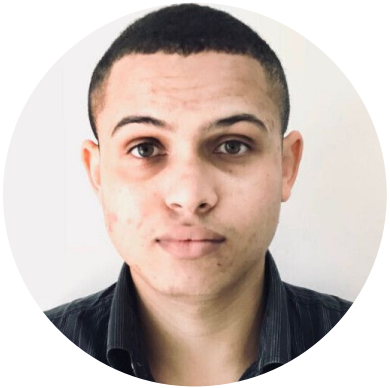
iLocation

A Telemetria das pessoas entregadoras do iFood

Os Verbosos - Grupo 1

Rogério Fidêncio

rogerio.fidencio@ifood.com.br



Lucas Rocha

rocha.lucas@ifood.com.br

Ana Beatriz Nunes

beatriz.nunes@ifood.com.br

Henrique Lima

henrique.lima@ifood.com.br

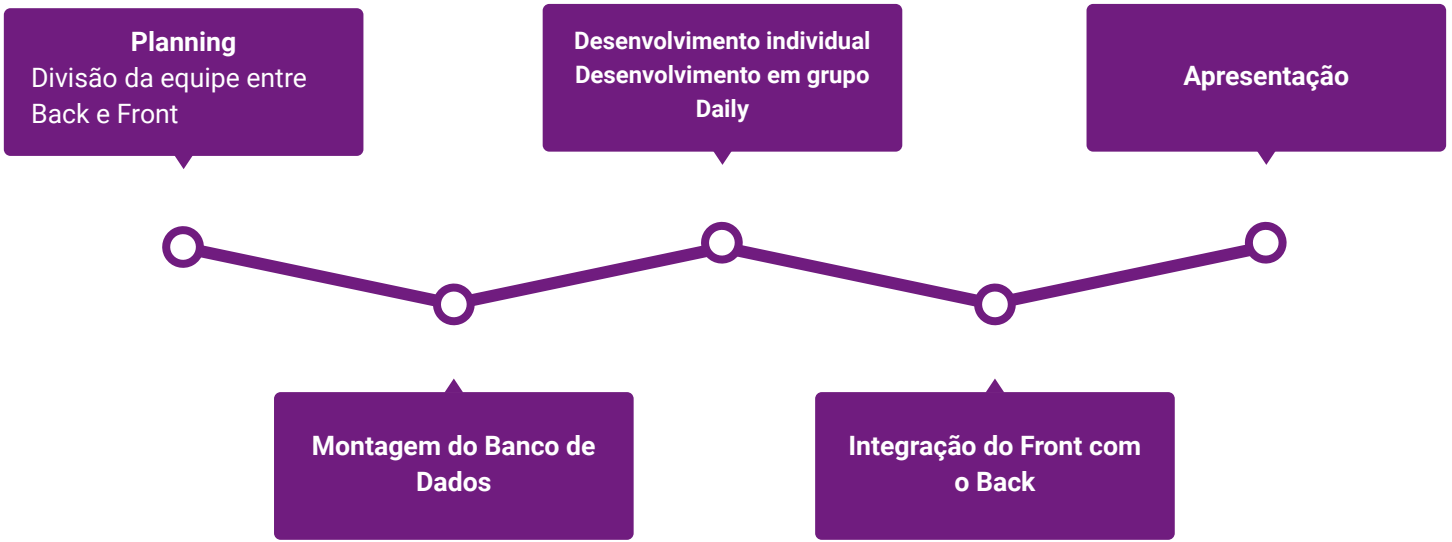
Carolina Amaral

amaral.carolina@food.com.br

Objetivo

Capturar e manter todo o histórico de telemetria (geolocalização) de um entregador para um determinado pedido

Visão geral


Planning
Divisão da equipe entre Back e Front
Desenvolvimento individual
Desenvolvimento em grupo
Daily
Apresentação
Montagem do Banco de Dados
Integração do Front com o Back

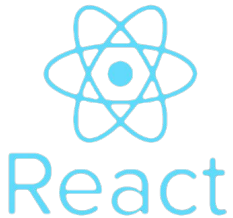
Organização no Trello

- **Padrão Kanban “to do, doing, done” + Revisão e Fase de testes**
- **Backlog**
- **Cards**
  - Detalhamento de tarefas
  - Priorização de tasks
  - Responsáveis
  - Data de entrega

Tecnologias utilizadas

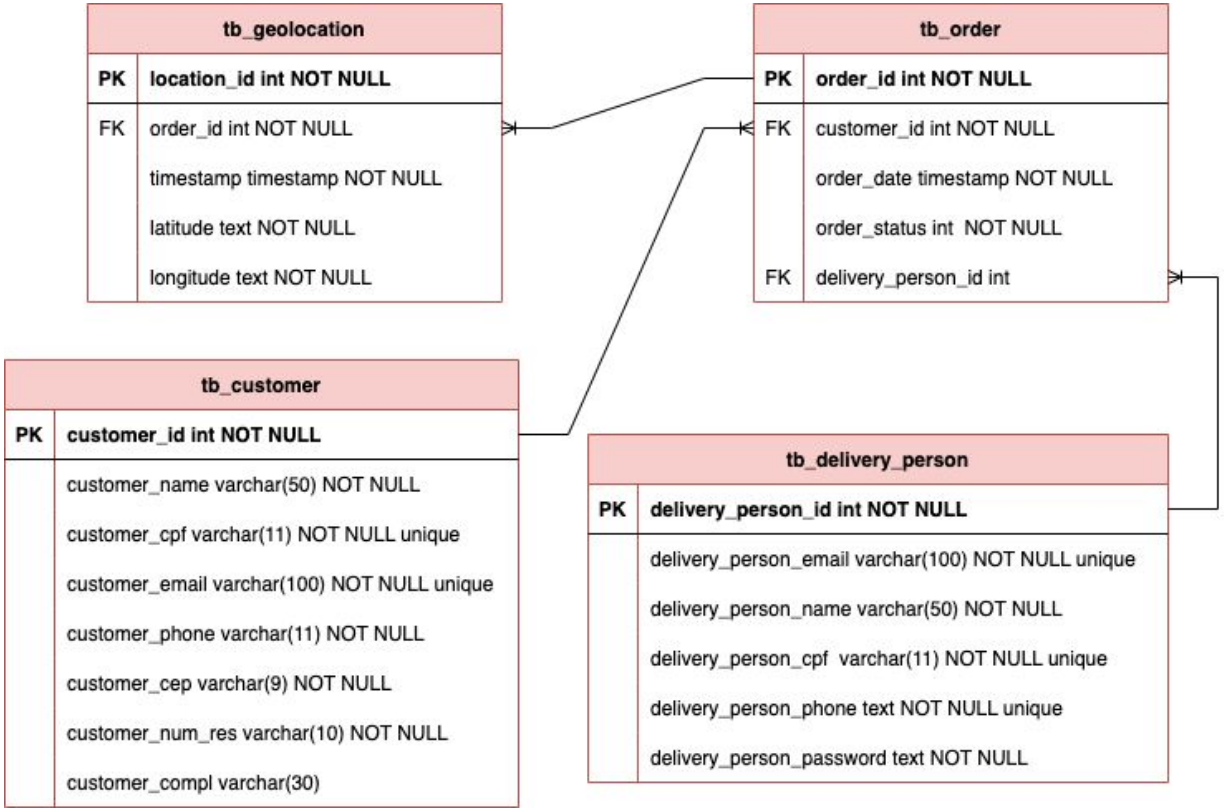
Back-end

Front-end



O PROJETO

O Projeto - BANCO DE DADOS

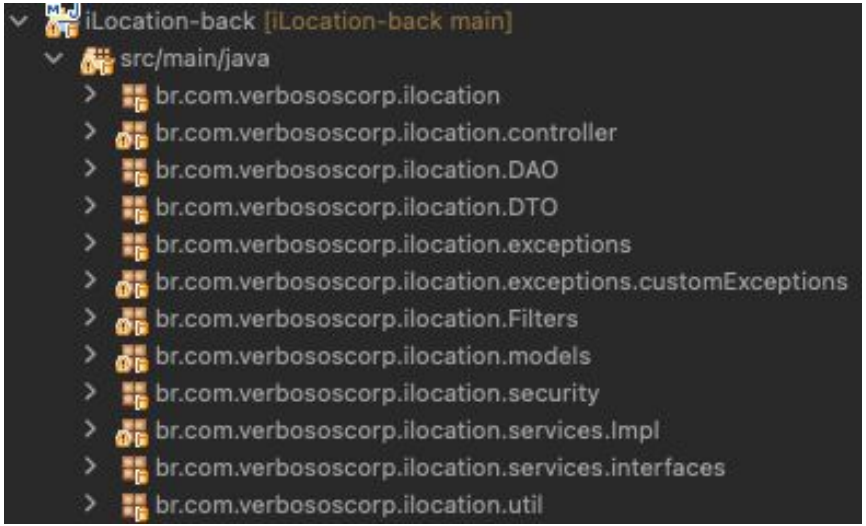


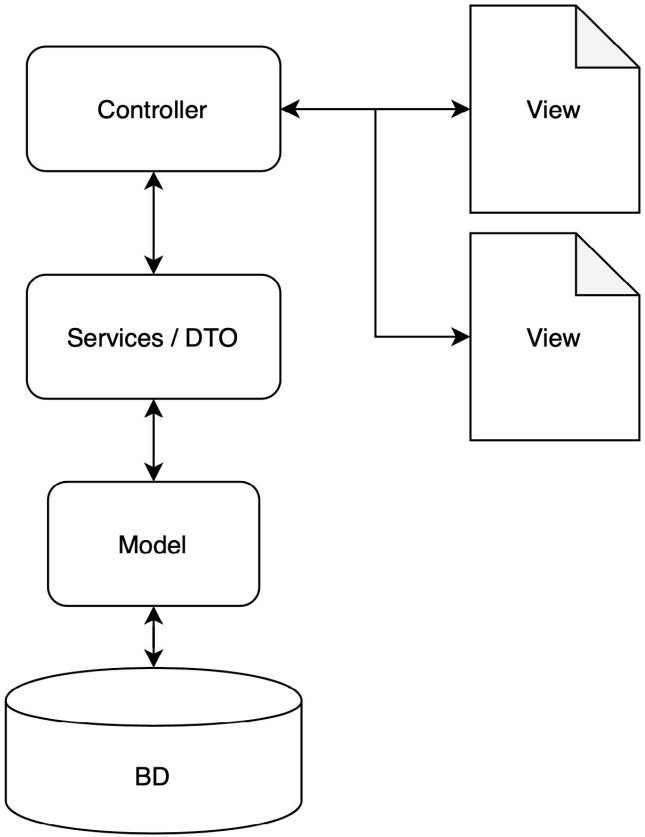
Na tabela “tb_order”
foi adicionado um índice parcial no campo de “order_status” para aumentar a performance ao buscar um pedido em andamento.

PostgreSQL

O Projeto - BACKEND

- Padrão MVC
- Divisão em camadas
- Divisão de responsabilidades
- Desacoplamento





O Projeto - BACKEND

ENDPOINTS MÍNIMOS

- Autenticação de entregador
- Consulta de todos os pedidos
- Atribuição do entregador para o pedido
- Alteração dos status dos pedidos
- Receber geolocalização do entregador
- Consultas de geolocalização por pedido
    - **Paginação**

ENDPOINTS EXTRAS

- Cadastrar Pessoa Entregadora
- Buscar pedido em andamento da pessoa entregadora logada
- Listar Clientes
- Consulta de Pedidos
    - Id
    - Status
    - Status Disponível
    - Pessoa Entregadora logada

O Projeto - BACKEND - Carol

```java
@Override
public Order assignDeliveryPerson(Integer orderID) throws DeliveryPersonNotAvailableException, OrderNotFoundException, OrderNotAvailableException {

    Integer userID = getContextData().getId();

    Optional<OrderDTO> orderValidation = dao.getCurrentOrderByDeliveryPersonId(userID);

    if (orderValidation.isPresent()) {
        throw new DeliveryPersonNotAvailableException();
    }

    Optional<Order> order = dao.findById(orderID);

    if (order.isEmpty()) {
        throw new OrderNotFoundException();
    }

    if (order.get().getStatus() != 0) {
        throw new OrderNotAvailableException();
    }

    Optional<DeliveryPerson> deliveryPerson = deliveryPersonDAO.findById(userID);

    order.get().setDeliveryPerson(deliveryPerson.get());
    order.get().setStatus(1);

    return dao.save(order.get());

}
```

```java
@PatchMapping("/assign/{orderID}")
public ResponseEntity<?> assignDeliveryPerson(@Validated @PathVariable Integer orderID) {
    try {
        service.assignDeliveryPerson(orderID);

    } catch (DeliveryPersonNotAvailableException | OrderNotAvailableException e) {
        throw new BadRequestException(e.getMessage());

    } catch (OrderNotFoundException e) {
        throw new ResourceNotFoundException(e.getMessage());

    } catch (Exception e) {
        throw new InternalServerErrorException(e.getMessage());
    }

    return ResponseEntity.noContent().build();
}
```

- Implementação dos Services
- Queries personalizadas
- DTO
- Controllers
- Exceções Personalizadas
- Testes

```java
@Query("select new br.com.verbososcorp.ilocation.DTO.OrderDTO" +
        "(order.id, order.status, order.deliveryPerson.id)" +
        " FROM Order as order " +
        "WHERE order.status = 1 " +
        "AND order.deliveryPerson.id = :id")
public Optional<OrderDTO> getCurrentOrderByDeliveryPersonId(@Param("id") Integer id);
```

O Projeto - BACKEND - Henrique

```java
@Override
public Authentication attemptAuthentication(HttpServletRequest request, HttpServletResponse response) throws AuthenticationException {
    try{
        DeliveryPersonAuthDTO user = new ObjectMapper().readValue(request.getInputStream(), DeliveryPersonAuthDTO.class);
        UsernamePasswordAuthenticationToken authenticationToken = new UsernamePasswordAuthenticationToken(user.getEmailOrPhone(), user.getPassword());

        userDTO = deliveryPersonService.findDeliveryPersonDTOByEmailOrPhone(user.getEmailOrPhone());

        return authenticationManager.authenticate(authenticationToken);

    } catch (IOException e) {
        response.setContentType(APPLICATION_JSON_VALUE);
        response.setStatus(401);
        ErrorMessage errorMessage = new ErrorMessage(401, new Date(), e.getMessage(), request.getServl
        try {
            new ObjectMapper().writeValue(response.getOutputStream(), errorMessage);
        } catch (IOException ex) {
            System.out.println(ex.getMessage());
        }
        return null;
    } catch (Exception e){
        response.setContentType(APPLICATION_JSON_VALUE);
        response.setStatus(500);
        ErrorMessage errorMessage = new ErrorMessage(500, new Date(), e.getMessage(), request.getServl
        try {
            new ObjectMapper().writeValue(response.getOutputStream(), errorMessage);
        } catch (IOException ex) {
            System.out.println(ex.getMessage());
        }
        return null;
    }
}
```

- Autenticação
- Implementação dos Services
- Queries personalizadas
- DTO
- Controllers
- Exceções Personalizadas
- Testes
- Spring Validation
- Paginação da geolocalização
- Front: Tracking e Integração com o Back

O Projeto - BACKEND - Rogério

```java
@PutMapping()
public ResponseEntity<?> registerGeoLocation(@Validated @RequestBody GeoLocation geoLocation){

    try {
        service.register(geoLocation);

    } catch (NoOrderAtributedToDeliveryPersonException e) {
        throw new BadRequestException(e.getMessage());

    } catch (Exception e) {
        throw new InternalServerErrorException(e.getMessage());
    }

    return ResponseEntity.status(HttpStatus.CREATED).build();
}
```

```java
@Entity
@Table(name = "tb_delivery_person")
public class DeliveryPerson {

    @Id
    @Column(name = "delivery_person_id")
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;

    @NotEmpty
    @Column(name = "delivery_person_name", nullable = false,
    private String name;

    @CPF
    @NotEmpty
    @Column(name = "delivery_person_cpf", nullable = false, length = 11)
    private String cpf;

    @Email
    @NotEmpty
    @Column(name = "delivery_person_email", nullable = false, length = 100)
    private String email;

    @NotEmpty
    @Column(name = "delivery_person_phone", nullable = false, length = 11)
    private String phone;

    @NotEmpty
    @Column(name = "delivery_person_password", nullable = false, columnDefinition = "TEXT")
    private String password;

    @OneToMany(mappedBy = "deliveryPerson")
    @JsonIgnoreProperties("deliveryPerson")
    private List<Order> orderGroup;
```

- Models
- Controllers
- Code reviews
- Indexação no banco de dados

O Projeto - FRONTEND

TELAS

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

EXTRA

- Apresentação de mapa na tela de tracking

O Projeto - FRONTEND - Lucas

```
import { useState, useEffect } from "react";

export default function useGetLocation() {
    const [coords, setCoords] = useState(null);

    useEffect(() => {
        function onSuccess(position) {
            setCoords([position.coords.latitude, position.coords.longitude, position.timestamp])
        }

        function onError(error) {
            console.log('Pesquisando localização...');
        }

        try {
            navigator.geolocation.watchPosition(onSuccess, onError, {enableHighAccuracy: false, timeout: 5000,
                maximumAge: 1})

        } catch (error) {
            onError(error);
        }
    }, [])

    return {coords}
}
```

- Telas iniciais da aplicação
- Auxílio na migração da tecnologia para Reactjs
- Página de Tracking
- Geolocation
- Mapa - biblioteca Leafletjs
- Integração dos endpoints do back com o Front

O Projeto - FRONTEND - Ana

- **Tela de login**
  - Inputs controlados
  - Validações básicas
  - Armazenamento de token no localstorage
  - Fluxo de telas do entregador com pedido em andamento
- **Tela de pedidos em aberto**
  - Acionamento de modal para iniciar rastreio
  - Uso de contexto para utilizar as informações do pedido por toda aplicação

```
try {
      const requestLogin = await fetch('https://ilocation.herokuapp.com/api/v1/login', {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json'
        },
        body: JSON.stringify(user)
      });

      const response = await requestLogin.json();

      if (response.status > 204) return;

      setAuthData({
        token: response.access_token
      });

      const requestOrder = await fetch('https://ilocation.herokuapp.com/api/v1/deliveryperson/currentorder', {
        method: 'GET',
        headers: {
          'Content-Type': 'application/json',
          Authorization: response.access_token
        }
      });

      const order = await requestOrder.json()

      if (order.status > 204) {
        navigate('/pedidos', { replace: true });
        return;
      }

      setOrderInfo(order);
      navigate('/rastreio', { replace: true });
    } catch (error) {
      console.log(error.message);
      navigate('/server_internal_error', { replace: true });
    }
  };
```

Tropeçamos e aprendemos...

1

Faltou análise das skills da equipe para definição da estratégia do frontend

- Problemas inesperados
- Troca de tecnologia no meio do projeto
- Frontend ficou pronto tardiamente
- Features adicionais ficaram de fora

2

Não usamos a prática TDD

- Testes ficaram para o final
- Construção dos testes ficou mais complicada

Mandamos bem!

1

Planning inicial

- Longa!
- Bom entendimento do projeto
- Divisão tarefas logo no início
- Começamos logo a codar

2

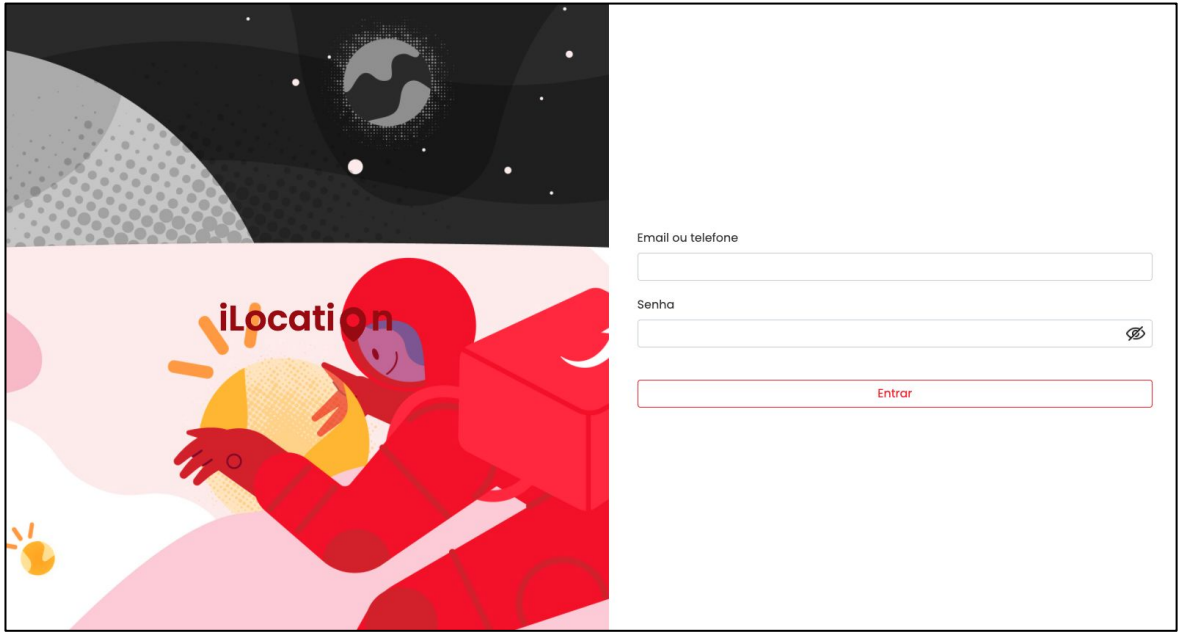
Trabalho em equipe

- Boa comunicação
- Disposição
- Ajuda mútua
- Resolução de problemas

VAMOS VER NA PRÁTICA?

Deploy

https://ilocation.vercel.app/



- Heroku
- Vercel

OBRIGADA!

iFood, Mentor Carlos Costa, Prof. Danilo, Banca, Gama