Desafio iLab - Ifood

O programa iLab tem como objetivo construir o futuro no curto prazo da área de Tecnologia e Data do iFood acelerando pessoas com perfil Junior para atuarem plenamente como um Foodlover.
Os Foodlovers do programa iLab foram organizados em grupos para realizar um desafio que consiste na criação de uma aplicação de telemetria.

Escopo do Desafio

Track History - Telemetria dos entregadores do Ifood

Neste projeto deve-se manter todo o histórico de telemetria de um entregador para um determinado pedido. A telemetria é encerrada quando há um evento de conclusão ou cancelamento (por parte do entregador).

Grupo Chocode

Nome do grupo: Chocode

Foodlovers:
Claudia Nogueira dos Anjos
Daniel Silveira
Raphaela Leite
Silvano Araújo
Vitor Eleuterio

Repositório: Chocode
Metodologia: KanBan-Github
Mentor Ifood: Danilo De Luca

Projeto

O projeto foi desenvolvido utilizando:

- Back End: Java e Spring Boot;
- Front End: HTML, CSS e Javascript.

Iniciamos com a importação de dependências necessárias pelo Spring Initializr, conforme segue abaixo:

➔ Spring Initializr
Maven Project
Spring Boot 2.6.4
Packaging Jar
Java 17

Dependências:

Spring Web, MySQL Driver
Spring Data JPA
Spring Boot DevTools
Lombok
Spring Security
H2 Database
MySQL Driver
PostgreSQL Driver
Dependências do JWT no arquivo *pom.xml*:

```
<dependency>
        <groupId>io.jsonwebtoken</groupId>
        <artifactId>jjwt-api</artifactId>
        <version>0.11.1</version>
</dependency>
<dependency>
        <groupId>io.jsonwebtoken</groupId>
        <artifactId>jjwt-impl</artifactId>
        <version>0.11.1</version>
        <scope>runtime</scope>
</dependency>
<dependency>
        <groupId>io.jsonwebtoken</groupId>
        <artifactId>jjwt-jackson</artifactId>
        <version>0.11.1</version>
        <scope>runtime</scope>
</dependency>
```

Utilizamos o banco relacional H2 e MySQL para os testes iniciais e posteriormente o Postgres. Abaixo temos as configurações dos properties.

application-test.properties

```
spring.datasource.url=jdbc:h2:mem:testdb
spring.datasource.username=sa
spring.datasource.password=
spring.h2.console.enabled=true
spring.h2.console.path=/h2-console
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true
```

application.properties

```
spring.profiles.active=test
spring.jpa.open-in-view=true
```

application-mysql.properties

```
spring.datasource.username = root
spring.datasource.password = *senha*
spring.datasource.url =
jdbc:mysql://localhost:3306/*nomedatabase*?useTimezone=true&serverTimezone=UTC
```

spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL8Dialect
spring.jpa.show_sql = true

application-postgres.properties

spring.datasource.username = *username*
spring.datasource.password =*senha*
spring.datasource.url = *url*
spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.PostgreSQLDialect
spring.jpa.show-sql=true
#JPA
spring.jpa.hibernate.ddl-auto=update

Modelagem do Banco de Dados

Realizamos a modelagem do banco de dados em consonância com a descrição do desafio.

➔ **H2, MySQL e Postgres**

- **DataBase**: chocode
- **Tabelas:**
  - entregador:
    - id (PK);
    - nome;
    - email;
    - senha;
    - url_imagem;

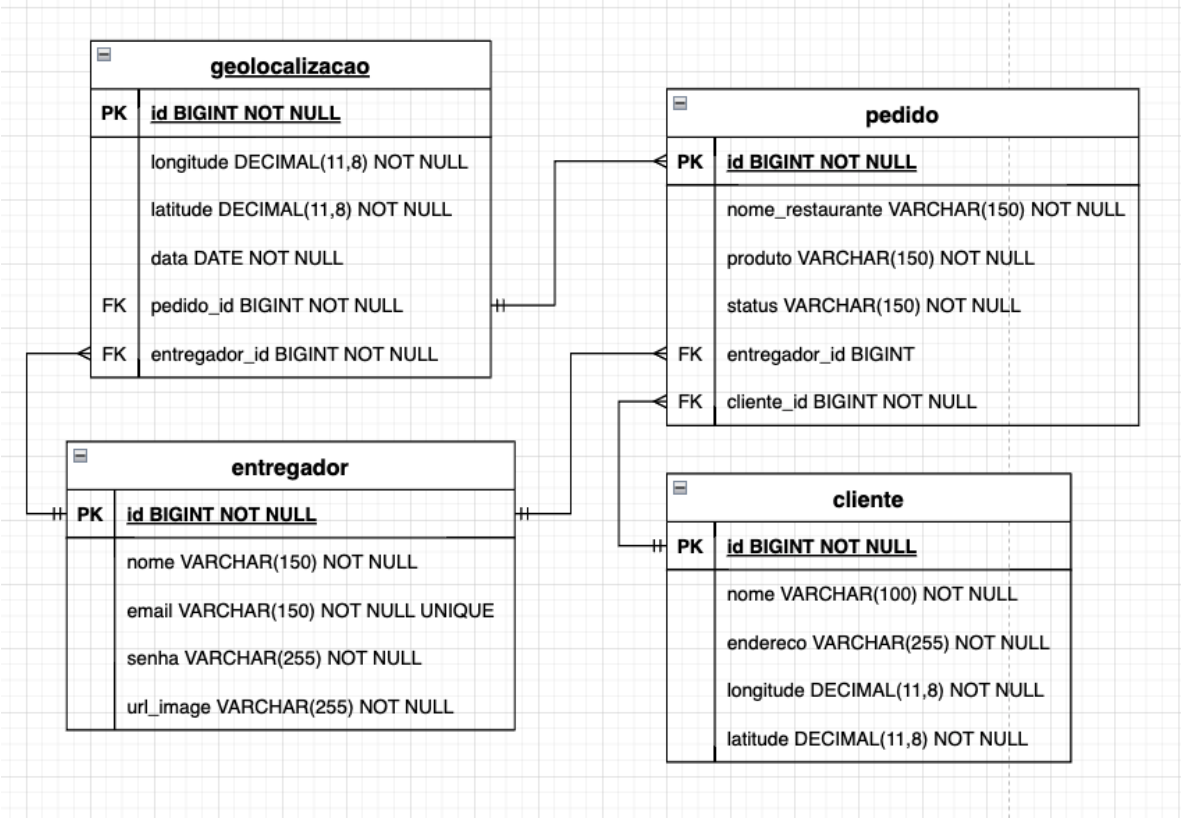
  - pedido:
    - id(PK);
    - nome_restaurante
    - status
    - produto
    - → cliente_id(FK)
    - → entregador_id(FK)
    - → geolocalizacao_id(FK)

  - cliente:
    - id(PK);
    - nome;
    - endereco;
    - longitude;
    - latitude;

  - geolocalizacao:
    - id(PK);
    - → pedido_id(FK);
    - → entregador_id(FK);
    - latitude;

- longitude;
- data;

Diagrama do Banco de Dados



Estrutura dos Pacotes

A estrutura dos pacotes, com as classes e interfaces, está descrita a seguir:

📦 Model @Entity

Entregador
Cliente
Pedido
Geolocalizacao

📦 DAO → JpaRepository

Entregador
Cliente
Pedido
Geolocalizacao

📦 DTO

GeolocalizacaoDTO
PedidoDTO
StatusDTO

📦 Services @Component

IPedidoService
PedidoServiceImpl
IEntregadorService
EntregadorServiceImpl
IGeolocalizacaoService
GeolocalizacaoServiceImpl

📦 Security → JWT

ChocodeEntryPoint
ChocodeFilter
ChocodeSecurityConfiguration
Token
TokenUtil

📦 Controller @RestController

PedidoController
EntregadorController
GelocalizacaoController
ClienteController

- 🔗 *Endpoints:*
  - - Endpoint para receber geolocalização do entregador
    POST
    "/geolocalizacao"
  - - Endpoint para alteração dos status dos pedidos
    PUT
    "/pedidos/{id}/status"
  - - Endpoint para consulta de pedidos
    GET
    "/pedido/listar"
  - - Endpoint para consultas de geolocalização por pedido
    GET
    "/pedidos/{id}/geolocalizacao"
  - - Endpoint para atribuição do entregador para o pedido
    PUT
    "/pedidos/{id_pedido}/entregador/{id_entregador}"

- - Endpoint para autenticação de entregador

  `POST`

  `"/entregador/login"`

Front End

A composição das telas segue a seguinte estrutura:

- ★ Autenticação do entregador
- ★ Consulta de pedidos
- ★ Atribuição de pedido pelo entregador
- ★ Alteração de status do pedido (cancelado/concluído)

Agenda do Grupo

Durante o desafio realizamos dailys com o Foodlover Danilo De Luca, mentorias com o professor Danilo da Gama Academy e reuniões extras para esclarecimentos de dúvidas.

Data	Horário	Assunto
22/03	14:00 - 15:00	Reunião com Danilo Ifood
23/03	13:40 - 14:20	Mentoria com Professor Danilo
Diário	14:45 - 15:00	Dailys com Danilo Ifood
25 e 28/03	10:20 - 11:00	Mentoria com Professor Danilo
30/03	13:20 - 14:20	Mentoria com Professor Danilo
25/03	13:00 - 13:45	Reunião para esclarecimento de dúvidas com Danilo Ifood

● Dúvidas

As principais dúvidas que surgiram durante o desenvolvimento do projeto:

- Utilização de banco em nuvem;
- Utilização do Cascade no Java;
- Relação no banco de dados do Entregador com Pedido e Geolocalização;
- Qual a melhor opção: Timestamp ou LocalDateTime na entidade Geolocalizacao;
- O email único na entidade Entregador de ter anotação @Column(unique=true);
- Sobre bloqueio de PR na main no git;
- Sobre fazer um PR da main para develop e atualizar a develop;
- Sobre fazer um pull para atualizar localmente e conflitos;

Anotações complementares

- Sobre pull no git:

```
### Importante garantir que comitou tudo na sua branch (git status) ###
1 git checkout main
2 git pull origin main
3 git checkout Daniel
4 git merge main

Primeiro traz a Main para sua branch, e depois da sua branch para a main
```

Apresentação