Especificación de Requisitos de Software

Proyecto: My Foodie - Aplicación de Ranking de Restaurantes

Revisión: 1.0

Ficha del documento

Fecha	Revisión	Autor	Verificado
2 de nov. de 25	1.0	SCRUM Master Product Owner Equipo de Desarrollo	Sergio Liévano Bryan Villabona

1. Introducción

1.1. Propósito

El propósito de este documento es definir y especificar de manera completa y no ambigua todos los requisitos funcionales y no funcionales para el desarrollo del software "My Foodie". Este documento servirá como la base contractual y técnica para el diseño, desarrollo, pruebas y validación del sistema, asegurando que el producto final cumpla con los objetivos del taller y las necesidades de los usuarios.

La audiencia a la que va dirigido este documento incluye:

- El **Equipo de Desarrollo** (Brayan Villabona, Sergio Lievano).
- El **Cliente/Sponsor** del proyecto (Docentes y personal de Campuslands).

1.2. Alcance

El producto a desarrollar, "My Foodie", es una aplicación web full-stack diseñada para permitir a los usuarios registrar, calificar y rankear restaurantes y sus platos. La aplicación guiará al usuario a través de un ecosistema gastronómico digital, permitiendo funcionalidades clave:

1. **Gestión de Usuarios y Roles:** Registro, inicio de sesión y diferenciación entre "Clientes" y "Administradores".
2. **Gestión de Contenido (Admin):** CRUD completo de Categorías y Platos.
3. **Flujo de Aprobación de Contenido:** Los Clientes "proponen" restaurantes, y los Administradores los "aprueban" o "rechazan".

4. **Sistema de Reseñas y Votación:** Los Clientes pueden crear, editar y eliminar sus propias reseñas, así como votar (Like/Dislike) en las reseñas de otros.
5. **Sistema de Ranking Ponderado:** Un algoritmo que calcula la calificación de un restaurante basándose en las reseñas, los votos y la antigüedad de las mismas.

El sistema se enfocará en cumplir estos objetivos. No incluirá funcionalidades transaccionales de un negocio real, como procesamiento de pagos, reservas en tiempo real o gestión de inventario de restaurantes.

1.3. Personal involucrado

Rol	Categoría profesional	Responsabilidades
Product Owner	**Estudiante / Sergio Lievano**	**Encargado de definir la visión del producto, gestionar el Product Backlog (Historias de Usuario) y validar el desarrollo del Frontend[cite: 21].**
Scrum Master	**Estudiante / Brayan Villabona y Sergio Lievano**	**Coordinar al equipo, facilitar la metodología SCRUM/KANBAN, gestionar el tablero y asegurar el cumplimiento de hitos[cite: 21].**
Desarrollador (Backend)	**Estudiante / Brayan Villabona**	**Responsable de construir la API REST, la lógica de negocio, la seguridad, la conexión a la base de datos y las transacciones[cite: 21].**
Desarrollador (Frontend)	**Estudiante / Sergio Lievano**	**Responsable de construir la aplicación cliente (UI/UX) en JavaScript Puro, consumir la API y gestionar el estado del frontend[cite: 21].**

1.4. Definiciones, acrónimos y abreviaturas

- **SRS:** Software Requirements Specification (Especificación de Requisitos de Software).

- **UI:** User Interface (Interfaz de Usuario).
- **UX:** User Experience (Experiencia de Usuario).
- **API:** Application Programming Interface (Interfaz de Programación de Aplicaciones).
- **REST:** Representational State Transfer (Estilo de arquitectura para APIs).
- **JWT:** JSON Web Token (Estándar para la creación de tokens de acceso).
- **CRUD:** Create, Read, Update, Delete (Operaciones básicas de datos).
- **DTO:** Data Transfer Object (Objetos que definen la estructura de datos para la validación de la API).
- **BD / DB:** Base de Datos.
- **DOM:** Document Object Model.

1.5. Referencias

- **IEEE Std 830-1998** - Recommended Practice for Software Requirements Specifications.
- **Guía del Taller "My Foodie"** - Documento proporcionado por Campuslands que define el alcance, las funcionalidades requeridas y las restricciones tecnológicas del proyecto.
- **Documentación oficial de Node.js, Express, MongoDB Driver y Passport-JWT.**
- **Estándar de JavaScript ES6+ (ECMAScript 2015).**

1.6. Resumen

Este documento está organizado en tres secciones principales, siguiendo el estándar IEEE 830:

- **Sección 1:** Introduce el proyecto "My Foodie", su propósito, alcance para el equipo y el docente, y las definiciones clave.
- **Sección 2:** Ofrece una descripción general del producto, sus funciones principales, las características de los dos tipos de usuarios (Cliente y Admin), y las restricciones tecnológicas y de arquitectura.
- **Sección 3:** Detalla los requisitos específicos. Define las interfaces de software (API REST), los 8 requisitos funcionales principales (agrupando las 40 historias de usuario) y los requisitos no funcionales (como seguridad, rendimiento e integridad de datos).

2. Descripción general

2.1. Perspectiva del producto

"My Foodie" es un sistema de software nuevo y autocontenido. Se desarrolla como una aplicación web full-stack, diseñada bajo una arquitectura desacoplada:

1. **Un Backend (API REST):** Construido con Node.js y Express, actúa como el servidor central. Es responsable de toda la lógica de negocio, la seguridad, la interacción con la base de datos y la ejecución de operaciones transaccionales.
2. **Un Frontend (Cliente Web):** Construido con HTML, CSS y JavaScript Puro (Vanilla JS), actúa como el cliente. Es una interfaz de usuario (UI) responsiva que consume la API REST del backend para mostrar datos e interactuar con el usuario.

El frontend no tiene lógica de negocio; solo renderiza la interfaz y delega todas las operaciones al backend. El backend gestiona la autenticación mediante tokens JWT.

2.2. Funcionalidad del producto

El sistema "My Foodie" proporcionará un conjunto de funcionalidades agrupadas en 8 módulos principales (que engloban las 40 historias de usuario):

1. **Módulo de Autenticación y Seguridad:**
    - Permite el registro de nuevos usuarios (HU-01) y el inicio de sesión (HU-02).
    - Gestiona el cierre de sesión (HU-03).
    - Protege las vistas del frontend basándose en si el usuario está autenticado (HU-04) o si es Administrador (HU-05).
    - Protege la API contra abuso (HU-33) y valida todos los datos de entrada (HU-37).
2. **Módulo de Gestión de Categorías (Admin):**
    - Permite al Administrador visualizar la lista de categorías (HU-06).
    - Permite al Administrador crear (HU-07), actualizar (HU-08) y eliminar (HU-09) categorías.
3. **Módulo de Flujo de Aprobación:**
    - Permite al Cliente proponer un nuevo restaurante (HU-10), que se guarda con estado "pendiente".
    - Permite al Administrador ver la lista de restaurantes "pendientes" (HU-11).
    - Permite al Administrador "aprobar" (HU-12) o "rechazar" (HU-13) dichas propuestas.

4. **Módulo de Gestión de Restaurantes (Admin):**
    - Permite al Administrador actualizar la información de restaurantes ya aprobados (HU-14).
    - Permite al Administrador eliminar un restaurante (y todos sus datos asociados, como platos y reseñas) de forma transaccional (HU-15).
5. **Módulo de Exploración y Descubrimiento (Cliente):**
    - Permite al Cliente ver la lista de todos los restaurantes aprobados (HU-16).
    - Permite filtrar esta lista por categoría (HU-17).
    - Permite ordenar esta lista por ranking, popularidad o fecha (HU-18).
    - Permite buscar restaurantes por nombre (HU-19).
    - Permite navegar a la página de detalle de un restaurante (HU-20).
6. **Módulo de Gestión de Platos (Admin):**
    - Permite a cualquier usuario visualizar los platos de un restaurante (HU-21).
    - Permite al Administrador añadir (HU-22), editar (HU-23) y eliminar (HU-24) platos del menú de un restaurante.
7. **Módulo de Reseñas y Ranking (Cliente):**
    - Permite al Cliente publicar una nueva reseña (calificación y comentario), limitado a una por restaurante (HU-25).
    - Permite a todos los usuarios ver la lista de reseñas en el detalle del restaurante (HU-26).
    - Permite al Cliente editar (HU-27) o eliminar (HU-28) *solo* sus propias reseñas.
    - Permite al Cliente votar con "Like" (HU-29) o "Dislike" (HU-30) en reseñas ajenas.
    - Activa el recálculo automático del ranking ponderado (HU-36) con cada una de estas acciones.
8. **Módulo de Perfil de Usuario y UX:**
    - Proporciona un Dashboard al Cliente con sus estadísticas (HU-31).
    - Proporciona una vista de "Mis Reseñas" (HU-32).

- Garantiza una experiencia responsive (HU-34) y un sistema de notificaciones claro (HU-35).
- Incluye utilidades de desarrollo como un Seeder (HU-40) y documentación de API (HU-38).

2.3. Características de los usuarios

El sistema "My Foodie" ha sido diseñado para dos (2) tipos de usuarios con roles y permisos claramente diferenciados:

Categoría	Usuario Cliente	Usuario Administrador
Definición	**Es el usuario final y el actor principal de la plataforma. Es el rol que se asigna por defecto a cualquier nuevo registro.**	**Es un usuario de confianza, miembro del equipo de "My Foodie", responsable de la calidad y gestión del contenido.**
Formación	**No se requiere formación técnica. Familiarizado con aplicaciones web y redes sociales.**	**Usuario con conocimientos del sistema y de la lógica de negocio.**
Funciones Clave	**- Proponer restaurantes (HU-10).** **- Explorar, filtrar, ordenar y buscar restaurantes (HU-16, 17, 18, 19).** **- Ver detalles de restaurantes (HU-20) y sus platos (HU-21).** **- Publicar (HU-25), editar (HU-27) y eliminar (HU-28) sus propias reseñas.** **- Votar (Like/Dislike) en reseñas ajenas (HU-29, HU-30).**	**- Todas las funciones del Cliente.** **- Aprobar (HU-12) y Rechazar (HU-13) restaurantes.** **- CRUD completo de Categorías (HU-07, HU-08, HU-09).** **- CRUD completo de Platos (HU-22, HU-23, HU-24).** **- Editar (HU-14) y Eliminar (HU-15) cualquier restaurante.**

Categoría	Usuario Cliente	Usuario Administrador
	- Ver su Dashboard y "Mis Reseñas" (HU-31, HU-32).	
Acceso	**Accede al frontend (cliente web). No tiene acceso al panel de administración.**	**Accede tanto al frontend como a las vistas especiales del panel de administración (HU-05).**

2.4. Restricciones

El desarrollo del sistema está sujeto a un conjunto de restricciones técnicas y de arquitectura inamovibles, definidas por los requisitos del taller:

- **R-1 (Stack Backend):** El servidor (API) debe desarrollarse obligatoriamente con **Node.js** y **Express**.
- **R-2 (Stack Frontend):** La interfaz de usuario debe desarrollarse obligatoriamente con **HTML**, **CSS** y **JavaScript Puro (Vanilla JS)**. El uso de frameworks o librerías de frontend (como React, Vue, Angular, JQuery) está estrictamente prohibido.
- **R-3 (Base de Datos):** La base de datos debe ser **MongoDB**. Se debe utilizar el **driver oficial de MongoDB (mongodb)**.
- **R-4 (Seguridad):** La autenticación debe implementarse usando **JWT** (JSON Web Tokens), y las contraseñas deben ser hasheadas con **Bcrypt**.
- **R-5 (Validación):** Todas las entradas de la API deben ser validadas en el backend usando **express-validator**.
- **R-6 (Documentación):** La API debe ser documentada interactivamente usando **swagger-ui-express**.
- **R-7 (Integridad de Datos):** El sistema debe implementar **Transacciones de MongoDB** para todas las operaciones críticas que involucren múltiples escrituras (ej. crear reseña y recalcular ranking).
- **R-8 (Arquitectura):** El Backend y el Frontend deben estar alojados en **repositorios de GitHub separados**.

2.5. Suposiciones y dependencias

- **S-1:** Se asume que los usuarios (Clientes y Administradores) accederán a la aplicación a través de un navegador web moderno (Google Chrome, Firefox, Safari, Edge) con JavaScript habilitado.
- **S-2:** Se asume que los usuarios disponen de una conexión a internet estable para la comunicación entre el frontend y el backend.
- **S-3:** El sistema depende de un servicio de alojamiento para la base de datos MongoDB (ej. MongoDB Atlas) o una instancia local accesible, cuya URI de conexión debe estar definida en la variable de entorno MONGO_URI.
- **S-4:** El frontend depende de que el backend esté en ejecución y accesible en la URL definida en su configuración (ej. http://localhost:4000/api/v1).

3. Requisitos específicos

3.1. Requisitos de interfaces externas

Esta sección detalla las interfaces del sistema con el mundo exterior, incluyendo el usuario (UI), el hardware, el software (API) y las comunicaciones.

3.1.1. Interfaces de usuario

La interfaz de usuario es una aplicación web responsiva (Cliente Web) construida en HTML, CSS y JavaScript Puro. Las vistas principales que se presentan al usuario son:

1. **Vista de Autenticación (index.html):**
    - Proporciona dos pestañas: "Iniciar Sesión" (HU-02) y "Registrarse" (HU-01).
    - Contiene los formularios para la entrada de credenciales (nombre, email, password).
    - Muestra notificaciones modales (HU-35) en caso de error (ej. "Credenciales inválidas").
2. **Vista de Dashboard (dashboard.html):**
    - Es la página de inicio para usuarios autenticados.
    - Muestra el Layout principal (Navegación + Cabecera).
    - Renderiza las tarjetas de estadísticas del usuario (Total Reseñas, Promedio, Favorito) (HU-31).
    - Muestra una cuadrícula con los restaurantes recomendados (mejor ranking).
3. **Vista de Explorar (restaurantes.html):**
    - Renderiza la cuadrícula principal de tarjetas de restaurantes (HU-16).

- Contiene los controles de interacción: Barra de Búsqueda (HU-19), Filtros de Categoría (HU-17) y Selector de Ordenamiento (HU-18).
- Implementa la paginación si los resultados exceden el límite por página.

4. **Vista de Detalle (detalle.html):**
    - Muestra la información completa de un solo restaurante (HU-20).
    - Presenta la lista de platos de ese restaurante (HU-21).
    - Muestra la lista de reseñas (HU-26).
    - Contiene el formulario para "Dejar tu opinión" (HU-25).
    - Renderiza dinámicamente los botones de administrador (CRUD de Platos, CRUD de Restaurante) si el usuario es "admin" (HU-14, HU-22, HU-23, HU-24).
5. **Vista de Perfil ("Mis Reseñas") (mis-resenas.html):**
    - Muestra un historial de todas las reseñas publicadas por el usuario autenticado (HU-32).
6. **Vistas de Administración (admin/):**
    - **Gestión de Categorías:** Interfaz de tabla que permite el CRUD de categorías (HU-06, HU-07, HU-08, HU-09).
    - **Aprobaciones:** Interfaz de tarjetas que permite visualizar (HU-11), aprobar (HU-12) y rechazar (HU-13) los restaurantes pendientes.

3.1.2. Interfaces de hardware

El sistema "My Foodie" es una aplicación web y no interactúa directamente con hardware especializado. Sus requisitos de hardware son los de un navegador web moderno:

- **Dispositivos de Entrada:** Teclado y ratón (para escritorio) o una interfaz táctil (para dispositivos móviles y tabletas), compatible con el diseño responsivo (HU-34).
- **Dispositivos de Salida:** Un monitor de computadora o pantalla de dispositivo móvil.

3.1.3. Interfaces de software (API)

El Frontend se comunica con el Backend a través de una API RESTful. El Backend expone los siguientes endpoints (agrupados por recurso), todos bajo el prefijo /api/v1:

- **Autenticación (/auth):**
    - POST /register: Registra un nuevo usuario (HU-01).

- POST /login: Autentica un usuario y devuelve un JWT (HU-02).

- **Usuarios (/usuarios):**
  - GET /mis-stats: (Protegido) Obtiene las estadísticas del dashboard (HU-31).
  - GET /mis-resenas: (Protegido) Obtiene el historial de reseñas del usuario (HU-32).
- **Categorías (/categorias):**
  - GET /: Obtiene la lista pública de todas las categorías (para HU-17).
  - POST /: (Admin) Crea una nueva categoría (HU-07).
  - GET /:id: Obtiene una categoría por ID.
  - PATCH /:id: (Admin) Actualiza una categoría (HU-08).
  - DELETE /:id: (Admin) Elimina una categoría (HU-09).
- **Restaurantes (/restaurantes):**
  - GET /: Obtiene la lista de restaurantes aprobados. Acepta queries ?categoria= (HU-17) y ?sort= (HU-18).
  - POST /: (Protegido) Un cliente propone un nuevo restaurante (HU-10).
  - GET /admin/pendientes: (Admin) Obtiene la lista de restaurantes pendientes (HU-11).
  - GET /:id: Obtiene los detalles de un restaurante (HU-20).
  - PATCH /:id: (Admin) Actualiza un restaurante (HU-14).
  - DELETE /:id: (Admin) Elimina un restaurante (y sus datos) (HU-15).
  - PATCH /:id/aprobar: (Admin) Aprueba un restaurante (HU-12).
- **Platos (Anidados en Restaurantes y /platos):**
  - GET /restaurantes/:id/platos: Obtiene los platos de un restaurante (HU-21).
  - POST /restaurantes/:id/platos: (Admin) Crea un nuevo plato (HU-22).
  - PATCH /platos/:id: (Admin) Actualiza un plato (HU-23).
  - DELETE /platos/:id: (Admin) Elimina un plato (HU-24).
- **Reseñas (Anidadas en Restaurantes y /resenas):**
  - GET /restaurantes/:id/resenas: Obtiene las reseñas de un restaurante (HU-26).

- POST /restaurantes/:id/resenas: (Protegido) Un cliente publica una reseña (HU-25).
- PATCH /resenas/:id: (Protegido) Un cliente edita su propia reseña (HU-27).
- DELETE /resenas/:id: (Protegido) Un cliente elimina su propia reseña (HU-28).
- POST /resenas/:id/like: (Protegido) Vota "Like" (HU-29).
- POST /resenas/:id/dislike: (Protegido) Vota "Dislike" (HU-30).

3.1.4. Interfaces de comunicación

- **Protocolo:** La comunicación entre el Frontend (Cliente) y el Backend (Servidor) se realiza exclusivamente a través del protocolo **HTTPS** (HTTP en desarrollo).
- **Autenticación:** Las solicitudes a endpoints protegidos deben incluir un **Token JWT** en la cabecera (header) de la petición HTTP, utilizando el esquema Authorization: Bearer <token>.
- **CORS:** El Backend está configurado para permitir solicitudes de origen cruzado (CORS) provenientes de la URL específica del Frontend (definida en process.env.FRONTEND_URL).

3.2. Requisitos funcionales

Los requisitos funcionales del sistema "My Foodie" se agrupan en los siguientes módulos, derivados de las 40 historias de usuario (HUs) definidas durante la fase de planificación.

3.2.1. Módulo 1: Autenticación y Seguridad

ID	Nombre del Requisito	HUs Asociadas
RF-01	Gestión de Autenticación	HU-01, HU-02, HU-03
RF-02	Gestión de Autorización	HU-04, HU-05

3.2.2. Módulo 2: Gestión de Contenido (Admin)

ID	Nombre del Requisito	HUs Asociadas
RF-03	CRUD de Categorías	HU-06, HU-07, HU-08, HU-09
RF-04	CRUD de Platos	HU-21, HU-22, HU-23, HU-24

3.2.3. Módulo 3: Flujo de Aprobación de Restaurantes

ID	Nombre del Requisito	HUs Asociadas
RF-05	Propuesta de Clientes	HU-10
RF-06	Gestión de Aprobaciones (Admin)	HU-11, HU-12, HU-13
RF-07	Mantenimiento de Restaurantes (Admin)	HU-14, HU-15

3.2.4. Módulo 4: Exploración y Descubrimiento

ID	Nombre del Requisito	HUs Asociadas
RF-08	Visualización y Navegación	HU-16, HU-20
RF-09	Filtrado y Búsqueda	HU-17, HU-19
RF-10	Ordenamiento	HU-18

3.2.5. Módulo 5: Sistema de Reseñas y Votación

ID	Nombre del Requisito	HUs Asociadas
RF-11	Gestión de Reseñas (Cliente)	HU-25, HU-27, HU-28
RF-12	Visualización de Reseñas	HU-26
RF-13	Sistema de Votación (Likes)	HU-29, HU-30

3.2.6. Módulo 6: Sistema de Ranking

ID	Nombre del Requisito	HUs Asociadas
RF-14	Cálculo del Ranking Ponderado	HU-36

3.2.7. Módulo 7: Perfil de Usuario y UX

ID	Nombre del Requisito	HUs Asociadas
RF-15	Panel de Usuario (Cliente)	HU-31, HU-32
RF-16	Interfaz de Usuario	HU-34, HU-35, HU-39

3.2.8. Módulo 8: Requisitos de Desarrollo

ID	Nombre del Requisito	HUs Asociadas
RF-17	Calidad y Pruebas (Backend)	HU-33, HU-37, HU-40
RF-18	Documentación	HU-38

3.3. Requisitos no funcionales

ID	Requerimiento No Funcional	Descripción (Resumen)
RNF-01	**Rendimiento**	El backend debe limitar las peticiones por IP (HU-33) para evitar sobrecargas. La carga de vistas del frontend debe ser eficiente.
RNF-02	**Seguridad**	Las contraseñas deben almacenarse hasheadas (Bcrypt). El acceso a la API debe estar protegido por JWT (HU-03) y la autorización debe diferenciar "cliente" de "admin" (HU-05).
RNF-03	**Integridad de Datos**	El sistema **debe** usar Transacciones de MongoDB (HU-36) en todas las operaciones que modifiquen el ranking (HU-25, 27, 28, 29, 30) y en la eliminación de restaurantes (HU-15).
RNF-04	**Validación**	Todos los datos de entrada a la API deben ser validados en el servidor (HU-37) usando express-validator.
RNF-05	**Usabilidad**	La interfaz debe ser intuitiva, responsiva y adaptable a dispositivos móviles (HU-34). Debe proveer retroalimentación clara al usuario (HU-35).
RNF-06	**Mantenibilidad**	El código debe estar desacoplado (Backend y Frontend en repositorios separados). El backend debe ser modular (Rutas, Controladores, Servicios). El frontend debe ser modular (Pages, Components, Services).
RNF-07	**Documentación**	La API del backend debe estar documentada y ser explorable vía Swagger (HU-38).

ID	Requerimiento No Funcional	Descripción (Resumen)
RNF-08	**Portabilidad**	La aplicación debe funcionar en cualquier navegador web moderno (Chrome, Firefox, Safari, Edge).

3.4 Requisitos de base de datos

El sistema utiliza una base de datos NoSQL (MongoDB). Aunque NoSQL es no relacional, el sistema mantiene la integridad referencial a nivel de aplicación mediante el uso de ObjectId.

Las colecciones principales son:

1. **usuarios**: Almacena los datos de registro, incluyendo el rol.
2. **categorias**: Lista simple de categorías gestionada por el admin.
3. **restaurantes**: La entidad central.
    - Se conecta 1:N con categorias (un restaurante tiene una categoría).
    - Se conecta 1:N con usuarios (un usuario creadoPor el restaurante).
4. **platos**:
    - Se conecta 1:N con restaurantes (un plato pertenece a un solo restaurante).
5. **reseñas**: La entidad más conectada.
    - Se conecta 1:N con restaurantes (una reseña es para un solo restaurante).
    - Se conecta 1:N con usuarios (una reseña tiene un solo autor).
    - Implementa la relación N:M para los votos, almacenando los _id de los usuarios que han dado like o dislike en arrays.