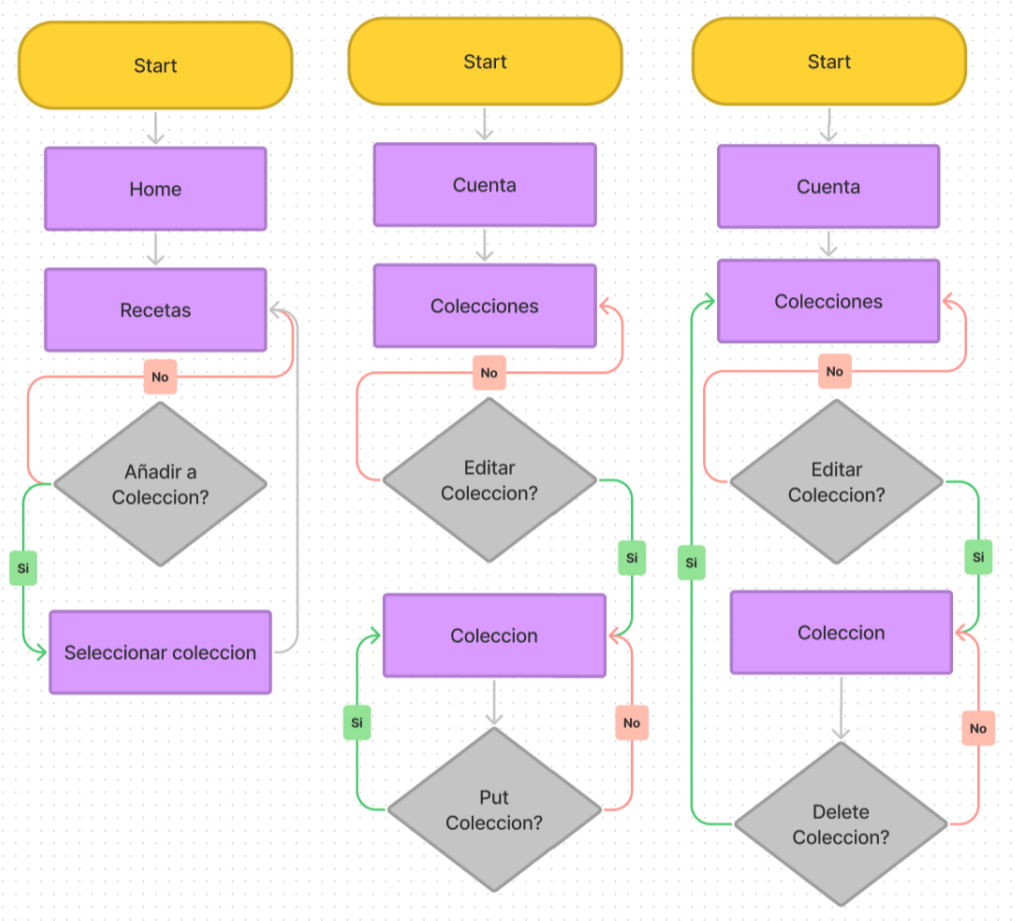


Figura 38: Flujos de añadir, editar y eliminar una colección

Por último, un usuario autenticado, podrá **comentar** una receta, en la vista de detalle de la receta. Podrá también darle a **like** o unlike a dicha receta, y finalmente podrá ver las recetas a las que le ha dado like en su Cuenta en el subapartado **Favoritos**, donde también podría eliminar esas recetas con like de la lista. Para ello como vemos en los siguientes diagramas, el usuario accede a la página de una receta, y como reflejan los bloques azules de acciones, podrá comentar o darle like si está autenticado.

En el supuesto tercero, desde su dashboard, podrá acceder a la vista de Favoritos y eliminar una receta de los favoritos:

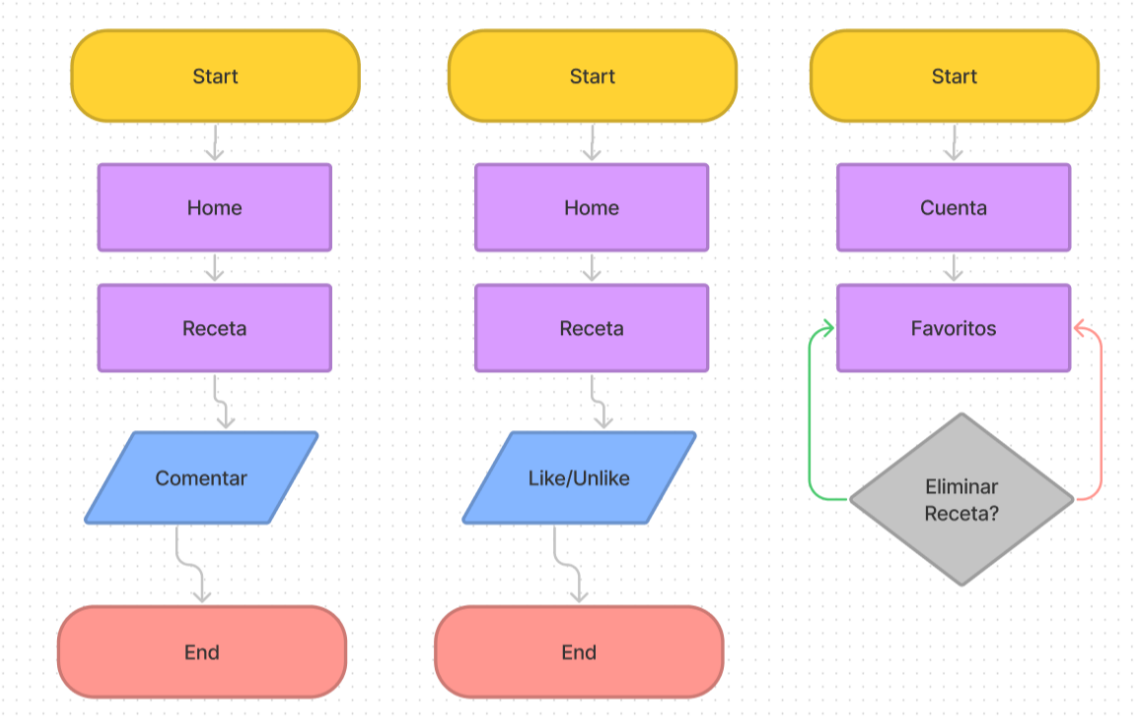


Figura 39: Flujos de interacción social sobre una receta

12.3 Diseño de la interacción

En esta fase procederemos a realizar el prototipado de la aplicación, que será una aproximación al diseño de las interfaces que tendrá el sistema.

Partiendo de los flujos y diagramas de interacciones planteados anteriormente, se ha procedido a diseñar la aplicación de manera que su utilización sea fácil y fluida para los potenciales usuarios y adaptando el contenido a dispositivos móviles y de escritorio.

A continuación dividiremos en subapartados cada una de las pantallas que formarán parte de la aplicación, junto con una breve descripción y explicación de las diferentes funcionalidades de cada una. Se incluyen tanto la vista de dispositivos móviles como desktop.

Podemos acceder a la totalidad de los diseños de las pantallas donde veremos realmente todo el flujo de un usuario a lo largo de la aplicación de forma muy detallada, en: **Figma**.

12.3.1 Inicio de sesión / registro

Tanto la página de inicio de sesión como de registro contendrá un formulario para el caso de uso correspondiente. Por simplicidad hemos incluido solo el diseño de registro de un usuario, accesible desde el enlace en la barra de navegación “Iniciar sesión”. Este enlace sólo será visible cuando el usuario no esté autenticado. El usuario se encontrará con un formulario que contendrá los campos de correo electrónico y contraseña en el caso de inicio de sesión, y nombre de usuario si está en la vista de registro. Esto servirá para autenticar a un usuario en el proceso de acceso a la sección de dashboard o administración. Así podemos verlo en la siguiente figura:

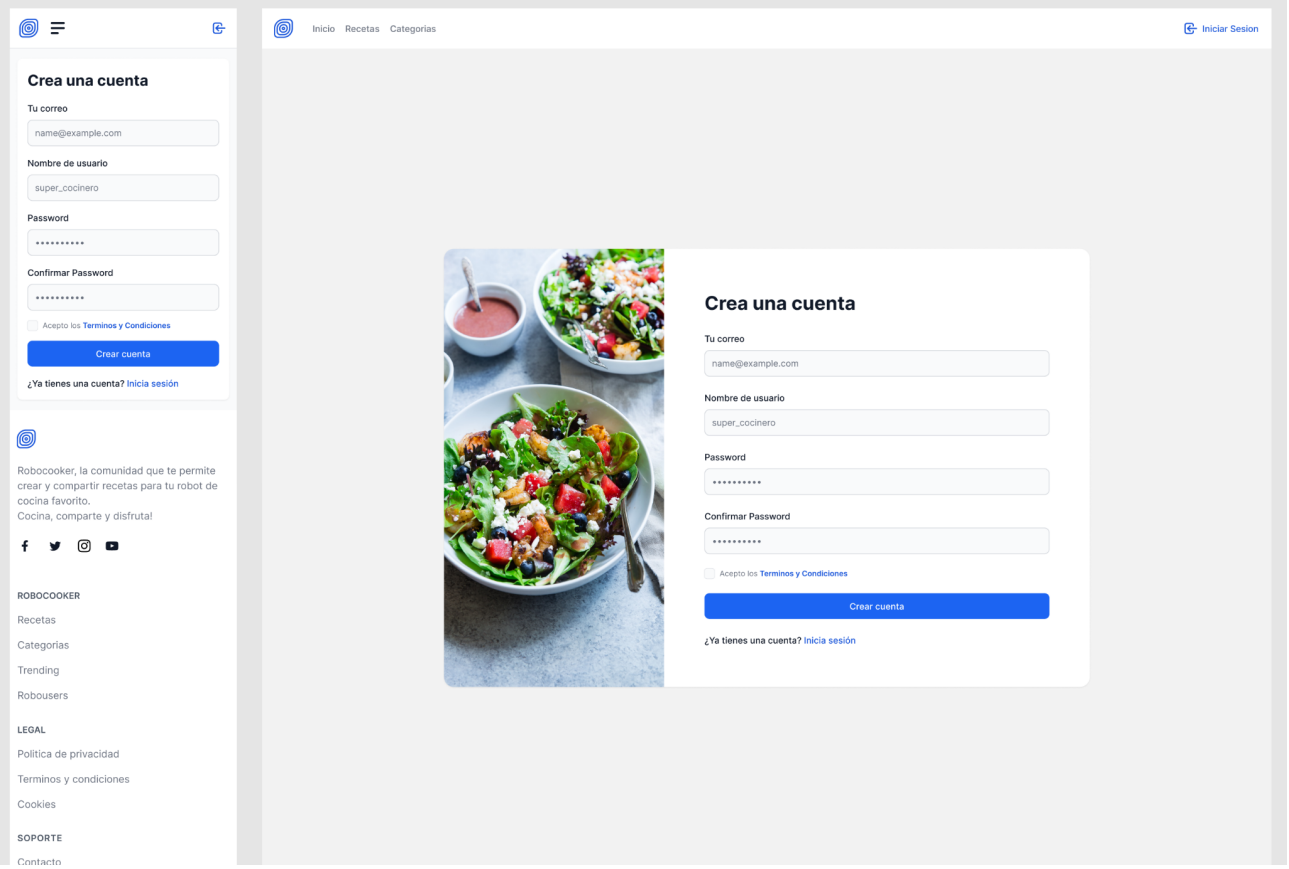


Figura 40: Inicio de sesión / registro. Figma.

12.3.2 Página de inicio

En la página de inicio podremos ver en primer lugar un hero que contendrá un input de búsqueda de una receta, el usuario podrá introducir una palabra clave para buscar una receta, al introducirlo y darle a buscar será redirigido a la página de recetas donde podrá encontrarse con los resultados ofrecidos por la aplicación.

Seguidamente tendremos categorías según las tendencias y los usuarios que más colaboran en la plataforma, procesadores de alimentos, y las últimas recetas publicadas. Todas estas opciones redirigirá al usuario a determinadas vistas, las cuales facilitarán el filtrado o segmentación para ir orientando la búsqueda de un contenido más específico.

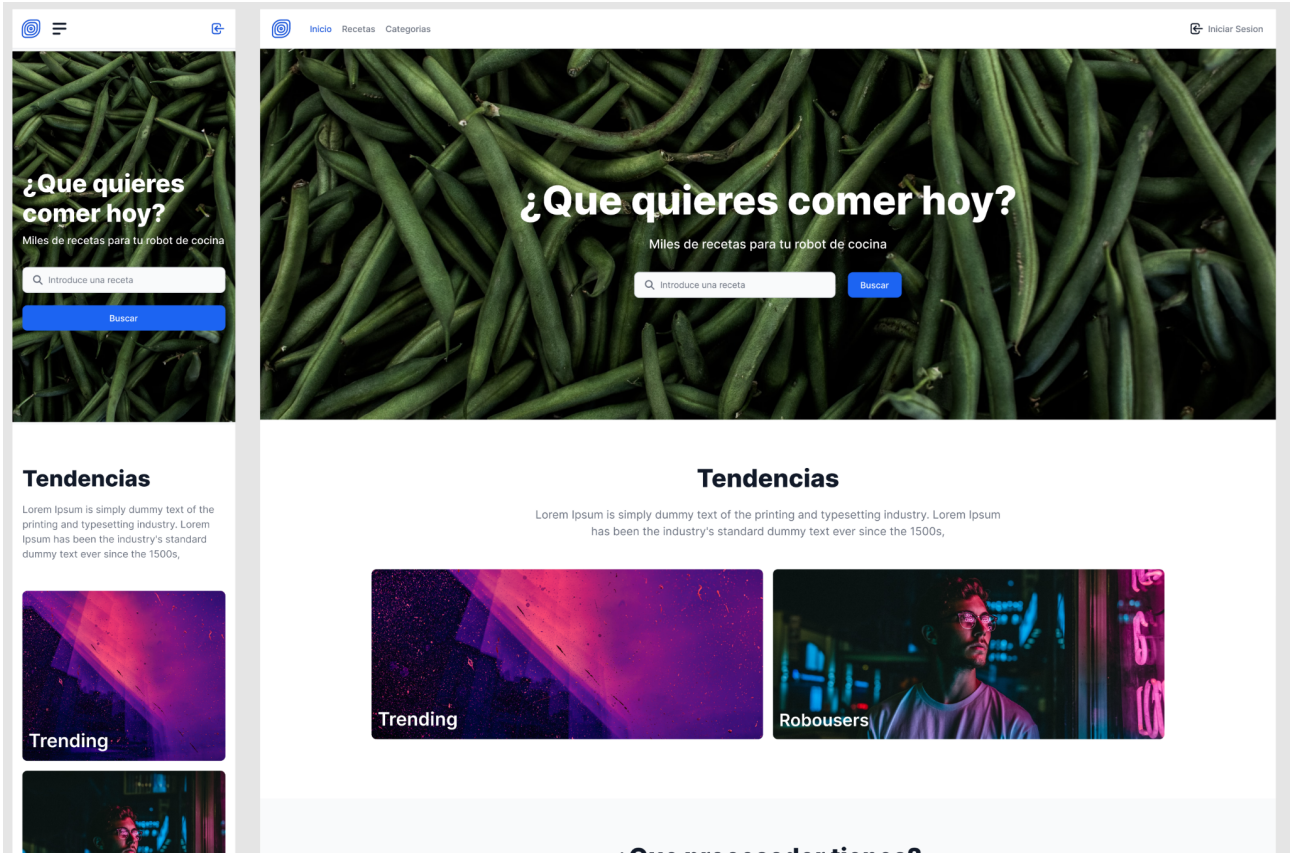


Figura 41: Página de inicio. Figma.

12.3.3 Trending / Robousers

En esta sección podremos encontrar las recetas que más “me gusta” han recibido a lo largo de un determinado periodo de tiempo.

Tendremos un podio donde estarán las 3 más valoradas, para darles una mayor visibilidad y las siguientes en un un grid a continuación. La sección de robousers será equivalente a las recetas trending pero con los usuarios que más aportan contenido. Después del podio podremos ver el un listado de recetas o usuarios (según el caso). El usuario podrá seleccionar una publicación y será redirigido a la página de detalle de dicha receta, o al perfil del usuario (según el supuesto de vista robousers). Como vemos en la siguiente figura:

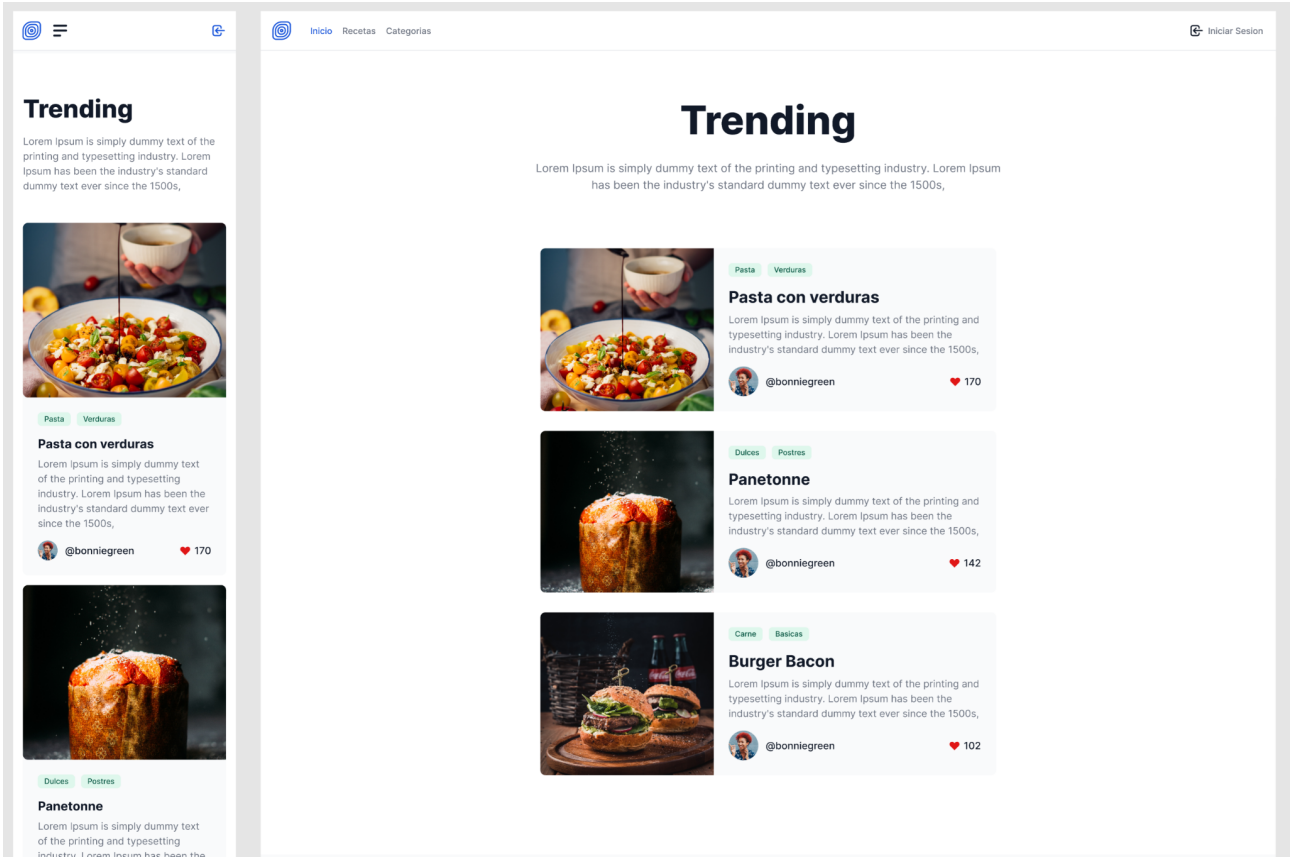


Figura 42: Trending / Robousers. Figma.

12.3.4 Categorías

En categorías podremos acceder a recetas según el tipo de alimento que queramos buscar. En esta sección podemos observar el listado de categorías en forma de rejilla, que nos ofrece un atractivo visual gracias a las tarjetas con imagen.

Al seleccionar una categoría el usuario será redirigido a una vista específica de dicha categoría seleccionada, encontrándose tres tipos de contenido, recetas más populares de dicha categoría, recetas más comentadas y las últimas recetas.

En la siguiente figura podemos ver una representación de las categorías disponibles:

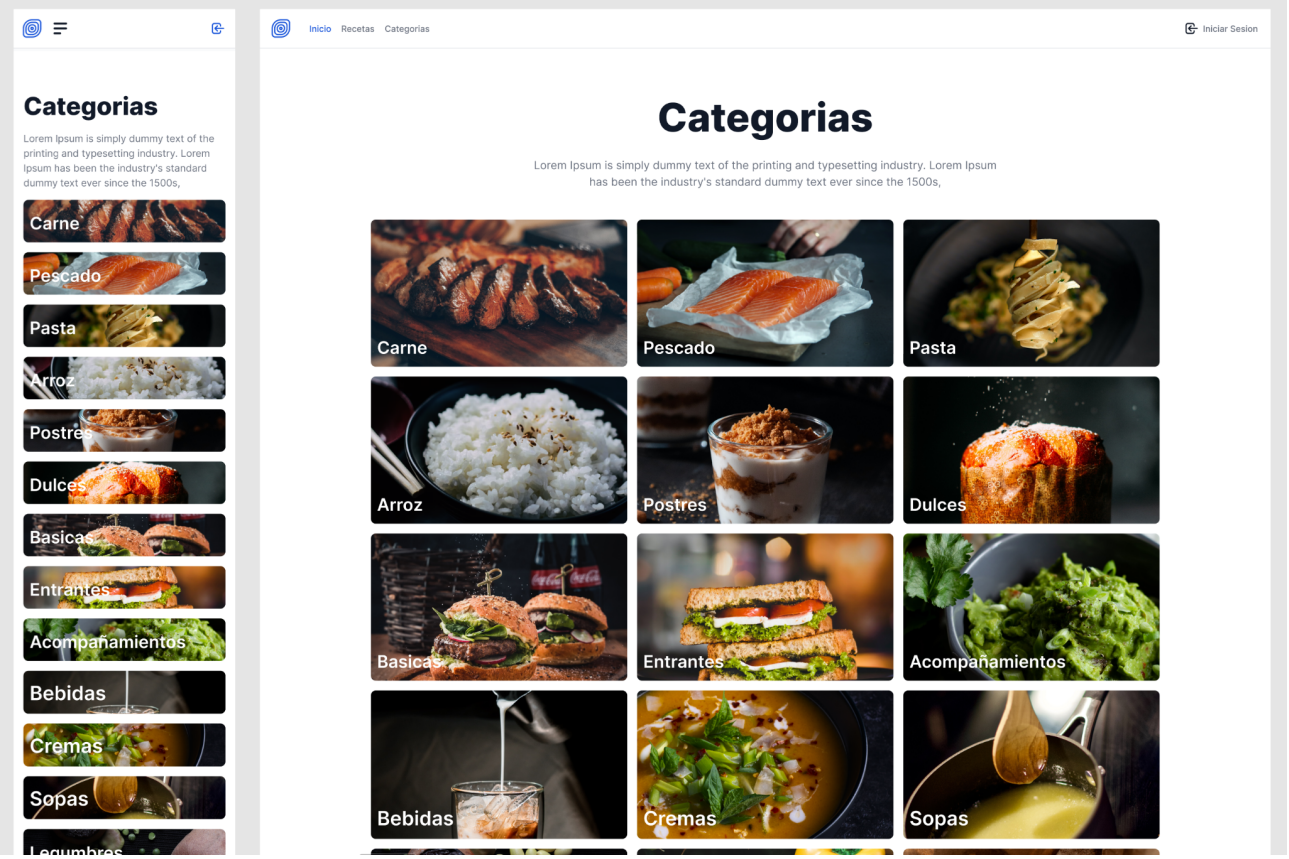


Figura 43: Categorías. Figma.

12.3.5 Recetas

A la sección de recetas podremos acceder a través de la barra de navegación, o buscando en el campo habilitado en la página de inicio.

En la página de recetas tendremos también tendremos un buscador donde podremos introducir una palabra clave para buscar una receta. Debajo de este buscador podremos ver los resultados obtenidos.

Si el usuario no realiza ninguna búsqueda inicial veremos siempre las últimas recetas publicadas en la plataforma. En la siguiente figura podemos ver el modelo de esta página:

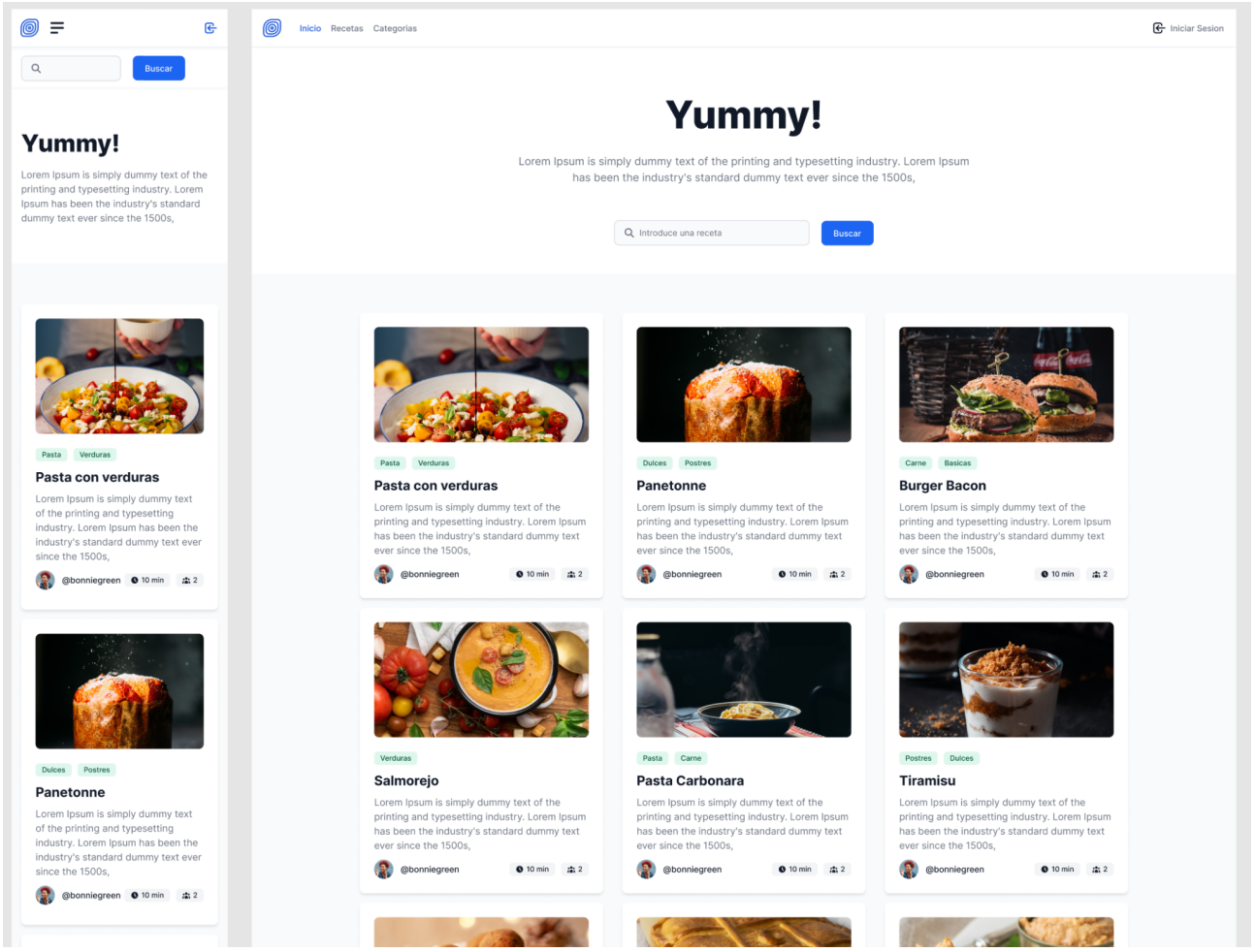


Figura 44: Recetas. Figma.

12.3.6 Receta

En la página de receta veremos los detalles de la misma, nombre, descripción, elaboración, ingredientes. Y si estamos autenticados podremos interactuar, dar “me gusta”, comentar,o añadir a los marcadores. También podremos acceder al perfil del creador de la misma.

Si es el propio usuario el que ha creado la receta, podrá acceder a la vista de edición mediante un botón de ajustes, gracias al mismo podrá editar cualquier campo de la receta en poco tiempo guardando el formulario respectivo al paso que quiera editar.

Un usuario podrá únicamente ver la elaboración si está registrado, verá el bloque de elaboración ligeramente borroso y un enlace que le sugerirá iniciar sesión, de esta manera promoveremos el registro de usuarios en nuestra plataforma.

En la siguiente figura podremos ver la representación de la página de receta:

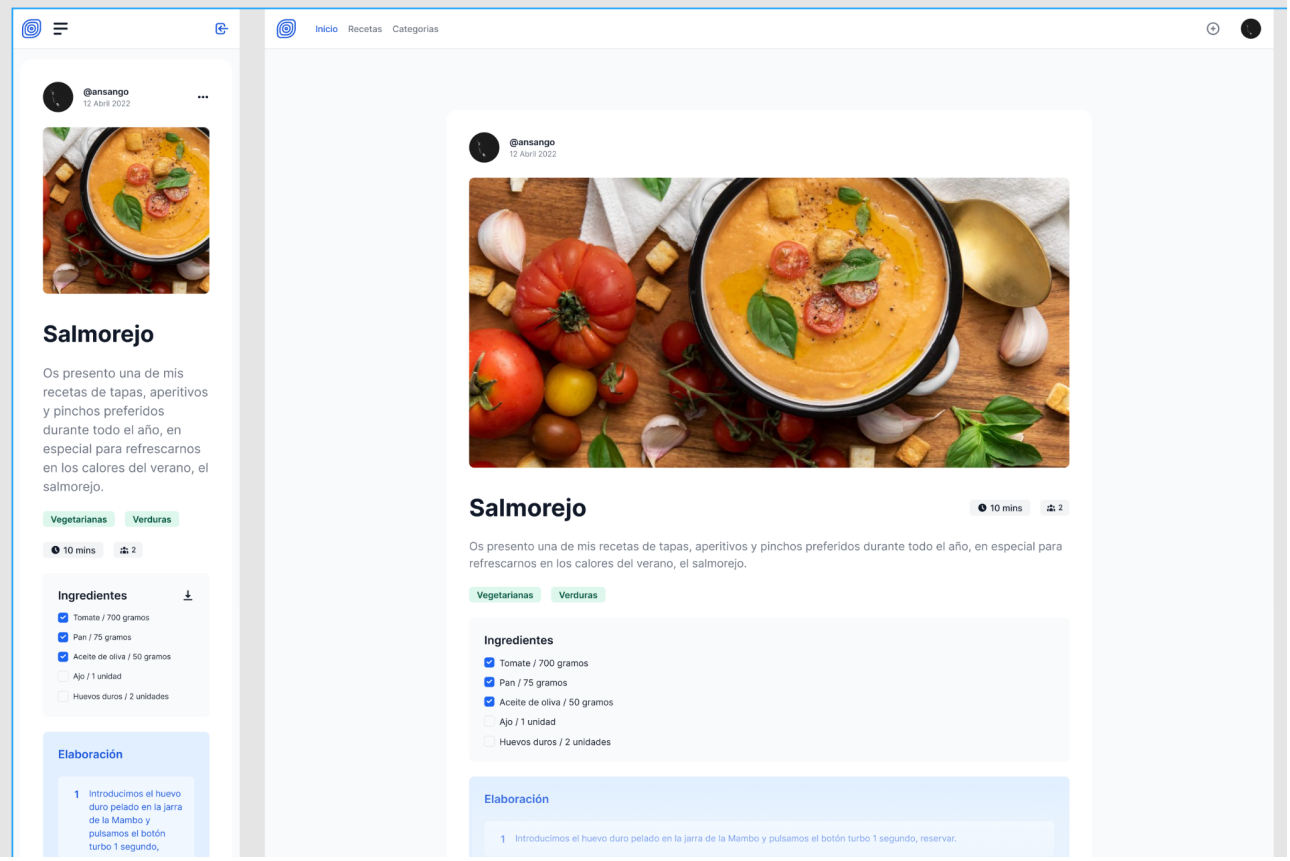


Figura 45: Receta. Figma.

12.3.7 Perfil

En el perfil de un usuario podremos ver en primer lugar un espacio que recogerá la información personal de un usuario registrado, dicha información contendrá, su nombre, una breve descripción, el número de publicaciones, la opción de seguirle, si estamos autenticados.

Posteriormente podremos ver la colección de publicaciones generada por el usuario, donde tendremos un listado de las mismas en forma de rejilla. Podremos acceder a cada publicación haciendo click sobre la misma, en este punto seremos reconducidos a la página de dicha receta, la cual podemos recordar de la figura anterior.

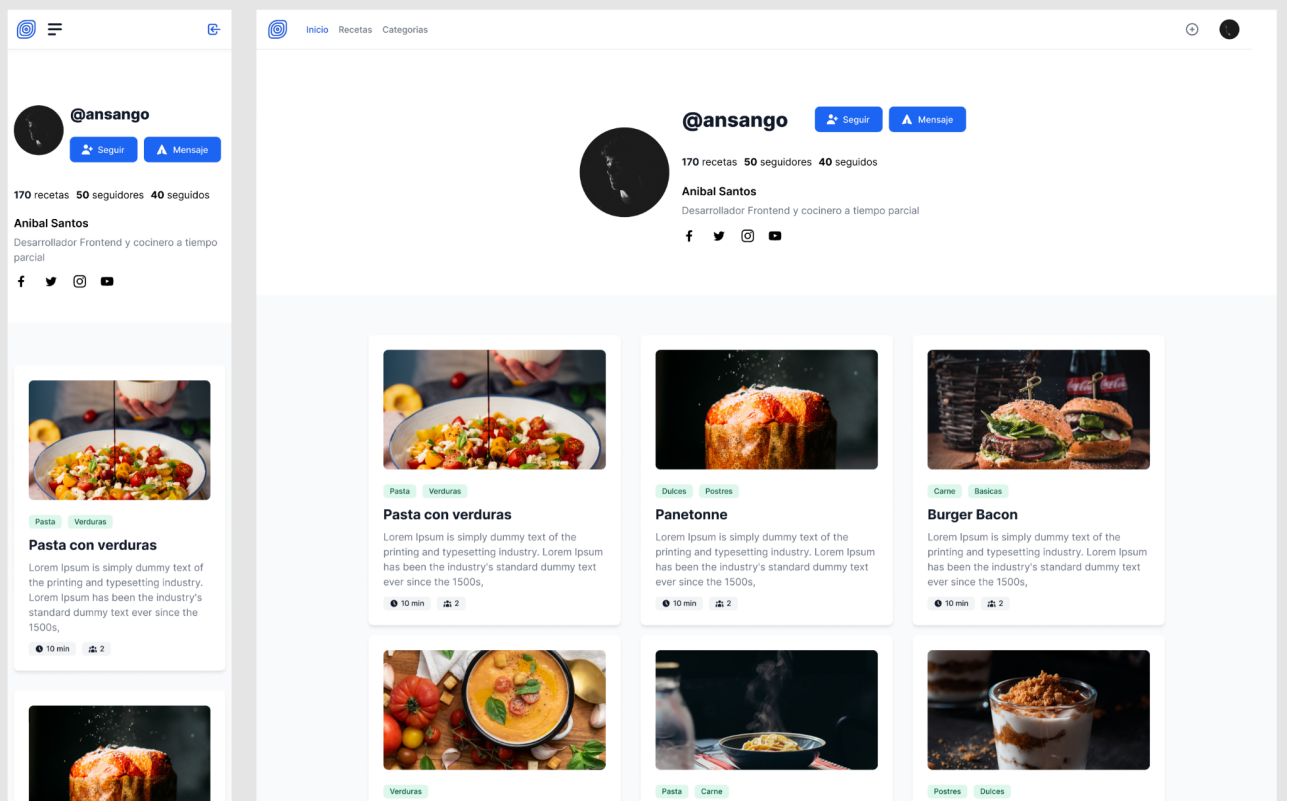


Figura 46: Perfil. Figma.

12.3.8 Crear / Editar receta

Ambas páginas están compuestas del mismo modelo de vista, aquí el usuario se encontrará un formulario compuesto por una serie de pasos, para realizar el proceso de creación de receta. En el caso de que sea nueva creación los campos aparecerán vacíos, y en caso de la edición aparecerán rellenos por la información previamente completada.

Para el acceso a estas vistas el usuario podrá acceder a la creación de una receta mediante el botón en la barra de navegación.

La página de edición de la receta será accesible desde las opciones en la página de la misma y en la página principal de administración. Para eliminar una receta podremos realizarla desde la propia página de edición de receta.

En la siguiente figura podemos ver como será la página de creación o edición de la receta:

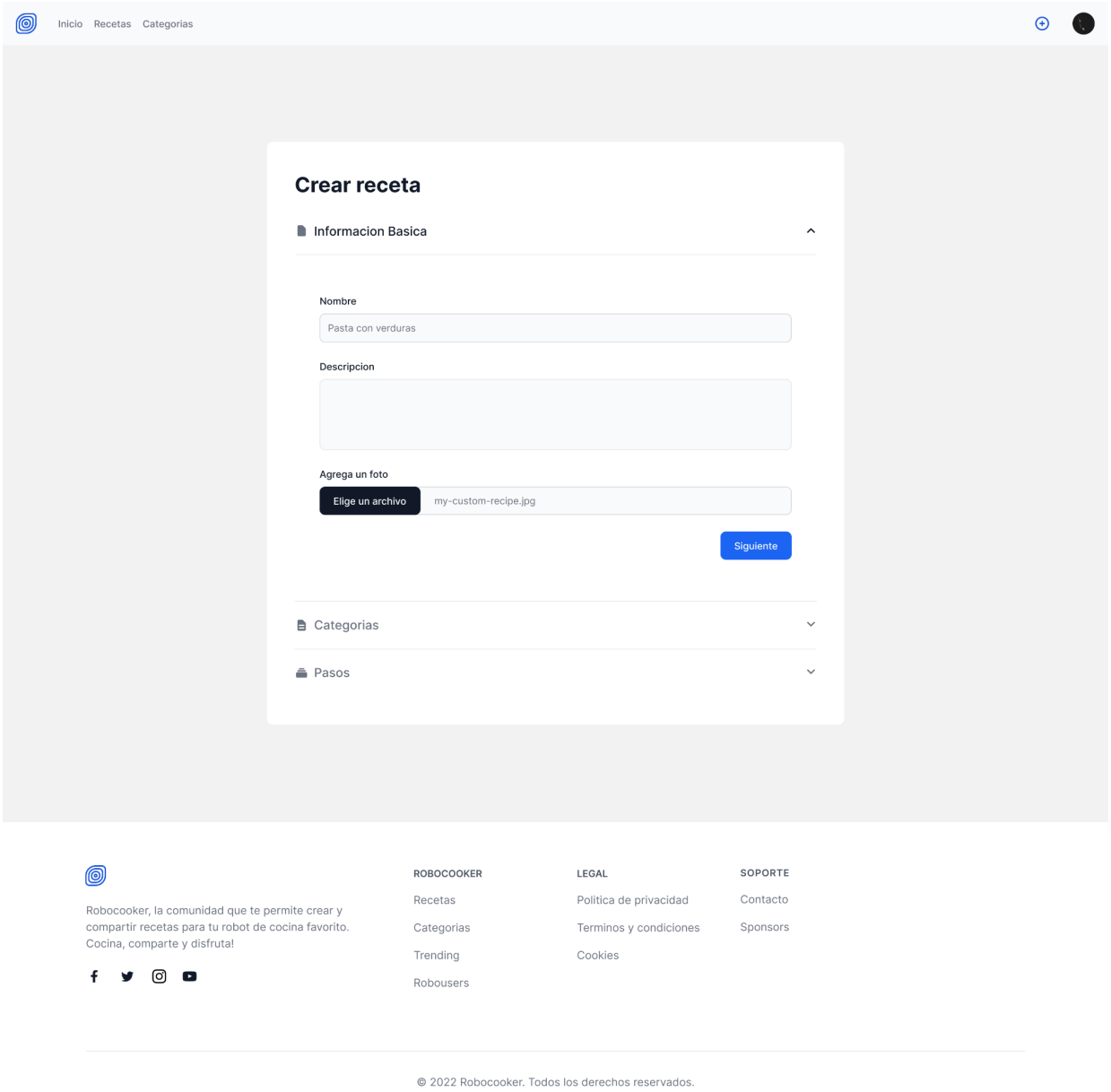


Figura 47: Crear receta. Figma.

12.3.9 Editar perfil

En esta página podremos editar toda la información de nuestra cuenta de usuario. Al acceder nos encontraremos con una serie de bloques. Cada bloque representa un formulario que podrá ser actualizado de forma unitaria.

El usuario podrá actualizar determinados bloques de información separada, así como cambiar su foto de perfil, modificar su contraseña, preferencias, redes sociales y/o eliminar su cuenta de forma permanente.

Cuando complete los procesos de guardado recibirá una notificación de que el proceso se ha completado con éxito o que por el contrario ha fallado por alguna razón. Así en todo momento tendrá un feedback de aquello que está sucediendo.

En la siguiente figura podemos ver la representación de la edición de nuestro perfil de usuario.

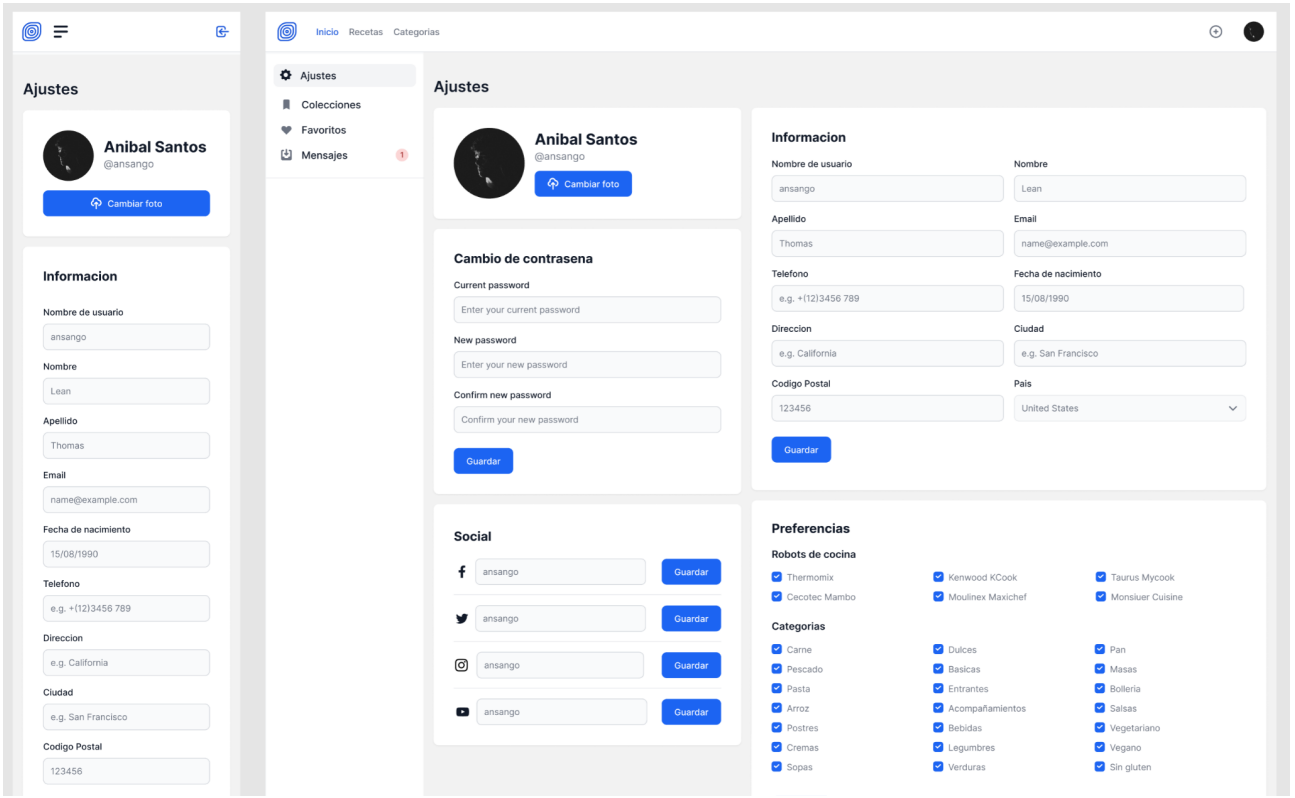


Figura 48: Editar perfil. Figma.

12.3.10 Colecciones

Las colecciones es una sección más avanzada a nivel de funcionalidad de la aplicación. En ella en primera instancia podrá encontrarse con un pequeño bloque donde estarán registradas todas las recetas que haya ido marcado en la correspondiente vista de receta.

Una vez situados en las colecciones el usuario podrá crear una mediante el botón de añadir colección. Aquí se le mostrará un modal para crear una colección, con su selección de recetas, una vez creada, le aparecerá en el panel de colecciones y el modal se cerrará.

En la siguiente figura podemos ver el panel de las colecciones ya creadas por un usuario:

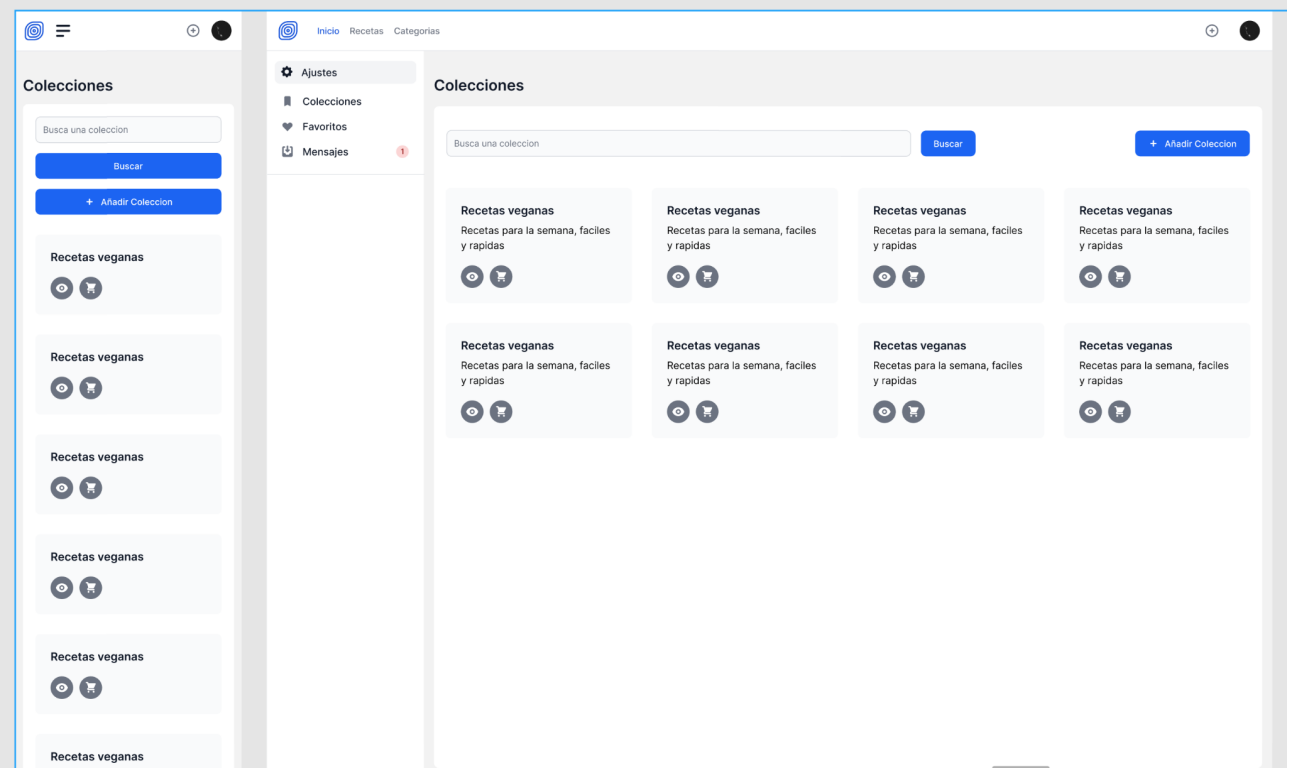


Figura 49: Colecciones. Figma.

Al acceder a la colección de recetas el usuario podrá visualizar una tabla con las recetas que conforman la colección. En dicha tabla, además de ver un resumen de algunas de las características de las recetas, podrá navegar a la página de detalle de dicha receta, o eliminarlas de la colección de forma permanente.

Además tendrá otra pestaña que le permitirá cambiar de vista para ver los ingredientes que conforman dicha colección de recetas. Aquí veremos toda la lista de la compra, con la suma de todos los ingredientes de todas las recetas, si los ingredientes coinciden en unidad y nombre se sumarán de forma automática para evitar tener duplicidades.

A nivel superior tendremos unas migas de pan que le facilitarán regresar a la vista general de colecciones de forma intuitiva.

En las siguientes figuras podemos ver por un lado la tabla mencionada anteriormente y la lista de ingredientes:

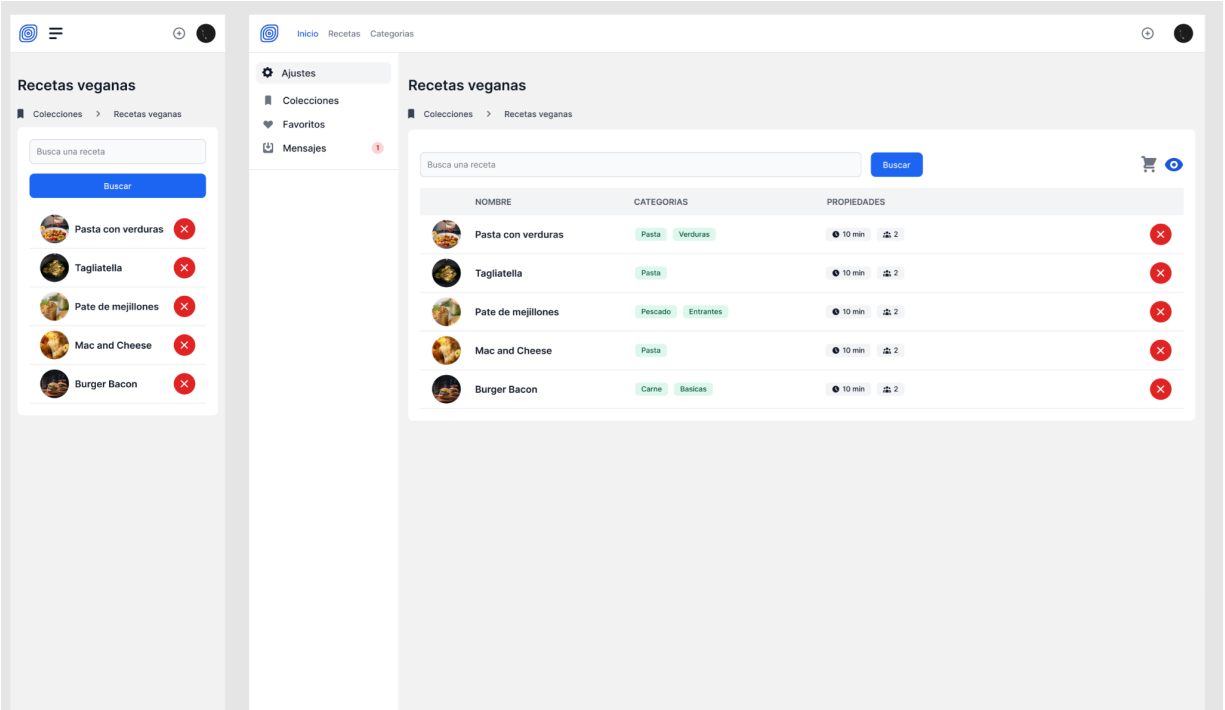


Figura 50: Colecciones / Recetas. Figma.

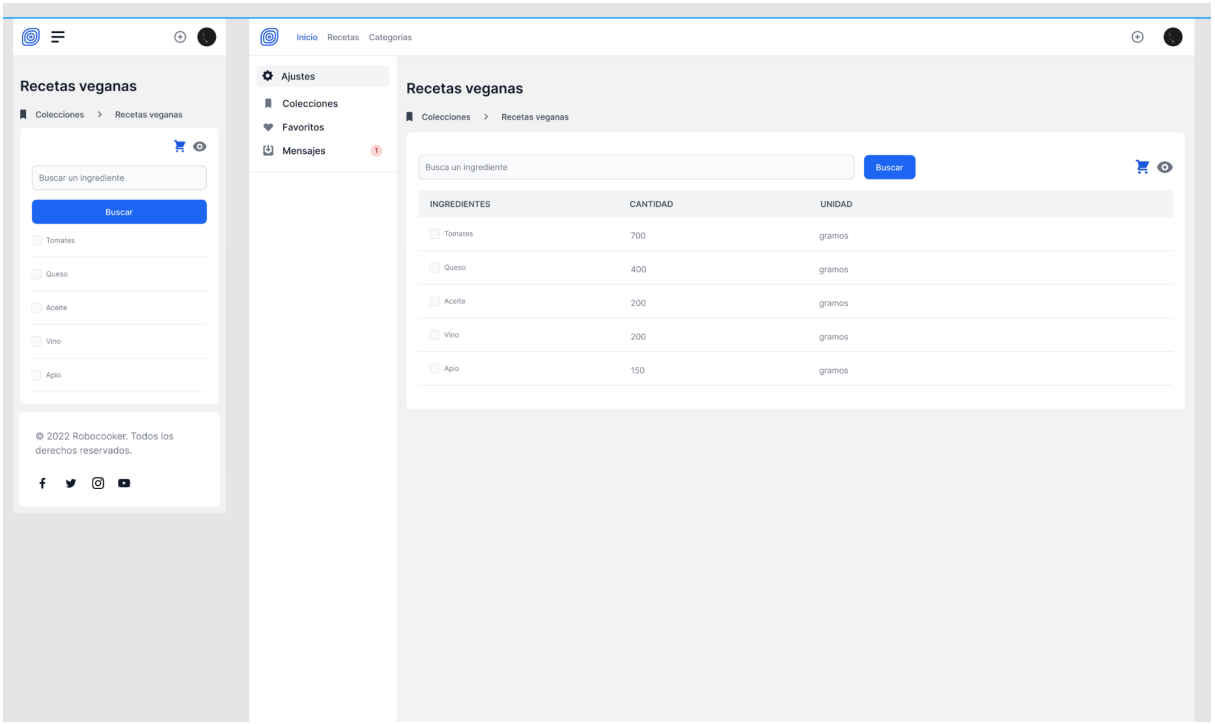


Figura 51: Colecciones / Ingredientes. Figma.

12.3.11 Favoritos

Por último en la sección de favoritos podremos ver las publicaciones a las que hemos reaccionado con un "me gusta". Podremos buscar una receta en concreto y/o eliminarla y que desaparezca de nuestros favoritos de forma permanente.

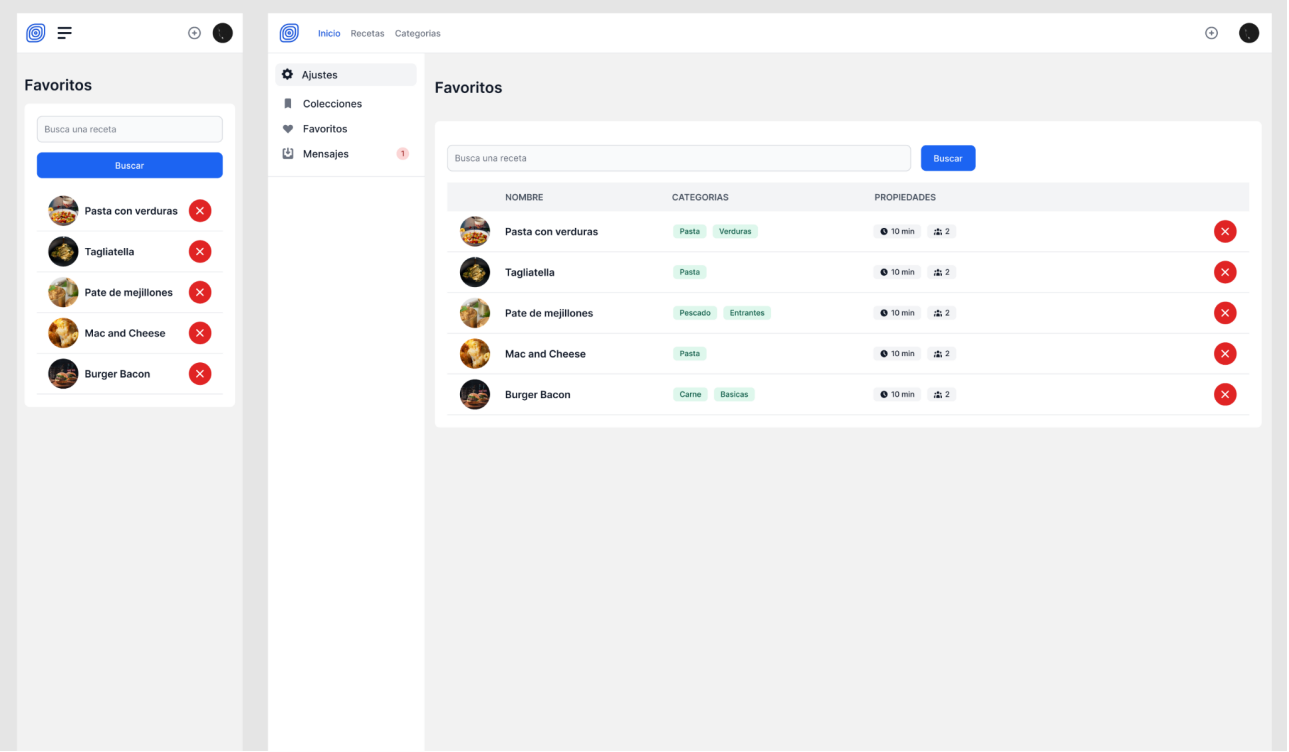


Figura 52: Favoritos. Figma.

13. Implementación

En este apartado describiremos el proceso puramente de desarrollo y publicación en producción de nuestro producto. Se subdivide en las fases de configuración del **entorno de desarrollo**, **desarrollo**, donde abordaremos las cuestiones de organización y creación de capas del código, la **seguridad**, donde abordaremos la autenticación de usuarios, protocolos encriptaciones y variables de entorno, la fase de **pruebas** de usuario e incidencias, el **versionado** y por último los **despliegues** en diferentes entornos.

13.1 Entorno de desarrollo

Para comenzar esta fase, crearemos en primer lugar nuestro proyecto en nuestro entorno local, para ello deberemos tener instalado Node.js y npm, abriremos la consola de comandos y ejecutaremos el siguiente comando:

npx create-next-app@latest --typescript

Este comando creará el proyecto en Next.js configurado ya para programar en Typescript, para arrancar el proyecto de forma local podremos ejecutar *npm run dev*, esto nos levantará un servidor local en http://localhost:3000/. El resto de documentación relativa a las configuraciones podremos consultarla en la documentación oficial de **Next.js**.

Posteriormente deberemos crear el repositorio repositorio remoto en Github de forma pública, configuraremos la rama de **develop** en git, para evitar trabajar con la rama **main,** que será nuestra rama principal para los despliegues en producción. El repositorio quedará accesible en este enlace.

A continuación configuraremos nuestro nuevo proyecto en **Vercel**, para poder tener automatizado el sistema de despliegue.

La configuración es sencilla, únicamente nos autenticamos mediante **Github** en la plataforma y seleccionaremos la opción de importación para importar el repositorio que acabamos de crear. Una vez acabe el proceso tendremos nuestro primer despliegue y un histórico de despliegues. Además podremos configurar las variables de entorno, esta característica es muy importante, ya que nos permitirá ocultar las configuraciones y las api keys de proveedores externos, que normalmente son tokens que no queremos hacer públicos en el código del repositorio.

En la siguiente figura podemos ver el histórico de despliegues del repositorio:

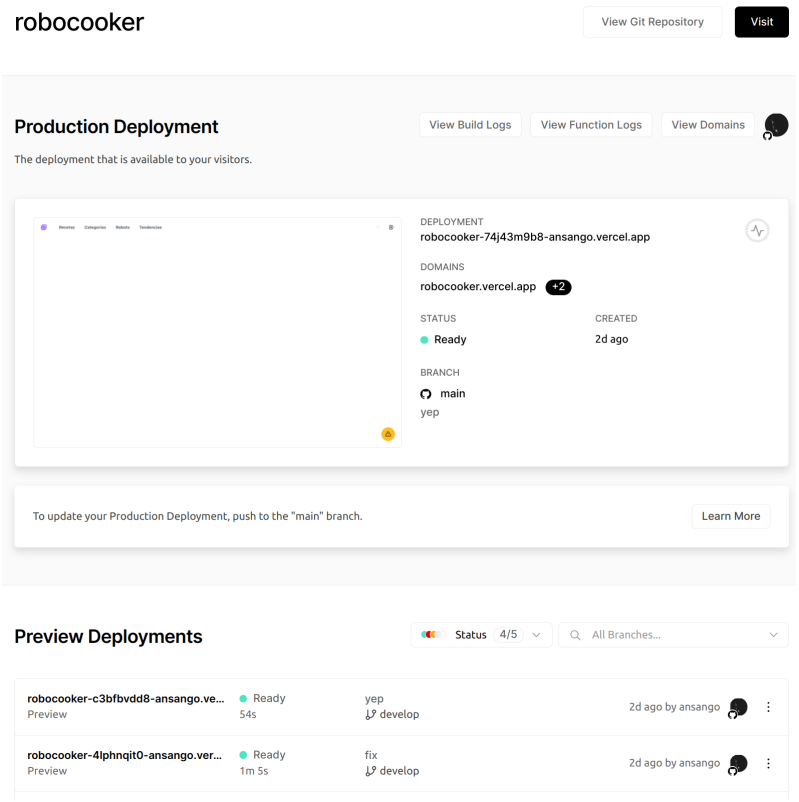


Figura 53: Robocooker - Vercel

Aquí podemos ver la información relativa al éxito o fallo de los despliegues, los commits en git que lo producen, urls de dominio alternativas, la url de dominio principal y las urls de los preview deployments, que corresponderá siempre a ramas que no son la **main**, en nuestro caso corresponderá a la rama **develop**.

Seguidamente crearemos una cuenta para el servicio de estáticos de **Cloudinary,** al crear la cuenta podremos tener acceso a la api key, al api secret y al nombre de nuestra nube. Nos copiaremos estos datos y a continuación en nuestro proyecto crearemos las claves en un fichero denominado “env.local”, que podremos encontrar en la raíz del proyecto. En la siguiente figura podemos ver como quedaría:

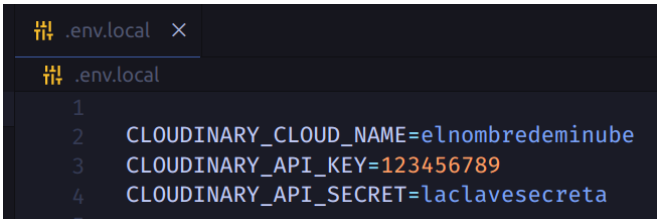


Figura 54: Variables de entorno

A continuación crearemos un cluster en **Atlas Mongodb**, y crearemos dentro de este nuestra primera base de datos, lo ideal sería tener dos, una para producción y otra para desarrollo. Para poder configurarlo con nuestro proyecto, únicamente accederemos a la configuración y aquí se nos proporcionará una url que conectaremos con nuestro driver de Mongodb en Node.js de la siguiente manera:

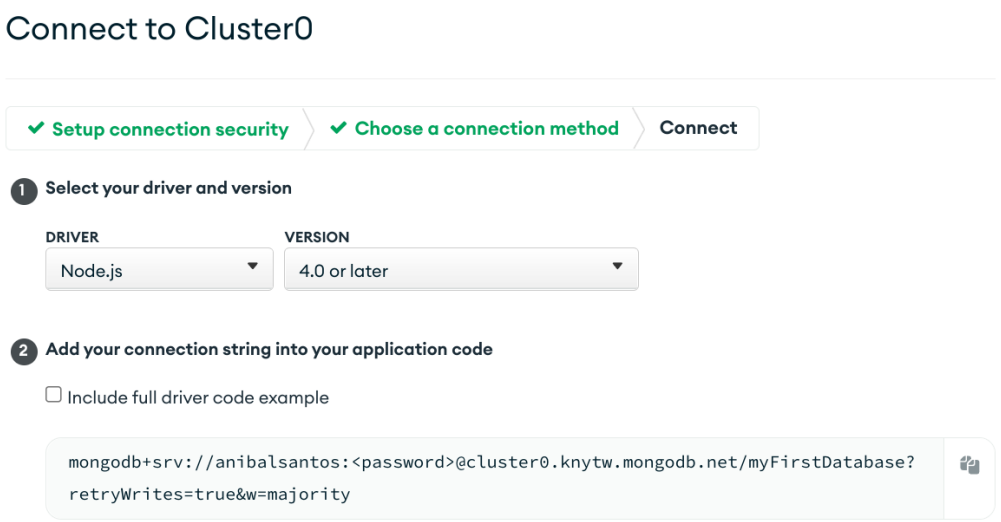


Figura 55: Cluster y conexión en Atlas Cloud

Copiaremos la url de Mongodb y la añadiremos a nuestro fichero de variables de entorno local:

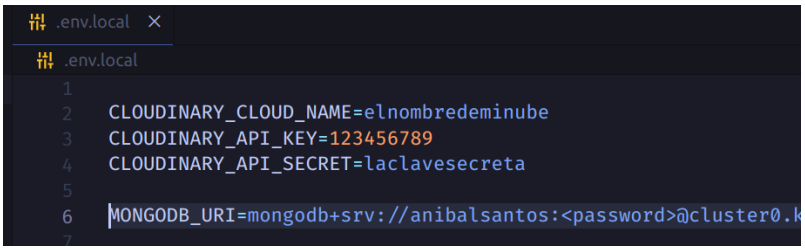


Figura 56: Variable de entorno de Mongodb

Con todo esto ya tendremos listas las configuraciones que necesita nuestro proyecto para que podamos trabajar de forma local consumiendo los servicios que utilizaremos en la nube. Únicamente nos quedaría añadir estas variables de entorno en Vercel, como ya mencionamos antes, esto podríamos hacerlo accediendo a los ajustes de nuestro proyecto, como se muestra en la siguiente figura:

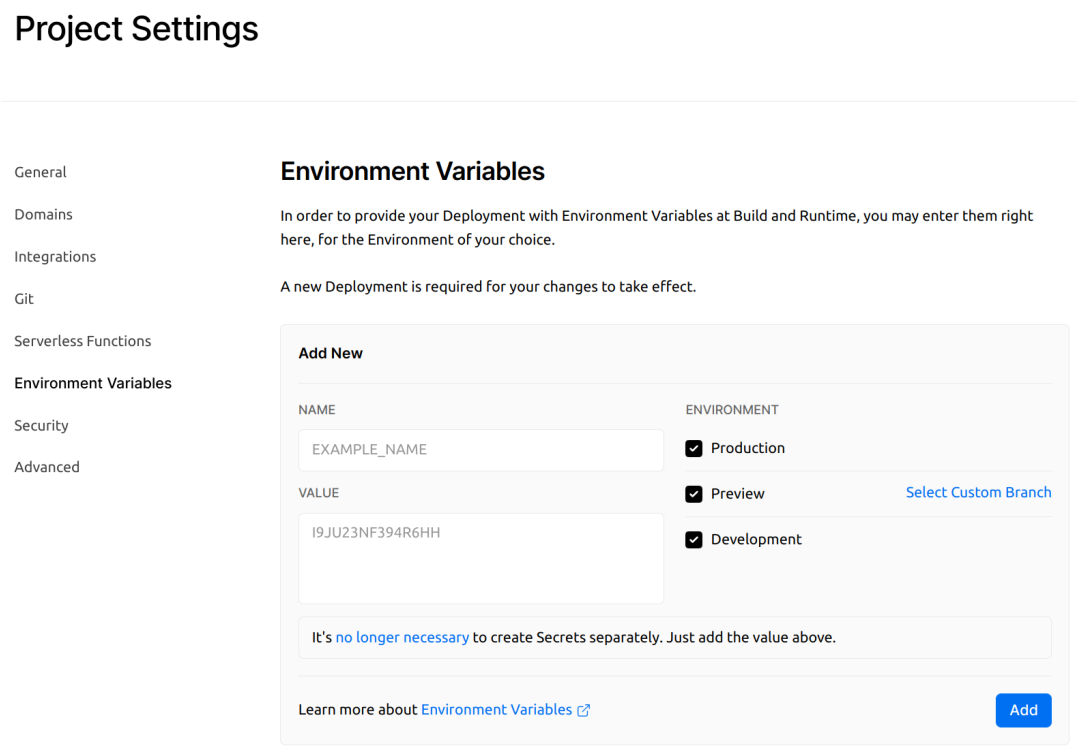


Figura 57: Variables de entorno en Vercel

A partir de aquí tendremos todas nuestras herramientas configuradas y podríamos empezar con el desarrollo del proyecto.

13.2 Desarrollo

El desarrollo del proyecto seguirá el índice comentado en el punto 8.2. Sprints, concretamente partiremos del sexto sprint, ya que los cinco anteriores estarían completados como hemos podido ir viendo a lo largo de la presente memoria.

Las dependencias de backend superados el sexto y séptimo sprint quedarían de la siguiente manera:

- **lib/api**: aquí tendremos la configuración de todo lo que consumirá la API:
  - **auth:** configuración de la librería **Passport.js**, para las autenticaciones de los usuarios.
  - **db:** todas las consultas contra base de datos organizadas por entidades.
  - **issues**: configuración y template para la generación de una tabla de incidencias en **Notion.**
  - **mail:** configuración y template con **Nodemailer,** para la generación de los servicios de mailing.
  - **middlewares:**
    - **auth:** encontraremos la configuración conjunta de, la sesión, y la inicialización de Passport y la sesión de Passport.

- **database:** el driver y la configuración con el cliente de **Mongodb**.
- **session:** la configuración de la sesión.

- **pages/api**: encontraremos todos los controladores de la API Rest, ya que Next.js nos provee de una configuración del router por dependencias y ficheros de forma dinámica.

Normalmente la relación entre el backend y la base de datos será de endpoint y consulta. los endpoints quedarían configurados mediante las páginas de la api y estos controladores accederán mediante operaciones a la base de datos. En las dos siguientes figuras podemos ver un ejemplo de página primeramente y posteriormente de una consulta a base de datos:

```
const handler = nc(options);
handler.use(database, ...auth);

handler.get(async (req, res) => {
  if (!req.user) return res.json({ account: null });
  const account = await findAccountByUserId(req.db, req.user.accountId);
  return res.json({ account });
});
```

Figura 58: API - Cuenta del usuario

```
export const findAccountByUserId = (db: Db, accountId: AccountId) => {
  return db
    .collection("accounts")
    .findOne({ _id: new ObjectId(accountId) })
    .then((account) => account || null);
};
```

Figura 59: Base de Datos - Cuenta del usuario

En cuanto al desarrollo del frontend,crearemos los componentes página y componentes visuales:

- **pages:** son los componentes que utilizaremos como contenedores de cada página de la aplicación. Tendremos los siguientes:
  - blender
  - category
  - dashboard:
    - profile
    - collections
    - my-recipes
    - favorites

        - recipe-add
    - profile
    - recipe
    - recovery
    - verify
    - _app
    - _document
    - 400
    - 500
    - blenders
    - categories
    - index
    - signin
    - signup
    - trending
    - robousers

Cada uno de estos componentes es página en nuestra aplicación. Y se compondrá de componentes genéricos o dedicados de la interfaz. En la siguiente figura podemos ver un componente página:

```tsx
const Categories: NextPage = () => {
  return (
    <MainLayout>
      <GenericHero
        title="Categorías"
        description="Aquí encontrarás todas las categorías de recetas. 21 c
      />
      <div className="p-5">
        <div className="container mx-auto">
          <motion.ul
            className="grid gap-5 sm:grid-cols-2 lg:grid-cols-3"
            initial={{ opacity: 0 }}
            animate={{ opacity: 1 }}
            transition={{ duration: 1 }}
          >
            {categories.map((category) => (
              <Card key={category._id} data={category} kind="category" />
            ))}
          </motion.ul>
        </div>
      </div>
      <div className="pb-20"></div>
    </MainLayout>
  );
};

export default Categories;
```

Figura 60: Componente página - Categorías

- **components**: serán los componentes que componen la ui, en muchos casos serán **dedicados a cada página**, pero tendremos también, **layouts**, que serán las diferentes capas de presentación, y **comunes**, que serán bloques consumidos normalmente por los layouts y otras entidades genéricas.

  A continuación podemos ver un ejemplo de un componente layout:

```tsx
const MainLayout: FC<Props> = ({ children }) => {      You, 3
  const { route } = useRouter();

  return (
    <>
      <Navbar />
      <motion.main
        key={route}
        initial={{ opacity: 0 }}
        animate={{ opacity: 1 }}
        transition={{ delay: 0.25 }}
        className="h-full"
      >
        {children}
      </motion.main>

      <Footer />
    </>
  );
};

export default MainLayout;
```

Figura 61: Componente layout - Main Layout

13.3 Seguridad

En el apartado de seguridad hemos implementado un mínimo de seguridad en nuestra aplicación, basado fundamentalmente en:

- **Protocolo HTTPS,** es un protocolo de comunicación de Internet que protege la integridad y la confidencialidad de los datos de los usuarios entre sus ordenadores y el sitio. El envío de datos mediante el protocolo HTTPS está protegido con el protocolo Seguridad en la capa de transporte (Transport Layer Security, TLS), que proporciona estas tres capas de seguridad principales:

  - Cifrado: se cifran los datos intercambiados para mantenerlos a salvo de miradas indiscretas.

  - Integridad de los datos: los datos no pueden modificarse ni dañarse durante las transferencias.

  - Autenticación: demuestra que nuestros usuarios se comunican con el sitio web previsto.

- **Passport.js**, es un middleware de autenticación para Node.js. Extremadamente flexible y modular, Passport puede incorporarse discretamente a cualquier aplicación web basada en Express. Esta librería nos permite manejar a los usuarios, las sesiones y el acceso a datos que no son públicos y que pertenecen a cada cuenta de usuario.

  El middleware nos decorará los servicios y podremos saber en todo momento si tiene acceso mediante la sesión de usuario configurada en Passport, si no existe la sesión, no podrá acceder a determinados servicios de la API.

- **Bcrypt,** es una librería de Node.js basada en una función de hashing de contraseñas diseñada por Niels Provos y David Mazières, basada en el cifrado Blowfish. Esta librería adaptada a node nos permitirá cifrar las contraseñas almacenadas en base de datos y no comprometer los accesos de los usuarios almacenados en base de datos.

- **Variables de entorno**: como ya hemos visto en los puntos anteriores de configuración de entorno de desarrollo, las variables de entorno nos permiten no hacer públicos nuestras claves secretas que normalmente se asocian a configuraciones del proyecto como la url del cluster de base de datos, o servicios externos utilizados, como es el caso, de Cloudinary, Notion o la configuración de correo con Nodemailer.

  Next.js configura por defecto **dotenv,** que es un módulo que carga las variables de entorno mediante un archivo de configuración (.env.local). De esta manera al subir nuestro código al repositorio remoto nuestras variables en Node no quedarán expuestas y accesibles a miradas indiscretas que puedan utilizarlas.

13.4 Testing

En el apartado de pruebas, lo idóneo sería haber configurado al menos test automatizado de pruebas unitarias con **Jest**, **Storybook** y End to end con **Cypress** y obtener mediante estas dos herramientas algún tipo de cobertura del código y funcionalidad, que podríamos ver representadas en **SonarQube**, pero por falta de tiempo quedará asignado al roadmap de desarrollo de calidad del producto.

En cambio, para poder medir qué deberíamos corregir o mejorar en nuestra aplicación, hemos realizado una serie de pruebas por features básicas que quedarán adjuntadas junto al presente proyecto y memoria en un documento externo denominado **Quality Assurance**.

Además hemos completado todo este proceso mediante pruebas de usuarios reales que han accedido a la plataforma sin ninguna información previa, y que tenían a su disposición la forma de notificar funcionamientos anómalos. Los usuarios pueden completar un pequeño formulario notificando la incidencia observada. Dicha incidencia será almacenada en una tabla en **Notion**, creada específicamente para la revisión de este tipo de sucesos.

Esto nos permitirá revisar y analizar el comportamiento de los usuarios, si el producto se ajusta a sus expectativas, y mejorar nuestro servicio. Las incidencias entrarán en nuestro roadmap para poder iterar las nuevas fases de desarrollo y asignar dichas incidencias a determinados sprints que puedan coincidir con mejoras o funcionalidades nuevas. A continuación se muestra como un usuario enviaría una incidencia encontrada:

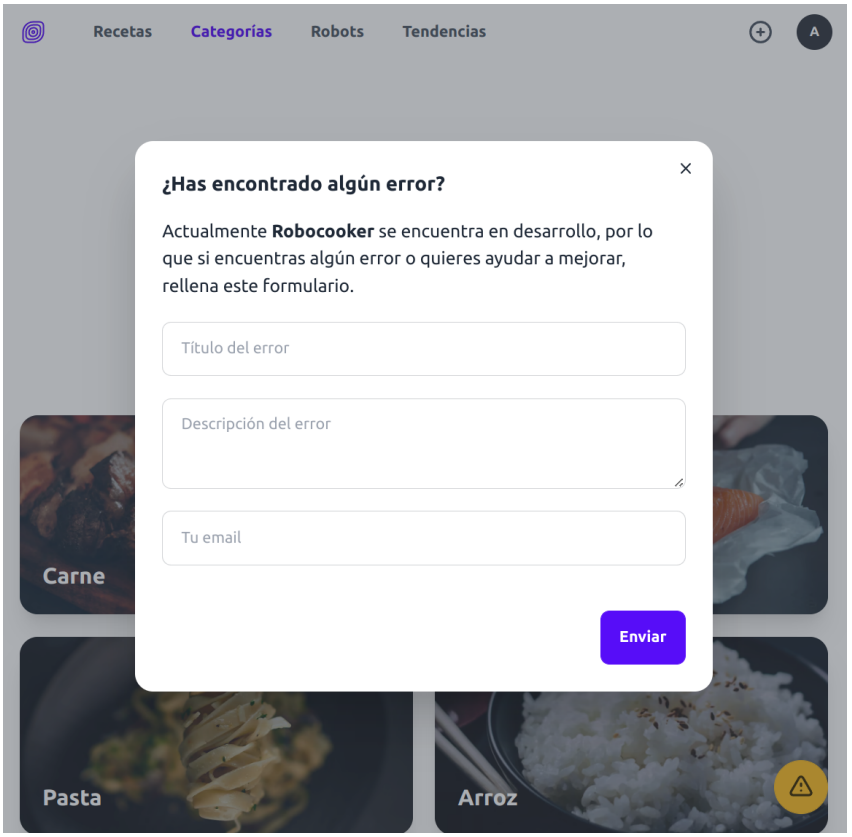


Figura 62: Notificación de incidencias - Robocooker

Como vemos en la figura anterior el usuario, accederá a un formulario donde completará los campos relativos al nombre del error, la descripción del mismo y un correo de contacto.

Posteriormente al enviar este formulario, se almacenará mediante un servicio, una entrada en una tabla en **Notion**, que hemos configurado para esta situación. A continuación podemos ver como se irían añadiendo dichas incidencias:

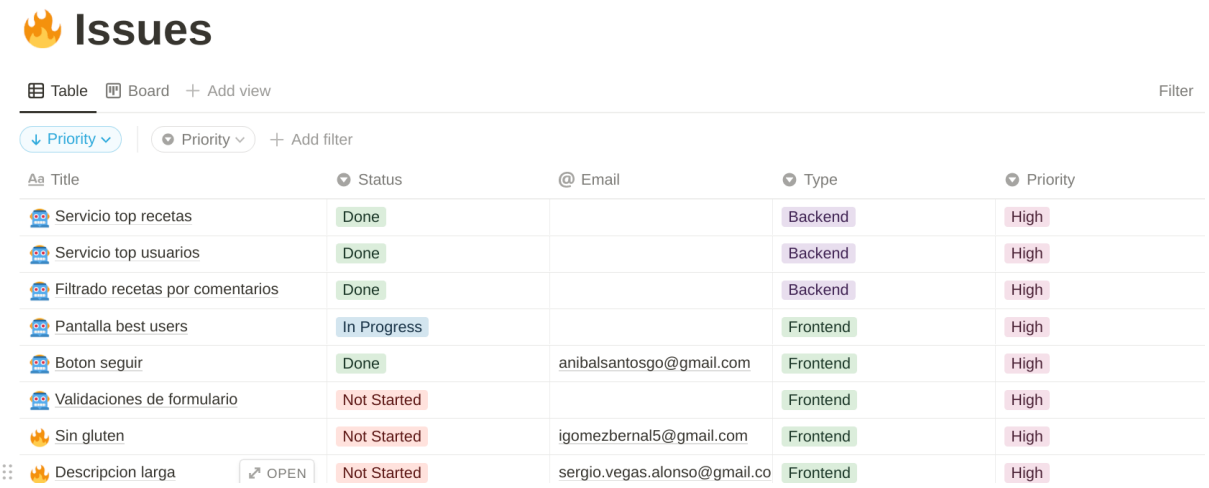


Figura 63: Notificación de incidencias - Notion

En la figura anterior podemos observar que las dos últimas filas corresponden a usuarios que han notificado una incidencia. Las incidencias creadas en esta tabla por los usuarios aparecen siempre con un “emoji” de fuego. Ya que las vistas de las tablas son configurables en Notion, nosotros podremos establecer qué tareas vamos a desarrollar y dejar para otro momento aquellas que no nos interesen. A continuación se presenta la misma tabla pero con la vista de Kanban, donde no estamos teniendo en cuenta otras incidencias:

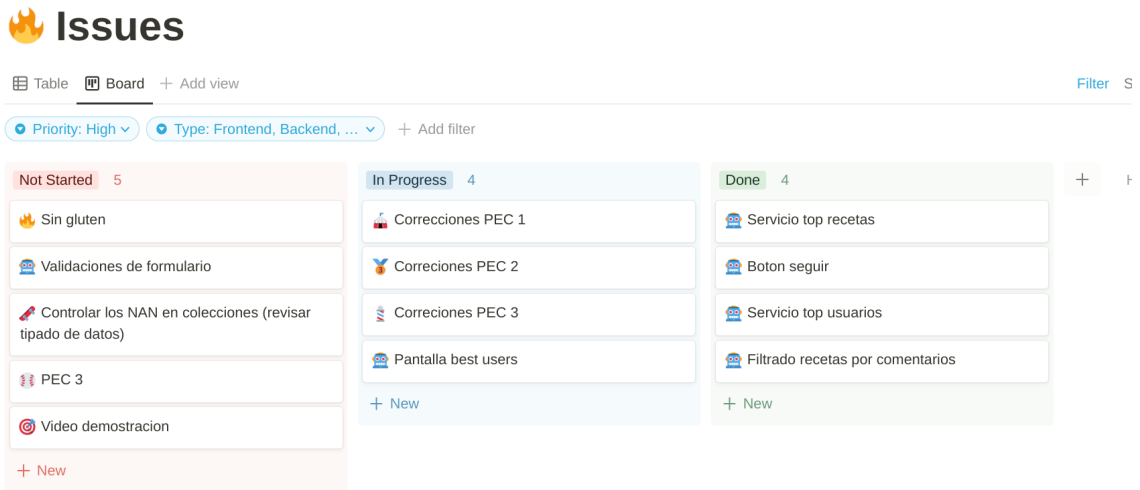


Figura 64: Kanban - Notion

13.5 Despliegues

En este último punto de la implementación comentaremos como desplegamos la aplicación en producción y el resto de entornos.

Para llevar a cabo esta operación mediante git, registramos nuestros cambios y commitearemos los mismos a la rama, si estamos en **develop** y hacemos push al repositorio remoto configurado en **Github**, **Vercel** automaticamente empezará a construir la aplicación en modo previo, esto es, que hará una compilación del código y desplegará el proyecto en una url que no puede ser meta indexada por un buscador. Nos asignará una url, como esta (https://robocooker-c3bfbvdd8-ansango.vercel.app/), para ver el resultado.

A continuación vemos un despliegue realizado en el entorno Preview, que es el que vercel asigna a las ramas que no son la principal:

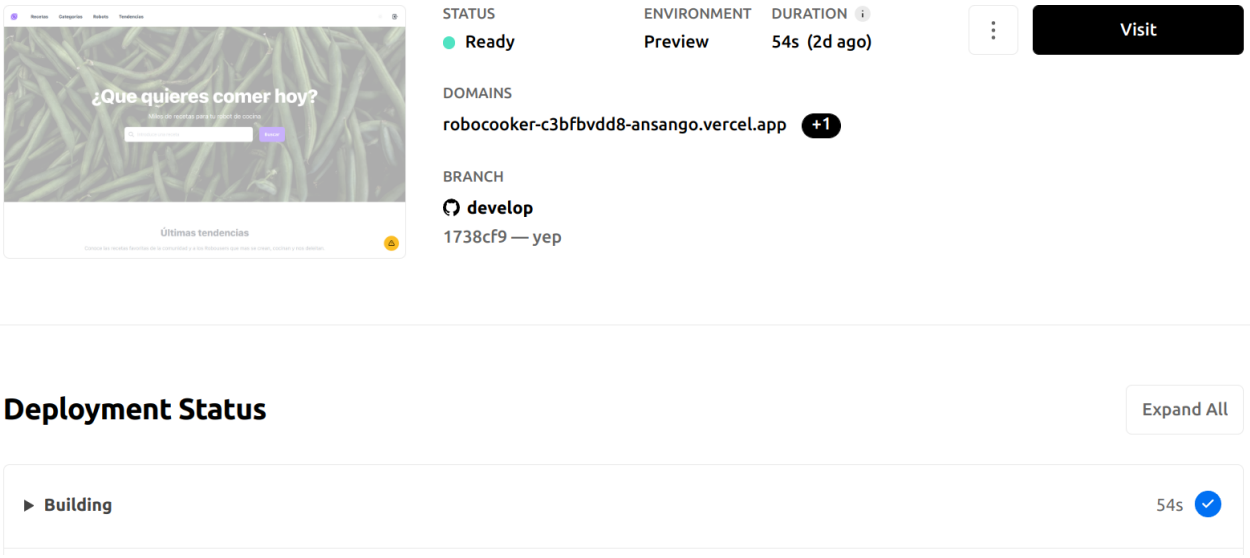


Figura 65: Despliegue previo - Vercel

En cambio si realizamos la operación anterior en la rama **main**, que es la que actualmente tenemos configurada para producción, **Vercel** hará el proceso anterior pero desplegará la url asignada a producción y podremos ver el resultado en nuestro dominio o url de producción.

En la siguiente figura podemos ver el despliegue a producción:

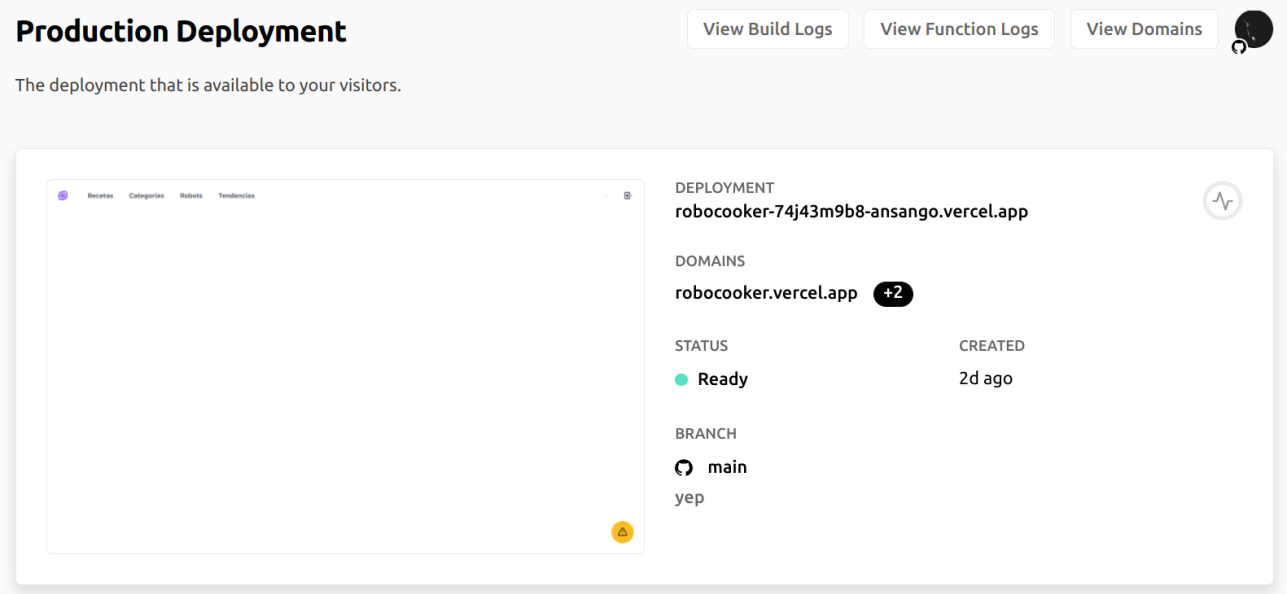


Figura 66: Despliegue en producción - Vercel

Realizado el despliegue a producción podremos ver el resultado en la url o dominio que hayamos configurado, en este caso tenemos una asignación propia del dominio de **Vercel**: https://robocooker.vercel.app/

14. Instalación y uso

Para poder instalar el proyecto necesitaremos tener instalado **git, Node.js,** y una cuenta en **Atlas Mongodb**. Para descargar el proyecto clonaremos con git, o lo descargamos de la siguiente url https://github.com/ansango/robocooker. Una vez descargado instalamos todas las dependencias con Node, mediante el comando npm install.

Cuando Node haya finalizado la instalación de las dependencias, procederemos a configurar las variables de entorno. Dentro del proyecto encontraremos un archivo de ejemplo (.envexample) para configurar las variables de entorno necesarias para poder tener el proyecto plenamente configurado. Debemos renombrarlo por .env.local y asignar los valores que vienen en la plantilla:

- MONGODB_URI=
- CLOUDINARY_CLOUD_NAME=
- CLOUDINARY_API_KEY=
- CLOUDINARY_API_SECRET=
- WEB_URI=
- NEXT_PUBLIC_WEB_URI=
- NODEMAILER_PORT=
- NODEMAILER_HOST=
- NODEMAILER_USER=
- NODEMAILER_PASS=
- NOTION_KEY=
- NOTION_DATABASE_ID=

Una vez configurado, podremos arrancar el proyecto de forma local con el comando *npm run dev* en http://localhost:3000/.

15. Proyección a futuro

Al planificar y desarrollar este proyecto, han faltado determinadas implementaciones sobre las cuales trabajaremos en el futuro, y abordarlas a través de iteraciones.

Podemos definir un **Roadmap** funcional, que nos servirá como guía para listar todo aquello que no hemos podido implementar o añadir, y que nos hubiera gustado tener, pero que por falta de tiempo no ha sido posible. El **Roadmap** lo definiremos en tres puntos, **tecnico, rectificación** y **funcional:**

- **Técnico:** será aquella planificación que nos permita incorporar herramientas técnicas con el objetivo de mejorar nuestra calidad técnica. Aquí podemos destacar la gran ausencia de pruebas unitarias, de componentes y end to end. Para ello se propone la incorporación de las siguientes herramientas:

    - **Jest**, es un framework para pruebas unitarias orientado a Javascript, con el que deberemos probar cada módulo de nuestro producto, tanto componentes como el resto de funciones que lo componen.

    - **Cypress**, es una herramienta Javascript de end-to-end testing. En otras palabras, permite comprobar que la performance de un producto de software recién desarrollado sea buena y corresponda con los requerimientos iniciales. Nos permitirá automatizar las pruebas de todo aquello que se espera que haga el producto desde el punto de vista de un cliente final.

    - **Storybook**, es otra herramienta que nos permitirá crear componentes de UI aislados de la lógica y el contexto de tu aplicación. Así podremos atomizar cada módulo y que cada uno haga la función que deba sin solapamiento de dependencias.

    - **SonarQube,** es una plataforma de código abierto para la inspección continua de la calidad del código a través de diferentes herramientas de análisis estático de código fuente. Proporciona métricas que ayudan a mejorar la calidad del código de un programa permitiendo a los equipos de desarrollo hacer seguimiento y detectar errores y vulnerabilidades de seguridad para mantener el código limpio.

- **Rectificación:** esta planificación está orientada a la corrección exclusiva de las pruebas recogidas en el documento de **Quality Assurance** adjuntado en el presente proyecto y las incidencias de la fase de desarrollo recogidas en **Notion.**

  Los resultados no son críticos, pero este punto es el que debería tener mayor prioridad a la hora de abordarse en nuevas iteraciones del desarrollo de producto.

- **Funcional:** nos han faltado implementaciones que hubieran dotado de una mayor calidad al producto inicial, como hemos diseñado el proceso de trabajo mediante iteraciones con sprints podremos ir implementando dichas funcionalidades mediante **Epics**. A continuación se listan las siguientes funcionalidades no implementadas, por orden de importancia:

  - **Optimización de performance y SEO**: dotar a nuestra web de un buen SEO marcará la diferencia para posicionar nuestro producto en los buscadores principales. Unido al SEO estará la medición y mejora de la performance del proyecto, ya que por ejemplo **Google,** tiene en cuenta este tipo de métricas para penalizar el posicionamiento web de un determinado sitio.

  - **Paginación de las entidades en backend:** con vista a futuro implementaremos un sistema de paginación para poder manejar el consumo de datos del backend, sobre todo en las publicaciones que realizan los usuarios, y poder servir poco a poco los datos.

  - **Filtros avanzados de búsqueda:** mejoraremos el sistema de búsqueda añadiendo filtros por categoría o robot, y así mostrar resultados más concretos en las páginas de búsqueda.

  - **Sistema de mensajería en tiempo real:** por último podremos conectar a unos usuarios con otros a través de un sistema de mensajería en tiempo real basado en websockets, con cifrado de los mensajes y serialización de las conversaciones (cifradas) en base de datos.

16. Conclusiones

16.1 Resultados

Después de finalizar el desarrollo del proyecto final de máster, se concluye, que la mayoir parte de los requisitos inicialmente se han completado satisfactoriamente, ya que debido al corto periodo de tiempo se hizo una previsión de producto mínimo viable para salir a producción. Si bien es cierto que con un margen mayor de plazos de entrega se podrían haber cubierto ciertas necesidades como la automatización de pruebas, o funcionalidades que hubieran dotado a la aplicación de un enriquecimiento mayor para el usuario final.

En cuanto al producto final obtenido, se ha logrado tener una interfaz elaborada, con animaciones, maquetación adaptada no solo a dispositivos de resolución móvil, sino también escritorio de alta resolución, logrando tiempos de carga bastante ajustados para el gran contenido visual recogido.

Como comentamos es importante destacar que uno de los principales pain points era realizar una aplicación manejable en dispositivos móviles, tanto en el proceso de creación de contenido como el manejo y visualización del mismo.

Respecto a la imagen de marca, se ha realizado un branding inicial para este producto que podremos seguir ampliando en el futuro para determinados proyectos relacionados. Éste se basa en un logotipo, colores uniformes y tipografía referencial e iconografía; esto ayudará al usuario a la asociación de nuestro branding con el producto o productos finales derivados.

Uno de los puntos más interesantes bajo mi punto de vista es la creación de una aplicación basada en contenido aparentemente estático y que cuenta con un backend plenamente funcional. El usuario tiene la sensación de estar navegando sobre una página web, donde los tiempos de carga son reducidos y muy manejables. Lo interesante de este punto es que con el diseño de la arquitectura por capas, podemos desacoplar la parte frontal y migrarla, o en su defecto el backend a otro sistema que nos convenga más a la hora de poder hacer la aplicación escalable.

En cuanto al proceso de codificación, como ya se comenta en el apartado de **Proceso de Trabajo**, se ha llevado a cabo por etapas, en este caso, sprints, implementando primero la lógica de negocio y el backend de una manera básica, para posteriormente tener la menor cantidad de lógica en la parte frontend.

Por último destacar la elección de Next.js como tecnología base, porque integra el ecosistema Javascript de forma orgánica y nos permite trabajar con los últimos avances técnicos de una manera fiable ya que esta

herramienta se encuentra bajo el soporte de Vercel. Esto nos permite desarrollar aplicaciones Javascript, compiladas en Rust, dejando de lado a Webpack y beneficiándose de tiempos de compilación muy reducidos. Podemos decir que es un paso hacia el futuro de la web.

El repositorio de código estará disponible en https://github.com/ansango/robocooker y el acceso a la plataforma quedará desplegado en https://robocooker.vercel.app/

16.2 Errores

Uno de los principales errores que he ido observando a lo largo del máster y de mi carrera profesional es el tiempo de estimación en cuanto al desarrollo. Normalmente cuando dominamos una tecnología, podemos hacer una estimación realista de todo aquello que queremos desarrollar. En este caso me enfrentaba a tecnologías con las que no había trabajado y/o profundizado. Por lo que aquí he intentado tener unos requisitos realistas, que de alguna manera fueran iterativos y me permitieran investigar un poco, pensar soluciones y correcciones con algún margen de maniobra.

Reducir un poco los requisitos me ha permitido dotar de un poco más de “aire” al proyecto, y así por ejemplo errores como estimar funcionalidades con más tiempo de desarrollo, han podido ser desechadas en los sprints con un bajo impacto sobre el producto mínimo viable. Esto ha dado lugar a algunas modificaciones en la planificación, pero finalmente no han sido, como comentaba, puntos críticos.

En este caso los principales errores han sido:

- Tratar de crear toda una librería de componentes propia, cuando existen soluciones mixtas o incluso frameworks destinados a este uso.

- Implementar un sistema de autenticación propio, que es plenamente funcional, pero podríamos haber implementado alguna solución más rápida como Firebase.

- Intentar que todos y cada uno de los módulos fueran clean code, obviando fases posteriores para la refactorización. Esto aunque ha conseguido que en general los módulos en muchos casos sean robustos, ha ralentizado el proceso de desarrollo.

Estos errores han reducido el tiempo de desarrollo, pero aún siendo errores puedo afirmar que me han permitido aprender tecnologías nuevas, y otro tipo de soluciones adaptables según plazos y producto.

Anexo 1. Entregables del proyecto

Lista de archivos entregados y su descripción:

Archivo: **PAC_FINAL_SantosGomez_Anibal.zip**
Contiene la entrega completa, documentación, presentación, el vídeo de presentación y el proyecto.

Archivo: **PAC_Final_prj_SantosGomez_Anibal**
Contiene el proyecto completo “robocooker”, frontend y backend, y un readme.md para poder configurarlo correctamente.

Archivo: **PAC_Final_mem_SantosGomez_Anibal**
Contiene la memoria, la arquitectura, las pruebas, todos los prototipos UX/UI, y el informe de autoevaluación.

1. Memoria completa del trabajo de fin de Máster.
    a. **PAC_Final_mem_SantosGomez_Anibal.pdf**
2. Arquitectura, de la aplicación y la base de datos:
    a. Clean Arch.png
    b. Database.png
3. Quality Assurance, pruebas de usuario:
    a. Quality Assurance.ods
4. UX, prototipos, flujos de usuario y sitemap:
    a. Prototypes
    b. User Flows
    c. Sitemap.png
5. Informe de autoevaluación
    a. Informe_Autoevaluacion.pdf

Archivo: **PAC_FINAL_vid_SantosGomez_Anibal**
Contiene el video de presentación de la aplicación.
PAC_FINAL_vid_SantosGomez_Anibal.mp4

Archivo: **PAC_FINAL_prs_SantosGomez_Anibal**
Contiene las presentaciones del Tribunal y Público en general en pdf
PAC_FINAL_prs_tribunal_SantosGomez_Anibal.pdf
PAC_FINAL_prs_publico_SantosGomez_Anibal.pdf

Anexo 2. Código fuente (extractos)

Frontend, ejemplo de un componente página:

```tsx
const Home: NextPage = () => {
  return (
    <MainLayout>
      <Hero />
      <Container>
        <ContainerHeader>
          <Subtitle title="Últimas tendencias" />
          <SubParagraph content="Conoce las recetas favoritas de la comunidad y a los Robousers que mas se crean, cocinan y nos del
        </ContainerHeader>
        <ContainerContent>
          <ContentTrending />
        </ContainerContent>
      </Container>
      <Container>
        <ContainerHeader>
          <Subtitle title="¿Que procesador tienes?" />
          <SubParagraph content="Si buscas recetas orientadas a un procesador de alimentos, este es tu sitio. Actualmente existen l
        </ContainerHeader>
        <ContainerContent>
          <ContentBlenders />
        </ContainerContent>
      </Container>
      <Container>
        <ContainerHeader>
          <Subtitle title="Elige categoría" />
          <SubParagraph content="En búsqueda de la receta perfecta, elige de entre las 21 categorias y que te cojan con las manos e
        </ContainerHeader>
        <ContainerContent>
          <ContentCategories />
          <ContainerLink>
            <ButtonLink href="/categories" label="Ver todas las categorías" />
          </ContainerLink>
        </ContainerContent>
      </Container>
      <div className="■bg-gray-50 □dark:bg-gray-800 py-5">
        <Container>
          <ContainerHeader>
            <Subtitle title="Últimas recetas" />
            <SubParagraph content="Estas son las recetas mas recientes que hemos creado, y que estan siendo vistas por miles de usu
          </ContainerHeader>
          <ContainerContent>
            <ContentLastRecipes />
            <ContainerLink>
              <ButtonLink href="/recipes" label="Ver todas las recetas" />
            </ContainerLink>
          </ContainerContent>
        </Container>
      </div>
    </MainLayout>
  );
};

export default Home;
```

Backend, ejemplo de un endpoint API Rest:

```
const handler = nc(options);
handler.use(database, ...auth);

handler.get(async (req, res) => {
  if (!req.query) {
    res.status(400).json({ error: "Bad request" });
    return;
  }
  try {
    const { id } = req.query;
    const recipe = await findRecipeByIdPopulated(req.db, id);
    return res.status(200).json({ recipe });
  } catch (error) {
    return res.status(500).json({ error: "Internal server error" });
  }
});

handler.delete(async (req, res) => {
  if (!req.user) {
    res.status(401).json({ error: "Unauthorized" });
    return;
  }
  const { id } = req.query as RecipeId;
  if (!id) {
    res.status(400).json({ error: "Missing id" });
    return;
  }
  try {
    const recipeDeleted = await deleteRecipeById(
      req.db,
      id,
      req.user.accountId.toString()
    );
    if (!recipeDeleted) {
      res.status(404).json({ error: "Error trying to delete recipe" });
      return;
    }
    return res.status(204).end();
  } catch (error) {
    return res.status(500).json({ error });
  }
});

export default handler;
```

Anexo 3. Bibliografía

- Next.js, *The React Framework,* 2022. https://nextjs.org/
- React, *A JavaScript library for building user interfaces.* 2022. https://reactjs.org/
- Node.js, *Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine,* 2022. https://nodejs.org
- Express.js, *Fast, unopinionated, minimalist web framework for Node.js,* 2022. https://expressjs.com/
- Typescript, *TypeScript is JavaScript with syntax for types,* 2022. https://www.typescriptlang.org/
- Tailwindcss, *Rapidly build modern websites without ever leaving your HTML.* 2022. https://tailwindcss.com/
- DaisyUI, *The most popular, free and open-source Tailwind CSS component library,* 2022. https://daisyui.com/
- Redux Toolkit, *The official, opinionated, batteries-included toolset for efficient Redux development,* 2022. https://redux-toolkit.js.org/
- Cloudinary, *Unleash the full potential of your digital media,* 2022. https://cloudinary.com/
- Vercel, *Develop. Preview. Ship. For the best frontend teams,* 2022. https://vercel.com/
- Github, *Where the world builds software,* 2022.https://github.com/
- Atlas Mongodb, *Database. Deploy a multi-cloud database* 2022. https://www.mongodb.com/atlas/database
- Sempere, P. *McLuhan en la era de Google: Memorias y profecías de la Aldea Global*. Madrid: Editorial Popular, 2007.
- Rogers, E. M. *Diffusion of Innovations*. New York: Free Press, 2003.
- Aibar, E. *La visión constructivista de la innovación tecnológica. Una introducción al modelo SCOT*. Barcelona: UOC, 2006.
- Funcook**,** *Red social de cocina para descubrir y compartir recetas de cocina, seguir cocineros, despejar dudas sobre temas culinarios o aportar ideas.* 2022. https://funcook.com/
- Cookpad, *La comunidad de cocina casera más grande del mundo. Ten siempre a mano todas tus recetas favoritas y compártelas con quien quieras.* 2022. https://cookpad.com/es

- Recetario Thermomix, *Encuentra más de 18.646 recetas para tu Thermomix,* 2022. https://www.recetario.es/
- Recetas Mambo**,** 2022. https://cecotec.es/recetas-mambo/
- Wikipedia, *Desarrollo ágil de software*, 2022. https://es.wikipedia.org/wiki/Desarrollo_%C3%A1gil_de_software
- J.S. Fernández. *¿Por qué utilizo Clean Architecture?*, XurxoDev, 2016. http://xurxodev.com/por-que-utilizo-clean-architecture-en-mis-proyectos/
- R. C. Martin. Clean Architecture, The Clean Code Blog, 2022. https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html
- G.Kitchen, *The home kitchen in the globalization era,* Consentino S.A, 2019
- Ubuntu, *Ubuntu is the modern, open source operating system on Linux for the enterprise server, desktop, cloud, and IoT.* 2022, https://ubuntu.com/download
- VSCode, *Visual Studio Code is a code editor redefined and optimized for building and debugging modern web and cloud applications,* 2022. https://code.visualstudio.com/
- Git *is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.* 2022. https://git-scm.com/
- npm, *Relied upon by more than 11 million developers worldwide, npm is committed to making JavaScript development elegant, productive, and safe,* 2022. https://www.npmjs.com/
- Postman, *Postman simplifies each step of the API lifecycle and streamlines collaboration,* 2022. https://www.postman.com/
- Mongodb, *When Does MongoDB Make Sense,* 2022. https://www.mongodb.com/
- Notion, *Used by the world's most innovative teams,* 2022. https://www.notion.so/
- Figma, *With Figma, everyone works towards a shared goal. This has enabled our product teams to ship new products faster and feel more confident in their decisions,* 2022. https://www.figma.com/
- Passport.js, *Passport is authentication middleware for Node.js. Extremely flexible and modular.* 2022. https://www.passportjs.org/
- Nodemailer, *Nodemailer is a module for Node.js applications to allow easy as cake email sending,* 2022. https://nodemailer.com/

- Bcrypt, *A library to help you hash passwords.* 2022. https://www.npmjs.com/package/bcrypt
- dotenv, *Dotenv is a zero-dependency module that loads environment variables from a .env file into process.env. Storing configuration in the environment separate from code is based on The Twelve-Factor App methodology.,* 2022. https://www.npmjs.com/package/dotenv
- Jest, *Jest is a delightful JavaScript Testing Framework with a focus on simplicity.* 2022, https://jestjs.io/
- Storybook, *Storybook is an open source tool for building UI components and pages in isolation. It streamlines UI development, testing, and documentation.* 2022. https://storybook.js.org/
- Cypress, *JavaScript End to End Testing Framework,* 2022. https://www.cypress.io/
- SonarQube, *Code Quality and Code Security,* 2022. https://www.sonarqube.org/

Anexo 4. Vita

Aníbal Santos Gómez, natural de Salamanca y residente en la misma localidad. Es Licenciado en Derecho por la Universidad de Salamanca, finalizando dichos estudios en 2012.

En el 2017 comienza el Ciclo Formativo de Grado Superior de Aplicaciones Multiplataforma en Ilerna Online, finalizando en 2019.

En septiembre de 2019 comienza el presente Máster y habiendo trabajado para Yowi, Iberdrola, Metrópolis.coop, Inditex y Stradivarius.

Apasionado por la tecnología, el desarrollo web, el ecosistema Javascript y el OpenSource.