

# Enunciado de la práctica U.T. 4

Desarrollar un sistema de gestión para un parque de atracciones que permita administrar visitantes, atracciones, y tickets. El sistema debe implementar una arquitectura basada en modelos y repositorios, con una interfaz de consola interactiva.

## Tablas

- Visitantes
  - id (Primary Key, autoincrement)
  - nombre (string, not null)
  - email (string, único)
  - altura (integer) : se guarda en cm
  - fecha\_registro (datetime)
  - preferencias (jsonb):
    - por ejemplo:
    - {
      - "tipo\_favorito": "extrema",
      - "restricciones": ["problemas\_cardiacos"],
      - "historial\_visitas": [
        - {"fecha": "2024-06-15", "atracciones\_visitadas": 8},
        - {"fecha": "2024-08-20", "atracciones\_visitadas": 12}
    - ]
    - }
- Atracciones
  - id (Primary Key, Autoincrement)
  - nombre (string, único, not null)
  - tipo (string): puede ser “extrema”, “familiar”, “infantil”, “acuatica”
  - altura\_minima (integer) : se guarda en cm
  - detalles (jsonb):
    - por ejemplo:
    - {
      - "duracion\_segundos": 60,
      - "capacidad\_por\_turno": 24,
      - "intensidad": 8,
      - "caracteristicas": ["looping", "caida\_libre", "giro\_360"],
      - "horarios": {
        - "apertura": "10:00",
        - "cierre": "22:00",

```

        "mantenimiento": ["14:00-15:00"]
    }
}

```

- activa (boolean, default=True)
- fecha\_inauguracion (date, default = NOW())
- Tickets
  - id (Primary Key, autoincrement)
  - visitante\_id (FK a Visitantes)
  - atraccion\_id (FK a Atracciones, NULL) : es null cuando el ticket vale para cualquier atracción
  - fecha\_compra (datetime)
  - fecha\_visita (date)
  - tipo\_ticket (string): puede ser “general”, “colegio”, “empleado”
  - detalles\_compra:  
por ejemplo:

```

{
    "precio": 45.99,
    "descuentos_aplicados": ["estudiante", "early_bird"],
    "servicios_extra": ["fast_pass", "comida_incluida"],
    "metodo_pago": "tarjeta"
}

```

- usado (boolean, default = False)
- fecha\_uso (datetime, NULL)

## CRUD Básico (5 Puntos)

### Creación (0.8 Puntos)

- crear\_visitante(nombre, email, preferencias\_json) (0.2 Puntos)
- crear\_atraccion(nombre, tipo, altura\_minima, detalles\_json) (0.2 Puntos)
- crear\_ticket(visitante\_id, fecha\_visita, tipo\_ticket, detalles\_compra\_json, atraccion\_id) (0.4 Puntos)

### Lectura (1 Punto)

- Obtener todos los visitantes (0.1 Puntos)
- Obtener todas las atracciones (0.1 Puntos)
- Obtener todos los tickets (0.1 Puntos)
- Obtener los tickets de un visitante específico (0.15 Puntos)
- Obtener los tickets vendidos para una atracción específica (0.15 Puntos)
- Obtener visitantes que tienen ticket para una atracción (directa o general) (0.3 Puntos)

- Obtener atracciones disponibles (activas) (0.1 Puntos)

## **Actualización (0.5 Puntos)**

- Marcar un ticket como usado (actualizar usado a True y poner fecha\_uso) (0.25 Puntos)
- Cambiar el estado activo/inactivo de una atracción (0.25 Puntos)

## **Eliminación (0.2 Puntos)**

- Eliminar un visitante (y sus tickets en cascada) (0.1 Puntos)
- Eliminar una atracción (0.1 Puntos)

## **Consultas (2.5 Puntos)**

- Visitantes con preferencia por las atracciones “extremas” (0.25 Puntos)
- Atracciones con intensidad mayor a 7 (0.25 Puntos)
- Tickets tipo “colegio” con precio menor a 30€ (0.25 Puntos)
- Atracciones con duración mayor a 120 segundos (0.25 Puntos)
- Visitantes con “problemas\_cardiacos” (0.25 Puntos)
- Atracciones que tengan “looping” y “caída libre” en sus características (0.5 Puntos)
- Tickets que tengan descuento “estudiante” (0.25 Puntos)
- Atracciones con al menos un horario de mantenimiento programado (0.5 Puntos)

## **Modificaciones en JSONB (2 Puntos)**

- Cambiar el precio de un ticket (0.5 Puntos)
- Eliminar una restricción a un visitante (0.5 Puntos)
- Añadir una nueva característica al array de características de una atracción (0.5 Puntos)
- Añadir una nueva visita al historial de visitas de un visitante (0.5 Puntos)

## **Consultas útiles (3 Puntos)**

- Listar visitantes ordenados por cantidad total de tickets comprados (descendiente 0.5 Puntos)
- Obtener las 5 atracciones más vendidas (en tickets específicos) (0.5 Puntos)
- Obtener visitantes que hayan gastado más de 100€ en tickets (suma de detalles\_compra→precio) (1 Punto)
- Atracciones compatibles para un visitante (Atracciones activas que coincidan en tipo, y donde el usuario cumpla con el mínimo de altura) (1 Punto)

## **Arquitectura obligatoria**

- I. Capa de modelos: definición de modelos Peewee para cada tabla
- II. Capa de repositorios: clases repositorio que encapsulan las operaciones con los modelos
- III. Menú de presentación: menú de consola con opciones numeradas

## **Gestión de excepciones y validación de datos**

Se deberá gestionar las excepciones que puedan surgir, y validar los datos siempre que sea necesario.

## **Formato de entrega**

Se entregará un .pdf con un enlace a una video-memoria de unos 5-10 minutos subida a YouTube, por ejemplo, comentada por encima, y con el programa funcionando para cada apartado, y una ligera explicación de problemas encontrados y decisiones de diseño si es que las hubiera. En la memoria se debe encontrar un enlace al repositorio del trabajo, donde cada integrante del grupo debe haber aportado en similar medida al proyecto.

No hace falta mostrar todas las funcionalidades, pero sí al menos 2 por apartado.