FIAP
ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

GUSTAVO CARVALHO – 552466

RONALD DE OLIVEIRA – 552364

NICOLY OLIVEIRA – 552410

NICOLLY DE ALMEIDA - 552579

VITOR TEIXEIRA – 552228

Nome do grupo: UniDevs

Nome da solução: GuinFlex

SPRINT 3

DOMAIN DRIVEN DESIGN

SÃO PAULO

2023

SUMÁRIO

1. JUSTIFICATIVA.....
2. OBJETIVO.....
3. DESCRIÇÃO DAS FUNCIONALIDADES.....
4. DIAGRAMA DE CLASSES.....
5. MODELO DO BANCO DE DADOS.....
6. RODANDO O PROGRAMA E APRESENTAÇÃO DE TELAS.

JUSTIFICATIVA

Após a análise, pesquisa e investigação do desafio proposto pela empresa Porto, compreendemos que o principal problema não é a aplicação de uma plataforma que conecte o cliente a porto, e sim, sobre a eficiência em acertar a escolha do melhor modal para resgatar os veículos dos clientes, sem necessitar totalmente da intervenção humana e retrabalhos. O foco para a solução está em como podemos melhorar este processo e torná-lo mais assertivo do ponto de vista da Porto e favorável ao cliente.

Em primeira instancia, temos como foco a criação de uma aplicação web, focada primeiramente no desempenho durante o uso de navegadores mobile. Isso se deve ao fato, de que em sua grande maioria, os clientes abririam o sinistro via dispositivos moveis, pois os mesmos, facilitam a coleta de informações sobre ocorrido por possuírem câmeras de fácil acesso.

Na aplicação, queremos que o usuário consiga realizar sua jornada de abertura do chamado, login, sinistro, acompanhamento, verificação, informações cadastradas, cancelamento e afins totalmente via website sem que se torne algo incompreensível e ineficaz para experiência.

OBJETIVOS

O ponto do projeto que demanda atenção redobrada é que o problema ocorre durante o processo de abertura de sinistro, demandando em torno de 60% dos casos intervenção humana e mais 10% a 15% de retrabalho do prestador de serviço. Chegando a ter casos de encaminhar o guincho incorreto para o local e precisar solicitar outro. Para evitar essas situações temos como objetivos;

- Nossa aplicação deve ser leve, intuitiva e agradável de se utilizar. É preciso que o cliente se sinta confortável e seguro para não gerar a necessidade de contato por outros canais de atendimento, como e-mails, telefonemas...
- A coleta de informações e sua qualidade tem extrema importância, pois a falta delas ou erros podem ocasionar a necessidade de intervenção humana e retrabalhos dos funcionários.

Descrição das funcionalidades

Durante a execução do programa Java, uma vez logado, serão apresentadas as seguintes funções:

Verificar suas informações: Irá permitir que o usuário que se encontra logado, verifique suas informações cadastradas no console.

Verificar histórico de chamados: Irá retornar as informações básicas de todos os chamados pertencentes ao usuário logado.

Alterar Status chamado: Irá permitir alterar o status de um chamado específico para ativo, fechado ou pausado.

Consulta informações de um chamado específico: Irá retornar as informações de um chamado específico do usuário.

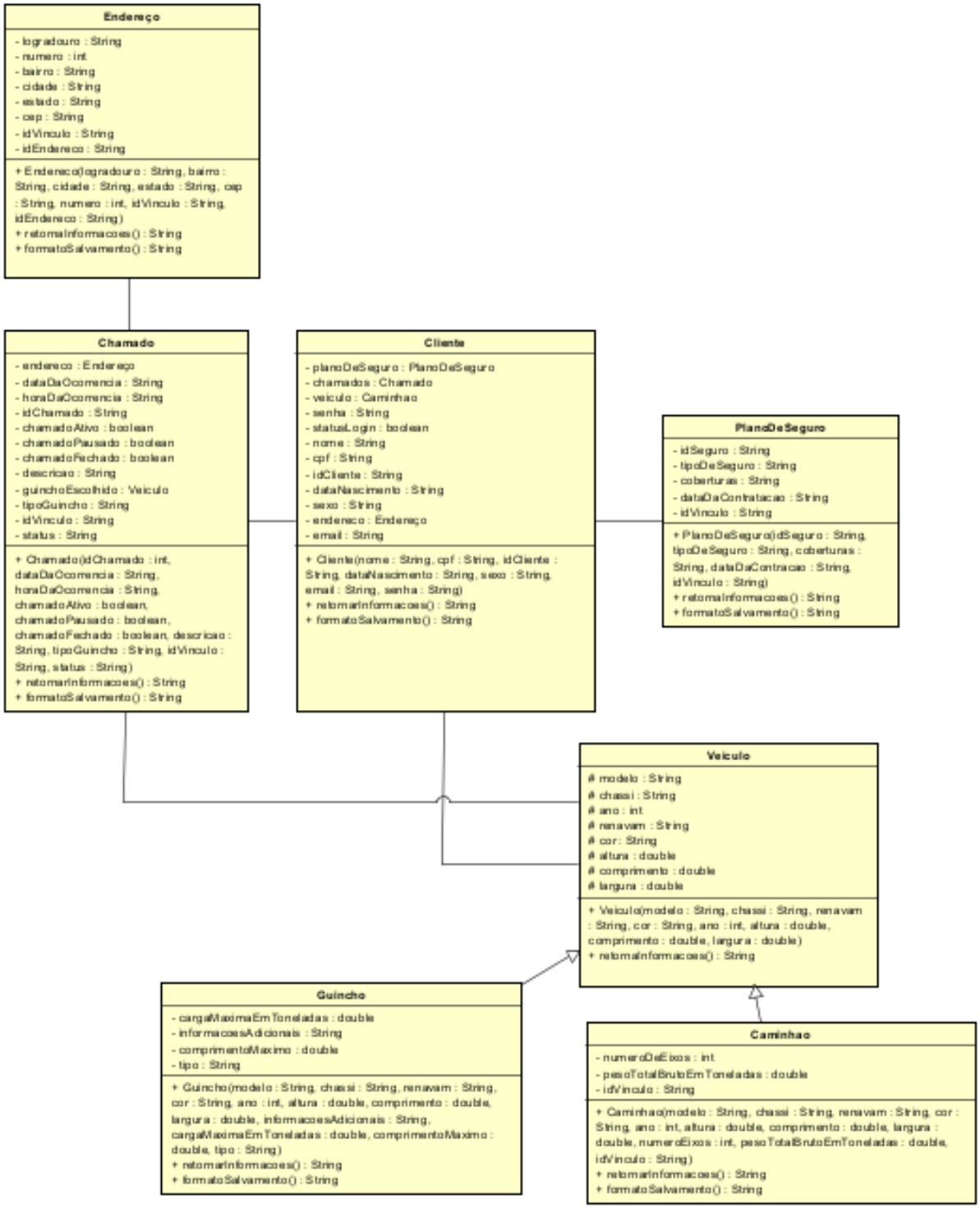
Enviar uma mensagem para o funcionário da Porto: Irá permitir o envio de uma mensagem, como o protótipo em Java procura simular as ações, a mensagem será armazenada em um arquivo em vez de enviada de fato.

Abrir um chamado: A funcionalidade permitirá a entrada de dados para que seja aberto um chamado, porém, não irá prosseguir caso o usuário já tenha um chamado em aberto.

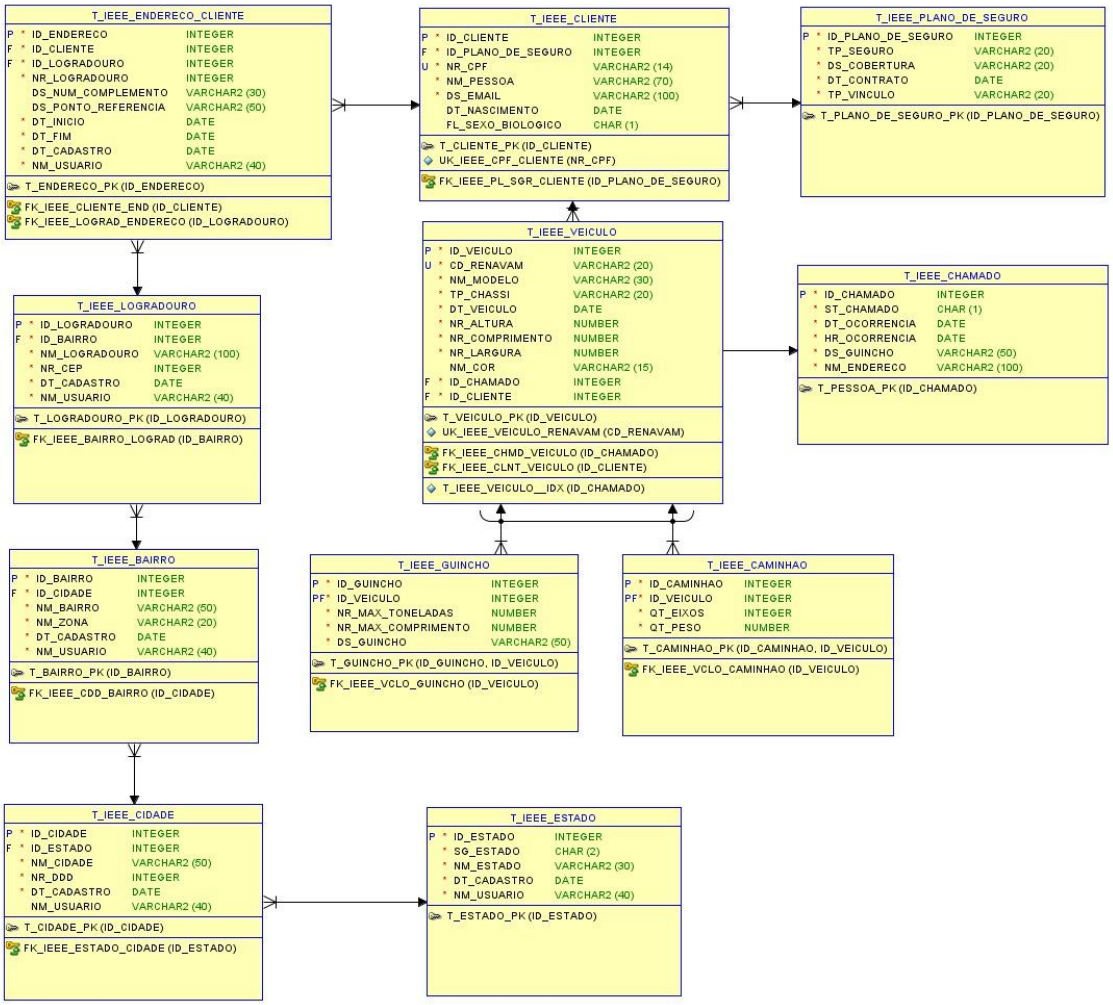
Deslogar sem salvar alterações: Irá encerrar a aplicação sem salvar as alterações que foram efetuadas nos arquivos que estão sendo usados para guardar os dados dos objetos.

Deslogar salvando alterações: Irá permitir que a aplicação encerre e que as informações que foram alteradas sejam salvas para futura execução.

Diagrama de Classes

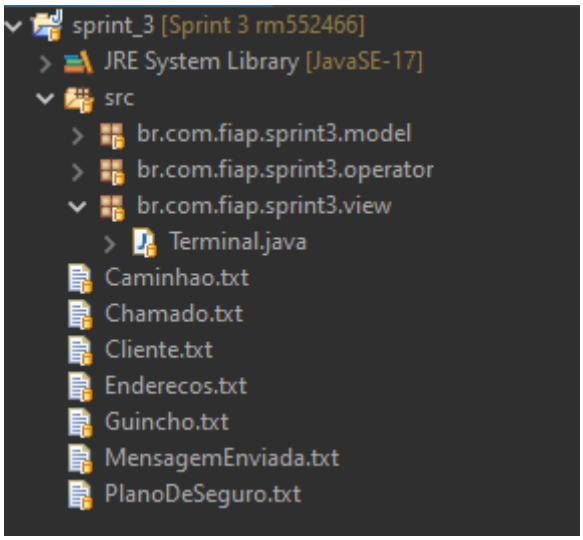


Modelo do banco de dados



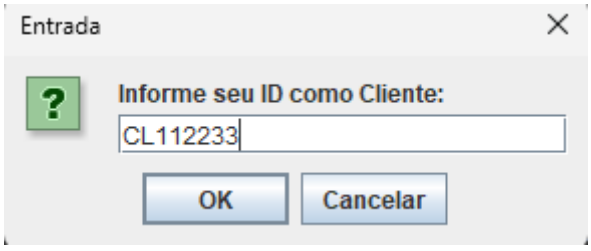
Rodando o programa e apresentação das telas

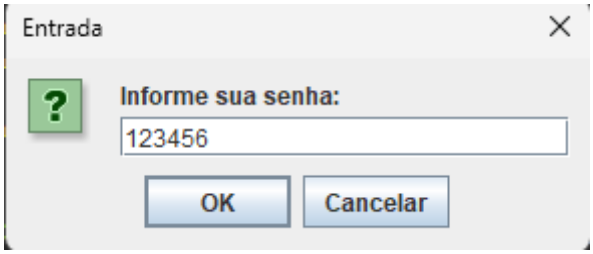
Para utilizar o programa, serão necessários os arquivos txt que irão simular a função de banco de dados, permitindo assim que objetos exemplos como clientes, tipo guincho e afins sejam criados ao rodar o arquivo “Terminal.java” no pacote “br.com.fiap.sprint3.view” que será o main.



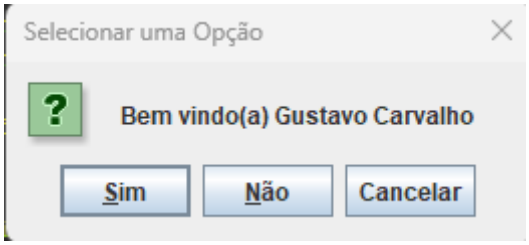
Quando o “Terminal.java” for iniciado, ele começará lendo todos os arquivos txt com exceção do “MensagemEnviada.txt” para criar os objetos que serão o ponto de partida para as interações em seguida.

Assim que terminar toda a leitura dos arquivos, criação dos objetos e armazena-los nos HashMaps que estão no SistemaDao.java no pacote “br.com.fiap.sprint3.operator”, irá ser apresentado o JOptionPane responsável por coletar o login e senha de um usuário que já esteja cadastrado para que seja permitido o uso do menu que irá realizar as operações por meio das funções descritas em tópico anterior. Um usuário exemplo válido que pode ser utilizado é o de ID “CL112233” e Senha “123456”.

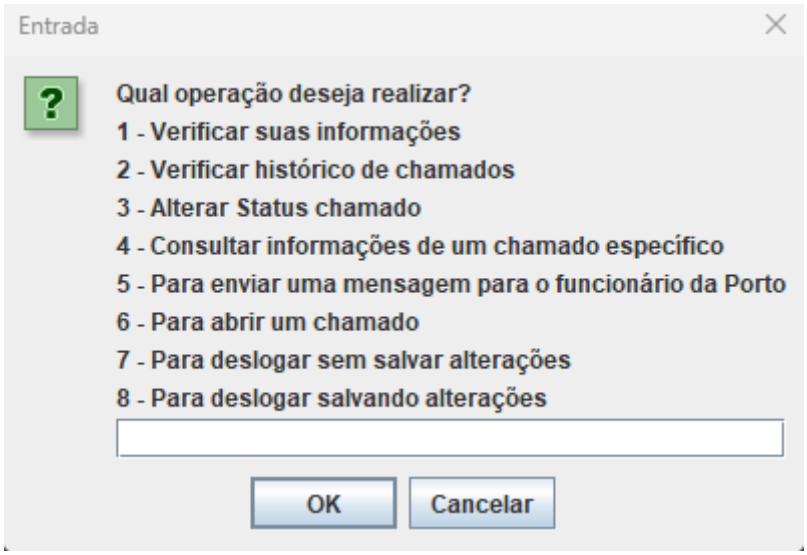




Uma vez inserido um usuário e senha válidos, será apresentada uma tela de bem vindo com o nome do usuário que efetuou o login.



Logo em seguida, irá aparecer o menu com as opções que podem ser utilizadas inserindo seu respectivo número.



Obs.: Algumas informações, por mais que sejam necessárias para a criação de objeto, não serão definidas pelo usuário com entrada de informação. Um exemplo disso, seria na função responsável por abrir o chamado, onde algumas informações são definidas pelo próprio programa, o guincho escolhido por exemplo será analisado o caminhão exemplo desse usuário e verificado dentro dos exemplos de guincho um que atenda a condição de decisão, informações como data e hora da abertura de chamado serão adotadas a do sistema no momento da abertura do chamado e os identificadores do chamado e endereço serão gerados aleatoriamente e verificados se já existe algum outro com o mesmo identificados para que não ocorra a tentativa de inserção de um chamado ou endereço com o mesmo identificador que outro já existente, seja pelo usuário logado ou de qualquer outro existente na aplicação.

Parte do código responsável por gerar um identificador novo

```
String idEndereco = "";
do {
    String prefixoEnd = "ED";
    int n = randI.nextInt(999999);
    idEndereco = prefixoEnd + n;
} while (enderecos.containsKey(idEndereco));//ID ENDEREÇO DEF
String idChamado = "";
do {
    String prefixoChamado = "CH";
    int n = randI.nextInt(999999);
    idChamado = prefixoChamado + n;
} while (chamados.containsKey(idChamado));//ID CHAMADO DEF
```

Parte do código responsável por pegar a data e hora

```
248     //DEFININDO DATA E HORA
249     DateTimeFormatter dtf = DateTimeFormatter.ofPattern("dd/MM/yyyy HH:mm:ss");
250     LocalDateTime now = LocalDateTime.now();
251     String[] arrDataHora = dtf.format(now).toString().split(" ");
252     String dataDaOcorrencia = arrDataHora[0];
253     String horaDaOcorrencia = arrDataHora[1];
254     //FINALIZANDO DATA E HORA
```