Tecnologías de la Información

Universidad Tecnológica de Tehuacán

Arquitectura de Software

DOCENTE: ING. José Miguel Carrera Pacheco

ALUMNOS: Juan Andrés Medina González

Oscar Flores Cerqueda

Axel Eduardo Sánchez Ventura

Francisco Xavier Gil Ginez

Alexis Montalvo Osorio

Fernando Vázquez Valeriano

SÉPTIMO CUATRIMESTRE

GRUPO: "A"

Actividad 4: De la Arquitectura al Prototipo Funcional

28/09/2025

Arquitectura basada en Microservicios

1. Introducción

El presente documento describe la arquitectura, componentes y despliegue del Sistema de Gestión, desarrollado bajo un enfoque de microservicios y desplegado en la nube. El proyecto fue desarrollado en equipo, tomando decisiones conjuntas en cada etapa del proceso, desde la selección de la arquitectura hasta la elección de tecnologías y la estrategia de despliegue. El objetivo principal fue implementar un prototipo funcional que validara la viabilidad de la arquitectura propuesta, integrando frontend, gateway, microservicios y base de datos en un entorno de producción real.

2. Arquitectura del Sistema

El sistema fue diseñado bajo el patrón de microservicios, decisión tomada después de analizar alternativas como un monolito o un enfoque serverless. Como equipo concluimos que la arquitectura de microservicios nos ofrecía mayor escalabilidad y flexibilidad para futuras iteraciones del sistema. Cada componente se encuentra desacoplado y cumple una función específica, lo cual facilita el mantenimiento y la integración de nuevas funcionalidades.

Componentes principales:

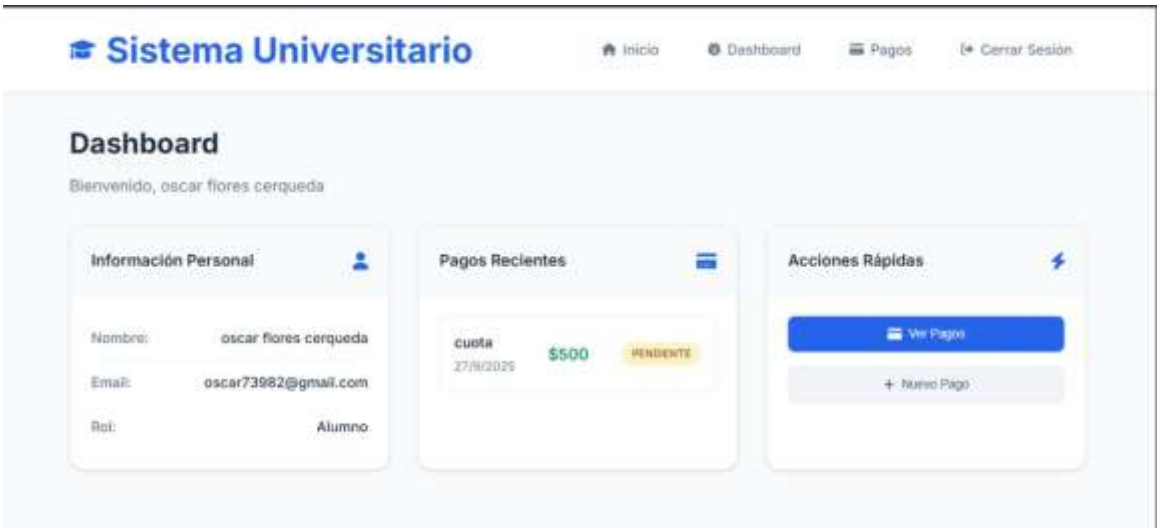
- Frontend: interfaz de usuario.
- API Gateway: punto de entrada único que gestiona y redirige solicitudes.
- Microservicios: Usuarios y Pagos.
- Base de Datos: Firebase Firestore.
- Infraestructura: Render (backend) y Vercel (frontend).

3. Frontend

Para el frontend decidimos emplear tecnologías sencillas pero efectivas que nos permitieran validar rápidamente la arquitectura:

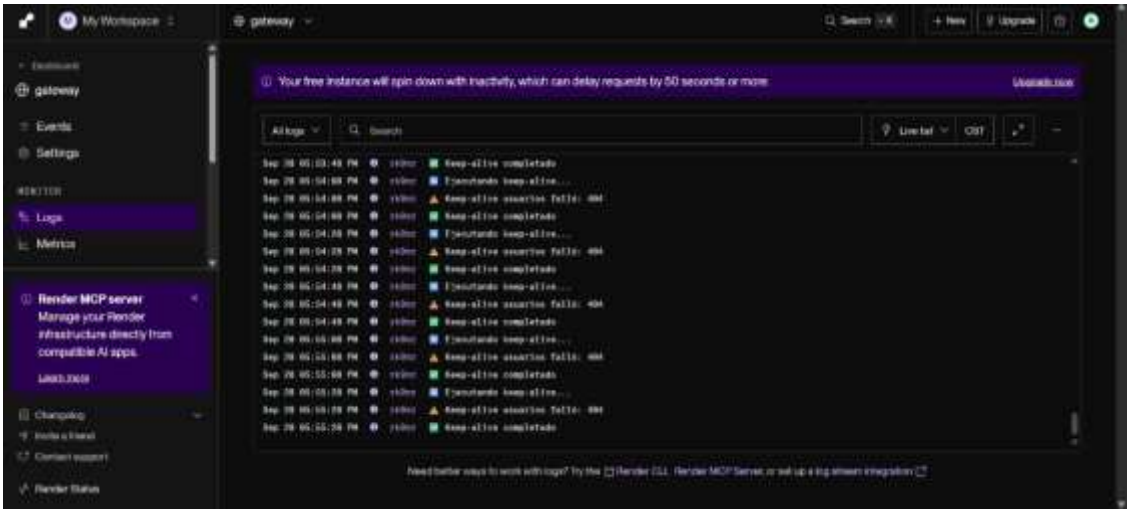
- HTML5 (estructura semántica)
- CSS3 (estilos responsivos)
- JavaScript Vanilla (lógica de interacción)
- Font Awesome y Google Fonts (UI)
- Fetch API (comunicación con backend)
- LocalStorage (persistencia de sesión)

El despliegue se realizó en Vercel y su código fuente está disponible en GitHub.



4. API Gateway

En el equipo decidimos incluir un API Gateway como punto único de entrada al sistema. Esto nos permitió centralizar la seguridad (CORS, validación de tokens) y simplificar la comunicación entre el frontend y los microservicios.

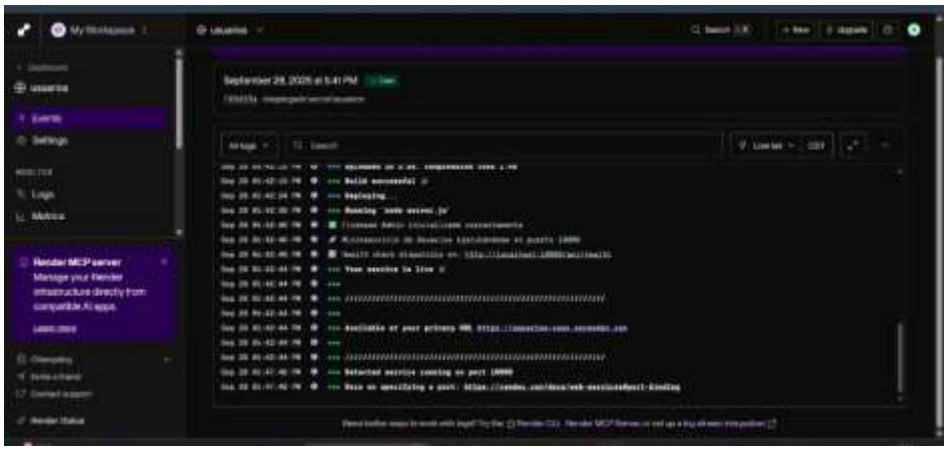


Tecnologías: Node.js, Express.js, Axios, dotenv, Render.com.

El despliegue está disponible en Render y su código fuente en GitHub.

5. Microservicio de Usuarios

Este servicio se encarga de la autenticación y administración de usuarios. Como equipo decidimos integrar Firebase Firestore como base de datos y emplear JWT para la autenticación stateless.



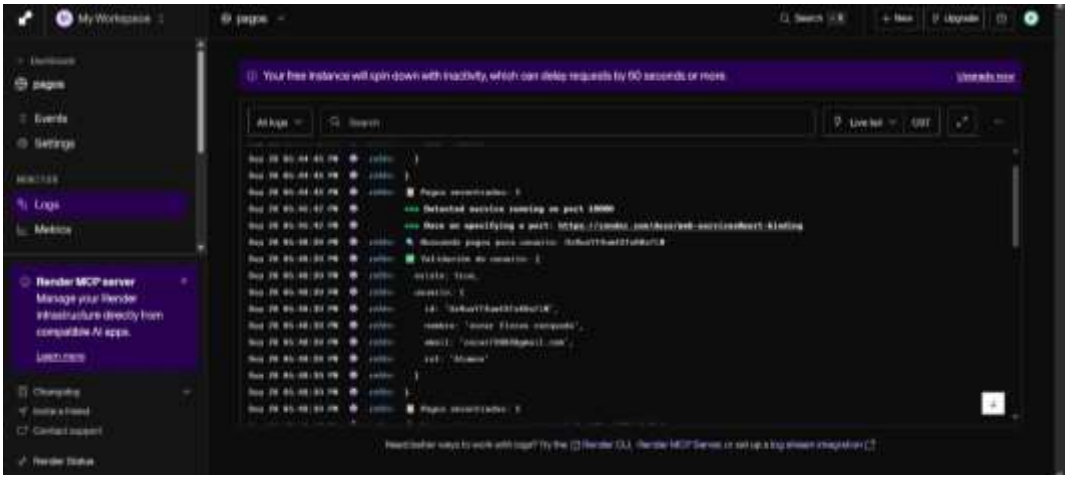
Tecnologías: Node.js, Express.js, Firebase Admin SDK, bcrypt, JWT, dotenv.

Funciones principales: registro, autenticación, validación de credenciales y manejo de tokens.

Se encuentra desplegado en Render y su código fuente está en GitHub.

6. Microservicio de Pagos

El microservicio de pagos se encarga de gestionar los registros de transacciones. En las discusiones de equipo concluimos que separar los pagos de los usuarios nos daba mayor flexibilidad para escalar y asegurar los datos de manera independiente.



Tecnologías: Node.js, Express.js, Firebase Admin SDK, dotenv.

Funciones principales: registro y consulta de pagos.

Se encuentra desplegado en Render y su código fuente está en GitHub.

7. Base de Datos

Decidimos usar Firebase Firestore como base de datos NoSQL, ya que nos ofrecía ventajas de escalabilidad automática, reglas de seguridad integradas y sincronización en tiempo real.

Definimos dos colecciones principales:

- usuarios
- pagos

8. Seguridad y Comunicación

Las decisiones de seguridad fueron discutidas en equipo y se implementaron las siguientes medidas:

- JWT Tokens: autenticación stateless.
- bcrypt: encriptación de contraseñas.
- CORS: control de accesos.
- Firebase Security Rules: reglas de acceso.
- REST API y JSON: comunicación entre microservicios.

9. Infraestructura y Despliegue

El despliegue fue realizado en la nube utilizando Render (para gateway y microservicios) y Vercel (para frontend). Como equipo discutimos y elegimos estos servicios por su facilidad de integración, plan gratuito y herramientas de monitoreo. Entre las características implementadas se incluyen:

- Servicios gratuitos con suspensión automática.

- Health checks para monitoreo.

- Variables de entorno configuradas.

- Keep-alive para garantizar disponibilidad constante.

10. Conclusiones

El proyecto validó que la arquitectura basada en microservicios es viable, segura y escalable en entornos reales. Todas las decisiones fueron tomadas en conjunto como equipo, analizando alternativas y eligiendo siempre la opción que aportara mayor valor al sistema. El prototipo cumplió con los objetivos planteados: desacoplamiento de servicios, integración con base de datos en la nube, autenticación JWT y despliegue exitoso en Render y Vercel.

Links a GitHub:

https://github.com/Oscar71k1/frontend-sistema-de-gestion

https://github.com/Oscar71k1/pagos

https://github.com/Oscar71k1/usuarios

https://github.com/Oscar71k1/microservicios