UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA
INTRODUCCIÓN A LA PROGRAMACIÓN Y COMPUTACIÓN 1
CATEDRÁTICO: ING. NEFTALI CALDERÓN
TUTOR ACADÉMICO: RODRIGO ANTONIO PORÓN DE LEÓN

MANUAL TÉCNICO

JOSUÉ DAVID VELÁSQUEZ IXCHOP
CARNÉ: 202307705
SECCIÓN: E

GUATEMALA, 27 DE ABRIL DEL 2,024

DESCRIPCIÓN GENERAL DE LA SOLUCIÓN

La solución técnica propuesta para el sistema de gestión de usuarios y publicaciones es un ejercicio de ingeniería de software que se basa en principios fundamentales de desarrollo web y buenas prácticas de programación. Este enfoque se centra en la creación de un backend robusto y escalable utilizando tecnologías como Node.js y Express.js, aprovechando su capacidad para manejar solicitudes HTTP de manera eficiente y su flexibilidad para diseñar arquitecturas RESTful.

En primer lugar, se adopta Node.js como entorno de ejecución del servidor debido a su naturaleza asincrónica y su capacidad para manejar un gran número de conexiones concurrentes de manera eficiente. Esto es especialmente útil en aplicaciones web donde se espera un alto tráfico de usuarios. Express.js se elige como framework web debido a su simplicidad y flexibilidad, que permiten definir fácilmente rutas, middleware y lógica de negocio de manera modular y escalable.

Una de las características clave de la solución técnica es el modelado de datos utilizando clases. Se definen dos clases principales: **Usuario** y **Post**, que representan entidades importantes en el sistema. Esta aproximación orientada a objetos facilita la organización y estructuración del código, así como la reutilización de funcionalidades comunes a través de la herencia y la encapsulación.

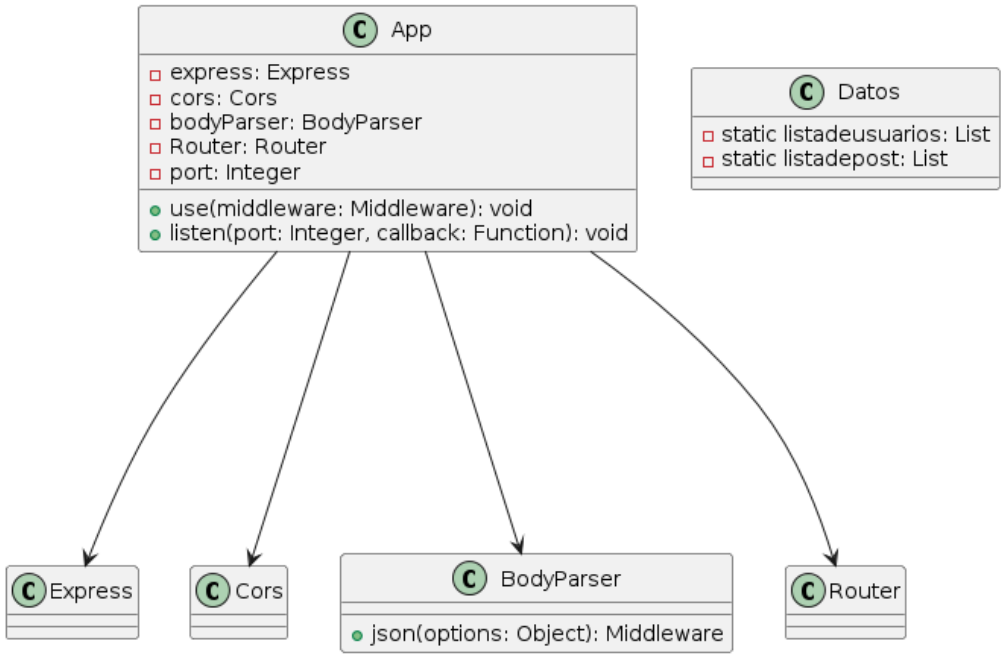
La persistencia de datos se aborda de manera abstracta en el código proporcionado, lo que permite flexibilidad en la elección de la tecnología de almacenamiento. Esto podría incluir bases de datos relacionales como MySQL o PostgreSQL, bases de datos NoSQL como MongoDB, o incluso almacenamiento en memoria utilizando estructuras de datos como arrays o mapas. La elección depende de los requisitos específicos del proyecto, como el rendimiento, la escalabilidad y la consistencia de los datos.

Otro aspecto importante de la solución es la implementación de validaciones de datos y medidas de seguridad. Se utilizan expresiones regulares para validar contraseñas durante el registro y el inicio de sesión, asegurando que cumplan con requisitos mínimos de complejidad. Además, se podría añadir capas de seguridad adicionales, como la protección contra ataques de inyección SQL y XSS, y la implementación de autenticación y autorización basada en tokens JWT (JSON Web Tokens) para proteger las rutas y funcionalidades sensibles.

El manejo de errores se realiza de manera adecuada mediante la captura de excepciones y la devolución de respuestas de error con códigos HTTP apropiados y mensajes descriptivos. Esto ayuda a los usuarios a comprender y resolver cualquier problema que surja durante el uso del sistema, mejorando la

experiencia del usuario y la confiabilidad del sistema en general.
En conclusión, la solución técnica propuesta para el sistema de gestión de usuarios y publicaciones es una combinación de tecnologías modernas, buenas prácticas de programación y enfoques de seguridad, diseñada para proporcionar un backend sólido, seguro y escalable que cumpla con los requisitos del proyecto y las expectativas de los usuarios finales.

DIAGRAMA DE CLASES



FUNCIONES

1. **Registro:**
   - Descripción: Esta función se encarga de registrar un nuevo usuario en el sistema.
   - Entrada: La entrada proviene del cuerpo de la solicitud HTTP (req.body) y debe contener los datos del nuevo usuario: código, nombres, apellidos, género, facultad, carrera, correo electrónico y contraseña.
   - Salida: La función devuelve un mensaje de éxito si el usuario se registra correctamente o un mensaje de error si ocurre algún problema durante el registro.
2. **DatosUsuarios:**
   - Descripción: Retorna la lista de usuarios registrados en el sistema.
   - Entrada: No requiere entrada.
   - Salida: Devuelve un objeto JSON que contiene la lista de usuarios.
3. **IniciarSesion:**
   - Descripción: Permite a un usuario iniciar sesión en el sistema.
   - Entrada: La entrada proviene del cuerpo de la solicitud HTTP (req.body) y debe contener el código y la contraseña del usuario que intenta iniciar sesión.
   - Salida: Si el inicio de sesión es exitoso, devuelve un mensaje de éxito junto con los datos del usuario. Si no, devuelve un mensaje de error indicando que el usuario no existe o la contraseña es incorrecta.
4. **ActualizarUsuarios:**
   - Descripción: Actualiza los datos de un usuario existente en el sistema.
   - Entrada: La entrada proviene del cuerpo de la solicitud HTTP (req.body) y debe contener los nuevos datos del usuario: código, nombres, apellidos, género, facultad, carrera, correo electrónico y contraseña.

- Salida: Devuelve un mensaje de éxito si la actualización se realiza correctamente o un mensaje de error si el usuario no existe.

5. **EliminarUsuarios:**
   - Descripción: Elimina un usuario del sistema.
   - Entrada: La entrada proviene del cuerpo de la solicitud HTTP (req.body) y debe contener el código del usuario que se desea eliminar.
   - Salida: Devuelve un mensaje de éxito si el usuario se elimina correctamente o un mensaje de error si el usuario no existe.
6. **CrearPost:**
   - Descripción: Crea una nueva publicación en el sistema.
   - Entrada: La entrada proviene del cuerpo de la solicitud HTTP (req.body) y debe contener los datos de la nueva publicación: código del usuario que realiza la publicación, descripción, imagen, categoría y si es anónimo o no.
   - Salida: Devuelve un mensaje de éxito si la publicación se crea correctamente o un mensaje de error si hay algún problema durante la creación.
7. **GetPost:**
   - Descripción: Obtiene la lista de publicaciones en el sistema, incluyendo datos del usuario que las realizó si no son anónimas.
   - Entrada: No requiere entrada.
   - Salida: Devuelve un objeto JSON que contiene la lista de publicaciones.
8. **GetReporteBarra:**
   - Descripción: Genera un reporte de las publicaciones más frecuentes por usuario.
   - Entrada: No requiere entrada.
   - Salida: Devuelve un objeto JSON que contiene el reporte con los usuarios y el número de publicaciones realizadas.
9. **ObtenerPublicaciones:**
   - Descripción: Obtiene la lista de publicaciones en el sistema.
   - Entrada: No requiere entrada.
   - Salida: Devuelve un objeto JSON que contiene la lista de publicaciones.
10. **EliminarPost:**
    - Descripción: Elimina una publicación específica del sistema.
    - Entrada: El ID de la publicación que se desea eliminar, que se obtiene a partir de los parámetros de la URL (req.params.id).

- Salida: Devuelve un mensaje de éxito si la publicación se elimina correctamente o un mensaje de error si la publicación no se encuentra.

Tabla de endpoints donde se describa el método, el endpoint, la entrada (no aplica para los GET y DELETE) y salida del endpoint y una pequeña descripción.

Método	Endpoint	Entrada	Salida	Descripción
GET	/DatosUsuarios	Ninguna	Lista de usuarios	Obtiene la lista de usuarios registrados.
GET	/obtenerPublicaciones	Ninguna	Lista de publicaciones	Obtiene la lista completa de publicaciones.
GET	/getPost	Ninguna	Lista de publicaciones	Obtiene la lista de publicaciones.
GET	/getReporteBarra	Ninguna	Lista de publicaciones	Obtiene un reporte de las publicaciones más frecuentes por usuario.
POST	/registro	Datos del nuevo usuario	Mensaje de éxito o error	Registra un nuevo usuario en el sistema.
POST	/iniciarSesion	Código y contraseña	Datos del usuario	Inicia sesión de un usuario en el sistema.
POST	/crearPost	Datos de la nueva publicación	Mensaje de éxito o error	Crea una nueva publicación en el sistema.
PUT	/ActualizarUsuarios	Datos actualizados del usuario	Mensaje de éxito o error	Actualiza los datos de un usuario existente.
DELETE	/EliminarUsuarios	Código del usuario a eliminar	Mensaje de éxito o error	Elimina un usuario del sistema.
DELETE	/eliminarPost/:id	ID de la publicación a eliminar	Mensaje de éxito o error	Elimina una publicación específica del sistema.