1. Front-end

1.1. Descrição

O front-end será responsável por coletar os dados do cliente, incluindo nome, endereço, bairro e uma lista de telefones. A lista de telefones será tratada como uma tabela separada, vinculada ao cliente.

1.2. Funcionalidades

Inserção de Dados de Clientes:

Campos: Nome, Endereço, Bairro, Lista de Telefones.

Validação do Nome: Maior que 10 caracteres e sem dados redundantes.

Validação da Lista de Telefones: Garantir vinculação 1 para 1.

1.3. Tecnologias Utilizadas

Angular para facilitar o desenvolvimento.

2. Back-end

2.1. Descrição

O back-end será responsável por receber e validar os dados dos clientes provenientes do front-end, realizando a integração através de APIs Rest.

2.2. Funcionalidades

Inserção de Dados de Clientes:

Validação do Telefone: Não nulo e em formato válido.

2.3. Tecnologias Utilizadas

Java 17 como linguagem principal.

Spring Boot na versão 3.2.3 para facilitar o desenvolvimento das apis no back-end.

3. Testes Unitários

3.1. Descrição

Serão criados testes unitários para as validações do back-end.

3.2. Funcionalidades

Testes para a Validação do Telefone:

Casos de teste para telefone nulo.

Casos de teste para telefone em formato inválido.

3.3. Tecnologias Utilizadas

JUnit para a criação e execução de testes unitários.

4. Documentação

4.1. JavaDoc

Utilização do JavaDoc para documentar o código-fonte.

Documentação detalhada para as classes, métodos e validações.

4.2. Deploy da Aplicação

Descrição passo a passo de como realizar o deploy da aplicação.

Configurações necessárias para o ambiente de produção.

5. Repositório no GitHub

5.1. Descrição

Todo o código-fonte, documentação e recursos serão armazenados no GitHub.

5.2. Repositório

O repositório estará disponível em <https://github.com/Januario86/totvs.git>

Este documento serve como guia para o desenvolvimento e implementação das funcionalidades mencionadas, garantindo uma arquitetura clara e organizada.

6. Implantação

API – JSON SERVER

Para a realização das integrações será utilizado o Json Server

1. Instalando: npm install -g json-server

2. Para rodar o servidor copie o arquivo db.json para a raiz do projeto e

execute o seguinte comando: json-server --watch db.json

Node.js

Node.js: Download e instalação Angular CLI: Instale globalmente utilizando o seguinte comando no terminal: npm install -g @angular/cli

Backend

Necessário ter o java 17+ instalado e o maven, para efetuar tal instalação seguir os direcionamentos

Maven - <https://medium.com/@januario86/o-que-%C3%A9-o-maven-e-como-instalar-1d5e9f29ac4c>

Java - <https://medium.com/@januario86/configurando-vari%C3%A1veis-de-ambiente-no-java-4fc0bb2dec83>

7. Divisão de pacotes

Foi divido em pacotes para facilitar manutenção e evolução da aplicação, para assim sempre manter o processo de melhoria contínua.

Sendo assim nossa api Rest ficou divida dessa maneira:

Texto

Descrição gerada automaticamente com confiança média

No pacote Config,

SpringDocConfig - Documentação da api através do springDoc(Swagger)  
CorsConfig - Responsável pelo controle dos fluxos através do CORS

AppConfig - Bean para o RestTemplate para execução da leitura e integração com os dados do jsonServer.

Exceptions

Um Handler global, seguindo os padrões de boas práticas para centralizar as exceções via requisição em todas as controllers que possuem no sistema ou que serão implementadas futuramente

Request/Controller/Service/Model/Response/Message

Request – Definição dos atributos para a request que será realizada via api

Controller – Responsável por receber a requisição e direciona-la ao servisse

Service – Disponbilza a comunicação com a camada do servidor e inclui também as regras de negócio da aplicação

Reponse – Utilizado o padrão Response Entity do framework spring para que seja devolvido um padrão contendo também o HttpStatus code.

Message – Através do arquivo message.properties, fica centralizado todas as mensagens que a api deverá exibir no sistema, facilitando a organização e manutenção