**Documento Técnico Proyecto ParkApp**

Sistema de información de parqueadero que administra y gestiona el servicio de parqueo pago a diferentes tipos de vehículos con costes de cobro distintos.

Es un sistema de información desarrollado con tecnología Javascript en el Backend, como servicio Api-Rest y gestor de bases de datos MySQL.

**Características**

* Tecnología Javascript como lenguaje de programación
* Node JS./Express/Sequelize/MySQL como herramientas utilizadas
* Arquitectura API-REST
* Jest/Supertest como Herramientas de testeo

**Descripción del modelo arquitectónico**

ParkApp se desarrolló como aplicación orientada al servicio. Se desarrollaron los endPoints como se describen a continuación:

**Ruta: /api/vehiculo/registro**

Método POST: utilizado para añadir vehículos a la Base de datos de ParkApp

Cuerpo de petición POST debe ir de la siguiente forma:

{

“placa”:”num\_placa”,

“tipo”:”valor numérico”

}

Placa y tipo son obligatorios, de no ser enviado alguno de los dos, el sistema rechazará la petición informado con código de estado 400 y un mensaje con el formato {message:"Faltan datos para el registro de un vehículo."}

**Mensajes de respuesta**

* Código 400, {message:"Faltan datos para el registro de un vehículo."} si falta alguno de los campos arriba listados.
* Código 202, {message:"Ya existe un vehículo con esa placa registrado."} si se intenta registrar un vehículo que ya está registrado.
* Código 200, {objeto con datos del vehículo creado} si se registra un nuevo vehículo a la base de datos. Objeto devuelto en el body de la respuesta

{

"id": 20,

"placa": "ABC123",

"tipo": "1",

"updatedAt": "2022-06-30T18:21:58.594Z",

"createdAt": "2022-06-30T18:21:58.594Z"

}

**Ruta: /api/parqueo/registro**

**Método POST**: utilizado para ingresar un parqueo de vehículo al parqueadero

Cuerpo de petición POST debe ir de la siguiente forma:

{

“placa”:”num\_placa”,

}

Placa es obligatorio, de no ser enviado, el sistema rechazará la petición informando con código de estado 400 y un mensaje con el formato {message:"Faltan datos para el registro de un vehículo."}

**Mensajes de respuesta**

* Código 400, {message:"Faltan datos para el registro de un vehículo."} si falta alguno de los campos arriba listados.
* Código 202, {message:" Ya existe un vehículo con esa placa ingresado al parqueadero."} si se intenta registrar un vehículo que ya está registrado.
* Código 200, {objeto con datos del parqueo} si se registra un nuevo parqueo de vehículo en el parqueadero. Objeto devuelto en el body de la respuesta.

{

"id": 46,

"placa": "ABC123",

"tipo": 0,

"fechaIngreso": "6/30/2022",

"horaIngreso": "1:28:45 AM",

"updatedAt": "2022-06-30T01:28:45.804Z",

"createdAt": "2022-06-30T01:28:45.804Z"

}

**Método PATCH**: utilizado para registrar una salida de vehículo del parqueadero

Cuerpo de petición PATCH debe ir de la siguiente forma:

{

“placa”:”num\_placa”,

}

Placa es obligatorio, de no ser enviado, el sistema rechazará la petición informado con código de estado 400 y un mensaje con el formato { message:”Falta información del vehículo para la asignación de un parqueadero”}

**Mensajes de respuesta**

* Código 400, { message:”Falta información del vehículo para la asignación de un parqueadero”}si falta alguno de los campos arriba listados.
* Código 202, {message:" No existe un vehículo con esa placa ingresado al parqueadero."} si se intenta registrar una salida de un vehículo que no está registrado.
* Código 200, {objeto con datos del parqueo} si la salida se registra sin problemas. Objeto devuelto en el body de la respuesta.

{

"Mensaje": "Salida de vehículo registrada",

"vehiculo": {

"id": 12,

"placa": "ABC123",

"tipo": 0,

"estancia": 0,

"createdAt": "2022-06-30",

"updatedAt": "2022-06-30"

}

}

**Ruta: /api/cuenta/cobro**

**Método GET**: utilizado para generar un archivo con las cuentas de cobros de los vehículos residentes

**Mensajes de respuesta**

* Código 400, {message:" Se presentó un problema en el proceso generación de listado de cuentas de cobro."}.
* Código 200, {message:"Listado ejecutado con éxito.", listado: resultado}

**Ruta: /api/reinicio/mes**

**Método GET**: utilizado para borrar los registros de estancias de vehículos oficiales y reestablece las estancias de los vehículos residentes.

**Mensajes de respuesta**

* Código status 400, {message:" Se pesentó un problema en el proceso de reinicio."}.
* Código 200, {message:” Reinicio ejecutado con éxito.."}

**Testing**

En cada ruta de servicio de endpoint, se encentra un archivo que en su nombre se identifica con **test**, y este se corresponde con las pruebas de cada endpoint en sus diferentes métodos.

**Descripción de carpetas de proyecto**

El proyecto cuenta con una estructura de carpetas arregladas de la siguiente forma que se corresponde con las rutas anteriormente descritas:

* **Api/cuenta:** Contiene todo los archivos ejecutables de rutas, controladores, modelos, servicios y test que son importantes para la generación del estado de cuentas de los vehículos residentes y genera el archivo con la información correspondiente a ese detalle. Valida la operación del servicio correspondiente a esa ruta como se describió en el apartado anterior.
* **Api/parqueo:** Contiene los archivos para gestionar la ruta correspondiente descrita anteriormente que registrar las entradas y salidas de los vehículos al parqueadero con sus respectivos importes de pago.
* **Api/reinicio:** Contiene los archivos para gestionar la ruta correspondiente descrita anteriormente que registrar reiniciar los valores de estancias de los diferentes tipos de vehículos.
* **Api/vehículo:** Contiene los archivos del aplicativo para gestionar el ingreso de cada nuevo vehículo al sistema.
* **config:** Contiene los archivos de configuración del servidor web, ORM Sequelize para el acceso a la base de datos.
* **util:** Contiene una librería para gestión y cálculo de fechas.
* **temp**: Carpeta que almacena temporalmente el archivo generado con las cuentas de cobro de los vehículos residentes.
* **Routes**: Contiene las rutas de los servicios API
* **Index y app**: Son los archivos principales del aplicativo.