Sistema de Recomendação de Modal e Atendimento para Veículos Pesados (***ModalAI***)

Thiago Gil Camargo RM551211 – SPV

Cauã Couto Basques RM97755 – 1TDSS

Eduardo Moura Batageli RM97611 – 1TDSS

Kaique Agostinho de Oliveira RM550815 – 1TDSS

Rafael Minoro Itokazo RM99988 – 1TDSS

**Sumário**

[1. Introdução 4](#_Toc144909046)

[2. Descrição do Projeto 4](#_Toc144909047)

[3. Justificativa e Objetivos 4](#_Toc144909048)

[4. Funcionalidades do Sistema 5](#_Toc144909049)

[5. Tecnologias Utilizadas 5](#_Toc144909050)

[6. Diagrama UML 7](#_Toc144909051)

[7. Justificativa para as classes e seus métodos 7](#_Toc144909052)

[8. Pasta controller 7](#_Toc144909053)

[9. ApoliceController.java 7](#_Toc144909054)

[10. CargaController.java 7](#_Toc144909055)

[11. MenuController.java 8](#_Toc144909056)

[12. VeiculoController.java 8](#_Toc144909057)

[13. Pasta intelligence: 8](#_Toc144909058)

[14. AprendizadoDeMaquina.java. 8](#_Toc144909059)

[15. ModalAI.java. 8](#_Toc144909060)

[16. ModeloDeAprendizado.java e ModeloDeRegressao.java 8](#_Toc144909061)

[17. ProcessamentoDeImagem.java 8](#_Toc144909062)

[18. Pasta model: 8](#_Toc144909063)

[19. Apolice.java, Carga.java, Veiculo.java. 9](#_Toc144909064)

[20. Pasta services: 9](#_Toc144909065)

[21. CRUD.java 9](#_Toc144909066)

[22. ListaDeObjetos.java e Singleton.java 9](#_Toc144909067)

[23. Pasta view: 9](#_Toc144909068)

[24. ApoliceFormView.java, CargaFormView.java, MenuView.java, VeiculoFormView.java. 9](#_Toc144909069)

[25. Pasta test: 9](#_Toc144909070)

[26. ApoliceControllerTest.java, CargaControllerTest.java, MenuControllerTest.java. 10](#_Toc144909071)

[27. Pasta main: 10](#_Toc144909072)

[28. Main.java. 10](#_Toc144909073)

[29. Conclusão 10](#_Toc144909074)

1. Introdução

Este documento tem como objetivo apresentar a documentação do projeto de um sistema para recomendação de modal, voltado ao atendimento de veículos pesados. Neste projeto, serão apresentadas as tecnologias de inteligência artificial que serão utilizadas, a forma como será feito o processamento de imagem para identificar as características dos veículos e cargas, recomendando assim o modal mais adequado para cada necessidade.

1. Descrição do Projeto

O projeto consiste em criar um sistema que utilize algoritmos de aprendizado de máquina, para que seja identificado de forma mais precisa o tipo de modal adequado para cada situação dentro da categoria de veículos pesados. Será utilizado também o processamento de imagem para identificar as características dos veículos como: tipo de carroceria, se o chassi é alongado ou não, comprimento do veículo, altura do veículo, peso sem carga, peso com carga, quantidade de eixos, entre outras variáveis. O sistema poderá ser integrado com as informações da apólice de seguro do cliente, para ter acesso a informações adicionais, como peso máximo permitido por lei.

1. Justificativa e Objetivos

O projeto visa reduzir o esforço operacional humano na escolha do modal de atendimento para veículos pesados. A ideia é utilizar da inteligência artificial e processamento de imagem para identificar as características dos veículos e cargas, indicando assim o modal mais adequado para cada situação. O objetivo principal é aumentar a eficiência do processo e melhorar a experiência do usuário, tornando-o mais ágil, mais eficaz e prevenindo futuros erros.

1. Funcionalidades do Sistema

- Identificação automática das características dos veículos pesados e suas cargas;

- Recomendação do modal de atendimento mais adequado para cada situação;

- Integração com as informações da apólice de seguro do cliente para acesso a informações adicionais, como peso máximo permitido por lei;

- Integração com sistemas de logística e aplicativos mobile;

- Envio de informações em tempo real sobre o andamento do atendimento para o cliente.

1. Tecnologias Utilizadas

A linguagem de programação **Python** será imprescindível para a realização do trabalho devido à sua sintaxe clara e concisa. Ela será utilizada para a análise de dados, automação de tarefas e desenvolvimento web, facilitando e automatizando os processos tanto para o cliente quanto para o operador da Porto.

**Java** é a linguagem que será utilizada para o desenvolvimento do aplicativo servidor, sistemas embarcados e dispositivos móveis. Essa linguagem será muito boa para o desenvolvimento do aplicativo que vai falar com o usuário.

A biblioteca **OpenCV** será utilizada na parte de processamento de imagem, fornecendo um conjunto abrangente de algoritmos e ferramentas para processar imagens e vídeos em tempo real, tornando a escolha do modal ainda mais assertiva.

Algoritmos de Aprendizado de Máquina, como Redes Neurais Convolucionais (**CNNs**), serão utilizados no processamento das imagens, tendo a capacidade de analisar e comparar dados através delas. Essa tecnologia também consegue identificar caso haja falhas nos veículos (que muitas vezes acabam não sendo informadas pelo segurado no momento da solicitação), sendo bastante eficaz em tarefas computacionais que envolvem o processamento de imagens.

Os serviços em nuvem, como **AWS** (*Amazon Web Service*) ou **Google Cloud**, serão utilizados pois oferecem uma ampla variedade de serviços, incluindo o armazenamento dos dados, processamento de dados, banco de dados, análise dos dados, entre outros. Esses serviços também permitem que a Porto utilize a computação em nuvem sem que ela precise investir grandes fortunas com *hardwares* ou infraestrutura em rede.

A modelagem das classes do projeto será feita a partir do diagrama de MER dentro do aplicativo da Oracle, denominado **SQL Developer Data Modeler**. Nele, serão realizados os modelos estruturais e lógicos do projeto, que possibilitarão o desenvolvimento dentro das linguagens mencionadas anteriormente.

1. **Diagrama** UML

A modelagem das classes do projeto pode ser visualizada no diagrama de classes UML abaixo:

Diagrama, Esquemático

Descrição gerada automaticamente

1. Justificativa para as classes e seus métodos
2. Pasta controller:
3. ApoliceController.java: Esta classe é responsável por gerenciar as operações relacionadas às apólices de seguro. Ela lida com a criação, leitura, atualização e exclusão de apólices, bem como a solicitação de recomendações com base em apólices específicas.
4. CargaController.java: Similar ao ApoliceController, esta classe gerencia as operações relacionadas às informações de carga. Ela permite criar, ler, atualizar e excluir dados de carga e pode ser usada em conjunto com o aprendizado de máquina para recomendações.
5. MenuController.java: O MenuController é responsável por controlar o menu principal do sistema. Ele exibe opções para o usuário e coleta as escolhas do usuário, direcionando o fluxo do programa com base nessas escolhas.
6. VeiculoController.java: Essa classe gerencia informações sobre veículos. Ela permite a criação, leitura, atualização e exclusão de informações de veículos e é essencial para determinar quais veículos são recomendados para o transporte de cargas.
7. Pasta intelligence:
8. AprendizadoDeMaquina.java: Esta classe é o coração da inteligência do sistema. Ela implementa o aprendizado de máquina e é responsável por treinar modelos com base nos dados disponíveis e fazer recomendações de modal com base nesses modelos.
9. ModalAI.java: A classe ModalAI utiliza os resultados do aprendizado de máquina para calcular as recomendações de modal de atendimento. Ela é uma parte crucial da inteligência do sistema.
10. ModeloDeAprendizado.java e ModeloDeRegressao.java: Essas classes podem estar envolvidas no processo de treinamento de modelos de aprendizado de máquina, contribuindo para a precisão das recomendações.
11. ProcessamentoDeImagem.java: Esta classe é responsável pelo processamento de imagens dos veículos pesados. Ela identifica características dos veículos por meio de processamento de imagem, o que é valioso para a geração de recomendações precisas.
12. Pasta model:
13. Apolice.java, Carga.java, Veiculo.java: Essas classes de modelo representam informações básicas sobre apólices, cargas e veículos. Elas armazenam os dados essenciais relacionados a essas entidades e servem como estruturas de dados fundamentais para o sistema.
14. Pasta services:
15. CRUD.java: Esta classe fornece as operações básicas de criação, leitura, atualização e exclusão de objetos. É reutilizada por diferentes partes do sistema para gerenciar diferentes tipos de objetos de forma genérica.
16. ListaDeObjetos.java e Singleton.java: Essas classes de serviço podem fornecer funcionalidades adicionais conforme necessário durante o desenvolvimento. Elas podem ser úteis para manter listas de objetos e garantir que exista apenas uma instância de determinada classe, respectivamente.
17. Pasta view:
18. ApoliceFormView.java, CargaFormView.java, MenuView.java, VeiculoFormView.java: Essas classes fazem parte da camada de visualização do sistema. Elas são responsáveis por exibir informações ao usuário, coletar entradas do usuário e permitir a interação do usuário com o sistema por meio de formulários e menus.
19. Pasta test:
20. ApoliceControllerTest.java, CargaControllerTest.java, MenuControllerTest.java: Essas classes de teste são usadas para testar as funcionalidades das classes de controle correspondentes. Elas garantem a qualidade e o funcionamento correto das classes de controle por meio de testes automatizados.
21. Pasta main:
22. Main.java: Este é o ponto de entrada principal do programa. Ele pode ser usado para iniciar e executar o sistema quando o programa é executado.
23. Conclusão

O sistema desse projeto irá desenvolver de forma automatizada a recomendação do modal mais adequado a partir da solicitação do segurado, reduzindo possíveis erros humanos e tornando todo o processo mais fácil e também mais assertivo. Através da modelagem das classes apresentadas neste documento, será possível desenvolver todo um software correto utilizando das linguagens e tecnologias que foram apresentadas.

O foco principal no desenvolvimento deste trabalho será o de atenuar os aborrecimentos advindos dos clientes pelo uso do fator humano na solicitação dos serviços emergenciais da Porto, melhorando o seu relacionamento com o usuário segurado.