Arquitectura de Software: Plataforma de Aprendizaje para Todos

Información General	
Nombre del proyecto	**Aprendizaje Para Todos (APT)**
Fecha de preparación	15 de septiembre de 2024
Fecha de cierre	27 de noviembre de 2024
Usuarios	Alumnos y Profesores
Preparado por	Grupo 1
Autorizado por	Alvaro Mamani Aliaga

1. Introducción:

1.1. Propósito

Este documento proporciona una visión arquitectónuca completa de la Plataforma de Aprendizaje para Todos. Su objetivo es evidenciar y comunicar las decisiones arquitectónicas significativas que se van a tomar en el sistema.

1.2. Alcance

Este documento describe la arquitectura de la plataforma tanto en la parte front-endm back-end como en la parte de la base de datos e infrectructurea en la nube.

1.3. Definiciones, Acrónimos y Abreviaturas

- API: Interfaz de Programación de Aplicaciones
- GCP: Google Cloud Platform
- NoSQL: No solo SQL
- UI: Interfaz de usuario
- UX: Experiencia de Usuario

2. Representación Arquitectónica

La Plataforma de Aprendizaje para Todos sigue una arquitectura de tres capas:

1. Capa de presentación: Maneja la interfaz de usuario y las interacciones del usuario
2. Capa de Aplicación: Contiene la lógica de negocio y procesa las solicitudes del usuario.
3. Capa de Datos: Gestiona el almacenamiento y recuperación de datos

3. Objetivos y Restricciones Arquitectónicas

3.1. Objetivos

- Seguridad: Implementar autenticación de usuarios y control de acceso basado en roles
- Usabilidad: Crear una interfaz intuitivas y fácil de usar para todos los tipos de usuarios
- Escalabilidad: Diseñar el sistema para manejar un número creciente de usuarios y recursos educativos

3.2. Restricciones

- Base de datos: Usar MongoDB, una base de datos NoSQL, para un almacenamiento eficiente de datos no estructurados
- APIs: Utilizar APIs que no tengan costo de Google Cloud Platform o AWS para funcionalidades adicionales

- Despliegue en la Nube: Diseñar para el despliegue en la nube como en GCP o AWS

4. Vistas de Caso de Uso

4.1. Actores

- Alumnos: Accesden a los libros y participan en foros
- Profesores: Publican libros y gestionan el acceso de los alumnos
- Administradores: Gestionan usuarios, aprueban contenido y supervisa la plataforma

4.2. Casos de Uso Clave

1. Registrar Material Educativo (CUS-1)
2. Registrar Libros Recibidos (CUS-2)
3. Consultar Consultar Libros (CUS-3)
4. Publicar Contenido (CUS-4)
5. Registrar Preguntas y Temas de Discusión (CUS-5)
6. Registrar Respuestas y Comentarios (CUS-6)
7. Mantener Información del Foro Educativo (CUS-7)
8. Registrar Usuario (CUS-8)
9. Actualización de Perfil (CUS-9)

5. Vista Lógica

5.1. Descripción General

- Gestión de Usuarios
- Gestión de Contenidos
- Gestión de Foros
- Búsquedas y Descubrimiento
- Autenticación y Autorización

5.2. Paquetes de Diseño Arquitectónicamente Significativos

- Gestión de Usuarios: Maneja el registro de usuarios, gestión de perfiles y asignación de roles
- Gestión de Contenido: Gestiona la creación, aprobación y organización de los libros publicados
- Gestión de Foros: Facilita la creación y moderación de los foros de discusión
- Búsqueda y descubrimiento: Permite a los usuarios encontrar libros y discusiones relevantes.
- Autenticación y Autorización: Asegura el acceso seguro a la plataforma y gestiona los permisos de usuario

6. Vista de Procesos

El sistema sigue estos procesos principales:

- Registro y autentificación de usuarios
- Publicación y aprobación de contenido
- Interacción en foros
- Descubrimiento y acceso a recursos

7. Vista de Despliegue

El sistema se desplegará en Google Cloud Platform (GCP) con los siguientes componentes.

- Front-end: Aplicación React alojada en GCP App Engine
- Back-end: Node.js con Express.js, desplegado en GCP Compute Engine
- Base de Datos: Instancia MongoDB en GCP Cloud SQL
- Almacenamiento de Archivos: GCP Cloud Storage para recursos educativos
- APIs: Integración con APIs de GCP para funcionalidades adicionales

8. Vista de Implementación

8.1. Descripción General

El sistema se implementa utilziando las siguientes tecnologías

- Front-end: React, Visual Studio Code
- Back-end: Node.js, Express.js
- Base de Datos: MongoDB
- Plataforma en la Nube: Google Cloud Platform (GCP)
- APIs Adicionales: Firebase, APIs de GCP

8.2. Capas

- Capa de Presentación: Componentes React y bibliotecas de UI
- Capa de Aplicación: Rutas y controladores de Expreses.js
- Capa de Acceso a Datos: Consultas MongoDB y modelos de datos

9. Vista de datos

El modelo de datos incluye estas identidades:

- Usuarios (Alumnos, profesores, administradores)
- Recursos Educativos (Libros, artículos)
- Foro (Temas, preguntas, respuestas)

Los datos se almacenán en MongoDB, utilizando su estructura flexible basada en documentos para un almacenamiento y recuperación eficientes de diversos tipos de contenido

10. Tamaño y rendimiento

El sistema está diseñado para manejar:

- Usuarios concurrentes:
- Almacenamiento de datos:
- Tiempo de respuesta:

11. Calidad

El sistema se adhiere a los siguientes atributos de calidad:

- Usabilidad: Diseño UI/UX intuitivo, cumplimiento de accesibilidad
- Eficiencia: Consultas de base de datos optimizadas, renderizado eficiente del front-end
- Confiabilidad: Copias de seguridad de datos, manejo de errores y registro
- Mantenibilidad: Estructura de código modular, control de versiones con GitHub
- Seguridad: Autentificación de usuarios, encriptación de datos, comunicaciones seguras de API

12. Riesgos Técnicos y Estrategias de mitigación

A. Riesgo: Pérdida de datos

   Mitigación: Copias de seguridad regulares, almacenamiento redundante en GCP

B. Riesgo: Problemas de rendimiento con aumento de carga de usuarios

   Mitigación: Implementar caché, indexación de base de datos y balanceo de carga

C. Riesgo: Vulnerabilidades de seguridad

   Mitigación: Auditorías de seguridad regulares, mantener todas las bibliotecas y dependencias actualizadas

13. Herramientas y tecnologías

- Front-end: React, Visual Studio Code
- Back-end: Node.js, Express.js
- Base de datos: MongoDB

14. Gráficos

Este gráfico muestra una visión horizontal del sistema y simplificada de la arquitectura, haciendo enfasis en cómo la API RESTful funciona como intermediario entre el cliente React, el servidor Node.js y la base de datos MongoDB. Aquí se pone énfasis en la comunicación fluida entre los componentes y el uso de la nube de Google para alojar el sistema.

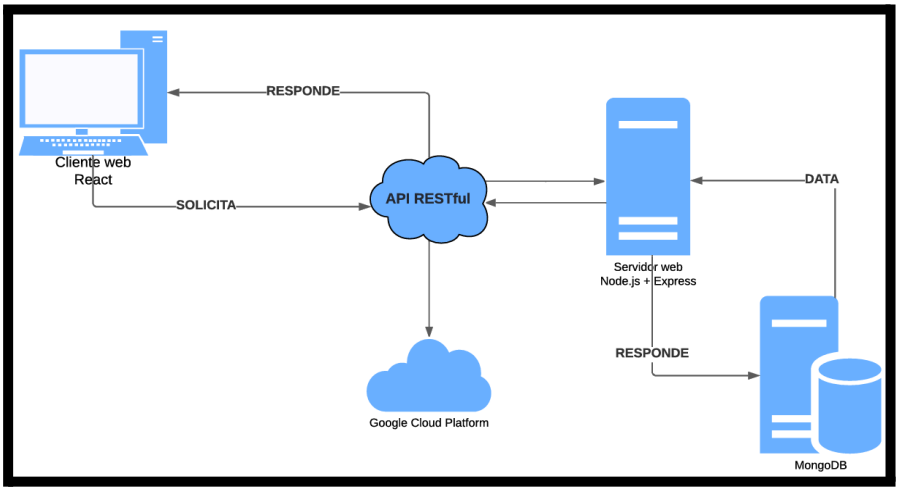

Gráfico 1 *(Hecho en lucidchart: Link de origen)*

En este gráfico se muestra la arquitectura de la plataforma en diferentes capas organizadas verticalmente, desde el cliente hasta la base de datos. Las flechas representan el flujo de solicitudes y respuestas entre estas capas. Es ideal para entender la relación entre el cliente, servidor y base de datos en términos de capas funcionales.

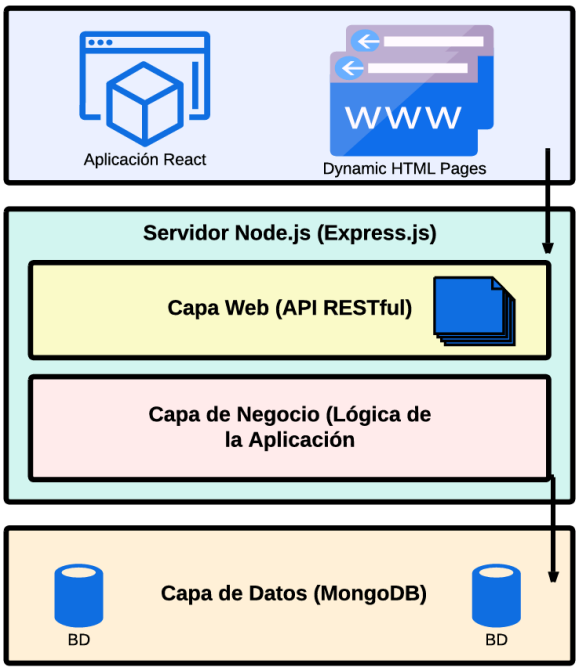

Gráfico 2 *(Hecho en Lucidchart: Link de origen)*