**SOLUCIÓN Y ANÁLISIS DEL SISTEMA PROPUESTO**

**Introducción:**

El objetivo de este informe es presentar el análisis integral del sistema el cual ha sido diseñado con un enfoque centrado en el cliente, con requisitos que son independientes, negociables, estimables, de alcance reducido y verificables, lo cual permite una mayor previsibilidad, agilidad y facilidad en el manejo de las iteraciones durante el desarrollo.

**Descripción del Sistema Propuesto**

El sistema web desarrollado tiene como objetivo principal optimizar la gestión del parque de diversiones, permitiendo una asignación más eficiente de los recursos disponibles, mejorando la gestión y detalle ocupacional que permita conocer al parque información sobre el uso de sus atracciones.

Se utilizo el lenguaje Java con el framework Spring Boot, el cual facilita la creación de aplicaciones robustas y escalables, el uso de Spring Boot es altamente efectivo para el desarrollo de aplicaciones web debido a su capacidad para simplificar configuraciones complejas y mejorar la productividad del desarrollo, se utilizó como motor de base de datos relacional “PostgreSql”.

**Back-end**

La arquitectura back-end sigue el enfoque MVC (Modelo-Vista-Controlador) utilizando Spring Boot en Java. En este diseño, las entidades representan los modelos de datos de la aplicación, mientras que los DTO (objetos de transferencia de datos) facilitan la transferencia de datos entre el back-end y el front-end. Los repositorios de Java Persistence API (JPA) proporcionan una capa de acceso a datos, mientras que los servicios empresariales encapsulan la lógica de la aplicación a través de interfaces y sus implementaciones. Los controladores son responsables de procesar las solicitudes HTTP y conectar la lógica empresarial con el mundo exterior. Además, se incluye un paquete de seguridad para gestionar la autenticación y autorización de usuarios, así como un conjunto de utilidades para funciones comunes a todo el proyecto.

**Patrones**

Un patrón es una solución comprobada y generalizable a un problema recurrente en el diseño de software. Proporciona una estructura y un enfoque común para la resolución de problemas, promoviendo la reutilización y la coherencia en el desarrollo de software.

* Patrón MVC (Modelo-Vista-Controlador): Este patrón se refleja en la organización de los paquetes Controlador, Servicio y Entidad. Permite separar la lógica de presentación de la lógica de negocio y el acceso a los datos, mejorando el modularidad y la mantenibilidad del código.
* Patrón de seguridad JWT (tokens web JSON): este patrón se utiliza para configurar la seguridad y la autenticación de las aplicaciones mediante tokens JWT. Organizado en el paquete de Seguridad, este enfoque garantiza que los datos y los recursos estén protegidos, asegurando que solo los usuarios autorizados puedan acceder a ellos a través de la validación del token JWT.

**Arquitectura**

1. Entity:

* Crea un paquete para las clases de datos.
* Defina entidades utilizando anotaciones de JPA para mapearlas a tablas de base de datos.

1. Dto:

* Crea un paquete para los DTO.
* Defina objetos de transferencia de datos que representen las estructuras de datos utilizadas en las solicitudes y respuestas HTTP.

1. Repository:

* Crea un paquete para los repositorios de acceso a datos.
* Utiliza Spring Data JPA para definir interfaces de repositorio que extienden JpaRepository.

1. Service:

* Crea un paquete para los servicios de negocio.
* Definir interfaces para la lógica de negocio utilizando el patrón de diseño Service.

1. ImplServicio:

* Crea un paquete para las implementaciones de servicios.
* Implementa la lógica de negocio en clases que implementan las interfaces de servicio.

1. Controlador:

* Crea un paquete para los controladores.
* Defina controladores utilizando anotaciones de Sprint MVC para gestionar las solicitudes HTTP.

1. Constantes:

* Define una clase para cada conjunto de constantes relacionadas o una clase general, según la necesidad.

1. Utilidades:

* Crea un paquete para utilidades comunes.
* Definir clases de utilidad para funcionalidades reutilizables en todo el proyecto.

1. Excepciones:

* Define clases específicas para las excepciones personalizadas.

**Encarpetado Proyecto**

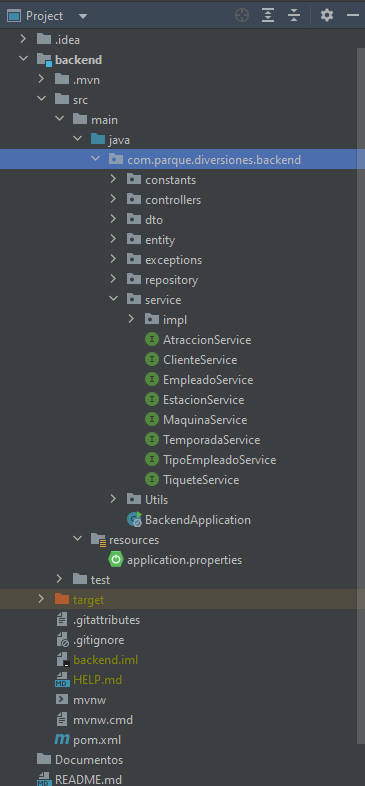


Ilustración 1Imgen encarpetado

Para la solución planteada se crearon las siguientes entidades:

**BaseEntity:** Los siguientes campos se crearon en una clase independiente a las entidades para poder extender de ella y hacer uso de sus atributos.

|  |  |  |
| --- | --- | --- |
| Tipo de dato | Nombre | Descripción |
| Long | id | Representa el identificador único de la entidad. Es generado automáticamente por la base de datos mediante la estrategia IDENTITY. |
| Boolean | estado | Indica si la entidad está activa o inactiva. Puede ser true (activa) o false (inactiva). Es opcional (nullable = true). |
| LocalDateTime | created\_at | Fecha y hora en la que se creó la entidad. Se formatea en el patrón yyyy-MM-dd HH:mm:ss. |
| LocalDateTime | updated\_at | Fecha y hora de la última actualización realizada a la entidad. Se formatea en el patrón yyyy-MM-dd HH:mm:ss. |
| LocalDateTime | deleted\_at | Fecha y hora en la que la entidad fue marcada como eliminada. |
| Long | created\_by | Identificador del usuario que creó la entidad. |
| Long | updated\_by | Identificador del usuario que realizó la última actualización en la entidad. |
| Long | deleted\_by | Identificador del usuario que marcó la entidad como eliminada. |

**Cliente:** Los siguientes campos se crearon para guardar los datos de los clientes que ingresan al parque de diversiones:

|  |  |  |
| --- | --- | --- |
| Tipo de dato | Nombre | Descripción |
| String | nombre | Representa el nombre del cliente. Tiene un límite de 60 caracteres y es obligatorio (nullable = false). |
| String | apellido | Representa el apellido del cliente. Tiene un límite de 60 caracteres y es obligatorio (nullable = false) |
| String | email | Almacena el correo electrónico del cliente. Tiene un límite de 60 caracteres y es obligatorio (nullable = false). |
| Integer | edad | Indica la edad del cliente en años. Es obligatorio (nullable = false). |
| Double | estatura | Representa la altura del cliente en metros. Es obligatorio (nullable = false). |
| String | numero\_identificacion | Almacena el número de documento único del cliente. Tiene un límite de 60 caracteres, es obligatorio (nullable = false), y debe ser único (unique = true). |
| String | telefono | Representa el número telefónico del cliente. Tiene un límite de 10 caracteres y es obligatorio (nullable = false). |
| List<Tiquete> | tiquetes | Lista de tiquetes asociados al cliente. Implementa una relación **Uno a Muchos** con la entidad Tiquete. Las operaciones en cascada (CascadeType.ALL) permiten que cualquier cambio en el cliente (como eliminación) afecte automáticamente a los tiquetes relacionados. También se utiliza orphanRemoval = true para eliminar automáticamente los tiquetes huérfanos (sin cliente). |
| List<Familiar> | familiares | Lista de familiares asociados al cliente. Implementa una relación **Uno a Muchos** con la entidad Familiar. Las operaciones en cascada (CascadeType.ALL) y orphanRemoval = true aseguran que cualquier cambio en el cliente afecte a sus familiares relacionados y elimine los familiares huérfanos. |

**Empleado:** Los siguientes campos se crearon para guardar los datos de los empleados que ingresan al parque de diversiones:

|  |  |  |
| --- | --- | --- |
| Tipo de dato | Nombre | Descripción |
| String | nombre | Representa el nombre del cliente. Tiene un límite de 60 caracteres y es obligatorio (nullable = false). |
| String | apellido | Representa el apellido del cliente. Tiene un límite de 60 caracteres y es obligatorio (nullable = false) |
| String | email | Almacena el correo electrónico del cliente. Tiene un límite de 60 caracteres y es obligatorio (nullable = false). |
| Integer | edad | Indica la edad del cliente en años. Es obligatorio (nullable = false). |
| String | numero\_identificacion | Almacena el número de documento único del cliente. Tiene un límite de 60 caracteres, es obligatorio (nullable = false), y debe ser único (unique = true). |
| String | telefono | Representa el número telefónico del cliente. Tiene un límite de 10 caracteres y es obligatorio (nullable = false). |
| String | horario\_laboral | Almacena el horario laboral del empleado (e.g., "9:00 AM - 6:00 PM"). Tiene un límite de 50 caracteres y es obligatorio (nullable = false). |
| TipoEmpleado | tipo\_empleado | Almacena el horario laboral del empleado (e.g., "9:00 AM - 6:00 PM"). Tiene un límite de 50 caracteres y es obligatorio (nullable = false). |
| List<Tiquete> | tiquetes | Lista de tiquetes asociados al empleado. Define una relación **Uno a Muchos** con la entidad Tiquete. Las operaciones en cascada (CascadeType.ALL) aseguran que cualquier cambio en el empleado afecte los tiquetes relacionados, y orphanRemoval = true elimina los tiquetes huérfanos (sin empleado). |

**Estacion:** Los siguientes campos se crearon para guardar los datos de las estaciones que se encuentran en el parque de diversiones:

|  |  |  |
| --- | --- | --- |
| Tipo de dato | Nombre | Descripción |
| String | nombre | Representa el nombre del cliente. Tiene un límite de 60 caracteres y es obligatorio (nullable = false). |
| String | descripcion | Almacena una descripción detallada de la estación, como sus características o propósito. Tiene un límite de 500 caracteres y es obligatorio (nullable = false). |
| Boolean | estado\_disponibilidad | Indica si la estación está disponible (true) o no disponible (false). Es obligatorio (nullable = false). |
| Double | porcentaje\_ocupacion | Representa el porcentaje de ocupación de la estación, expresado como un valor decimal. Es obligatorio (nullable = false). |
| List<DayOfWeek> | días\_habilitacion | Lista de los días de la semana en los que la estación está habilitada para operar. Utiliza un @ElementCollection para almacenar esta información en una tabla secundaria denominada dias\_habilitacion. Cada día se almacena como texto en la base de datos mediante @Enumerated(EnumType.STRING). |
| List<Temporada> | temporadas | Lista de temporadas en las que la estación está operativa. Define una relación **Muchos a Muchos** con la entidad Temporada. Se gestiona a través de una tabla de unión denominada estacion\_temporada, que asocia estaciones con temporadas. |
| List<Atraccion> | estacion | Define una relación **Muchos a Muchos**. La lista se mapea con la entidad Atraccion, donde el campo estacion en Atraccion hace referencia a esta estación. Se utiliza la anotación @OneToMany para definir esta relación y cascade = CascadeType.ALL para que las operaciones de persistencia (como guardar, actualizar y eliminar) se propaguen automáticamente a las atracciones asociadas. El atributo orphanRemoval = true asegura que si una atracción es eliminada de la lista, también se eliminará en la base de datos. |

**Familiar:** Los siguientes campos se crearon para guardar los datos de los familiares que requiere un cliente menor de edad para ingresar al parque de diversiones:

|  |  |  |
| --- | --- | --- |
| Tipo de dato | Nombre | Descripción |
| Long | id | Representa el identificador único de la entidad. Se genera automáticamente utilizando la estrategia GenerationType.IDENTITY. |
| String | numero\_identificacion | Almacena el número único de identificación del familiar. Tiene un límite de 60 caracteres, es obligatorio (nullable = false), y debe ser único (unique = true). |
| String | nombre | Representa el nombre del cliente. Tiene un límite de 60 caracteres y es obligatorio (nullable = false). |
| String | apellido | Representa el apellido del cliente. Tiene un límite de 60 caracteres y es obligatorio (nullable = false) |
| String | email | Almacena el correo electrónico del cliente. Tiene un límite de 60 caracteres y es obligatorio (nullable = false). |
| String | telefono | Representa el número telefónico del cliente. Tiene un límite de 10 caracteres y es obligatorio (nullable = false). |
| String | relacion | Indica la relación del familiar con el cliente, como "Padre", "Hermano", "Cónyuge", etc. Tiene un límite de 30 caracteres y es obligatorio (nullable = false). |
| Cliente | cliente | Relación **Muchos a Uno** con la entidad Cliente. Representa al cliente con quien este familiar está relacionado. La columna cliente\_id en la base de datos almacena la referencia y es obligatoria (nullable = false). |

**Maquina:** Los siguientes campos se crearon para guardar las maquinas existentes en el parque de diversiones:

|  |  |  |
| --- | --- | --- |
| Tipo de dato | Nombre | Descripción |
| String | nombre | Representa el nombre del cliente. Tiene un límite de 100 caracteres y es obligatorio (nullable = false). |
| String | descripcion | Almacena una descripción detallada de la máquina, como sus características o funcionalidad. Tiene un límite de 500 caracteres y es obligatorio (nullable = false). |
| Boolean | estado\_disponibilidad | Indica si la máquina está disponible (true) o no disponible (false). Es obligatorio (nullable = false). |
| String | motivo\_inhabilitacion | Almacena la razón por la cual la máquina está inhabilitada. Es opcional (nullable = true) y tiene un límite de 500 caracteres. |
| Estacion | estacion | Relación **Muchos a Uno** con la entidad Estacion. Indica la estación en la que está ubicada la máquina. Es obligatorio (nullable = false). |
| Empleado | empleado\_mantenimiento | Relación **Muchos a Uno** con la entidad Empleado. Representa al empleado responsable del mantenimiento de la máquina. Es opcional (nullable = true). |

**Temporada:** Los siguientes campos se crearon para guardar las temporadas del parque de diversiones:

|  |  |  |
| --- | --- | --- |
| Tipo de dato | Nombre | Descripción |
| String | nombre | Representa el nombre único de la entidad, utilizado como identificador textual. Es obligatorio (nullable = false) y debe ser único (unique = true). |
| String | fecha\_inicio | Almacena la fecha de inicio de la entidad o del evento relacionado. Se formatea como yyyy-MM-dd y es obligatorio (nullable = false). |
| String | fecha\_fin | Indica la fecha de finalización de la entidad o del evento relacionado. Se formatea como yyyy-MM-dd y es obligatorio (nullable = false). |
| String | descripcion | Proporciona una descripción detallada de la entidad o del evento. Tiene un límite de 500 caracteres y es obligatorio (nullable = false). |
| Boolean | estado\_disponibilidad | Indica si la entidad está disponible (true) o no disponible (false). Es obligatorio (nullable = false). |

**TipoEmpleado:** Los siguientes campos se crearon para guardar los tipos de empleados del parque de diversiones:

|  |  |  |
| --- | --- | --- |
| Tipo de dato | Nombre | Descripción |
| String | cargo | Representa el cargo o rol específico de una persona o entidad, como un título laboral o una posición jerárquica. Tiene un límite de 60 caracteres y es obligatorio (nullable = false). |
| String | descripcion | Proporciona una descripción detallada o explicativa del cargo. Puede incluir información sobre responsabilidades, funciones o características del rol. Tiene un límite de 500 caracteres y es obligatorio (nullable = false). |

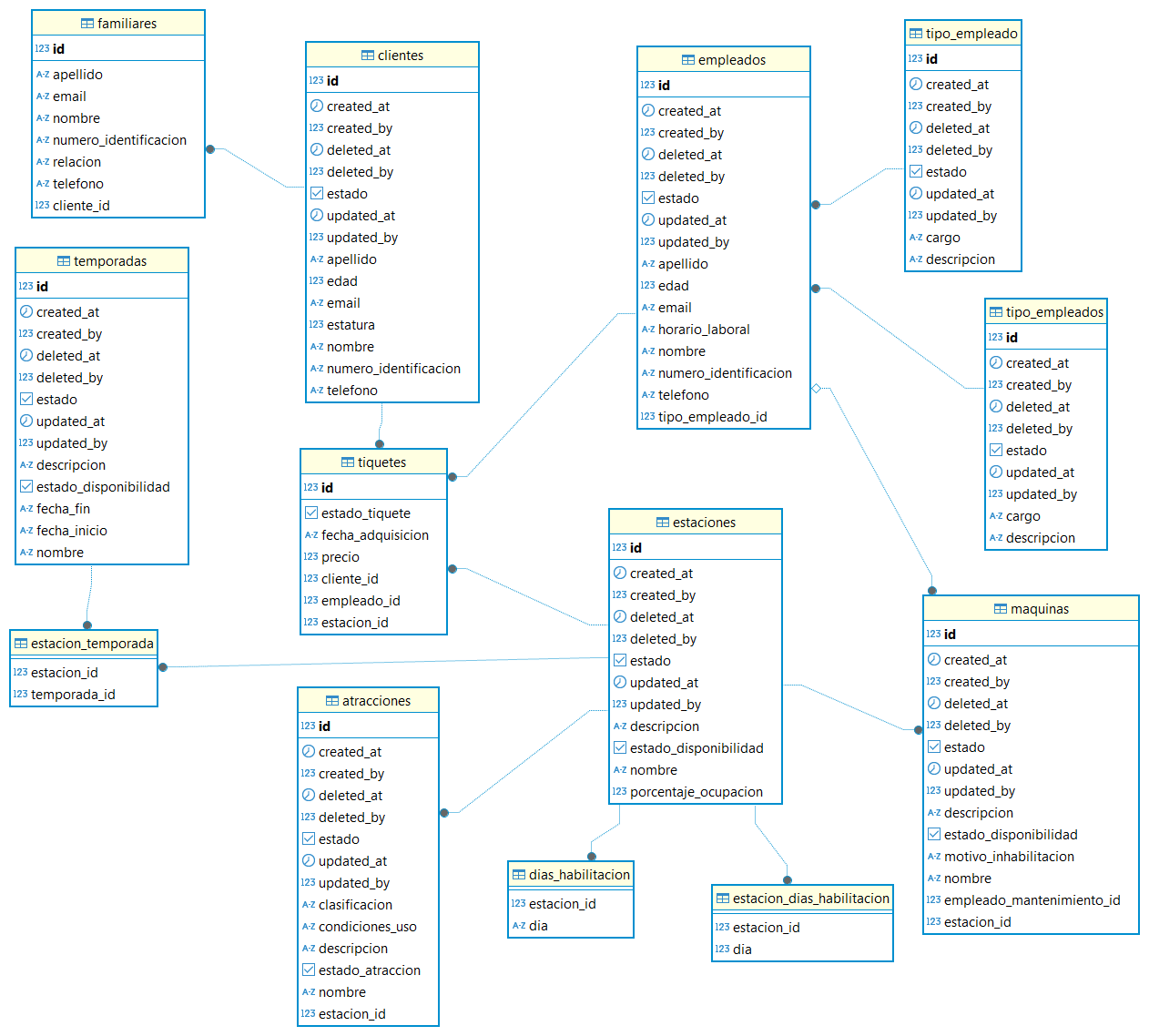
**Tiquete:** Los siguientes campos se crearon para guardar un tiquete para el ingreso a las maquinas del parque de diversiones:

|  |  |  |
| --- | --- | --- |
| Tipo de dato | Nombre | Descripción |
| Long | id | Representa el identificador único de la entidad. Se genera automáticamente utilizando la estrategia GenerationType.IDENTITY. |
| Boolean | estado\_tiquete | Indica si el tiquete está activo (true) o inactivo (false). Es opcional (nullable = true). |
| String | fecha\_adquisicion | Fecha en la que se adquirió el tiquete. Es obligatorio (nullable = false). |
| Estacion | estacion | Relación **Muchos a Uno** con la entidad Estacion. Representa la estación asociada al tiquete. Es obligatorio (nullable = false). |
| Cliente | cliente | Relación **Muchos a Uno** con la entidad Cliente. Indica el cliente que adquirió el tiquete. Es obligatorio (nullable = false). |
| Empleado | empelado | Relación **Muchos a Uno** con la entidad Empleado. Define al empleado que realizó la venta o gestión del tiquete. Es obligatorio (nullable = false) y se recupera de forma diferida (FetchType.LAZY). |
| Double | precio | Costo monetario del tiquete. Es obligatorio (nullable = false). |

**Atraccion:** Los siguientes campos se crearon para guardar las atracciones que tiene el parque de diversiones:

|  |  |  |
| --- | --- | --- |
| Tipo de dato | Nombre | Descripción |
| String | nombre | Representa el nombre de la atracción. Tiene un límite de 60 caracteres y es obligatorio (nullable = false). |
| String | descripcion | Proporciona una descripción detallada de la atracción, incluyendo información relevante como su temática o características principales. Tiene un límite de 500 caracteres y es obligatorio (nullable = false). |
| String | clasificacion | Indica la categoría o clasificación de la atracción, como "emocionante", "familiar", "acuática", etc. Tiene un límite de 500 caracteres y es obligatorio (nullable = false). |
| String | condiciones\_uso | Define las condiciones o requisitos que deben cumplirse para usar la atracción, como restricciones de edad, altura o peso. Tiene un límite de 500 caracteres y es obligatorio (nullable = false). |
| Boolean | estado\_atraccion | Indica si la atracción está operativa (true) o fuera de servicio (false). Es obligatorio (nullable = false). |
| Estacion | Estacion\_id | Este atributo representa una referencia a la entidad Estacion, lo que implica que cada instancia de la entidad que contiene este atributo está asociada a una estación específica. Se utiliza la anotación @ManyToOne, lo que indica que la relación entre la entidad actual (por ejemplo, Atraccion) y Estacion es de muchos a uno (muchas atracciones pueden estar asociadas a una estación). |

**Diagrama Entidad Relación Base de Datos (parque\_atracciones)**



Este modelo entidad-relación representa la base de datos de un sistema de gestión para un parque de atracciones. Incluye las siguientes entidades principales:

* Clientes y Familiares: Gestión de clientes, con datos personales y relaciones familiares.
* Empleados y Tipos de Empleados: Registro de empleados, sus roles y horarios laborales.
* Atracciones y Máquinas: Detalles de las atracciones, condiciones de uso, y las máquinas asociadas con su estado de mantenimiento.
* Tiquetes: Registro de tiquetes, precio, estado, y cliente asociado.
* Estaciones y Temporadas: Gestión de estaciones, porcentaje de ocupación, días de habilitación, y temporadas asociadas.

Relaciones entre entidades: Vinculación clara entre clientes, estaciones, temporadas, y empleados para asegurar un control integral del parque.

**Documentación de Servicios del Backend**

A continuación, se documentan los servicios expuestos por la API del backend, que son utilizados para la funcionalidad del sistema. Todos los endpoints siguen la estructura RESTful.

ClienteController

|  |  |  |
| --- | --- | --- |
| Método | Endpoint | Descripción |
| Get | /api/v1/clientes | Obtiene la información de todos los clientes. |
| Get | /api/v1/clientes/1 | Obtiene la información de un cliente especifico mediante su ID. |
| Delete | /api/v1/clientes/1 | Elimina un cliente especifico por su ID. |
| Put | /api/v1/clientes/1 | Actualiza los datos de un cliente existente por su ID. |
| Post | /api/v1/clientes | Crea un nuevo cliente en el sistema. |

EmpleadoController

|  |  |  |
| --- | --- | --- |
| Método | Endpoint | Descripción |
| Get | /api/v1/empleados | Obtiene la información de todos los empleados. |
| Get | /api/v1/empleados/1 | Obtiene la información de un empleado especifico mediante su ID. |
| Delete | /api/v1/empelados/1 | Elimina un empleado especifico por su ID. |
| Put | /api/v1/empleados/1 | Actualiza los datos de un empleado existente por su ID. |
| Post | /api/v1/empleados | Crea un nuevo empleado en el sistema. |

EstacionController

|  |  |  |
| --- | --- | --- |
| Método | Endpoint | Descripción |
| Get | /api/v1/estaciones | Obtiene la información de todas las estaciones. |
| Get | /api/v1/estaciones/1 | Obtiene la información de una estación especifica mediante su ID. |
| Delete | /api/v1/estaciones/1 | Elimina una estación especifica por su ID. |
| Put | /api/v1/estaciones/1 | Actualiza los datos de una estación existente por su ID. |
| Post | /api/v1/estaciones | Crea una nueva estación en el sistema. |

MaquinaController

|  |  |  |
| --- | --- | --- |
| Método | Endpoint | Descripción |
| Get | /api/v1/maquinas | Obtiene la información de todas las maquinas. |
| Get | /api/v1/maquinas/1 | Obtiene la información de una maquina especifica mediante su ID. |
| Delete | /api/v1/maquinas/1 | Elimina una máquina especifica por su ID. |
| Put | /api/v1/maquinas/1 | Actualiza los datos de una máquina existente por su ID. |
| Post | /api/v1/maquinas | Crea una nueva máquina en el sistema. |

AtraccionController

|  |  |  |
| --- | --- | --- |
| Método | Endpoint | Descripción |
| Get | /api/v1/atracciones | Obtiene la información de todas las atracciones. |
| Get | /api/v1/atracciones/1 | Obtiene la información de una atracción especifica mediante su ID. |
| Delete | /api/v1/atracciones/1 | Elimina una atracción especifica por su ID. |
| Put | /api/v1/atracciones/1 | Actualiza los datos de una atracción existente por su ID. |
| Post | /api/v1/atracciones | Crea una nueva atracción en el sistema. |

TipoEmpleadoController

|  |  |  |
| --- | --- | --- |
| Método | Endpoint | Descripción |
| Get | /api/v1/tipoEmpleados | Obtiene la información de todos los tipos de empleados. |
| Get | /api/v1/tipoEmpleados/1 | Obtiene la información de un tipo de empleado especifico mediante su ID. |
| Delete | /api/v1/tipoEmpleados/1 | Elimina un tipo de empleado especifico por su ID. |
| Put | /api/v1/tipoEmpleados/1 | Actualiza los datos de un tipo de empleado existente por su ID. |
| Post | /api/v1/tipoEmpleados | Crea un nuevo tipo de empleado en el sistema. |

TemporadaController

|  |  |  |
| --- | --- | --- |
| Método | Endpoint | Descripción |
| Get | /api/v1/temporadas | Obtiene la información de todas las temporadas. |
| Get | /api/v1/temporadas/1 | Obtiene la información de una temporada especifica mediante su ID. |
| Delete | /api/v1/temporadas/1 | Elimina una temporada especifica por su ID. |
| Put | /api/v1/temporadas/1 | Actualiza los datos de una temporada existente por su ID. |
| Post | /api/v1/temporadas | Crea una nueva temporada en el sistema. |

TiqueteController

|  |  |  |
| --- | --- | --- |
| Método | Endpoint | Descripción |
| Get | /api/v1/tiquetes | Obtiene la información de todos los tiquetes. |
| Get | /api/v1/tiquetes/1 | Obtiene la información de un tiquete especifico mediante su ID. |
| Delete | /api/v1/tiquetes/1 | Elimina un tiquete especifico por su ID. |
| Put | /api/v1/tiquetes/1 | Actualiza los datos de un tiquete existente por su ID. |
| Post | /api/v1/tiquetes | Crea un nuevo tiquete en el sistema. |