UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA
ESTRUCTURA DE DATOS
CATEDRÁTICO: ING. NEFTALÍ CALDERÓN
TUTOR ACADÉMICO: RODRIGO PORÓN

MANUAL TÉCNICO

JAVIER ANDRÉS VELÁSQUEZ BONILLA
CARNÉ: 202307775
SECCIÓN: E

GUATEMALA, 27 DE ABRIL DEL 2,024

INTRODUCCIÓN

La aplicación web de foro social es una plataforma interactiva desarrollada en React que permite a los usuarios crear y visualizar posts. Esta aplicación es un espacio donde los usuarios pueden compartir ideas, discutir temas de interés y interactuar con otros miembros de la comunidad.

El sistema se basa en una arquitectura de componentes, un enfoque que permite una mayor reutilización de código y una mejor organización del código fuente. La aplicación se compone de varios componentes, cada uno de los cuales tiene una responsabilidad específica y se encarga de una parte específica de la funcionalidad de la aplicación

OBJETIVOS

1. GENERAL

El propósito principal de este proyecto es familiarizar al estudiante con el lenguaje de programación JavaScript y la biblioteca React. Se busca que los estudiantes no solo comprendan los conceptos básicos de programación, sino que también puedan aplicarlos de manera práctica en un entorno de desarrollo real. A través de este proyecto, los estudiantes tendrán la oportunidad de elaborar la lógica necesaria para presentar una solución efectiva a la problemática planteada, que es la creación y gestión de un foro social.

2. ESPECÍFICOS

- Utilizar JavaScript y React como herramientas principales para el desarrollo de software, lo que incluye la implementación de conceptos de programación orientada a componentes y el manejo del estado y ciclo de vida de los componentes.

- Construir una aplicación con interfaz gráfica que permita una interacción intuitiva con el usuario, utilizando componentes de React y la biblioteca Next UI para la creación de la interfaz de usuario

ALCANCES DEL SISTEMA

El sistema de gestión de foros sociales implementado en React tiene como objetivo principal proporcionar una plataforma interactiva y fácil de usar para la creación y visualización de posts. Este documento proporciona una explicación detallada del código fuente, desglosando cada componente, funcionalidad y lógica de programación utilizada en el proyecto. Su finalidad es ofrecer instrucciones claras y precisas que faciliten futuras modificaciones, asegurando la integridad y eficiencia del sistema a lo largo del tiempo.

El propósito fundamental de este manual técnico es dotar al equipo de desarrollo de los recursos necesarios para comprender, mantener y mejorar el sistema de gestión de foros sociales. A través de una descripción exhaustiva del código fuente, se busca proporcionar una base sólida para la gestión, expansión y evolución efectiva del software. Además, este manual busca establecer un marco de referencia que permita garantizar la coherencia y la calidad del sistema en todas sus etapas de desarrollo y mantenimiento.

El sistema tiene el alcance de permitir a los usuarios crear nuevos posts, visualizar posts existentes y "likear" o "dislikear" posts. Cada post incluye detalles como el nombre del usuario, el email, el texto del post, la categoría y la imagen. El sistema también proporciona una interfaz de administración para la gestión de los posts.
Este sistema está diseñado para ser escalable y fácilmente modificable, permitiendo la adición de nuevas funcionalidades y mejoras en el futuro. Con este manual técnico, los desarrolladores tendrán una guía completa para entender y expandir el sistema según sea necesario.

ESPECIFICACIÓN TÉCNICA

- REQUISITOS DE HARDWARE ○ El equipo que se utilizó cuenta con las siguientes características: - Procesador Ryzen 5 5500U de 6 núcleos, 2.10GHz - 8GB de RAM DDR4 a 3200Mhz - 256GB de almacenamiento SSD
- REQUISITOS DE SOFTWARE ○ El equipo que se utilizó cuenta con las siguientes características: - Sistema Operativo Windows 10 Home de 10 Bits

DESCRIPCIÓN DE LA SOLUCIÓN

La solución implementada es una aplicación web de foro social desarrollada en React. La aplicación permite a los usuarios ver posts existentes y crear nuevos posts. La aplicación se compone principalmente de dos componentes: AdminPage y Post. El componente AdminPage es responsable de la gestión de los posts. Utiliza el hook useState para mantener un estado local de los posts. Al inicio, este estado se inicializa como un array vacío.

El componente AdminPage también utiliza el hook useEffect para llamar a la función getPosts cuando el componente se monta. La función getPosts hace una petición GET a la API en "http://localhost:3000/api/posts/get" para obtener los posts existentes. Los posts obtenidos se almacenan en el estado local posts.

Además, el componente AdminPage proporciona un botón para navegar a la página de creación de nuevos posts. Cuando se hace clic en este botón, se llama a la función handleNewPost, que utiliza el hook useNavigate para redirigir al usuario a la página '/foro/nuevo'.

Finalmente, el componente AdminPage renderiza una lista de componentes Post, uno para cada post en el estado posts. Cada componente Post recibe los datos del post correspondiente como props.

El componente Post es responsable de la visualización de un post individual. Recibe los datos del post como props y los utiliza para renderizar los detalles del post, incluyendo el nombre del usuario, el email, el texto del post, la categoría y la imagen. También proporciona un botón que permite a los usuarios "likear" o "dislikear" el post.

La solución implementada es una aplicación web de publicación de posts. La aplicación permite a los usuarios crear, leer, actualizar y eliminar posts. La aplicación está construida con Node.js y Express en el backend, y utiliza una estructura de datos en memoria para almacenar los posts.

Backend:
El backend de la aplicación se basa en Node.js y Express. Se han definido varias rutas para manejar las operaciones CRUD (Crear, Leer, Actualizar, Eliminar) en los posts:

- GET /api/posts: Devuelve todos los posts.
- GET /api/posts/:id: Devuelve un post específico por su ID.
- POST /api/create: Crea un nuevo post.
- PUT /api/posts/:id: Actualiza un post específico por su ID.
- DELETE /api/posts/:id: Elimina un post específico por su ID.
-

Además, se ha implementado una funcionalidad de autenticación que permite a los usuarios iniciar sesión en la aplicación. Los usuarios se autentican proporcionando su código y contraseña, y la aplicación verifica estos detalles antes de permitir el acceso.

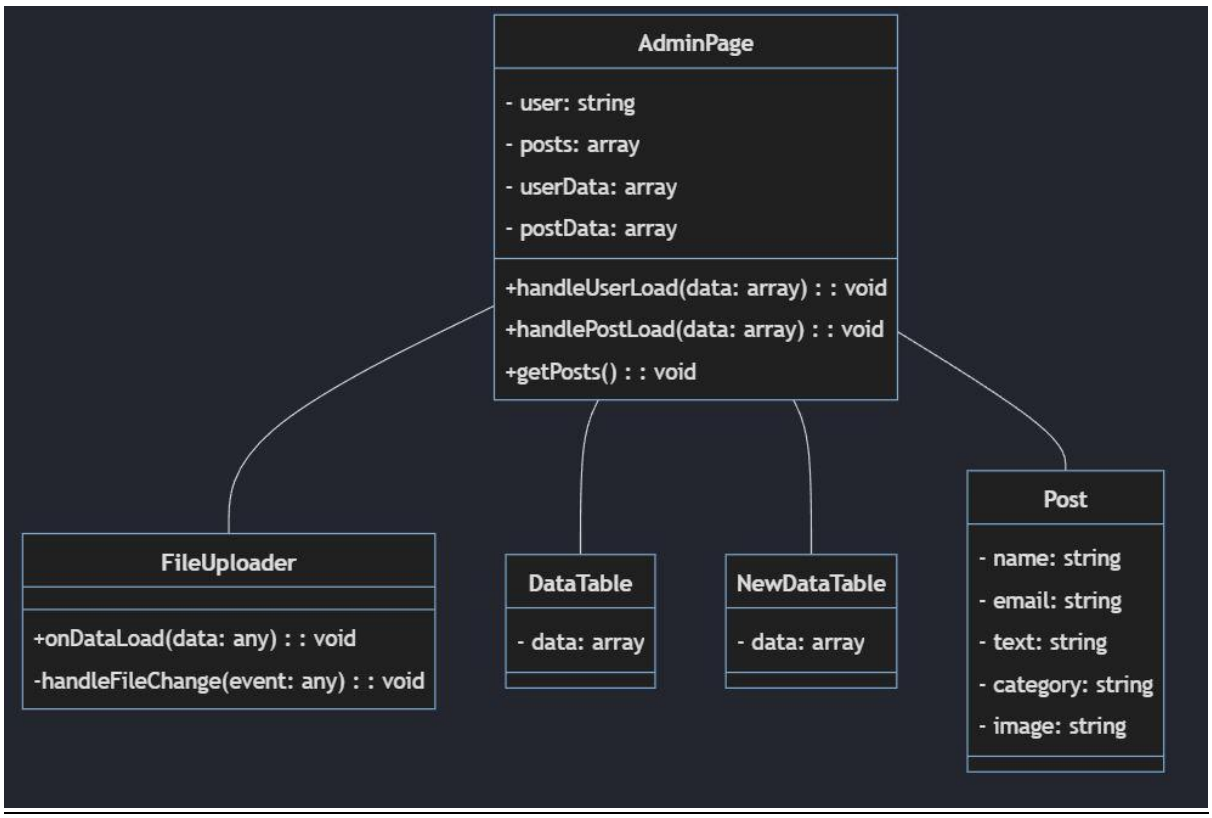
Datos:
Los posts y los detalles del usuario se almacenan en estructuras de datos en memoria. Cada post es un objeto con propiedades como ID, descripción, código de usuario, categoría, imagen, anonimato, marca de tiempo, ID de usuario, nombre, correo electrónico y texto.
Funcionamiento:
Cuando un usuario envía una solicitud a la aplicación, Express maneja la solicitud y la dirige a la ruta y al controlador adecuados. El controlador luego realiza la operación requerida (por ejemplo, crear un nuevo post o recuperar posts existentes) y envía una respuesta al cliente.

La aplicación está diseñada para ser fácil de usar y eficiente, proporcionando a los usuarios una forma rápida y sencilla de compartir y leer posts.

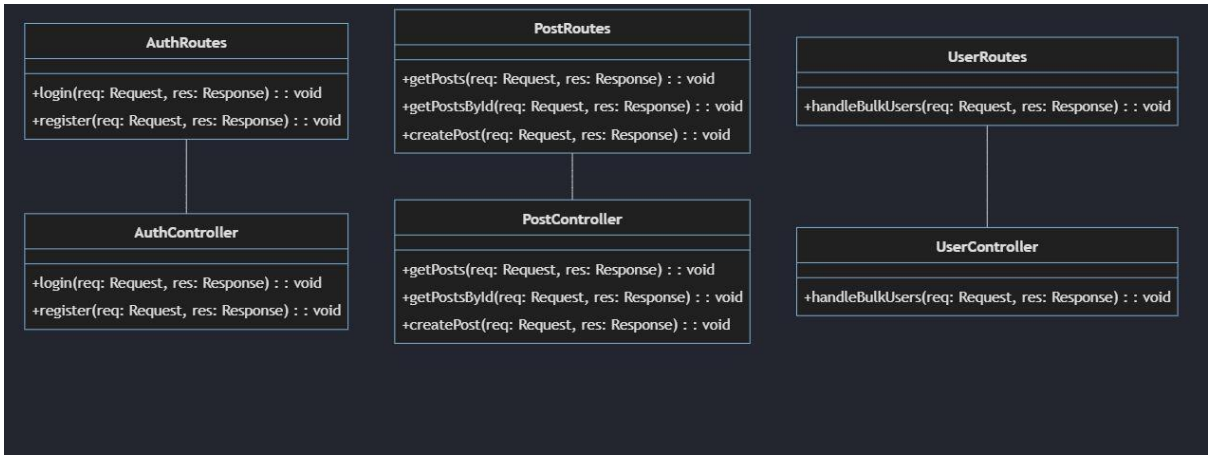
LÓGICA DEL PROGRAMA

Diagramas de clases:

Frontend



Backend



Tablas de Endpoints:

Método	Endpoint	Entrada	Salida	Descripción
GET	/api/posts/get	No aplica	Array de objetos `Post`	Obtiene todos los posts existentes
GET	/api/posts/get/:id	No aplica	Objeto `Post` o mensaje de error	Obtiene un post específico por su ID
POST	/api/posts/create	Objeto `Post`	Mensaje de éxito y objeto `Post` creado	Crea un nuevo post

Método	Endpoint	Entrada	Salida	Descripción
POST	/api/auth	{ codigo, password }	{ token, user }	Autentica al usuario y devuelve un token y los detalles del usuario.
GET	/api/auth	N/A	{ user }	Devuelve los detalles del usuario autenticado.
POST	/api/create	{ description, userCode, category, image, anonymous }	{ message, post }	Crea un nuevo post y devuelve un mensaje y los detalles del post creado.
GET	/api/posts	N/A	[{ post }]	Devuelve una lista de todos los posts.
GET	/api/posts/:id	N/A	{ post }	Devuelve los detalles de un post específico por su ID.
PUT	/api/posts/:id	{ description, userCode, category, image, anonymous }	{ message, post }	Actualiza un post específico por su ID y devuelve un mensaje y los detalles del post actualizado.
DELETE	/api/posts/:id	N/A	{ message }	Elimina un post específico por su ID y devuelve un mensaje.
POST	/api/register	{ codigo, name, apellidos, genero, facultad, carrera, email, password }	{ message, user }	Registra un nuevo usuario y devuelve un mensaje y los detalles del usuario registrado.
GET	/api/users	N/A	[{ user }]	Devuelve una lista de todos los usuarios.
GET	/api/users/:id	N/A	{ user }	Devuelve los detalles de un usuario específico por su ID.
PUT	/api/users/:id	{ codigo, name, apellidos, genero, facultad, carrera, email, password }	{ message, user }	Actualiza un usuario específico por su ID y devuelve un mensaje y los detalles del usuario actualizado.
DELETE	/api/users/:id	N/A	{ message }	Elimina un usuario específico por su ID y devuelve un mensaje.