🛠 Manual Técnico

Este manual está dirigido a desarrolladores y personal de TI encargado del mantenimiento y evolución del sistema.

1. Arquitectura General

- **Flask**: microframework Python para backend.
- **MySQL**: motor de base de datos relacional.
- **Socket.IO**: notificaciones y seguimiento en tiempo real.
- **Leaflet.js**: mapas interactivos.
- **IA**: módulo `recommendation_service.py` para recomendaciones.

2. Despliegue

1. Clonar el repositorio:
    - git clone https://github.com/dawary12/App-de-Gesti-n-de-Pedidos-Motorizado.git

2. Crear y activar entorno virtual:
    1. python -m venv env
    2. .\env\Scripts\activate

3. Instalar dependencias:
    - pip install -r requirements.txt

4. Configurar la base de datos:
    - Importar el archivo DB_my_proyec.sql

3. Base de Datos

- Archivo de esquema: `DB_my_proyec.sql`.
- Tablas principales: `users`, `products`, `categories`, `orders`.
- Relaciones: FK de `orders.user_id` a `users.id`, etc.

4. Estructura del Código

```
Proyectofinal/
├── app/
│  ├── IA/ # Lógica del sistema de recomendaciones
│  ├── routes/ # Rutas Flask (auth, admin, cliente, motorizado)
│  ├── static/ # Recursos estáticos (CSS, imágenes)
│  ├── templates/ # Plantillas HTML
│  ├── models.py # Modelos de datos SQLAlchemy
│  ├── location_api.py # API para localización en tiempo real
│  ├── db.py, extensions.py # Configuración de base de datos y extensiones
├── run.py # Punto de entrada principal
├── DB_my_proyec.sql # Script de base de datos
├── README.md # Instrucciones del proyecto
├── Documentacion.md
├── requirements.txt
```

5. Módulos Clave

- auth.py: login, registro, roles.
- location_api.py: endpoints para posición y mapa.
- recommendation_service.py: lógica IA.
- routes/admin.py: CRUD de administración.
- routes/client.py, routes/motor.py: funcionalidades de cada rol.