FIAP – FACULDADE INFORMATICA ADMINISTRACAO PAULISTA

Análise e Desenvolvimento de Sistemas

AFN ACESSIBILIDADES

Tela Interativa Para Pessoas Com Deficiência

Sprint 4 -Domain Driven Design Using Java

Bruno Cesar da Silva – RM560499 – 1tdsq;

Felipe Olecsiuc Damasceno – RM559433 - 1tdspr;

Natália de Oliveira Santos – RM560306 - 1tdspr;

Sumário

Descritivo

AFN Acessibilidades O projeto AFN Acessibilidades tem como objetivo desenvolver um sistema de tela interativa acessível, que fornecerá informações detalhadas sobre as linhas de metrô da ViaMobilidade, com foco em acessibilidade e inclusão. A proposta surge da necessidade de melhorar a experiência de navegação e orientação para pessoas com diferentes tipos de deficiência (visual, auditiva, física ou cognitiva) que utilizam o transporte público. O transporte é um serviço essencial, e é imperativo que ele esteja disponível para todos de forma equitativa, garantindo que ninguém seja excluído por conta de barreiras físicas ou de comunicação.

A solução proposta envolve a criação de uma interface interativa, intuitiva e inclusiva, que utiliza uma combinação de recursos visuais, auditivos e táteis para facilitar o uso. O sistema contará com uma tela sensível ao toque, botões de acessibilidade com respostas em áudio, e ajustes de contraste e tamanho de texto para usuários com deficiência visual. Além disso, o sistema integrará um recurso de Text-to-Speech para converter texto em áudio e um AudioGuia, que fornecerá orientações em diferentes idiomas e para diferentes estações de metrô. Além de fornecer também o recurso do Speech-to-Text que converte o áudio em texto, ajudando pessoas com alguma limitação que atrapalhe a digitação.

Objetivos principais:

1. Garantir a acessibilidade para pessoas com deficiência no uso do transporte público.

2. Facilitar a navegação pelas informações das linhas de metrô de forma inclusiva e sem barreiras.

3. Promover a segurança dos usuários, fornecendo informações claras e compreensíveis.

4. Fornecer suporte multimodal, utilizando texto, áudio e tátil, adaptando-se às necessidades de diferentes usuários.

Este projeto visa promover um ambiente inclusivo e acessível, onde todos os passageiros possam ter uma experiência de transporte eficiente e segura, sem restrições de acessibilidade.

Esse conjunto de classes, apresentado no diagrama no final desse documento, modela adequadamente as funcionalidades requeridas, permitindo que o sistema seja ampliado para atender novas demandas e integrando ferramentas que promovam uma interação inclusiva e eficaz.

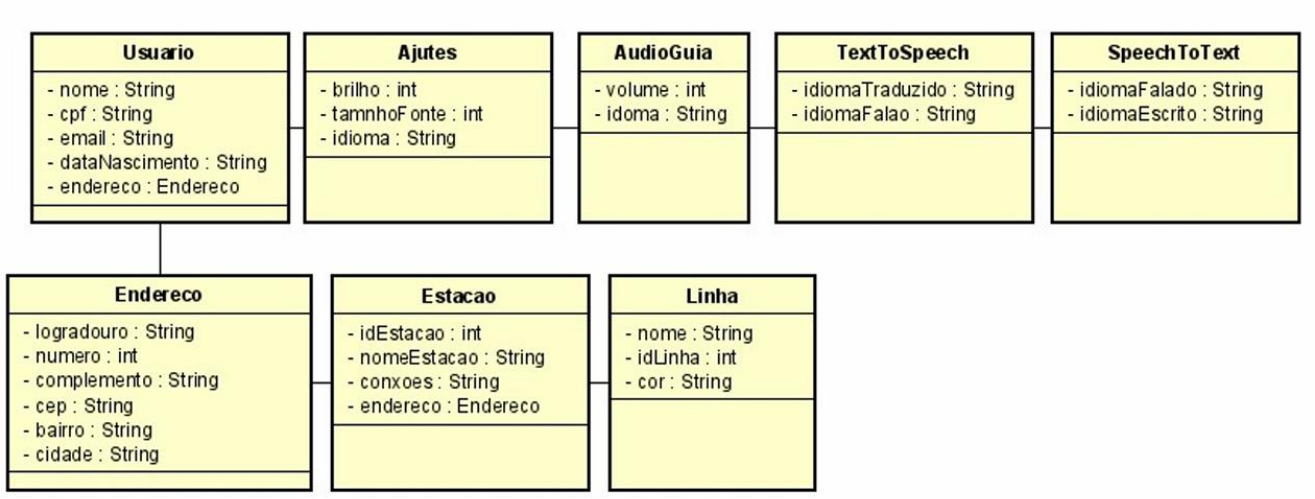
Justificativa: O projeto AFN Acessibilidades justifica-se pela necessidade urgente de tornar o transporte público mais acessível e inclusivo para todos os usuários, independentemente de suas habilidades. Muitas pessoas enfrentam desafios significativos ao tentar acessar informações nas estações de metrô devido a barreiras físicas ou de comunicação. Este sistema visa eliminar essas barreiras, garantindo que

pessoas com deficiência visual, auditiva, física ou cognitiva possam navegar de maneira eficiente e segura.

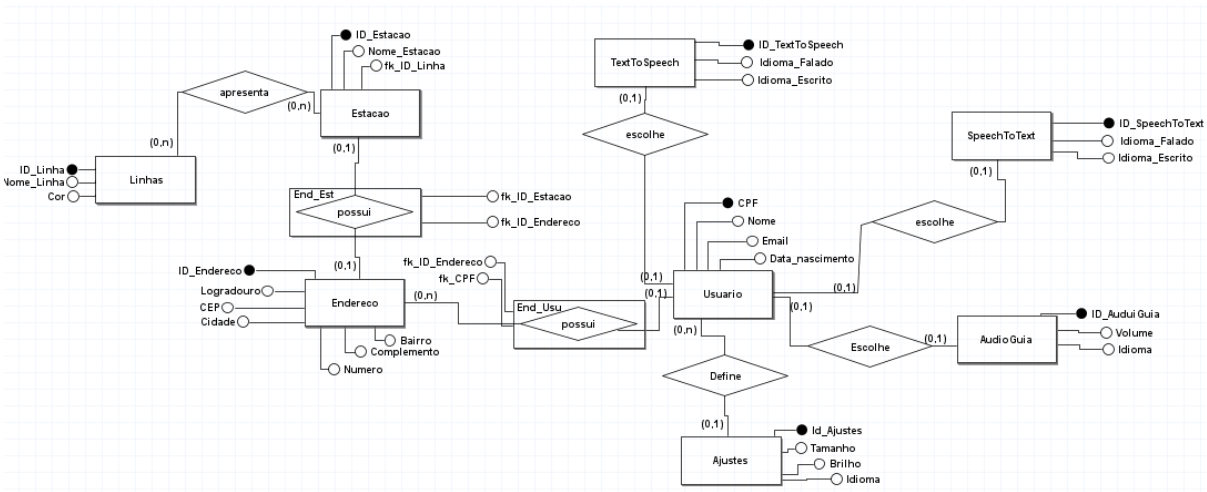
Com a implementação de uma interface multimodal e ferramentas adaptativas, o projeto busca não apenas facilitar a locomoção dos usuários, mas também promover a inclusão social e garantir que todos tenham acesso igualitário às informações.

A utilização de tecnologias como botões de acessibilidade, feedback em áudio e ajustes de contraste e texto alinha-se com as melhores práticas de design inclusivo. Ao modelar o sistema com classes que refletem essas funcionalidades, o projeto assegura que suas soluções atendam de maneira prática e eficiente às necessidades dos usuários, validando a entrega e alinhando os objetivos com a realidade do contexto proposto.

Diagrama de Classes



Modelagem de Dados



Descrição das Principais Funcionalidades

A solução AFN é um sistema desenvolvido em Java para auxiliar usuários no transporte metroviário. O sistema possui funcionalidades que utilizam lógica computacional para facilitar a

navegação e interação dos usuários, incluindo:

• Cadastro e Autenticação de Usuários: Implementação de um banco de dados para armazenamento de usuários.

• Integração com Speech-To-Text e Text-To-Speech: Permite a interação por comandos de voz.

• Localização de Estações: Permite ao usuário encontrar a estação mais próxima.

• Sistema de AudioGuia: Fornece informações audíveis sobre o transporte e direções.

• Conexão com Banco de Dados: Gerenciamento de informações por meio de uma camada DAO (Data Access Object).

Metodos lógicos

1**. Volume**: esse método estabelece um mínimo e máximo do volume do áudio guia, o áudio varia de 0 a 100. Caso o usuário insira um valor fora dessa escala o sistema retorna uma mensagem de erros.

```
public void setVolume(int volume) {  1 usage
    if (volume >= 0 && volume <= 100) {
        this.volume = volume;
        System.out.println("Volume ajustado para: " + volume);
    } else {
        System.out.println("Erro: O volume deve estar entre 0 e 100.");
    }
```

2. Brilho: define uma escala do brilho da tela, onde pode variar de 25 a 100, caso o usuário insira um valor menor que 25 o sistema automaticamente define o brilho

como 25, já se for inserido um valor maior que 100 automaticamente o brilho é definido como 100.

```
public void setBrilho(int brilho) {  1 usage
    if (brilho < 25) brilho = 25;
    if (brilho > 100) brilho = 100;
    this.brilho = brilho;
}
```

3. Cep: Esse método valida se o cep inserio é valido, o valor inserido deve ser no modelo 00000-000. Caso esteja valido a mensagem “Cep valido” aparece no terminal, já se não for valido a mensagem “Cep invalido” é exibida.

```
public String validarCEP() {  no usages
    if (this.cep.matches( regex: "\\d{5}-\\d{3}")) {
        return "CEP válido";
    } else {
        return "CEP inválido";}
}
```

4. Tamanho Fonte: Esse método permite que a fonte do texto do usuário seja definido. Ele estabelece que:

-se a entrada for menor que 8 o sistema automaticamente ira definir a fonte como 8 e exibirá a mensagem com “Tamanho da fonte muito pequeno! Ajustando para 8”;

-se a entrada for maior que 72 o sistema define a fonte como 72 e exibe a mensagem “Tamanho da fonte muito grande! Ajustando para 72”;

- Já se o valor inserido pelo usuário estiver dentro da escala de 8 a 72 ele apenas define a fonte com o valor escolhido

```
public void setTamanhoFonte(int tamanhoFonte) {  1 usage
    if (tamanhoFonte < 8) {
        System.out.println("Tamanho da fonte muito pequeno! Ajustando para 8.");
        this.tamanhoFonte = 8; // Ajusta para o mínimo permitido
    } else if (tamanhoFonte > 72) {
        System.out.println("Tamanho da fonte muito grande! Ajustando para 72.");
        this.tamanhoFonte = 72; // Ajusta para o máximo permitido
    } else {
        this.tamanhoFonte = tamanhoFonte; // Mantém o valor se dentro do intervalo
    }
```

Tabela de Endpoints

URL	Description
/ajustes/{idioma}	DELETE
/ajustes	POST (consumes: application/json)
/ajustes	GET (produces:application/json)
/ajustes/{idioma}	PUT (consumes: application/json)
/audioguia/{idioma}	DELETE
/audioguia	POST (consumes: application/json)
/audioguia	GET (produces:application/json)
/audioguia/{id}	PUT (consumes: application/json)
/speechToText/idiomaFalado	DELETE
/speechToText	POST (consumes: application/json)
/speechToText	GET (produces:application/json)
/speechToText	PUT (consumes: application/json)
/tts/{idiomaTraduzido}	DELETE
/tts	POST (consumes: application/json)
/tts	GET (produces:application/json)
/tts/{idiomaTraduzido}	PUT (consumes: application/json)
/usuario/{cpf}	DELETE
/usuario	POST (consumes: application/json)
/usuario	GET (produces:application/json)

/speechToText	PUT (consumes: application/json)
/tts/{idiomaTraduzido}	DELETE
/tts	POST (consumes: application/json)
/tts	GET (produces:application/json)
/tts/{idiomaTraduzido}	PUT (consumes: application/json)
/usuario/{cpf}	DELETE
/usuario	POST (consumes: application/json)
/usuario	GET (produces:application/json)
/usuario	PUT (consumes: application/json)
/endereco/{cep}	DELETE
/endereco	POST (consumes: application/json)
/endereco	GET (produces:application/json)
/endereco	PUT (consumes: application/json)
/linha/{idLinha}	DELETE
/linha	POST (consumes: application/json)
/linha	GET (produces:application/json)
/linha	PUT (consumes: application/json)
/estacao/{idEstacao}	DELETE
/estacao	POST (consumes: application/json)
/estacao	GET (produces:application/json)
/estacao	PUT (consumes: application/json)
/hello	GET (produces:text/plain;charset=UTF-8)

Protótipo

Abaixo está a tela do protótipo do frontend que chamam o backend do java:

Procedimetos para rodar aplicação

- JDK deve ser o 21
- Instalar o Quarkus Tools para rodar corretamente