

Desenvolvimento de API REST para Endereços

Este projeto visa criar uma API eficiente para consulta de endereços utilizando CEP, priorizando escalabilidade e usabilidade.

**Viviana**

Presenter

Projeto de Teste

Apresentação do Projeto de Teste

Explorando os ajustes necessários para a consulta de CEP na aplicação.

Objetivo do Projeto

Uma solução prática e eficiente para consulta de endereços

#1

Desenvolvimento de uma API REST

O projeto se concentrou no desenvolvimento de uma API REST, que permite a consulta de endereços utilizando um CEP fornecido. A API é projetada para ser robusta e eficiente, facilitando a integração com outros sistemas.

#2

Uso do ViaCEP

A solução faz uso do serviço ViaCEP, que fornece dados reais de endereços brasileiros. Essa integração garante que as informações retornadas sejam precisas e atualizadas, proporcionando uma melhor experiência ao usuário.

#3

Foco em organização e escalabilidade

Um dos principais objetivos foi garantir que a API seja organizada e escalável. Isso significa que é possível expandir e entender o funcionamento da aplicação, mesmo para usuários iniciantes.

#4

Implementação de front-end simples

Além da API, um front-end simples foi desenvolvido para permitir a visualização dos dados. Isso fornece uma interface amigável para usuários que desejam consultar endereços sem a necessidade de integração técnica.

Tecnologias Utilizadas

Explorando as ferramentas essenciais para o desenvolvimento eficaz de software.

Java 17

A linguagem de programação principal utilizada no desenvolvimento do projeto, reconhecida por sua robustez e versatilidade, permitindo a criação de aplicações escaláveis e de alto desempenho.

Postman

Aplicativo utilizado para testar e documentar os endpoints da API, possibilitando a verificação de funcionalidades e a detecção de erros de forma ágil e eficaz.

Spring Boot

Framework que facilita a criação de APIs REST, proporcionando uma estrutura simplificada e eficiente para o desenvolvimento de aplicações web, reduzindo o tempo de implementação.

Maven

Ferramenta de gerenciamento de dependências e construção do projeto, permitindo uma gestão eficiente das bibliotecas necessárias e garantindo a consistência do ambiente de desenvolvimento.



Selenium, Rest Assured, Spring Test

Usados para testar aplicações, tanto no back-end quanto no front. Foram realizados métodos desde testes unitários e de integração, quando testes funcionais com essas ferramentas.

Cenários abordados

Explorando as principais funcionalidades do sistema de consulta de CEP

Consulta de CEP Válido

Esta funcionalidade permite que os usuários consultem um CEP válido através do endpoint GET /cep/{cep}. Ao realizar a consulta, o sistema retorna dados de endereço, incluindo logradouro, bairro, cidade e estado, proporcionando informações rápidas e precisas.



CEP Não Encontrado

Caso o CEP consultado não seja encontrado, o sistema responde com um status 404 e uma mensagem apropriada. Essa funcionalidade é essencial para a transparência no uso do sistema, informando os usuários sobre a inexistência do CEP consultado.



Tratamento de CEP Inválido

Quando um CEP inválido é inserido, o sistema retorna uma mensagem de erro personalizada com o status 400. Isso garante que os usuários sejam informados sobre a natureza do erro e possam corrigir suas entradas de forma eficiente.



Interface Web

A interface web oferece acesso fácil às funcionalidades de consulta de CEP. Os usuários podem acessar a plataforma diretamente pelo navegador, através do link <http://localhost:8080/>, facilitando a utilização do serviço de forma prática e acessível.



Estrutura do Projeto

Análise das camadas e suas interações



Camadas do Projeto

O projeto é dividido em duas camadas principais: Controller e Service. O Controller é responsável por gerenciar as requisições e respostas HTTP, atuando como intermediário entre o cliente e o servidor. Por outro lado, a camada Service contém a lógica de negócios, onde são processadas as regras e operações que definem o funcionamento da aplicação.



Service

A camada Service é onde reside a lógica de negócios do sistema. Nela, são implementadas as regras e processos que governam o comportamento da aplicação. Essa separação permite uma melhor organização do código e facilita a manutenção e a escalabilidade do sistema.



Controller

A camada Controller é fundamental para a comunicação entre a interface do usuário e o backend. Ele recebe as solicitações do cliente, processa essas solicitações e envia as respostas adequadas. Isso garante que as interações sejam gerenciadas de maneira eficiente.



Interface web

Uma interface web simples foi desenvolvida para demonstrar de uma maneira visual para que usuários não tenham que depender da análise técnica para utilizar a aplicação. Nela, é possível verificar todos os métodos de inserção, bem como todos os retornos para eles.

Testes Automatizados

Estratégias e Comandos para Testes Eficazes

#1

Consulta de CEP Válido

Este teste verifica se uma consulta de CEP válido retorna um status 200 e os dados corretos, assegurando que a API forneça as informações esperadas ao usuário.

#2

Consulta de CEP Inválido

Quando um CEP inválido é consultado, a API deve retornar um status 400 acompanhado de uma mensagem personalizada. Este teste valida a capacidade da API de lidar com entradas erradas.

#3

Consulta de CEP Não Encontrado

Neste cenário, a consulta de um CEP que não existe deve resultar em um status 404 e uma mensagem apropriada. Este teste garante que a API não apenas funcione, mas também informe o usuário adequadamente quando não encontrar dados.

#4

Interface Web

Além dos testes da API, a interface web também foi testada para garantir que a página seja carregada corretamente, e retornos esperados para os diversos casos de uso, proporcionando uma melhor experiência ao usuário.

#5

Scripts de Teste no Postman

Cada requisição no Postman possui testes validados na aba 'Tests', permitindo que os desenvolvedores verifiquem rapidamente o comportamento da API após alterações.

#6

Collection Exportada

A coleção de testes foi exportada e entregue junto ao projeto, localizada na pasta 'src/main/resources'. Isso facilita o acesso e a execução dos testes por outros desenvolvedores.

#7

Comandos para Execução dos Testes

Os testes podem ser executados usando comandos específicos no Maven. Isso inclui testes da API, testes do código de negócio e testes do front-end, garantindo que todas as partes do sistema sejam testadas adequadamente.

#8

Testes da API

Para executar os testes da API, o comando `mvn test -Dtest=com.nttdata.teste_santander.api.*` deve ser utilizado, permitindo uma verificação eficiente das funcionalidades da API.

#9

Testes do Código de Negócio

Os testes do código de negócio podem ser executados com o comando `mvn test -Dtest=com.nttdata.teste_santander.codigo.*`, assegurando que a lógica de negócio esteja funcionando como esperado.

#10

Testes do Front-end

Para garantir que a interface do usuário funcione corretamente, os testes do front-end são executados com o comando `mvn test -Dtest=com.nttdata.teste_santander.web.*`. Isso é crucial para manter uma boa experiência do usuário.

Melhorias Futuras

Estratégias para otimização e eficiência nos testes

Conclusão do Projeto

Reflexões Finais e Agradecimentos



Capacidade de criar uma solução eficaz

O projeto demonstrou a habilidade de desenvolver uma solução simples e funcional, integrando recursos externos de maneira eficiente para atender às necessidades dos usuários.



Organização do código

Foi dada atenção especial à estrutura e organização do código, o que facilita a manutenção e a escalabilidade do projeto, tornando-o mais acessível para desenvolvimentos futuros.



Experiência do usuário

O foco na experiência do usuário foi primordial, garantindo que todos os envolvidos tivessem uma interação fluida e intuitiva com a solução proposta.



Agradecimento

Agradeço a oportunidade de me desenvolver através desse desafio. E também, a presença de todos aqui hoje.


Apêndices

Recursos e etapas para executar o projeto




Link para Collection Postman:

Disponível no repositório do projeto e na pasta `src/main/resources`.



Link para arquivo UML:

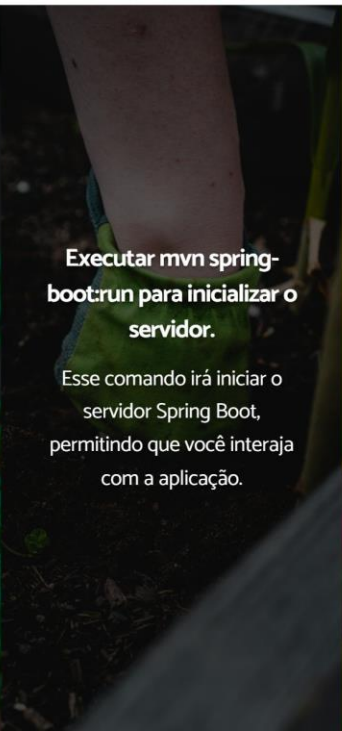
Disponível no repositório do projeto e na pasta `src/main/resources`.



Clonar o repositório e baixar dependências.

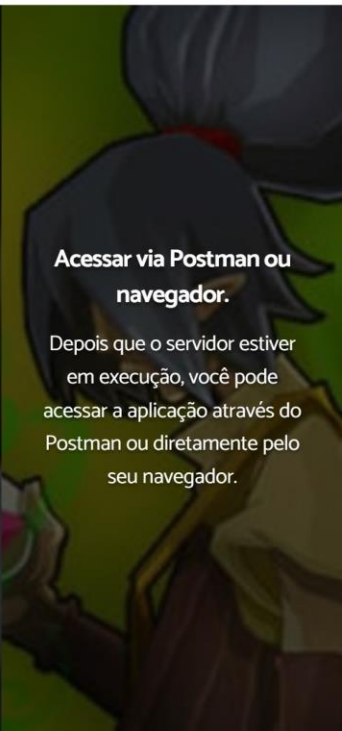
Use o comando `git clone` para copiar o repositório para sua máquina local.

Use o comando `mvn clean install -DskipTests` para baixar dependências para sua máquina local.



Executar `mvn spring-boot:run` para inicializar o servidor.

Esse comando irá iniciar o servidor Spring Boot, permitindo que você interaja com a aplicação.



Acessar via Postman ou navegador.

Depois que o servidor estiver em execução, você pode acessar a aplicação através do Postman ou diretamente pelo seu navegador.

