

Sérgio Cerqueira Santos e Victor Lopes dos Santos

Relatório final Olympicsfollow

Salvador, Bahia, Brasil

Agosto 2024

Sérgio Cerqueira Santos e Victor Lopes dos Santos

Relatório final Olympicsfollow

Relatório final do desenvolvimento do sistema
OlympicsFollow para a disciplina de Progra-
mação Web.

Instituto Federal da Bahia – IFBA

Curso de Análise e Desenvolvimento de Sistemas

Campus Salvador

Salvador, Bahia, Brasil

Agosto 2024

1 Arquitetura do OlympicsFollow

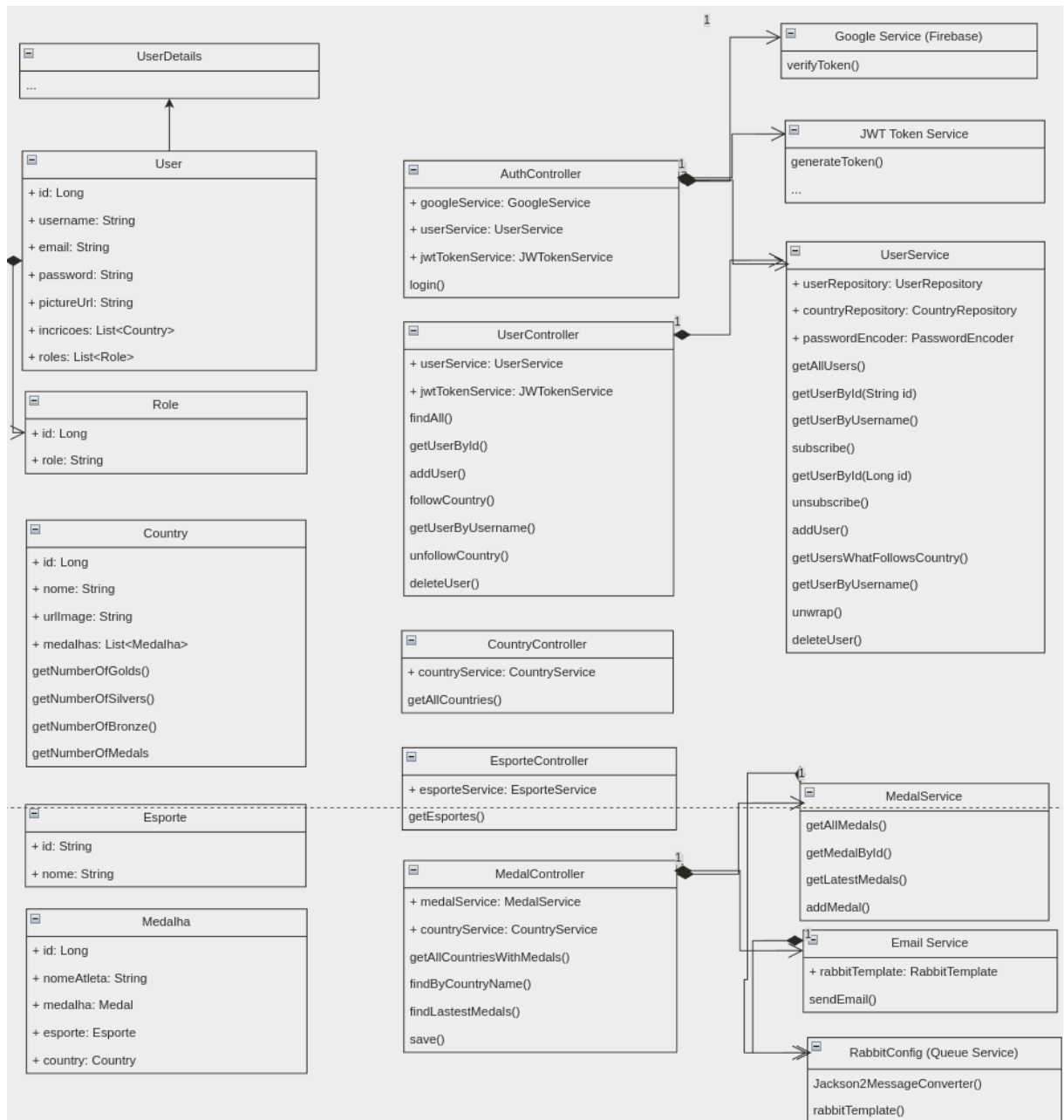
1.1 Estruturação do sistema

O sistema "OlympicsFollow" possui como objetivo a visualização do quadro de medalhas da olimpíadas, destacando os esportes em que cada país ganhou uma medalha e dando a flexibilização ao usuário de poder "seguir" alguns países, sendo notificado no email quando alguma nação que ele é seguidor ganha medalha. Ademais, com uma estruturação feita em microsserviços para garantir a escalabilidade da aplicação, o sistema foi desenvolvido com 2 tecnologias: Spring Boot no backend e React no frontend.

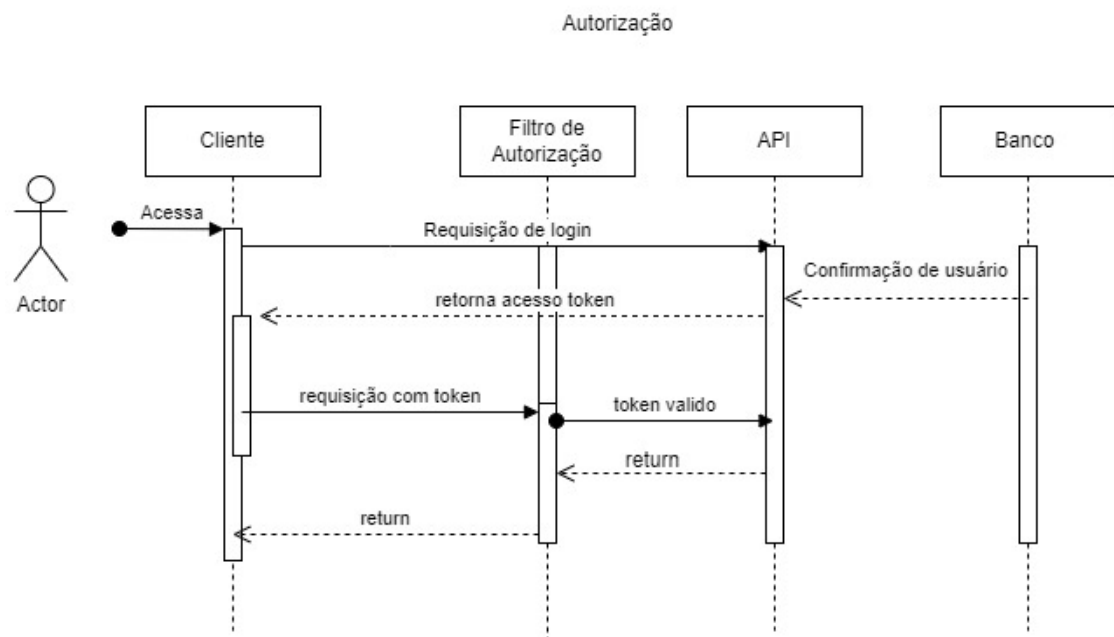
Microsserviços é uma abordagem arquitetural que define que uma aplicação deve ser constituída de pequenos serviços independentes, cujos se comunicam entre si utilizando APIs bem definidas. Os serviços, por serem independentes, além de separar as responsabilidades, evita a duplicação de código, tornando o sistema mais limpo e escalável. Nesse âmbito, foram definidos 4 microsserviços: Email, API de cadastro de medalhas, Eureka server e Gateway.

O frontend, lado do cliente, por sua vez, é onde se concentra toda a parte visual do OlympicsFollow, cujo é composto por 3 telas. Desenvolvido em ReactJS, uma biblioteca de desenvolvimento web Javascript, foi possível atingir uma boa estruturação, separando os "Componentes" de sua lógica através do uso de "Custom Hooks". Os Hooks Customizáveis, como também são conhecidos, é uma forma de desenvolvimento, abstração e estruturação do sistema que permite separar a lógica da parte visual, garantindo uma flexibilidade maior e mais escalável.

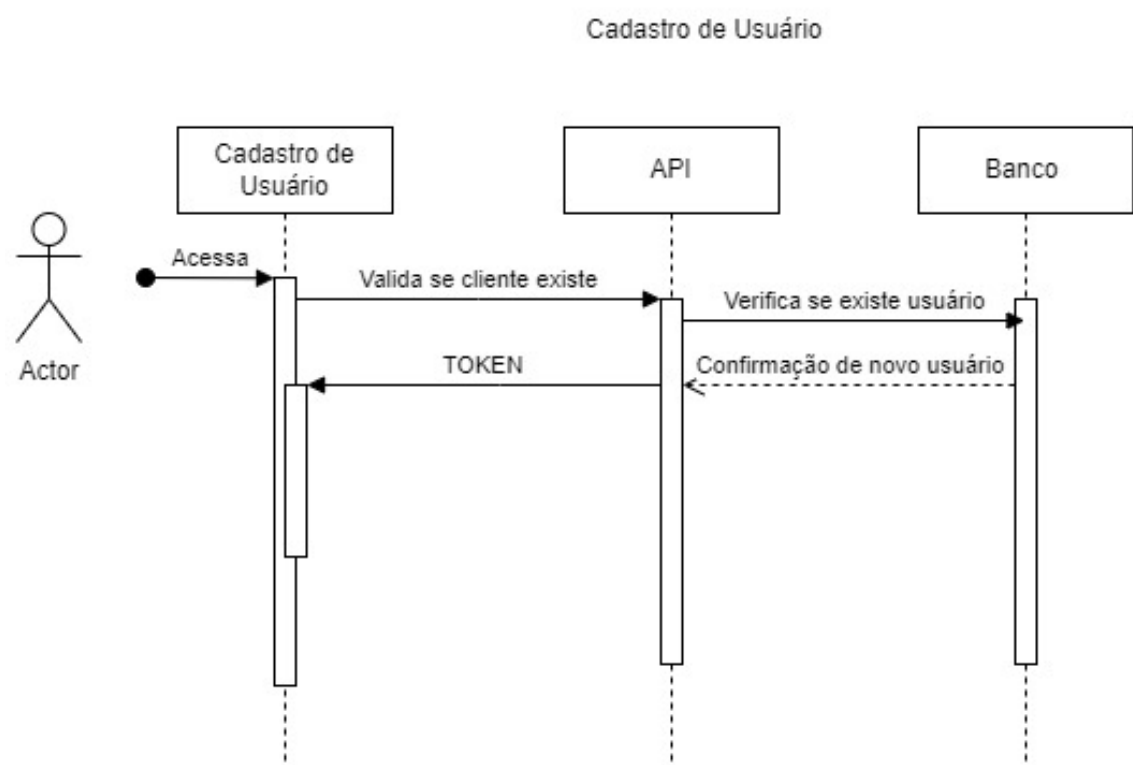
1.2 Diagrama de classes UML



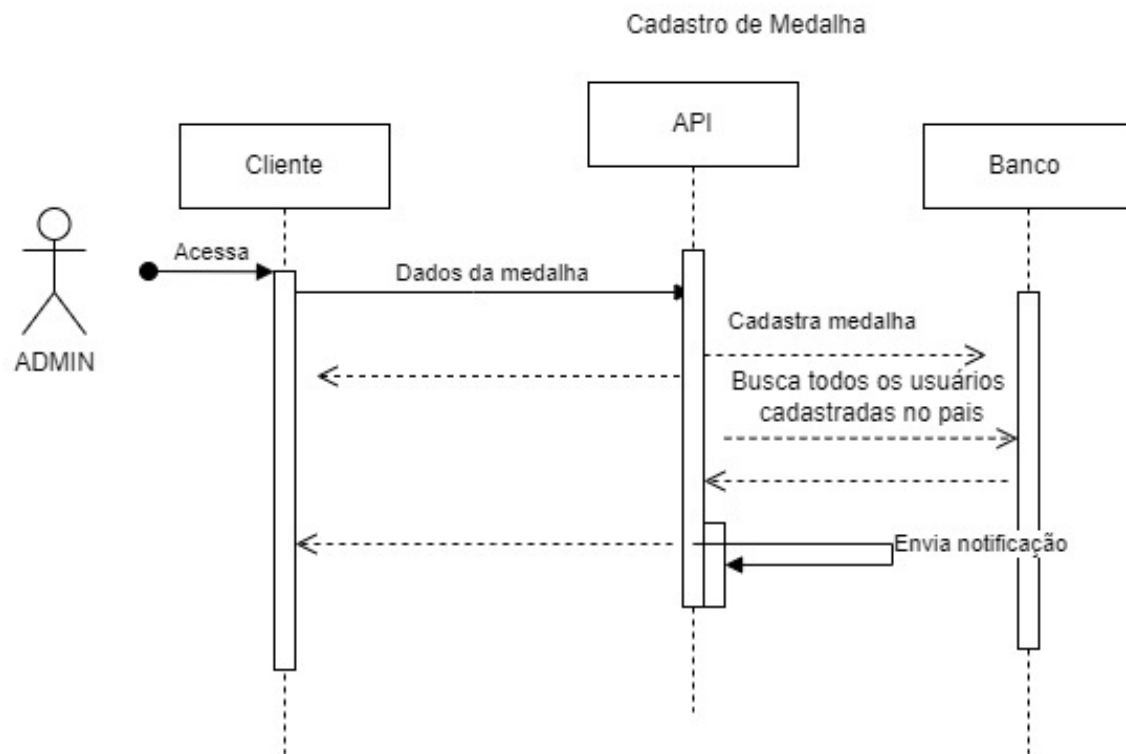
1.3 Diagrama de sequência



•



•

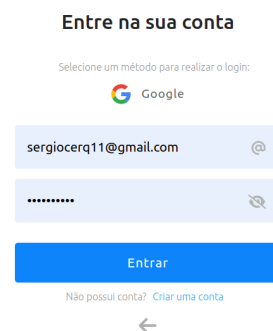


1.4 Fluxo do sistema

- Inicialmente, a tela de início é a landing page, que contém informações sobre as olimpíadas e o quadro de medalhas.




- Para realizar o login, é só clicar no botão "Login" localizado na barra de navegação.



- Caso não tenha uma conta ainda, você pode criar uma se cadastrando no sistema.




-
- Com o login realizado, é possível acessar a conta ou seguir um país.




Experiência Personalizada

Garantimos que cada usuário tenha uma experiência única e personalizada, com foco nos países que mais lhe interessam.



Acessibilidade
















Nossa plataforma está comprometida em ser inclusiva e acessível a todos, com uma interface fácil de usar e compatível com dispositivos móveis.

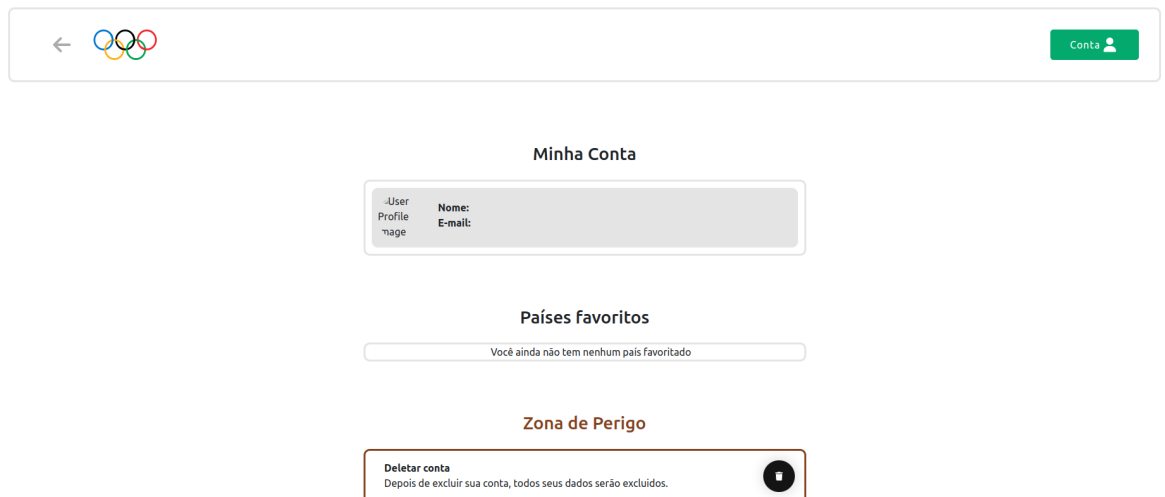


Histórico Olímpico

Veja o histórico de participações dos países, com detalhes sobre medalhas e os atletas que venceram as modalidades.

Quadro de Medalhas

País	Ouro 	Prata 	Bronze 	Total	Ações
 Brasil	3	2	3	8	
 Afeganistão	0	0	0	0	
 Albânia	0	0	0	0	
 Argélia	0	0	0	0	
 Andorra	0	0	0	0	
 Armênia	0	0	0	0	



1.5 Tecnologias escolhidas

1.5.1 Spring Boot - Backend

- Eureka Server: O Eureka Server foi utilizado como servidor de descoberta de serviços para microsserviços. Ele permite que os serviços se registrem e descubram uns aos outros de forma automática, simplificando a comunicação entre os microsserviços sem a necessidade de configurar manualmente endereços IP ou URLs.
- Gateway: O API Gateway atua como um ponto de entrada único para todas as requisições que chegam aos microsserviços. Ele oferece funções como roteamento e balanceamento de carga, simplificando a interação com o backend e evitando expor diretamente os microsserviços ao público.
- Email: A API de Email do Google foi utilizada no sistema para enviar notificações de forma eficiente e segura. Utilizar uma api externa facilita a correta entrega dos e-mails enviados para os usuários, como as notificações sobre medalhas conquistadas pelos países que eles seguem.
- Rabbitmq: O RabbitMQ é utilizado para comunicação assíncrona entre microsserviços, garantindo que as mensagens (notificações de conquistas de medalhas) sejam processadas de forma robusta e sem sobrecarregar os serviços.
- OpenFeign: O OpenFeign foi utilizado para facilitar a comunicação entre microsserviços por meio de chamadas HTTP. Ele permite criar clientes HTTP de forma declarativa, simplificando a interação entre serviços de maneira mais legível e reutilizável.

- Flyway: O Flyway foi utilizado para gerenciar as migrações do banco de dados de forma controlada e automática. Ele permite versionar alterações no esquema do banco de dados, garantindo que todas as instâncias do sistema estejam sincronizadas com a versão correta do banco.
- Validation API: Ela foi usada para garantir que os dados de entrada sejam validados antes de serem processados pelo sistema. Isso inclui validação de campos obrigatórios, formatos de dados, tamanhos mínimos/máximos, evitando que quaisquer informações sejam cadastradas no banco de dados.
- JWT Token: O JWT foi utilizado para gerenciar a autenticação e autorização no sistema. Ele permite que o backend valide os usuários de forma segura e eficiente, sem a necessidade de manter sessões no servidor.
- Swagger: O Swagger foi integrado para documentar e testar as APIs do sistema de forma automática.

1.5.2 ReactJS - Frontend

- Chakra UI: O Chakra UI foi escolhido como biblioteca de componentes para construir interfaces de usuário acessíveis, responsivas e facilmente customizáveis. Ele oferece uma grande variedade de componentes prontos, permitindo o desenvolvimento rápido de interfaces.
- Axios: O axios foi utilizado pela sua simplificação na hora de realizar requisições HTTP. Dentro do front-end, foi criada uma estrutura versátil e dinâmica, sob o uso das rotas e métodos do axios, facilitando e abstraindo dessa forma, a lógica de requisições HTTP.
- Firebase: O firebase foi escolhido para gerenciar serviços em tempo real, como a autenticação dos usuários. No caso do Olympicsfollow, o firebase pôde ser utilizado para autenticação de usuários utilizando os serviços da Google.
- Framer Motion: O framer motion foi utilizado para adicionar animações suaves e interativas à interface. Ele é uma poderosa biblioteca para animações em React, permitindo criar transições e interações visuais que tornam a interface mais fluida.
- Lottie React: essa biblioteca foi escolhida para incorporar animações complexas de forma otimizada. Isso permite a exibição de animações leves e de alta qualidade no frontend, como é o exemplo das animações nas telas de login, register e landing.

- React Router DOM: O React Router DOM foi escolhido para gerenciar as rotas do sistema de forma eficiente. Ele permite a navegação entre diferentes páginas do sistema de maneira simplificada, permitindo diversas configurações para as rotas, além de rotas protegidas, ou seja, rotas que só são acessíveis com autenticação.
- Sonner: O Sonner foi escolhido para fornecer notificações visuais e alertas na tela do usuário. Ele facilita a exibição de mensagens para o mesmo, como confirmações de ações ou mensagens de erro.

1.6 Instruções para instalação e execução

As instruções para instalação e execução do sistema podem ser encontradas nos seguintes repositórios:

1.6.1 Backend

<https://github.com/vlopess/OlymFollow-Backend>

1.6.2 Frontend

<https://github.com/sergiocerq/OlymFollow-FrontEnd>