FOODIERANKJD

Davisson Adriel Roman Carreño
Joan Sebastian Gomez Serrano

U1

DOCENTE:

Pedro Gomez

CAMPUSLANDS
GRUPO U1
FLORIDABLANCA
2025
EXPRESS

1. SITUACIÓN PROBLEMA

En el mundo gastronómico elegir un buen restaurante se basa en opiniones y sugerencias colectivas mayormente, estar bien alimentados y con energía es la tarea más importante de cada individuo. El punto más importante es dónde encontrar un restaurante que brinde los servicios óptimos y preferencias que el cliente espera, las opiniones exteriores y que tan bien se desempeña en la relación de cliente-empleado. Satisfacer al cliente es crucial para su experiencia, reputación del restaurante y pronto regresó, en ocasiones este punto se quiebra ya que al no conocer bien un restaurante puede decepcionar las expectativas del cliente, puede no adaptarse a lo que cliente desea y provocar confusiones por lo que es fundamental implementar una herramienta que acople y solucionar aquellos inconvenientes al encontrar un restaurante que se adapte a los deseos del cliente y la mejor experiencia.

2. LEVANTAMIENTO DE REQUERIMIENTOS

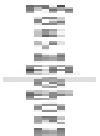
A continuación se presenta la estructura general diseñada para lograr el correcto levantamiento de los requerimientos, en la cual se especifica toda la información relevante del proyecto a desarrollar.

- **Información general del proyecto**
  - Nombre del proyecto: FOODIERANKJD
  - Fecha de reunión: 27/10/2025

- **Integrantes del proyecto**
  - Product Owner: Davisson Adriel Roman
  - Scrum Master: Joan Sebastian Serrano
  - Developers: Davisson Adriel Roman y Joan Sebastian Serrano

- **Objetivo del proyecto:** Desarrollar una aplicación full-stack utilizando Node.js y Express para el backend, y HTML y CSS puro para el frontend, que permita a los usuarios registrar, calificar y rankear restaurantes y platos. El proyecto estará enfocado en implementar un sistema completo de gestión de usuarios, reseñas, categorías y rankings, diferenciando los permisos de usuario y administrador. Además, se integrará un sistema de autenticación segura, validaciones robustas y un frontend que consuma eficientemente la API desarrollada, asegurando una experiencia fluida y funcional.

- **Necesidades del cliente**
  - ¿Cual es el problema actual que se quiere resolver?



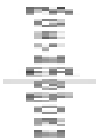
Se requiere la creación de una plataforma web que centralice la información gastronómica, valide las reseñas y brinde un sistema de calificación más ordenado y útil tanto para usuarios como para administradores.

- ¿Qué motivó este proyecto?

  Crear una solución práctica y accesible que mejore la forma en que las personas descubren y evalúan restaurantes y platos. A nivel técnico, este proyecto representa una oportunidad para integrar conocimientos de desarrollo full-stack, aplicando autenticación segura, manejo de roles y consumo de APIs con un diseño web funcional y limpio. Además, busca fomentar una experiencia digital más confiable y participativa en el ámbito gastronómico.

- **Funcionalidades requeridas**
  - Gestión de usuarios: Registro, inicio de sesión, asignación de roles y autenticación mediante JWT.
  - Gestión de restaurantes y platos: CRUD de restaurantes y platos y validaciones pertinentes para evitar duplicados.
  - Gestión de reseñas y ratings: CRUD de reseñas, asignación de like/dislike en reseñas ajenas y cálculo de calificación promedio para platos y restaurantes.
  - Categorías: CRUD de categorías.
  - Filtros y listados: Implementación de filtro de búsqueda y parámetros de ordenamiento según popularidad y ranking.

- **Requisitos técnicos**
  - Desarrollado con Node.js y Express, usando MongoDB con operaciones transaccionales.
  - Uso obligatorio de: dotenv, express, express-rate-limit, express-validator, mongodb, semver, swagger-ui-express, passport-jwt, jsonwebtoken, y bcrypt.
  - Arquitectura modular (rutas, controladores, modelos, configuración) y escalable.
  - Manejo centralizado de errores y respuestas con códigos HTTP correctos.
  - Validaciones en rutas con express-validator.
  - Variables de entorno en .env para credenciales y configuración.
  - CORS configurado para permitir conexión desde el frontend.
  - README con: descripción del proyecto, instalación, variables de entorno, ejemplos de endpoints y enlace al repositorio del frontend.



3. REQUERIMIENTOS

3.1. Requerimientos Funcionales

RF01

Gestión de usuarios

- Se debe permitir la creación de usuarios con correo electrónico y contraseña pertinentes.

Puntos clave del negocio

1. HASH de la contraseña para asegurar correctamente la contraseña en la base de datos.
2. Rol de usuario por defecto.

RF02

LOGIN | usuario

- El sistema de Login debe permitir el ingreso del usuario ya registrado anteriormente para brindar el acceso a la aplicación.

Puntos clave del negocio

1. Para realizar el LOGIN a la aplicación y acceder se debe haber hecho un registro por parte del usuario.
2. La contraseña debe coincidir con la guardada en la base de datos (contraseña parseada).

RF03

LOGIN | Administrador

- El sistema debe permitir el ingreso de administradores por rol de administrador para gestionar restaurantes, platos y categorías.

Puntos clave del negocio

1. Ingreso válido y verificación del rol de administrador.
2. Contraseña encriptada en la base de datos.

RF04

CRUD de restaurantes

- El sistema debe permitir la creación de restaurantes nuevos incluyendo su información importante, la edición teniendo en cuenta los campos anteriores y la eliminación del mismo.

Puntos clave del negocio

1. Las acciones anteriores deben ser realizadas únicamente por el rol de administrador.
2. Evitar restaurantes duplicados con información idéntica.

RF05

CRUD de platos

- El aplicativo web debe permitir la creación de platos con su información necesaria y enlace con el restaurante correspondiente, edición de los platos teniendo en cuenta los campos a modificar y la eliminación de los mismos.

Puntos clave del negocio

1. Funcionalidades que únicamente realiza el administrador.
2. Evitar duplicidad de platos.

RF06

CRUD de categorías

- Mediante el rol de administrador el aplicativo web debe permitir crear categorías tanto para platillos como restaurantes para dividir los tipos de comidas y ambientes de restaurantes, edición de las categorías y la eliminación de la misma.

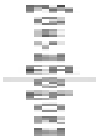
Puntos clave del negocio

1. Acciones solo permitidas por el rol de administrador.
2. Las categorías aplican tanto platos como restaurantes tomando en cuenta su diferencia.

RF07

CRUD de reseñas

- El aplicativo debe permitir la creación de reseñas asociadas a un restaurante con su nombre de usuario o de pila, calificación de 1.0 a 5.0 y una descripción de la



calificación realizada. Edición de la reseña si el usuario lo desea y la eliminación de la misma.

Puntos clave del negocio

1. Manejo mínimo de 1.0 a máximo 5.0 de calificación.
2. Cada reseña debe tener el nombre de la persona que la realizó.

RF08

Visualizar restaurantes

- El aplicativo web debe permitir el renderizado y vista independiente de los restaurantes disponibles con imagen e información correspondiente.

Puntos clave del negocio

1. Proporción y diseño correcto para el renderizado de los restaurantes.
2. Información completa del restaurante.
3. Vista independiente de restaurantes.

RF09

Visualizar platos

- Al dar click en los restaurantes disponibles a parte de la información del restaurante deben renderizarse los platos correspondientes con imagen, nombre, precio, y ambientados en el restaurante que se eligió.

Puntos clave del negocio

1. Vista completa de los platos disponibles brindados por el restaurante.

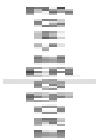
RF10

Funcionalidad por categorías

- Al elegir una categoría en el contexto de los restaurantes deben aparecer los restaurantes que poseen aquella categoría, en el contexto de los platos es la misma funcionalidad teniendo en cuenta la diferencia de los platos.

Puntos clave del negocio

1. Diferenciar categorías de restaurantes y categorías de platos.
2. Vista completa de las categorías tanto de platos como restaurantes.



RF11

Habilitar las opciones de “Like” y “dislike” para reseñas

- La sección de reseñas debe permitir a los usuarios dar “Like” y “Dislike” a otras reseñas que no sean propias.

Puntos clave del negocio

1. Manejar únicamente dos opciones: “Like” y “dislike”.
2. Los usuarios no pueden dar “Like” y “dislike” a reseñas propias.

RF12

Implementación de barras de búsqueda para restaurantes y platos

- Añadir barras de búsqueda para realizar búsquedas previas y concretas tanto de platos como de restaurantes.

Puntos clave del negocio

1. Las barras de búsqueda deben ser sensibles a las mayúsculas.
2. Añadir búsquedas previas y absolutas.

RF13

Cálculo de rating

- El aplicativo web debe calcular y mantener actualizado un Ranking Ponderado para cada restaurante.

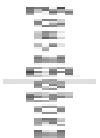
Puntos clave del negocio

1. Se debe ponderar la calificación promedio de todas las calificaciones que ha recibido el restaurante.
2. El ranking debe ser recalculado tras cada nueva reseña.

RF14

Impedir errores relacionados a valores no válidos en el aplicativo web

- El aplicativo debe evitar errores tras ingresar valores que superan los establecidos (ej. precio: -100) y notificar el error que el usuario está cometiendo



Puntos clave del negocio

3. Las barras de búsqueda deben ser sensibles a las mayúsculas.
4. Añadir búsquedas previas y absolutas.

RF15

Implementar la creación de administradores mediante una clave segura del sistema

- El aplicativo debe poder crear usuarios con rol de administrador por medio de una clave que verifique y valide la creación del administrador.

Puntos clave del negocio

1. No vulnerar las acciones del administrador

3.2. Requerimientos No Funcionales

RNF01

Escalabilidad hacia proximas mejoras

- La arquitectura de la base de datos y la aplicación debe estar diseñada para soportar un crecimiento si se quieren añadir más funcionalidades.

Puntos clave del negocio

1. Permitir que el aplicativo crezca sin revertir la arquitectura utilizada.

RNF02

Diseños y estilos profesionales e intuitivos

- Paleta de colores y estilos que estén ambientados profesionalmente hacia la gastronomía y a la vez el profesionalismo de la plataforma.

Puntos clave del negocio

1. Utilizar colores cálidos que no afecten el aspecto visual del usuario.
2. Fuentes adaptadas al diseño.

RNF03

Usabilidad

- La interfaz de usuario debe ser responsive (adaptable a diferentes tamaños de pantalla)

Puntos clave del negocio

1. Adaptación a diferentes tamaños de móviles para diferentes usuarios.

RNF04

Implementación de versionado

- Aplicar versionado al aplicativo web en caso de nuevas actualizaciones y optimizaciones además de un sistema que controle versiones anteriores en caso de que los usuarios la posean.

Puntos clave del negocio

1. Alertar y notificar al usuario que usa una versión antigua.
2. Implementación de versiones para próximas mejoras.

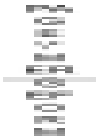
RNF05

Documentación de la API

- El servicio backend debe contar con documentación técnica actualizada que describa todos los endpoints, parámetros y respuestas para facilitar la integración frontend/backend.

Puntos clave del negocio

1. Documentación técnica de la API.



4. HISTORIAS DE USUARIO CON CRITERIOS DE ACEPTACIÓN

HISTORIA DE USUARIO			
Prioridad: Alta			
CÓDIGO DEL REQUERIMIENTO:	RF01	**Actor**	Usuario
NOMBRE DEL REQUERIMIENTO	**Gestión de usuarios**		
Descripción			
Como usuario, puedo registrarme en el sistema creando una cuenta con mis datos, con el fin de acceder a las funcionalidades del aplicativo.			
Funcionalidad			
El sistema debe permitir la creación de usuarios mediante un formulario de registro donde se capture correo y contraseña, así como datos personales adicionales y relevantes. La contraseña se almacenará de forma segura.			
Criterios de aceptación	1. El sistema debe validar que el correo no esté previamente registrado. 2. El sistema debe encriptar la contraseña antes de almacenarla. 3. El sistema debe solicitar el rol de usuario		
Restricciones			
Solo se aceptan correos válidos y únicos.			

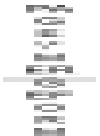
HISTORIA DE USUARIO			
Prioridad: Alta			
CÓDIGO DEL REQUERIMIENTO:	RF02	**Actor**	Usuario
NOMBRE DEL REQUERIMIENTO	**Login de usuario**		
Descripción			
Como usuario registrado, puedo iniciar sesión en el sistema con mis credenciales para acceder a la aplicación y sus funcionalidades.			
Funcionalidad			
El sistema debe permitir el ingreso de usuarios previamente registrados validando el correo y la contraseña. La contraseña ingresada debe compararse con la versión encriptada almacenada en la base de datos.			
Criterios de aceptación	1. El sistema debe verificar que el usuario esté registrado. 2. El sistema debe validar que la contraseña coincida con la registrada (encriptada). 3. El sistema debe mostrar un mensaje de error si las credenciales son incorrectas.		
Restricciones			
Solo los usuarios registrados pueden iniciar sesión. Las contraseñas deben estar encriptadas.			

HISTORIA DE USUARIO			
Prioridad: Alta			
CÓDIGO DEL REQUERIMIENTO:	RF03	**Actor**	Administrador
NOMBRE DEL REQUERIMIENTO	**Login de administrador**		
Descripción			
Como administrador, puedo iniciar sesión en el sistema utilizando mis credenciales y rol correspondiente para gestionar restaurantes, platos y categorías.			
Funcionalidad			
El sistema debe permitir el ingreso de usuarios con rol de administrador verificando las credenciales y su rol. Las contraseñas deben estar encriptadas y validadas correctamente antes de conceder acceso a las funciones administrativas.			
Criterios de aceptación	1. El sistema debe validar que el usuario posea el rol de administrador. 2. El sistema debe permitir acceso solo con credenciales correctas. 3. El sistema debe rechazar intentos de acceso de usuarios sin rol de administrador.		
Restricciones			
Solo los administradores registrados pueden acceder a las funciones de gestión. Las contraseñas deben estar encriptadas.			

HISTORIA DE USUARIO			
Prioridad: Alta			
CÓDIGO DEL REQUERIMIENTO:	RF04	**Actor**	Administrador
NOMBRE DEL REQUERIMIENTO	**CRUD de restaurantes**		
Descripción			
Como administrador, puedo crear, editar y eliminar restaurantes en el sistema con su respectiva información para mantener actualizada la base de datos del aplicativo.			
Funcionalidad			
El sistema debe permitir al administrador registrar nuevos restaurantes, editar su información (nombre, ubicación, descripción, categoría, etc.) y eliminarlos. Debe evitar duplicados con la misma información.			
Criterios de aceptación	1. El sistema debe validar que los restaurantes no estén duplicados. 2. El administrador debe poder modificar cualquier campo de un restaurante existente. 3. El administrador debe poder eliminar un restaurante del sistema.		
Restricciones			
Solo los administradores pueden realizar estas acciones. No se deben permitir restaurantes duplicados.			

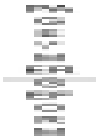
HISTORIA DE USUARIO			
Prioridad: Alta			
CÓDIGO DEL REQUERIMIENTO:	RF05	**Actor**	Administrador
NOMBRE DEL REQUERIMIENTO	**CRUD de platos**		
Descripción			
Como administrador, puedo crear, editar y eliminar platos asociados a los restaurantes con su información correspondiente (nombre, descripción, precio, imagen).			
Funcionalidad			
El sistema debe permitir gestionar los platos vinculados a un restaurante. Se podrán crear nuevos platos, modificar los existentes y eliminarlos. Debe evitar duplicidades en los nombres de los platos.			
Criterios de aceptación	1. El sistema debe permitir al administrador crear platos asociados a un restaurante. 2. El sistema debe evitar duplicidad de nombres de platos. 3. El sistema debe permitir editar o eliminar platos existentes.		
Restricciones			
Solo los administradores pueden gestionar los platos. No se permiten platos duplicados.			

HISTORIA DE USUARIO			
Prioridad: Alta			
CÓDIGO DEL REQUERIMIENTO:	RF06	**Actor**	Administrador
NOMBRE DEL REQUERIMIENTO	**CRUD de categorías**		
Descripción			
Como administrador, puedo crear, editar y eliminar categorías para clasificar tanto restaurantes como platos, diferenciando sus tipos.			
Funcionalidad			
El sistema debe permitir al administrador gestionar las categorías del aplicativo, asignándolas a platos o restaurantes según corresponda.			
Criterios de aceptación	1. El sistema debe permitir crear nuevas categorías. 2. El sistema debe permitir editar y eliminar categorías existentes. 3. Las categorías deben diferenciar entre platos y restaurantes.		
Restricciones			
Solo el administrador puede crear, editar o eliminar categorías.			



HISTORIA DE USUARIO			
Prioridad: Media			
CÓDIGO DEL REQUERIMIENTO:	RF07	**Actor**	Usuario
NOMBRE DEL REQUERIMIENTO	**CRUD de reseñas**		
Descripción			
Como usuario, puedo crear, editar y eliminar reseñas sobre los restaurantes, incluyendo una calificación de 1.0 a 5.0 y una descripción.			
Funcionalidad			
El sistema permitirá a los usuarios escribir reseñas asociadas a un restaurante, editar sus propias reseñas y eliminarlas cuando lo deseen. Cada reseña debe incluir nombre del usuario, descripción y calificación numérica..			
Criterios de aceptación	1. El sistema debe permitir calificaciones entre 1.0 y 5.0. 2. Cada reseña debe incluir el nombre del usuario. 3. Solo el autor puede editar o eliminar su reseña.		
Restricciones			
No se permiten calificaciones fuera del rango 1.0–5.0..			

HISTORIA DE USUARIO			
Prioridad: Media			
CÓDIGO DEL REQUERIMIENTO:	RF08	**Actor**	Usuario
NOMBRE DEL REQUERIMIENTO	**Visualizar restaurantes**		
Descripción			
Como usuario, puedo visualizar los restaurantes disponibles con su respectiva información e imagen representativa.			
Funcionalidad			
El sistema debe renderizar una lista de restaurantes mostrando su nombre, imagen, descripción, categoría e información detallada.			
Criterios de aceptación	1. El sistema debe mostrar todos los restaurantes registrados. 2. Cada restaurante debe incluir su imagen e información completa. 3. Debe existir una vista independiente con detalles del restaurante seleccionado.		
Restricciones			
La información mostrada debe ser coherente con los datos almacenados en la base de datos.			



HISTORIA DE USUARIO			
Prioridad: Media			
CÓDIGO DEL REQUERIMIENTO:	RF09	**Actor**	Usuario
NOMBRE DEL REQUERIMIENTO	**Visualizar platos**		
Descripción			
Como usuario, puedo visualizar los platos disponibles de un restaurante con su información correspondiente.			
Funcionalidad			
Al seleccionar un restaurante, el sistema debe mostrar todos los platos asociados, incluyendo nombre, imagen, precio y descripción.			
Criterios de aceptación	1. Los platos deben estar vinculados al restaurante seleccionado. 2. El sistema debe renderizar nombre, imagen, descripción y precio de cada plato. 3. Debe mostrarse una vista clara y ordenada de los platos.		
Restricciones			
Solo se mostrarán los platos asociados a restaurantes existentes.			

HISTORIA DE USUARIO			
Prioridad: Media			
CÓDIGO DEL REQUERIMIENTO:	RF10	**Actor**	Usuario
NOMBRE DEL REQUERIMIENTO	**Funcionalidad por categorías**		
Descripción			
Como usuario, puedo filtrar restaurantes o platos según su categoría para facilitar la búsqueda.			
Funcionalidad			
Al seleccionar una categoría, el sistema mostrará únicamente los restaurantes o platos asociados a ella. Las categorías deben diferenciarse entre restaurantes y platos.			
Criterios de aceptación	1. El sistema debe permitir filtrar restaurantes por categoría. 2. El sistema debe permitir filtrar platos por categoría. 3. Las categorías de restaurantes y platos deben manejarse de forma independiente.		
Restricciones			
Las categorías deben estar previamente registradas en el sistema.			

HISTORIA DE USUARIO			
Prioridad: Media			
CÓDIGO DEL REQUERIMIENTO:	RF11	**Actor**	Usuario
NOMBRE DEL REQUERIMIENTO	**Like y Dislike en reseñas**		
Descripción			
Como usuario, puedo dar “Like” o “Dislike” a reseñas de otros usuarios para expresar mi opinión sobre ellas.			
Funcionalidad			
El sistema debe permitir a los usuarios reaccionar con “Like” o “Dislike” a reseñas ajenas, registrando la cantidad total de cada reacción.			
Criterios de aceptación	1. Un usuario no puede reaccionar a su propia reseña. 2. Solo puede dar una reacción (Like o Dislike) por reseña. 3. El sistema debe mostrar el conteo actualizado de reacciones.		
Restricciones			
No se permite dar ambas reacciones simultáneamente.			

HISTORIA DE USUARIO			
Prioridad: Alta			
CÓDIGO DEL REQUERIMIENTO:	RF12	**Actor**	Usuario
NOMBRE DEL REQUERIMIENTO	**Barras de búsqueda para restaurantes y platos**		
Descripción			
Como usuario, puedo buscar restaurantes o platos específicos utilizando una barra de búsqueda sensible a mayúsculas y minúsculas.			
Funcionalidad			
El sistema debe implementar barras de búsqueda para encontrar rápidamente restaurantes o platos por nombre.			
Criterios de aceptación	1. La búsqueda debe ser sensible a mayúsculas y minúsculas. 2. Debe dinamizar las actualizaciones de los dato singresados. 3. Debe mostrar resultados concretos y completos.		
Restricciones			
No se deben mostrar resultados inexistentes o duplicados.			

HISTORIA DE USUARIO			
Prioridad: Media			
CÓDIGO DEL REQUERIMIENTO:	RF13	**Actor**	Sistema
NOMBRE DEL REQUERIMIENTO	**Cálculo de rating**		
Descripción			
Como sistema, debo calcular y mantener actualizado el ranking ponderado de cada restaurante en función de las calificaciones recibidas.			
Funcionalidad			
El sistema debe calcular el promedio ponderado de las calificaciones de cada restaurante y actualizarlo cada vez que se agregue una nueva reseña..			
Criterios de aceptación	1. El sistema debe recalcular el promedio tras cada reseña. 2. El valor mostrado debe reflejar las calificaciones actuales. 3. El ranking debe actualizarse en tiempo real o tras recargar la vista.		
Restricciones			
No deben permitirse cálculos con valores fuera del rango de calificación.			

HISTORIA DE USUARIO			
Prioridad: Alta			
CÓDIGO DEL REQUERIMIENTO:	RF14	**Actor**	Sistema
NOMBRE DEL REQUERIMIENTO	**Impedir errores por valores no válidos**		
Descripción			
Como sistema, debo validar los datos ingresados para evitar errores causados por valores fuera de los límites permitidos (ej. precios negativos).			
Funcionalidad			
El sistema debe verificar los valores ingresados por los usuarios y mostrar mensajes de error cuando los datos no cumplan las validaciones establecidas.			
Criterios de aceptación	1. El sistema debe impedir registros con valores fuera del rango permitido. 2. Debe mostrar un mensaje claro de error. 3. Debe evitar interrupciones o fallos en la aplicación.		
Restricciones			
No se permiten valores negativos ni campos vacíos obligatorios.			

HISTORIA DE USUARIO			
Prioridad: Alta			
CÓDIGO DEL REQUERIMIENTO:	RF15	**Actor**	Sistema/Administrador
NOMBRE DEL REQUERIMIENTO	**Creación de administradores mediante clave segura**		
Descripción			
Como sistema, debo permitir la creación de cuentas de administrador únicamente mediante una clave de verificación segura para mantener la integridad del sistema.			
Funcionalidad			
El sistema debe validar una clave segura antes de crear una cuenta con rol de administrador, garantizando que no se vulneren los privilegios administrativos.			
Criterios de aceptación	1. El sistema debe requerir una clave especial para crear administradores. 2. El sistema debe asignar automáticamente el rol de “administrador” tras la validación. 3. El sistema debe rechazar intentos de creación sin clave válida.		
Restricciones			
No se permite crear administradores sin clave segura verificada.			

HISTORIA DE USUARIO			
Prioridad: Media			
CÓDIGO DEL REQUERIMIENTO:	RNF01	**Actor**	Sistema
NOMBRE DEL REQUERIMIENTO	**Escalabilidad hacia próximas mejoras**		
Descripción			
El sistema debe estar diseñado para soportar la incorporación de nuevas funcionalidades sin comprometer su rendimiento ni su arquitectura actual.			
Funcionalidad			
La arquitectura de la base de datos y del backend debe ser modular, facilitando el crecimiento y la integración de nuevos módulos o servicios.			
Criterios de aceptación	1. La base de datos debe admitir expansión sin refactorizaciones mayores. 2. El backend debe poder integrar nuevas rutas y servicios. 3. El rendimiento no debe verse afectado significativamente.		
Restricciones			

HISTORIA DE USUARIO			
Prioridad: Media			
CÓDIGO DEL REQUERIMIENTO:	RNF02	**Actor**	Usuario
NOMBRE DEL REQUERIMIENTO	**Diseños y estilos profesionales e intuitivos**		
Descripción			
La interfaz del sistema debe contar con un diseño profesional, atractivo e intuitivo, adecuado al contexto gastronómico.			
Funcionalidad			
El sistema debe implementar una paleta de colores cálidos, tipografía legible y estilos coherentes con la temática gastronómica.			
Criterios de aceptación	1. La interfaz debe ser visualmente coherente. 2. Los colores deben mantener una buena legibilidad. 3. Los estilos deben ser consistentes en toda la aplicación.		
Restricciones			

HISTORIA DE USUARIO			
Prioridad: Alta			
CÓDIGO DEL REQUERIMIENTO:	RNF03	**Actor**	Usuario
NOMBRE DEL REQUERIMIENTO	**Usabilidad**		
Descripción			
La interfaz de usuario debe ser adaptable a diferentes dispositivos y tamaños de pantalla.			
Funcionalidad			
El diseño debe ser responsive, asegurando una visualización óptima tanto en computadoras como en dispositivos móviles.			
Criterios de aceptación	1. El sistema debe adaptarse correctamente a distintos tamaños de pantalla. 2. Los componentes deben conservar su funcionalidad en móviles. 3. No debe producirse distorsión visual ni pérdida de información.		
Restricciones			

HISTORIA DE USUARIO			
Prioridad: Media			
CÓDIGO DEL REQUERIMIENTO:	RNF04	**Actor**	Sistema
NOMBRE DEL REQUERIMIENTO	**Implementación de versionado**		
Descripción			
El sistema debe incluir un control de versiones que permita identificar actualizaciones y mantener versiones anteriores para compatibilidad.			
Funcionalidad			
El aplicativo debe mostrar al usuario si está utilizando una versión antigua y ofrecer la posibilidad de actualizar.			
Criterios de aceptación	1. El sistema debe identificar la versión actual del software. 2. Debe notificar al usuario en caso de versión desactualizada. 3. Debe mantener registro histórico de versiones.		
Restricciones			

HISTORIA DE USUARIO			
Prioridad: Alta			
CÓDIGO DEL REQUERIMIENTO:	RNF05	**Actor**	Desarrollador
NOMBRE DEL REQUERIMIENTO	**Documentación de la API**		
Descripción			
El backend debe contar con documentación técnica actualizada que describa sus endpoints, parámetros y respuestas.			
Funcionalidad			
La API debe estar documentada mediante herramientas como Swagger o Postman, asegurando la comprensión e integración con el frontend.			
Criterios de aceptación	1. La documentación debe estar actualizada. 2. Debe incluir ejemplos de solicitudes y respuestas. 3. Debe ser accesible para todos los desarrolladores del proyecto.		
Restricciones			
Solo se aceptan correos válidos y únicos.			

5. METODOLOGÍA

Para el desarrollo de este proyecto, se implementó la metodología ágil Kanban. Este método se basa en la visualización del flujo de trabajo, el control de tareas en curso (WIP) y la mejora continua de los procesos. Su aplicación permitió al equipo mantener flexibilidad,

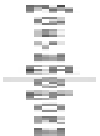
adaptarse a cambios en las prioridades y asegurar entregas continuas de valor, sin depender de ciclos fijos como en Scrum.

La organización del proyecto se realizó mediante un tablero Kanban, estructurado en columnas que reflejaban los estados de cada tarea: Por hacer, En progreso, En validación y Finalizado. Cada historia de usuario y requisito se representó con una tarjeta, lo que facilitó la transparencia del avance, la detección de cuellos de botella y la toma de decisiones efectivas.

- Fases principales del proyecto
    - Recolección de información
    - Levantamiento de requerimientos
    - Planeación y diseño
    - Desarrollo de funcionalidades
    - Validación y mejora
    - Verificación final y documentación
- Tecnología y librerías utilizadas
    - Node.js para la lógica de negocio y manejo de peticiones.
    - MongoDB para la persistencia de datos con colecciones e índices.
    - Express.js para estructurar el backend.
    - JavaScript como lenguaje principal.
    - Git/GitHub para control de versiones y trabajo colaborativo.
    - dotenv para la configuración segura de credenciales.
    - cors para habilitar la conexión segura entre el backend y el frontend.
    - cookie-parser para interpretar y gestionar las cookies enviadas por el cliente, facilitando la autenticación y el manejo de sesiones.
    - express-validator para validar los datos recibidos en las rutas del servidor.
    - express-rate-limit para limitar la cantidad de solicitudes por usuario.
    - bcrypt para el cifrado seguro de contraseñas.
    - passport-jwt para la autenticación mediante el uso de JSON Web Tokens (JWT) para validar usuarios y proteger rutas privadas.

El equipo Scrum se constituyó de la siguiente manera:

- **Producto Owner** (Davisson Adriel Roman): Encargado de definir la visión del producto, priorizar funcionalidades y asegurar que el producto final cumpla con las necesidades del usuario.
- **Scrum Master** (Joan Sebastian Serrano): Responsable de garantizar la correcta aplicación de Scrum y eliminar obstáculos en el proceso
- **Developers** (Davisson Adriel y Sebastian Serrano): Encargado de construir el producto técnico a través de las tareas asignadas por Sprint.



6. EVIDENCIA DE PLANTEAMIENTO DE PLATAFORMA DE TRABAJO

Acá se debe documentar toda la evidencia de trabajo colaborativo con los siguientes elementos:

- Link del repositorio Backend: https://github.com/Sebas404040/FOODIERANK
- Link del repositorio Frontend: https://github.com/Davisson-Adriel/FRONT_FOOD
- Link de videos de reuniones: https://drive.google.com/drive/folders/1OV5cuRo293_W1R4XjBPh9j6BTwkmZlPp?usp=drive_link
- Link Video Final: https://drive.google.com/drive/folders/1mkRHrYstlsGbaLUllh2QQ8lvcXqbuvoH?usp=drive_link
- Link del tablero clickup backend: https://sharing.clickup.com/90132667950/b/h/6-901321762897-2/cf7625048836b0b

- Link del tablero clickup frontend: https://sharing.clickup.com/90132667950/b/h/6-901321762997-2/9bf7d1823077621

- BackLog: https://sharing.clickup.com/90132667950/l/h/2ky56qhe-253/17b7f6785d673e6

-

7. CONCLUSIONES

- La aplicación logró integrar herramientas y librerías ampliamente utilizadas en la industria, consolidando una solución full-stack robusta y segura, adaptable a distintos escenarios reales.
- El trabajo bajo SCRUM nos permitió distribuir eficientemente responsabilidades y mantener una comunicación fluida, favoreciendo la entrega continua de avances.
- Se aplicaron mecanismos de autenticación y validaciones estrictas, garantizando la protección de los datos y la privacidad de los usuarios, lo cual es esencial en cualquier sistema moderno.
- La arquitectura modular adoptada asegura que el sistema pueda crecer y mantenerse a largo plazo, facilitando la incorporación de nuevas funcionalidades sin comprometer la organización del código.
- El uso de herramientas como Swagger y un README detallado contribuyó a una documentación clara, que facilita la integración de nuevos desarrolladores y la comprensión para usuarios finales.