

Proyecto Final

Presentado por:

Stiven Aguirre Granada - *staguirreg@unal.edu.co*
Andrés Felipe Perdomo Uruburu - *aperdomou@unal.edu.co*
Cesar Camilo Velandia Cubillos - *cevelandiac@unal.edu.co*
Manuel Santiago Mori Ardila - *mmori@unal.edu.co*

Profesor:




Oscar Eduardo Alvarez Rodriguez
oalvarezr@unal.edu.co

Viernes 8 de Febrero



Universidad Nacional de Colombia
Facultad de Ingeniería
Departamento de Ingeniería Sistemas e Industrial
2024

Anexo de Documentos Relacionados:

-  Taller 1 y 2 de proyecto
-  Taller de requerimientos
-  Informe_Codigo_Limpio

PUNTO 2: Levantamiento De Requerimientos

Para la elección del proyecto, se consideraron diversas ideas y, al llegar a una propuesta enfocada en recetas, cada integrante aprobó la idea, alcanzando un consenso mediante votación.

Actualmente, muchas personas buscan inspiración para cocinar, pero a menudo enfrentan dificultades para encontrar recetas confiables y adaptadas a sus preferencias dietéticas. Las plataformas existentes, al no estar dedicadas exclusivamente al mundo culinario, limitan la interacción significativa y dificultan el intercambio de experiencias, así como la mejora de recetas a través de retroalimentación directa.

Por ello, la plataforma propuesta tiene como objetivo ofrecer una experiencia integral para la gestión de recetas de cocina. Se iniciará con un **sistema de registro y autenticación** mediante **correo electrónico**, garantizando un **acceso seguro y personalizado**. Una vez registrados, los usuarios podrán **crear, editar y eliminar** sus propias recetas, incluyendo detalles como el título, descripción, ingredientes, pasos de preparación y contenido multimedia (imágenes y videos) para hacer más claro y atractivo el proceso.

Un **sistema de búsqueda avanzado** será clave para la exploración de recetas. Permitirá a los usuarios **filtrar** por tipo de comida, restricciones dietéticas, nivel de dificultad y tiempo de preparación, asegurando que encuentren fácilmente recetas ajustadas a sus necesidades. Para promover la interacción, se incluirá la opción de **calificar y comentar** las recetas, fomentando un **ambiente colaborativo** donde los usuarios puedan intercambiar ideas y recomendaciones.

Cada usuario tendrá un **perfil** donde podrá **gestionar sus recetas publicadas**, revisar las **valoraciones recibidas** y **seguir a otros cocineros**, creando una comunidad activa de aprendizaje e intercambio. También se incluirá un **sistema de notificaciones por correo electrónico** y la posibilidad de **almacenar recetas favoritas** para acceder a ellas rápidamente.

PUNTO 3: Análisis De Requerimientos

En base al punto anterior identificamos los siguientes requerimientos:

Requerimientos funcionales:

1. Sistema de registro y autenticación de usuarios
2. Publicación de recetas con imágenes y descripción
3. Búsqueda y filtrado de recetas
4. Calificación y comentarios en recetas
5. Edición y eliminación de recetas por el autor
6. Perfil de usuario con recetas publicadas y valoraciones
7. Sistema de notificaciones por correo electrónico
8. Sistema de etiquetas personalizadas para recetas
9. Seguimiento de otros usuarios y feed personalizado

10. Almacenamiento de recetas favoritas
11. Sistema de sugerencias de recetas populares
12. Modo oscuro para la interfaz

Requerimientos no funcionales:

1. Usabilidad
2. Escalabilidad
3. Rendimiento
4. Seguridad
5. Compatibilidad
6. Mantenibilidad

Mediante el método MoSCow y añadiendo estimados de esfuerzo con la secuencia fibonacci realizamos la siguiente priorización de los requerimientos:

Requerimientos funcionales:

● **Must have:**

1. **Sistema de registro y autenticación de usuarios (5)**
Se debe permitir a los usuarios registrarse y autenticarse mediante correo, asegurando acceso seguro a la plataforma.
2. **Publicación de recetas con imágenes y descripción (8)**
Los usuarios deben poder publicar recetas con fotos, descripción, ingredientes y pasos de preparación.
3. **Búsqueda y filtrado de recetas (13)**
El Sistema tiene que realizar búsquedas eficientes con filtros básicos por tipo de comida, tiempo de preparación y etiquetas personalizadas.
4. **Edición y eliminación de recetas por el autor (5)**
Los usuarios necesitan tener la opción de editar o eliminar sus propias recetas para mantener el contenido actualizado y organizado.

● **Should have:**

1. **Calificación y comentarios en recetas (5)**
El sistema debería de permitir la interacción entre usuarios, esto mediante la calificación y comentarios en las recetas.
2. **Perfil de usuario con recetas publicadas y valoraciones (8)**
El usuario debería de poder ver el historial de publicaciones y las valoraciones recibidas por parte de otros usuarios.
3. **Sistema de notificaciones por correo electrónico (5)**
Los usuarios deberían recibir notificaciones cuando reciban comentarios o valoraciones en sus recetas.
4. **Sistema de etiquetas personalizadas para recetas (8)**
Se debería mejorar la organización y búsqueda mediante el uso de etiquetas definidas por los usuarios.

- **Could have:**

1. **Seguimiento de otros usuarios (8)**

Los usuarios podrían seguir a otros para ver sus publicaciones para estar al tanto de su actividad, así como de nuevas recetas por parte de esos usuarios.

2. **Almacenamiento de recetas favoritas (5)**

Función para que los usuarios guarden más fácilmente recetas de su interés en una lista de favoritos, a la cual puedan acceder en cualquier momento.

- **Won't Have:**

1. **Sistema de sugerencias de recetas populares (8)**

No es una funcionalidad prioritaria para una primera versión del sistema. Esto debido a la ausencia de aún un número significativo de recetas en la página.

2. **Modo oscuro para la interfaz (5)**

Mejora estética para personalizar la experiencia visual, pero no esencial en la funcionalidad ni objetivo del proyecto.

Requerimientos no funcionales:

- **Must have:**

1. **Usabilidad (13)**

La interfaz debe ser intuitiva y fácil de usar, permitiendo a los usuarios navegar por la plataforma sin dificultades.

2. **Rendimiento (8)**

Las páginas deben cargar rápidamente, especialmente en la búsqueda de recetas y la visualización de contenido multimedia.

- **Should have:**

1. **Seguridad (8)**

El sistema debería de contar con autenticación segura mediante JWT para proteger el acceso a las cuentas de usuario y garantizar la privacidad de la información.

- **Could have:**

1. **Mantenibilidad (3)**

El sistema podría buscar ser fácil de mantener y actualizar, utilizando patrones de diseño que favorezcan la modularidad y la reutilización del código.

- **Won't Have:**

1. **Escalabilidad (8)**

La plataforma no está diseñada para un gran número de usuarios o crecimiento masivo en esta versión.

2. **Compatibilidad (5)**

La plataforma no buscará ser accesible desde dispositivos móviles y se enfocará en el sitio de escritorio.

Requerimientos funcionales:

Requisito	Descripción	Esfuerzo Estimado	Esfuerzo	Tiempo Estimado
Sistema de registro y autenticación de usuarios	Permite a los usuarios crear cuentas y autenticarse mediante correo electrónico para gestionar sus recetas y perfiles.	Moderado: Requiere configurar una base de datos para usuarios, implementar validación de datos y manejar sesiones de usuario de manera segura.	5	4 días
Publicación de recetas con imágenes y descripción	Los usuarios pueden crear y compartir recetas, incluyendo imágenes y detalles sobre ingredientes y preparación.	Alto: Involucra manejo de carga y almacenamiento de imágenes, validación de formularios y creación de vistas personalizadas para las publicaciones.	8	6 días
Búsqueda y filtrado de recetas	Ofrece búsqueda con filtros (categoría, tipo de dieta, dificultad) para encontrar recetas fácilmente.	Alto: Implica la creación de una lógica de búsqueda avanzada y la optimización de consultas para mantener un buen rendimiento con grandes volúmenes de datos.	13	10 días
Edición y eliminación de recetas por el autor	Los usuarios pueden modificar o eliminar sus recetas publicadas.	Moderado: Requiere manejo de permisos y validación de datos para evitar la edición o eliminación no autorizada.	5	4 días
Calificación y comentarios en recetas	Los usuarios pueden valorar y comentar recetas para compartir opiniones y experiencias.	Moderado: Necesita implementar un sistema de puntuación y manejo de comentarios, asegurando la validación y moderación básica de las interacciones de los usuarios.	5	4 días
Perfil de usuario con recetas publicadas y valoraciones	Muestra la información del usuario, recetas creadas y las valoraciones recibidas.	Moderado: Implica el diseño de una vista de perfil personalizada y la integración con el sistema de publicación y calificación de recetas.	8	6 días
Sistema de notificaciones por correo electrónico	Envía notificaciones a los usuarios sobre comentarios en sus recetas o actividades relacionadas.	Moderado: Necesita la integración con servicios de correo electrónico y la gestión de eventos para activar las notificaciones.	5	4 días
Sistema de etiquetas personalizadas para recetas	Permite asignar etiquetas a las recetas para facilitar su clasificación y búsqueda.	Bajo: Implementación sencilla de etiquetas y su asociación a las recetas en la base de datos.	8	6 días

Seguimiento de otros usuarios y feed personalizado	Los usuarios pueden seguir a otros y ver sus recetas recientes en un feed.	Alto: Requiere lógica de seguimiento, consultas frecuentes a la base de datos y la creación de un feed en tiempo real.	8	6 días
Almacenamiento de recetas favoritas	Permite a los usuarios guardar recetas en una lista de favoritos para consultarlas después.	Bajo: Consiste en añadir una tabla de favoritos en la base de datos y crear una vista para consultar las recetas guardadas.	5	4 días
Sistema de sugerencias de recetas populares	Sugiere recetas basadas en las más valoradas y populares.	Moderado: Implementación de lógica para recopilar datos de popularidad y mostrarlos en la interfaz de usuario.	8	6 días
Modo oscuro para la interfaz	Ofrece a los usuarios la opción de cambiar la apariencia a modo oscuro para mejorar la experiencia de usuario.	Bajo: Ajuste de estilos mediante CSS y configuración de la preferencia del usuario.	5	4 días

Requerimientos no funcionales:

Requisito	Descripción	Esfuerzo Estimado	Esfuerzo	Tiempo Estimado
Usabilidad	La página debe ser intuitiva y fácil de navegar para todos los usuarios, independientemente de su experiencia previa.	Moderado: Involucra pruebas de usuario y ajustes constantes para garantizar una navegación fluida y lógica.	5	4 días
Rendimiento	El tiempo de carga debe ser inferior a 2 segundos para la mayoría de las páginas, incluso con una gran cantidad de datos.	Alto: Implica optimización de código, uso de caché y reducción del tamaño de los recursos servidos (imágenes, scripts).	8	6 días
Seguridad	Implementación de autenticación segura y cifrado de datos sensibles, como contraseñas.	Alto: Necesita configurar medidas de seguridad como el cifrado de contraseñas, protección contra ataques de fuerza bruta y validación de datos de entrada.	8	6 días
Mantenibilidad	El código debe ser limpio y fácil de entender, utilizando herramientas de formato y linters.	Bajo: Requiere la configuración de herramientas como ESLint y Prettier para mantener la consistencia del código.	5	4 días

Compatibilidad	Debe funcionar correctamente en los navegadores modernos más comunes (Chrome, Firefox, Safari, Edge).	Moderado: Involucra pruebas en distintos navegadores y ajustes para asegurar que las funcionalidades sean compatibles en todos ellos.	5	4 días
Escalabilidad	La aplicación debe ser capaz de manejar un aumento significativo de usuarios y recetas sin pérdida de rendimiento.	Alto: Requiere diseño de backend escalable, manejo de concurrencia y optimización de consultas a la base de datos.	8	6 días

PUNTO 4: Análisis Gestión De Software

Tiempo

El tiempo estimado para el desarrollo del proyecto lo hemos basado a partir del punto anterior. Somos 4 estudiantes en etapa de aprendizaje, por lo que los tiempos asignados reflejan esta situación. Para mayor facilidad en el análisis agrupamos las funcionalidades en módulos.

- Distribución del Tiempo Estimado:**

Requerimiento	Tiempo
Sistema de registro y autenticación de usuarios	4 días
Publicación de recetas con imágenes y descripción	6 días
Búsqueda y filtrado de recetas	10 días
Edición y eliminación de recetas por el autor	4 días
Calificación y comentarios en recetas	4 días
Perfil de usuario con recetas publicadas y valoraciones	6 días
Sistema de notificaciones por correo electrónico	4 días
Sistema de etiquetas personalizadas para recetas	6 días
Seguimiento de otros usuarios y feed personalizado	6 días
Almacenamiento de recetas favoritas	4 días

Sistema de sugerencias de recetas populares	No incluido
Modo oscuro para la interfaz	No incluido
Tiempo total	42 Días

Costo

Estimamos los costos con sueldos de desarrolladores, servicios en la nube y herramientas externas. para esto hicimos una conversión de costo en dólares (USD) a pesos colombianos (COP) de **1 USD = 4176 COP**.

1. Sueldos de Desarrolladores

- **Desarrollador Junior:** \$2'200.000 COP / mes
- **Back-End / Front-End (Full-Stack):** \$5'400.000 COP / mes
- **Tester:** \$2'600.000 COP / mes

2. Servicios en la Nube

- **Microsoft Azure:** \$87.159 COP / mes
- **Google Cloud:** \$94.090 COP / mes
- **AWS:** \$91.060 COP / mes

3. Herramientas de Desarrollo

- **Visual Studio Code:** Gratuito
- **Microsoft Visual Studio IDE:** \$186.768 COP / mes
- **JetBrains PyCharm Professional:** \$103.345 COP / mes

● Costo Estimado Mensual:

Concept	Costo en USD	Costo en COP
Sueldos	\$2.441,52	\$10'200.000
Servicios en la nube	\$65,20	\$272.309
Herramientas	\$69,47	\$290.113
Total Mensual	\$2.576,19	\$10'762.422

Alcance:

El proyecto tiene como objetivo entregar una **plataforma web funcional** que permita a los usuarios **gestionar y compartir** recetas de cocina de manera sencilla e intuitiva. El sistema debe ofrecer una experiencia amigable, donde el usuario pueda **registrarse y loguearse, crear y publicar** recetas con imágenes, **buscar y filtrar** recetas según distintos criterios, así como **interactuar con otras publicaciones** mediante **calificaciones y comentarios**. Además, se busca que cada usuario pueda recibir **notificaciones** relevantes y **gestionar** sus propias recetas.

Definimos el alcance del proyecto dividiendo las funcionalidades en **incluidas** y **fuera del alcance** para esta versión inicial, esto a partir de la priorización realizada anteriormente de los requerimientos.

1. Funcionalidades Incluidas:

- Sistema de registro y autenticación
- Publicación de recetas con imágenes y descripción
- Búsqueda y filtrado de recetas
- Edición y eliminación de recetas
- Calificación y comentarios
- Perfil de usuario
- Sistema de notificaciones
- Sistema de etiquetas personalizadas

2. Funcionalidades Fuera del Alcance:

- Seguimiento de otros usuarios y feed personalizado
- Almacenamiento de recetas favoritas
- Sistema de sugerencias de recetas populares
- Modo oscuro para la interfaz

Al dividir las funcionalidades garantizamos que la plataforma sea funcional para los usuarios objetivos, priorizando las funcionalidades más críticas para el funcionamiento adecuado de la página.

PUNTO 5: Diagrama Y Especificación De Casos De Uso Del Sistema

Utilizaremos el siguiente diagrama de casos de uso como referencia para guiarnos en la estructuración de la interacción de los usuarios y el funcionamiento de la plataforma esperado:



PUNTO 8: Diseño y Arquitectura

Identificación de la Arquitectura:

Para nuestra plataforma, hemos elegido una **arquitectura modular basada en MVC (Modelo-Vista-Controlador)**. Esta estructura es ideal porque nos permite separar la lógica de negocio, la presentación y el control del flujo de datos, haciendo el desarrollo mucho más sencillo. Además, facilita que trabajemos en equipo.

En este caso, el **backend** seguirá una arquitectura RESTful usando **Nest.js** para gestionar las solicitudes y manejar la lógica de negocio. Por su parte, el **frontend** será desarrollado en **Flutter**, garantizando una experiencia consistente de escritorio.

Componentes Principales

A continuación, describimos los componentes clave de nuestra arquitectura y cómo se comunican entre sí:

Backend (Nest.js)

- **Modelo:**
 - Se encarga de la representación de los datos de la plataforma, como usuarios, recetas y comentarios.
 - Aquí está toda la lógica de negocio, como validar las recetas y manejar la autenticación.
- **Controlador:**
 - Recibe las solicitudes del frontend, las procesa y decide qué hacer con ellas.
 - Gestiona la interacción entre el Modelo y las Vistas.
- **Base de Datos (MySQL):**
 - Almacena toda la información de la plataforma de forma estructurada.
 - Relaciona tablas como *Usuarios*, *Recetas*, *Comentarios* y *Categorías*.
 - Ofrece integridad y consistencia en los datos a través de relaciones bien definidas.

Frontend (Flutter)

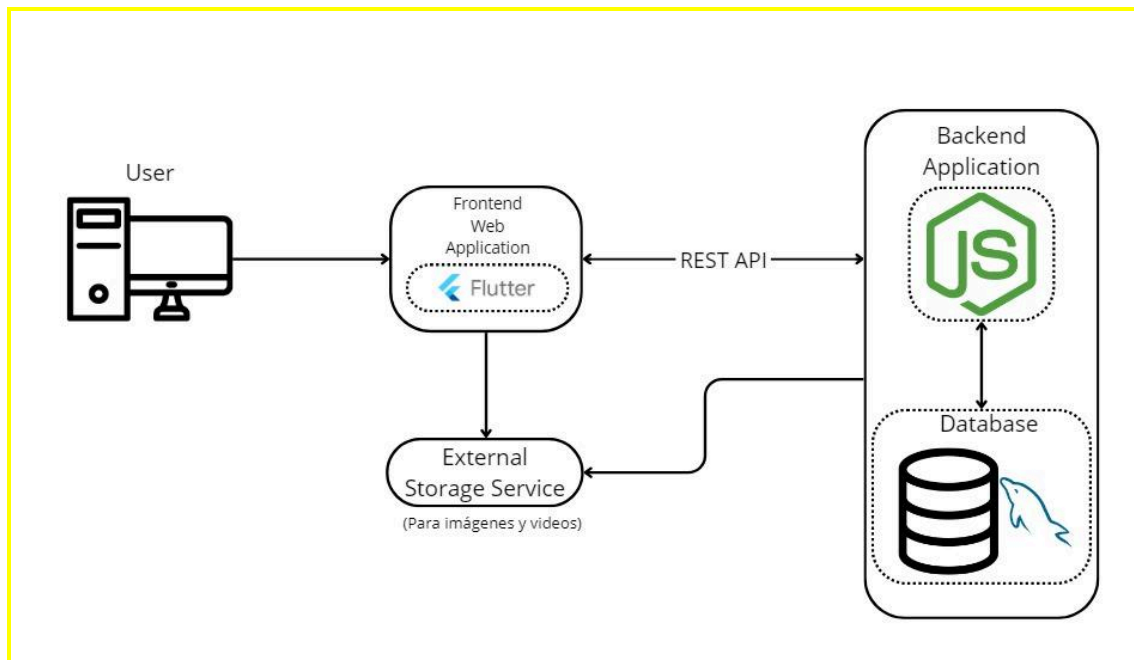
- **Vista (Interfaz de Usuario):**
 - Muestra la información de manera visual y atractiva a los usuarios.
 - Permite buscar recetas, interactuar con comentarios y visualizar contenido multimedia.
 - Soporta múltiples dispositivos, ajustándose a diferentes tamaños de pantalla.

API REST:

- **Interfaz de comunicación entre el frontend y el backend.**
 - Facilita el intercambio de datos.
 - Utiliza JWT para la autenticación segura de los usuarios.

Diagrama de Arquitectura

El siguiente diagrama de arquitectura describe cómo interactúan los distintos componentes del sistema:



1. El usuario accede a la plataforma desde su dispositivo.
2. La Vista (Flutter) muestra la interfaz de usuario y captura las interacciones (cómo buscar recetas o dejar comentarios).
3. El backend procesa la solicitud, interactúa con la base de datos MySQL para obtener la información solicitada y genera una respuesta.
4. El backend procesa la solicitud, consulta la base de datos y devuelve una respuesta al frontend.
5. La Vista actualiza la información mostrada al usuario.

Justificación

- **Arquitectura MVC:** Nos permite dividir el trabajo en diferentes módulos, lo que facilita la colaboración en el equipo del proyecto.
- **Modularidad:** Cada parte del sistema está separada, lo que facilita futuras actualizaciones y nuevas funcionalidades.
- **Flutter:** Garantiza una experiencia de usuario fluida en dispositivos móviles y web.
- **MySQL:** Elegimos esta base de datos por su estabilidad, capacidad de manejar grandes volúmenes de datos y su compatibilidad con múltiples entornos.
- **Seguridad con JWT:** Asegura que solo los usuarios autenticados puedan acceder a ciertas partes del sistema.

PUNTO 9: Patrones de diseño

En el desarrollo de nuestra plataforma, sabemos que el uso de **patrones de diseño** es esencial para garantizar un sistema modular, mantenible y fácil de expandir. Por eso, elegimos algunos patrones clave que se ajustan a las necesidades del proyecto. Estos patrones nos ayudarán a estructurar el código de forma limpia y eficiente, evitando soluciones ineficientes y facilitando la modularidad.

Model-View-Controller (MVC)

- El patrón **MVC** nos permite separar la lógica de negocio (Model), la interfaz de usuario (View) y el control del flujo de datos (Controller). Esto hace que la aplicación sea más fácil de entender, mantener y modificar.
- Mantiene cada parte de la aplicación bien organizada y desacoplada. Si queremos cambiar la forma en la que se muestran las recetas, no necesitamos tocar la lógica de negocio.

Implementación esperada:

- El **Model** representa las recetas, los usuarios y sus interacciones.
- El **View** se encarga de mostrar las recetas, los comentarios y las notificaciones.
- El **Controller** gestiona las solicitudes de los usuarios y decide qué mostrar o cómo procesar las acciones.

Repository

- Facilita la gestión de la base de datos, actuando como una capa intermedia entre la lógica de negocio y las operaciones de persistencia (guardar, consultar o actualizar datos).
- Nos ayuda a mantener el acceso a la base de datos centralizado y fácil de modificar. Si queremos cambiar el motor de base de datos en el futuro, sólo tenemos que modificar el **Repository** en lugar de toda la lógica de la aplicación.

Implementación esperada:

- Consultar todas las recetas destacadas.
- Actualizar la información de un usuario.
- Guardar un nuevo comentario en la base de datos.

Singleton

- Asegura que solo exista una única instancia de ciertas clases esenciales, como la configuración de la aplicación o el servicio de autenticación.
- Evita problemas de inconsistencia y asegura que todos los módulos de la plataforma usen la misma configuración o servicio compartido.

Implementación esperada:

- Servicio de autenticación con JWT.
- Gestión de configuración global de la aplicación.
- Servicio de notificaciones en tiempo real.

Decorator

- Permite añadir funcionalidades de forma dinámica sin modificar el código original. Es perfecto para extender las funciones básicas de la plataforma, como ofrecer opciones avanzadas para usuarios premium.
- Facilita la personalización de la funcionalidad sin hacer el sistema rígido o complejo.

Implementación esperada:

- Añadir opciones avanzadas de filtrado de recetas solo para usuarios registrados.

- Implementar distintas políticas de moderación de comentarios según el tipo de usuario.

Escogimos estos patrones para permitir que el sistema sea más modular, reutilizable y fácil de mantener a largo plazo, asegurando una arquitectura bien estructurada y de calidad.