Documentación Técnica - Pista Manager

1. Introducción

Este documento detalla la arquitectura, módulos y funciones del sistema 'Pista Manager', diseñado para la gestión integral de una pista de hielo. El sistema abarca desde la venta de taquilla y control de acceso hasta la gestión escolar y reportes financieros.

Fecha de Generación: 2025-12-24

2. Arquitectura del Sistema

Tecnologías Principales:
- Backend: FastAPI (Python 3.10+)
- Base de Datos: PostgreSQL (Producción) / SQLite (Dev)
- ORM: SQLAlchemy
- Frontend: HTML5, JavaScript (Vanilla), Bootstrap 5
- Infraestructura: Docker & Docker Compose

Arquitectura:
El sistema sigue una arquitectura monolítica modular, donde el backend expone una API RESTful consumida por un frontend servido estáticamente.

3. Modelo de Base de Datos

El modelo de datos se define en 'app/models.py' y consta de las siguientes entidades principales:

1. SesionPatinaje: Representa un ticket vendido. Almacena hora de entrada/salida, costos, si rentó andadera, y el método de pago (Efectivo/Tarjeta).
2. Instructor: Profesores de la escuela. Incluye tarifas de honorarios.
3. RentaInstructor: Tabla intermedia que vincula una sesión de patinaje con un instructor para clases particulares, controlando el estado de pago del honorario.
4. Alumno: Estudiantes de la escuela (Artistico/Hockey).
5. PagoEscuela: Registro de mensualidades e inscripciones.
6. ReservaEvento: Gestión de fiestas y rentas completas.

4. API y Lógica de Negocio

Ubicación: app/main.py y app/routers/

A. Módulo de Ventas (Taquilla)
- POST /check-in/: Genera un nuevo ticket. Calcula costo inicial basándose en tarifa, andadera y clase. Asigna UUID único al ticket.
- POST /check-out/: (Legacy) Cierre manual de ticket. Calcula costo final y tiempo excedido.

B. Módulo de Acceso (Torniquetes) - app/routers/acceso.py
- POST /control-acceso/validar-entrada/: Activa el ticket. Registra 'hora_entrada' real. Previene 'Passback' (doble entrada).
- POST /control-acceso/validar-salida/: Cierra el ticket. Verifica que haya entrado previamente. Calcula el

costo final usando 'services.calculadora'. Si hay saldo pendiente, niega la salida.

C. Módulo Financiero - app/routers/reportes.py
- GET /reportes/cierre-dia/: Genera el balance contable del día.
  * Separa ingresos por método de pago (Efectivo vs Tarjeta).
  * Calcula ingresos de Taquilla + Escuela + Andaderas.
  * Resta egresos (Honorarios de instructores procesados).
  * Devuelve utilidad neta.

D. Módulo Escuela - app/routers/escuela.py
- Gestión CRUD de Alumnos, Profesores y Pagos de Mensualidades.
E. Módulo de Lealtad (Club Pista) - app/routers/lealtad.py
- POST /lealtad/registro/: Inscripción rápida con Nombre y Celular.
- GET /lealtad/buscar/{telefono}: Búsqueda de socio para asignar puntos en venta.
- GET /lealtad/top-mensual/: Generación de Ranking (Leaderboard) filtrado por mes.
- POST /lealtad/canjear/: Redención de premios.
- DELETE /lealtad/eliminar-datos/: Anonimización de datos personales (Derecho al Olvido).

5. Frontend

Ubicación: static/

1. index.html (Taquilla):
   - Interfaz principal de ventas.
   - Integración con 'JsBarcode' para generación de códigos CODE128.
   - Impresión dual: Ticket térmico (POS) y A4.

2. acceso.html (Monitor):
   - Simulación de torniquetes.
   - Input para pistola de código de barras.
   - Feedback visual (Semáforo Verde/Rojo) y sonoro (alertas).

3. cierre.html (Administración):
   - Dashboard financiero.
   - Visualización de arqueo de caja.
   - Botones de impresión dedicados para reportes (A4 y Tira POS).

4. app.js:
   - Lógica de cliente. Manejo de llamadas asíncronas (fetch) a la API.
   - Control de impresión y manipulación del DOM.

6. Seguridad y Validaciones

1. Validación de Flujo: Se implementó una regla de negocio estricta donde no es posible registrar una salida sin una entrada previa, mitigando errores de cobro.
2. Anti-Passback: Un ticket no puede ser usado para entrar dos veces consecutivas.
3. Integridad de Datos: Uso de transacciones SQL para asegurar que las ventas y asignaciones de clases se guarden atómicamente.