INSTITUTO TECNOLÓGICO DE COSTA RICA

UNIDAD DE COMPUTACIÓN

Campus Tecnológico Local San Carlos

Proyecto #2

Diseño de Software

Integrantes:

María Laura Alpízar Rodríguez, grupo 50

Melany Karina Bustos Cerdas, grupo 50

María Belén Córdoba Arroyo, grupo 50

Profesor:

Jonathan Solis Parajeles

Noviembre 27, 2025

Contenido

Enlace al repositorio:https://github.com/LauraAlpizar/proyecto-api-educativa

Introducción

El presente proyecto se centra en el diseño y desarrollo de una API para una plataforma educativa que busca estandarizar la gestión de cursos, materiales académicos y procesos de autenticación mediante tecnologías modernas. Según Leones Zambrano et al. (2025), las APIs se han convertido en un componente fundamental en los ecosistemas educativos digitales, ya que permiten la integración escalable de servicios y facilitan el desarrollo de aplicaciones multiplataforma.

Este proyecto surge de la necesidad de proporcionar una base tecnológica sólida para sistemas educativos que requieren gestión centralizada de cursos, autenticación segura y documentación completa de sus servicios. La propuesta incluye endpoints para gestión completa de cursos, sistema de autenticación JWT, y documentación interactiva mediante Swagger/OpenAPI, siguiendo estándares para el desarrollo de APIs web.

Análisis del problema:

En el desarrollo de software educativo, la falta de estandarización en las APIs y la carencia de documentación adecuada representan desafíos significativos. Aunque existen múltiples frameworks y herramientas, muchas implementaciones presentan limitaciones en seguridad, escalabilidad y mantenibilidad (Lercher et al, 2024).

El problema identificado consiste en que las instituciones educativas frecuentemente no cuentan con una API estandarizada que ofrezca autenticación segura, gestión completa de entidades académicas y documentación interactiva para facilitar a desarrolladores frontend y aplicaciones móviles.

Aspectos principales del problema:

- Falta de estandarización: Las APIs educativas suelen implementarse sin seguir patrones de diseño establecidos.

- Seguridad insuficiente: Múltiples sistemas carecen de mecanismos robustos de autenticación y autorización.

- Documentación inadecuada: La falta de documentación clara y actualizada dificulta la integración en sistemas.

- Escalabilidad limitada: Muchas de las arquitecturas existentes en sistemas educativos no permiten el crecimiento ordenado del sistema.

El proyecto de la API educativa busca resolver estos problemas al ofrecer:

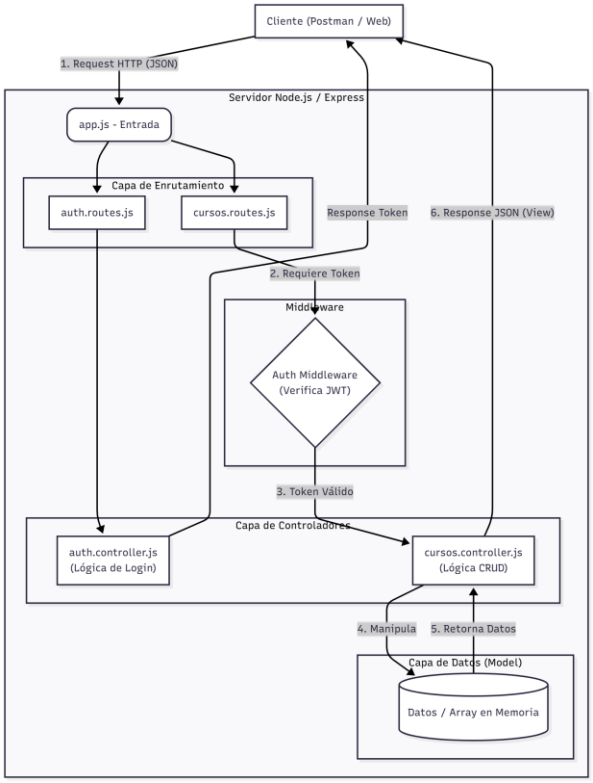
- Implementación de patrón de diseño Controller-Service-Repository.
- Autenticación JWT.
- Documentación Swagger /OpenAPI completa e interactiva.
- Estructura modular y escalable.

Solución del Problema

Para dar solución el problema planteado, se implementó una API RESTful con las siguientes características técnicas:

Arquitectura y Patrones de Diseño:

- Se implementó el patrón MVC (Model-View-Controller) para realizar una separación clara de responsabilidades entre capas.
- Middleware de autenticación para validación de tokens JWT.
- Controladores modulares: Se organiza por dominios (auth, cursos).



Características Técnicas Implementadas

Autenticación y Seguridad:

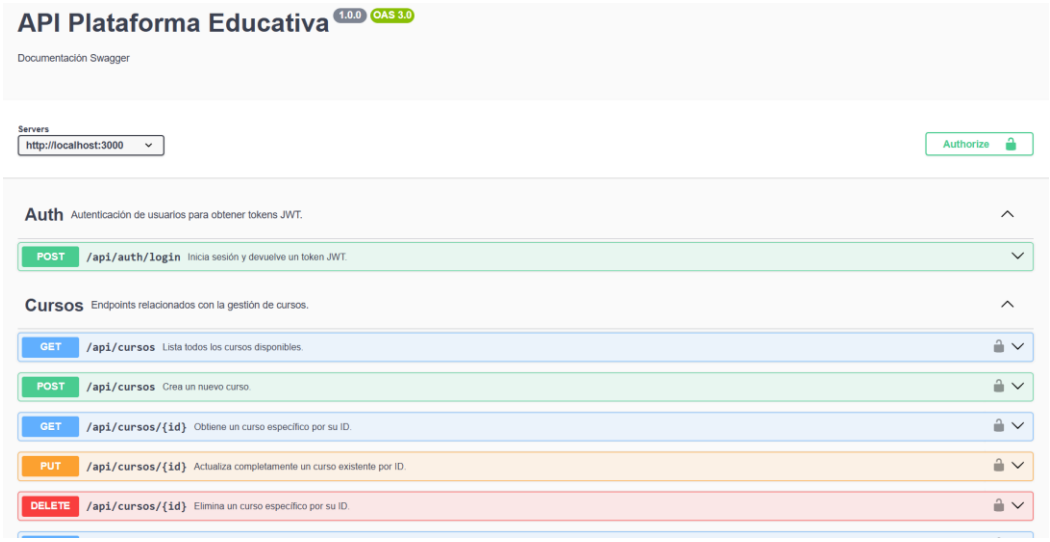
```
// Implementación JWT
const token = jwt.sign(
    { id: usuarioDemo.id, email: usuarioDemo.email },
    process.env.JWT_SECRET,
    { expiresIn: "1h" }
);
```

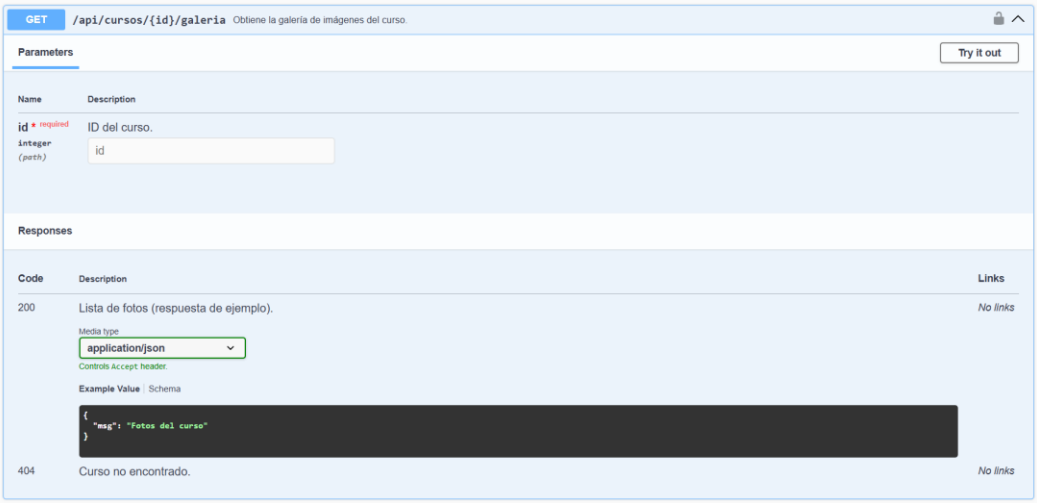
Gestión de Cursos:

- CRUD completo para entidad Curso
- Endpoints para operaciones académicas (notas, galería, anuncios)
- Validación de datos de entrada

Documentación API:

- Documentación interactiva con Swagger UI
- Especificación OpenAPI 3.0.0
- Ejemplos de request/response para todos los endpoints





Tecnologías Utilizadas:

- Node.js y Express para el servidor backend
- JWT para autenticación stateless

- Swagger /OpenAPI para documentación automática
- Bcryptjs para hashing de contraseñas

Como indica Microsoft (2024), "las APIs RESTful bien diseñadas siguen principios de uniformidad, statelessness y cacheabilidad, lo que las hace escalables y mantenibles a largo plazo".

Endpoints Implementados

- POST /api/auth/login – Autenticación
- GET /api/cursos - Listar cursos
- GET /api/cursos/:id - Obtener curso específico
- POST /api/cursos - Crear curso
- PUT /api/cursos/:id - Actualizar curso
- DELETE /api/cursos/:id - Eliminar curso
- GET /api/cursos/:id/notas - Obtener notas
- POST /api/cursos/:id/notas - Agregar nota
- GET /api/cursos/:id/galeria - Obtener galería
- POST /api/cursos/:id/galeria - Subir foto
- GET /api/cursos/:id/anuncios - Obtener anuncios

Despliegue en Azure

Como parte del requisito de exponer la API en un entorno de nube moderno, la solución fue desplegada en **Azure App Service**, una plataforma administrada que permite ejecutar aplicaciones web y APIs con altos estándares de disponibilidad, seguridad y escalabilidad. Según Microsoft (2024), App Service constituye una opción ideal para arquitecturas REST por su soporte integrado para despliegue continuo, entornos administrados y herramientas de monitoreo.

Proceso de Despliegue

1. **Creación del App Service:**
   Se configuró el recurso utilizando el runtime Node.js 20 LTS sobre sistema operativo Linux, siguiendo las recomendaciones de Azure para APIs ligeras y de alto rendimiento.
2. **Configuración Crítica del Startup Command:**
   Se estableció el comando de inicio: node src/app.js

Esta configuración garantiza que Azure identifique correctamente el punto de entrada del proyecto y evita errores relacionados con rutas internas del entorno.

3. **Despliegue Continuo (CI/CD) y Control de Versiones:**
    a. Se implementó un flujo automatizado utilizando GitHub Actions, definido en el archivo: .github/workflows/deploy.yml
    b. El despliegue se ejecuta automáticamente con cada actualización aprobada en la rama principal del repositorio.
    c. Dado que el trabajo es colaborativo, se empleó una bifurcación (*fork*) del repositorio central para aislar el desarrollo grupal, manteniendo la estabilidad del código antes de su despliegue.
    d. De esta forma, Azure solo ejecuta y publica versiones comprobadas, cumpliendo con buenas prácticas de control de versiones.

Políticas de Seguridad y Configuraciones Implementadas

Para fortalecer la integridad y disponibilidad del servicio se configuraron las siguientes políticas:

1. **HTTPS Only:**
   Activado para garantizar que todas las comunicaciones entre clientes y el servidor se realicen mediante canales encriptados.
2. **Health Check:**
   Configurado para que Azure supervise continuamente el estado real de la API, permitiendo detectar fallos y gestionar automáticamente instancias problemáticas.
3. **CORS:**
   Ajustado para permitir solicitudes desde aplicaciones frontend en desarrollo, manteniendo control sobre los orígenes permitidos.

Resultado del Despliegue

La API quedó publicada mediante un dominio generado por Azure App Service, accesible globalmente.
La documentación Swagger UI puede consultarse en la ruta: https://api-educativa-plataforma-fxdefhfrb7dedzhm.canadacentral-01.azurewebsites.net/api/docs/#/

Ventajas del Uso de Azure App Service

El despliegue en Azure aporta beneficios significativos:

- **Escalabilidad automática** ante mayor demanda.
- **Alta disponibilidad**, propia de la infraestructura de Azure.
- **CI/CD integrado**, facilitando ciclos de actualización seguros.
- **Políticas robustas de seguridad**, como HTTPS Only, monitoreo e integridad del servicio.

Análisis de resultados:

Tarea/Requerimiento	Estado	Observaciones
Implementación de 10 endpoints	Completo	11 endpoints implementados.
Autenticación JWT	Completo	Middleware de validación y generación de tokens.
Documentación Swagger	Completo	100% de endpoints documentados con ejemplos.
Patrón de diseño	Completo	Implementado patrón MVC con separación de responsabilidades.
Despliegue en Azure	Completo	Configuración lista, incluye políticas de seguridad HTTPS Only y Health Check.
Estructura de commits	Completo	Seguimiento con nomenclatura convencional.

Conclusiones

- La API implementada proporciona una base sólida para sistemas educativos, ofreciendo autenticación segura y gestión completa de cursos mediante endpoints RESTful estandarizados.
- La implementación del patrón MVC asegura una arquitectura mantenible y escalable, facilitando la adición de nuevas funcionalidades y el mantenimiento del código existente.
- La documentación Swagger/OpenAPI integrada representa una ayuda al desarrollador, ya que permite reducir el tiempo de integración y facilita el entendimiento de los servicios disponibles.
- El sistema de autenticación JWT implementado proporciona seguridad robusta mientras mantiene la escalabilidad.
- El despliegue en Azure App Service garantiza que la API opere bajo estándares de escalabilidad, disponibilidad y seguridad de clase mundial. La implementación de políticas como HTTPS Only, Health Check y flujos CI/CD permite un servicio confiable y alineado con prácticas profesionales en entornos productivos.

Recomendaciones

Mejoras Técnicas:

- Implementar rate limiting para prevenir abuso de uso de la API y asegurar calidad de servicio.
- Agregar logging estructurado para mejor monitoreo y troubleshooting.

Mejoras de Funcionalidad:

- Expandir modelo de datos incluyendo entidades para estudiantes, profesores y matrículas.
- Implementar paginación para endpoints que puedan retornar grandes volúmenes de datos.
- Agregar filtros y búsqueda para mejorar la usabilidad de los endpoints de listado.

Seguridad y Performance:

- Agregar compresión para optimizar transferencia de datos.
- Implementar caching para reducir carga en endpoints frecuentemente accedidos.

Referencias

Leones Zambrano, W. P., Macias Bazurto, L. M. & Macias Bazurto, G. M. (2025). *El desarrollo de interfaces de programación de aplicaciones (APIs) dinamiza el acceso a contenidos en plataformas de educación virtual*. LATAM Revista Latinoamericana de Ciencias Sociales y Humanidades, 6(2), 3039–3047. https://doi.org/10.56712/latam.v6i2.3816

Microsoft Azure. (2024). API Design Best Practices. Microsoft Documentation. https://docs.microsoft.com/en-us/azure/architecture/best-practices/api-design

Lercher, A., Glock, J., Macho, C., & Pinzger, M. (2024). *Microservice API Evolution in Practice: A Study on Strategies and Challenges*. *Journal of Systems and Software*, 215, 112110. https://doi.org/10.1016/j.jss.2024.112110