UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA
UNIDAD DE PRÁCTICAS DE INGENIERÍA Y EPS
Ing. Floriza Avila

INFORME 4

MANUAL TECNICO

Josué Samuel de la Cruz Medina - 202247844
Daniel Estuardo Salvatierra Macajola – 202202768
19/09/2024

Introducción

Este manual técnico proporciona una visión detallada de la arquitectura y la implementación de la página web realizada con React y MongoDB. La aplicación utiliza diversas validaciones para poder funcionar de una manera segura y rápida para el usuario.

Arquitectura del Sistema

El sistema está diseñado para gestionar usuarios, publicaciones, cursos, catedráticos y comentarios a través de una interfaz simple y fácil de usar. Utiliza la base de datos Mongo DB para manejar eficientemente las operaciones relacionadas con estos elementos.

Requisitos del Sistema

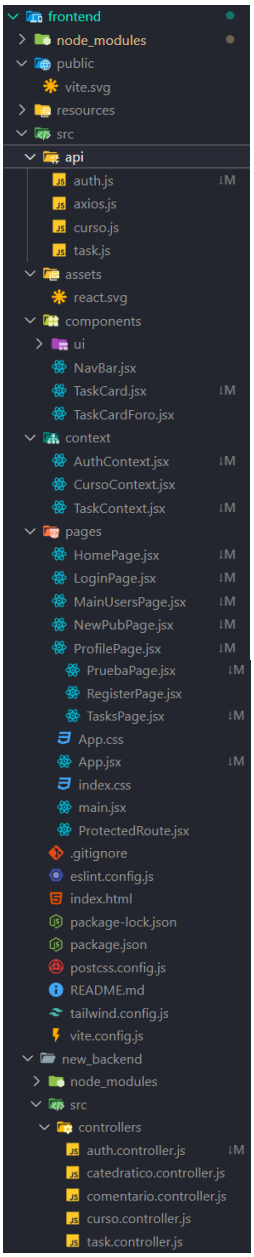
Software:

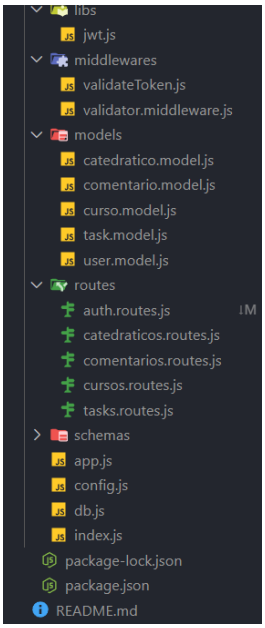
- **Backend**: Node.js con Express
- **Base de datos**: MongoDB
- **Frontend**: React
- **Control de versiones**: Git
- **Autenticación**: JWT (JSON Web Tokens)
- **ORM/ODM**: Mongoose (para la interacción con MongoDB)

Herramientas:

- **Editor de código**: Visual Studio Code o cualquier IDE que soporte JavaScript
- **Postman:** para pruebas de API
- **MongoDB Compass:** opcional, para visualización de datos

Estructura del Proyecto





Frontend:

- **api**: Contiene las funciones para realizar peticiones a la API desde el frontend (puede incluir archivos como auth.js o courses.js para manejar las llamadas a los endpoints correspondientes).
- **assets**: Aquí se almacenan archivos estáticos como imágenes, íconos, fuentes y archivos CSS.

- **components**: Componentes reutilizables del frontend, como formularios, botones, menús.
- **context**: Se incluyen los proveedores de contexto para la gestión del estado global (como AuthContext para la autenticación).
- **pages**: Contiene las diferentes páginas de la aplicación, como LoginPage.js, CoursePage.js, etc.

Backend:

- **controllers**: Aquí se almacena la lógica de cada recurso. Por ejemplo, authController.js puede manejar la lógica de autenticación, mientras que courseController.js se encarga de la lógica de los cursos.
- **libs**: Para almacenar funciones auxiliares o utilidades que no están relacionadas directamente con el modelo o controlador. Por ejemplo, aquí podrías tener una función para enviar correos electrónicos.
- **middlewares**: Aquí se colocan los middlewares, que son funciones que se ejecutan entre la petición del cliente y la respuesta del servidor. Un ejemplo sería un middleware de autenticación.
- **models**: Contiene los esquemas de datos de MongoDB. Por ejemplo, User.js, Course.js y Post.js representan los esquemas para usuarios, cursos y publicaciones.
- **routes**: Define las rutas o endpoints del servidor (por ejemplo, /login, /register, /courses).
- **schemas**: Aquí se pueden incluir los esquemas de validación para las peticiones que recibe el backend (como validaciones de los datos enviados en el registro de usuarios).

4. Descripción de las Funcionalidades

A. Registro e Inicio de Sesión de Usuarios

El sistema permite que los usuarios se registren proporcionando carnet, nombres, apellidos, correo y una contraseña. Una vez registrados, pueden iniciar sesión usando las credenciales que proporcionaron.

- **Registro**:
    - Ruta: /register
    - Método: POST
    - Campos:
        - email: String
        - password: String
        - carnet: int
        - nombres: string
        - apellidos: string
    - Validaciones:
        - El carnet debe ser único.
        - La contraseña debe tener al menos 6 caracteres.
- **Inicio de Sesión**:
    - Ruta: /login
    - Método: POST
    - Autenticación usando JWT.
    - Devuelve un token JWT para autenticar futuras solicitudes.

B. Gestión de Cursos y Profesores

- **Cursos**:
    - Ruta: /cursos
    - Método: GET
    - Información sobre todos los cursos disponibles, almacenados en la colección cursos.
- **Profesores**:

- Ruta: /catedraticos
- Método: GET
- Muestra la lista de profesores registrados en la base de datos en la colección catedraticos.

C. Publicaciones y Comentarios

- **Publicaciones**:
  - Ruta: /task
  - Método: POST
  - Los usuarios pueden crear publicaciones sobre un curso o profesor específico.
  - Campos:
    - userId: ObjectId del usuario que crea la publicación.
    - Curso o profesor: ObjectId del curso o profesor.
    - Mensaje: Texto de la publicación.
- **Comentarios**:
  - Ruta: /comentarios
  - Método: POST
  - Los usuarios pueden comentar las publicaciones de otros usuarios.
  - Campos:
    - postId: ObjectId de la publicación.
    - userId: ObjectId del usuario que realiza el comentario.
    - mensaje: Texto del comentario.