NELUMBO

CONSULTORES

Reto Técnico de Ingreso Laboral

RETO NELUMBO

BACKEND

Estimado,

A continuación, describimos la especificación del reto que hemos diseñado para validar el proceso de ingreso a nuestra empresa. Las condiciones mínimas de aceptación son las indicadas en este documento, cualquier esfuerzo por demostrar capacidades superiores a las requeridas es bien valorado por nuestra empresa.

El reto a nivel técnico consiste en el desarrollo de la API con las funcionalidades y estructura indicadas a continuación.

¡Mucho éxito!
Nelumbo Consultores

CONSIDERACIONES GENERALES

Se requiere una APIREST para el control de vehículos en los parqueaderos de varios socios, además considerar un histórico de los vehiculos parqueados con anterioridad para los indicadores que se solicitaran

El API debe tener un sistema de usuarios protegido por autorización jwt token. El TOKEN tendrá una expiración de 6 horas.

Se manejarán 2 roles (ADMIN, SOCIO).
El sistema debe precargar un usuario admin, se va a manejar el email único para poder ingresar

usuario: admin@mail.com
pass: admin

El administrador es el único que puede agregar usuarios de tipo SOCIO

PERMISOLOGÍA

Se van a listar los permisos de cada rol, luego estará especificada cada funcionalidad

El rol admin

- Debe poder crear usuarios con ROL SOCIO
- Debe poder hacer un CRUD de parqueaderos y asociarle un socio
- Puede revisar listado/detalle de vehículos en un parqueadero en especifico
- Puede enviar emails a los socios (simulación)
- Puede ver indicadores

El rol socio

- Pueden registrar entrada de vehículos por algún parqueadero
- Pueden registrar salida de vehículos por algún parqueadero, siempre y cuando el vehículo haya tenido una entrada.
- Puede ver listado de los parqueaderos que tiene asociados
- Puede ver listado/detalle de todos los vehículos en un parqueadero especifico que le pertenezca
- Puede ver indicadores

FUNCIONALIDADES

Crear los endpoints necesario para la autenticación de usuarios (login/registro/logout)

Endpoints CRUD de parqueaderos, considerar campo que incluya costo por hora

Endpoint POST "registre ingreso" a través de su número de placa y id del parqueadero, internamente se registra la fecha-hora del ingreso.
La respuesta del endpoint es un código http 201 con el siguiente body

```
{
 "id": 0, {id generado del registro}
}
```

se debe validar que la placa ingresada no coincida con algún vehículo que se encuentre dentro de algún parqueadero del sistema.
en tal caso regresar un código http 400 con el siguiente body

```
{
 "mensaje": "No se puede Registrar Ingreso, ya existe la placa en este u otro parqueadero"
}
```

Endpoint POST "registrar salida" a través de su número de placa y el id del parqueadero, internamente se registra la fecha-hora de la salida.
La respuesta del endpoint es un código http 200 con el siguiente body

```
{
"mensaje": "Salida registrada"
}
```

se debe validar que el vehículo este en ese momento dentro de las instalaciones, si no se regresa error 400

```
{
"mensaje": "No se puede Registrar Salida, no existe la placa en el parqueadero"
}
```

Endpoint GET "Listado de vehículos", devuelve el listado de vehículos que se encuentran actualmente parqueados en un parqueadero
Respuesta

```
[
{
"id": 0,
"placa": "",
"fechaIngreso": "",
....
```

```
...
}, {...}
]
```

Se debe realizar un **microservicio** de simulación de envío de correo
Debe tener un endpoint POST que reciba

```
{
   "email": "",
    "placa": "",
    "mensaje": "",
   “parqueaderoNombre”: “”
}
```

Debe imprimir en log la solicitud que le llego

Devolver una respuesta 200 con

```
{
   "mensaje": "Correo Enviado"
}
```

Desde al API de registro de vehículos se debe llamar a este microservicio de simulación de correo usando un endpoint POST con los mismos parámetros que necesita el microservicio

```
{
   "email": "",
    "placa": "",
    "mensaje": "",
   “parqueaderoId”: “”
}
```

Responder lo que llega del microservicio
Validar que la placa ingresada se encuentre en el parqueadero.

Indicadores

- (**ADMIN** y **SOCIO**) regresar los 10 vehículos que más veces se han registrado en los diferentes parqueaderos y cuantas veces han sido

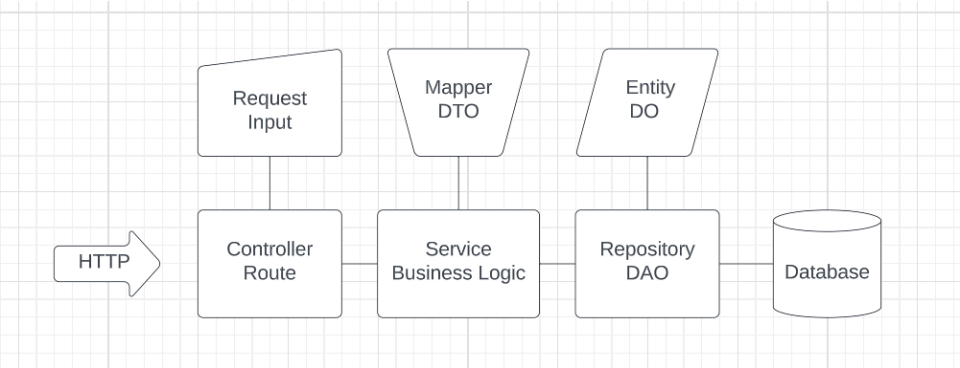
- **(ADMIN** y **SOCIO)** regresar los 10 vehículos que más veces se han registrado en un parqueadero y cuantas veces han sido
- **(ADMIN** y **SOCIO)** verificar de los vehículos parqueados cuales son por primera vez en ese parqueadero
- (**SOCIO**) obtener las ganancias de hoy, esta semana, este mes, este año de un parqueadero en especifico
- **(ADMIN** y **SOCIO)** Buscar vehículos parqueados mediante coincidencia en la placa, ejemplo ingreso "HT", puede regresar los vehículos con placa "123HT4" | "HT231E"

Consideraciones a tener en cuenta:

- El parqueadero en su creación se configurará cuantos vehículos puede tener en un mismo momento, y el costo por hora del vehículo
- Validar la capacidad al momento de los ingresos (error 400 con mensaje descriptivo)
- La placa del vehículo siempre debe tener 6 caracteres de longitud, alfanumérica. No permite caracteres especiales ni la letra ñ
- Cuando se registra la salida de un vehículo se debe mover a la tabla historial con su fecha de salida

Observaciones:

- Usar Excepciones cuando se necesiten regresar las respuestas con error 400 detectadas dentro de las clases servicio
- Implementar patrón de diseño de servicios|repositorios para la lógica manejar los package (controllers, services, entities, repositories) donde cada capa realice su función
- Manejar la protección de los endpoints por rol y token válidos.



- Usar PostgreSQL
- Usar Java SpringBoot
- Entregar colección de postman, documentando cada request.
- Definir estructura del código.
- Plantear un modelo entidad relación para la solución.

- Para la entrega se debe proporcionar enlace para descargar en github la solución, y en el README cualquier observación que se debe tener en cuenta de cómo correr el proyecto en local