**ESP.**

**Descripción del Proyecto**

El proyecto consiste en la creación de un **medidor de ocupación de un local**, donde podrás visualizar cómo varía la ocupación a lo largo del día, mostrando la carga de personas de forma horaria para cada día de la semana (por ejemplo, en un gimnasio o cualquier lugar con alta afluencia de público). Los datos de ocupación se simulan como si provinieran de un local real, pero en lugar de obtener datos en vivo, se controlan manualmente a través de una **API RESTful**.

En este sistema, cada día de la semana se representará por una **barra vertical** que muestra el porcentaje de ocupación para cada hora del día. Las barras estarán coloreadas para indicar diferentes niveles de ocupación (verde para baja ocupación, amarillo para media ocupación, y rojo para alta ocupación).

La API permitirá que puedas agregar, modificar o eliminar los datos de ocupación manualmente, lo que simula cómo podría funcionar con una fuente de datos real, como sensores o un sistema de gestión de reservas.

**Herramientas y Cómo Usarlas**

1. **React** (Frontend)
   * **Uso:** React se usará para construir la **interfaz de usuario** interactiva del medidor de ocupación. Tendrás una vista dinámica donde se muestran los datos de ocupación y los usuarios podrán ver en tiempo real cómo cambia la ocupación a lo largo del día.
   * **Cómo usarlo:**
     + Crearás un componente **OcupationBar** que representará las barras de cada día de la semana.
     + Con React, podrás hacer uso del **estado** para gestionar los valores de ocupación (como el porcentaje de ocupación) que provienen de la API.
     + Utilizarás **useEffect** para hacer peticiones a la API y obtener los datos de ocupación cuando el componente se renderiza.
     + Podrás **actualizar la UI dinámicamente** en función de los datos de la API.
2. **Tailwind CSS** (Estilo)
   * **Uso:** Tailwind CSS se utilizará para **dar estilo** a la interfaz, haciéndola visualmente atractiva y funcional, sin tener que escribir mucho CSS personalizado.
   * **Cómo usarlo:**
     + Usarás las clases de Tailwind para crear un diseño **limpio y moderno**, con un enfoque en la simplicidad y en las **barras de ocupación**.
     + Tailwind te permitirá **personalizar el diseño** de las barras (usando gradientes de color para mostrar diferentes niveles de ocupación), y las etiquetas de las horas.
     + Utilizarás **clases de diseño responsivo** de Tailwind para garantizar que la interfaz sea accesible y funcional en diferentes tamaños de pantalla.
3. **Python (Flask o FastAPI)** (Backend/API)
   * **Uso:** Usarás Python con un framework como **Flask** o **FastAPI** para construir una API RESTful que servirá los datos de ocupación y permitirá que se actualicen de manera simulada. Este backend será el responsable de manejar las peticiones de la interfaz de usuario.
   * **Cómo usarlo:**
     + Crearás un **servidor API** que tiene rutas para obtener los datos de ocupación, agregar nuevos datos y modificarlos. Por ejemplo, rutas como:
       - GET /ocupacion: Obtiene los datos de ocupación de todos los días.
       - POST /ocupacion: Permite agregar nuevos datos de ocupación.
       - PUT /ocupacion/{id}: Modifica los datos de ocupación de un día específico.
       - DELETE /ocupacion/{id}: Elimina los datos de ocupación de un día específico.
     + La API estará alimentada por datos controlados manualmente, lo que simula que se están recibiendo de un sistema de ocupación real.
     + Usarás **Flask** o **FastAPI** para crear las rutas y manejar las solicitudes HTTP.
     + Los datos podrían ser guardados en un archivo **JSON** local o en una **base de datos ligera** como SQLite para facilitar el acceso a ellos y la manipulación.
4. **JavaScript (AJAX / Fetch)** (Comunicación Frontend-Backend)
   * **Uso:** Utilizarás JavaScript con el método **fetch** o **Axios** para hacer solicitudes a la API backend desde el frontend React.
   * **Cómo usarlo:**
     + Utilizarás **fetch** en React para hacer peticiones **GET**, **POST**, **PUT** y **DELETE** a la API.
     + La respuesta de la API se usará para actualizar el estado del componente de React que muestra las barras de ocupación.
     + Los datos obtenidos del backend se mostrarán en tiempo real en el gráfico, permitiendo que el usuario vea cómo cambia la ocupación en función de las actualizaciones de la API.
5. **Base de Datos (Opcional, si usas Flask o FastAPI)**
   * **Uso:** Si decides usar una base de datos, puedes emplear **SQLite** para almacenar los datos de ocupación, lo que te permitirá realizar consultas, actualizaciones y eliminaciones de manera más estructurada.
   * **Cómo usarlo:**
     + Si usas **Flask** o **FastAPI**, puedes integrar **SQLAlchemy** o **Tortoise ORM** para interactuar con la base de datos.
     + Los datos de ocupación pueden ser almacenados en una tabla que tenga columnas como día, hora, y ocupación (porcentaje).
     + Esto te permitirá persistir los datos y hacer consultas más complejas si lo deseas en el futuro.

**Flujo del Proyecto**

1. **Frontend:**
   * En el frontend, usarás React para construir la interfaz visual del medidor de ocupación. Tendrás un componente con barras que muestran el nivel de ocupación por hora de cada día.
   * React se encargará de hacer solicitudes a la API para **actualizar** y **mostrar los datos dinámicamente**.
2. **Backend/API:**
   * El backend en Python (Flask o FastAPI) proporcionará una API que **responde a las peticiones del frontend**. La API recibirá y enviará datos de ocupación para cada día de la semana.
   * Utilizarás las rutas de la API para **simular la ocupación real** del local, permitiendo que se modifiquen y actualicen los datos de manera controlada.
3. **Comunicación:**
   * El frontend y el backend estarán conectados mediante **peticiones HTTP** (GET, POST, PUT, DELETE). El frontend hará solicitudes a la API para obtener los datos o actualizar la ocupación en tiempo real.

**Características y Funcionalidades:**

* **Visualización interactiva**: Las barras verticales muestran el nivel de ocupación de un local en cada hora del día.
* **Control de datos**: Los datos de ocupación son controlados a través de la API, simulando un sistema real de monitoreo de ocupación.
* **API RESTful**: La API permite la creación, lectura, actualización y eliminación de los datos de ocupación.
* **Escalabilidad**: Aunque es un prototipo simple, el sistema puede crecer en el futuro para manejar más datos y realizar análisis más complejos de ocupación.

**Posibles Mejoras Futuras:**

* **Integración con un sistema real**: Puedes integrar este proyecto con un sistema real de sensores o datos en vivo.
* **Interactividad avanzada**: Agregar funcionalidades de **notificación en tiempo real** si la ocupación supera un cierto umbral, por ejemplo, cuando un gimnasio está demasiado lleno.
* **Autenticación de usuarios**: Implementar autenticación y autorización para que solo ciertas personas puedan modificar los datos de ocupación.

Este proyecto no solo es útil y práctico, sino que también te permitirá destacar en tu portafolio al demostrar habilidades tanto en **frontend** (React y Tailwind CSS) como en **backend** (Python con Flask/FastAPI), y te da la flexibilidad para expandirlo en el futuro.

**ENG.**

The project involves the creation of an occupancy meter for a venue, where you can visualize how the occupancy varies throughout the day. It displays hourly occupancy levels for each day of the week (e.g., for a gym or any place with high public traffic). The occupancy data is simulated as if from a real venue but is manually controlled via a RESTful API.

In this system, each day of the week is represented by a vertical bar showing the percentage of occupancy for every hour of the day. The bars are color-coded to indicate different levels of occupancy (green for low, yellow for medium, and red for high).

The API allows manual addition, modification, or deletion of occupancy data, simulating how it would work with a real data source, such as sensors or a reservation management system.

Tools and How to Use Them

React (Frontend)

* Usage: React is used to build the interactive user interface for the occupancy meter. It provides a dynamic view where occupancy data is displayed in real-time, showing changes throughout the day.
* How to Use:
  + Create an OccupationBar component to represent the bars for each day of the week.
  + Use React state to manage occupancy values (e.g., percentages) retrieved from the API.
  + Utilize useEffect to fetch occupancy data from the API when the component renders.
  + Dynamically update the UI based on API data.

Tailwind CSS (Styling)

* Usage: Tailwind CSS is used to style the interface, creating a visually appealing and functional design without extensive custom CSS.
* How to Use:
  + Use Tailwind classes for a clean, modern design, focusing on the layout and appearance of occupancy bars.
  + Leverage Tailwind's gradient features to represent different occupancy levels.
  + Implement responsive design classes to ensure the UI is accessible across various screen sizes.

Python (Flask or FastAPI) (Backend/API)

* Usage: Python with Flask or FastAPI is used to build a RESTful API for serving and updating simulated occupancy data. The backend handles requests from the frontend.
* How to Use:
  + Create an API server with routes for managing occupancy data, such as:
    - GET /occupancy: Retrieves occupancy data for all days.
    - POST /occupancy: Adds new occupancy data.
    - PUT /occupancy/{id}: Updates occupancy data for a specific day.
    - DELETE /occupancy/{id}: Deletes occupancy data for a specific day.
  + Simulate data from a real system by controlling it manually through the API.
  + Use Flask or FastAPI to define routes and handle HTTP requests.
  + Store data in a local JSON file or a lightweight database like SQLite for structured access and manipulation.

JavaScript (AJAX / Fetch) (Frontend-Backend Communication)

* Usage: JavaScript fetch or Axios is used for sending requests to the API backend from the React frontend.
* How to Use:
  + Use fetch in React to make GET, POST, PUT, and DELETE requests to the API.
  + Use API responses to update React component states that display occupancy bars.
  + Show updated data in real-time in the graph, reflecting API changes.

Database (Optional)

* Usage: If a database is used, SQLite can store occupancy data, enabling more structured queries, updates, and deletions.
* How to Use:
  + With Flask or FastAPI, use SQLAlchemy or Tortoise ORM to interact with the database.
  + Store occupancy data in a table with columns for the day, hour, and percentage.
  + Enable data persistence and allow for more complex queries.

Project Flow

1. Frontend:
   * React renders the visual interface for the occupancy meter.
   * Requests API data to update and display dynamically.
2. Backend/API:
   * Python API provides endpoints for occupancy data retrieval and updates.
   * Simulates real venue occupancy through controlled data.
3. Communication:
   * HTTP requests (GET, POST, PUT, DELETE) connect the frontend to the backend.

Features and Functionalities

* Interactive Visualization: Vertical bars represent hourly occupancy levels for each day.
* Data Control: API-managed data simulates a real occupancy monitoring system.
* RESTful API: Create, read, update, and delete occupancy data.
* Scalability: Designed for potential expansion, such as advanced data analysis.

Future Improvements

* Real System Integration: Connect with real-time sensors or data sources.
* Advanced Interactivity: Add real-time notifications for high occupancy levels.
* User Authentication: Implement user roles for modifying occupancy data.

This project not only showcases practical and useful features but also highlights proficiency in frontend (React and Tailwind CSS) and backend (**Python with Flask/FastAPI) development, with room for future expansion.**