Especificaciones para la app

La aplicación debe cumplir las siguientes funcionalidades:

1. Crear usuarios, para lo cual se solicitará el nombre del usuario, contraseña, y cualquier otro dato que considere necesario. La imagen de perfil sólo podrá elegirse posteriormente a la creación del usuario.

2. Iniciar la sesión de un usuario. En caso de que este no exista, deberá indicarse dicho hecho y sugerir que se registre en la aplicación.

3. Una vez iniciada una sesión, se deben mostrar tres columnas en la pantalla principal:

a. Una primera columna debe mostrar un listado con los servidores a los que el usuario pertenece. Si no pertenece a ningún servidor, entonces mostrará un mensaje indicando esto. Por defecto , ningún servidor estará seleccionado. Sin embargo, al seleccionar uno, se deberá cargar una segunda columna con los canales que posea ese servidor. Además, esta columna debe tener un botón para crear un servidor nuevo.

b. La segunda columna debe mostrar un listado de los canales del servidor seleccionado, si no tiene ningún canal creado muestra un mensaje indicándolo. Por defecto , ningún canal estará seleccionado. Sin embargo, al seleccionar uno, se deberá cargar una tercera columna con los mensajes del chat de ese canal. Además, esta columna debe tener un botón para crear un canal nuevo.

c. La tercera columna mostrará los mensajes ordenados cronológicamente, con el más reciente en la parte inferior del chat. Si no hay ningún mensaje en el chat mostrará un mensaje indicando este hecho. Por supuesto, esta columna debe contar con un cuadro de texto para escribir un nuevo mensaje.

4. Los mensajes de un chat sólo pueden ser modificados o eliminados por el usuario que los ha creado.

5. Debe contar con un componente que permita mostrar el perfil del

usuario logueado. En el perfil del usuario se podrán actualizar los datos

personales del usuario, incluyendo la imagen del mismo.

****

utilizando HTML, CSS, JavaScript, Flask (como framework para la REST API) y MySQL (como base de datos). esta es una estructura básica

Estructura de archivos y carpetas:

- app

- static

- css

- style.css

- js

- script.js

- templates

- index.html

- login.html

- profile.html

- app.py

Pasos para comenzar:

Crear la estructura de carpetas y archivos

Configurar el entorno virtual y asegúrarnos de tener Flask y las bibliotecas necesarias instaladas.

Definir las rutas en app.py para manejar las solicitudes y respuestas de la API y la interfaz de usuario.

Diseña las plantillas HTML (index.html, login.html, profile.html) para representar la interfaz de usuario de la aplicación.

Estiliza las plantillas con CSS (style.css) para mejorar el aspecto visual de la aplicación.

Implementa las funciones en JavaScript (script.js) para manejar la interacción del usuario y realizar llamadas a la REST API.

Configurar/ conectar la base de datos MySQL para almacenar y recuperar la información de los usuarios, servidores, canales y mensajes.

Definir las rutas en Flask (app.py) para manejar las solicitudes de la REST API y conectar con la base de datos.

Implementar las funciones en Flask para gestionar la creación de usuarios, inicio de sesión, obtención de servidores, canales y mensajes, y la actualización de los datos de usuario.

Probar la app para estar seguros de que todas las funcionalidades corran bien.

esta es solo una visión general y hay muchos detalles adicionales a tener en cuenta al construir la app, como la autenticación, la validación de datos, el manejo de errores

---

rutas en app.py para manejar las solicitudes y respuestas de la API y la interfaz de usuario.

Primero, las rutas para las páginas de inicio y perfil del usuario

Luego, las rutas para la API que manejarán las solicitudes de creación de usuarios, inicio de sesión, obtención de servidores, canales y mensajes, y actualización de los datos de usuario:

****

```
// Funciones para interactuar con la REST API y cargar los datos en la interfaz de usuario

// Función para cargar la lista de servidores
function loadServers()
    // Realizar una llamada a la API para obtener los servidores del usuario
    // Usar fetch para realizar solicitudes HTTP
    // Llenar con los datos recibidos

// Función para cargar la lista de canales de un servidor seleccionado
function loadChannels(serverId)
    // Realizar una llamada a la API para obtener los canales del servidor seleccionado
    // Llenar el elemento <ul> con los datos recibidos


// Función para cargar la lista de mensajes de un canal seleccionado
function loadMessages(channelId)
    // Realizar una llamada a la API para obtener los mensajes del canal seleccionado
    // Llenar el elemento <ul> con los datos recibidos

// Función para crear un nuevo servidor
function createServer()
```

```
    // Mostrar un cuadro de diálogo para que el usuario ingrese el nombre del nuevo servidor
    // Realizar una llamada a la API para crear el servidor con el nombre ingresado
    // Recargar la lista de servidores para mostrar el nuevo servidor creado

// Función para crear un nuevo canal en el servidor seleccionado
function createChannel(serverId)
    // Mostrar un cuadro de diálogo para que el usuario ingrese el nombre del nuevo canal
    // Realizar una llamada a la API para crear el canal con el nombre ingresado y asociarlo al servidor seleccionado
    // Recargar la lista de canales para mostrar el nuevo canal creado

// Función para enviar un nuevo mensaje al canal seleccionado
function sendMessage(channelId)
    // Obtener el texto del mensaje ingresado por el usuario
    // Realizar una llamada a la API para enviar el mensaje al canal seleccionado
    // Recargar la lista de mensajes para mostrar el nuevo mensaje enviado

// Función para iniciar sesión
function login()
    // Obtener el nombre de usuario y contraseña ingresados por el usuario
    // Realizar una llamada a la API para autenticar al usuario
    // Si la autenticación es exitosa, redireccionar a la página principal
    // Si la autenticación falla, mostrar un mensaje de error al usuario

// Función para actualizar el perfil del usuario
function updateProfile()
    // Obtener los datos del perfil ingresados por el usuario
    // Realizar una llamada a la API para actualizar los datos del perfil en la base de datos
```

// Mostrar un mensaje de éxito al usuario
