Documentación de Solución - API de Gestión de Flotas para Parque Salitre Mágico

Autor: Brigitte Utria **Proyecto:** Implementación de API de Gestión de Parque en Node.js
Fecha: 14/11/2024

Introducción

Este documento describe la solución implementada para el reto de desarrollar una aplicación de gestión para el parque de atracciones "Parque Salitre Mágico". Se detallan el diseño de la base de datos, las condiciones asumidas y una explicación del diagrama Entidad-Relación (ER). El objetivo de esta API es proporcionar funcionalidades para gestionar clientes, empleados, tiquetes y atracciones del parque.

Análisis de la Necesidad

La API de gestión tiene como objetivo proporcionar funcionalidades para gestionar clientes, empleados, tiquetes y atracciones del parque. Esto incluye:

- **Creación de registros**: Permitir la adición de nuevos clientes, empleados, tiquetes y atracciones.
- **Actualización de registros**: Habilitar la modificación de registros existentes.
- **Eliminación de registros**: Facilitar la eliminación de registros no deseados.
- **Verificación de datos**: Asegurar que los datos ingresados sean válidos y consistentes.

Diagrama Entidad-Relación (ER)

El diseño de la base de datos se basa en un diagrama ER que define las entidades principales y sus relaciones.

Entidades Principales

Clientes (Customers)

- id: Identificador único del cliente.
- name: Nombre del cliente.
- nid: Cédula del cliente.
- telephone: Teléfono del cliente.
- email: Correo electrónico del cliente.
- height: Estatura del cliente.
- age: Edad del cliente.
- parent_name: Información de contacto del tutor (en caso de ser menor de edad).

Empleados (Employees)

- id: Identificador único del empleado.
- name: Nombre del empleado.
- role: Rol del empleado (administrativo, logística, publicidad, operador, mantenimiento).
- work_schedule: Horario de trabajo del empleado.

Tiquetes (Tickets)

- id: Identificador único del tiquete.
- type: Tipo de tiquete (single, return).
- price: Precio del tiquete.
- station_id: Identificador de la estación donde se adquirió el tiquete.

Atracciones (Attractions)

- id: Identificador único de la atracción.
- name: Nombre de la atracción.
- description: Descripción de la atracción.
- classification: Clasificación de la atracción.
- use_conditions: Condiciones de uso de la atracción.
- status: Estado de la atracción (disponible, no disponible).

Relaciones

- **Un Cliente** puede tener múltiples **Tiquetes**.
- **Un Tiquete** está asociado a una **Estación** y una **Atracción**.
- **Una Atracción** puede tener múltiples **Tiquetes** asociados.

Diseño de la Base de Datos

La base de datos fue diseñada para soportar las operaciones CRUD de clientes, empleados, tiquetes y atracciones. Se tomaron en cuenta las siguientes decisiones clave:

- **Normalización**: Se aplicaron las reglas de normalización para evitar la redundancia de datos.
- **Referencias**: Se utilizaron claves foráneas en las tablas de **Tiquetes** para referenciar a las tablas de **Estaciones** y **Atracciones**.
- **Validación de Datos**: Se implementaron validaciones en el servidor para asegurar la consistencia de los datos ingresados.

Aclaraciones y Condiciones Asumidas

- **Asociación de Tiquetes**: Cada tiquete debe estar asociado a una estación y una atracción existente.
- **Manejo de Precios**: Los precios de los tiquetes se manejan como valores numéricos.
- **Estructura Básica**: La API maneja solo lo esencial, con la posibilidad de expansión futura.

Conclusión

La solución implementada cumple con algunos de los requisitos del reto, proporcionando una API funcional para la gestión del parque Salitre Mágico. La base de datos está diseñada para ser escalable y mantener la integridad de los datos. Las condiciones y suposiciones tomadas durante el desarrollo aseguran una solución robusta y eficiente.