**FACULDADE DE INFORMÁTICA E ADMINISTRAÇÃO PAULISTA**

João Pedro Fontana Villagra | RM553343

Vinicius Malavia Lorenzetti | RM 553121

Marcelo Mendes Galli | RM 553654

**Projeto: DentalSin**

**SÃO PAULO**

**2024**

João Pedro Fontana Villagra | RM553343

Vinicius Malavia Lorenzetti | RM 553121

Marcelo Mendes Galli | RM 553654

Trabalho descritivo do desenvolvimento do Projeto Challenge em parceria com a empresa Odontoprev apresentado à Faculdade de Informática e Administração Paulista, como parte dos requisitos para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

SM: Karina Costa Paltrinieri Ronqui

SM: Antonio Sergio Rodrigues Figueiredo

**SÃO PAULO**

**2024**

# SUMÁRIO

[INTRODUÇÃO 4](#_Toc181603220)

[OBJETIVO 4](#_Toc181603221)

[PÚBLICO-ALVO 4](#_Toc181603222)

[DESCRIÇÃO DO SISTEMA 4](#_Toc181603223)

[INSTRUÇÕES PARA RODAR A APLICAÇÃO 5](#_Toc181603224)

[ENDPOINTS DA API 5](#_Toc181603225)

[PRINTS DAS REQUISIÇÕES 6](#_Toc181603226)

[DIAGRAMA DE CLASSES 8](#_Toc181603227)

[DIAGRAMA DE ENTIDADE E RELACIONAMENTO (DER) 9](#_Toc181603228)

Introdução

O Challenge desse ano letivo foi em parceria com a Odontoprev, uma empresa especializada em uma plataforma de cuidados dentários, que fornece diferentes tipos de planos com distinções de coberturas e serviços proferidos. Entretanto, qual seria o problema em questão a se solucionar? Os sinistros, os sinistros nada mais são que a materialização do risco em algum meio segurador, o real problema acontece quando os sinistros são considerados ruins, ou seja, um alto custo de consultas indevidas que resulta em prejuízo a empresa fornecedora do plano.

Objetivo

Desenvolver uma aplicação mobile, que por meio de uma análise preditiva de dados, conexão com banco de dados e APIs, seja capaz de relatar um possível diagnóstico de um paciente de acordo com os sintomas que o mesmo forneceu, indicando seu tipo de plano, dados pessoais, e possbilitando o contato com algum dentista. Por meio desse aplicativo, uma triagem seria feita, identificando o problema de maneira prematura sem que o usuário tenha que marcar uma consulta, logo, não tendo que usar algum serviço do plano. Permitindo que ele consiga cuidados preventivos e/ou marque logo uma consulta para não resultar em algo pior.

Público-Alvo

Quem seria esse possível usuário? Qualquer um que necessite de tratamento dentário. Claro que pode se estimar uma faixa etária de 15 a 60 anos. A explicação para isso é que o sistema é voltado para indivíduos que já possuem todos os dentes permanentes e estão mais suscetíveis a problemas dentários, além de que a triagem de casos atuaria tanto no começo de complicações odontológicas quanto com elas já em atuação. Em relação ao público pagante, essa aplicação está sendo desenvolvida com o intuito de que a própria Odontoprev adquira, com o objetivo de gerenciar os possíveis sinistros nos planos que fornecem.

Descrição do Sistema

O sistema consiste em uma API Spring Boot em Java, que permite a criação, atualização, deleção e busca de cadastros feitos por usuários na plataforma Mobile, sendo todos esses registros armazenados em um banco de dados relacional Oracle. Até o momento, temos duas entregas (Sprints) que a seguir serão separadas com data para exemplificar o período de desenvolvimento da aplicação:

**1º Sprint (02/09 a 07/10)**

* 27/09: Criação do projeto SpringBoot, diretórios, classes e repositórios.
* 27/09: Adição dos métodos Post, Get e GetById.
* 28/09: Criação do README.
* 07/10: Inserção dos endpoints e instruções de como rodar a aplicação.

**2º Sprint (08/10 a 08/11)**

* 17/10: Implementação do Lombok na classe Cadastro.
* 18/10: @Operation e @ApiResponse para auxílio nas requisições.
* 18/10: Criação do link para HATEOAS.
* 22/10: Atualização das validações.
* 23/10: Criação dos Enums para Sexo e TipoPlano.
* 28/10: Adição dos diagramas e renomeação

Instruções para Rodar a Aplicação

O usuário apenas deve rodar a classe JavaChallengeApplication, visto que o banco de dados Oracle ja está criado e com as tabelas configuradas, e o usuário e senha é automaticamente passado na application.properties, não necessitando que o programa seja rodado com parâmetros. Caso utilização do IntelliJ Ultimate, a API pode ser utilizada pela própria IDE com a extensão dos Endpoints. Na ausência dele, é recomendado Softwares como Postman ou Insomnia, por meio da requisição <http://localhost:8080/cadastros>.

[Vídeo explicativo.](https://www.youtube.com/watch?v=MIxtdDENVyA&ab_channel=Jo%C3%A3oPedro)

Endpoints da API

1. Criar Cadastro - POST /cadastros

* Descrição: Cria um novo cadastro no sistema.
* Corpo da Requisição:

A screen shot of a computer

Description automatically generated

2. Listar Cadastros - GET /cadastros

* Descrição: Retorna uma lista de cadastros.

3. Consultar Cadastro por ID - GET /cadastros/{id}

* Descrição: Retorna um cadastro específico.

4. Atualizar Cadastro - PUT /cadastros/{id}

* Descrição: Atualiza os dados de um cadastro.

5. Deletar Cadastro - DELETE /cadastros/{id}

* Descrição: Deleta um cadastro do sistema.

Prints das requisições

GET http://localhost:8080/cadastros

A screenshot of a computer program

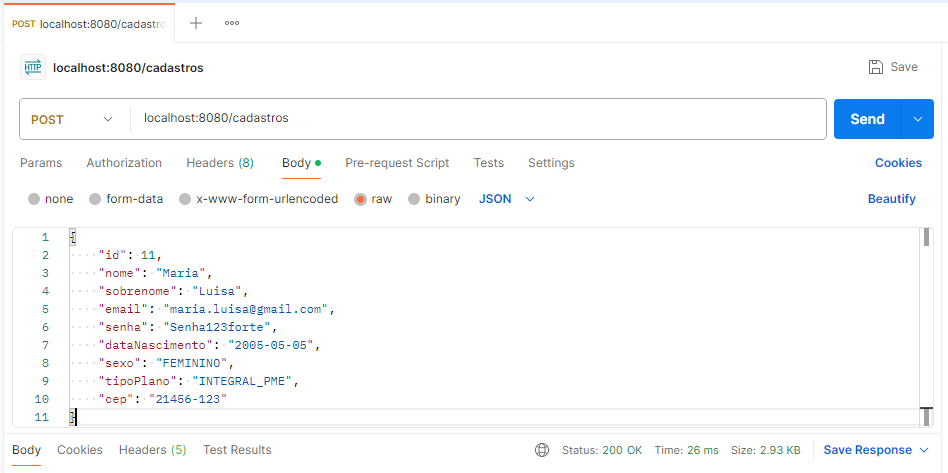
Description automatically generated

GET http://localhost:8080/cadastros/7

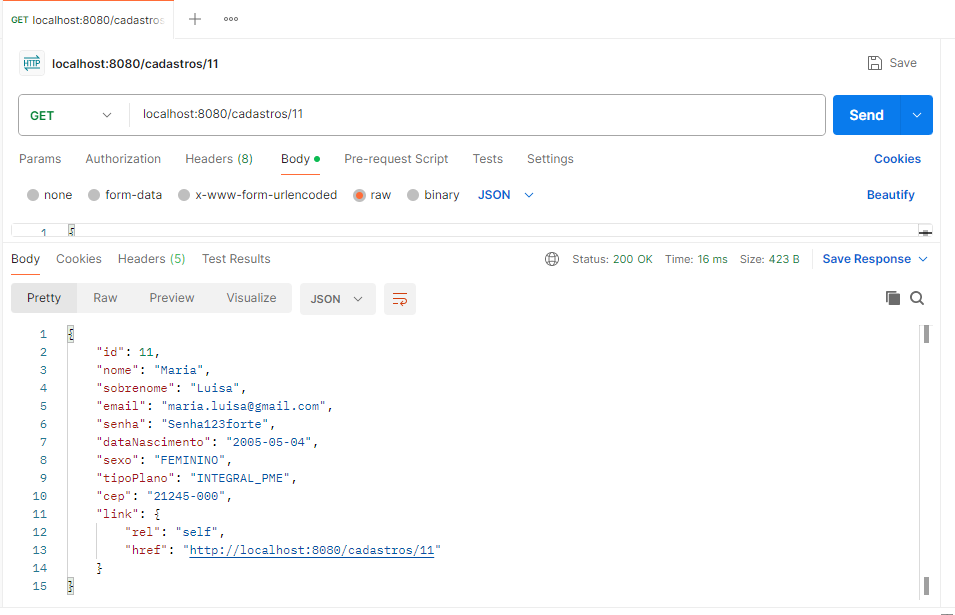
A screenshot of a computer

Description automatically generated

POST localhost:8080/cadastros



PUT localhost:8080/cadastros/11 (alteração no CEP do cadastro de ID 11)



DELETE localhost:8080/cadastros/11

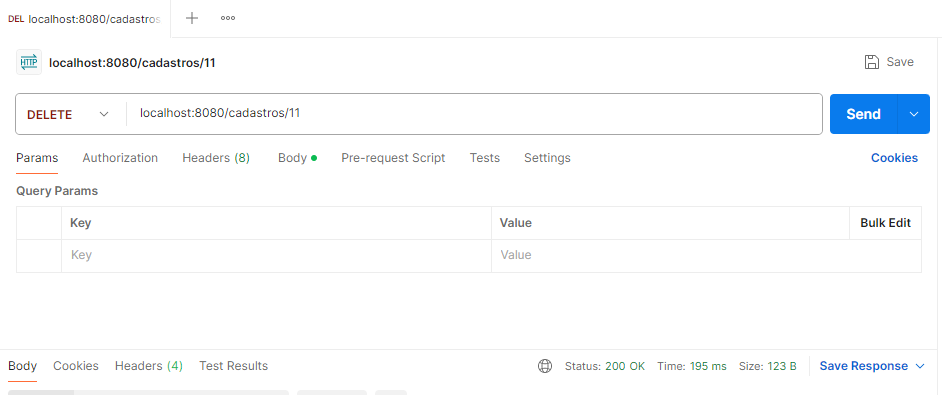


Diagrama de Classes

**A screenshot of a computer

Description automatically generated**

Diagrama de Entidade e Relacionamento (DER)

**A screenshot of a computer flow chart

Description automatically generated**