Manual Técnico - Sistema de Parqueo para invitados, Condominio La Perla.

1. Estructura del Proyecto

```
app/
├── models/              # Modelos y lógica de negocio
│   ├── base.py          # Clases base abstractas
│   ├── usuario.py       # Gestión de usuarios
│   ├── vehiculo.py      # Gestión de vehículos
│   ├── lista_espera.py  # Sistema de cola
│   └── salidas_temporales.py # Gestión de movimientos
├── static/              # Archivos estáticos
│   ├── css/             # Estilos
│   └── img/             # Imágenes
├── templates/           # Plantillas HTML
│   └── index.html       # Página principal
└── db_config.py         # Configuración de base de datos
```

2. ¿Qué hace cada archivo? (Módulos)

base.py

Usa las clases base abstractas para todos los modelos, ya que con esto se creamos una estructura estándar CRUD (Create, Read, Update y Delete).

Es como el molde principal. Tiene dos partes importantes:

- ModeloBase: Da estructura básica a todos los modelos.
- GestorBase: Dice qué funciones debe tener cada gestor y cualquier cambio aquí afectan el resto de modelos.
- ModeloBase es la clase padre:

```
ModeloBase (Padre)
    ├── Usuario (Hija)
    ├── Vehiculo (Hija)
    ├── ListaEspera (Hija)
    └── SalidaTemporal (Hija)
```

usuario.py

Maneja todo sobre usuarios:

- Registro de usuarios.
- Actualización de datos.
- Eliminación de usuarios.
- Consulta de usuarios.

Atributos principales

class Usuario:

- cedula
- nombre
- teléfono
- email

vehiculo.py

Maneja todo sobre carros:

- Registro de vehículos.
- Asociación con usuarios.
- Consulta de vehículos.
- Eliminación de vehículos.

Atributos principales:

class Vehiculo:

- placa
- marca
- modelo
- id_usuario

lista_espera.py

Maneja la fila de espera:

- Sistema de cola para vehículos.
- Implementa FIFO (First In, First Out) .
- Funcionalidades:
- Agregar vehículos a espera.
- Procesar siguiente en cola.
- Cancelar espera.

Atributos principales:

class ListaEspera:

- id_vehiculo
- fecha_solicitud
- estado

salidas_temporales.py

Controla cuando hay que mover carros:

- Gestiona movimientos de vehículos.
- Maneja salidas temporales.
- Registro de movimientos.
- Control de posiciones.
- Retorno de vehículos.

Atributos principales:

class SalidaTemporal:

- id_vehiculo
- id_espacio_fila
- posicion_origen
- estado

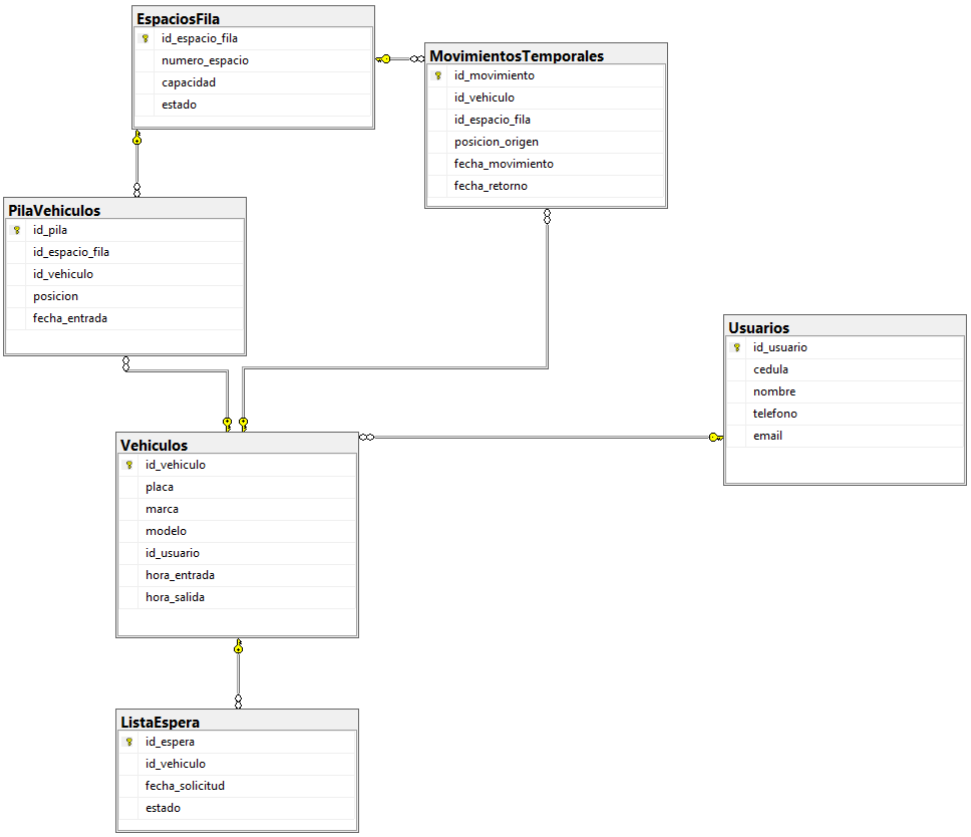
3. Cómo se conectan entre sí

1. Primero registras usuarios.
2. Luego registras sus carros.
3. Los carros se estacionan en fila.
4. Si no hay espacio, van a lista de espera.
5. Si hay que sacar un carro, se usa salidas_temporales.

4. Base de Datos

Tenemos estas tablas:

- Usuarios: guarda datos de la gente.
- Vehiculos: guarda datos de los carros.
- EspaciosFila: los lugares de estacionamiento.
- PilaVehiculos: qué carro está dónde.
- ListaEspera: los que están esperando.
- MovimientosTemporales: registro de movimientos.



5. Qué necesitas para que funcione

- Python 3.8 o más nuevo.
- Flask instalado.
- SQL Server .
- Estas librerías:
- flask
- pyodbc
- datetime

6. Si algo sale mal

- Los errores se guardan en logs.
- Hay mensajes de debug para ayudar.
- Cada parte tiene su propio manejo de errores.

Referencias

Brandl, G. (2024). Sphinx documentation. Sphinx. https://www.sphinx-doc.org/

Documentación de Python - Guía de Estilo. (2024).
https://google.github.io/styleguide/pyguide.html

Cómo Escribir Documentación en Python. (2024).
https://peps.python.org/pep-0257/

Manual de Documentación para Desarrolladores. (2024).
https://docs.readthedocs.io/

Manual de Usuario Doxygen - Herramienta para Documentar Código. (2024).
https://www.doxygen.nl/manual/index.html

Tutorial de MkDocs - Crea Documentación Fácilmente. (2024).

https://www.mkdocs.org/