**Documentação Técnica — Estapar Parking Manager**

**1. Introdução**

O sistema **Estapar Parking Manager** é um backend desenvolvido em Java com Spring Boot para gerenciar estacionamentos, controlando número de vagas disponíveis, entradas e saídas de veículos, faturamento por setor e regras de preço dinâmico. Ele se integra a sensores e cancelas por meio de Webhooks e expõe uma API REST para consulta e operação.

**2. Visão Geral do Sistema**

**Tecnologias utilizadas**

* **Java 17**
* **Spring Boot**
* **Spring Data JPA / Hibernate**
* **MySQL**
* **Swagger / OpenAPI**
* **Docker (Simulador externo)**

**Componentes principais**

* Backend Spring Boot com API REST
* Banco de dados MySQL para persistência
* Integração com simulador de garagem via Webhook (/webhook)
* Documentação automática da API com Swagger

**Diagrama geral (descreva ou anexe)**

scss

Simulador → Webhook (/webhook) → Backend Spring → Banco de Dados MySQL → API REST → Swagger

**3. Endpoints da API**

**GarageController**

* GET /garage → Abre a garagem e libera para operação

**ParkingController**

* POST /parking/park
  + Request:

json

{

"licensePlate": "ZUL0001",

"lat": -23.561684,

"lng": -46.655981

}

* + Response: Car parked

**PriceController**

* POST /price/plate-status
  + Request:

json

{

"licensePlate": "ZUL0001"

}

* + Response:

json

{

"licensePlate": "ZUL0001",

"priceUntilNow": 5.00,

"entryTime": "2025-01-01T12:00:00Z",

"timeParked": "1h",

"lat": -23.561684,

"lng": -46.655981

}

**RevenueController**

* GET /revenue?date=2025-01-01&sector=A
  + Response:

json

{

"amount": 1500.00,

"currency": "BRL",

"timestamp": "2025-01-01T23:59:59Z"

}

**4. Estrutura do Banco de Dados**

**Tabelas**

* **Vehicle**
  + id (PK)
  + licensePlate
  + entryTime
  + exitTime
  + sector
  + price
* **Spot**
  + id (PK)
  + lat
  + lng
  + sector
  + occupied
* **Revenue**
  + id (PK)
  + date
  + sector
  + amount

**Relacionamentos**

* Um setor possui várias vagas
* Um veículo ocupa uma vaga por vez

**5. Regras de Negócio**

* **Preço dinâmico por lotação**
  + Até 25% → -10% desconto
  + Até 50% → preço normal
  + Até 75% → +10% acréscimo
  + Até 100% → +25% acréscimo
* **Lotação**
  + Setor fechado ao atingir 100% até liberar uma vaga

**6. Como Rodar o Projeto**

1. Clone o repositório:

bash

git clone <repo-url>

1. Configure application.properties:

ini

spring.datasource.url=jdbc:mysql://localhost:3306/estapar

spring.datasource.username=admin

spring.datasource.password=admin

1. Suba o Docker do simulador:

arduino

docker run -d --network="host" cfontes0estapar/garage-sim:1.0.0

1. Execute o projeto:

arduino

mvn spring-boot:run

1. Acesse Swagger:

bash

http://localhost:3003/swagger-ui/index.html

**7. Como Testar**

* Use Postman, Insomnia ou Swagger UI para fazer requisições.
* Confira os logs no console.
* Valide os dados diretamente no MySQL.

**8. Melhorias Futuras**

* Implementar autenticação JWT
* Monitoramento com Spring Boot Actuator
* Cache para consultas frequentes
* Deploy em ambiente cloud (AWS, GCP, Azure)