



Universidad Tecnológica  
del Norte de Guanajuato  
Organismo Público Descentralizado del Gobierno del Estado  
“Educación y progreso para la vida”

# **Sistema de Gestión Académica de Planeación y Evidencias**

## **ING. EN DESARROLLO Y GESTIÓN DE SOFTWARE**

### **Desarrollo Web Integral**

**Profesor: Gabriel Barron Rodrigues**

**Alumnos: Palomares Barrientos  
Schoenstantt Andrea  
Mata Hernández José Manuel**

**Octubre 2025**

## Índice

Introducción .....	4
Metodología Ágil y Planeación del Proceso .....	5
Razones por las que Scrum fue seleccionado: .....	5
Revisión continua y mejora incremental: .....	5
Adaptabilidad frente a cambios: .....	6
Alternativas evaluadas: .....	6
Evidencias de Daily Stand-Up .....	7
Historias de Usuario .....	10
Backlog de Actividades .....	14
Planificación del Proyecto .....	15
Configuración, arquitectura y autenticación .....	15
Planeaciones, evidencias y reportes .....	15
Notificaciones, despliegue y pruebas .....	16
Arquitectura del Software .....	17
Arquitectura Multicapa (N-Tier) basada en el patrón MVC (Modelo–Vista–Controlador) .....	17
Diagrama de Arquitectura .....	17
Estructura general del diagrama .....	18
<b>Capas de la arquitectura</b> .....	18
<b>Capa de Lógica de Negocio (Backend / Controlador y Servicio)</b> .....	18
<b>Capa de Persistencia (Repositorio / DAO / Modelo)</b> .....	18
<b>Capa de Base de Datos (Data Layer)</b> .....	19
<b>Flujo de datos</b> .....	19
<b>Entorno de ejecución</b> .....	19
Frameworks y entorno de desarrollo .....	22
Frontend: .....	22
Justificación técnica: .....	22
Entre sus ventajas destacan: .....	22
Responsabilidades del Frontend: .....	22
Backend: .....	22

Ventajas técnicas: .....	22
Responsabilidades del Backend:.....	22
Control de Versiones y Planeación del Proyecto en GitHub .....	23
Configuraciones clave: .....	23
Control de Versiones y Planeación del Proyecto en GitHub .....	23
Plataforma de Control de Versiones .....	23
Sus principales ventajas son: .....	23
Estrategia de Ramas .....	24
Políticas de Acceso y Merge .....	25
<b>Automatización con GitHub Workflows</b> .....	25
Workflows .....	26
fullstack-ci.yml.....	26
test.yml.....	27
Evidencia Técnica de Integración y Pruebas.....	28
Seguridad e Integración de Componentes .....	32
Servicios REST Propios (5):.....	33
Servicios REST implementados (back-end propio).....	33
<b>Seguridad aplicada (OWASP base)</b> .....	34
Despliegue y Escaneo de Seguridad .....	35
Docker en Back (Pruebas) .....	35
Docker file en Front .....	36
Docker Compose.yml .....	36
Mongo Atlas .....	38
Docker Hub .....	39
Backend .....	39
Docker Hub FronEnd.....	40

## Introducción

El proyecto “Gestión de Planeación Académica” fue desarrollado adoptando la metodología ágil Scrum, debido a su capacidad para facilitar la entrega continua de valor, fomentar la colaboración entre los miembros del equipo y adaptarse a requerimientos cambiantes propios del entorno educativo. Scrum permitió organizar el trabajo en ciclos cortos e iterativos (sprints de 7 días), garantizando avances constantes y verificables sobre el sistema, los cuales fueron revisados junto con el Product Owner para asegurar el cumplimiento de objetivos académicos y funcionales.

Durante el desarrollo, se definieron roles específicos Scrum Master, Product Owner y Equipo de Desarrollo, se llevaron a cabo ceremonias ágiles como reuniones Daily Stand-Up, Sprint Planning, Sprint Review y Sprint Retrospective, y se priorizó el Product Backlog conforme al valor entregado al usuario final. El objetivo principal fue construir una plataforma funcional que permita a docentes y coordinadores registrar, gestionar y evaluar planeaciones didácticas, evidencias y reportes de desempeño académico, integrando tecnologías modernas como React, Node.js, MongoDB y Docker.

Gracias a este enfoque iterativo y colaborativo, el proyecto evolucionó progresivamente, permitió la corrección temprana de errores, favoreció la retroalimentación continua e impulsó la construcción de un sistema escalable, seguro y alineado a las necesidades institucionales.

## Metodología Ágil y Planeación del Proceso

Metodología seleccionada: Scrum

### Justificación:

El proyecto “Gestión de Planeación Académica” se desarrolló utilizando Scrum debido a su enfoque iterativo y colaborativo, que se adapta perfectamente a entornos educativos con requerimientos cambiantes.

Esta metodología permite entregar versiones funcionales del sistema de manera incremental, garantizando la calidad y la retroalimentación constante del usuario final (coordinadores y docentes).

- **Iteraciones cortas (Sprints):** Los sprints se definieron con una **duración de 7 días**, lo que permite mantener un ritmo constante de avance, revisión y mejora. Esta duración equilibra la necesidad de obtener entregables frecuentes con la recomendación de no tener períodos demasiado cortos, asegurando así una planificación más realista y productiva.
- **Roles definidos:**
  - **Scrum Master:** responsable de supervisar el proceso y eliminar impedimentos.
  - **Product Owner:** define prioridades, requisitos y acepta entregables.
  - **Equipo de Desarrollo:** diseña, programa, prueba y documenta las funcionalidades.
- **Reuniones periódicas:**
  - **Daily Stand-Up:** reunión breve diaria de seguimiento.
  - **Sprint Planning:** definición de objetivos del sprint.
  - **Sprint Review:** presentación de avances al finalizar el sprint.
  - **Sprint Retrospective:** análisis de mejoras para la siguiente iteración.
  - **Priorización del Backlog:** enfoque en las funcionalidades críticas (planeaciones, avances, evidencias y reportes).
  - **Flexibilidad ante cambios:** se permite la adaptación de requisitos durante el desarrollo sin afectar la estructura global del proyecto.

Razones por las que Scrum fue seleccionado:

- **Iteraciones breves y continuas (7 días):**
- Se buscó obtener entregables funcionales de manera semanal, permitiendo una retroalimentación constante del Product Owner (docente o coordinador).
- Esta frecuencia permite corregir errores y realizar ajustes antes de pasar al siguiente módulo o funcionalidad.
- El ritmo se mantuvo ágil gracias al tamaño reducido del equipo (2 desarrolladores) y la comunicación directa con el Product Owner.

Revisión continua y mejora incremental:

- Cada sprint concluye con un entregable mínimo viable (por ejemplo, un endpoint funcional, una vista React o una integración parcial con la base de datos).
- Esto asegura evidencia continua de avance, ideal para proyectos académicos de corta duración.

#### Adaptabilidad frente a cambios:

- En el entorno universitario, los requerimientos del profesor y las fechas de entrega pueden variar. Scrum facilita la replanificación al cierre de cada sprint sin comprometer la calidad ni el progreso global del proyecto.

#### Alternativas evaluadas:

Metodología	Ventajas	Desventajas	Razón de selección / no selección
Scrum	Iteraciones cortas, revisión continua, flexibilidad	Requiere coordinación constante	Seleccionada: permite entregables diarios y retroalimentación inmediata para equipo reducido (2 devs)
Kanban	Flujo continuo, flexible	No establece entregables por tiempo	No seleccionada: requiere entregables concretos por sprint
XP	Código de alta calidad, pruebas constantes	No define planificación de entregables	No seleccionada: estructura académica requiere seguimiento de HU y sprints
Waterfall	Planificación estricta	No flexible, Feedback tardío	No seleccionada: cambios frecuentes y entregas incrementales en curso académico

Scrum permite obtener entregables funcionales casi diarios (endpoints, vistas, integración parcial), recibir retroalimentación inmediata del PO (profesor), y replanificar al final de cada sprint sin comprometer el producto global.

## Evidencias de Daily Stand-Up

Las reuniones diarias se realizaron de manera asíncrona a través de mensajes de WhatsApp, debido a la disponibilidad y horarios del equipo.

Cada integrante enviaba su reporte diario respondiendo a las tres preguntas básicas de Scrum:

- ¿Qué hice ayer?
- ¿Qué haré hoy?
- ¿Hay algún impedimento o en qué necesito apoyo?

Este método permitió mantener comunicación constante, dar seguimiento a las tareas asignadas y detectar posibles bloqueos.

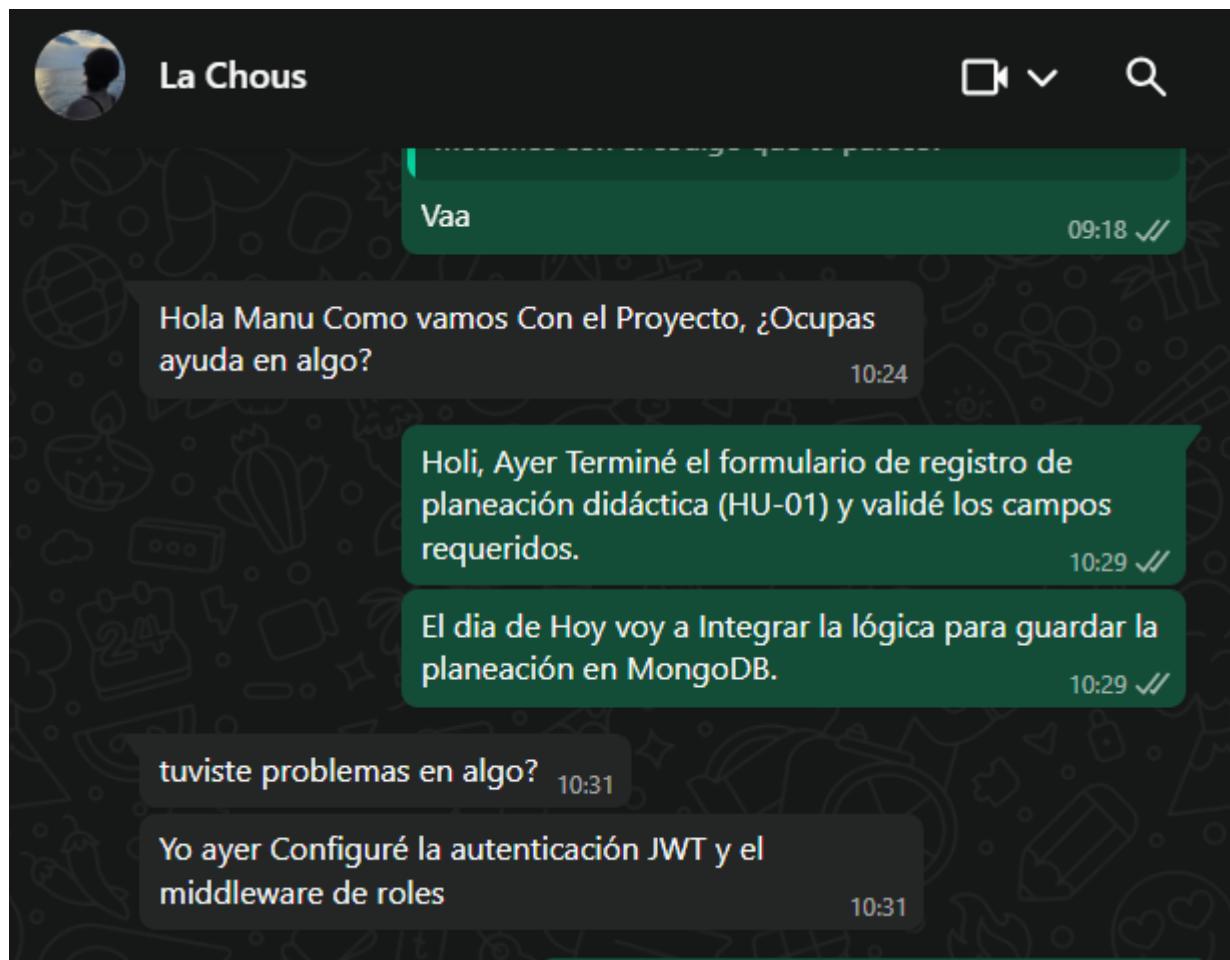
La evidencia de estas interacciones se encuentra en el chat de WhatsApp del equipo, donde se documentaron los avances diarios de cada integrante.

Evidencia (ejemplo de mensaje Daily Stand-Up):

Fecha: ()

Canal: WhatsApp

FOTO DE EVIDENCIA





La Chous



Como a las 2:30 11:45 ✓

Vale 11:45

Holi Holi, Estoy tebiendo algunos problemas con el  
Proyecto

12:22 ✓



12:22 ✓

hola Manu, Como que problemas 12:49

Yo el dia de ayer subi Subí evidencias de prueba a  
Cloudinary, y hoy voy a Validar la vista de  
confirmación al cargar archivo.

12:49

yo el dia de ayer Implementé envío de correos  
automáticos y el dia de hoy estoy probando el flujo  
completo de notificación de entrega.

12:58 ✓

☺ Espero acabarlo antes de ir a revision 12:58 ✓



La Chous

□ ✓ 🔎 ⋮

Holi holi 12:50 ✓✓

Necesitas ayuda en algo 12:50 ✓✓

Holi, pues el dia de ayer Ajusté los estilos del formulario de planeación y validé los límites de texto y hoy voy a conectar el formulario con el backend usando Axios y probar con usuario autenticado.

12:50

Gracias 12:50

Yo ayer Revisé la lógica de expiración del JWT y la renovación automática y eserpe el dia de hoy Implementar refresh token y middleware de autorización por rol.

12:58 ✓✓

Si necesitas ayuda me dices. 12:58

Puedes checar esto plis esque ese rato me marco un pequeño bug en el endpoint PUT /api/usuarios/:id

12:59 ✓✓

Si manu ahorita lo checo 12:59

## Historias de Usuario

### HU-01 — Registrar planeación didáctica

Como docente, quiero registrar mi planeación didáctica, para evidenciar mi avance académico y cumplir con la planeación institucional.

#### Criterios de aceptación

- El formulario debe incluir los campos: asignatura, periodo, objetivos, actividades, recursos y fecha.
- Todos los campos son obligatorios y deben validar longitud máxima de texto.
- Debe permitir guardar y editar planeaciones.
- Las planeaciones deben almacenarse correctamente en MongoDB bajo el usuario autenticado.

### HU-02 — Subir evidencia

Como docente, quiero subir evidencias (archivos PDF o imágenes), para respaldar mis planeaciones y demostrar cumplimiento.

#### Criterios de aceptación

- Peso máximo permitido: **10 MB**.
- Tipos de archivo aceptados: **.pdf, .jpg, .png**.
- Las evidencias deben almacenarse en un **servicio externo seguro** (Cloudinary o S3).
- Al subir, la API debe devolver **URL pública segura** con token o firma temporal.
- Cada evidencia debe asociarse correctamente a su planeación (planeacionId).

### **HU-03 — Ver reporte por parcial**

Como coordinador, quiero visualizar el porcentaje de cumplimiento de los docentes por parcial, para dar seguimiento académico y evaluar avances.

#### **Criterios de aceptación**

- Endpoint: GET /api/reportes/por-parcial?periodo=2025-1.
- Solo accesible a roles coordinador o administrador (autenticación JWT).
- El reporte debe mostrar porcentaje total de cumplimiento por docente.
- Permitir exportar resultados a PDF y Excel.
- Visualización en frontend con gráficos (por ejemplo, Chart.js).

### **HU-04 — Autenticación y control de acceso**

Como usuario (docente, coordinador o administrador), quiero iniciar sesión con credenciales seguras, para acceder al sistema de planeación según mis permisos.

#### **Criterios de aceptación**

- Endpoint POST /api/auth/login recibe *email* y *contraseña*.
- Contraseñas almacenadas hasheadas con bcrypt.
- Retorna token JWT con rol y expiración (15 min).
- Roles definidos:
  - Docente: acceso a planeaciones y evidencias propias.
  - Coordinador: acceso a reportes de docentes.
  - Administrador: gestión de usuarios y configuración.
- Rutas protegidas mediante middleware `authorize(roles)`.

## HU-05 — Gestión de usuarios y roles

Como administrador, quiero gestionar los usuarios y asignarles roles, para controlar el acceso y las funciones dentro del sistema.

### Criterios de aceptación

- Endpoints:
  - POST /api/usuarios → crea usuario nuevo.
  - PUT /api/usuarios/:id → actualiza rol o datos.
  - DELETE /api/usuarios/:id → desactiva usuario.
  - Solo accesible al rol **administrador**.
- Validaciones: no permitir duplicados de email.
- Campos obligatorios: nombre, correo, rol, contraseña.
- Al crear, debe enviarse notificación por correo al usuario (API externa).

## HU-06 — Notificaciones automáticas

Como coordinador, quiero que el sistema envíe notificaciones automáticas por correo, para recordar a los docentes los plazos de entrega de evidencias o planeaciones.

### Criterios de aceptación

- Servicio de notificaciones basado en **Observer pattern**.
- Se dispara evento al crear o actualizar una planeación.
- Integra API de correo (SendGrid / Nodemailer).
- Contenido del correo incluye: nombre del docente, periodo y fecha límite.
- Logs de envío deben almacenarse en la base de datos (notifications collection).

## **HU-07 — Aprobación de planeaciones por coordinación**

Como coordinador, quiero aprobar o rechazar las planeaciones enviadas por los docentes, para asegurar el cumplimiento de los lineamientos institucionales.

### **Criterios de aceptación**

- Las planeaciones deben tener estado: *Pendiente, Aprobada o Rechazada*.
- El coordinador puede dejar comentarios de retroalimentación.
- El docente recibe notificación al cambiar el estado.
- Registro de fecha y usuario que realizó la validación.

## **HU-08 — Registro de evidencias de capacitación docente**

Como docente, quiero registrar evidencias de cursos o talleres tomados, para acreditar mi actualización profesional ante la coordinación.

### **Criterios de aceptación**

- Formulario con campos: nombre del curso, institución, fecha y horas acreditadas.
- Permitir adjuntar PDF o imagen de constancia.
- Validación y aprobación por parte del coordinador académico.
- Reporte de cursos acreditados por periodo.

## Backlog de Actividades

El Product Backlog del proyecto “Gestión de Planeación Académica” se compone de las historias de usuario definidas junto con el Product Owner.

Cada historia fue priorizada según su impacto funcional y su dependencia técnica.

En la siguiente tabla se muestran las historias que conforman el backlog inicial del proyecto:

ID	Historia de Usuario	Prioridad	Sprint	Estado
HU-01	Registrar planeación didáctica	<span style="color: red;">■ Alta</span>	Sprint 1	Completada
HU-02	Subir evidencia	<span style="color: red;">■ Alta</span>	Sprint 1	Completada
HU-03	Ver reporte por parcial	<span style="color: red;">■ Alta</span>	Sprint 2	En desarrollo
HU-04	Autenticación y control de acceso	<span style="color: red;">■ Alta</span>	Sprint 2	Completada
HU-05	Gestión de usuarios y roles	<span style="color: orange;">■ Media</span>	Sprint 3	Pendiente
HU-06	Notificaciones automáticas	<span style="color: orange;">■ Media</span>	Sprint 3	Pendiente
HU-07	Aprobación de planeaciones por coordinación	<span style="color: red;">■ Alta</span>	Sprint 1	Completada
HU-08	Registro de evidencias de capacitación docente	<span style="color: orange;">■ Media</span>	Sprint 2	En desarrollo

Cada historia de usuario cuenta con sus criterios de aceptación y definición técnica, los cuales aseguran el cumplimiento de los requerimientos del usuario final.

La priorización se realizó en conjunto con el Product Owner, considerando el valor para el sistema y la complejidad de implementación.

## Planificación del Proyecto

Configuración, arquitectura y autenticación

Fecha	HU	Actividad	Issue	Commit	Responsable	Estado
13/10/25	HU-04	Configuración entorno, Git, estructura backend/front	#1, #2	base backend	Andrea	Hecho
14/10/25	HU-04	Diseño arquitectura, definición frameworks	#3, #4	correción funciones / arquitectura	Andrea	Hecho
15/10/25	HU-04	Implementación login + JWT + roles	#9	implementacion login y permisos	Andrea	Hecho
16/10/25	HU-05	Configuración BD Mongo + modelos Usuario/Roles	#6	Agrega .gitignore / BD / env	Andrea	Hecho
17/10/25	HU-05	CRUD usuarios (admin) + validaciones bcrypt	#5	registro usuarios admin	Andrea	Hecho

Planeaciones, evidencias y reportes

Fecha	HU	Actividad	Issue	Commit	Responsable	Estado
20/10/25	HU-01	Endpoints CRUD planeaciones didácticas	#10	conexion backend + front	Andrea	Hecho
21/10/25	HU-07	Estados de planeación (aprobado/rechazado)	#11	avance componente planeación	Andrea	Hecho
22/10/25	HU-02	Carga de evidencias (PDF/IMG)	#12	validación evidencias	Andrea	Hecho
23/10/25	HU-02	Asociación evidencia–planeación (Cloud storage)	#12	testing evidencias	Andrea	Hecho
24/10/25	HU-03	Reportes por parcial + export PDF/Excel	#13	exportación reportes PDF	Andrea	Hecho

## Notificaciones, despliegue y pruebas

Fecha	HU	Actividad	Issue	Commit	Responsable	Estado
25/10/25	HU-03/HU-08	Integración backend+frontend + cursos docentes	#11	backend+front integración	Andrea	Hecho
26/10/25	HU-05/HU-08	Gestión usuarios + evidencias capacitación	#15	registro usuarios + cursos	Andrea	Hecho
27/10/25	HU-03	Mejoras reportes + README y API docs	#13	README y api	Andrea	Hecho
28/10/25	HU-06	Notificaciones automáticas (Observer + SendGrid)	#14	API sendgrid funcional	Andrea	Hecho
29/10/25	HU-18	Manual de usuario + CI Workflow	#18	Manual usuario + CI	Manu	Hecho
30/10/25	HU-06	Docker + Server config + pruebas finales	#17, #19	Dockerfile + Server.js	Manu	Hecho

# Arquitectura del Software

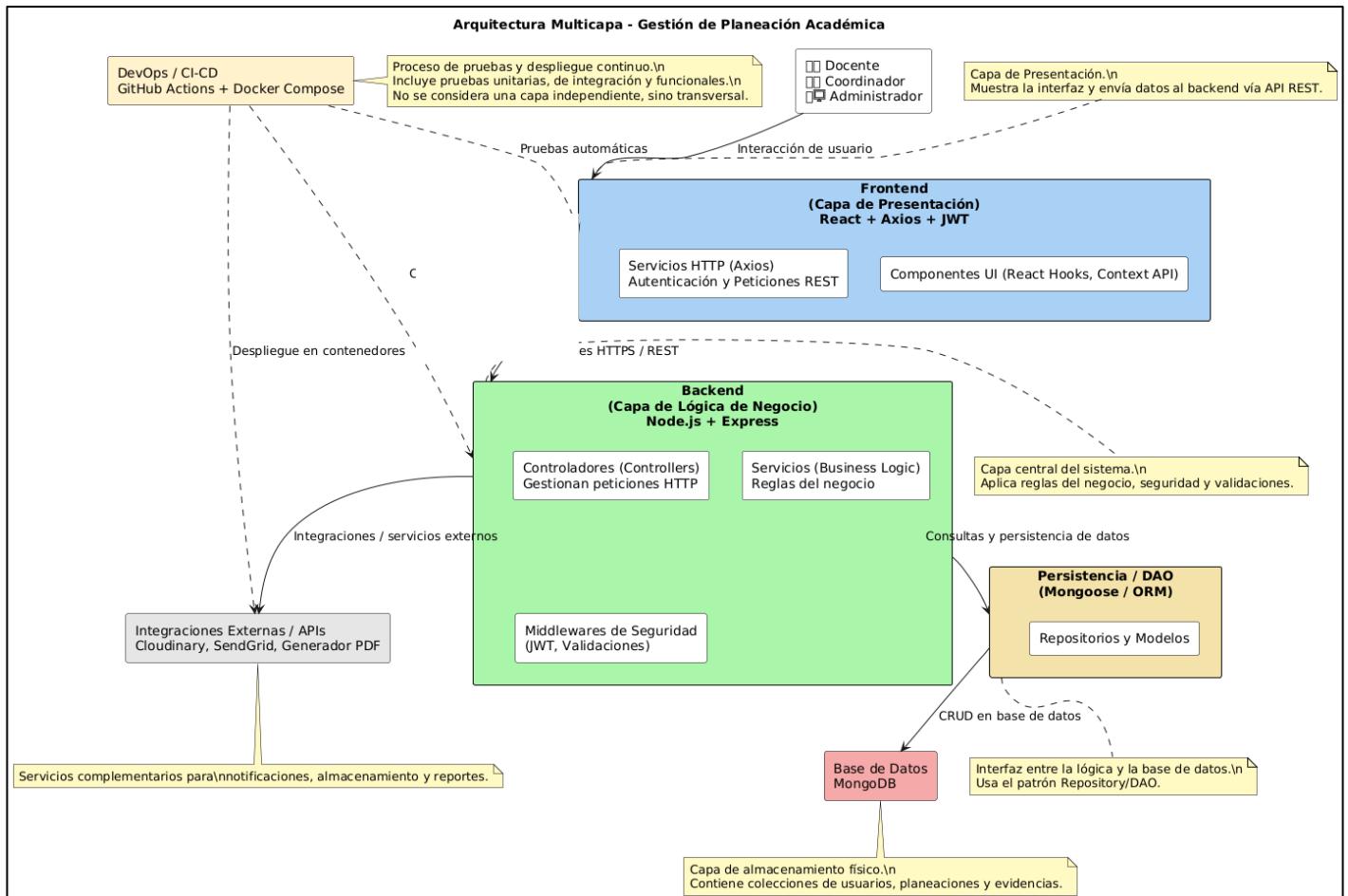
Arquitectura Multicapa (N-Tier) basada en el patrón MVC (Modelo–Vista–Controlador)

## Justificación técnica:

La arquitectura multicapa implementada en el sistema “Gestión de Planeación Académica” permite una separación clara de responsabilidades, garantizando mantenibilidad, seguridad y escalabilidad.

Cada capa del sistema cumple una función específica dentro del flujo de comunicación entre cliente, servidor y base de datos:

## Diagrama de Arquitectura



Las relaciones entre capas siguen una jerarquía clara basada en el principio de separación de responsabilidades.

- La capa de Presentación (Frontend) depende de la Lógica de Negocio (Backend), que a su vez depende de la Persistencia (DAO/Repository) para acceder a la Base de Datos.
- Además, la capa de Integración interactúa con servicios externos como Cloudinary, SendGrid o generadores de reportes, comunicándose de forma segura mediante API REST.
- Esta estructura garantiza la trazabilidad y el desacoplamiento de los módulos, permitiendo una fácil expansión del sistema sin afectar el núcleo del software.

## Estructura general del diagrama

El diagrama de arquitectura presentado representa una arquitectura multicapa (N-tier), organizada en cuatro niveles principales: presentación, lógica de negocio, persistencia y base de datos, además de un proceso transversal de pruebas e integración continua.

El modelo se basa en el patrón de diseño Modelo–Vista–Controlador (MVC), ampliamente utilizado en aplicaciones web modernas por su capacidad para separar las responsabilidades y mejorar la mantenibilidad del sistema.

## Capas de la arquitectura

Esta capa corresponde a la interfaz de usuario, donde se realiza la interacción directa con el sistema. Puede estar desarrollada en frameworks como **Angular**, **React** o **Vue.js**, y su función principal es mostrar la información proveniente del backend y enviar solicitudes mediante **servicios REST**. Aquí se manejan eventos, formularios y visualizaciones gráficas que permiten al usuario comunicarse con la aplicación.

Su objetivo es ofrecer una experiencia intuitiva, validando datos básicos antes de enviarlos al servidor.

## Capa de Lógica de Negocio (Backend / Controlador y Servicio)

En esta capa se concentra el núcleo funcional del sistema.

Está desarrollada con tecnologías como Spring Boot o Node.js, y se encarga de recibir las peticiones del frontend a través de los controladores, procesarlas, aplicar las reglas del negocio y devolver las respuestas adecuadas.

Los controladores gestionan las solicitudes entrantes, mientras que los servicios ejecutan la lógica empresarial (por ejemplo, cálculos, validaciones o decisiones de negocio).

Esta capa implementa los patrones MVC y Inyección de Dependencias, que permiten una estructura modular y facilitan las pruebas unitarias.

## Capa de Persistencia (Repositorio / DAO / Modelo)

Esta capa gestiona la comunicación entre la lógica de negocio y la base de datos.

Utiliza el patrón Repository para abstraer las operaciones de acceso a datos y desacoplar la lógica de persistencia del resto de la aplicación.

En entornos Java, se suele implementar mediante Spring Data JPA o Hibernate, los cuales permiten interactuar con la base de datos a través de objetos y entidades sin depender directamente de consultas SQL.

Su función es almacenar, recuperar, actualizar o eliminar los datos de manera eficiente y segura, garantizando la integridad de la información.

## Capa de Base de Datos (Data Layer)

Esta capa contiene el almacenamiento físico de la información.

Puede estar compuesta por un sistema de gestión de bases de datos relacional (PostgreSQL, MySQL) o no relacional (MongoDB), dependiendo de las necesidades del proyecto.

En ella residen las tablas o colecciones que almacenan los datos persistentes del sistema, como usuarios, productos, roles o transacciones.

El backend se comunica con esta capa a través de los repositorios definidos en la capa de persistencia.

## Pruebas (Testing e Integración Continua):

Las pruebas no constituyen una capa independiente dentro de la arquitectura, sino un proceso transversal que se aplica sobre todas las capas del sistema.

Incluye pruebas unitarias (para controladores y servicios), pruebas de integración (para validar la comunicación entre backend y base de datos) y pruebas funcionales (para el frontend).

Estas pruebas se ejecutan en el entorno de integración continua (CI/CD), usando herramientas como GitHub Actions y Docker Compose, garantizando la calidad del software antes del despliegue.

## Flujo de datos

El flujo de información dentro de la arquitectura sigue los siguientes pasos:

- El **usuario** interactúa con la **interfaz gráfica** desde el navegador o aplicación móvil.
- El **frontend (capa de presentación)** envía la solicitud al **backend (lógica de negocio)** mediante una **API REST**.
- El **controlador** correspondiente recibe la petición y la deriva al **servicio** apropiado.
- El **servicio** aplica las reglas del negocio y solicita los datos necesarios a la **capa de persistencia**.
- El **repositorio o DAO** ejecuta la consulta en la **base de datos** y devuelve los resultados al **servicio**.
- El **servicio** procesa la respuesta y la envía nuevamente al **controlador**.
- El **controlador** retorna la respuesta al **frontend**, la prueba final de la muestra al usuario.
- Paralelamente, la **capa de pruebas de integración** puede interceptar y evaluar estos procesos para validar su correcto funcionamiento.

De manera transversal, el proceso de pruebas e integración continua supervisa la correcta comunicación entre las capas antes del despliegue, asegurando la calidad y estabilidad del sistema.

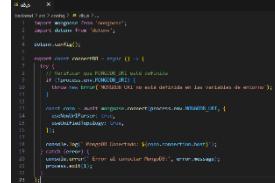
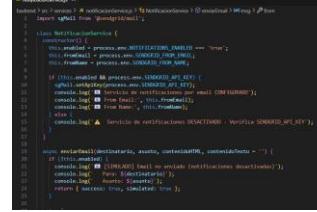
## Entorno de ejecución

Cada capa de la arquitectura se ejecuta en un entorno distinto:

- **Frontend:** ejecutado en el navegador web del usuario o en un cliente móvil.
- **Backend:** alojado en un servidor de aplicaciones (por ejemplo, Tomcat, Apache o un contenedor Spring Boot).
- **Base de datos:** desplegada en un servidor local o remoto, gestionando la persistencia de datos.

- **Pruebas:** ejecutadas dentro de un entorno de **integración continua (CI/CD)**, como Jenkins o GitHub Actions, donde se validan las versiones antes del despliegue.

## Patrones de Diseño Aplicados

Capa / Componente	Patrón de Diseño	Tipo	Descripción / Aplicación	Código
Presentación (Frontend)	Modelo–Vista–Controlador (MVC)	Estructural	Separación entre lógica de presentación, estado y componentes React (Vistas).	
Lógica de Negocio (Backend)	Inyección de Dependencias	Creacional	Los controladores reciben los servicios como dependencias, reduciendo acoplamiento.	
Persistencia (DAO / Repository)	Repository / DAO Pattern	Estructural	Los modelos Mongoose encapsulan las operaciones CRUD.	
Conexión a Base de Datos	Singleton	Creacional	config/db.js mantiene una única instancia de conexión MongoDB.	
Notificaciones (Servicios externos)	Observer	Comportamiento	services/notificacionService.js observa eventos y dispara correos automáticos.	
Pruebas y CI/CD	Mock Objects / Integration Test Pattern	Emergente	Se simulan componentes para ejecutar pruebas unitarias e integradas con Jest y Supertest.	

## Frameworks y entorno de desarrollo

Frontend: React

Justificación técnica:

- React fue seleccionado como framework principal para la capa de presentación debido a su arquitectura basada en componentes reutilizables, que permite construir interfaces dinámicas, modulares y fáciles de mantener.

Entre sus ventajas destacan:

- Desarrollo rápido: reutilización de componentes y manejo eficiente del estado mediante Hooks.
- Interfaz interactiva: actualizaciones instantáneas en la vista gracias al Virtual DOM.
- Compatibilidad con librerías externas: integración fluida con Axios, Redux, Bootstrap o Material UI.
- Comunidad activa y soporte continuo: garantiza estabilidad y acceso a documentación actualizada.

Responsabilidades del Frontend:

- Interfaz de usuario (UI) adaptable a diferentes roles (docente, coordinador, administrador).
- Consumo de la API REST para enviar y recibir información en tiempo real.
- Implementación de formularios interactivos y validaciones dinámicas.
- Representación visual de reportes, gráficas y progreso de planeaciones.

Backend: Node.js con Express

- Node.js con Express se eligió como entorno backend por su rendimiento no bloqueante (asíncrono), escalabilidad y facilidad para implementar servicios RESTful.
- Ofrece integración sencilla con MongoDB y APIs externas, además de soportar despliegues en contenedores Docker.

Ventajas técnicas:

- Eficiencia: maneja múltiples solicitudes simultáneamente sin saturar el servidor.
- Flexibilidad: estructura modular y ligera para controladores, rutas y servicios.
- Compatibilidad: integración directa con JWT para autenticación segura y envío de correos vía APIs.
- Escalabilidad horizontal: ideal para microservicios o despliegues distribuidos.

Responsabilidades del Backend:

- Gestión de peticiones HTTP desde el frontend.
- Implementación de reglas de negocio y validaciones.
- Conexión segura con la base de datos MongoDB.
- Generación y exportación de reportes (PDF, Excel).
- Integración de servicios externos (notificaciones, almacenamiento).

## Control de Versiones y Planeación del Proyecto en GitHub

Herramienta	Función Principal
Node.js y npm	Ejecución del servidor backend y gestión de dependencias.
React.js	Construcción del frontend modular e interactivo.
VS Code	Entorno de desarrollo principal con soporte para ESLint y Prettier.
Git + GitHub	Control de versiones, repositorio remoto y automatización.
Docker	Contenerización de backend y frontend para despliegue local o remoto.
MongoDB Compass / Atlas	Administración y monitoreo de la base de datos.

Configuraciones clave:

- Variables de entorno definidas en .env.
- Scripts automatizados en package.json (npm run dev, npm test).
- Uso de GitHub Issues para seguimiento de tareas y bugs.
- Despliegue de pruebas mediante Docker Compose.

## Control de Versiones y Planeación del Proyecto en GitHub

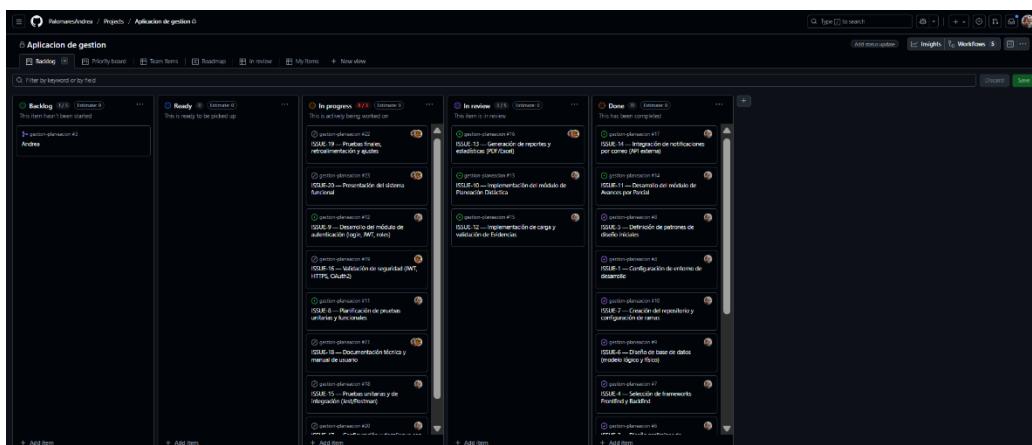
Plataforma de Control de Versiones

Plata Justificación técnica:

GitHub se adoptó por ser una herramienta robusta que facilita la colaboración y garantiza trazabilidad en todo el ciclo de desarrollo.

Sus principales ventajas son:

- Control detallado de versiones: permite rastrear cada cambio en el código.
- Colaboración simultánea: varios desarrolladores pueden trabajar en paralelo.
- Automatización: soporte nativo para GitHub Actions, CI/CD y despliegues automáticos.
- Trazabilidad: seguimiento completo de issues, commits y pull requests.
- Integración directa con VS Code y Node.js: permite flujos de trabajo eficientes desde el entorno de desarrollo.



```
PS D:\gestion-academica\gestion-planeacion\frontend> git log --oneline
dfffb8c2 (HEAD -> Andrea, origin/Andrea) README y api
e8c13aa implementacion de registro de usuarios mediante el admin y listado de los usuarios
9aa09a5 correccion de detalles mas correcto funcionamiento de exportacion y importacion de reportes pdf
03be998 implementacion de login y manejo de rutas con permisos dependientes de su rol
549d48f testeando cambios
02184eb Agrega .gitignore correctamente e ignora backend/.env
78de638 ajuste masivo de frontend con nuevas funcionalidades y implementacion de API para gmali de SendGrid
629091b correccion en flujo de Avance componente
f0dd15f conexion masiva de backend con front y diseños simples
321189f Agrega API de geolocalización con modelo, rutas y controlador
80b910c correccion de funciones
a9824b1 base backend
PS D:\gestion-academica\gestion-planeacion\frontend>
```

## Estrategia de Ramas

Rama	Propósito	Descripción
main	Rama principal	Contiene la versión estable del proyecto. Solo se actualiza mediante <i>Pull Requests</i> revisados y aprobados.
Andrea	Rama de desarrollo	Aquí se integran nuevas funcionalidades antes de ser fusionadas a main.
Manuel	Rama de desarrollo	Aquí se integran nuevas funcionalidades antes de ser fusionadas a main.

Enlace a Github: <https://github.com/PalomaresAndrea/gestion-planeacion>

## Políticas de Acceso y Merge

Para mantener la integridad del código fuente, se establecieron políticas de control:

- Solo los colaboradores autorizados pueden realizar push directo a la rama main.
- Todos los cambios deben pasar por un Pull Request (PR).
- Se exige revisión de código y aprobación antes de realizar el merge.
- En caso de conflictos, estos deben resolverse manualmente por el desarrollador responsable.
- Se emplean labels (etiquetas) para clasificar los issues:
  - **Bug:** errores identificados.
  - **Feature:** nuevas funcionalidades.
  - **Enhancement:** mejoras o refactorización.
  - **Documentation:** actualizaciones de documentación técnica.

## Automatización con GitHub Workflows

Para garantizar la calidad, consistencia y despliegue eficiente del sistema, se implementaron Workflows automáticos mediante GitHub Actions, definidos en archivos .yml.

- **Workflow principales:**

### Pruebas Automáticas (Test Workflow)

- Se ejecuta automáticamente al realizar push o pull request sobre la rama develop.
- Ejecuta las pruebas unitarias con Jest y verifica que el código compile correctamente.
- Reporta los resultados en la pestaña de Actions.

### Validación de Pull Requests (PR Validation)

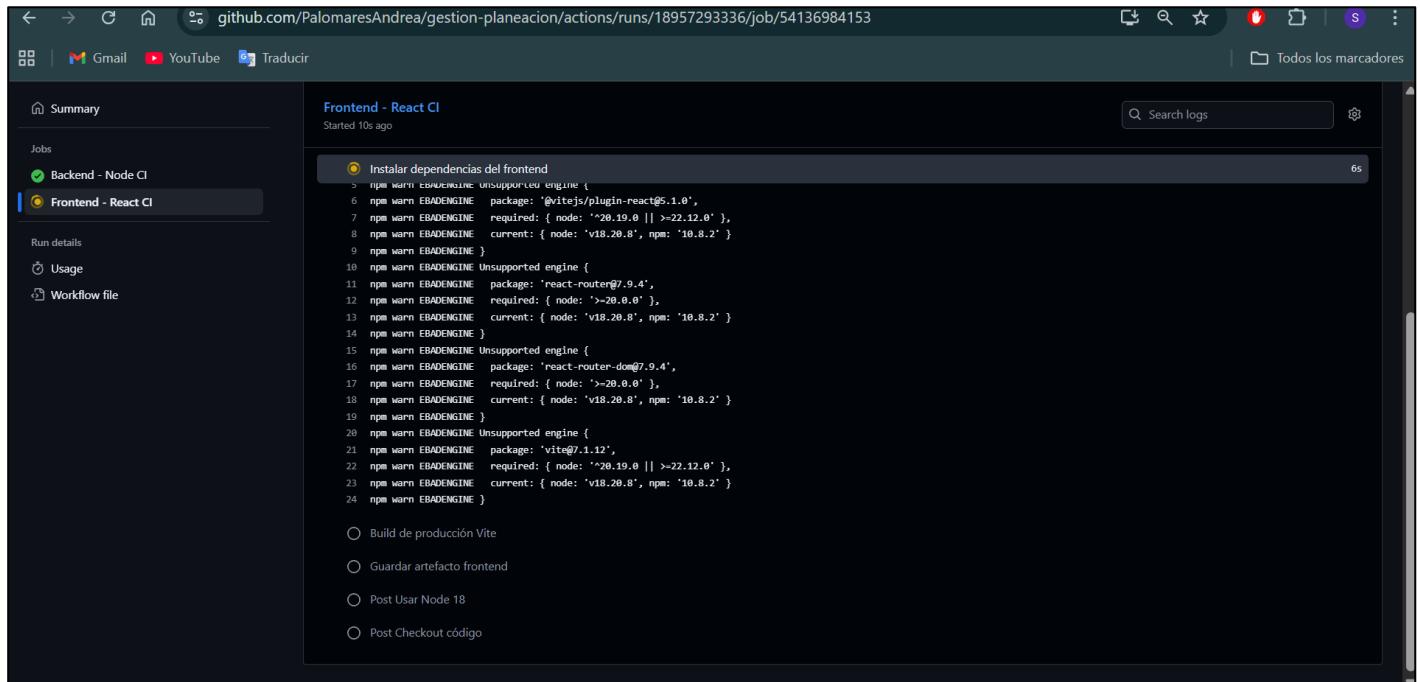
- Revisa automáticamente los cambios propuestos antes de su integración a main.
- Evalúa la calidad del código, convenciones, dependencias y conflictos.
- Si alguna verificación falla, el PR queda bloqueado hasta ser corregido.

### Despliegue Temporal (Deploy Workflow)

- Compila automáticamente el proyecto backend y frontend.
- Genera un contenedor Docker de prueba.
- Despliega el sistema en un entorno temporal o servidor local para validación final.

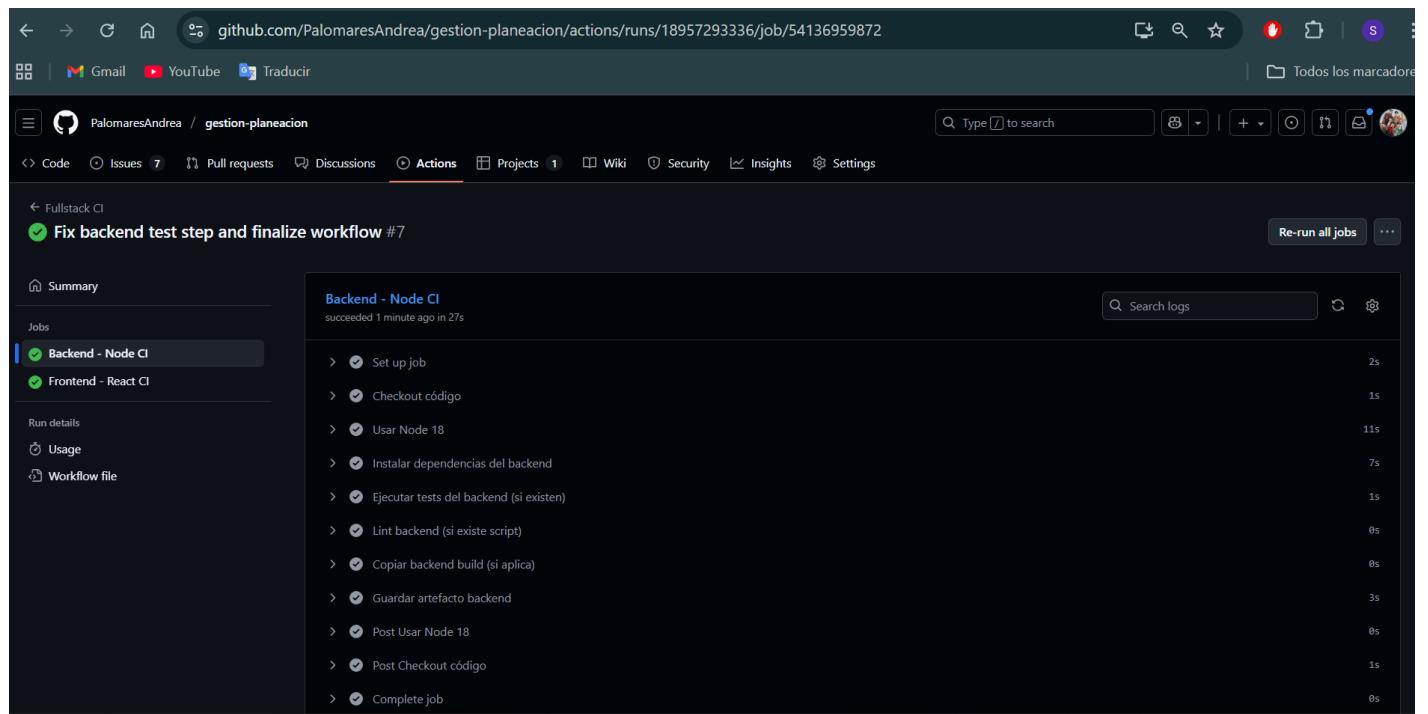
# Workflows

## fullstack-ci.yml



```
Frontend - React CI
Started 10s ago
6s
Instalar dependencias del frontend
5 npm warn EBADENGINE unsupported engine {
6 npm warn EBADENGINE   package: 'vite@5.1.0',
7 npm warn EBADENGINE   required: { node: '^20.19.0 || >22.12.0' },
8 npm warn EBADENGINE   current: { node: 'v18.20.8', npm: '10.8.2' }
9 npm warn EBADENGINE }
10 npm warn EBADENGINE Unsupported engine {
11 npm warn EBADENGINE   package: 'react-router-dom@7.9.4',
12 npm warn EBADENGINE   required: { node: '>20.0.0' },
13 npm warn EBADENGINE   current: { node: 'v18.20.8', npm: '10.8.2' }
14 npm warn EBADENGINE }
15 npm warn EBADENGINE Unsupported engine {
16 npm warn EBADENGINE   package: 'react-router-dom@7.9.4',
17 npm warn EBADENGINE   required: { node: '>20.0.0' },
18 npm warn EBADENGINE   current: { node: 'v18.20.8', npm: '10.8.2' }
19 npm warn EBADENGINE }
20 npm warn EBADENGINE Unsupported engine {
21 npm warn EBADENGINE   package: 'vite@7.1.12',
22 npm warn EBADENGINE   required: { node: '^20.19.0 || >22.12.0' },
23 npm warn EBADENGINE   current: { node: 'v18.20.8', npm: '10.8.2' }
24 npm warn EBADENGINE }

○ Build de producción Vite
○ Guardar artefacto frontend
○ Post Usar Node 18
○ Post Checkout código
```



```
Backend - Node CI
succeeded 1 minute ago in 27s
Re-run all jobs ...
2s
○ Set up job
1s
○ Checkout código
11s
○ Usar Node 18
7s
○ Instalar dependencias del backend
1s
○ Ejecutar tests del backend (si existen)
0s
○ Lint backend (si existe script)
3s
○ Copiar backend build (si aplica)
0s
○ Guardar artefacto backend
0s
○ Post Usar Node 18
1s
○ Post Checkout código
0s
○ Complete job
```

test.yml

Backend - Node CI  
Started 20s ago

Run details

Usage

Workflow file

Summary

Jobs

Backend - Node CI

User Node 18

Instalar dependencias del backend

```
1 ► Run npm install
4
5 removed 30 packages, and audited 162 packages in 6s
6 28 packages are looking for funding
7 run 'npm fund' for details
8 1 moderate severity vulnerability
9 To address all issues, run:
10 npm audit fix
11 Run 'npm audit' for details.
```

Ejecutar tests del backend (si existen)

```
1 ► Run npm test --if-present || echo "Sin tests"
4
5 > backend@1.0.0 test
6 > echo "Error: no test specified" && exit 1
7 Error: no test specified
8 Sin tests
```

Lint backend (si existe script)

```
1 ► Run npm run lint --if-present
```

Copiar backend build (si aplica)

```
1 ► Run mkdir -p ../build/backend
```

Guardar artefactos backend

Search logs

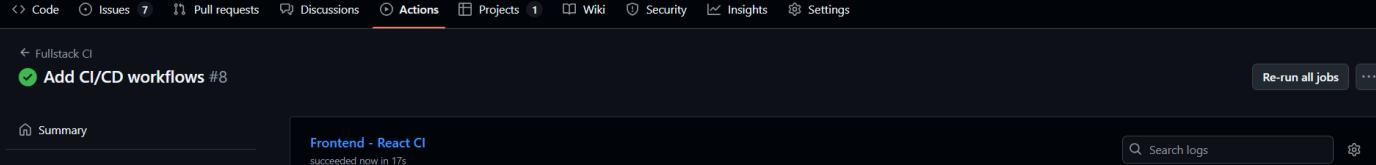
6s

8s

0s

0s

0s



The screenshot shows a GitHub Actions CI/CD workflow for a 'Frontend - React CI' job. The job status is 'Succeeded now in 17s'. The workflow steps are as follows:

- Set up job (0s)
- Checkout código (2s)
- Usar Node 18 (2s)
- Instalar dependencias del frontend (6s)
- Build de producción Vite (2s)
- Guardar artefacto frontend (0s)
- Post Usar Node 18 (0s)
- Post Checkout código (0s)
- Complete job (0s)

On the left sidebar, the 'Frontend - React CI' job is selected. The 'Summary' tab is active. Other tabs include 'Code', 'Issues', 'Pull requests', 'Discussions', 'Actions', 'Projects', 'Wiki', 'Security', 'Insights', and 'Settings'. The top navigation bar shows the repository 'PalomaresAndrea / gestion-planeacion' and a search bar with the placeholder 'Type to search'. The top right corner features user profile icons.

All workflows	Filter workflow runs			
Showing runs from all workflows				
8 workflow runs	Event ▾	Status ▾	Branch ▾	Actor ▾
<b>✓ Add CI/CD workflows</b> Fullstack CI #8: Commit <a href="#">7ae3d13</a> pushed by <a href="#">PalomaresAndrea</a>	main	⌚ 1 minute ago	...	⌚ 42s
<b>✓ Fix backend test step and finalize workflow</b> Fullstack CI #7: Commit <a href="#">af7de38</a> pushed by <a href="#">PalomaresAndrea</a>	main	⌚ 25 minutes ago	...	⌚ 47s

## Evidencia Técnica de Integración y Pruebas

La integración de componentes se evidencia a través de las herramientas utilizadas durante el ciclo de vida del proyecto:

- GitHub Projects: Planeación ágil del flujo de trabajo mediante issues y tareas enlazadas.
- Git CLI y VS Code: Control de versiones y gestión de ramas (main, Andrea, Manuel).
- Docker Compose: Despliegue de backend y frontend en contenedores.
- GitHub Actions: Automatización del pipeline de pruebas, validación de PR y despliegue temporal.

### 1. Captura de **Swagger** al consumir un endpoint.

**Autenticación** Gestión de usuarios y autenticación

**POST** /api/auth/registrar Registrar nuevo usuario

Parameters

No parameters

Request body **required**

application/json

Edit Value | Schema

```
{ "email": "juan.perez@universidad.edu.mx", "password": "MiClaveEsSegura123!", "nombre": "Juan Pérez López", "rol": "profesor", "numeroEmpleado": "EMP2025001", "departamento": "Ingeniería de Sistemas", "telefono": "+52 55 1234 5678", "especialidad": "Desarrollo de Software", "materias": [ "Programación Avanzada", "Bases de Datos", "Ingeniería de Software" ] }
```

Execute Clear

Request URL

http://localhost:4000/api/auth/registrar

Server response

Code Details

201 Response body

```
{ "message": "usuario registrado exitosamente.", "token": "eyJhbGciOiJIUzI1NiIsInR5cIjI6IkpXVCJ9.eyJpZCI6IjY5MDJlZDM1M2VYTNjMDRlMWRkNjQ1ZiIiSImIhdCI6MTc2MTc5OTQ3OCwiZXhwIjoxNzY0MzKxNDc4fQ.FM-dV1rZEaJkoebD9xbmNGV0Td3oZqKYAX390eLyLvo", "usuario": { "id": "6902ed353eda3c04e1dd645f", "email": "juan.perez@universidad.edu.mx", "nombre": "Juan Pérez López", "rol": "profesor" } }
```

Explicar Download

Response headers

```
access-control-allow-origin: *  
connection: keep-alive  
content-length: 359  
content-type: application/json; charset=utf-8  
date: Thu, 30 Oct 2025 04:44:38 GMT  
etag: W/"167-3YHiv0zYnra4CxByxmDFTVNZBBE"  
keep-alive: timeout=5  
x-powered-by: Express
```

Responses

Code	Description	Links
201	Usuario registrado exitosamente	No links

localhost:4000/api-docs/#/Autenticación/post\_api\_auth\_login

Importar favoritos | Lenovo Support | Lenovo | McAfee | youtube.com > Jey... | Docs | Videojuegos - Goo...

GET /api/auth/usuarios Obtener todos los usuarios (solo admin)

POST /api/auth/login Iniciar sesión

Parameters

No parameters

Request body required

application/json

```
{ "email": "juan.perez@universidad.edu.mx",  
  "password": "MiClaveSegura123!" }
```

Execute Clear

Responses

Curl

```
curl -X 'POST' \  
  'http://localhost:4000/api/auth/login' \  
  -H 'accept: */*' \  
  -H 'Content-Type: application/json' \  
  -d ' {  
    "email": "juan.perez@universidad.edu.mx",  
    "password": "MiClaveSegura123!"  
  }'
```

Request URL

http://localhost:4000/api/auth/login

Server response

Code Details

localhost:4000/api-docs/#/Autenticación/post\_api\_auth\_login

Importar favoritos | Lenovo Support | Lenovo | McAfee | youtube.com > Jey... | Docs | Videojuegos - Goo...

Curl

```
curl -X 'POST' \  
  'http://localhost:4000/api/auth/login' \  
  -H 'accept: */*' \  
  -H 'Content-Type: application/json' \  
  -d ' {  
    "email": "juan.perez@universidad.edu.mx",  
    "password": "MiClaveSegura123!"  
  }'
```

Request URL

http://localhost:4000/api/auth/login

Server response

Code Details

200 Response body

```
{ "message": "Login exitoso.",  
  "token": "eyJhbGciOiJIb3JuZW1hbmR5cC16IkpxVCJ9.eyJpZC1G1jY5MDJ1ZDM1N2VYTNjM0R1MwRkNjQ1Z1i1sIm1hdC16HTc2Mtc50Tg4Kw1ZXhW1joxNzY0Htzk00gwfQ.-OvrVXHBNzgBaw0k0vYCaLdpet1d0zUkrYClprwmu",  
  "usuario": {  
    "id": "6902ed353ed43c304ed1dd645",  
    "email": "juan.perez@universidad.edu.mx",  
    "nombre": "Juan Pérez López",  
    "rol": "profesional",  
    "numeroEmpleado": "EMP2025001",  
    "departamento": "Ingeniería de Sistemas"  
  } }
```

Download

Response headers

```
access-control-allow-origin: *  
connection: keep-alive  
content-length: 412  
content-type: application/json; charset=utf-8  
date: Thu, 30 Oct 2025 04:51:20 GMT  
etag: W"/...-c1024prg63tmcusbe3h01NK+k"  
last-modified: Thu, 30 Oct 2025 04:51:20 GMT  
x-powered-by: Express
```

Responses

Code	Description	Links
200	Login exitoso	No links
400	Campos requeridos faltantes	No links
401	Credenciales inválidas	No links
500	Error del servidor	No links

GET /api/auth/perfil Obtener perfil del usuario autenticado

PUT /api/auth/perfil Actualizar perfil del usuario

## 2. Captura de Docker corriendo (docker ps).

```
desktop-linux Docker Desktop          npipe:///./pipe/dockerDesktopLinuxEngine
PS C:\Users\mataj\Documents\Global\gestion-planeacion> docker build -t gestion-planeacion .
[+] Building 60.6s (7/16)
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 982B
=> [internal] load metadata for docker.io/library/node:20-alpine
=> [auth] library/node:pull token for registry-1.docker.io
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [frontend-build 1/6] FROM docker.io/library/node:20-alpine@sha256:6178e78b972f79c335df281f4b7674a2d85071aae2af020ffa39f0a770
=> => resolve docker.io/library/node:20-alpine@sha256:6178e78b972f79c335df281f4b7674a2d85071aae2af020ffa39f0a770265435
=> => sha256:da04d522c98fe12816b2bcdff8413fc73645f8fa60f287c672f58bcc7f0fa38 444B / 444B
=> => sha256:e74e4ed823e9560b3fe51c0cab47dbdfc4b12453604319408ec58708fb9e720 1.26MB / 1.26MB
=> => sha256:60e45a9660cfaebbac9bba98180aa28b3966b7f2462d132c46f51a1f5b25a64 42.75MB / 42.75MB
=> => sha256:2d35ebdb57d9971fea0cac1582aa78935adf8058b2cc32db163c98822e5dfa1b 3.80MB / 3.80MB
=> => extracting sha256:2d35ebdb57d9971fea0cac1582aa78935adf8058b2cc32db163c98822e5dfa1b
=> => extracting sha256:60e45a9660cfaebbac9bba98180aa28b3966b7f2462d132c46f51a1f5b25a64
=> => extracting sha256:e74e4ed823e9560b3fe51c0cab47dbdfc4b12453604319408ec58708fb9e720
=> => extracting sha256:da04d522c98fe12816b2bcdff8413fc73645f8fa60f287c672f58bcc7f0fa38
=> [internal] load build context
=> => transferring context: 18.48MB
=> [frontend-build 2/6] WORKDIR /app/frontend
=> [backend 2/6] WORKDIR /app/backend

```

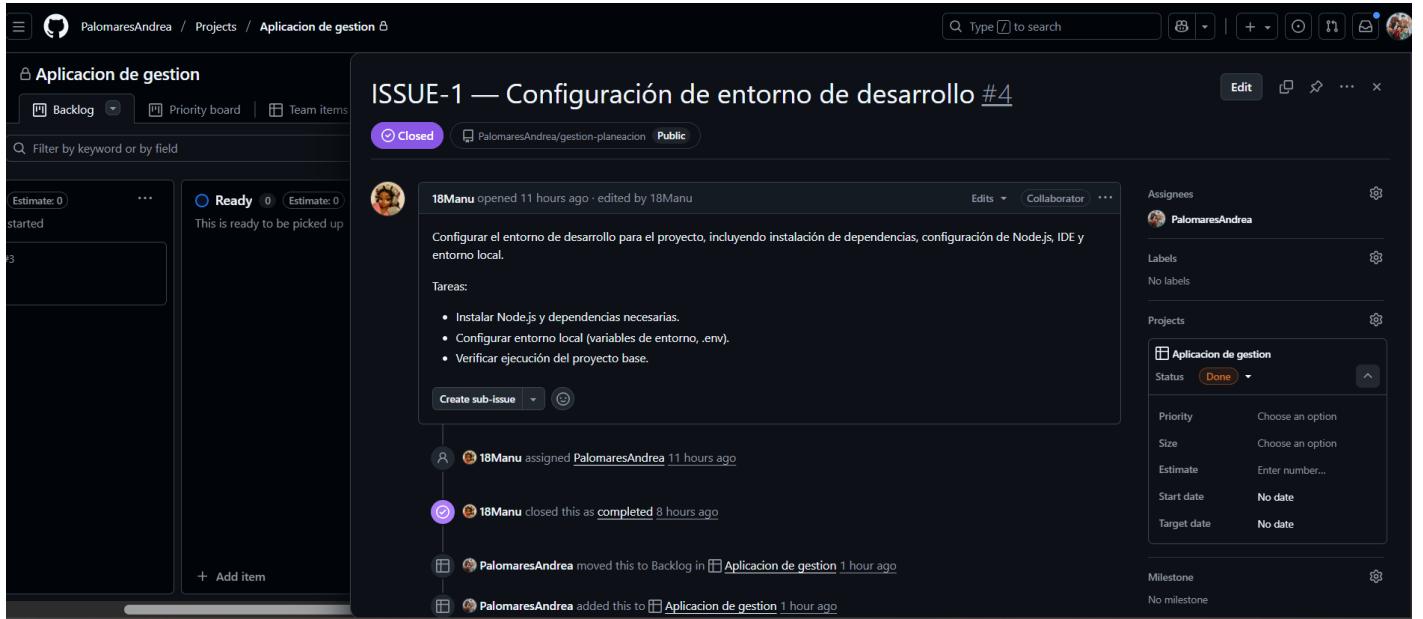
```
C:\Users\mataj>docker images
REPOSITORY          TAG      IMAGE ID      CREATED        SIZE
gestion-planeacion  latest   ef092fb6c776  32 seconds ago  356MB

C:\Users\mataj>
```

```
PS C:\Users\mataj\Documents\Global\gestion-planeacion> docker run -d -p 3000:3000 --name gestion-app gestion-planeacion
e30b128049da2d6537fcf75731d549138d7f1a8890b0937080d897722d9aece6
PS C:\Users\mataj\Documents\Global\gestion-planeacion>
PS C:\Users\mataj\Documents\Global\gestion-planeacion> docker run -d -p 3000:3000 --name gestion-app gestion-planeacion
e30b128049da2d6537fcf75731d549138d7f1a8890b0937080d897722d9aece6
```

## 3. Captura del tablero de GitHub Projects con actividades completadas.

#### 4. Captura del issue cerrado (PR mergeado).



The screenshot shows a Jira issue page for ISSUE-1, titled "Configuración de entorno de desarrollo #4". The issue is marked as "closed" and "public". The description states: "Configurar el entorno de desarrollo para el proyecto, incluyendo instalación de dependencias, configuración de Node.js, IDE y entorno local." The "Tareas" section lists:

- Instalar Node.js y dependencias necesarias.
- Configurar entorno local (variables de entorno, .env).
- Verificar ejecución del proyecto base.

The "Assignees" field shows "PalomaresAndrea". The "Labels" field is empty. The "Projects" field shows "Aplicacion de gestion". The "Status" is set to "Done". The "Priority", "Size", and "Estimate" fields are empty. The "Start date" and "Target date" are set to "No date". The "Milestone" field is empty.

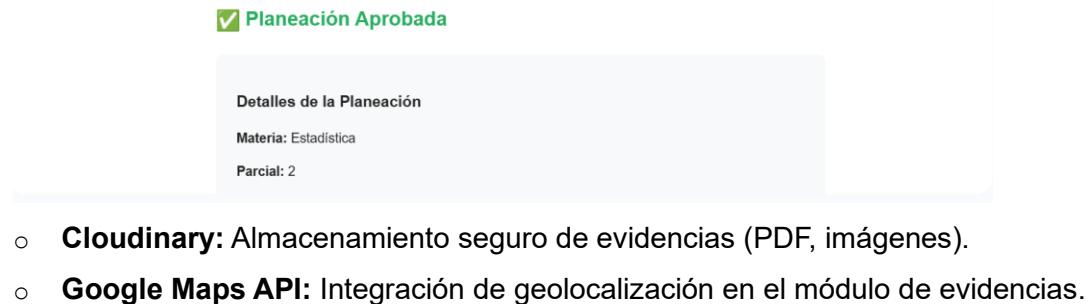
Recent activity on the issue:

- 18Manu assigned PalomaresAndrea 11 hours ago
- 18Manu closed this as completed 8 hours ago
- PalomaresAndrea moved this to Backlog in Aplicacion de gestion 1 hour ago
- PalomaresAndrea added this to Aplicacion de gestion 1 hour ago

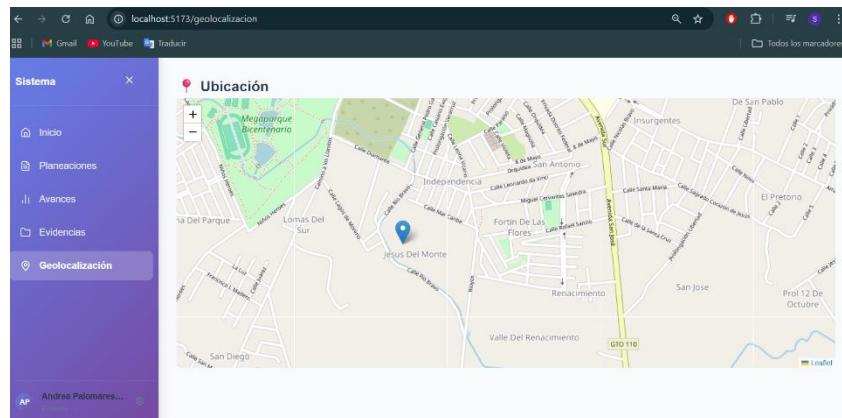
## Seguridad e Integración de Componentes

El sistema implementa **mecanismos de seguridad y autenticación** para garantizar la integridad de los datos:

- Autenticación y Autorización:
  - Uso de JWT (JSON Web Token) para validar credenciales y roles.
  - Las rutas protegidas se gestionan mediante middlewares auth.js y roles.js.
- Comunicación Segura (HTTPS):
  - El proxy Nginx con certificado SSL asegura el intercambio de datos entre frontend y backend.
  - Esto cumple con los principios de confidencialidad y autenticidad del estándar OWASP
- APIs de Terceros Integradas:
  - **SendGrid:** Envío de notificaciones automáticas por correo.



- **Cloudinary:** Almacenamiento seguro de evidencias (PDF, imágenes).
- **Google Maps API:** Integración de geolocalización en el módulo de evidencias.



## Servicios REST Propios (5):

El sistema implementa un conjunto de servicios REST propios, desarrollados con Node.js, Express y MongoDB, que exponen funcionalidades clave para la gestión académica.

Todos los servicios cuentan con autenticación JWT, middleware de validación y documentación mediante Swagger.

### Servicios REST implementados (back-end propio)

Servicio	Endpoint	Seguridad
Autenticación	/api/auth/login /api/auth/perfil	JWT obligatorio
Usuarios	/api/auth/usuarios	Solo Admin (RBAC)
Planeaciones	/api/planeaciones	JWT + validación de rol
Avances	/api/avances	JWT
Evidencias	/api/evidencias	JWT + multer p/archivos
Reportes	/api/reportes	JWT + Coordinador/Admin
Geolocalización	/api/geolocalizacion	Endpoint público + fallback

The screenshot shows the Swagger UI interface for the API documentation. It lists the following endpoints:

- Planeaciones** (Endpoints para planeaciones):
  - GET /api/planeaciones: Obtener todas las planeaciones (con filtros opcionales)
  - POST /api/planeaciones: Crear una nueva planeación didáctica CON ARCHIVO PDF
  - GET /api/planeaciones/ciclo-actual: Obtener planeaciones del ciclo escolar actual
  - GET /api/planeaciones/historial: Obtener historial de planeaciones por profesor y materia
  - GET /api/planeaciones/{id}: Obtener una planeación por ID
  - PUT /api/planeaciones/{id}: Actualizar una planeación
  - PUT /api/planeaciones/{id}/revisar: Revisar planeación (para coordinador)
  - GET /api/planeaciones/{id}/archive: Descargar archivo de planeación
  - GET /api/planeaciones/{id}/ver: Ver archivo de planeación en el navegador
- Avances** (Endpoints para control de avances por parcial):
  - GET /api/avances: Obtener todos los avances registrados (con filtros opcionales)
  - POST /api/avances: Crear un nuevo avance
  - GET /api/avances/estadisticas-profesor: Obtener estadísticas de cumplimiento por profesor
  - GET /api/avances/reporte-general: Obtener reporte general de cumplimiento
  - GET /api/avances/gráficas: Obtener datos para gráficas de cumplimiento
  - GET /api/avances/{id}: Obtener un avance por ID
  - PUT /api/avances/{id}: Actualizar avance por ID
  - DELETE /api/avances/{id}: Eliminar avance por ID
  - POST /api/avances/recordatorios: Envío recordatorios por correo a profesores con avances pendientes
- Evidencias** (Endpoints para gestión de evidencias de capacitación docente):
  - GET /api/evidencias: Obtener todas las evidencias registradas (con filtros opcionales)
  - POST /api/evidencias: Crear nueva evidencia
  - GET /api/evidencias/buscar: Buscar evidencias por nombre de curso, institución o profesor
  - GET /api/evidencias/estadísticas-profesor: Obtener estadísticas de evidencias por profesor
  - GET /api/evidencias/reporte-general: Obtener reporte general de evidencias
  - GET /api/evidencias/{id}: Obtener una evidencia por ID
  - PUT /api/evidencias/{id}: Actualizar evidencia por ID
  - DELETE /api/evidencias/{id}: Eliminar evidencia por ID
  - PUT /api/evidencias/{id}/validar: Validar o rechazar evidencia (para coordinador)
- Reportes** (Endpoints para reportes institucionales y exportación):
  - GET /api/reportes/institucional: Obtener reporte institucional consolidado
  - GET /api/reportes/profesor: Obtener reporte detallado por profesor
  - GET /api/reportes/exportar: Exportar reporte en diferentes formatos
- Autenticación** (Gestión de usuarios y autenticación):
  - POST /api/auth/registro: Registrar nuevo usuario

## Seguridad aplicada (OWASP base)

Medida	Implementación
JWT Auth	Login + middleware verifyToken
RBAC (Roles)	Coordinador / Docente
CORS Controlado	Config en Express
Validación de datos	Body / headers sanitize
Errores controlados	Middleware global
No credenciales en repo	.env + dotenv
Protección uploads	multer + type check

```
! deploy-preview.yml M Geolocalizacion.jsx U Layout.jsx M .env X App.jsx M
backend > .env
1 PORT=4000
2 MONGODB_URI=mongodb://localhost:27017/gestion-planeacion
3 NODE_ENV=development
4
5 # SendGrid Configuration
6 SENDGRID_API_KEY=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
7 SENDGRID_FROM_EMAIL=palomaresschoenstantt@gmail.com
8 SENDGRID_FROM_NAME=Sistema de Gestión Académica
9
10 # Configuración de notificaciones
11 NOTIFICATIONS_ENABLED=true
12 REMINDER_DAYS_BEFORE=3
```

## Despliegue y Escaneo de Seguridad

- **Despliegue en Docker:**

El sistema se despliega mediante Docker Compose, ejecutando los servicios frontend, backend y mongo en contenedores independientes.

Esto facilita la portabilidad y la integración con GitHub Actions CI/CD.

- **Infraestructura en la nube:**

La base de datos está alojada en MongoDB Atlas, mientras que el frontend puede publicarse en Render o Vercel, y el backend en Railway (cuentas gratuitas de estudiante).

- **Escaneo de seguridad (OWASP / Snyk):**

El proyecto se sometió a un análisis de dependencias mediante Snyk, identificando y mitigando vulnerabilidades conforme a los 10 principios OWASP, especialmente en autenticación, validación de entradas y exposición de datos sensibles.

Docker en Back (Pruebas)

```
backend > 📄 Dockerfile
1  # Imagen base
2  FROM node:18
3
4  # Crear carpeta app
5  WORKDIR /app
6
7  # Copiar package.json
8  COPY package*.json ./
9
10 # Instalar dependencias
11 RUN npm install
12
13 # Copiar resto del código
14 COPY . .
15
16 # Exponer puerto del backend
17 EXPOSE 4000
18
19 # Iniciar servidor
20 CMD ["npm", "run", "dev"]
21
```

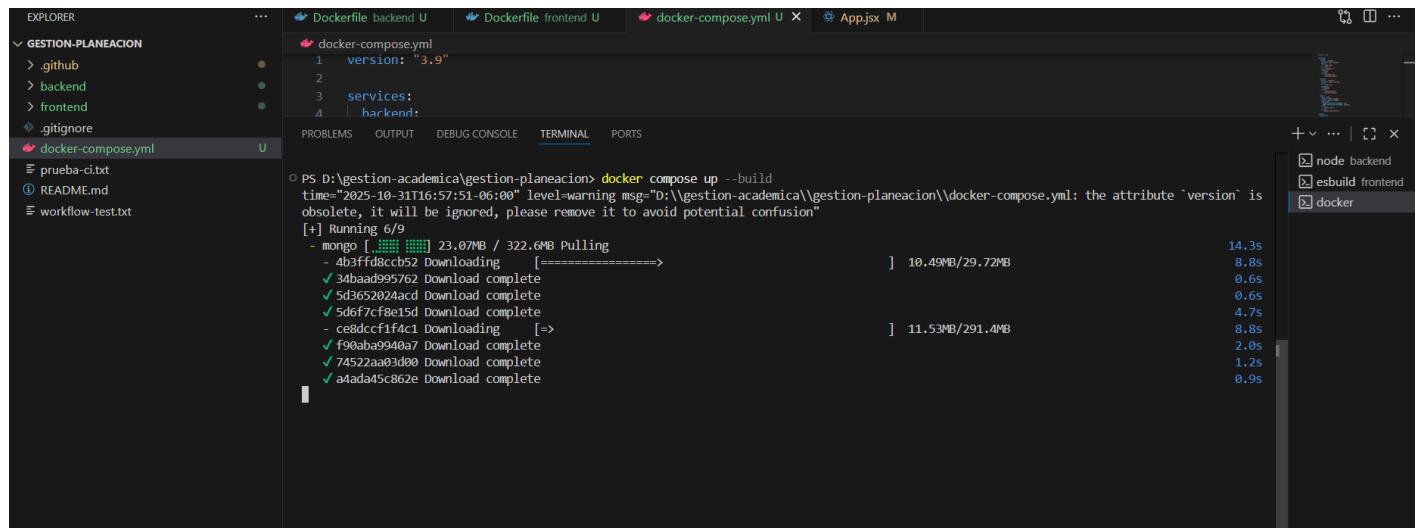
## Docker file en Front

```
frontend > 🐦 Dockerfile
1  FROM node:18
2
3  WORKDIR /app
4
5  COPY package*.json ./
6  RUN npm install
7
8  COPY . .
9
10 EXPOSE 5173
11
12 CMD ["npm", "run", "dev", "--", "--host"]
13
```

## Docker Compose.yml

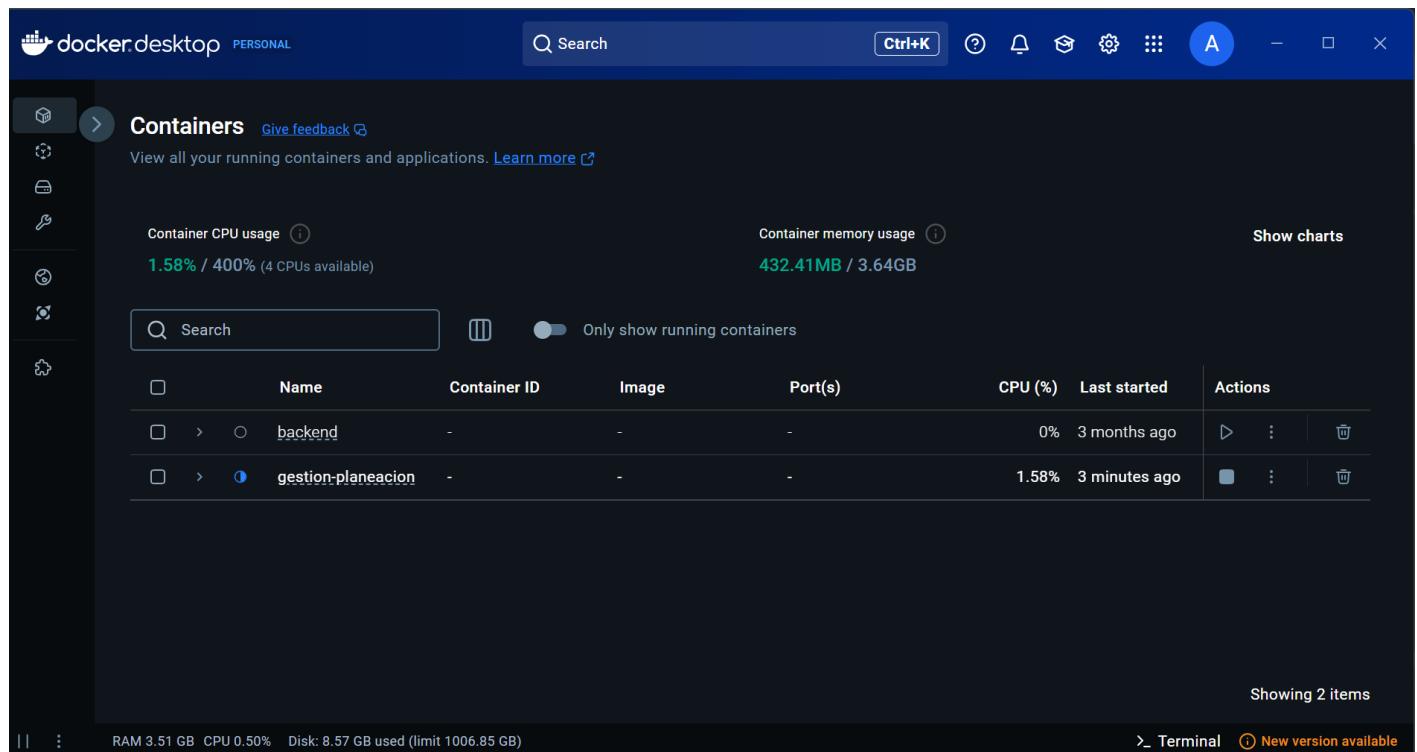
```
docker-compose.yml
1  version: "3.9"
2
3  services:
4    backend:
5      build: ./backend
6      container_name: backend
7      ports:
8        - "4000:4000"
9      env_file:
10        - ./backend/.env
11      depends_on:
12        - mongo
13      volumes:
14        - ./backend:/app
15        - /app/node_modules
16
17  frontend:
18    build: ./frontend
19    container_name: frontend
20    ports:
21      - "5173:5173"
22    depends_on:
23      - backend
24    volumes:
25      - ./frontend:/app
26      - /app/node_modules
27
28  mongo:
29    image: mongo
30    container_name: mongodb
```

## Construir Imágenes



```
PS D:\gestion-academica\gestion-planeacion> docker compose up --build
time="2025-10-31T16:57:51-06:00" level=warning msg="D:\\gestion-academica\\gestion-planeacion\\docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion"
[+] Running 6/9
- mongo [██████████] 23.07MB / 322.6MB Pulling
- 4b3ff8ccb52 Downloading [=====]
  ✓ 3abaa995762 Download complete
  ✓ 5d3652024acd Download complete
  ✓ 5d6f7fc8e15d Download complete
  - ce8dccff1f4c1 Downloading [=→]
  ✓ f90ab9940a7 Download complete
  ✓ 74522aa03d00 Download complete
  ✓ a4ada45c862e Download complete
```

## Imagen en Docker desktop



Containers [Give feedback](#)

View all your running containers and applications. [Learn more](#)

Container CPU usage [i](#) 1.58% / 400% (4 CPUs available)

Container memory usage [i](#) 432.41MB / 3.64GB

Show charts

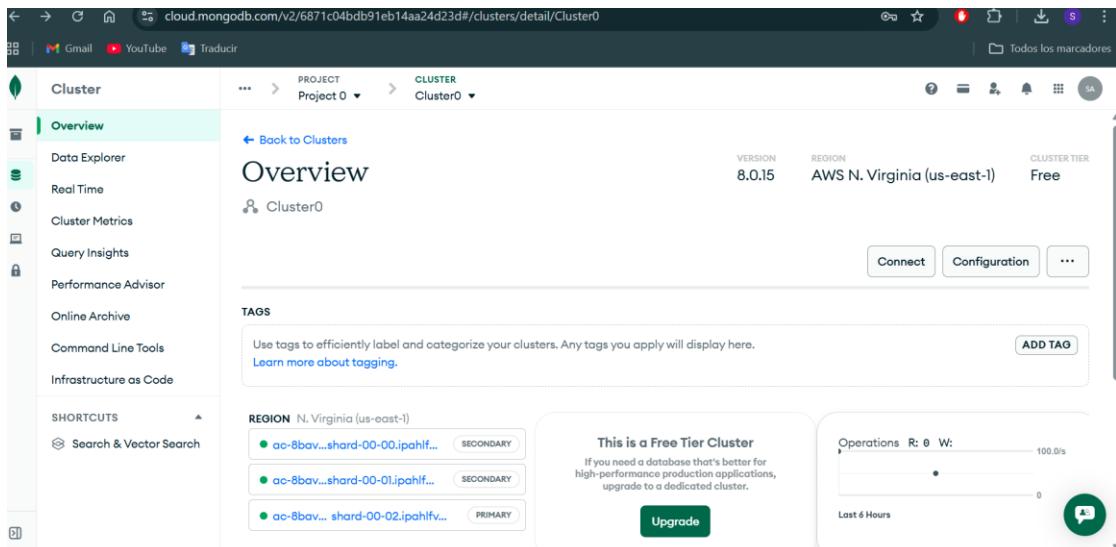
	Name	Container ID	Image	Port(s)	CPU (%)	Last started	Actions
<input type="checkbox"/>	backend	-	-	-	0%	3 months ago	<a href="#">...</a> <a href="#">Remove</a>
<input type="checkbox"/>	gestion-planeacion	-	-	-	1.58%	3 minutes ago	<a href="#">...</a> <a href="#">Remove</a>

Showing 2 items

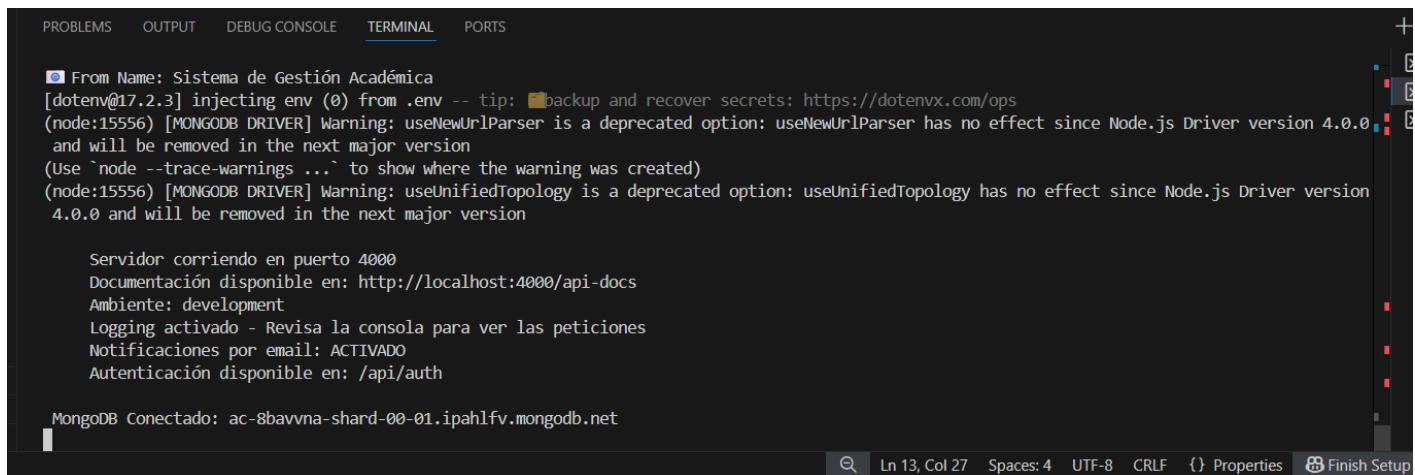
RAM 3.51 GB CPU 0.50% Disk: 8.57 GB used (limit 1006.85 GB)

[Terminal](#) [New version available](#)

## Mongo Atlas



The screenshot shows the 'Overview' page for a MongoDB cluster named 'Cluster0'. The cluster is located in the 'AWS N. Virginia (us-east-1)' region and is in the 'Free' tier. The version is 8.0.15. The 'Data Explorer' sidebar is open, showing options like 'Real Time', 'Cluster Metrics', 'Query Insights', 'Performance Advisor', 'Online Archive', 'Command Line Tools', and 'Infrastructure as Code'. The main panel displays the cluster's status, including its region, tier, and a list of shards: 'ac-8bav... shard-00-00.ipahlfv.mongodb.net' (SECONDARY), 'ac-8bav... shard-00-01.ipahlfv.mongodb.net' (SECONDARY), and 'ac-8bav... shard-00-02.ipahlfv.mongodb.net' (PRIMARY). A 'Connect' button and a 'Configuration' button are available. A 'Tags' section allows for labeling and categorizing the cluster. The 'Operations' chart shows activity over the last 6 hours.



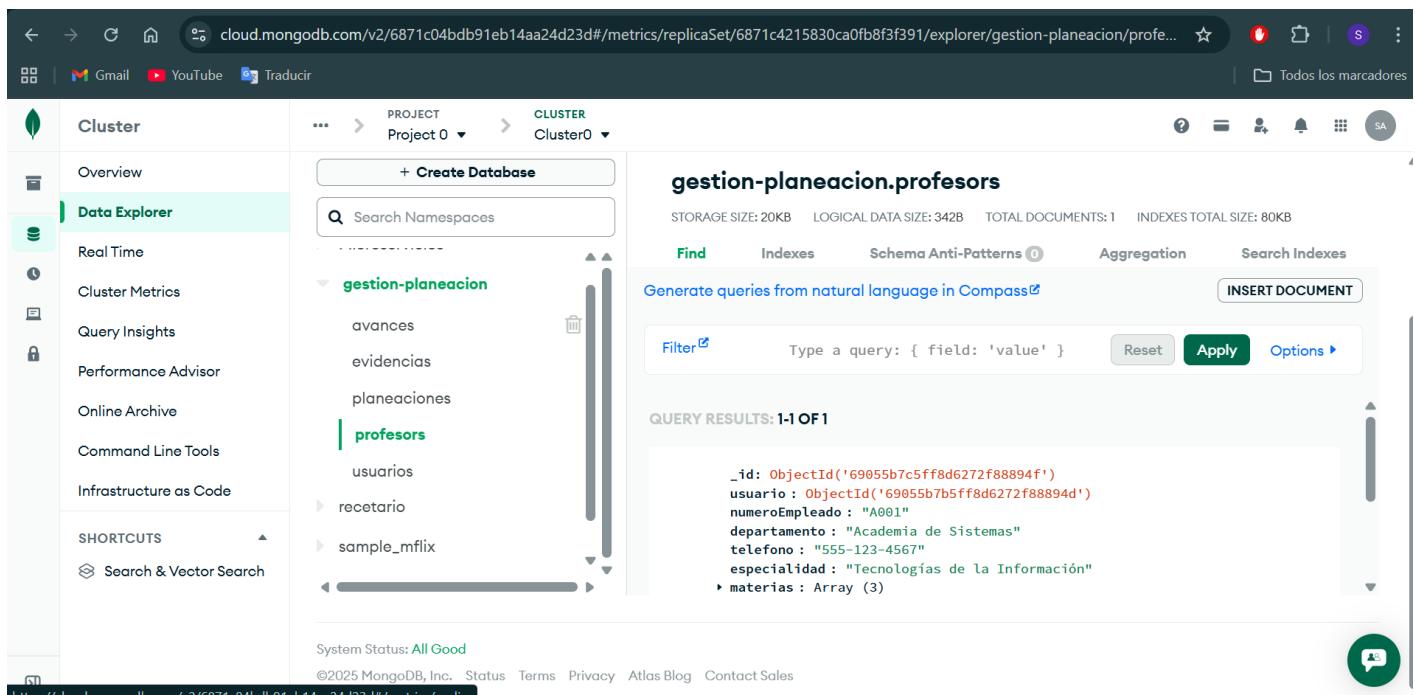
The screenshot shows a terminal window in a code editor (VS Code) with the following output:

```
From Name: Sistema de Gestión Académica
[dotenv@17.2.3] injecting env (0) from .env -- tip: ⚡ backup and recover secrets: https://dotenvx.com/ops
(node:15556) [MONGODB DRIVER] Warning: useNewUrlParser is a deprecated option: useNewUrlParser has no effect since Node.js Driver version 4.0.0 and will be removed in the next major version
(Use `node --trace-warnings ...` to show where the warning was created)
(node:15556) [MONGODB DRIVER] Warning: useUnifiedTopology is a deprecated option: useUnifiedTopology has no effect since Node.js Driver version 4.0.0 and will be removed in the next major version

Servidor corriendo en puerto 4000
Documentación disponible en: http://localhost:4000/api-docs
Ambiente: development
Logging activado - Revisa la consola para ver las peticiones
Notificaciones por email: ACTIVADO
Autenticación disponible en: /api/auth

MongoDB Conectado: ac-8bavna-shard-00-01.ipahlfv.mongodb.net
```

The terminal also shows the current file path: 'cloud.mongodb.com/v2/6871c04bdb91eb14aa24d23d#/.env'.



The screenshot shows the 'Data Explorer' page for the 'gestion-planeacion.profesores' database. The sidebar lists namespaces: 'avances', 'evidencias', 'planeaciones', 'profesores' (which is selected), 'usuarios', 'recetario', and 'sample\_mflix'. The main panel displays the 'gestion-planeacion.profesores' collection with the following document:

```
_id: ObjectId('69055b7c5ff8d6272f88894f')
usuario: ObjectId('69055b7b5ff8d6272f88894d')
numeroEmpleado: "A001"
departamento: "Academia de Sistemas"
telefono: "555-123-4567"
especialidad: "Tecnologías de la Información"
materias: Array (3)
```

# Docker Hub

## Backend

```
▶ Authenticating with existing credentials...
Login Succeeded
PS D:\gestion-academica\gestion-planeacion\backend> docker build -t andreapalomares31/proyecto-backend:1.0 -t andreapalomares31/proyecto-backend:latest .
  ▷ [+] Building 3.2s (5/10)
    => [internal] load build definition from Dockerfile
    => => transferring dockerfile: 331B
    => [internal] load metadata for docker.io/library/node:18
    => [auth] library/node:pull token for registry-1.docker.io
    => [internal] load .dockerrcignore
    => => transferring context: 2B
    => [internal] load build context
    => => transferring context: 24B
    => [1/5] FROM docker.io/library/node:18@sha256:c6ae79e38498325db67193d391e6ec1d224d96c693a8a4d943498556716d3783
    => => resolve docker.io/library/node:18@sha256:c6ae79e38498325db67193d391e6ec1d224d96c693a8a4d943498556716d3783
  ▷ Live Share
  ▷ In 15 Col 1 Spaces: 4 LITE 8 CFILE ⌂ Properties ⌂ Finish S

PS D:\gestion-academica\gestion-planeacion\backend> docker push andreapalomares31/proyecto-backend:1.0
The push refers to repository [docker.io/andreapalomares31/proyecto-backend]
b62854868f73: Pushed
3697be50c98b: Pushed
3e6b9d1a9511: Pushing [=====>] 5.243MB/48.49MB
37927ed901b1: Pushing [=====>] 4.194MB/24.02MB
79b2f47ad444: Pushing [==>] 4.194MB/64.4MB
1bfd8c8695c7: Pushed
e23f099911d6: Pushing [>] 4.194MB/211.4MB
d18507b43cb0: Pushing [=====>] 5.243MB/25.5MB
cda7f44f2bdd: Pushed
c6b30c3f1696: Pushing [=====>] 5.243MB/45.68MB
461077a72fb7: Pushed
8983147d0b2c: Pushed
32fb04f536e2: Pushing [=====>] 4.194MB/47.81MB
```

Name	Last Pushed	Contains	Visibility	Scout
andreapalomares31/proyecto-backend	less than a minute ago <small>IMAGE</small>		Public	Inactive

General	Tags	Image Management <small>BETA</small>	Collaborators	Webhooks	Settings <small>•</small>
<input type="checkbox"/>	<input type="checkbox"/> Sort by <small>Newest</small>	<input type="text"/> Filter tags	<input type="button" value="Delete"/>		
<small>TAG</small>					
<input type="checkbox"/>	<input checked="" type="radio"/> <a href="#">latest</a>		docker pull andreapalomares31/proyecto-backend:latest		
Last pushed 3 minutes by <a href="#">andreapalomares31</a>					
<input type="checkbox"/>	<small>Digest</small>	<small>OS/ARCH</small>	<small>Last pull</small>	<small>Compressed size</small> <small>446.84 MB</small>	
	<a href="#">e897f851e593</a>	linux/amd64	less than 1 day		

## Docker Hub FronEnd

```
6
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

● PS D:\gestion-academica\gestion-planeacion> cd .\frontend
○ PS D:\gestion-academica\gestion-planeacion\frontend> docker build --progress=plain -t andreapalomares31/proyecto-frontend:latest .
#0 building with "desktop-linux" instance using docker driver

#1 [internal] load build definition from Dockerfile
#1 transferring dockerfile: 190B 0.0s done
#1 DONE 0.0s

#2 [internal] load metadata for docker.io/library/node:22-alpine
#2 DONE 0.6s

#3 [internal] load .dockerignore
#3 transferring context: 2B done
#3 DONE 0.0s

#4 [1/5] FROM docker.io/library/node:22-alpine@sha256:b2358485e3e33bc3a33114d2b1bdb18cdbe4df01bd2b257198eb51beb1f026c5
#4 resolve docker.io/library/node:22-alpine@sha256:b2358485e3e33bc3a33114d2b1bdb18cdbe4df01bd2b257198eb51beb1f026c5 0.0s done
#4 DONE 0.0s

#5 [internal] load build context
|
```

```
PS D:\gestion-academica\gestion-planeacion\frontend> docker push andreapalomares31/proyecto-frontend:latest
The push refers to repository [docker.io/andreapalomares31/proyecto-frontend]
37ed258c9add: Pushed
29da8e244f0f: Pushing [=====>] 2.097MB/24.46MB
5ff8b913658c: Pushed
2d35ebdb57d9: Pushing [======>] 2.097MB/3.802MB
fc2cca81d0de: Pushing [=>] 2.097MB/51.57MB
e1acb8d81a6a: Pushing [======>] 1.261MB/1.261MB
b41199c27ba3: Pushed
8d51ba08d664: Pushed
ec9ee28cfcb8: Pushing [=>] 2.097MB/43.28MB
|
```

The screenshot shows the Docker Hub repository page for 'andreapalomares31/proyecto-frontend'. The left sidebar shows the user's profile and navigation links for repositories, collaborations, settings, and usage. The main content area displays the repository details, including the repository name, last pushed time, repository size, and a description section with fields for adding a description and a category. The 'General' tab is selected, showing the 'Tags' section which lists a single tag 'latest'. The 'Docker commands' section contains the command 'docker push andreapalomares31/proyecto-frontend:tagname'. On the right, there is an advertisement for 'buildcloud' with text about accelerating image build times and executing builds on optimally-dimensioned cloud infrastructure.

andreapalomares31 / proyecto-frontend / General

andreapalomares31/proyecto-frontend

Last pushed 2 minutes ago · Repository size: 118.6 MB · ⭐0 · ⏺0

Add a description

Add a category

General Tags Image Management BETA Collaborators Webhooks Settings

Tags

This repository contains 1 tag(s).

Tag	OS	Type	Pulled	Pushed
latest	Node.js	Image	less than 1 day	2 minutes

See all

Using 0 of 1 private repositories.

Docker commands

To push a new tag to this repository:

```
docker push andreapalomares31/proyecto-frontend:tagname
```

buildcloud

Build with Docker Build Cloud

Accelerate image build times with access to cloud-based builders and shared cache.

Docker Build Cloud executes builds on optimally-dimensioned cloud infrastructure with dedicated per-organization isolation.

## Conclusión

La aplicación de la metodología ágil Scrum en el desarrollo del sistema “*Gestión de Planeación Académica*” permitió lograr un proceso estructurado, flexible y orientado a resultados, asegurando entregables funcionales y verificados en cada sprint. El trabajo iterativo, sumado a la comunicación constante mediante reuniones diarias y la integración continua en GitHub, contribuyó a la identificación temprana de riesgos, a la solución oportuna de bloqueos y al cumplimiento de los objetivos establecidos.

El uso de herramientas y prácticas modernas como contenedores Docker, MongoDB Atlas, pipelines de CI/CD, estándares OWASP y almacenamiento seguro en la nube garantizó calidad técnica, escalabilidad y seguridad en la solución implementada. Asimismo, la gestión del backlog, el diseño modular, las pruebas automatizadas y la integración de servicios externos consolidaron un producto final robusto, mantenable y adaptable a nuevas mejoras o requerimientos futuros.

Finalmente, este proyecto demuestra que la combinación de metodologías ágiles, buenas prácticas de ingeniería de software y tecnologías modernas permite construir sistemas eficientes e innovadores en contextos académicos reales, fortaleciendo el aprendizaje práctico y preparando al equipo para desafíos profesionales en entornos productivos.

## Bibliografías

- Schwaber, K., & Sutherland, J. (2020). *The Scrum Guide: The Definitive Guide to Scrum: The Rules of the Game*. Scrum.org. <https://scrumguides.org>
- Pressman, R. S., & Maxim, B. R. (2019). *Software Engineering: A Practitioner's Approach* (9th ed.). McGraw-Hill Education.
- Sommerville, I. (2016). *Software Engineering* (10th ed.). Pearson.
- Cohn, M. (2010). *Succeeding with Agile: Software Development Using Scrum*. Addison-Wesley Professional.
- Fowler, M. (2002). Patterns of Enterprise Application Architecture. Addison-Wesley.
- Buschmann, F., Henney, K., & Schmidt, D. (2007). *Pattern-Oriented Software Architecture: On Patterns and Pattern Languages*. Wiley.