



Unilasalle-RJ
CENTRO UNIVERSITÁRIO LA SALLE DO RIO DE JANEIRO
Sistemas de Informação

Patrick Delfim Borges
Ronaldo da Silva Guimarães Júnior

Aplicação para gerenciamento de *Feature Toggle*

Niterói

2022



Patrick Delfim Borges

Ronaldo da Silva Guimarães Júnior

Aplicação para gerenciamento de *Feature Toggle*

Monografia apresentada ao Centro Universitário La Salle do Rio de Janeiro (UNILASALLE-RJ) como parte dos requisitos para aprovação na disciplina de Projeto Final II do curso de Bacharelado em Sistemas de Informação.

Orientador: Prof. Raphael Silva De Abreu

Niterói

2022

Catálogo elaborado pelo Processamento Técnico da Biblioteca do Unilasalle-RJ

Borges, Patrick Delfim

Aplicação para gerenciamento de Feature Toggle /
Patrick Delfim Borges, Ronaldo da Silva Guimarães Júnior.
Niterói: Unilasalle-RJ, 2022.

33p

Orientador: Prof. Ms. Raphael Silva De Abreu.

Trabalho de conclusão de curso (Bacharel em Sistemas de
Informação) – Unilasalle-RJ – Centro Universitário La Salle-RJ.

1. Computação. 2. Software - Desenvolvimento 3.
Software - Gerenciamento. I. Guimarães Júnior, Ronaldo da
Silva, (co-autor). II. Título.

CDD 004

Patrick Delfim Borges

Ronaldo da Silva Guimarães Júnior

Aplicação para gerenciamento de Feature Toggle

Anteprojeto apresentado ao Centro Universitário La Salle do Rio de Janeiro (UNILASALLE-RJ) como parte dos requisitos para aprovação na disciplina de Projeto Final II do curso de Bacharelado em Sistemas de Informação.

APROVADO EM: 08/12/2022

BANCA EXAMINADORA

Prof. Ms. Raphael Silva De Abreu (orientador)

Centro Universitário La Salle do Rio de Janeiro - UNILASALLE - RJ

Prof. Ms. Thiago Silva de Souza

Centro Universitário La Salle do Rio de Janeiro - UNILASALLE - RJ

Prof. Msc. Mario João Junior

Centro Universitário La Salle do Rio de Janeiro - UNILASALLE - RJ

Niterói

2022

RESUMO

Feature Toggle é uma técnica poderosa que permite às equipes envolvidas no desenvolvimento modificarem o comportamento do sistema sem alterar diretamente no código. Porém, feature toggles introduzem complexidade no código, podendo levar a cenários catastróficos. Esta complexidade pode ser diminuída se implementado de forma inteligente e com ferramentas apropriadas. Visto isso, o presente projeto tem por finalidade o desenvolvimento de um sistema para gerenciamento de feature toggle. O objetivo do projeto é construir uma solução desacoplada para simplificar e centralizar o gerenciamento das funcionalidades dos sistemas. O sistema é altamente configurável em relação às funcionalidades, permitindo ativar ou desativar em tempo real e definir valores com possíveis variações. Para funcionalidades mais complexas, também é possível definir regras para ativação baseadas no contexto da aplicação do cliente. Para atingir este objetivo, foi necessária a elaboração de um sistema que disponibiliza uma API REST para consultar as *feature toggles* e um painel administrativo para fazer o gerenciamento de forma rápida e simples.

PALAVRAS-CHAVE: Feature toggle. API REST. Desacoplamento.

ABSTRACT

Feature Toggle is a powerful technique that allows teams involved in development to modify system behavior without changing code directly. However, feature toggles introduce complexity into the code, which can lead to catastrophic scenarios. This complexity can be reduced if implemented in a smart way and with appropriate tools. Given this, the purpose of this project is the development of a feature toggle management system. The aim of the project is to build a decoupled solution to simplify and centralize the management of systems features. The system is highly configurable in terms of features, allowing you to activate or deactivate it in real time and define values with possible variations. For more complex features, it is also possible to define rules for activation based on the context of the client application. To achieve this goal, it was necessary to create a system that provides a REST API to query feature toggles and an administrative panel to manage them quickly and easily.

KEYWORDS: Feature toggle, REST API, decoupling.

LISTA DE ILUSTRAÇÕES

Figura 1 – Representação em diagrama de estrutura de dados.....	18
Figura 2 – Representação em diagrama de caso de uso.....	19
Figura 3 – Diagrama representando o fluxo geral da aplicação.....	22
Figura 4 – Exemplo do corpo de uma requisição enviando o contexto.....	23
Figura 5 – Representação em diagrama da arquitetura em camadas.....	25
Figura 6 – Fluxograma processo de cadastro de uma funcionalidade.....	27
Figura 7 – Exemplo de feature com o valor fixo.....	28
Figura 8 – Exemplo de feature com o valor contendo variações.....	29
Figura 9 – Fluxograma processo validação de funcionalidades.....	30
Figura 10 – Exemplo do fluxo de processamento das regras para uma requisição..	31
Figura 11 – Gráfico mostrando a distribuição do algoritmo de variação.....	32
Figura 12 – Tela inicial do front-end da aplicação.....	33
Figura 13 – Tela de cadastro do front-end da aplicação.....	34
Figura 14 – Tela de projetos do front-end da aplicação.....	35
Figura 15 – Tela de funcionalidades do front-end da aplicação.....	36
Figura 16 – Cadastro de nova funcionalidade no front-end da aplicação.....	37
Figura 17 – Tela de edição de funcionalidade no front-end da aplicação.....	38
Figura 18 – Tela de seleção de segmentos de regras no front-end da aplicação.....	39
Figura 19 – Tela de cadastro dos valores da funcionalidade quando vinculado ao segmento de regra.....	40
Figura 20 – Tela de cadastro de regras de segmento.....	41

LISTA DE ABREVIATURAS E SIGLAS

HTTPS	HyperText Transfer Protocol Secure
REST	Representational State Transfer
CD	Continuous Delivery
API	Application programming interface
SPA	Single page Application
SEO	Search engine optimization
ORM	Object Relational Mapping

SUMÁRIO

1 Introdução.....	11
2 Tecnologias Utilizadas.....	13
2.1 Front-End.....	13
2.1.1 React.....	13
2.1.2 Webpack.....	14
2.1.3 Typescript.....	14
2.2 Back-End.....	15
2.2.1 HyperText Transfer Protocol Secure.....	15
2.2.2 Node.js.....	15
2.2.3 Express.....	16
2.2.4 NestJS.....	16
3 Análise.....	16
3.1 Minimundo.....	17
3.2 Diagrama de Estrutura de Dados.....	18
3.3 Diagrama de Casos de Uso.....	19
3.3.1 Descrição dos Casos de Uso.....	19
3.3.1.1 Criar Projeto.....	19
3.3.1.2 Criar Funcionalidade.....	20
3.3.1.3 Criar Agregado de regras customizadas.....	20
3.3.1.4 Relacionar funcionalidades com agregado de regras.....	21
4 Projeto.....	22

4.1 Arquitetura.....	22
4.2 Implementação do sistema PR Toggles	24
4.2.1 Front-End.....	24
4.2.2 Back-End.....	25
4.3 Funcionamento do sistema	26
5 Interfaces.....	33
5.1 Login.....	33
5.2 Cadastro.....	34
5.3 Seus projetos.....	35
5.4 Funcionalidade.....	36
5.5 Criar nova funcionalidade.....	37
5.6 Editar funcionalidade selecionada.....	38
5.7 Selecionar grupo de segmentos.....	39
5.8 Configurar valores para grupo de segmentos.....	40
5.9 Criar um novo grupo de segmentos.....	41
6 Considerações Finais.....	42
7 Trabalhos Futuros.....	42
Referências Bibliográficas.....	44

1 Introdução

Sistemas de software complexos são repletos de funcionalidades (*features*) para gerenciar diversos recursos, estados e comportamentos do sistema. Segundo IEEE *et al.* (1998) *features* é o nome dado a funcionalidades implementadas em sistemas que refletem um conjunto de requisitos funcionais. Estas podem ser simples, como um botão que navega para outra página web ou complexas, como algoritmos de cálculo de rotas para um sistema logístico de entregas.

Com práticas modernas de desenvolvimento de software e a popularização do manifesto Ágil, foram desenvolvidos diversos conceitos para agilizar o processo de desenvolvimento de software. Um desses conceitos é o chamado entrega contínua (*Continuous Delivery* - CD), que prega uma abordagem de ciclos curtos de entrega incremental de software para implantação rápida em um ambiente de produção. Como benefício, o CD fornece um framework para equipes automaticamente fazerem o processo de build, teste e preparação para o lançamento em produção, tornando assim o processo eficiente, resiliente, rápido e seguro (RAHMAN *et al.*, 2016).

Essa metodologia de trabalho já se provou eficiente, sendo utilizada pelas maiores empresas de tecnologia do mundo, como a Netflix, Google e Facebook, segundo Rahman *et al.* (2016); entretanto, há desafios em implementá-la. Uma das principais prioridades de empresas de tecnologia tem sido reinventar o seu processo de lançamento de *features* para que ele se torne um processo seguro, porém, na prática, essa atividade se dá com grande dificuldade e de forma muito custosa. Embora existam tais dificuldades de implementação, essas companhias experimentaram e testaram diversas soluções para superar esses problemas. Uma dessas soluções é chamada de feature toggle (RAHMAN *et al.*, 2016).

Feature toggle é uma poderosa, porém simples, técnica que permite às equipes envolvidas no desenvolvimento modificarem o comportamento do sistema sem alterar diretamente no código. Na prática, o time deve configurar variáveis em condicionais com o poder de selecionar dinamicamente qual funcionalidade será executada em determinado momento para um usuário final. Essa técnica permite que equipes adicionem funcionalidades incompletas, revertam códigos defeituosos em produção, façam testes A/B e testagem de funcionalidades experimentais com uma parcela pequena dos usuários do sistema (MAHDAVI-HEZAVEH *et al.*, 2021).

Um exemplo prático para expressar o quão poderosa é esta solução seria a implementação de feature toggle para auxiliar na transição de um algoritmo de cálculo de frete da versão 1 para a versão 2 de forma segura. Para isso, uma feature toggle é criada direcionando 20% dos acessos a funcionalidade versão 1 e o restante para versão 2. Com isso, os desenvolvedores diminuem riscos ao alternar entre versões de funcionalidades cruciais para o funcionamento do sistema. Outro exemplo seria a realização de testes A/B para validar a aceitação das novas cores de um simples botão em um sistema front-end.

Contudo, quando feito de forma inapropriada, o processo de adicionar um feature toggle pode causar impactos negativos em seus sistemas, como o aumento de complexidade do código, código morto e falhas no sistema. Em 2012, por exemplo, a ativação de um feature toggle por engano levou a Knight Capital Group, uma multinacional americana de serviços financeiros, à falência perdendo cerca de 400 milhões de dólares em um período de 45 minutos por conta das consequências do mau funcionamento do sistema. O ocorrido se deu quando a empresa acidentalmente reutilizou um antigo feature toggle que havia sido configurado há 8 anos. Causando a execução de funcionalidades novas e antigas em diferentes servidores ao mesmo tempo (RAHMAN et al., 2016; HODGSON, 2017; MAHDAVI-HEZAVEH et al., 2021).

Portanto, o presente projeto visa desenvolver um produto de software, chamado de PR Toggles, para auxiliar outras empresas a simplificar o gerenciamento das funcionalidades do sistema que implementará a ferramenta. Essa solução é baseada na tecnologia de *feature toggles*, com funções que permitem desligar e ligar funcionalidades de seu sistema de forma dinâmica, na nuvem e em tempo de execução, sem a necessidade de recompilar todo o projeto de software.

Com isso, fornece aos desenvolvedores a possibilidade de configurar a realização de testes A/B, isolar funcionalidades a condições específicas e aumentar a segurança no *deploy* de novas funcionalidades. Com o intermédio do sistema, será permitido desativá-las em tempo real em caso um erro seja identificado, não sendo necessário fazer o deploy novamente. Sendo assim este produto garante uniformização e facilidade na implementação de funcionalidades.

O presente trabalho será apresentado da seguinte forma: no capítulo 2, será apresentado as tecnologias utilizadas. No capítulo 3, é apresentada a análise de requisitos com inclusão de minimundo e os respectivos diagramas do projeto. Em seguida, no capítulo 4, será explicada a arquitetura do sistema, assim como detalhes de seu funcionamento interno. Por fim, no capítulo 5, será apresentada a interface do painel administrativo do sistema, explicando também o seu funcionamento.

2 Tecnologias Utilizadas

As subseções deste capítulo irão descrever as linguagens, frameworks, bibliotecas e serviços utilizados no desenvolvimento do projeto. A escolha dessas tecnologias foi feita levando em consideração o quão elas se adequam ao modelo do sistema proposto e como funcionam bem juntas.

2.1 Front-End

No front-end da aplicação foi escolhido como stack principal o React para auxiliar na criação dos componentes visuais do sistema. Auxiliado pelo typescript, esse conjunto se torna uma poderosa ferramenta para a escalabilidade do projeto, apoiando na sua organização e facilidade de manutenção. Por ser inviável enviar diversos arquivos para o navegador do usuário, foi utilizado o webpack para transpilar os arquivos typescript em javascript e também juntar todos os arquivos em um só, facilitando assim a disponibilização do sistema front-end para o cliente.

2.1.1 React

React é uma biblioteca desenvolvida pelo Facebook e atualmente mantida de maneira open source com a proposta de facilitar o desenvolvimento de aplicações web front-end de forma declarativa, componentizada e reutilizável. A sua popularidade e comunidade forte tornam essa biblioteca uma excelente opção para fazer parte da lista de ferramentas a ser utilizada nesse projeto (FEDOSEJEV, 2015).

Graças ao ecossistema do React, foi possível utilizar diversas bibliotecas de apoio ao desenvolvimento, como por exemplo o react-query, que é altamente utilizado para facilitar requisições HTTP ao servidor. Essa biblioteca auxilia na requisição, cache, sincronização e atualização de dados do servidor na sua aplicação react. Também foi utilizado o React-hook-form, uma biblioteca muito

poderosa para auxiliar na gerência de formulários e validações dentro dos mesmos. Por fim, graças a filosofia de componentização do react, foi utilizada a biblioteca ChakraUI, que fornece para o usuário diversos componentes de interface altamente customizáveis e reutilizáveis.

2.1.2 Webpack

Para facilitar a experiência do desenvolvedor, projetos javascript de front-end hoje em dia são desenvolvidos de forma modular, ou seja, em diversos módulos diferentes separados por arquivo. Porém, torna-se inviável disponibilizar diversos arquivos javascript para o usuário fazer o download em seu navegador. Com base nessas ideias, foi utilizado o Webpack como bundler do nosso front-end (BOUZID, 2022).

O Webpack é um biblioteca de ferramentaria para unificar e minificar os arquivos javascript. Esse recurso facilita, então, a disponibilização de sistemas web com alta carga de javascript em um único arquivo, otimizando assim a utilização da banda larga do usuário quando o website é acessado e consequentemente minimizando requisições ao servidor (BOUZID, 2022).

2.1.3 Typescript

O typescript foi desenvolvido pela Microsoft com a proposta de ser uma extensão do javascript. Sua função é resolver diversos problemas que desenvolvedores encontram ao criar aplicações de larga escala com a linguagem javascript. Pelo fato da mesma ser uma linguagem antiga criada para ser utilizada de maneira simples, ela não contém funcionalidades básicas para a manutenabilidade de sistemas grandes (GOLDBERG, 2022).

A checagem de tipos é a funcionalidade introduzida pelo typescript em códigos javascript para garantir a segurança em seu desenvolvimento. Com ela é possível ter a garantia que contratos de tipagem e interfaces serão seguidos por

toda base de código, garantindo, assim, que problemas de sintaxe sejam descobertos antes da entrega do software e prevenindo comportamentos inesperados (GOLDBERG, 2022).

2.2 Back-End

Para o back-end da aplicação foi utilizado um servidor HTTP em Node.js com o framework NestJS. A implementação é feita em linguagem JavaScript junto ao superconjunto TypeScript.

2.2.1 HyperText Transfer Protocol Secure

O HTTPS – Protocolo de transferência de Hipertexto Seguro(Hypertext Transfer Protocol Secure) é um protocolo da camada de aplicação da internet. É dividido em duas partes, cliente e servidor, que se comunicam por meio de mensagens individuais, sem estabelecer uma conexão. O cliente envia uma solicitação e o servidor retorna uma resposta. Os dados transmitidos nessa comunicação são criptografados, sendo possível verificar a autenticidade do servidor e do cliente através de certificados digitais. O formato dessas solicitações e respostas são definidos pelo protocolo HTTPS. Pode ser utilizado para trocar vários tipos de recursos, como páginas web, imagens, vídeos, entre vários outros. O acesso a esses recursos é feito por meio de uma URL (KUROSE, James et al., 2006).

2.2.2 Node.js

Node.js é uma plataforma que permite escrever aplicações em linguagem JavaScript no lado do servidor. Se tornou altamente popular no desenvolvimento de aplicações web devido ao seu modelo non-blocking thread (não bloqueante), ou seja, ela não paralisa um processamento enquanto o servidor está realizando I/O.

Para isso ela utiliza uma abordagem assíncrona de I/O com um modelo orientado a eventos (PEREIRA, 2022).

2.2.3 Express

Express é um framework web baseado no módulo http do Node.js, que simplifica bastante ações que são repetitivas ao utilizar somente o módulo base. Ele é altamente configurável, seguindo a filosofia de configuração sobre convenção, o que significa que os desenvolvedores podem aplicar a arquitetura mais adequada para o projeto (MARDAN, 2014).

2.2.4 NestJS

NestJS é um framework JavaScript para criação de aplicações eficientes e escaláveis para servidores. Em sua implementação utiliza o framework Express, mas provém uma camada de abstração ao mesmo tempo que mantém isso disponível para ser manipulado, caso o desenvolvedor deseje.

O Nest utiliza da filosofia convenção sobre configuração, disponibilizando de imediato uma arquitetura de aplicação altamente escalável, testável, desacoplada e fácil de manter. (NESTJS, 2022)

3 Análise

O processo de análise e elicitação de requisitos do software começa pelo desenvolvimento do minimundo e subsequentemente a elaboração dos diagramas para melhor entendimento do produto a ser desenvolvido. O minimundo tem como premissa exemplificar o problema de adoção de feature toggle em um ambiente real do ponto de vista do cliente do sistema.

Para o melhor entendimento, é importante entender alguns conceitos-chave que serão frequentemente utilizados durante o desenvolvimento. Esses conceitos são: PR Toggles - Representa o sistema proposto; Cliente - Empresa que possui um ou vários sistemas e está implementando feature toggles utilizando o PR Toggles; Projeto - Representa um sistema do cliente; Usuário final - Pessoa comum que está acessando a aplicação do cliente.

3.1 Minimundo

O setor de tecnologia da informação de um cliente apresenta dificuldades de gerenciar os seus lançamentos de funcionalidades. Eles pretendem incorporar metodologias ágeis em seu fluxo de desenvolvimento e a utilização dos recursos de entrega contínua.

Após muito se estudar as opções, eles acabaram conhecendo o sistema PR Toggles, que consiste em uma solução robusta para auxiliar outras empresas de tecnologia a ultrapassar dificuldades no lançamento de novas funcionalidades de forma contínua e segura.

O sistema funciona da seguinte forma: o cliente cria uma conta e pode cadastrar diversos projetos. Em cada projeto é possível ter diversos toggles de configuração para exercer diversas funções. As possibilidades de configuração desses toggles são: regular (funcionalidade simples de liga e desliga), por contexto (cidade e idade do usuário final que está acessando a aplicação), lançamento gradual e teste A/B. Após o cliente configurar todos os toggles existentes no painel de administração, ele irá chamar uma API da PR Toggles em seu software para conectar o mesmo com as configurações definidas no painel administrativo. Essa requisição pode ser feita tanto do back-end quanto do front-end da aplicação do cliente. A partir daí, com apenas alguns cliques no painel será possível definir como e quando as novas funcionalidades do seu sistema serão disponibilizadas para os usuários finais.

Com isso o cliente tem a possibilidade de implementar funcionalidades no código de produção mesmo sem ela estar completa (entrega contínua), com a segurança de que isso não vai afetar toda a sua base de usuários finais. Além disso, a empresa descobriu diversas outras possibilidades para seu sistema, como, por exemplo: (i) o desligamento de funcionalidades que geram gargalos no sistema em momentos de pico com o intuito de preservar o servidor de ficar fora do ar (ii) a possibilidade de lançamento gradual de uma nova funcionalidade para verificar como ela se comporta no ambiente de produção e em caso de falha não afete a grande massa de todos os usuários da sua aplicação; (iii) o poder de trazer

melhores insights para o time de produto com testes A/B de funcionalidades e a sua coleta de métricas de aceitação, etc.

3.2 Diagrama de Estrutura de Dados

O banco de dados segue o modelo relacional, utilizando o sistema de gerenciamento de banco de dados MySQL. A Figura 1 apresenta o modelo relacional.

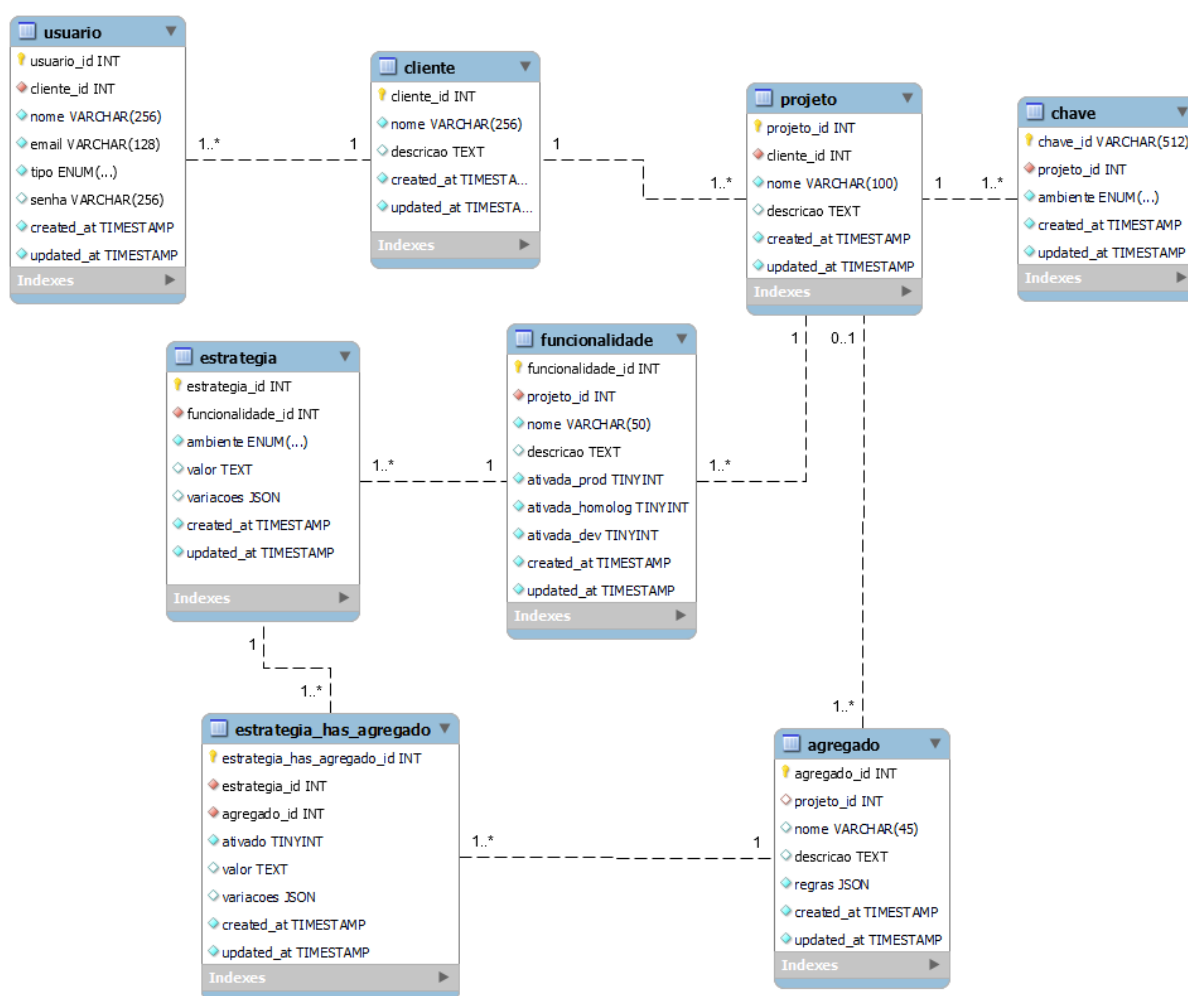


Figura 1: Representação em diagrama de estrutura de dados.

Fonte: Os autores, 2022.

De acordo com o modelo da Figura, um cliente pode ter vários projetos; um projeto pode ter várias funcionalidades e chaves (uma para cada ambiente); as funcionalidades possuem estratégias, que é onde é armazenado o valor e variações referente a cada ambiente; cada estratégia pode possuir um agregado, essa

entidade é responsável por guardar o segmento de regras; por questões de aproveitamento, um agregado também pode pertencer a um projeto, podendo assim ser utilizado em várias estratégias diferentes; a tabela resultante do relacionamento N para N entre estratégia e agregado serve para armazenar o valor e as variações de um agregado para uma estratégia.

3.3 Diagrama de Casos de Uso

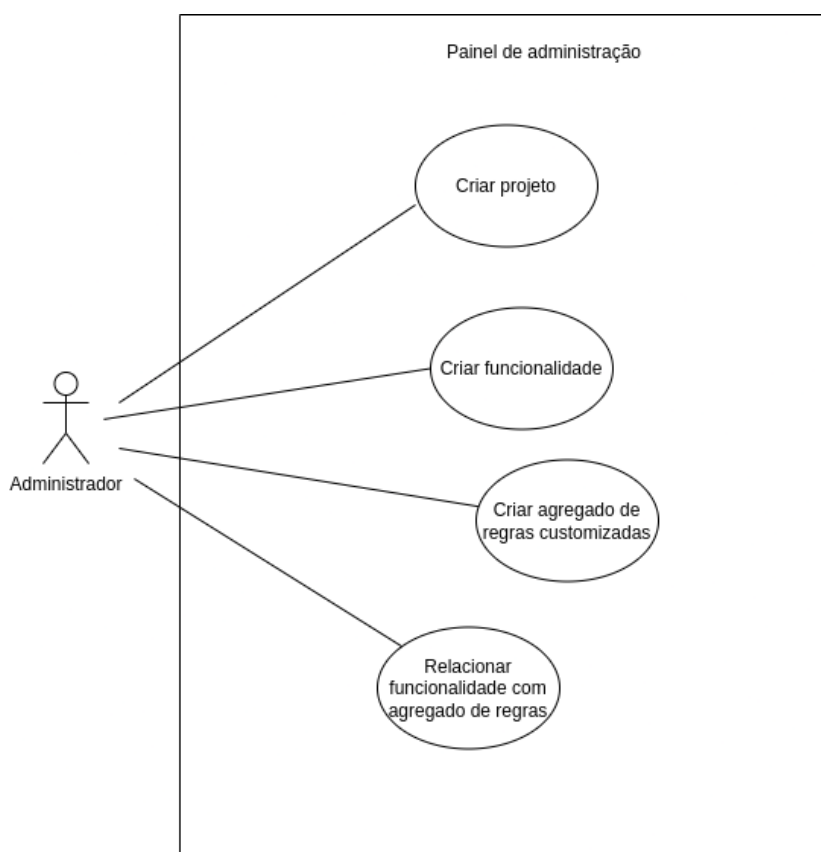


Figura 2 Representação em diagrama de caso de uso.
Fonte: Os autores, 2022.

3.3.1 Descrição dos Casos de Uso

3.3.1.1 Criar Projeto

Ator: Administrador

Descrição: No presente cenário, o administrador realiza o procedimento

de criar um novo projeto.

Pré-Condições:

- O administrador é cadastrado no sistema.
- O administrador já foi autenticado pelo sistema

Fluxo principal:

- 1) O administrador seleciona a opção “Criar projeto”.
- 4) Sistema aciona um formulário para cadastrar novo projeto.
- 5) O administrador preenche o formulário e seleciona a opção enviar.
- 6) Sistema valida os campos do formulário.
- 7) Sistema registra novo projeto.[A1]

Fluxo alternativo:

- A1
1. Sistema exibe mensagem de sucesso.
 2. Sistema exibe mensagem de erro de servidor.

3.3.1.2 Criar Funcionalidade

Ator: Administrador

Descrição: No presente cenário, o administrador realiza o procedimento de criar nova funcionalidade.

Pré-Condições:

- O administrador é cadastrado no sistema.
- O administrador já foi autenticado pelo sistema
- O administrador já selecionou o projeto onde quer realizar a operação

Fluxo principal:

- 1) O administrador seleciona a opção “Criar funcionalidade”.
- 4) Sistema aciona um formulário para cadastrar nova funcionalidade.
- 5) O administrador preenche o formulário e seleciona a opção enviar.
- 6) Sistema valida os campos do formulário.
- 7) Sistema registra nova funcionalidade.[A1]

Fluxo alternativo:

- A1
1. Sistema exibe mensagem de sucesso.
 2. Sistema exibe mensagem de erro de servidor.

3.3.1.3 Criar Agregado de regras customizadas

Ator: Administrador

Descrição: No presente cenário, o administrador realiza o procedimento de criar nova agregado de regras customizadas para expandir a capacidade da feature em contextos específicos.

Pré-Condições:

- O administrador é cadastrado no sistema.
- O administrador já foi autenticado pelo sistema
- O administrador já selecionou o projeto onde quer realizar a operação

Fluxo principal:

- 1) O administrador seleciona o ambiente do sistema onde ele quer trabalhar.
- 2) Administrador seleciona a funcionalidade.
- 3) O administrador seleciona a opção “criar novo segmento”.
- 4) Sistema aciona um formulário para cadastrar novo segmento de regras.
- 5) O administrador preenche o formulário com conjunto de chaves, valores e regras a serem aplicadas.
- 6) Sistema valida os campos do formulário.
- 7) Sistema registra novo segmento de regras.[A1]

Fluxo alternativo:

- A1
1. Sistema exibe mensagem de sucesso.
 2. Sistema exibe mensagem de erro de servidor.

3.3.1.4 Relacionar funcionalidades com agregado de regras

Ator: Administrador

Descrição: No presente cenário, o administrador vincula a funcionalidade específica com um conjunto de regras criado anteriormente.

Pré-Condições:

- O administrador é cadastrado no sistema.
- O administrador já foi autenticado pelo sistema
- O administrador já selecionou o projeto onde quer realizar a operação
- O administrador já selecionou a funcionalidade e o ambiente
- O administrador já criou um conjunto de regras

Fluxo principal:

- 1) Sistema mostra para o Administrador a lista de regras disponíveis para a utilização.
- 2) O administrador seleciona uma regra.
- 4) Sistema aciona um formulário.
- 3) Administrador cadastra novos valores a serem retornados pela funcionalidade caso o segmento de regras escolhido seja preenchido pelo contexto.
- 6) Sistema valida os campos do formulário.
- 7) Sistema registra dados.[A1]

Fluxo alternativo:

- A1
1. Sistema exibe mensagem de sucesso.
 2. Sistema exibe mensagem de erro de servidor.

4 Projeto

PR Toggles foi planejado tendo como público alvo administradores de projetos de software, gerentes de produto, desenvolvedores e analistas de testes. O objetivo é fornecer um painel de administração para o usuário criar projetos e configurar suas feature toggles e oferecer uma API pública para permitir que a ativação de uma feature seja mediada pelas configurações definidas no painel de administração. Mais especificamente, um desenvolvedor pode em sua aplicação consultar a API para decidir a ativação de um feature Toggle. Essa ativação por sua vez pode ser configurada com regras no painel de administração. Nas sessões subsequentes será demonstrado a arquitetura escolhida, detalhes da implementação e eventuais trade-offs.

4.1 Arquitetura

A Figura 3 representa o fluxo de comunicação entre o sistema do cliente e o PR Toggles, os retângulos maiores são os atores e os retângulos indicados pela cor branca representam os processos pelos quais eles são responsáveis.

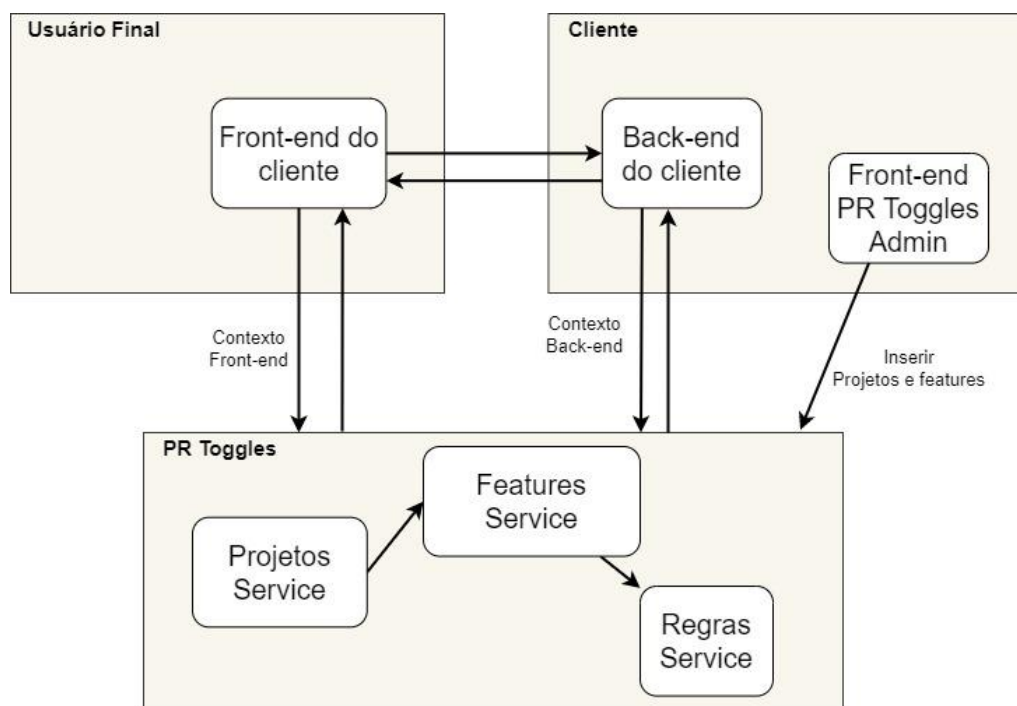


Figura 3: Diagrama representando o fluxo geral da aplicação.
Fonte: Os autores, 2022.

Inicialmente, o cliente é responsável por acessar o painel administrativo (Front-end PR Toggles Admin) e cadastrar os seus projetos e features. Após isso, ele deve desenvolver novas features em seu código de aplicação (por exemplo, um novo botão). Para decidir se uma feature será apresentada para o usuário final, o cliente deve colocar no seu código uma chamada API da PR Toggles e obter a relação de *features* do projeto em questão, e então criar ramificações no código baseadas nesse retorno. Essas chamadas podem ser feitas tanto do back-end quanto do front-end do seu sistema.

O usuário final é responsável por acessar a aplicação do cliente (Front-end do cliente), o que irá disparar as chamadas para a API da PR Toggles.

A PR Toggles é responsável por receber essas requisições, identificar de qual projeto está vindo (Projetos Service), listar as *features* ativas (*Features Service*) e calcular os seus valores (Regras Service). As funções de cada um dos *services* serão explicadas mais a fundo no capítulo 4.3.

Os valores das *features* de um projeto podem ser alterados dependendo do contexto, que é um conjunto de dados que representa o estado da aplicação que está rodando para o usuário final no momento da requisição. O estado de uma aplicação são as informações armazenadas no programa em um determinado tempo, por exemplo, o ID do usuário autenticado, qual navegador utilizado, dentre outros. A Figura 4 apresenta um exemplo de informações de contexto em json que devem ser enviadas para a API PR Toggles. Importante notar que a coleta destas informações de contexto depende do desenvolvedor. No exemplo na Figura abaixo, informações são coletadas no front-end do cliente, que está interagindo com o usuário final.

```
{
  "chave_ambiente": "cb9b2950-c3a6-42d8-9108-cc3209dbb835",
  "contexto": {
    "user_id": "99309201102",
    "estado": "rio de janeiro",
    "cidade": "niteroi",
    "sexo": "masculino",
    "idade": 23
  }
}
```

Figura 4: Exemplo do corpo de uma requisição enviando o contexto.
Fonte: Os autores, 2022.

Com estas informações de contexto o sistema PR Toggles pode executar um conjunto de regras pré-definidas pelo cliente que decidirão pela ativação da funcionalidade em questão.

4.2 Implementação do sistema PR Toggles

4.2.1 Front-End

Para o front-end da aplicação PR Toggles foi adotado a arquitetura Single page Application (SPA), que consiste na ideia do website ser inteiramente renderizado e atualizado de forma dinâmica no navegador do usuário em caso de interações e novas requisições ao servidor, ao contrário de recarregar a página inteira a cada interação. Os trade-offs de se utilizar uma arquitetura SPA é o fato de aplicações que adotam essa arquitetura tem o desempenho pior com a indexação de mecanismos de busca e podem demorar mais do que o eventual no primeiro carregamento, já que todo o código javascript do website precisa ser transferido de uma só vez para o navegador do usuário. Entretanto, foi analisado que para o caso de uso do painel de administração o desempenho necessário seria alcançado satisfatoriamente, e pela natureza extremamente dinâmica que o painel precisa ter foi decidido que essa escolha teria mais pontos positivos que negativos em sua implementação.

Outra influência de arquitetura empregada foi a arquitetura em camadas (n-tier architecture) onde o sistema é dividido nas seguintes camadas: Apresentação, Domínio e Serviços. Esta arquitetura consiste na ideia de dividir sua aplicação em três camadas, sendo elas apresentação, domínio e serviços. A Figura 5 apresenta esta arquitetura. Inicialmente, a camada de apresentação contém todos componentes visuais do sistema, assim como toda a lógica de funcionalidades que o usuário interage. Esta camada visual por sua vez faz solicitações e recebe retorno de dados da camada de domínio. A camada de domínio é responsável por separar logicamente os modelos e contratos de funções (interfaces). A camada de domínio lida com a definição de todas as funções de processamento, incluindo execuções de comandos, tratamento de erros, cálculos e quaisquer decisões lógicas para que sejam perpassadas para a camada de serviços. Por fim, a camada de serviços lida com o fornecimento de serviços para a camada de domínio. De forma prática, a camada de serviços se comunica com o mundo exterior por meio de requisições de

APIs REST e solicita alterações na página por meio das APIs do navegador. Exemplos destes serviços são qualquer comunicação com um servidor WEB back-end.



Figura 5: Representação em diagrama da arquitetura em camadas.
Fonte: Os autores, 2022.

4.2.2 Back-End

Foi utilizado NestJS como framework para desenvolvimento back-end. Esse framework é utilizado para construir aplicações em node.js integrado ao typescript. O mesmo é dividido em seções diferentes. Que são os modules, controllers e services. Essa divisão é pré estabelecida pelo framework para garantir a escalabilidade, testabilidade e desacoplamento. A camada de modulo encapsula toda lógica pertencente a algum domínio específico da aplicação, nela estarão presentes os services e os controllers específicos a determinado domínio. O controller é a camada que gerencia as requisições HTTP para a aplicação back-end, ele verifica qual requisição está chegando e encaminha as chamadas para a camada de service. Por fim, a camada de service é composta de diversos métodos que gerenciam toda a lógica do domínio de negócio, incluindo cálculos, chamadas a outros services como por exemplo service de acesso ao banco de dados, etc.

Essa arquitetura é baseada fortemente na ideia de injeção de dependências para fornecer forte desacoplamento da aplicação, essa ideia é um padrão de projeto que torna a classe independente de suas dependências. Para atingir esse resultado a dependência é atrelada a um contrato de interface que é utilizado tanto na

dependência quanto na classe a ser injetada. A dependência de uma classe é injetada na mesma por meio de seu construtor.

Para auxiliar na conexão com o banco de dados é utilizado um ORM (Object-Relational Mapping), o mesmo é uma técnica que auxilia na conexão entre banco de dados e programação orientada a objeto. Os ORMs são utilizados para mapear os dados do banco relacional em objetos de forma simplificada. Para este projeto foi utilizado o Prisma como ORM. Essa ferramenta é uma biblioteca ORM para node.js com typescript embutido, se tornando o candidato ideal para esse projeto por conta de sua sinergia com as tecnologias já empregadas no sistema.

4.3 Funcionamento do sistema

O sistema foi construído para possibilitar testes e facilitar o processo de disponibilização de novas features em uma aplicação, definindo valores para cada funcionalidade de um projeto, que podem conter variações em porcentagem, possibilitando possíveis testes A/B.

Para o melhor entendimento de todo o fluxo da aplicação e seus detalhes, a Figura 6 apresenta um fluxograma partindo da etapa inicial onde é feito o cadastro das *features* e segue pelas condições até a disponibilização destas features para o sistema do cliente.

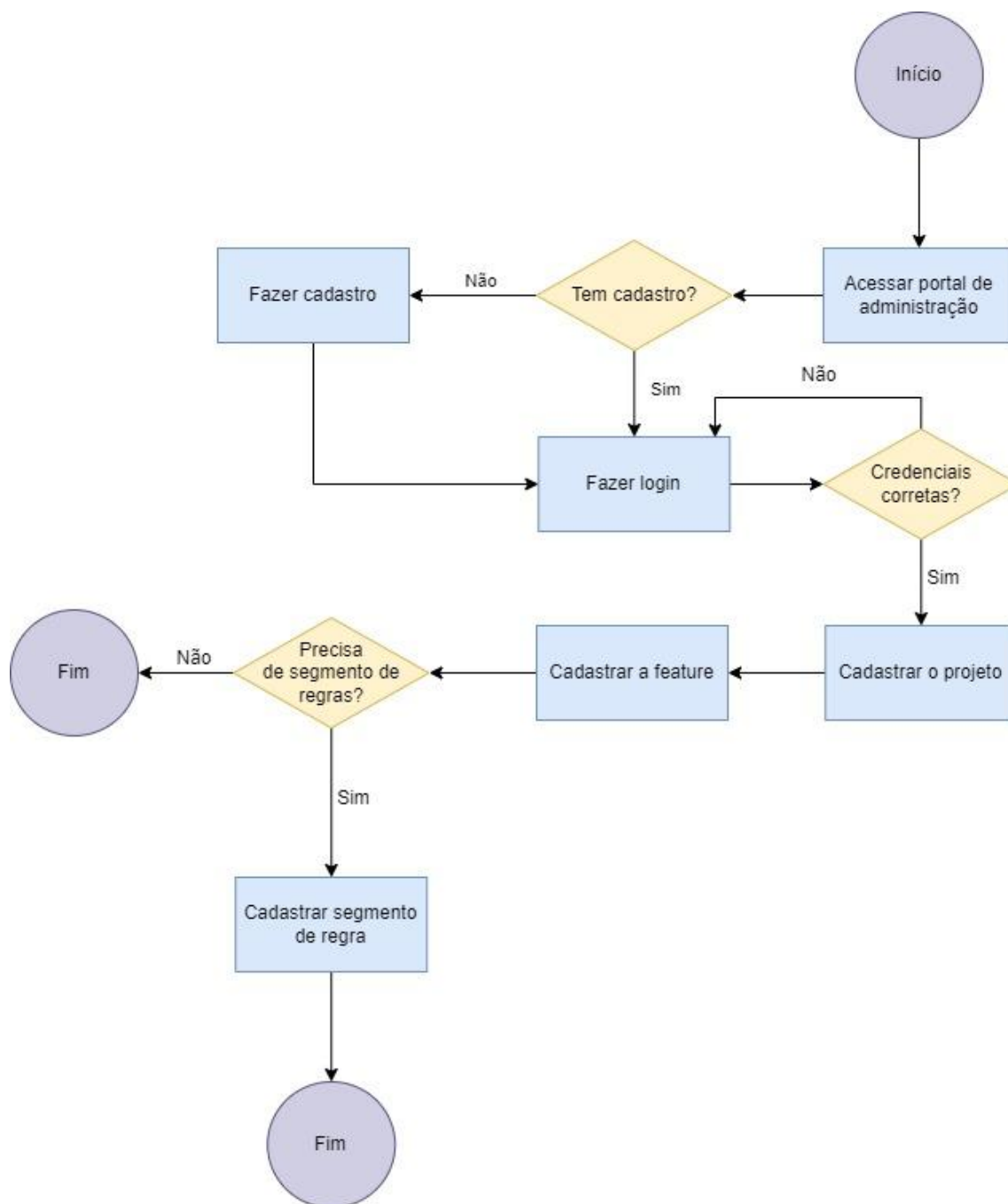


Figura 6: Fluxograma processo de cadastro de uma funcionalidade.
Fonte: Os autores, 2022.

Ao adentrar o painel administrativo, o cliente precisa se cadastrar ou autenticar a sua conta já existente. Caso ele opte por se autenticar, o sistema faz uma verificação de credenciais e permite o acesso ao painel ou o retorno à tela de login é disparado. Com o processo de autenticação concluído, o cliente pode acessar ou cadastrar um novo projeto para iniciar o gerenciamento de suas *features*.

Em um projeto é possível cadastrar e gerenciar suas *features*, onde deve ser informado seu respectivo valor e possíveis variações dele em porcentagem. Para casos de uso mais específicos, existe a possibilidade de cadastrar um segmento de regras para a *feature*, que também irá conter seu próprio valor e possíveis variações em porcentagem. Essas regras são utilizadas no processo de validação da *feature* pelo *Regras Service*, que irá comparar o contexto enviado frente a elas, e caso o mesmo se encaixe nas regras, será utilizado o valor e variações do segmento de regras.

A Figura 7 apresenta um exemplo de feature contendo um valor fixo. Em que um cliente define a criação de uma nova feature (bandeira vermelha) e o mesmo utiliza o sistema PR Toggles para ativar/desativar a feature para todos os usuários finais, distribuindo o mesmo valor para todos.

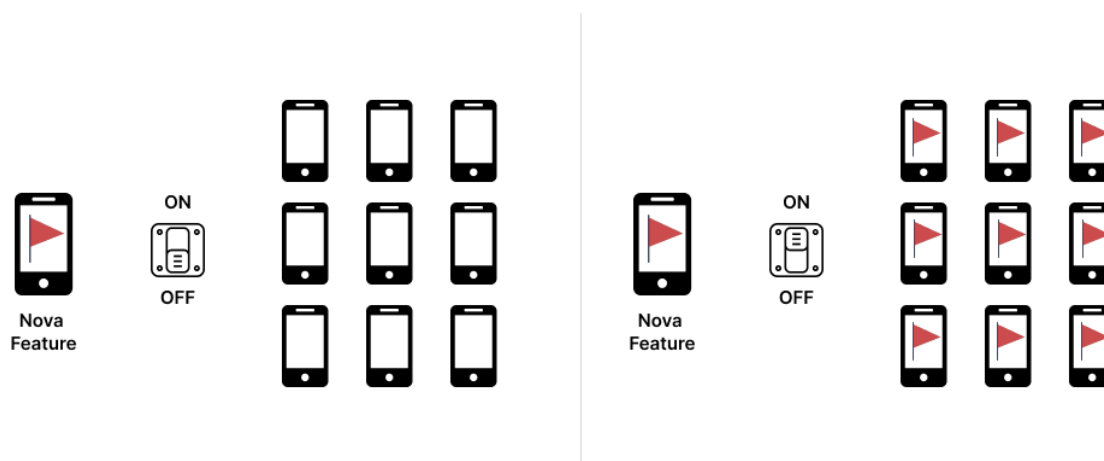


Figura 7: Exemplo de feature com o valor fixo.
Fonte: Os autores, 2022.

A Figura 8 apresenta uma capacidade mais avançada do sistema PR Toggles, a variação de valores de uma feature. Na figura um desenvolvedor (cliente) utiliza o sistema PR Toggles e define 3 valores para uma feature. Neste exemplo, é definido que 50% dos usuários finais deverão receber o valor 1 (bandeira vermelha), 30% dos usuários finais deverão receber o valor 2 (bandeira azul) e 20% dos usuários finais deverão receber o valor 3 (bandeira verde).

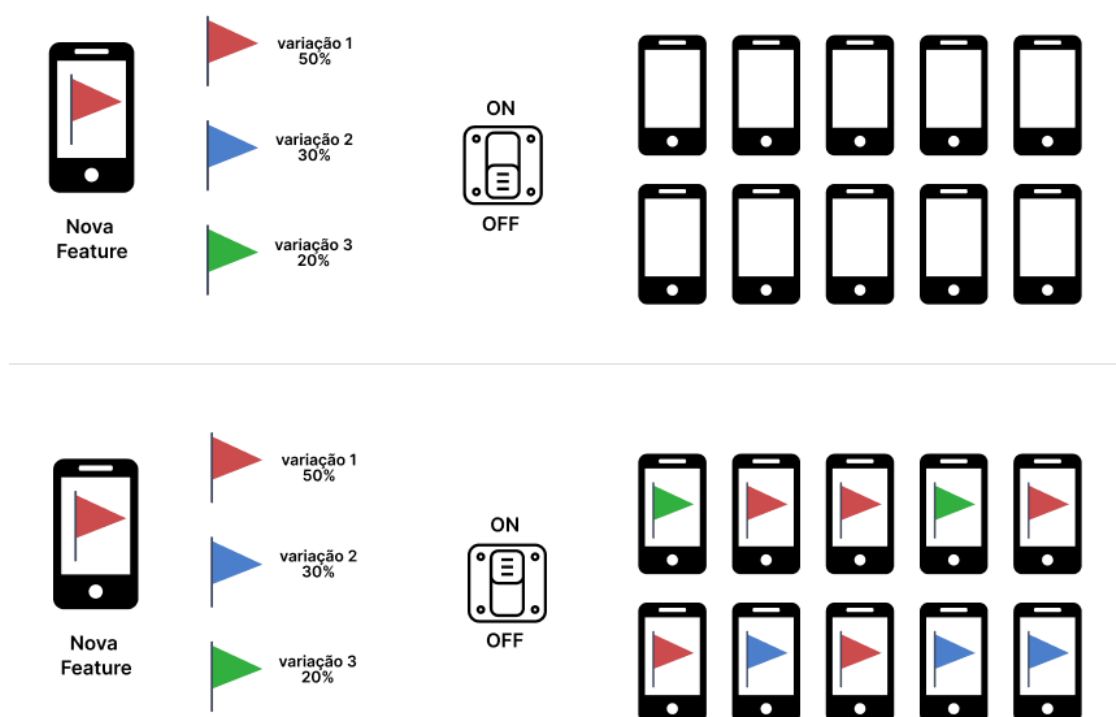


Figura 8: Exemplo de feature com o valor contendo variações.
 Fonte: Os autores, 2022.

Para entender a capacidade do sistema de validar as funcionalidades, a Figura 9 apresenta um fluxograma onde é exemplificado todo o processo de computação referente a chamada de recuperar funcionalidades de um projeto. Esse fluxograma demonstra o trabalho coordenado entre os projetos, features e regras services para identificar se uma feature é válida e qual o valor que será retornado.

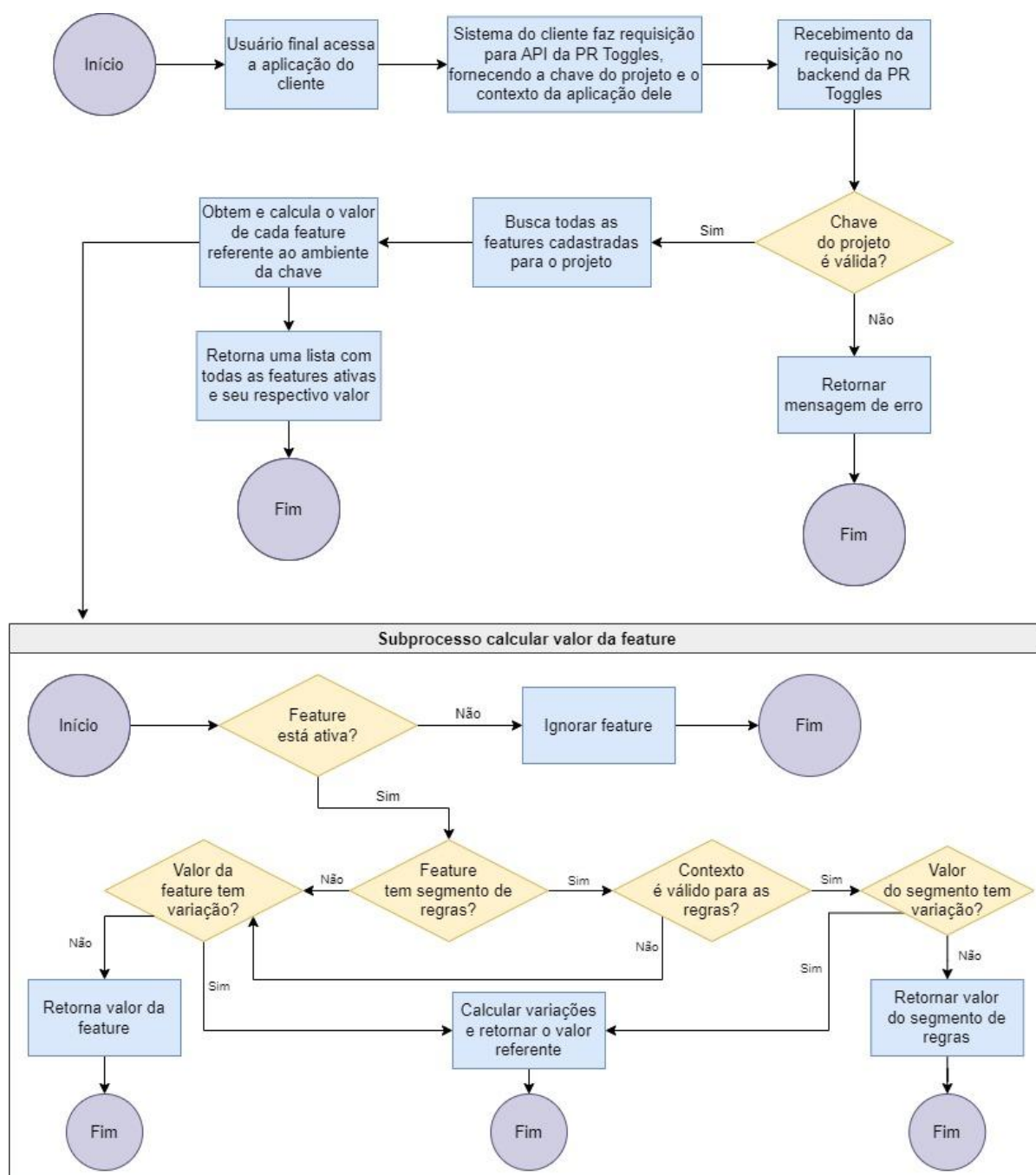


Figura 9: Fluxograma processo validação de funcionalidades.
Fonte: Os autores, 2022.

Tendo os projetos, suas funcionalidades e segmentos de regras cadastrados, o cliente deve modificar sua aplicação para consumir a API da PR Toggles para obter a relação de funcionalidades ativas e seus valores. Na chamada ao serviço deve ser informada a chave que identifica o projeto e o ambiente do qual está sendo feita a requisição, e caso necessário, os dados referentes ao contexto da aplicação do cliente.

O fluxo se inicia com o usuário final acessando a aplicação do cliente, o que irá disparar as requisições em busca das funcionalidades. Quando o sistema PR Toggles recebe a requisição, o primeiro passo é validar se a chave enviada existe no banco de dados, caso contrário irá retornar erro como resposta. Essa validação será feita pelo *Projeto Service*. Sendo a chave válida, identifica qual o projeto e ambiente ela representa e busca todas as funcionalidades referentes na base.

Em sequência, irá rodar um processo de validação para cada funcionalidade para montar a lista de funcionalidades e seus valores. Esse processo é feito pelo *Feature Service* e consiste em, primeiramente, verificar se a funcionalidade está ativa, se não, ignora. Seguindo, o *Regras Service* irá checar pela existência de um segmento de regras, caso exista, irá validar se o contexto enviado se encaixa nas regras, se sim, o valor retornado será o do segmento de regras, se não, retorna o valor normal da funcionalidade. A Figura 10 ilustra este fluxo.

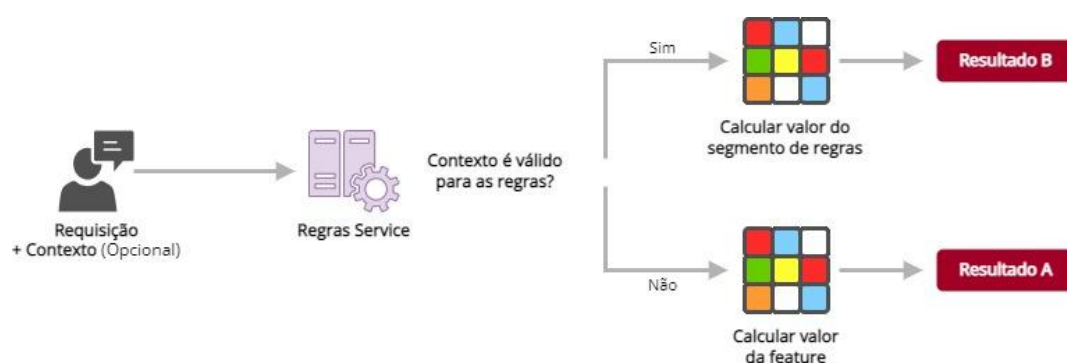


Figura 10: Exemplo do fluxo de processamento das regras para uma requisição.
Fonte: Os autores, 2022.

Caso o valor da funcionalidade ou o segmento de regras tenha variações, é feito o cálculo de um hash baseado em uma informação fixa do usuário final para definir em qual porcentagem ele será distribuído, garantindo com que este receba sempre o valor da mesma faixa de porcentagem sempre que acessar.

O hash é feito utilizando o algoritmo MD5, que tem como objetivo gerar uma string hexadecimal de tamanho fixo. A informação utilizada para fazer o hash é a junção do ID do usuário (que deve ser informado via contexto) com um *salt* que é único para cada funcionalidade cadastrada. O *salt* evita que um usuário caia sempre na mesma faixa de porcentagem para todas as funcionalidades. Transformando o

resultado desse hash para inteiro e dividindo pelo maior número da cadeia hexadecimal, gera um resultado entre 0 e 1. Este resultado é utilizado para definir de qual faixa de porcentagem da variação o usuário final irá receber o valor.

Esse método se provou rápido e confiável, produzindo distribuições planas e uniformes. O gráfico da figura 11 ilustra a frequência dessa distribuição.

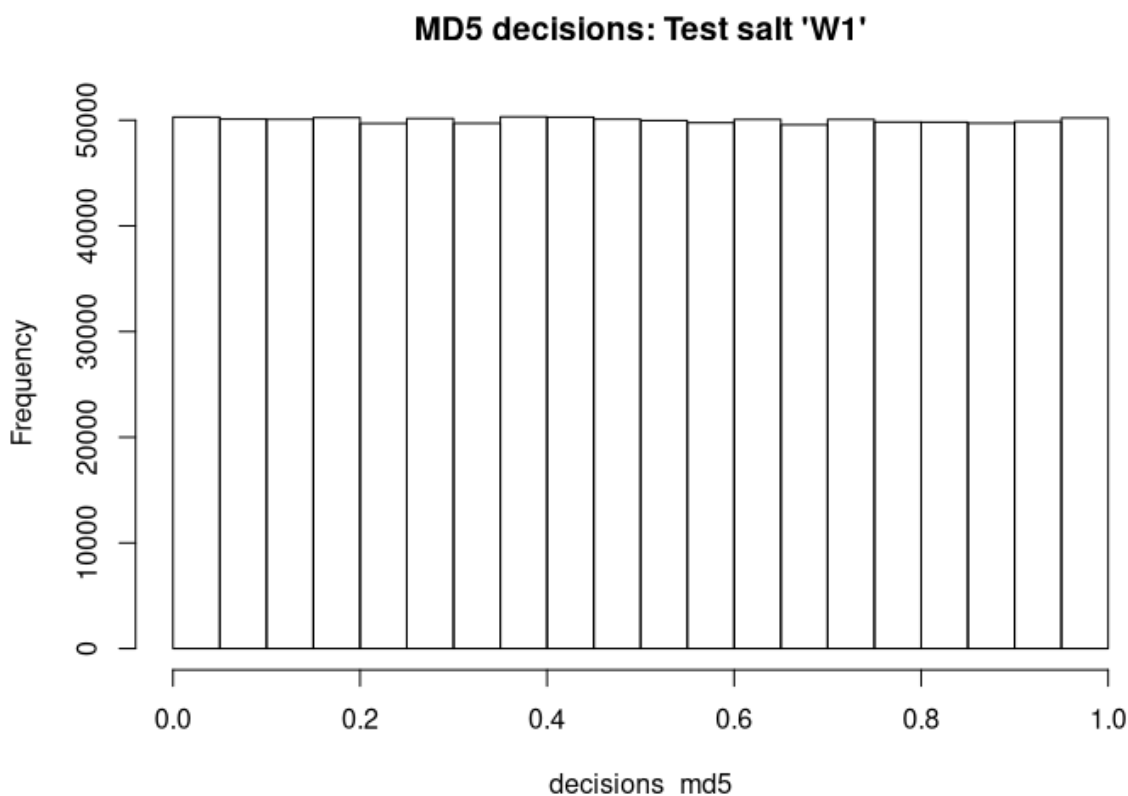


Figura 11: Gráfico mostrando a distribuição do algoritmo de variação.
 How to use hash functions for split test assignment. Documentação Mojito, 2022. Acesso em:
 novembro de 2022. Disponível em:
 <<https://mojito.mx/docs/example-hash-function-split-test-assignment>>.

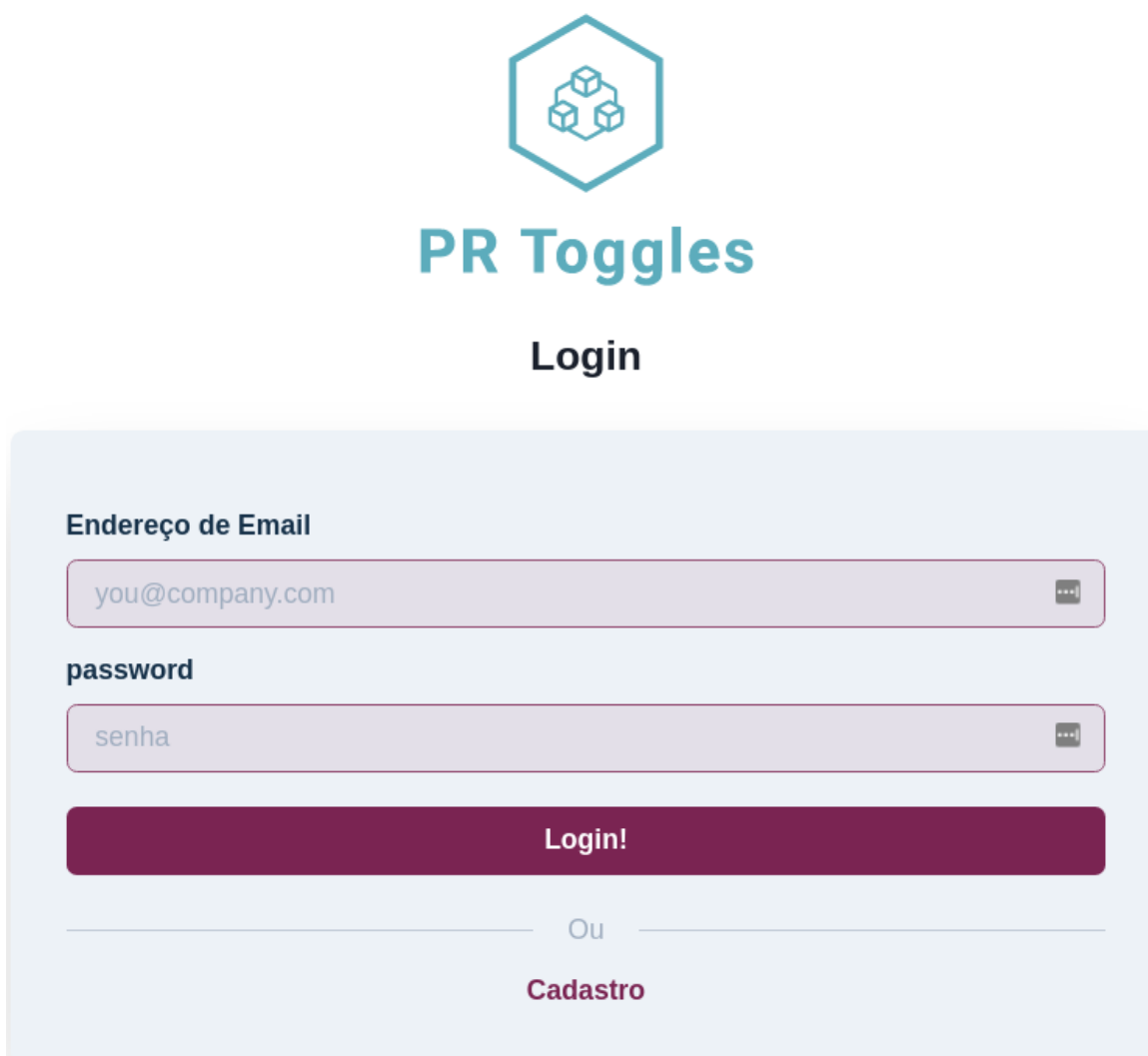
No final, retorna como resposta uma lista contendo todas as funcionalidades ativas e seus valores para o projeto e ambiente no qual a chave informada pertence.

5 Interfaces

Nesta seção será exibido as telas do painel administrativo do sistema PR toggles, assim como a descrição de cada tela.

5.1 Login

A tela inicial do nosso sistema é a tela de login apresentada pela Figura 12. Nessa tela o usuário deve incluir o seu email e senha cadastrada para conseguir ser autenticado ou o mesmo tem a possibilidade de criar uma nova conta. Caso o email ou a senha do usuário esteja incorreta, um pop up vai alertar que ocorreu algum problema na autenticação.



The image shows a login interface for 'PR Toggles'. At the top, there is a logo consisting of a teal hexagon with three cubes inside. Below the logo, the text 'PR Toggles' is displayed in a large, bold, teal font, followed by 'Login' in a smaller, bold, black font. The login form itself is a light blue rounded rectangle containing two input fields: 'Endereço de Email' with the placeholder 'you@company.com' and 'password' with the placeholder 'senha'. Both fields have a small 'x' icon on the right. Below these fields is a large, dark red button labeled 'Login!'. At the bottom, there is a horizontal line with the word 'Ou' in the center, and below that, the word 'Cadastro' in a dark red font.

Figura 12: Tela inicial do front-end da aplicação.
Fonte: Os autores, 2022.

5.2 Cadastro

Ao clicar em cadastro o usuário será direcionado para a tela apresentada na Figura 13, onde será necessário ele inserir as informações correspondentes. Ao clicar em cadastrar o mesmo será credenciado no sistema e o acesso vai ser permitido no painel administrador com sucesso. Caso o email do usuário já esteja cadastrado, uma mensagem de erro será exibida.



PR Toggles

Experimente nossa versão gratuita...
Comece a configurar o seu projeto agora mesmo!

Nome da empresa

Nome usuario

Endereço de Email

senha

confirmação de senha

Cadastrar!

— Ou —

Login

Figura 13: Tela de cadastro do front-end da aplicação.

Fonte: Os autores, 2022.

5.3 Seus projetos

A tela inicial do painel de administração apresentada na Figura 14 disponibilizará a lista de todos os seus projetos cadastrados. Nessa tela o usuário tem a possibilidade de cadastrar um novo projeto no sistema ou escolher um projeto existente para ajustar suas configurações.



Figura 14: Tela de projetos do front-end da aplicação.
Fonte: Os autores, 2022.

5.4 Funcionalidades

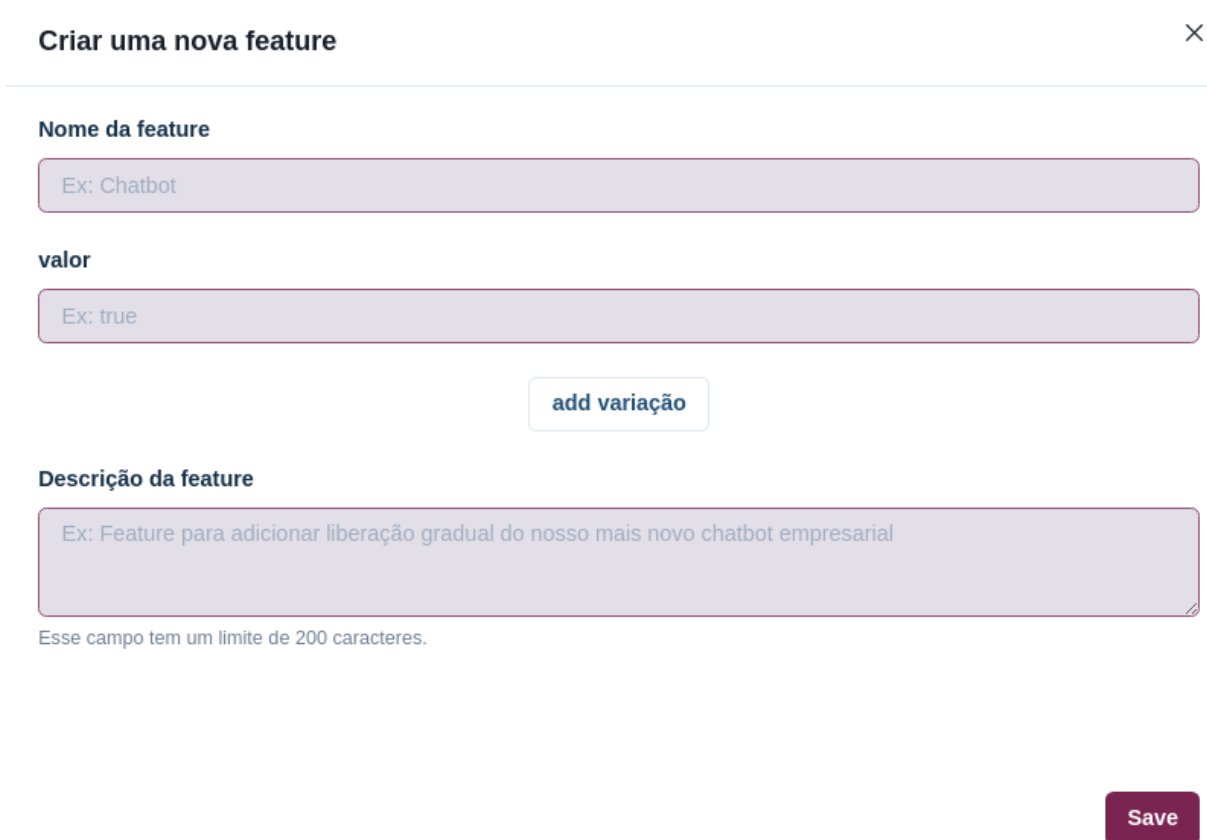
Após o usuário selecionar seu projeto ele é direcionado para a tela apresentada na Figura 15 onde é listada as funcionalidades vinculadas a aquele projeto. Nessa tela é disponibilizado as abas que é possível selecionar em qual ambiente do sistema o cliente tem a possibilidade de trabalhar e indicadores de ligado e desligado para a respectiva funcionalidade em seu respectivo ambiente. Ao clicar em Criar funcionalidade o sistema vai navegar para a tela onde é possível criar uma nova funcionalidade. Ao clicar no card de funcionalidade, será apresentado os detalhes para customizar as configurações de funcionalidades. Nessa tela também é possível desligar e ligar a funcionalidade em seu ambiente com apenas um click no botão de switch.



Figura 15: Tela de funcionalidades do front-end da aplicação.
Fonte: Os autores, 2022.

5.5 Criar nova funcionalidade

Ao clicar em criar nova funcionalidade uma aba lateral é aberta com o seguinte formulário. Neste formulário é preciso indicar o nome da feature que será o identificador na aplicação do usuário. Também é possível incluir um valor a ser retornado e se desejável a variação deste valor que será parametrizado com base em porcentagem. Por fim, opcionalmente é permitido cadastrar uma descrição. Essa tela é exemplificada na figura Figura 16.



O formulário, intitulado "Criar uma nova feature", contém os seguintes campos e elementos:

- Nome da feature:** Um campo de texto com o exemplo "Ex: Chatbot".
- valor:** Um campo de texto com o exemplo "Ex: true".
- add variação:** Um botão para adicionar uma variação ao valor.
- Descrição da feature:** Um campo de texto com o exemplo "Ex: Feature para adicionar liberação gradual do nosso mais novo chatbot empresarial".
- Save:** Um botão para salvar a nova feature.

Uma nota no rodapé do formulário indica: "Esse campo tem um limite de 200 caracteres."

Figura 16: Cadastro de nova funcionalidade no front-end da aplicação.
Fonte: Os autores, 2022.

5.6 Editar funcionalidade selecionada

Ao clicar no card de uma funcionalidade a tela explicitada pela Figura 17 será exibida com os dados pré preenchidos. Nessa tela é apresentado a possibilidade de realizar toda a customização da nossa funcionalidade no ambiente selecionado. Na parte superior da tela é possível acessar a aba segmentos para cadastrar variações dos valores dessa funcionalidade com base no contexto do usuário.

Atualizar feature: Layout especial de natal

Funcionalidade

Segmentos

valor - 20%

100

Variações são configuradas com base em porcentagem.

variação	peso (%)	
120	60	<div></div>
130	20	<div></div>

add variação

Descrição da feature

Ex: Feature para adicionar liberação gradual do nosso mais novo chatbot empresarial

Esse campo tem um limite de 200 caracteres.

Save

Figura 17: Tela de edição de funcionalidade no front-end da aplicação.
Fonte: Os autores, 2022.

5.7 Selecionar grupo de segmentos

Na aba segmentos é possível selecionar um grupo de segmentos de regras que já foi criado anteriormente. Esse grupo é criado a nível de projeto e pode ser aproveitado em diversas funcionalidades. A aba de seleção de grupo de segmentos é apresentada na Figura 18.



Figura 18: Tela de seleção de segmentos de regras no front-end da aplicação.
Fonte: Os autores, 2022.

5.8 Configurar valores para grupo de segmentos

Após o administrador do sistema selecionar um grupo de segmentos ele tem a possibilidade de incluir valores específicos caso o contexto onde esse grupo se encontra tente recuperar os valores das funcionalidades. A tela de adicionar os valores para grupos de segmentos é exemplificada pela Figura 19.

Atualizar feature: Layout especial de natal

Funcionalidade Segmentos

Gerencie valores de segmentos:

grupo_A

Criar novo segmento

grupo_A

Ativado

valor (opcional) - 50%

200

Variações são configuradas com base em porcentagem.

variação

250

peso

50

add variação

Save

Figura 19: Tela de cadastro dos valores da funcionalidade quando vinculado ao segmento de regra.
Fonte: Os autores, 2022.

5.9 Criar um novo grupo de segmentos

Ao clicar em criar novo segmento o usuário será redirecionado para a tela abaixo onde ele pode criar um novo grupo de regras para o seu sistema. No exemplo explicitado pela figura Figura 20 é criado condições onde o usuário ao acessar o sistema do cliente será da região nordeste com idade maior que 21 ou região norte. Esses dados são enviados em formato de contexto do sistema do cliente para a aplicação PR Toggles.

Atualizar feature: snow

×

Funcionalidade

Segmentos

nome do segmento

grupo_1_norte_nordeste

Condições de segmentação:

regiao

Igual a (=)

nordeste

🗑

And

idade

Maior que (>)

21

🗑

And Gate

Or

regiao

Igual a (=)

norte

🗑

And Gate

Or Gate

Descricao do segmento (opcional)

moradores do nordeste com idade maior que 21 ou moradores do norte

Cancel

Save

Figura 20: Tela de cadastro de regras de segmento.
Fonte: Os autores, 2022.

6 Considerações Finais

O desenvolvimento desta aplicação foi importante para exemplificar que uma solução de gerenciamento de *feature toggles* auxilia na implementação de ideias de entrega contínua contornando os desafios que é criar sua própria solução para suas para este problema.

A ideia de criar um software best-of-breed¹ garante que a utilização de *feature toggle* se torne padrão na indústria com uma solução robusta e estável, prevenindo assim, implementações displicentes e inconsistentes com o padrão do mercado.

7 Trabalhos Futuros

Como trabalhos futuros será necessário transformar o produto em um SaaS onde será implementado todo o sistema de cobrança de mensalidade e suas particularidades, assim como, incluir a diferenciação do plano gratuito para o plano pago do sistema. Com o intuito de melhorar a usabilidade do usuário, o sistema contará com a seção de auditoria e logs, onde será possível os administradores dos projetos olharem as alterações que suas funcionalidades sofreram e realizarem relatórios internos.

Para garantir a segurança em sistemas críticos será implementada regras de permissionamento onde usuários diferentes de uma empresa terão acessos restritos ao painel de administrador, evitando assim que funcionalidades sejam ativas em produção por engano por um usuário não autorizado.

Será criado uma biblioteca para auxiliar o cliente a integrar o sistema PR Toggles com o dele, garantindo assim a busca e utilização das funcionalidades de forma simplificada, a biblioteca a ser desenvolvida tem com premissa buscar os dados do usuário, enviar contexto, fazer cache da resposta caso a comunicação entre sistemas falhe e fazer a revalidação automaticamente em intervalos de tempo para garantir que o usuário tenha os atributos das funcionalidades de forma mais atualizada possível.

A necessidade de incluir gráficos e dashboards no painel de administração para gerar métricas quanto a utilização das funcionalidades pelo usuário é de extrema importância para a completude do sistema, esses gráficos são significativos

¹ O melhor sistema em seu nicho ou categoria.

para conferir quais funcionalidades estão sendo mais utilizadas e validar testes A/B realizados no sistema.

Referências bibliográficas

Artemij Fedosejev. React.js Essentials, 2015. Acesso em: Novembro de 2022. Disponível em: <https://www.google.com.br/books/edition/React_js_Essentials/Rhl1CgAAQBAJ?hl=pt-BR&gbpv=0&kptab=overview>

IEEE Std. 829, "IEEE Std 829: Standard for Software Test Documentation", Institute of Electrical and Electronic Engineers, USA, 1998.

KUROSE, James. F. & ROSS, Keith W. Redes de Computadores e a internet: Uma abordagem top-down, 3ª Edição, Editora Pearson, São Paulo– SP, 2006.

Mahdavi-Hezaveh, R., Dremann, J., & Williams, L. (2021). Software development with feature toggles: practices used by practitioners. Empirical Software Engineering, 26(1), 1-33.

Mohamed Bouzid. Webpack for Beginners: Your Step-by-Step Guide to Learning Webpack 4, 2020. Acesso em: Novembro de 2022. Disponível em: <https://www.google.com.br/books/edition/Webpack_for_Beginners/aHHtDwAAQBAJ?hl=pt-BR&gbpv=1>

MARDAN, Azat. Express. js Guide: The Comprehensive Book on Express. js. Azat Mardan, 2014. Acesso em: Novembro de 2022. Disponível em: <https://books.google.com.br/books?hl=pt-BR&lr=&id=5eGRAwAAQBAJ&oi=fnd&pg=PP6&dq=express+framework+js&ots=nkrhw_crKJ&sig=KHtBTzxSOFv8FqgsoNHSo6xf_-w#v=onepage&q=express%20framework%20js&f=false>

NESTJS. Documentação NestJS. 2022. Acesso em: Novembro de 2022. Disponível em: <<https://docs.nestjs.com/>>.

RAHMAN, Md Tajmilur et al. Feature toggles: practitioner practices and a case study. In: Proceedings of the 13th international conference on mining software repositories. 2016. p. 201-211.

PEREIRA, Caio Ribeiro. Aplicações web real-time com Node. js. Editora Casa do Código, 2014. Acesso em: Novembro de 2022. Disponível em: <https://books.google.com.br/books?hl=pt-BR&lr=&id=Wm-CCwAAQBAJ&oi=fnd&pg=PT7&dq=nodejs&ots=_er3-lxsaM&sig=md9nbRDp6C0APm85GIE-U1nqkBQ#v=onepage&q&f=false>

Pete Hodgson. Feature Toggles. 2017. Acesso em: Junho de 2022. Disponível em: <<http://martinfowler.com/articles/feature-toggles.html>>