****

**UNIVERSIDAD PRIVADA DE TACNA**

**FACULTAD DE INGENIERIA**

**Escuela Profesional de Ingeniería de Sistemas**

**Proyecto Deadline Collision**

Curso: ***Calidad y Pruebas de software***

Docente: *Patrick Cuadros Quiroga*

Integrantes:

***Vargas Gutierrez, Angel Jose 2020066922***

***Angel Chino***

**Tacna – Perú**

***2025***

| CONTROL DE VERSIONES | | | | | |
| --- | --- | --- | --- | --- | --- |
| Versión | Hecha por | Revisada por | Aprobada por | Fecha | Motivo |
| 1.0 | MPV | ELV | ARV | 10/10/2020 | Versión Original |

Sistema *{Nombre del Sistema}*

Documento de Arquitectura de Software

Versión *{1.0}*

| CONTROL DE VERSIONES | | | | | |
| --- | --- | --- | --- | --- | --- |
| Versión | Hecha por | Revisada por | Aprobada por | Fecha | Motivo |
| 1.0 | AVG | AC | AVG | 10/06/2025 | Versión Original |

INDICE GENERAL

**Contenido**

[1. INTRODUCCIÓN 4](#_z8vtuaub0j9j)

[1.1. Propósito (Diagrama 4+1) 4](#_zhph1byg7jb4)

[1.2. Alcance 4](#_x67zba5pw5wc)

[1.3. Definición, siglas y abreviaturas 4](#_c2a0lqqlon0x)

[1.4. Organización del documento 5](#_p52uj5uyj94y)

[2. OBJETIVOS Y RESTRICCIONES ARQUITECTÓNICAS 5](#_fkwgr0cxjbh4)

[2.1.1. Requerimientos Funcionales 5](#_z6thf8woofrp)

[2.1.2. Requerimientos No Funcionales – Atributos de Calidad 5](#_u7rpsh59607e)

[***1. REPRESENTACIÓN DE LA ARQUITECTURA DEL SISTEMA 5***](#_jw4r4w88w2h8)

[1.1.1. Diagramas de Casos de uso 6](#_a2kn4n4d5epu)

[1.2.1. Diagrama de Subsistemas (paquetes) 7](#_6nbwpzezp47)

[1.2.2. Diagrama de Secuencia (vista de diseño) 7](#_8cruv5javh3o)

[1.2.3. Diagrama de Colaboración (vista de diseño) 8](#_vz14f44gvuv3)

[1.2.4. Diagrama de Objetos 8](#_fplvv4fdqf5s)

[1.2.5. Diagrama de Clases 9](#_9c797wap4f49)

[1.2.6. Diagrama de Base de datos (relacional o no relacional) 9](#_3bhfzh9ufwsj)

[1.3.1. Diagrama de arquitectura software (paquetes) 10](#_ht04f5ocg2kg)

[1.3.2. Diagrama de arquitectura del sistema (Diagrama de componentes) 10](#_4xjir16z2zdi)

[1.4.1. Diagrama de Procesos del sistema (diagrama de actividad) 11](#_u23c71v99pi8)

[1.5.1. Diagrama de despliegue 12](#_jfa19g837p1w)

### 1. INTRODUCCIÓN

### 1.1. Propósito (Diagrama 4+1)

El propósito de este documento es describir la arquitectura del sistema "Deadline Collision Predictor", un calendario académico inteligente que integra IA para ayudar a los estudiantes a registrar y organizar sus actividades (proyectos, exámenes y tareas), evitando colisiones de fechas y mejorando la gestión del tiempo. El sistema se documenta bajo la perspectiva del modelo 4+1 de vistas arquitectónicas: lógica, procesos, desarrollo, física y casos de uso.

### 1.2. Alcance

El sistema permite a los usuarios:

* Registrar actividades académicas mediante texto libre, asistido por IA.
* Detectar automáticamente colisiones de eventos en el calendario.
* Visualizar y gestionar sus actividades desde una interfaz web moderna.
* Subir horarios en imagen o PDF y extraer automáticamente los eventos mediante OCR e IA.
* Consultar estadísticas de avance y distribución de actividades.

### 1.3. Definición, siglas y abreviaturas

* IA: Inteligencia Artificial
* OCR: Reconocimiento Óptico de Caracteres
* JWT: JSON Web Token
* API: Interfaz de Programación de Aplicaciones
* DB: Base de Datos

### 1.4. Organización del documento

El documento está organizado en secciones que cubren los objetivos, restricciones, vistas arquitectónicas (casos de uso, lógica, implementación, procesos, despliegue) y atributos de calidad del software.

### 2. OBJETIVOS Y RESTRICCIONES ARQUITECTÓNICAS

### 2.1.1. Requerimientos Funcionales

* Registro y autenticación de usuarios.
* Registro de eventos (tareas, exámenes, proyectos) mediante texto libre o formularios.
* Procesamiento de texto con IA para extraer fecha, hora, tipo y nombre de la actividad.
* Detección automática de colisiones de eventos.
* Visualización de calendario y estadísticas.
* Subida y procesamiento de horarios en imagen/PDF.
* Gestión CRUD de eventos.

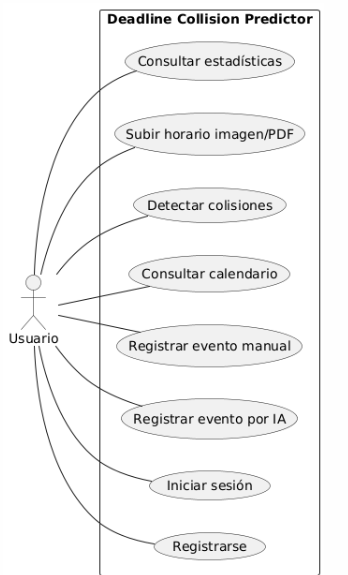
### 2.1.2. Requerimientos No Funcionales – Atributos de Calidad

* Seguridad: Autenticación JWT, contraseñas hasheadas.
* Usabilidad: Interfaz web intuitiva y responsiva.
* Rendimiento: Respuesta rápida en operaciones comunes.
* Mantenibilidad: Código modular y documentado.
* Escalabilidad: Arquitectura preparada para múltiples usuarios concurrentes.
* Portabilidad: Desplegable en cualquier servidor Node.js con MySQL/MariaDB.

# **REPRESENTACIÓN DE LA ARQUITECTURA DEL SISTEMA**

* 1. Vista de Caso de uso

### Diagramas de Casos de uso

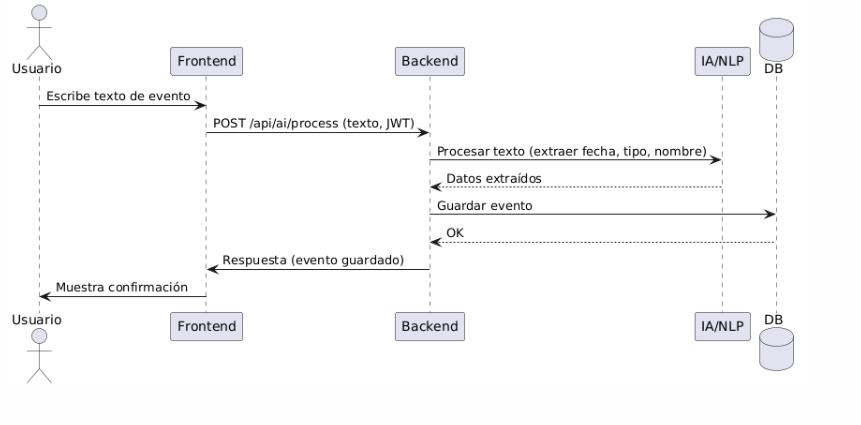
**

* 1. Vista Lógica

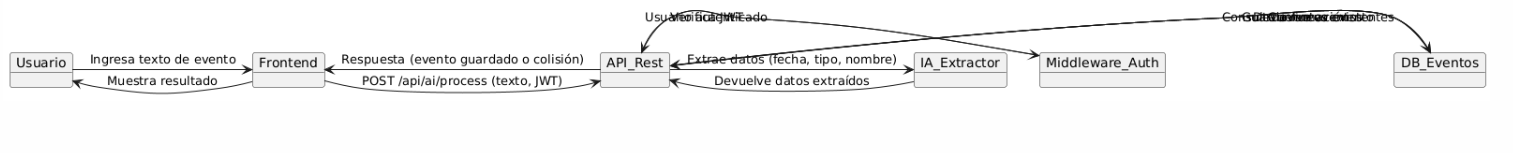
### 

### Diagrama de Subsistemas (paquetes)

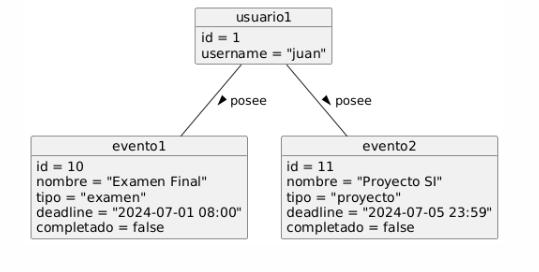
### Diagrama de Secuencia (vista de diseño)



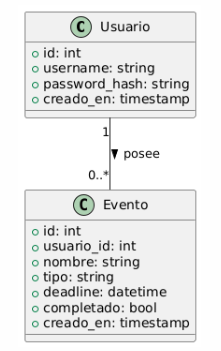
### Diagrama de Colaboración (vista de diseño)



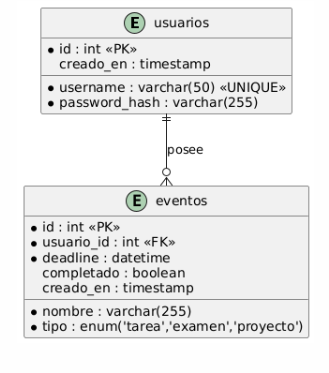
### Diagrama de Objetos



### Diagrama de Clases

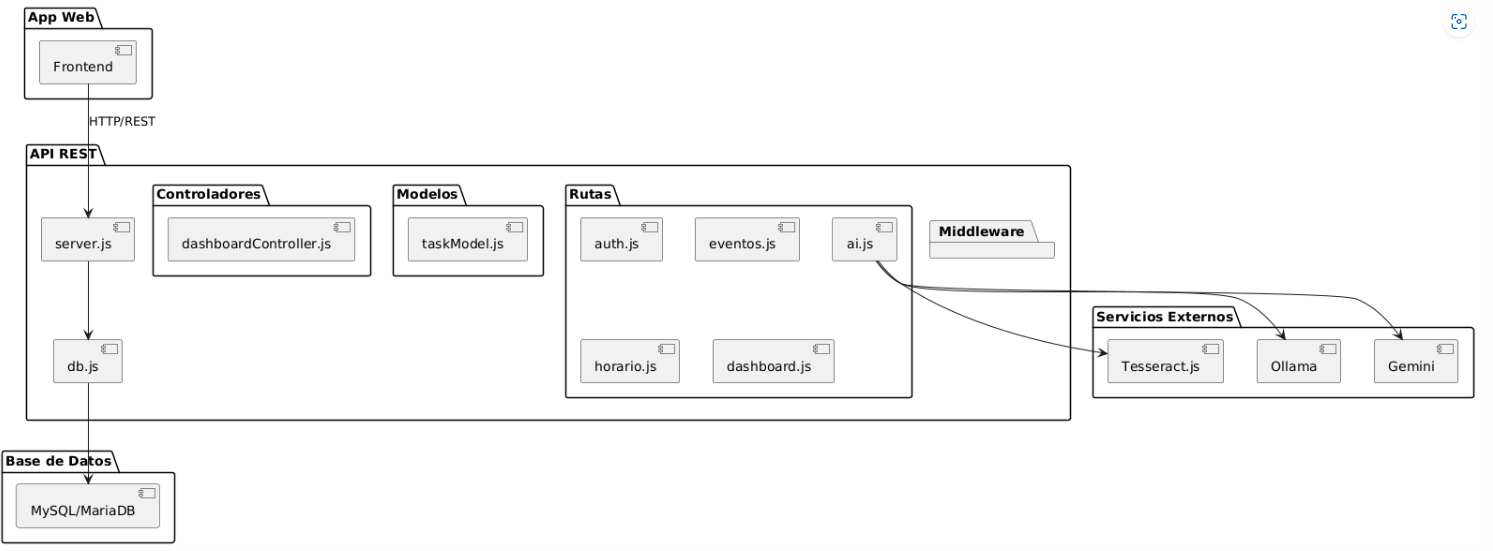


### Diagrama de Base de datos (relacional o no relacional)

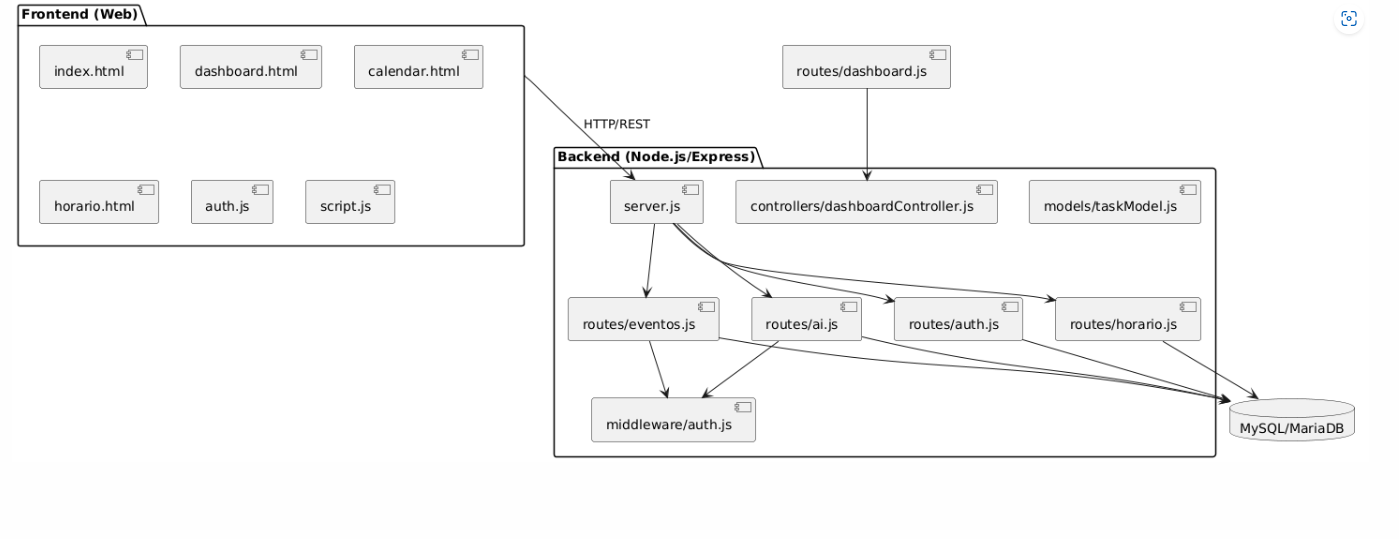


* 1. Vista de Implementación (vista de desarrollo)

### Diagrama de arquitectura software (paquetes)

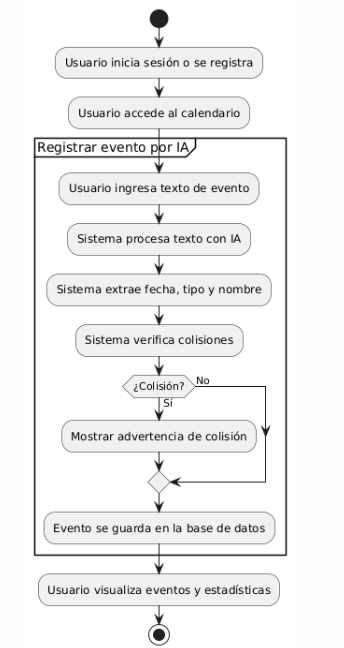


### Diagrama de arquitectura del sistema (Diagrama de componentes)

**

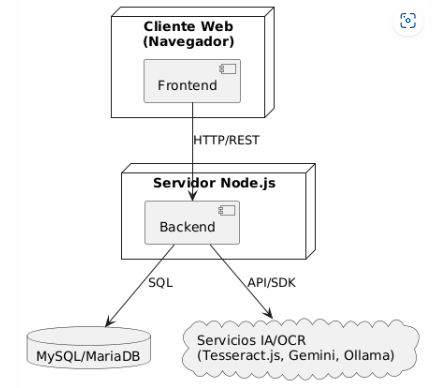
* 1. Vista de procesos

### Diagrama de Procesos del sistema (diagrama de actividad)



* 1. Vista de Despliegue (vista física)

### Diagrama de despliegue

**

4. ATRIBUTOS DE CALIDAD DEL SOFTWARE

Escenario de Funcionalidad

El sistema permite registrar, consultar y gestionar eventos académicos, con ayuda de IA para el procesamiento de lenguaje natural y OCR para horarios.

Escenario de Usabilidad

Interfaz web moderna, responsiva y fácil de usar, con formularios claros y feedback inmediato.

Escenario de confiabilidad

Validación de datos en backend y frontend.

Manejo de errores y mensajes claros al usuario.

Pruebas automatizadas (tests y e2e).

Escenario de rendimiento

Respuesta rápida en operaciones CRUD.

Procesamiento eficiente de imágenes y texto.

Escenario de mantenibilidad

Código modular, organizado por responsabilidades.

Uso de controladores, modelos y middlewares.

Documentación y comentarios en el código.

Otros Escenarios

Seguridad: JWT, contraseñas hasheadas, protección de rutas.

Escalabilidad: Preparado para múltiples usuarios.

Portabilidad: Desplegable en cualquier entorno [Node.js](http://node.js).