

BRIDGE
FUND

Full Stack Developer

Coding challenge

Process

You will receive an invite to the private GitLab repository via email. Once you have access, please commit the project code to the repository.

Keep it simple and effective. But feel free to use any technology that helps you build your application (Think of databases, caching, Docker, etc...).

Feel free to contact us if something is unclear or when you have any other questions.

Good luck!

Introduction

This challenge consists of 5 user stories. PB-01 until PB-05. For a company called ParkingBusiness.

You are to help them build a backend application for managing “parking spaces” inside a parking building.

This is a pilot project for one of the buildings ParkingBusiness owns. But it’s the assumption that they want to use the software for other buildings as well in the near future.

Context

The building looks like the following:

- Floor 1: Parking spaces 1 – 75
- Floor 2: Parking spaces 76 – 150

Each parking space in the building has a requirement:

- | | | |
|------------------|---|-----------------|
| Space 1 – 50: | Spaces used for residents of the building | No charge |
| Space 51 – 130: | Cars | € 5,00 per hour |
| Space 131 – 150: | Motorcycles | € 3,00 per hour |

PB-01: Overall Project Setup

“As a **Developer** I want to have an environment to build the application and deliver to my client”.

Requirements:

- **Setup** your project in a **Git** hosting of your choice.

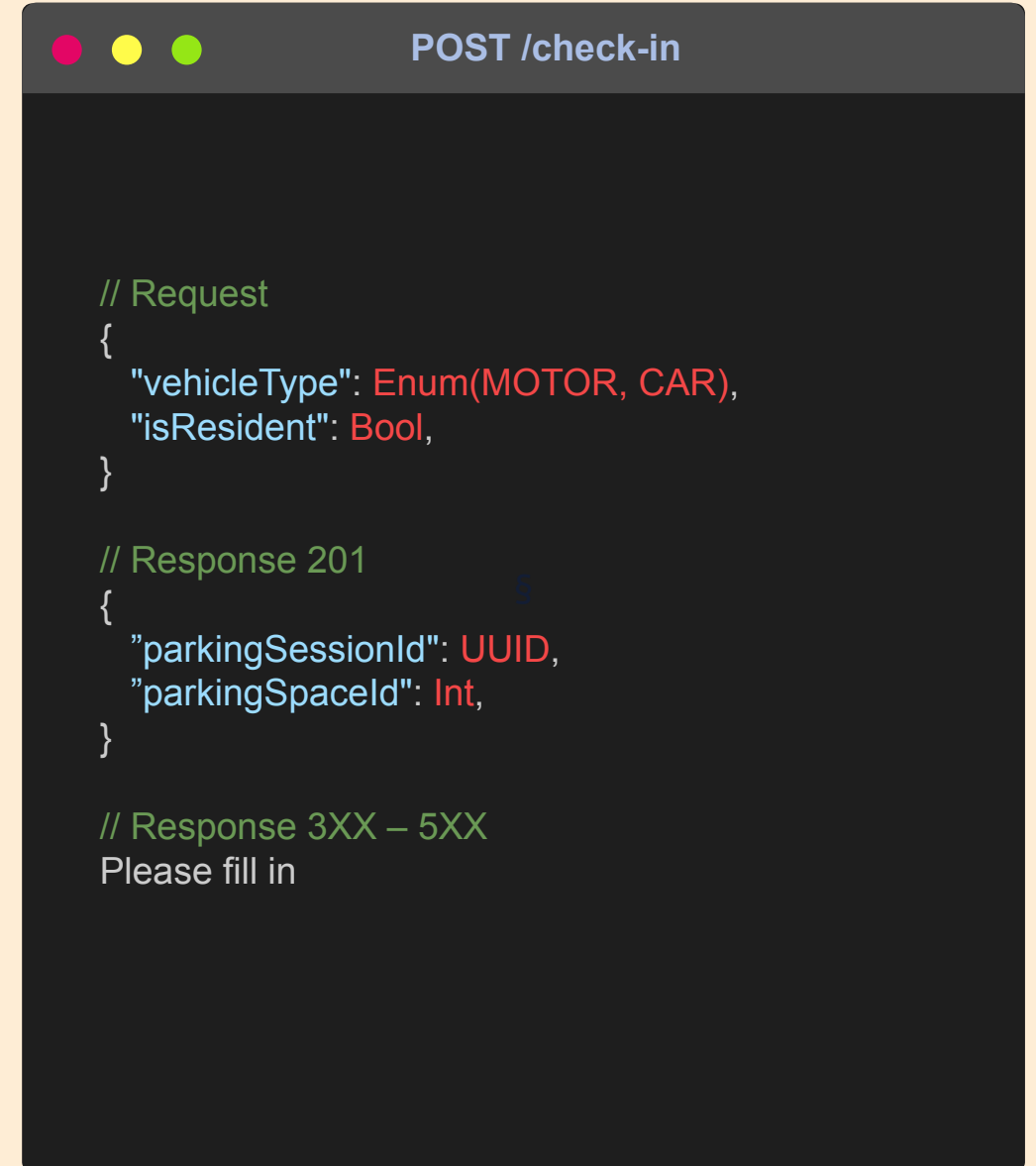
! We like the NestJS framework, but use whatever technology shows your skills best !

PB-02: Check In

“As **ParkingBusiness** I want to have an endpoint to check-in and register vehicles into my building. So I can start a parking session and track the time parked.”

Requirements:

- Implement an endpoint with roughly these specifications.



```
POST /check-in

// Request
{
  "vehicleType": Enum(MOTOR, CAR),
  "isResident": Bool,
}

// Response 201
{
  "parkingSessionId": UUID,
  "parkingSpaceId": Int,
}

// Response 3XX – 5XX
Please fill in
```

PB-03: Check Out

“As **ParkingBusiness** I want to have an endpoint to check-out vehicles. So I can end a parking session and report what to charge the vehicle owner”.

Requirements:

- Implement an endpoint with roughly these specifications.

```
POST /check-out

// Request
{
  "parkingSessionId": UUID,
  "isResident": Bool,
}

// Response 201
{
  "sessionLengthInHoursMinutes": Number,
  "parkingSpaceId": Int,
}

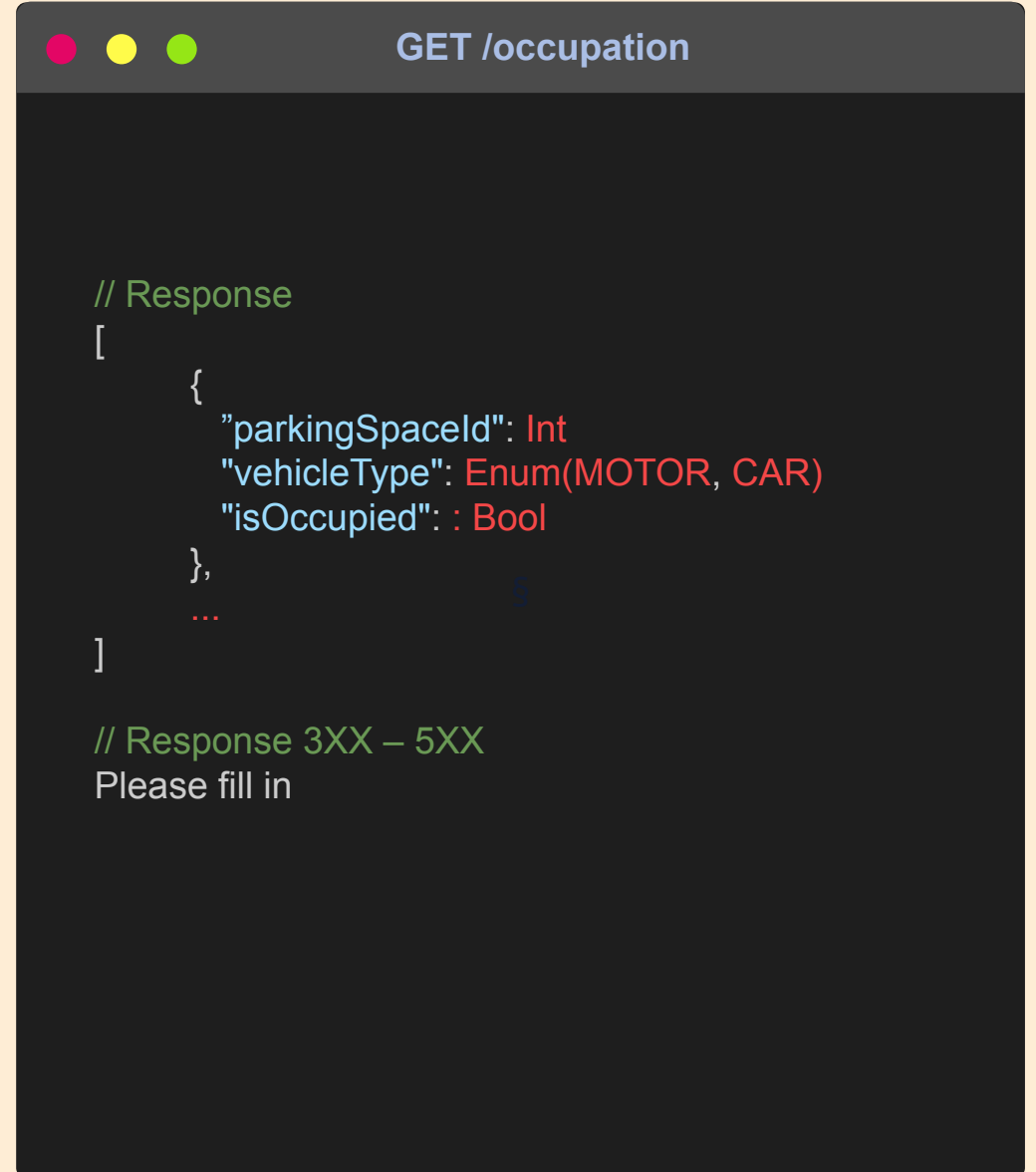
// Response 3XX – 5XX
Please fill in
```


PB-04: Occupation

“As **ParkingBusiness** I want have an endpoint that gives me overview on the total occupation of all spaces. So I have insights. And can display upfront if I’m fully occupied for a particular vehicle type”.

Requirements:

- Implement an endpoint with roughly these specifications.



```
GET /occupation

// Response
[
  {
    "parkingSpaceId": Int
    "vehicleType": Enum(MOTOR, CAR)
    "isOccupied": : Bool
  },
  ...
]

// Response 3XX – 5XX
Please fill in
```

PB-05: Spike Ticket

“As **ParkingBusiness** I want to explore what more I can do with the API. So hopefully I can improve my service, profit or software”.

Requirements:

- Investigate if you can come up with other features or models for the business.
- Explain why you think it is a good idea and note this down as user stories.
- Add this into a file called ROADMAP.md at the root level of your project.

PB-05: Bonus Ticket

“As **ParkingBusiness** I want to visualize the output of **GET /occupation**. So I can visualize/display it to my customers at the entrance of the building”.

Requirements:

- Build a simple UI.

PB-06: Bonus ticket

“<Your user story>”.

Requirements:

Implement what you have noted in ticket PB-04