Universidad Latina de Costa Rica

Proyecto Final

Sistemas Operativos II

Preparado por

Fabián Abarca

Lizandro Hernández

Rodrigo Horvilleur

08 de Abril del 2025

v1.2.0

Tabla de Contenidos

Historial de Revisiones

Nombre	Fecha	Razón de los Cambios	Versión
Fabián Abarca	08/04/2025	Creación del documento inicial	0.1.0
Lizandro Hernández	08/04/2025	Actualización punto 1.3	0.1.1
Rodrigo Horvilleur	09/04/2025	Implementación de Diagrama de Clases	0.2.0
Lizandro Hernández	10/04/2025	Cambios en puntos 2.2, 2.3, 2.6	0.2.1
Rodrigo Horvilleur	10/04/2025	Implementación de Diagramas (Flujo y Secuencial)	0.3.0
Fabián Abarca	10/04/2025	Cambios en punto 3 y derivados	1.0.0
Rodrigo Horvilleur	11/04/2025	Revisión #1	1.1.0
Fabián Abarca	11/04/2025	Revisión #2	1.1.1
Lizandro Hernández	11/04/2025	Revisión Final	1.2.0

1. Introducción

1.1 Propósito del Documento

Este documento define los requerimientos funcionales y no funcionales del sistema **AI Storage**. Está destinado a desarrolladores, testers, gerentes de proyecto y partes interesadas que participan en su desarrollo, despliegue y mantenimiento.

1.2 Alcance del Producto

AI Storage es un backend que utiliza **Express.js** y **OpenAI GPT** para permitir a los usuarios crear, consultar y administrar eventos a través de mensajes en lenguaje natural. El sistema procesa estos mensajes, interpreta acciones, y maneja eventos en una base de datos **SQLite**.

El sistema está diseñado para ser un backend RESTful seguro, escalable y fácil de integrar con interfaces web y móviles.

1.3 Definiciones, Acrónimos y Abreviaturas

- **API**: Application Programming Interface
- **CRUD**: Create, Read, Update, Delete
- **ERS**: Especificación de Requerimientos de Software
- **GPT**: Generative Pre-trained Transformer
- **CORS**: Cross-Origin Resource Sharing

1.4 Referencias

- Express.js Documentation
- OpenAI API Documentation
- Zod Documentation

1.5 Resumen del Documento

Este documento organiza la información de manera estructurada: primero se define el producto y su contexto, luego se describen los requisitos, la calidad esperada, las interfaces, el diseño previsto y los métodos de verificación.

2. Resumen del Producto

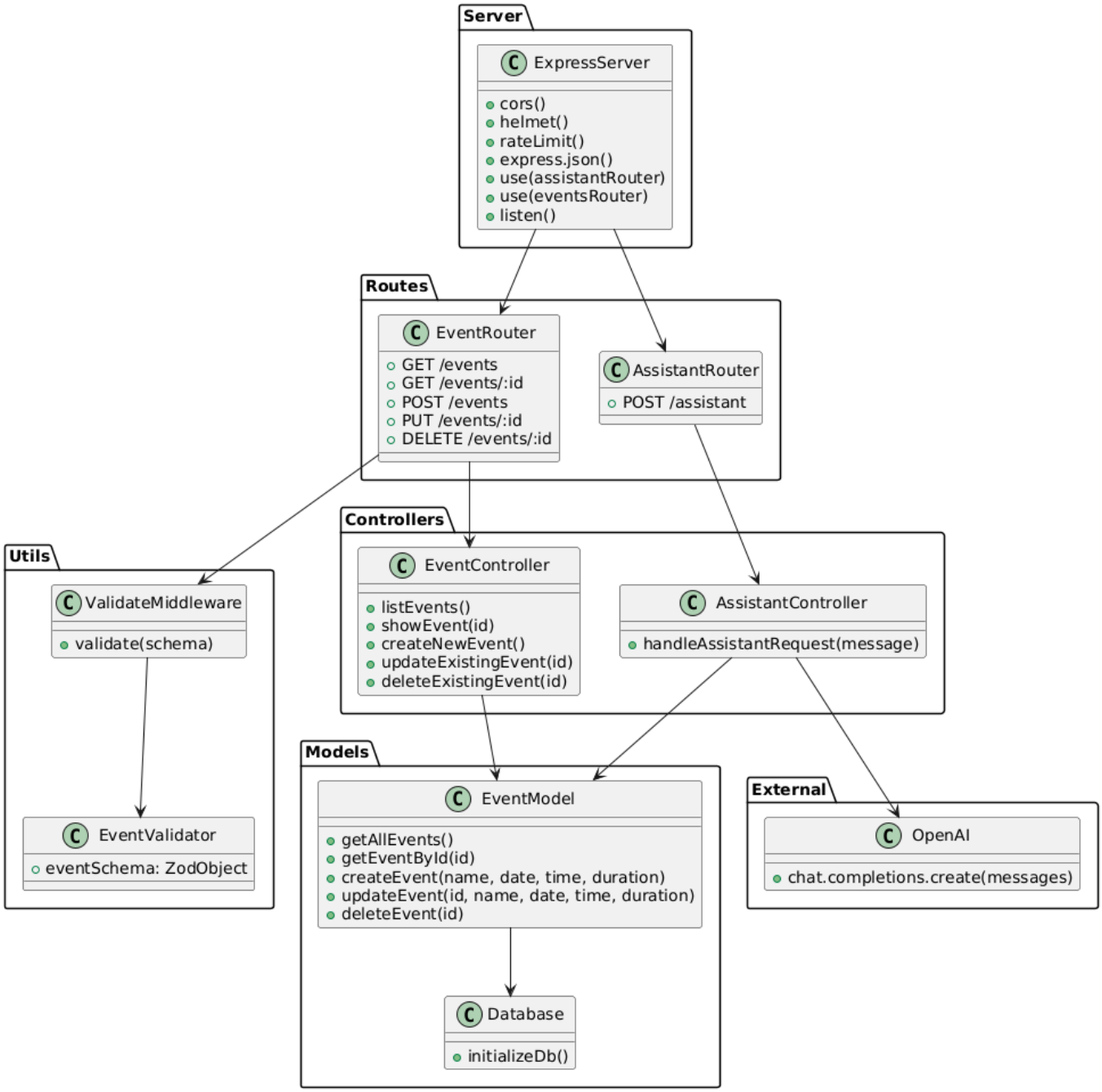
2.1 Perspectiva del Producto

AI Storage es un sistema **independiente** que actúa como un **backend inteligente**. No depende de versiones anteriores ni de otros sistemas heredados. Se conecta con **OpenAI GPT API** y proporciona endpoints para manipular eventos almacenados en **SQLite**.

2.2 Funciones del Producto

- Recibir solicitudes de creación, consulta y gestión de eventos mediante lenguaje natural.
- Interpretar instrucciones usando **OpenAI GPT**.
- Crear eventos con **nombre**, **fecha**, **hora** y **duración**.
- Consultar eventos por **fecha** o **ID**.

- Actualizar y eliminar eventos.
- Validar datos de entrada utilizando **Zod**.
- Proteger el servidor con **Helmet**, **CORS** y **Rate Limit**.



2.3 Restricciones del Producto

- Solo se permiten métodos HTTP seguros (`POST`, `GET`, `PUT`, `DELETE`).
- Limitación de 100 solicitudes por IP cada 15 minutos.
- Acceso restringido por **CORS** a orígenes permitidos.

2.4 Características del Usuario

- Usuarios no técnicos que interactúan mediante un frontend.
- Administradores que puedan consultar eventos por ID.

- Integradores de sistemas que consumen la API vía HTTP.

2.5 Suposiciones y Dependencias

- La clave de API de OpenAI debe estar activa y válida.
- Se espera un servidor Node.js operativo en producción.
- Conectividad estable a internet para llamadas a OpenAI.

2.6 Asignación de Requisitos

Requisito	Componente asignado
Procesamiento de lenguaje	Assistant Controller + OpenAI
CRUD de eventos	Event Controller + Event Model
Validaciones	Validate Middleware + Zod
Seguridad	Express middleware (Helmet, Rate-Limit, CORS)

3. Requisitos

3.1 Interfaces Externas

3.1.1 Interfaces de Usuario

- API RESTful (usada por Postman o Frontend Web).

3.1.2 Interfaces de Hardware

- No aplica (backend ejecutado en servidor estándar).

3.1.3 Interfaces de Software

- Express.js
- OpenAI API
- SQLite3
- Zod

3.2 Funcionales

- El sistema debe interpretar mensajes en lenguaje natural.
- El sistema debe crear, listar, actualizar y eliminar eventos.
- El sistema debe validar la entrada del usuario antes de realizar cualquier operación en la base de datos.

3.3 Calidad del Servicio

3.3.1 Rendimiento

- El sistema debe responder a las solicitudes en menos de 500 ms para operaciones CRUD locales.

3.3.2 Seguridad

- Uso de **Helmet** para cabeceras seguras.
- Aplicación de **CORS** controlado.
- Protección contra DDoS con **express-rate-limit**.

3.3.3 Confiabilidad

- Manejo de errores de OpenAI y base de datos de manera robusta.

3.3.4 Disponibilidad

- 99% de disponibilidad con despliegue en servidores adecuados (ej. PM2 + Nginx).

3.4 Cumplimiento

- Cumplir con estándares de APIs RESTful.
- Cumplir con políticas de protección de datos si se almacena información sensible.

3.5 Diseño e Implementación

3.5.1 Instalación

- Instalación mediante `npm install`.
- Configuración del archivo `.env`.

3.5.2 Distribución

- Código fuente administrado en repositorios Git.

3.5.3 Mantenibilidad

- Separación de responsabilidades (routers, controllers, models, utils).

3.5.4 Reusabilidad

- Los servicios y modelos son modulares.

3.5.5 Portabilidad

- Desplegable en cualquier servidor compatible con Node.js.

3.5.6 Costo

- Uso de OpenAI API implica costos por volumen de tokens consultados.

3.5.7 Plazo

- Desarrollo estimado en 2 a 3 semanas para MVP.

3.5.8 Prueba de Concepto

- Implementar pruebas unitarias y de integración para controladores y modelos.

- Verificación mediante Postman.