**UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL**

**FACULDADE DE COMPUTAÇÃO**

**BACHARELADO EM SISTEMAS DE INFORMAÇÃO**


Murilo Esteca Arelhano, Mariane Mori Guiraldelli

# GO DEVREL: UMA FERRAMENTA PARA APOIAR PROFISSIONAIS DE *DEVELOPER RELATIONS*

**CAMPO GRANDE**

**2023**

Murilo Esteca Arelhano, Mariane Mori Guiraldelli

# GO DEVREL: UMA FERRAMENTA PARA APOIAR PROFISSIONAIS DE *DEVELOPER RELATIONS*

Monografia de Graduação apresentada à Faculdade de Computação da Universidade Federal de Mato Grosso do Sul como requisito parcial para a obtenção do grau de bacharel em Sistemas de Informação.

Orientador

Prof. Dr. Awdren de Lima Fontão

**CAMPO GRANDE**

**2023**

## RESUMO

O mercado de tecnologia é competitivo e dinâmico, com empresas buscando atrair desenvolvedores para suas plataformas e tecnologias. Nesse cenário, os profissionais de DevRel (do inglês, *Developer Relations)* são cada vez mais importantes, pois ajudam as empresas a estabelecerem interações de confiança e colaboração com as comunidades de desenvolvedores. Por isso, é muito importante o surgimento de estratégias validadas no mercado, para mitigar riscos e problemas de DevRel. A partir desses pontos, o intuito desse trabalho é criar uma ferramenta web que ajude no compartilhamento do modelo Go DevRel para profissionais de *Developer Relations*, além de mostrar as decisões tomadas nesse processo. Além disso, apoiar os profissionais de DevRel, comunidades de desenvolvedores e coletar *feedbacks* desses profissionais, analisá-los e extrair mais conhecimento dos resultados desses *feedbacks.*



**Palavras-chave:** Developer Relations; DevRel; estágios; lições aprendidas; ecossistema de software.

## ABSTRACT

The technology market is competitive and dynamic, with companies striving to attract developers to their platforms and technologies. In this scenario, DevRel professionals (short for Developer Relations) are becoming increasingly important as they help companies establish trusted and collaborative interactions with developer communities. Therefore, the emergence of market-validated strategies to mitigate DevRel risks and issues is of great importance. The purpose of this work is to create a web tool that facilitates the sharing of the Go DevRel model among Developer Relations professionals, while also showcasing the decisions made during this process. Additionally, it aims to support DevRel professionals, developer communities, and gather feedback from these professionals, analyze them, and extract further knowledge from the results of this feedback.



**Keywords:** Developer Relations; DevRel; stages; lessons learned; software ecosystem.

## SUMÁRIO

## LISTA DE TABELAS

## LISTA DE FIGURAS

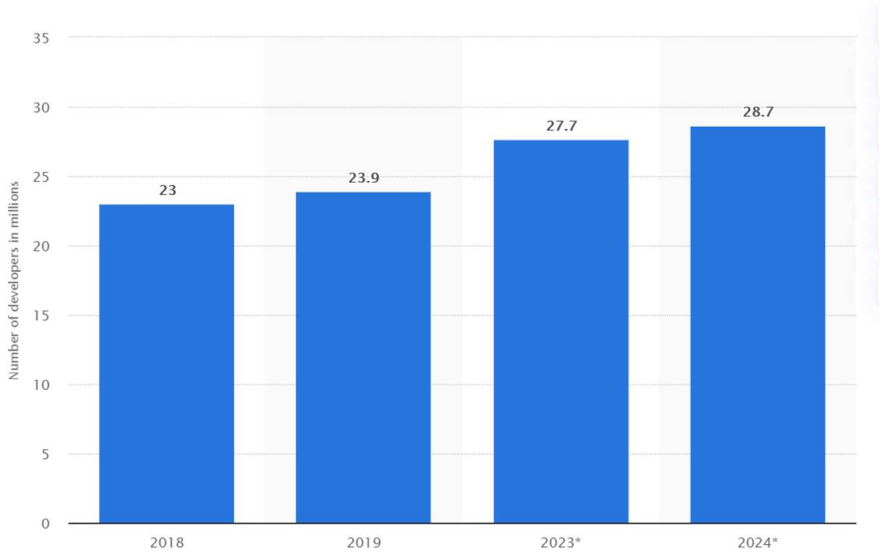
# 1 INTRODUÇÃO

## 1.1 Contexto

É fato que a internet e a tecnologia tiveram um salto gigantesco nas últimas décadas, e isso acarretou o surgimento de inúmeras áreas de pesquisa e trabalhos relacionados. Juntamente ao crescimento dessas, o crescimento do número de desenvolvedores também aumentou consideravelmente, como mostrado na

Figura 1(Evans Data, 2022).

Figura 1 – Número de desenvolvedores de software (em milhões) pelo mundo




Fonte: https://www.statista.com/statistics/627312/worldwide-developer-population/

Novos temas começaram a entrar em pauta, tais como sistemas operacionais, novas linguagens de programação, plataformas de desenvolvimento, inúmeras bibliotecas, *frameworks* e ferramentas que compunham os Ecossistemas de Software (ECOS).

Os ECOS são conjuntos complexos de software, serviços, plataformas, recursos e comunidades que interagem entre si de forma a fornecer soluções de software para usuários. Esses ecossistemas podem ser compostos por uma variedade de atores, incluindo fornecedores de software, desenvolvedores, provedores e comunidades de desenvolvimento. Além de artefatos de softwares que compartilham de uma tecnóloga

e plataformas em comum (Santos & Viana, 2016). Esse tipo de apoio ao desenvolvedor é fundamental para manter o crescimento e interesse no ECOS (Santos & Viana, 2016).

Esses ECOS precisavam cada vez mais de desenvolvedores interessados em suas tecnologias, para que o ECOS possa se manter recebendo contribuições (novas e atualizações), com a finalidade de conseguir atender à crescente demanda dos usuários. Por essas razões, as empresas precisaram passar a investir seu tempo em atrair mais desenvolvedores e disseminar o uso das ferramentas providas em seu ECOS e entender o que faria com que novos desenvolvedores ficassem instigados a usar suas plataformas (Thengvall, 2018).

Um exemplo é o ECOS o iOS da Apple, que inclui o sistema operacional iOS, a App Store[1], o Xcode[2] (ferramenta de desenvolvimento), além de outros serviços, como iCloud, Apple Music[3] e Apple Pay. Esse ecossistema permite que desenvolvedores criem aplicativos para a plataforma iOS, que sejam compatíveis com outros serviços da Apple, como o Apple Watch e iPhones. A App Store, por sua vez, oferece uma plataforma de distribuição das aplicações centralizada e segura para seus usuários.

Esses tipos de ferramentas e aplicações são fundamentais para funcionamento do ECOS, porém o que define o sucesso não é apenas a quantidade de serviços oferecidos e sim a boa qualidade do ECOS, praticidade, facilidade e experiência que o desenvolvedor tem ao criar aplicações com essa plataforma.

## 1.2 Motivações

Com o crescimento das empresas e com o surgimento de novos ECOS, a preocupação em trazer novos desenvolvedores para essas plataformas também aumentou e começaram a surgir mais estratégias para melhorar as interações entre os desenvolvedores e os ECOS. Encontrar uma estratégia eficaz para gerenciar esses ambientes e manter os desenvolvedores na plataforma não é uma tarefa fácil. As

[1] https://www.apple.com/br/app-store/
[2] https://developer.apple.com/xcode/
[3] https://music.apple.com/br/browse

empresas começaram a procurar profissionais para comunicar, monitorar e ajudar os desenvolvedores com questões em relação com os ECOS. Isso é essencial para poderem continuar criando tecnologias e crescendo suas plataformas de software (Fontão, 2019). Dessa nova demanda surgiu o profissional de DevRel, ou Relação com Desenvolvedores (do inglês*, Developer Relations*), que ficou responsável por gerenciar essa relação com os desenvolvedores (Singh Gill, Pawanpreet, 2022). Essa área é responsável por criar uma comunidade engajada e satisfeita de desenvolvedores, fornecendo suporte técnico e recursos para ajudá-los a criar soluções utilizando a tecnologia da empresa. Essa relação é fundamental para o sucesso de uma empresa de tecnologia, já que os desenvolvedores engajados e influentes podem ajudar a promover a marca, fornecer um *feedback* valioso sobre os produtos e serviços da empresa e até mesmo criar soluções e ideias usando a tecnologia disponibilizada (Singh Gill, 2022).

Tendo em vista da importância do profissional de DevRel dentro das empresas de tecnologia que planejam expandir seus ECOS, esses profissionais devem desenvolvam técnicas e ferramentas para ajudar na governança de todo esse ECOS e melhorar o engajamento da comunidade (Oliveira, et al., 2021).

Entretanto essa não é uma tarefa tão trivial, assim como a governança de outras áreas da tecnologia e desenvolvimento de software, a gestão desses ecossistemas pode se tornar muito complexas ao decorrer do crescimento da comunidade entorno do ECOS (Fontão, 2019). E por esse motivo a criação de modelos de governança e ferramentas que possam ajudar esses profissionais a gerenciarem e monitorar esses ecossistemas se torna tão importante e essencial para o amadurecimento do ECOS.

## 1.3 Objetivos

O objetivo deste trabalho é desenvolver uma ferramenta que possa apoiar os profissionais de DevRel compartilhando o conhecimento contido no modelo DevGo[4] e suas lições aprendidas, além de criar uma forma de receber *feedbacks* desses

[4] : DevGo: um modelo para governança de desenvolvedores em ecossistema de software móvel a partir de Developer Relations (Fontão, 2019).

profissionais para melhorar a base de conhecimento e trazer novas lições aprendidas. Criar também uma funcionalidade que direcione o usuário para o estágio no qual se encontra para facilitar o aprendizado do modelo.

### 1.4 Organização

Este trabalho está organizado em quatro capítulos. O Capítulo 1 descreve a introdução dando um contexto sobre ao qual esse trabalho está inserido e apresenta inicialmente as motivações e objetivos esperados com o desenvolvimento do trabalho. O Capítulo 2 apresenta os trabalhos relacionados e ferramentas. O Capítulo 3 apresenta o referencial teórico necessário para entender o escopo do trabalho, ferramentas utilizadas em seu desenvolvimento, técnicas aplicadas em partes da aplicação e outros conceitos necessários. O Capítulo 4 apresenta tópicos relacionados ao processo de desenvolvimento do software, arquitetura, coleta e análise de requisitos, além de mostrar as funcionalidades presentes na aplicação. Por fim, o Capítulo 5 traz as conclusões e considerações finais desse trabalho.

## 2 TRABALHOS RELACIONADOS

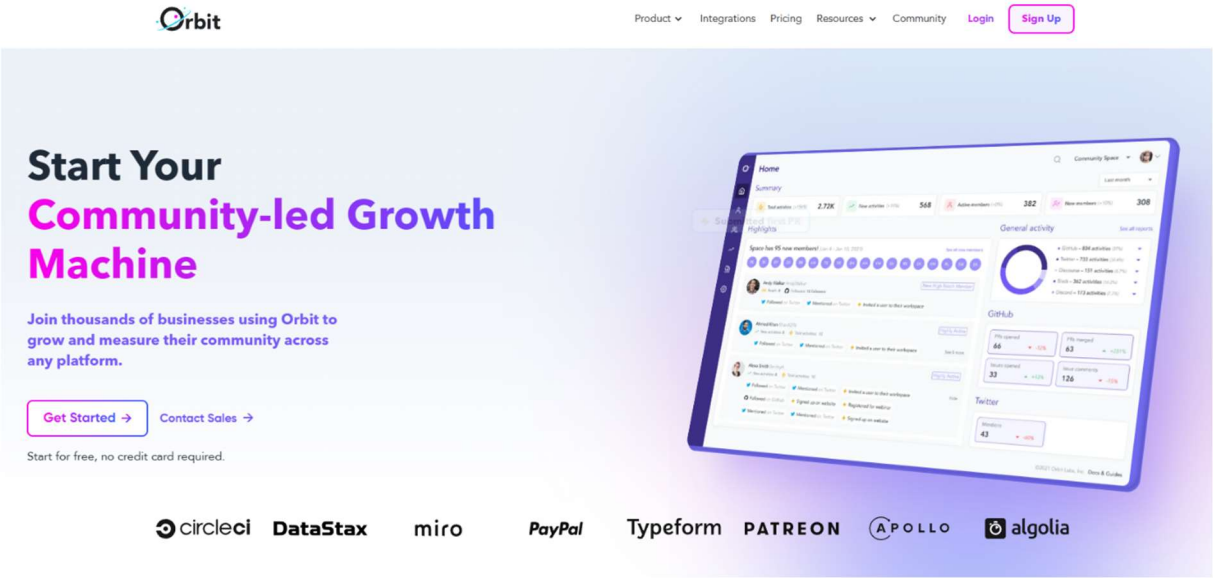
Neste capítulo serão apresentados ferramentas e trabalhos relacionados que auxiliam os profissionais de *Developer Relations*.

### 2.1.1 Orbit

A ferramenta Orbit é um sistema de gestão[5] que realiza a integração de dados com diversas plataformas de desenvolvimento e comunicação e auxilia os gestores da comunidade na visualização de atividades que acontecem na comunidade, por exemplo, mensagens e interações em chats de conversa, atividades realizadas em repositórios de código e outras funcionalidades de maneira centralizada. Dessa forma, a principal proposta da ferramenta é auxiliar os profissionais de DevRel no monitoramento da comunidade do ECOS.

[5] https://orbit.love/

Figura 2 - Página inicial do website da Orbit



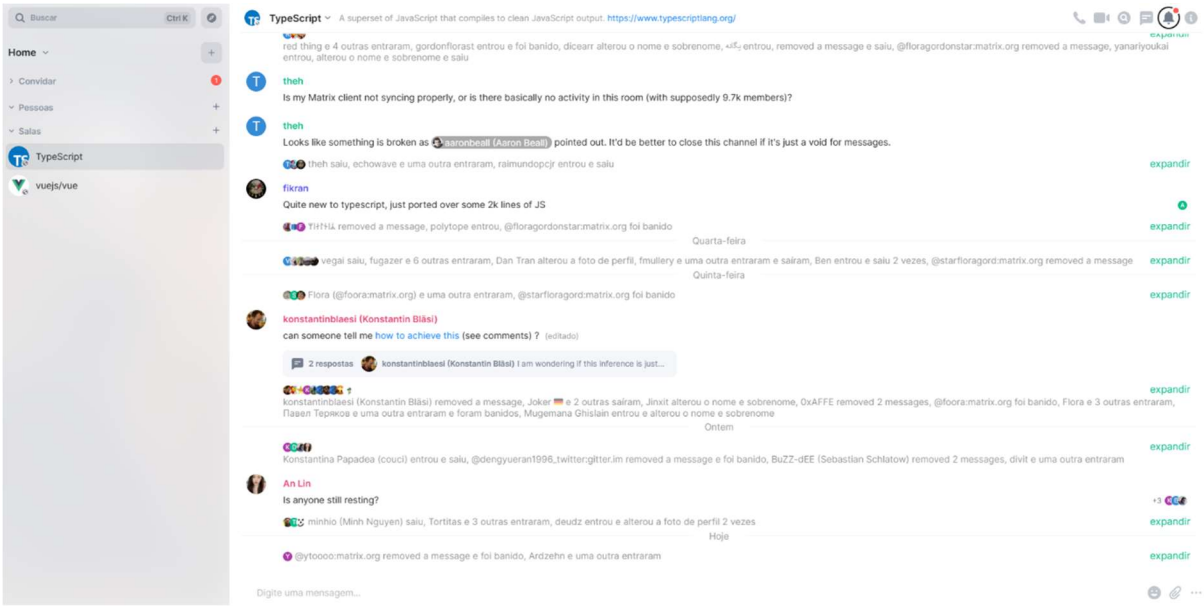

Fonte: https://orbit.love/

### 2.1.2 Gitter

O aplicativo Gitter é uma plataforma de chat para gerenciar e conectar comunidades de desenvolvedores[6]. Com o Gitter é possível criar salas de bate-papo com os usuários para gerar discussões e coletar *feedbacks* das comunidades entorno do ECOS, como mostrado na Figura 3. Os usuários podem interagir entre si ajudando uns aos outros com resoluções de problemas ou discussões sobre melhorias e novas funcionalidades.

[6] https://gitter.im/

Figura 3 – Tela da comunidade do TypeScript no Gitter

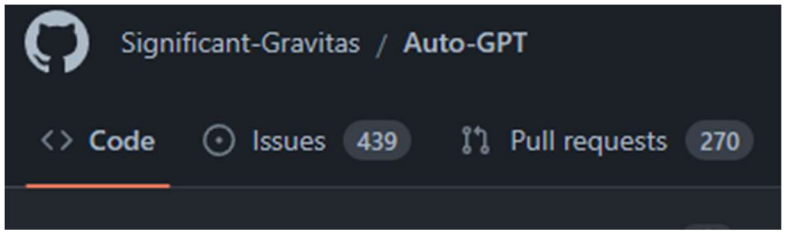



Fonte: https://app.gitter.im/#/room/#typescript:gitter.im

### 2.1.3 GitHub

O Github[7] apesar de ser conhecida como um serviço para repositórios Git, oferece diversas funcionalidades para criar e se comunicar com a comunidade de desenvolvedores entorno de um ECOS. Existem diversos serviços que o GitHub oferece que podem auxiliar o profissional de DevRel na gestão das comunidades. Dentre elas, estão a possibilidade de abrir novas *issues*, pull *requests*, criar *threads* de discussões sobre o projeto e ainda gerenciar as tarefas que estão sendo executadas pela equipe.

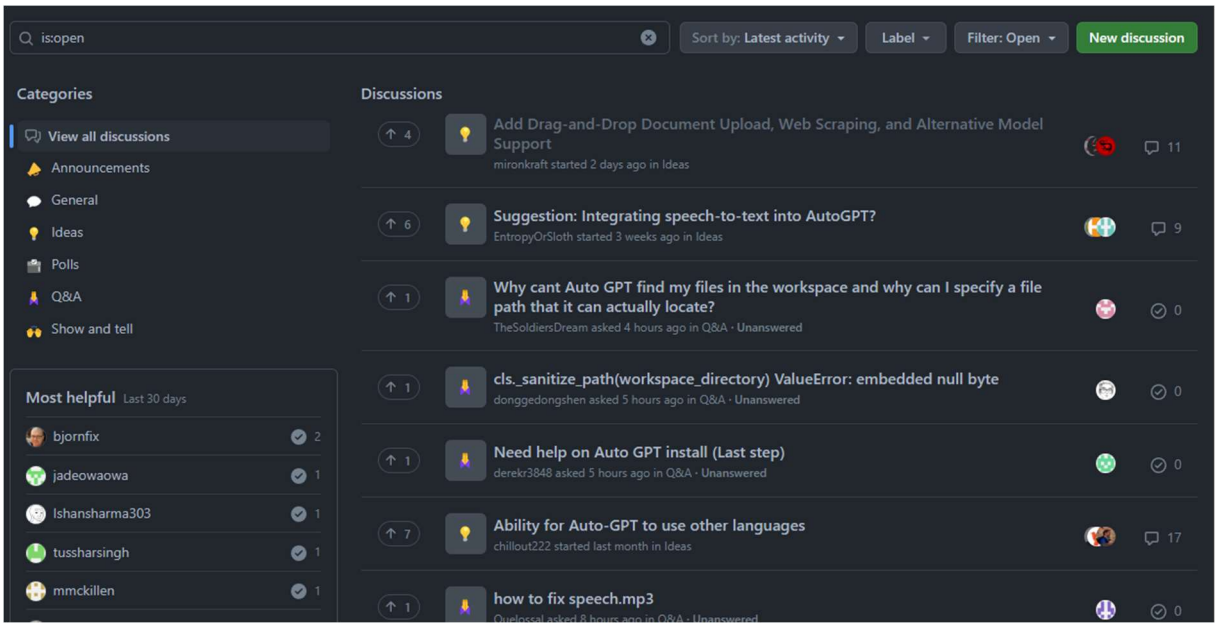
Figura 4 – Ferramentas de *Issues* e *Pull Requests* do github




Fonte: https://github.com/Significant-Gravitas/Auto-GPT

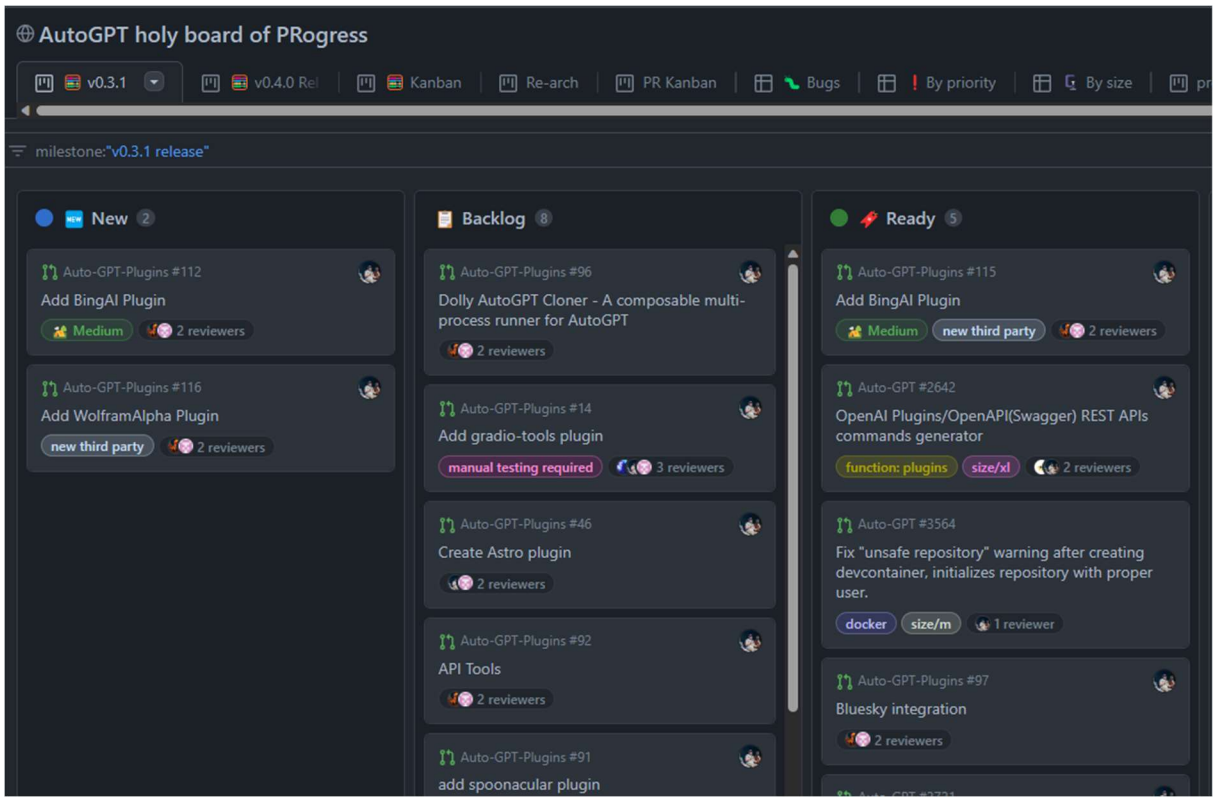
[7] https://github.com/

Figura 5 – Ferramenta para gerar discussões sobre o projeto




Fonte: https://github.com/Significant-Gravitas/Auto-GPT/discussions

Figura 6 – Ferramenta de gestão do projeto




Fonte: https://github.com/orgs/Significant-Gravitas/projects/1

# 3 REFERENCIAL TEÓRICO

Neste capítulo serão apresentados conceitos fundamentais para o desenvolvimento da ferramenta Go DevRel, englobando conceitos sobre *o Modelo DevGo* e ferramentas que permearam o desenvolvimento do software e infraestrutura necessária para o funcionamento da aplicação.

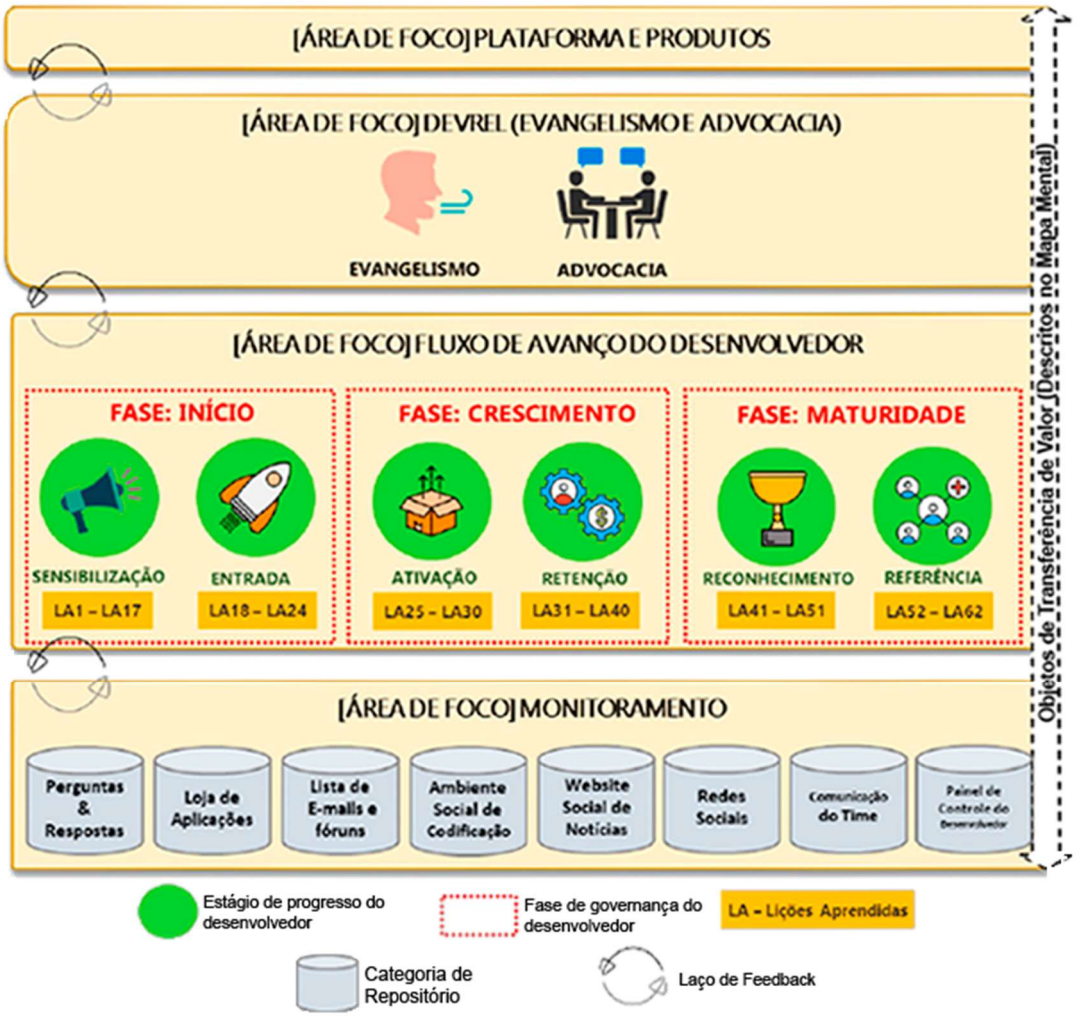
## 3.1 Modelo DevGo

O DevGo (do inglês, *Developer Governance*) consiste em um modelo composto por elementos estruturais e por um conjunto de lições aprendidas para a criação e manutenção de um ECOS próspero para a organização central e para os desenvolvedores (Fontão, 2019). Os elementos estruturais dos modelos ajudam na organização e visão de quais elementos estão sendo aplicados dentro do ECOS.

Na camada mais alta do modelo estão presentes as **áreas de foco** que representam as áreas no qual a organização que cuida do ECOS deve gerenciar e monitorar. Cada uma dessas áreas contém, pelo menos, um objetivo organizacional e cada uma das áreas de foco podem conter fases que estão relacionadas com o fluxo do desenvolvedor no ecossistema.

O modelo compreende Quatro áreas de foco: Plataforma e Produtos, DevRel (Evangelismo e Advocacia), Fluxo de Avanço do Desenvolvedor e, por fim, Monitoramento (Fontão, 2019) como mostrado na Figura 7. Cada uma destas áreas de foco que compõem a abordagem proposta pelo modelo e ajudam na organização para a governança dos desenvolvedores.

Figura 7 – Áreas de foco




Fonte: DevGo: um modelo para governança de desenvolvedores em ecossistema de software móvel a partir de developer relations. (Fontão, 2019)

Em especial a área de foco denominada “Fluxo de Avanço do Desenvolvedor” mostrado na Figura 7 contém três fases, as quais contém estágios que compreendem um período de desenvolvimento do desenvolvedor (Fontão, 2019), e cada um desses estágios contém um conjunto lições aprendidas.

## 3.2 Sistema de controle de versões

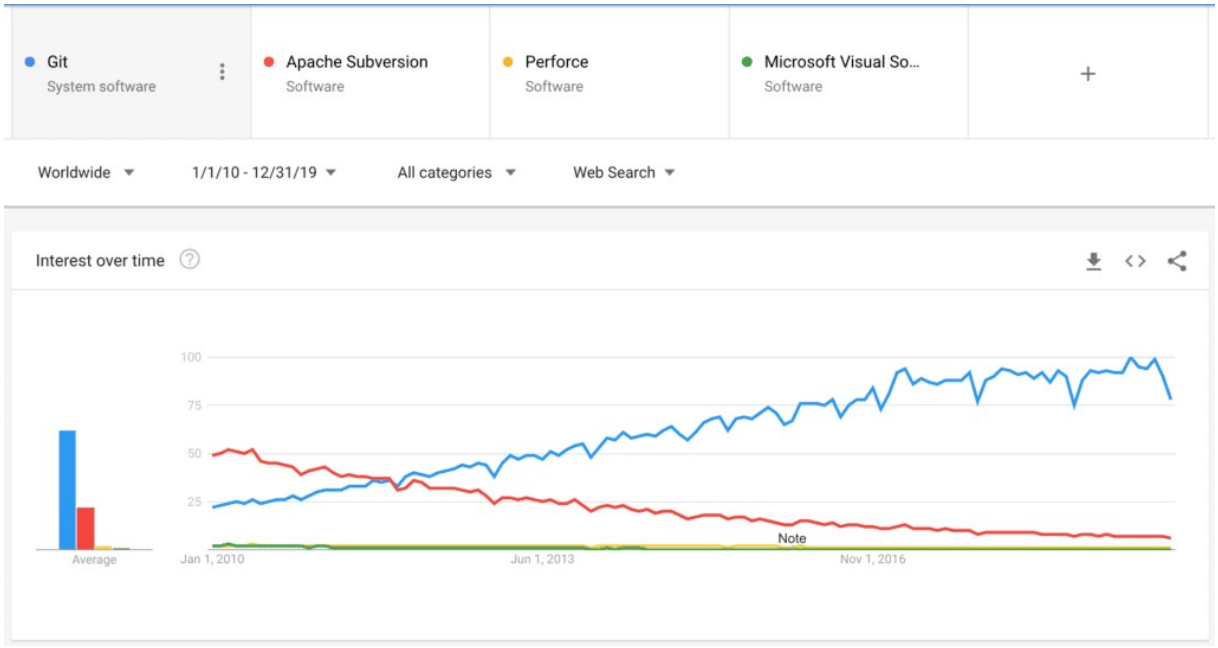
A gigantesca maioria dos sistemas são desenvolvidos por equipes com diversas pessoas, em lugares diferentes e máquinas diferentes. Essa distribuição e descentralização no processo de desenvolvimento de softwares levou a criação de sistemas para gerenciamento do código de maneira distribuída e organizada.

Atualmente o mais popular e adotado pela comunidade no geral é o Git, nessa seção serão apresentadas as ferramentas de versionamento e gerenciamento de códigos que foram utilizadas na criação da ferramenta Go DevRel.

### 3.2.1 Git

O Git é um sistema de controle de versão distribuído, que permite que desenvolvedores trabalhem em conjunto em projetos de software, rastreando as alterações feitas e mantendo um histórico de todas as versões do código. Ele foi criado em 2005 por Linus Torvalds[8], o criador do sistema operacional Linux, e desde então se tornou uma das ferramentas mais populares entre os desenvolvedores de software, como mostrado na Figura 8, esse comportamento ainda se mostra mais discrepante atualmente.

Figura 8 – Popularidade dos sistemas de controle de versões de 2010 a 2019




Fonte: https://sourcelevel.io/blog/7-git-best-practices-to-start-using-in-your-next-commit

A importância do Git se dá pelo fato de que ele permite que equipes trabalhem de forma colaborativa e organizada, garantindo que as alterações feitas no código-fonte possam ser facilmente rastreadas e revertidas, caso necessário. Além disso, o

[8] https://www.atlassian.com/br/git/tutorials/what-is-git

Git facilita a integração de diferentes contribuições de código e a realização de testes de software automatizados, o que ajuda a manter a qualidade do código e a evitar erros e conflitos.

Entre os motivos para escolher usar o Git hoje em dia estão a sua facilidade de uso, a sua ampla adoção pela comunidade de desenvolvedores, a sua capacidade de gerenciar grandes projetos e a sua integração com outras ferramentas de desenvolvimento de software, como o GitHub. Além disso, o Git oferece recursos avançados, como ramificação de código (*branching*), que permite que diferentes versões do código sejam desenvolvidas ao mesmo tempo, sem interferir umas nas outras.

### 3.2.2 GitHub

O GitHub é uma plataforma online que fornece hospedagem para projetos de software usando o Git como sistema de controle de versão[9]. Ela permite que desenvolvedores trabalhem em conjunto, compartilhem código e documentação, gerenciem *bugs* e recebam *feedback* de outros desenvolvedores. Além disso, o GitHub disponibiliza ferramentas para gestão de projetos e integração contínua, o que pode otimizar o processo de desenvolvimento de software.

Existem vários motivos para aderir ao isso do Github, mas a principal motivação a adoção da plataforma é que, atualmente o GitHub a ferramenta mais utilizada pela comunidade de desenvolvedores, que facilita a o uso dado a quantidade de conteúdo disponível online. Outra razão para utilizar o GitHub é sua capacidade de se integrar com outras ferramentas de desenvolvimento de software, como o JIRA, o Trello e o Slack. Além disso, o GitHub disponibiliza recursos avançados, como integração contínua e implantação contínua, que permitem que o código seja testado e implantado automaticamente em servidores de produção, acelerando o processo de desenvolvimento.

## 3.3 Aplicações Web

[9] https://docs.github.com/en/get-started/quickstart/hello-world

Aplicações web são softwares projetados para serem acessados através de um navegador web. As aplicações web são cada vez mais relevantes no mundo digital, oferecendo uma série de vantagens em relação aos aplicativos tradicionais, como a facilidade de acesso, a disponibilidade em qualquer lugar com conexão à internet, a escalabilidade e a manutenção simplificada.

As aplicações web comumente têm uma arquitetura de cliente-servidor (do inglês, *client/server*). A parte do cliente, que também pode ser chamada de *frontend,* é a parte da aplicação que é executada no navegador do usuário. Ela é responsável pela apresentação da interface do usuário e interação com o usuário, bem como pela coleta de dados para serem enviados para o servidor. O servidor, também chamado de *backend*, é responsável pelo processamento dos dados recebidos do cliente, pela interação com o banco de dados e pela execução de outras tarefas lógicas e de negócios.

Diferentemente dos sistemas tradicionais *desktop*, as aplicações web não requerem instalação, sendo acessíveis através de qualquer navegador web, e em qualquer plataforma. As atualizações em aplicações web são geralmente feitas no servidor e refletidas instantaneamente em todos os usuários, enquanto nos sistemas desktop as atualizações precisam ser instaladas manualmente em cada máquina. Comumente exigem menos recursos de hardware e software do que os sistemas desktop, o que pode resultar em custos menores de manutenção e suporte.

Sendo assim, as aplicações web oferecem várias vantagens, incluindo maior acessibilidade, facilidade de atualização, compatibilidade multiplataforma e custos reduzidos.

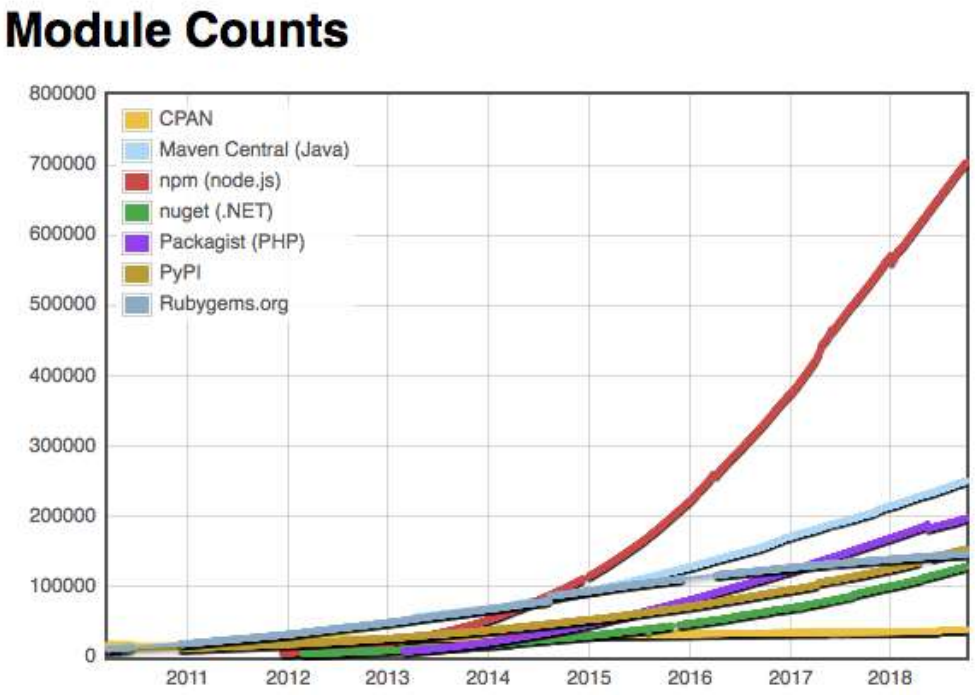
### 3.4 Node.js

A rápida adoção de aplicações web no mundo todo trouxe consigo diversas tecnologias novas e impactantes para o mercado, mas sem dúvidas umas das tecnologias mais promissoras até os dias atuais é o Node.js. O Node.js se autodenomina como um *runtime* de código aberto para JavaScript (JS), linguagem que é usada por padrão para programação de websites nos navegadores. Por esse motivo a adoção dessa plataforma foi extraordinariamente grande, pois possibilitava o uso de JavaScript, que já era uma linguagem amplamente conhecida, no ambiente

local fora do navegador de internet, diminuindo o atrito de novos desenvolvedores *backend* e melhorando a compreensão do código. O motor V8 utilizado pelo Google Chrome é o que possibilita que o Node.js execute código JS fora do navegador.

Outra vantagem do Node.js é a sua compatibilidade com uma ampla variedade de bancos de dados, incluindo MongoDB, MySQL, PostgreSQL e Redis, permitindo que os desenvolvedores escolham a melhor opção para atender às necessidades específicas do seu aplicativo. O Node.js também foi adotado por diversas empresas do setor de tecnologia e serviços, tais como Netflix, PayPal, LinkedIn, e Walmart, várias outras também usam o Node.js em partes de suas aplicações, o que colaborou ainda mais para sua adoção, aumento no número de bibliotecas e adesão da comunidade (Figura 9).

Figura 9 – Contagem de módulos do node em relação a outras linguagens



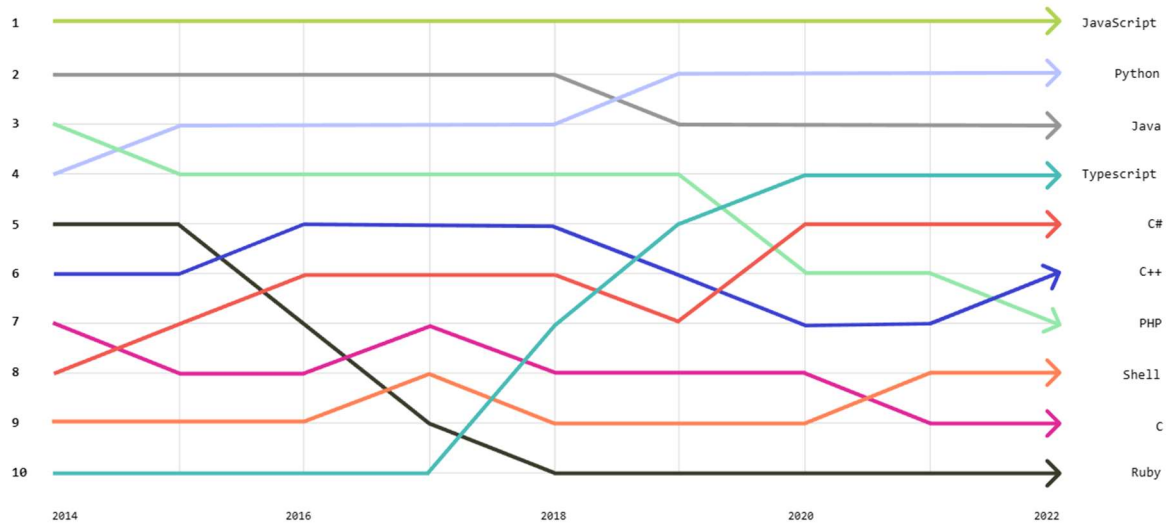

Fonte: https://blog.npmjs.org/post/180868064080/this-year-in-javascript-2018-in-review-and-npms.html

Até o memento da escrita desse trabalho o repositório do Node.js no GitHub conta com mais de 95 mil estrelas e 1.3 mil *issues* abertas[10], o que demostra um

[10] https://github.com/nodejs/node

grande envolvimento da comunidade de desenvolvedores com a plataforma e com o JavaScript, como podemos observar na Figura 10. Filtrando pelas vagas disponíveis no LinkedIn por Node.js, foram encontradas mais de 4.5 mil vagas abertas no momento, essa adoção do Node.js pelos desenvolvedores faz que a plataforma tenha diversas bibliotecas úteis para o desenvolvimento, além de uma vasta documentação e conteúdo de apoio pela internet.

Figura 10 – Linguagens de programação mais usadas em 2022




Fonte: https://octoverse.github.com/2022/top-programming-languages

Tendo isso em vista, fica evidente que as motivações para a adoção dessa plataforma são inúmeras, a plataforma é eficiente, popular, escalável e com desempenho ótimo para aplicações de diversos portes.

### 3.4.1 React

React é um dos *frameworks* mais populares para o desenvolvimento de aplicações em JavaScript[11]. Criada pelo Facebook, ela foi lançada em 2013 e, desde então, tem ganhado cada vez mais adeptos no mercado de desenvolvimento web, uma grande comunidade e o surgimento de diversas bibliotecas que trabalham em conjunto com o ecossistema do React.

[11] https://react.dev/

Uma das grandes vantagens do React é sua capacidade de criar interfaces de usuário dinâmicas e responsivas, com alto desempenho e facilidade de manutenção. Isso é possível graças ao seu sistema de componentes reutilizáveis, que permite que o desenvolvedor crie e mantenha a interface de forma modular, facilitando a reutilização de código e tornando o processo de desenvolvimento mais ágil, utilizando para isso o JSX que é uma extensão de sintaxe para JavaScript que facilita na manutenção e na criação de componentes Figura 11.

Figura 11 – Exemplo da sintaxe do JSX


Fonte: https://pt-br.legacy.reactjs.org/docs/introducing-jsx.html

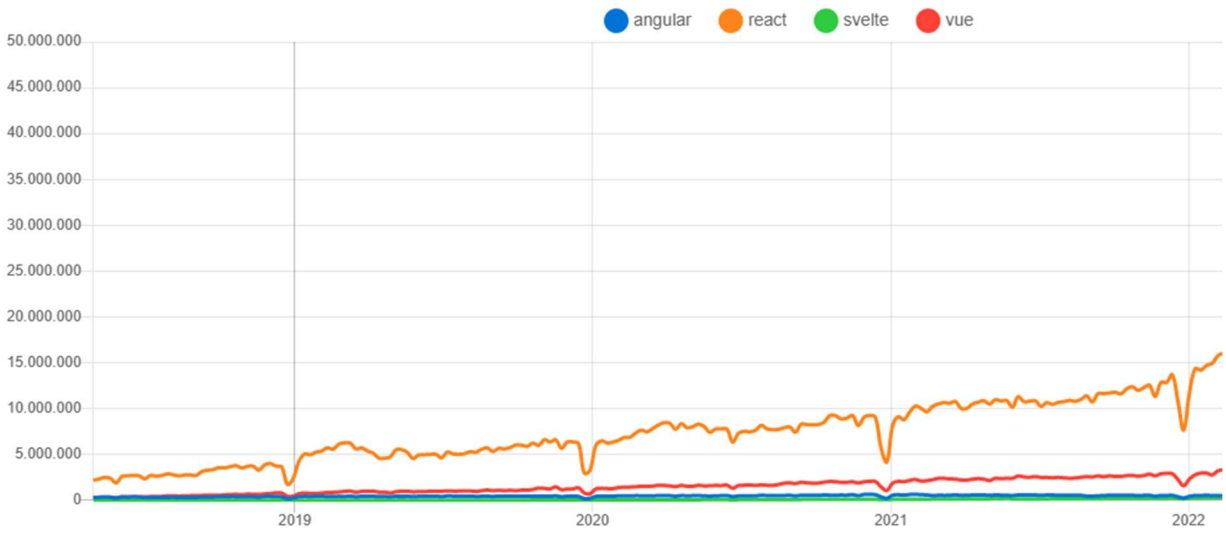
Além disso, o React é altamente flexível e pode ser utilizado tanto para a criação de aplicações web completas quanto para o desenvolvimento de componentes específicos que podem ser incorporados em outras aplicações. Sua comunidade ativa e engajada é uma das maiores vantagens, com muitos recursos, ferramentas e bibliotecas de código aberto.

No mercado atual, o React é um dos *frameworks* mais populares e amplamente utilizadas para o desenvolvimento de aplicações web, como pode ser observado no gráfico da Figura 12. De acordo com uma pesquisa realizada pelo Stack Overflow em 2020, o React é o *framework* mais utilizada pelos desenvolvedores, com mais de 68% dos respondentes afirmando que a utilizam regularmente.

Os motivos para adotar o React em um projeto são muitos. Além das vantagens já mencionadas, o React tem uma curva de aprendizado relativamente baixa, o que torna mais fácil para os desenvolvedores aprenderem e começarem a utilizá-lo. Além disso, sua comunidade ativa e engajada fornece suporte e recursos valiosos, e há

muitas oportunidades de trabalho disponíveis para desenvolvedores que sabem como utilizá-lo.

Figura 12 – Número de downloads do React em relação a outras bibliotecas




Fonte: https://npmtrends.com/angular-vs-react-vs-svelte-vs-vue

Outra grande vantagem do React é sua compatibilidade com outras ferramentas e tecnologias populares, como Redux, GraphQL e Next.js, permitindo a criação de aplicações mais complexas e sofisticadas. Em resumo, o React é um *framework* poderosa e flexível que pode ajudar a criar interfaces de usuário dinâmicas e responsivas de forma mais rápida, fácil e eficiente.

### 3.4.2 NestJs

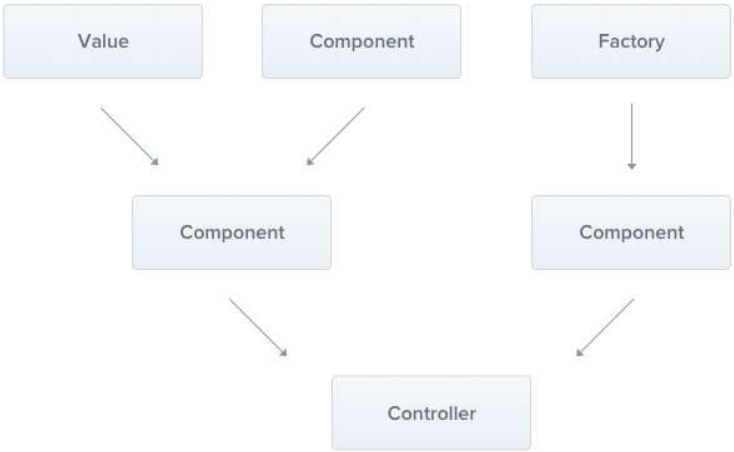
De acordo com os próprios desenvolvedores o NestJs é um "*framework* progressivo para a construção de aplicativos eficientes, confiáveis e escaláveis do lado do servidor". Muito utilizada na comunidade de desenvolvimento com mais de 50 mil estrelas no GitHub e 270 contribuidores no repositório[12]. O principal ponto forte do NestJs é que o *framework* impõe uma arquitetura robusta para aplicação. Essa estrutura tem uma grande influência de outro *framework* chamado Angular que é mais antigo e extensivamente validado pela comunidade.

[12] https://github.com/nestjs/nest

O *framework* traz consigo também um conjunto de bibliotecas e ferramentas que transformam ela em uma plataforma completa para o desenvolvimento de *backend*. Ele é construído com base na arquitetura do Angular, tornando-o uma opção popular para aqueles familiarizados com essa estrutura. O NestJS utiliza TypeScript, uma de programação fortemente tipada, o que ajuda a evitar erros e bugs comuns durante o desenvolvimento.

O NestJS também inclui suporte integrado para a construção de APIs RESTful e GraphQL, o que pode ajudar a agilizar o processo de desenvolvimento. Além de ter uma extensa documentação com diversos exemplos para diferentes casos do dia a dia. Outra vantagem essencial do NestJs é o gerenciamento de injeção de dependências que faz dele um *framework* mais completo.

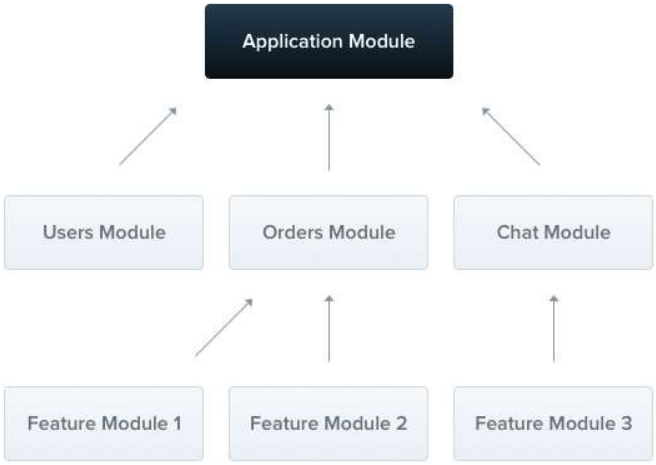
Figura 13 – Exemplo de injeção de dependência em módulos




Fonte: https://docs.nestjs.com/fundamentals/custom-providers

Outro ponto central a arquitetura do NestJs é a divisão da aplicação em módulos, que contêm diferentes provedores de serviços e funcionalidades. Esses módulos são gerenciados pelo sistema de injeção de dependência ajudando na alocação de memória e gerenciamento de dependências. Também ajudam com a organização e desacoplamento da aplicação. Por fim ajuda na organização de testes automatizados na aplicação e compartilhamento dos módulos entre outras aplicações.

Figura 14 – Gerenciamento e dependência de módulos no NestJs




Fonte: https://docs.nestjs.com/modules

O NestJS é uma opção que se encaixa bem para diversos cenários, proporcionando uma arquitetura modular que permite que você crie aplicativos escaláveis e fáceis de manter. Além disso, o NestJS tem uma comunidade ativa e crescente que pode ajudar a apoiar o desenvolvimento contínuo e aprimoramento do *framework*.

### 3.5 PostgreSQL e MongoDB

O PostgreSQL é um sistema gerenciador de banco de dados relacional de código aberto[13]. Ele foi criado em 1986 por um grupo de pesquisadores da Universidade da Califórnia em Berkeley, liderados por Michael Stonebraker. Inicialmente chamado de Postgres[14], o projeto foi desenvolvido a partir de uma tentativa de implementar um sistema de gerenciamento de banco de dados relacional que combinasse as características de um sistema de gerenciamento de banco de dados relacionais com as de um sistema de gerenciamento de banco de dados orientados a objetos.

[13] https://pt.wikipedia.org/wiki/PostgreSQL
[14] https://wiki.postgresql.org/wiki/Introdu%C3%A7%C3%A3o_e_Hist%C3%B3rico

Em 1996, o nome Postgres foi mudado para PostgreSQL para refletir melhor as suas funcionalidades.

O PostgreSQL é amplamente utilizado em diversos setores da indústria, como financeiro, saúde, educação, mídia e tecnologia. O sistema é valorizado pela sua escalabilidade, segurança e confiabilidade. Ele suporta várias linguagens de programação, incluindo Java, Python, C++ e Ruby, e pode ser usado em diversos sistemas operacionais, como Linux, Windows e MacOS. O PostgreSQL também é compatível com o SQL padrão ANSI e suporta recursos avançados, como transações ACID e integridade referencial.

Por outro lado, o MongoDB é um sistema gerenciador de banco de dados NoSQL, criado em 2007 pela empresa 10gen, agora conhecida como MongoDB Inc. O MongoDB foi desenvolvido como uma alternativa ao modelo relacional tradicional e é baseado em um modelo de documentos, onde os dados são armazenados como documentos BSON (*Binary* JSON). O MongoDB é escalável, flexível e fácil de usar, tornando-se uma opção popular para aplicativos da web, aplicativos móveis e *IoT*.

O MongoDB é usado principalmente em aplicativos da web modernos que lidam com grandes quantidades de dados não estruturados. Ele também é amplamente utilizado em áreas como marketing, publicidade e análise de dados. O MongoDB suporta consultas avançadas, índices e replicação automática para alta disponibilidade.

Em termos de diferenças, o PostgreSQL é um sistema de gerenciamento de banco de dados relacional, enquanto o MongoDB é um sistema de gerenciamento de banco de dados NoSQL baseado em documentos. Isso significa que o PostgreSQL é mais adequado para aplicativos que requerem um esquema de banco de dados estruturado, enquanto o MongoDB é mais adequado para aplicativos que requerem flexibilidade e escalabilidade com dados não estruturados. Além disso, o PostgreSQL é mais rigoroso em relação a integridade referencial e transações ACID, enquanto o MongoDB é mais flexível e escalável.

### 3.6 JSON Web Token

O JSON Web Token (JWT) é um padrão aberto e adotado por diversas empresas usado para autenticação e autorização em aplicativos, permitindo que os usuários façam login em suas contas e acessem recursos protegidos. O token é composto de três partes principais, cada uma com seu papel definido para garantir a integridade e confiabilidade das informações, como mostrado nesse exemplo:

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiYWRtaW4iOnRydWV.
TJVA95OrM7E2cBab30RMHrHDcEfxjoYZgeFONFh7HgQ

O token é representado no formato *base64*, e é divido em três partes. A primeira parte representa o cabeçalho do token, onde se encontram as informações, por exemplo, sobre qual o tipo e o algoritmo utilizado para validação do token. Uma das vantagens em seguir essa padronização é que há diversas outras informações que também podem ser adicionadas opcionalmente ao cabeçalho do token. Após a decodificação do token, as informações são apresentadas no formato JSON[15]:

{"alg": "HS256", "typ": "JWT"}

A segunda parte do token contém um corpo com informações que podem variar de acordo com a aplicação e o contexto no qual está sendo usado. Geralmente é utilizado para trafegar dados do usuário autenticado, como permissões e informações pessoais. Tendo em vista que esses dados são validados posteriormente pelo servidor da aplicação, as informações contidas no corpo do token são confiáveis e seguras:

{“sub": "1234567890”, “name": "Johjn Doe", "iat": 1516239022}

Por fim, a última parte do é a assinatura do token, a qual pode ser utilizada um *hash* como segredo ou chaves públicas e privadas. Essa assinatura fica responsável por verificar a validade do token e suas informações, outra possibilidade de assinar o

[15] https://www.json.org/json-pt.html

token é usando JSON Web Signatures (JWS, RFC 7515[16]) e encripta-lo usando JSON Web Encryption (JWE, RFC 75167[17]).

A importância do JWT nas aplicações hoje em dia se deve em parte à crescente necessidade de segurança nas comunicações online. Com a explosão de dados e informações sensíveis que são trocados online, é fundamental proteger as informações dos usuários e garantir que apenas pessoas autorizadas tenham acesso a elas. Além disso o JWT conta com diversas bibliotecas em diversas linguagens amplamente utilizadas no mercado, tornando sua implementação mais fácil e rápida.

Como resultado, há uma demanda crescente por desenvolvedores com conhecimento em JWT e habilidades em autenticação e autorização de aplicativos da web e móveis.

### 3.7 Infraestrutura

Computação em nuvem e serviços de infraestrutura gerenciados em nuvem são termos que vem ganhando força no mercado de maneira acelerada. A computação em nuvem trouxe diversos benefícios, como:

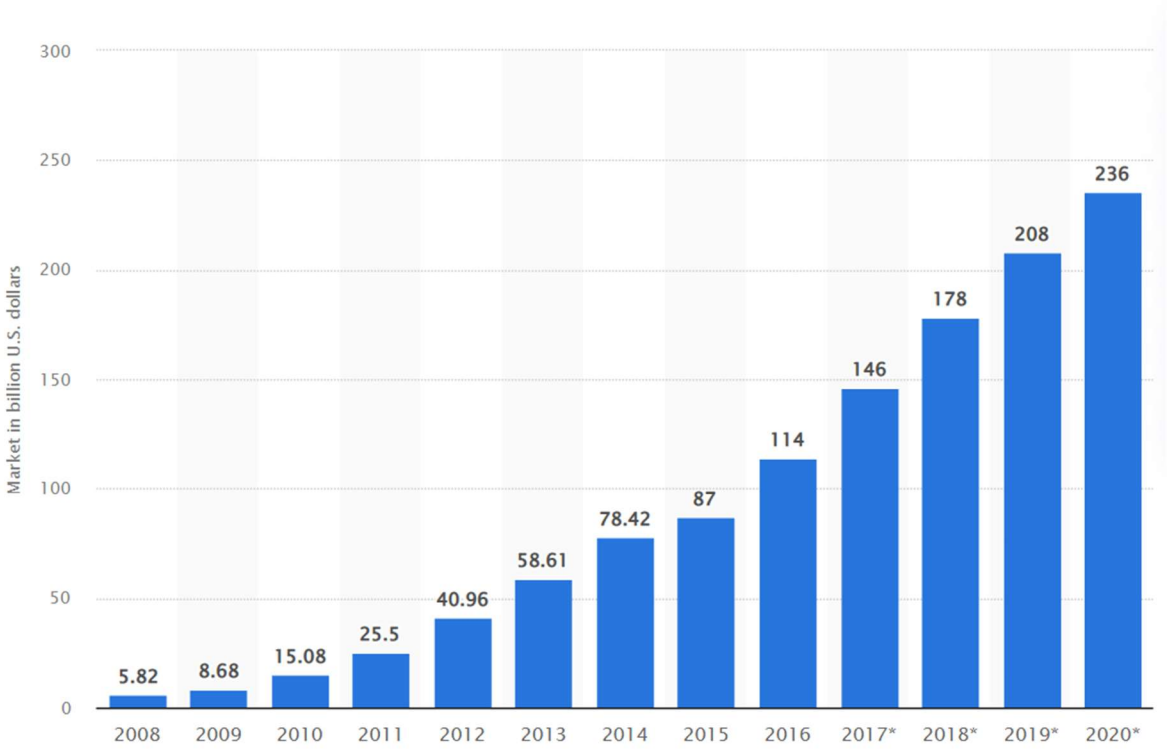
- Redução de custos operacionais e de infraestrutura, que permite às empresas economizarem com hardware, software, energia e manutenção.
- O aumento da segurança e da confiabilidade dos serviços em nuvem, que contam com backups, criptografia e proteção contra-ataques cibernéticos.
- Custos calculados de acordo com o uso do sistema, sendo mais fácil calcular gastos compatíveis com a necessidade da empresa.

A inovação e a transformação digital exigem soluções ágeis, personalizadas e integradas para os desafios do mercado. Além disso a demanda por aplicativos em nuvem, que ofereçam maior flexibilidade, escalabilidade e eficiência para as empresas são constantes.

---

[16] https://datatracker.ietf.org/doc/html/rfc7515
[17] https://datatracker.ietf.org/doc/html/rfc7516

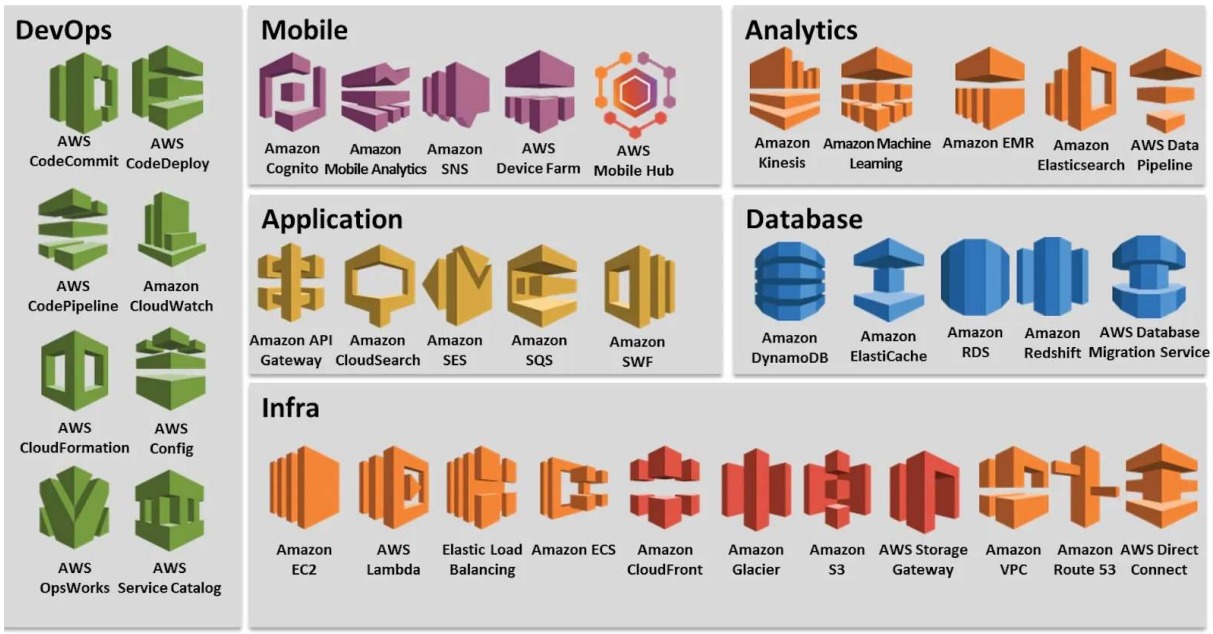
Figura 15 – Tamanho do mercado de computação em nuvem




Fonte: https://www.statista.com/statistics/510350/worldwide-public-cloud-computing/

Há um catálogo imenso de serviços de computação em nuvem, para atender as mais diversas demandas dos clientes. Por exemplo, podemos subir diversos tipos de bancos de dados, *proxies* de rede, balanceadores de carga, cache e serviços de networks, dentre inúmeros outros, como exemplificado na Figura 16. Esses serviços facilitam a manutenibilidade da infraestrutura como um todo. Pois eximem o administrador de infraestrutura de diversas responsabilidades, com preocupação com segurança, rede ou espaço de disco para bancos de dados por exemplo.

Figura 16 - Exemplo dos serviços oferecidos pela Amazon Web Services




Fonte: https://blog.cloudyali.io/aws-services-how-many-are-there-really

### 3.7.1 Fly.io

Fly.io é uma ferramenta de serviço gerenciado e automatizado para realizar a implantação de serviços web de maneira rápida e prática[18]. Com apenas alguns cliques a ferramenta dos dá a capacidade de implantar serviços criados em container diretamente de repositórios no GitHub ou localmente. Esse serviço também oferece infraestrutura para criação bancos de dados, servidores de cache e balanceamento de carga totalmente gerenciados pelo serviço.

Uma das maiores vantagens e motivações para o uso é a possibilidade de realizar a implantação de serviços em container utilizando Docker. Reduzindo ainda mais a complexidade de implantação do serviço e garantindo maior confiabilidade entre os ambientes. O Fly.io permite configurar todas as variáveis de ambiente, além de inúmeras outras configurações de serviços disponíveis na plataforma.

[18] https://fly.io/

Figura 17 – Exemplo de arquivo de configuração Fly.io

```
app = "godevrelbackend"
kill_signal = "SIGINT"
kill_timeout = 5
processes = []

[build]
build-target = "production"

[env]
BASE_CALLBACK_URL = "http://localhost:4000"
DB_HOST = "motty.db.elephantsql.com"
DB_NAME = "ckzqgfsy"
DB_PASS = "jSyAchq6523vokwvRZkyrSqvfAPzI00W"
DB_PORT = 5432
DB_USER = "ckzqgfsy"
MONGO_DB_HOST = "go-devrel-production1.p6fof.mongodb.net"
MONGO_DB_NAME = "forms"
MONGO_DB_PASS = "oTnun0TExfBqcJQx"
MONGO_DB_PORT = 27017
MONGO_DB_USER = "devrelAdmin"
NAME = "go-devrel-back-end"
PORT = 4000
REDIS_HOST = "redis-10013.c262.us-east-1-3.ec2.cloud.redislabs.com"
REDIS_PORT = 10013
REDIS_USER = "default"
REDIS_PASSWORD = "YBdF4FMFf60CJkFMxubwF196Grc1JPBL"
SMTP_HOST = "smtp.gmail.com"
SMTP_PASSWORD = "ozqmorctxwcadfcz"
SMTP_PORT = 587
SMTP_USER = "godevreldevelopment@gmail.com"

[experimental]
```

Front: De autoria própria

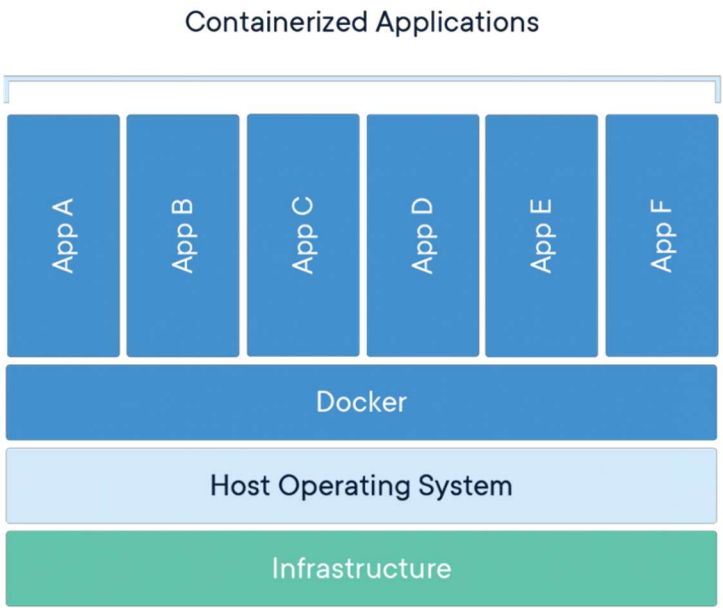
### 3.7.2 Docker

Docker é uma inovação em tecnologia de virtualização e desenvolvimento de software. O Docker foi lançado em 2013 por Solomon Hykes[19], com o objetivo de simplificar a criação e o gerenciamento de aplicativos em contêineres. O conceito de contêineres não é novo, mas o Docker tornou essa tecnologia mais acessível e fácil de usar. Ele é mais leve, mais rápido e mais eficiente do que as máquinas virtuais, permitindo que os desenvolvedores criem, testem e implantem aplicativos em contêineres isolados.

[19] https://pt.wikipedia.org/wiki/Docker_(software)

A principal diferença entre o Docker e as máquinas virtuais é que o Docker usa a virtualização a nível do sistema operacional, enquanto as máquinas virtuais usam a virtualização de hardware. Em outras palavras, as máquinas virtuais emulam um sistema operacional completo, enquanto os contêineres do Docker compartilham o kernel do sistema operacional hospedeiro, como pode ser observado na Figura 18.

Figura 18 – Exemplo de execução dos containers no SO




Fonte: https://www.docker.com/resources/what-container/

Isso significa que o Docker é mais leve, mais rápido e mais eficiente do que as máquinas virtuais, porque não precisa emular todo o hardware e o sistema operacional. Além disso, o Docker permite que os desenvolvedores criem, testem e implantem aplicativos com facilidade, independentemente do ambiente de execução, já que ele empacota as dependências do aplicativo em contêineres isolados.

Atualmente, o Docker é amplamente utilizado na indústria de TI para facilitar a implantação e o gerenciamento de aplicativos em diferentes ambientes. Os contêineres do Docker são portáteis e podem ser executados em qualquer sistema operacional que suporte o Docker, o que torna o desenvolvimento e o teste de aplicativos mais fáceis e rápidos.

Além disso, o Docker é geralmente correlacionado com a computação em nuvem, uma vez que a virtualização em contêineres permite a implantação de

aplicativos em ambientes em nuvem de forma mais eficiente e escalável. A natureza portátil dos contêineres do Docker também facilita a migração de aplicativos entre diferentes ambientes em nuvem.

# 4 PLATAFORMA DE APOIO PARA PROFISSIONAIS E PESQUISADORES, GO DEVREL

Neste capítulo será apresentado o processo de desenvolvimento utilizado para construção da aplicação Go DevRel, passando por todas as etapas do desenvolvimento do projeto. Além disso serão apresentadas as funcionalidades finais do projeto.

## 4.1 Processo de desenvolvimento

O processo de desenvolvimento de um software é complexo, com diversas etapas, artefatos que devem ser gerados e pessoas envolvidas. Entretanto é um processo antigo que passou por várias melhorias e derivou a criação de várias metodologias diferentes. Por exemplo, Kanban, Lean, XP (do inglês, *eXtreme Programming*) e Scrum, entre outros que surgiram ao decorrer de décadas. Para esse trabalho em especial foi adotado a metodologia adaptada, baseada na abordada no Scrum.

Inicialmente no processo de desenvolvimento houve reuniões semanais para realizar definição de alguns dos requisitos iniciais do software. Seguida da criação das tarefas que foram geradas a partir dos requisitos e sua priorização. Em seguida começou o processo do desenvolvimento da aplicação, acompanhada com reuniões quinzenais.

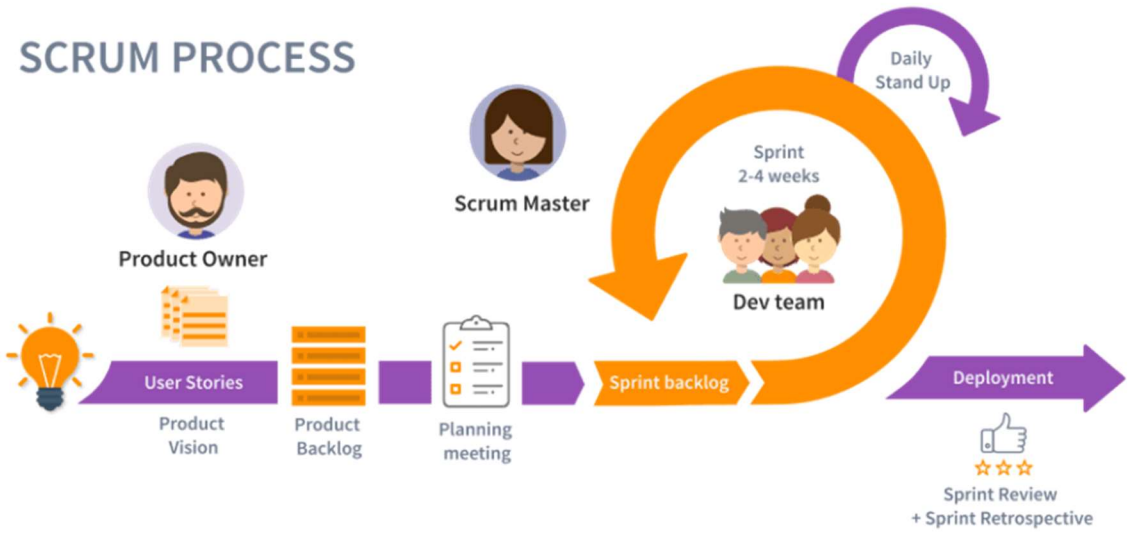
### 4.1.1 Metodologia de desenvolvimento Scrum

O Scrum é um *framework* de desenvolvimento de software que tem como objetivo principal aumentar a eficiência e a produtividade das equipes de desenvolvimento. Ele foi criado no início dos anos 90 por Ken Schwaber e Jeff

Sutherland[20], que buscavam uma forma mais ágil e flexível de gerenciar projetos de software.

O processo de desenvolvimento de software com Scrum se baseia em ciclos curtos e iterativos, chamados de *sprints*. Cada *sprint* tem duração fixa, geralmente de duas a quatro semanas, e tem como objetivo produzir um incremento funcional do software. Durante a *sprint*, a equipe trabalha em conjunto para atingir um objetivo comum e entregar valor ao cliente.

Figura 19 – Infográfico do processo do Scrum




Fonte: https://usemobile.com.br/metodologia-scrum-desenvolvimento/

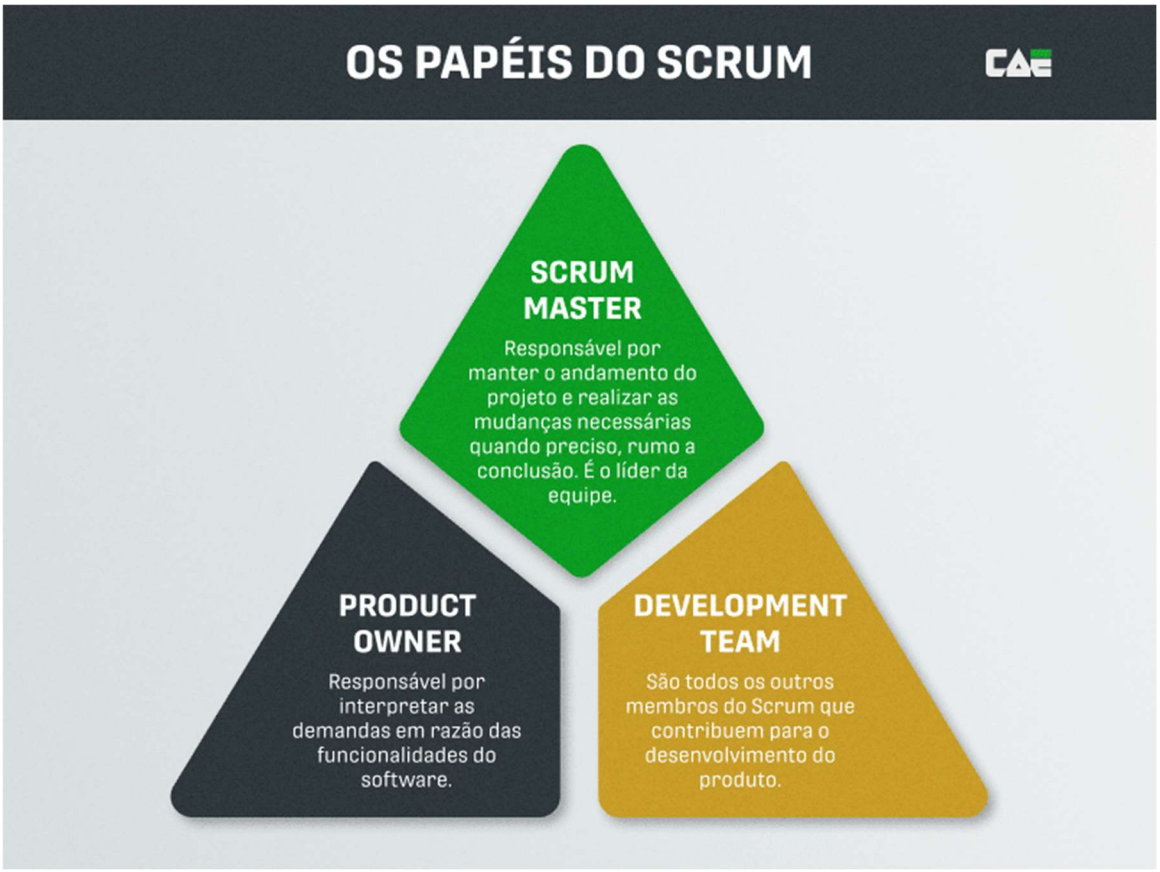
O Scrum utiliza uma série de artefatos, como o *backlog* do produto, que é uma lista de funcionalidades que devem ser desenvolvidas, e o *backlog* da *sprint*, que é uma lista de tarefas que devem ser executadas durante a *sprint*. Além disso, a equipe se reúne diariamente em uma reunião chamada de Daily Scrum, na qual todos compartilham o que fizeram no dia anterior, o que pretendem fazer no dia atual e quais obstáculos estão enfrentando.

O Scrum também utiliza papéis bem definidos, como o Scrum Master, que é responsável por garantir que o *framework* esteja sendo utilizado corretamente e que a equipe esteja seguindo as práticas recomendadas. Por fim, o Product Owner, que é

[20] https://pt.wikipedia.org/wiki/Scrum

responsável por definir o *backlog* do produto e garantir que o software atenda às necessidades do cliente.

Figura 20 – Papeis dentro do Scrum




Fonte: https://caetreinamentos.com.br/blog/gestao-empresarial/scrum-master-e-product-owner-o-que-sao-e-o-que-fazem

### 4.1.2 Coleta e análise de requisitos

A coleta dos requisitos é o processo de coleta, análise, documentação e validação das necessidades e expectativas dos usuários e *stakeholders* em relação ao software a ser desenvolvido. O objetivo desse processo é identificar os requisitos funcionais e não funcionais que o software deve atender para atender às necessidades dos usuários.

Os requisitos coletados são documentados em um documento de especificação de requisitos, que serve como base para o processo de design, implementação, testes e validação da ferramenta.

A coleta dos requisitos foi realizada por meio de reuniões com o cliente. As reuniões foram realizadas como Prof. Dr. Awdren de Lima Fontão, que é referência no campo de *Developer Relations*. A cada interação, era realizado a revisão dos requisitos e das funcionalidades da ferramenta. Os encontros foram realizados por meio da ferramenta Google Meet, e para menores dúvidas utilizava-se o recurso de mensagens instantâneas através de e-mails.

Após as entrevistas foram gerados os requisitos funcionais do sistema, descritos na Tabela 1.

Tabela 1 - Tabela de requisitos funcionais do sistema

| Código - Nome | Descrição |
|---|---|
| **RF001** - Exibir a descrição do modelo | O sistema deve ter uma aba na qual sua finalidade seja a exibição e descrição completa do modelo proposto. |
| **RF002** - Sumário para os tópicos do modelo | A página que exibe o modelo deve apresentar um sumário na lateral que ajude o usuário a navegar pelos tópicos e um que ajude a navegar pela página que está. |
| **RF003** - Mostrar lições aprendidas | Nos tópicos das fases é necessário exibir para o usuário as lições aprendidas para o estágio em questão. |
| **RF004** - Passar para o próximo tópico | Ainda na seção do modelo, no final da página, dever ter algum meio que facilite a passagem para o próximo tópico, um botão, ou até apenas o link. |
| **RF005** - Seção de formulários | Deve haver uma seção específica para o usuário poder passar *feedbacks* sobre o modelo, o sistema deve ajudar o usuário com uma descrição e exemplo ou um link para a wiki explicando o que fazer. |
| **RF006** - Seleção do estágio para o feedback | O usuário deve escolher um estágio do qual deseja selecionar elementos do modelo e enviar o formulário. |
| **RF007** - Mostrar os formulários do estágio | Ao selecionar um estágio para preenchimento, deve ser exibido os formulários disponíveis do estágio selecionado. |
| **RF008** - Status de preenchimento do formulário | Quando um formulário for enviado o usuário deve ter uma resposta visual de quais formulários já foram enviados e preenchidos, sendo destacados os já enviados. |
| **RF009** - Entrar e preencher o formulário | Ao selecionar o formulário o usuário deve ser redirecionado para uma página especifica onde será |

| | |
|---|---|
| | preenchido as questões e feito sua submissão por um botão de enviar e confirmar o preenchimento. |
| **RF010** - Criação de conta do usuário | O usuário pode criar uma conta na plataforma usando e-mail e senha para autenticação, serão pedidos para o cadastro apenas nome completo, data de nascimento, e-mail e senha. |
| **RF011** - Login | Deve haver um botão no menu superior para o usuário realizar o login com e-mail e senha. |
| **RF012** - Exibir confirmação de login do usuário | Quando o usuário terminar o login, o menu superior deve exibir a foto e nome do usuário para mostrar que está autenticado na aplicação. |
| **RF013** - Edição de perfil | O usuário deve poder editar suas informações pessoais, como nome completo, data de nascimento e senha e adicionar um CPF a conta, porém, não editar o e-mail que é usado como chave primária. |
| **RF014** - Edição de senha | Em especial o usuário deve ter uma opção para a edição de senha sendo necessário passar a senha atual para fazer a troca. |
| **RF015** - Sair da conta | O usuário deve poder fazer o logout de sua conta pelo seu dashboard ou menu superior. |
| **RF016** - Página de wiki | É necessário que tenha uma *wiki* com informações das funcionalidades, tutoriais, exemplos, o que fazemos com os dados e políticas de privacidade e alguma introdução sobre o assunto para pessoas leigas no assunto, o sistema de tópicos deve ser igual ao requisito **RF002.** |
| **RF017** - Mais sobre status de preenchimento | Deve-se mostrar um status parcial de preenchimento em cada um nos estágios, para saber que ele já foi parcialmente preenchido. |
| **RF018** - Deleção da conta | O usuário tem por direito deletar todos os dados dele de nossas bases de dados, então criar um botão de deletar conta no dashboard. |
| **RF019** - Dados sobre formulários preenchidos | Exibir no dashboard dados sobre a quantidade de formulários já enviados por estágio e informações adicionais da conta, como data de criação etc. |
| **RF020** - Retorno visual para ações | Ao usuário realizar ações de maior importância retornar um pop-up com a mensagem de status e em caso de preenchimento de formulário com mensagem de agradecimento |

| | |
|---|---|
| **RF021** - Direcionar o usuário para conteúdo | O sistema deve ter uma funcionalidade onde o usuário responde algumas questões e é direcionado para a parte específica do modelo onde mais lhe interessa, para economizar tempo e deixar mais interessante para o usuário |

### 4.1.3 Escolha das ferramentas de apoio

Para o gerenciamento das tarefas referentes ao projeto, foi utilizado a plataforma do Jira. O Jira é uma ferramenta de gerenciamento de projetos e problemas criada pela Atlassian[21]. Ela é amplamente utilizada por equipes de desenvolvimento de software, mas também pode ser usada para gerenciar projetos em diversas outras áreas. Algumas de suas funcionalidades incluem:

- Gerenciamento de tarefas e projetos: o Jira permite criar e acompanhar tarefas, atribuir responsabilidades, definir prazos, acompanhar o progresso e gerenciar projetos em geral.
- Rastreamento de problemas: a ferramenta permite criar tickets para problemas identificados e acompanhar sua resolução. É possível atribuir tarefas a membros da equipe, priorizar os problemas e monitorar o progresso de cada um.
- Colaboração: o Jira permite que várias pessoas trabalhem em um projeto simultaneamente e colaborem em tempo real. Os membros da equipe podem compartilhar informações, discutir ideias e se comunicar por meio de comentários, notificações e outras ferramentas de comunicação integradas.
- Personalização: é possível personalizar o Jira de acordo com as necessidades da equipe. É possível criar campos personalizados, fluxos de trabalho e regras de negócios específicas para cada projeto ou departamento.
- Relatórios e métricas: o Jira oferece uma série de relatórios e métricas para ajudar a equipe a monitorar o progresso e a produtividade. É possível visualizar informações como tempo gasto em cada tarefa, número de problemas resolvidos e outras métricas importantes para o sucesso do projeto.

[21] https://www.atlassian.com/br/software/jira

Em resumo, o Jira é uma ferramenta versátil e poderosa para gerenciamento de projetos e problemas, que ajuda equipes a colaborar, monitorar o progresso e alcançar seus objetivos.

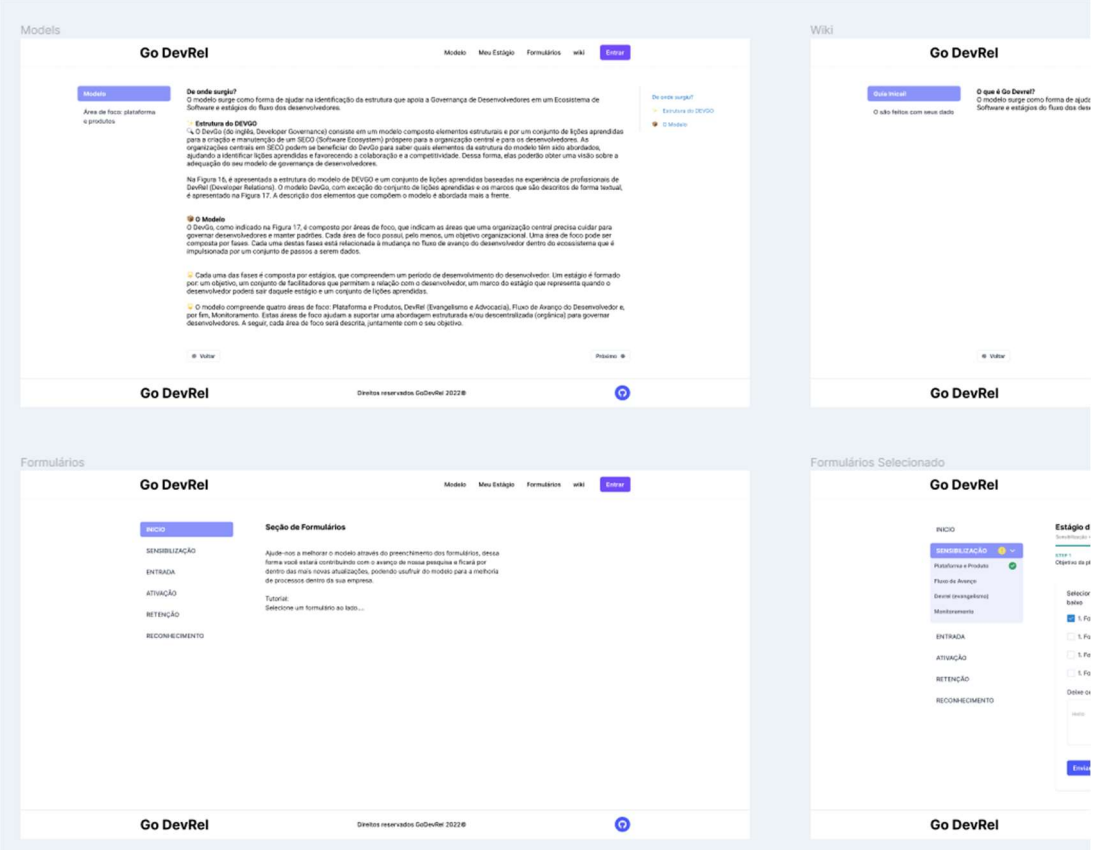
Além do Jira, foram usadas as ferramentas Confluence, também do ecossistema da Atlassian, para realizar a documentação dos artefatos do software gerados no desenvolvimento. Por exemplo, documento de requisitos, análises de outras funcionalidades, resumos das reuniões e outras anotações sobre o software. O Confluence é uma plataforma de colaboração e gestão de conhecimento, que permite que equipes trabalhem juntas para criar, organizar e compartilhar informações em um ambiente centralizado[22]. Algumas das suas principais funcionalidades incluem:

- Criação e compartilhamento de conteúdo: o Confluence permite que equipes criem e compartilhem conteúdo de forma colaborativa, como documentos, especificações de projeto, manuais, relatórios e outros tipos de informações.
- Organização de informações: a plataforma permite que as equipes organizem suas informações em páginas, pastas e espaços temáticos, facilitando a localização e o acesso aos dados.

Por fim, foi utilizado também a ferramenta Figma [23]para prototipagem inicial das telas, geradas a partir da coleta de requisitos nas fases iniciais do projeto, como mostrado a seguir na Figura 21. Foram feitos vários protótipos de alta fidelidade com a tela final.

[22] https://www.atlassian.com/br/software/confluence/
[23] https://www.figma.com/

Figura 21 – Protótipos feitos no Figma



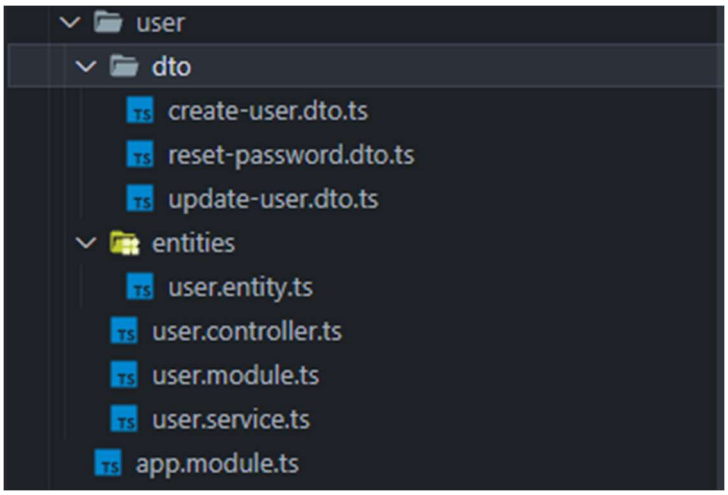

Fonte: De autoria própria

### 4.1.4 Arquitetura da aplicação

A arquitetura de software é a forma com que seu sistema será organizado a nível de código e componentes. Por exemplo como serão organizados classes, métodos, interfaces e módulos do seu sistema. Existem várias camadas de abstração que uma arquitetura de software pode trazer para ajudar na manutenibilidade do sistema ou para evitar bugs e facilitar alterações no sistema. Tendo isso em vista podemos afirmar que: "Arquitetura é o conjunto de decisões que você queria ter tomado logo no início do um projeto” (Martin, 2017).

Por esses motivos foi escolhido o *framework* NestJs para o projeto, por já ter sido previamente muito bem arquitetada e trazer um padrão de arquitetura muito conciso. Na arquitetura do NestJs pode ser observada várias similaridades com padrões apresentados no livro *Clean Architecture,* como os principios SOLID, camadas de separação do código e desacoplamento dos componentes, como apresentado na Figura 22.

Figura 22 – Exemplo da arquitetura do código do backend




Fonte: De autoria própria

Como pode-se observar na Figura 22 *user* representa um modulo na arquitetura proposta pelo NestJs, que ajuda primeiro na separação de responsabilidade entre módulos do sistema. Isso consequentemente traz uma melhor manutenibilidade para a aplicação como um todo. Além disso cara arquivo contém um único uma única responsabilidade, o que também auxilia na manutenção do código e é um dos principais princípios do SOLID, o Princípio da responsabilidade única (do inglês, *Open Closed Principle*).

Ainda podemos observar na raiz da pasta *user* na Figura 22 um arquivo com a extensão “.module”, esse arquivo fica responsável pelo gerenciamento de dependências do módulo (Figura 23). Essa abordagem vai de encontro com outro princípio do *Clean Architecture*, a injeção de depencencia, que prega que todo componente externo que uma classe fizer uso deve ser injetado através do construtor da classe por exemplo.

Figura 23 – Gerenciamento de modulo do NestJs


Fonte: De autoria própria

Em resumo, esses principios sempre tem o mesmo intuito, facilitar a manutenção do código, deixando ele mais legivel e abaixando a complexidade cognitava geral, além de evitar possiveis erros. Isso facilita muito a implementação do código e ajuda na implementação de futuras *features*.

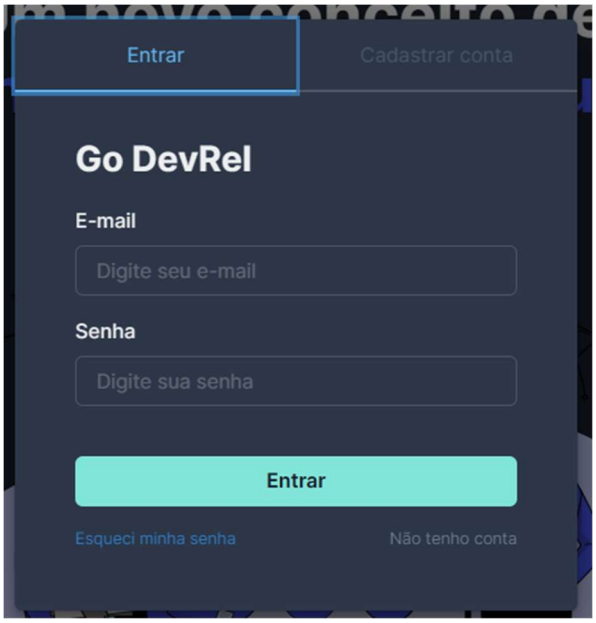
## 4.2 Demonstração das funcionalidades implementadas

A ferramenta foi criada com o intuito de ajudar qualquer profissional de DevRel ou desenvolvedor, por isso diversas funcionalidades presentes estão disponíveis a qualquer tipo de público. Por outro lado, a ferramenta também foi feita para a coleta de *feedbacks* da comunidade sobre o modelo. Por conta disso, a ferramenta conta com uma página administrativa também.

### 4.2.1 Visão do usuário final

Dentre as funcionalidades disponíveis ao público geral se encontra o login (Figura 24) e o formulário de cadastro (Figura 25) que podem ser acessados a partir do menu superior da página. O cadastro de usuário é necessário para que possa ser exibido o menu para responder os formulários. Sem o login, o usuário não pode fazer nenhuma requisição para os formulários existentes.

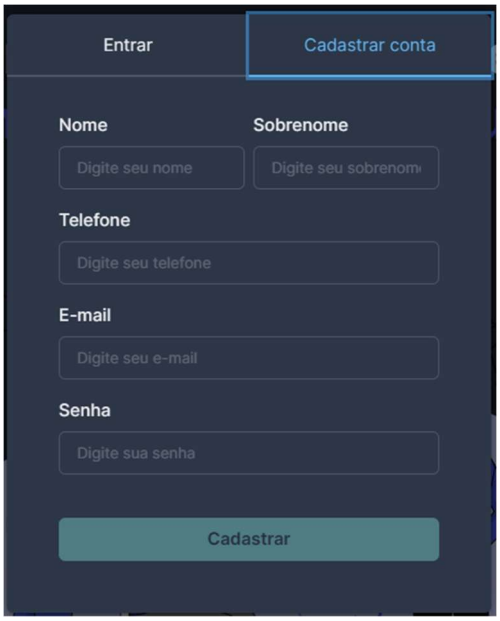
Figura 24 – Formulário de login




Fonte: De autoria própria

O formulário de cadastro contém campos de nome, telefone, e-mail e senha, esses campos contam com algumas validações, como obrigatoriedade de e-mail, telefone e senha, dentre outras validações. A senha possui um requisito de no mínimo 8 caracteres e no e-mail é obrigatório o uso de um caractere arroba seguido do endereço.

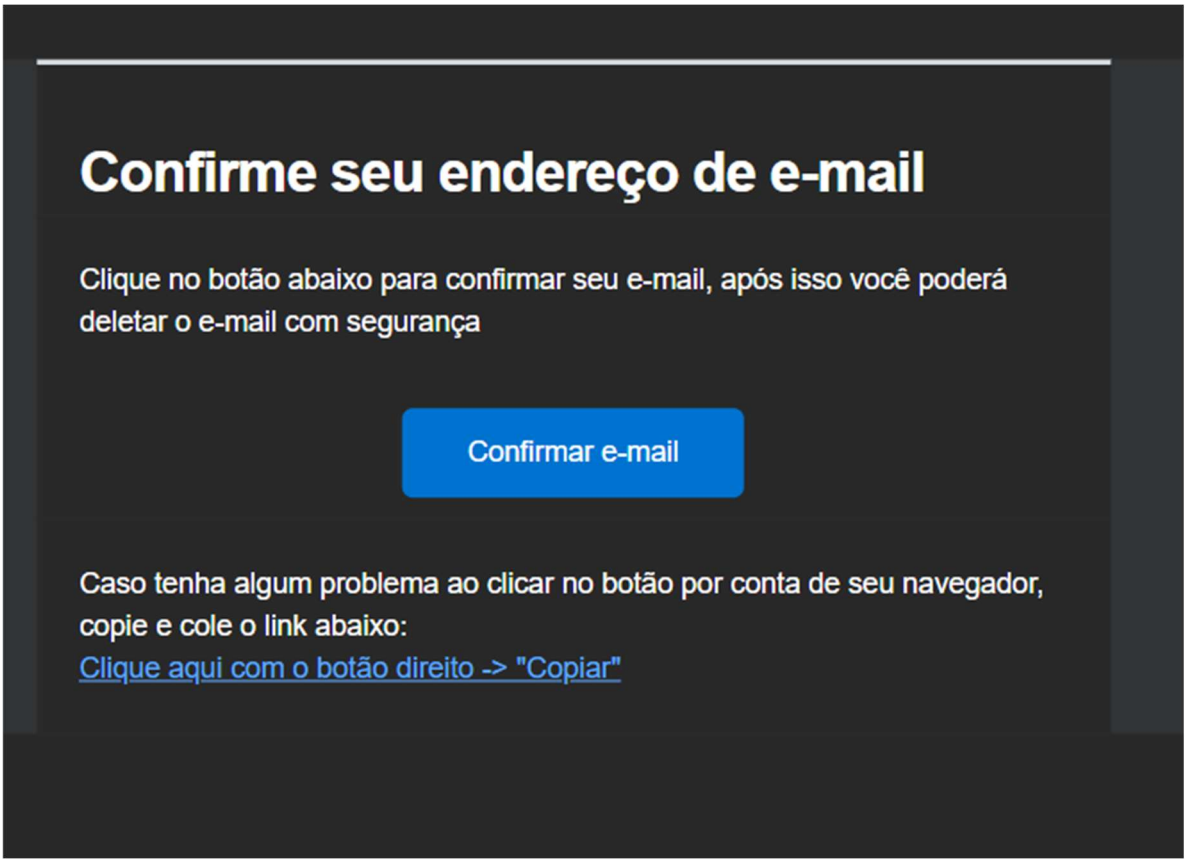
Figura 25 – Formulário de cadastro de nova conta




Fonte: De autoria própria

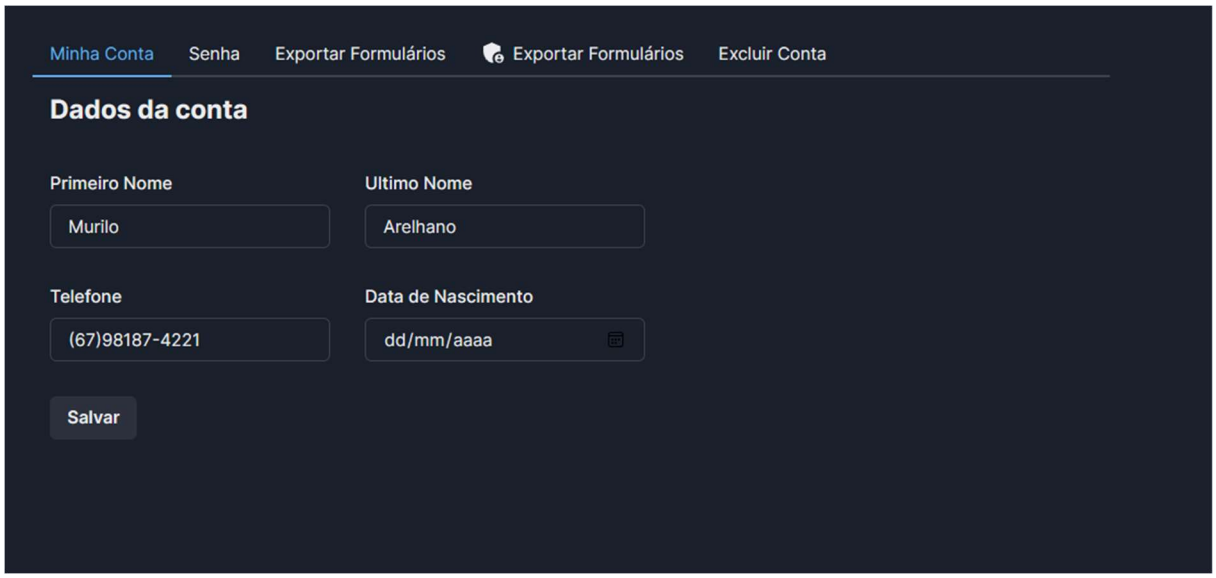
Após ter a conta criada, o usuário receberá um e-mail para a validação da conta (Figura 26), para fins de implementações futuras. O usuário terá logo em seguida direito a acessar um painel administrativo para gerenciar dados básicos da conta, como nome, idade (Figura 27) e mudar a senha (Figura 28). Por alguns motivos de conversão não é possível mudar o e-mail do usuário, estão disponíveis para a edição apenas telefone, nome, e data de nascimento.

Figura 26 – Exemplo de e-mail de validação da conta




Fonte: De autoria própria

Figura 27 – Página de edição dos dados da conta do usuário



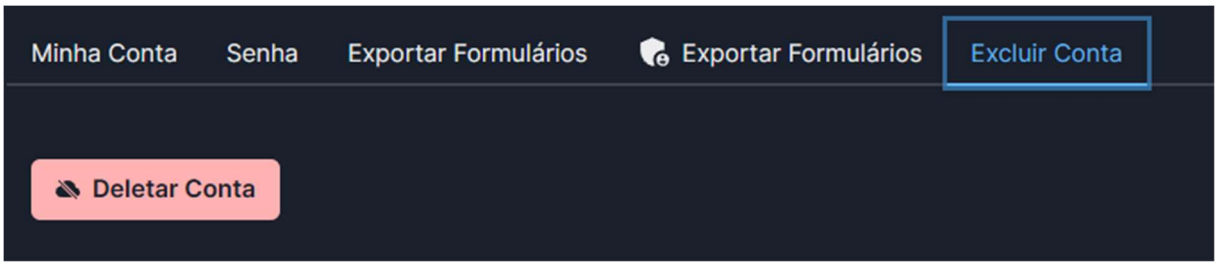

Fonte: De autoria própria

Figura 28 – Página de mudar senha do usuário


Fonte: De autoria própria

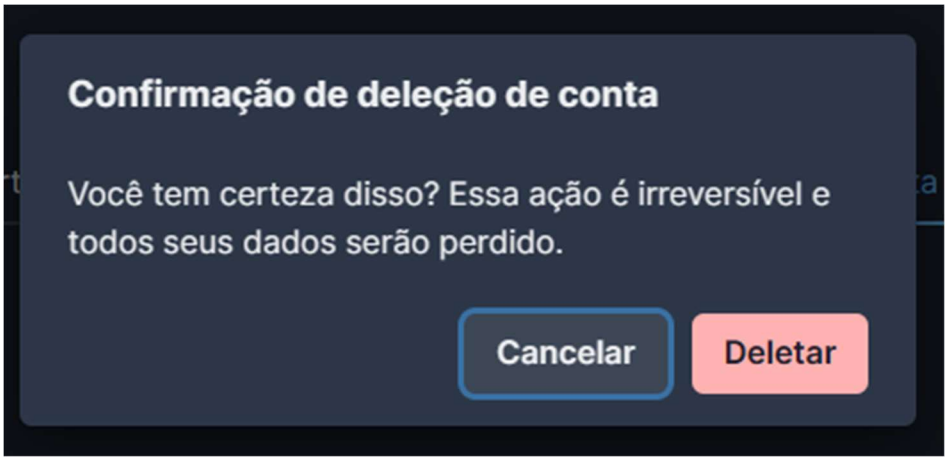
Ainda sobre a conta do usuário, é possível que o usuário queira encerrar a conta e desvincular seus dados da ferramenta. Para isso foi criado uma abada no gerenciamento de conta do usuário, que possibilita a exclusão da conta e todos os dados associados do usuário (Figura 29). Ao clicar no botão de deletar a conta, será exibido uma janela de confirmação da ação (Figura 30), tendo em vista que essa ação é irreversível dentro do sistema.

Figura 29 – Página de exclusão da conta do usuário




Fonte: De autoria própria

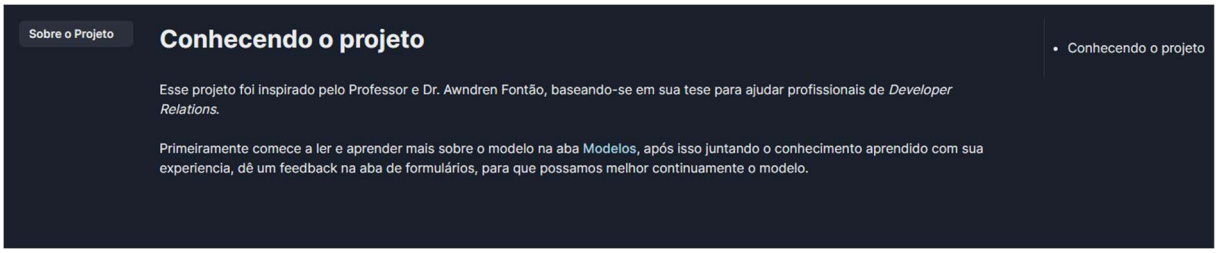
Figura 30 – Modal de confirmação de exclusão de conta




Fonte: De autoria própria

Foi adicionado ao projeto também uma página chamada “Wiki”, com o intuito de explicar mais sobre o projeto, sua história, propósito e o que será feito com os dados inseridos na ferramenta posteriormente. Essa página foi criada para dar mais credibilidade a ferramenta e deixar que os usuários tenham mais confiança em contribuir com os formulários.

Figura 31 – Página “Wiki” da ferramenta



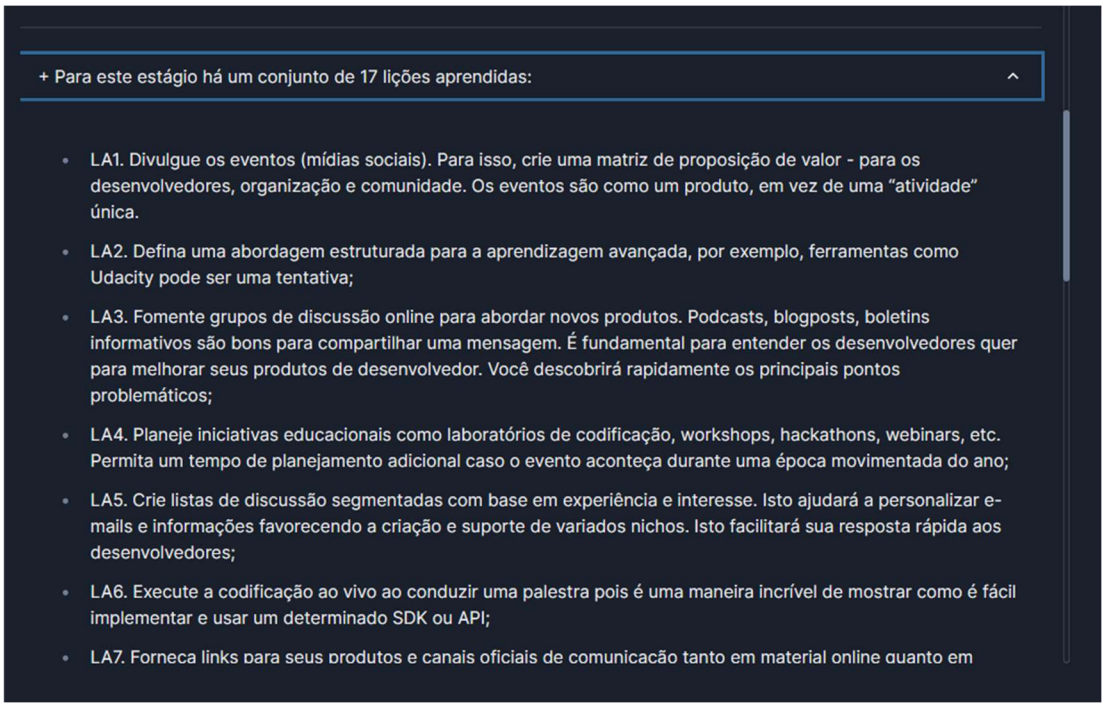

Fonte: De autoria própria

Uma das principais funcionalidades da ferramenta é apresentar o modelo Go DevRel para compartilhar o conhecimento com a comunidade de desenvolvedores e profissionais de DevRel. A página de apresentação do modelo (Figura 32) conta com um componente que gera o menu lateral esquerdo com diferentes páginas e um sumário da página atual. Além disso, em cada uma das páginas referentes as fases, se encontram as respectivas lições aprendidas (Figura 33).

Figura 32 – Página de apresentação do modelo


Fonte: De autoria própria

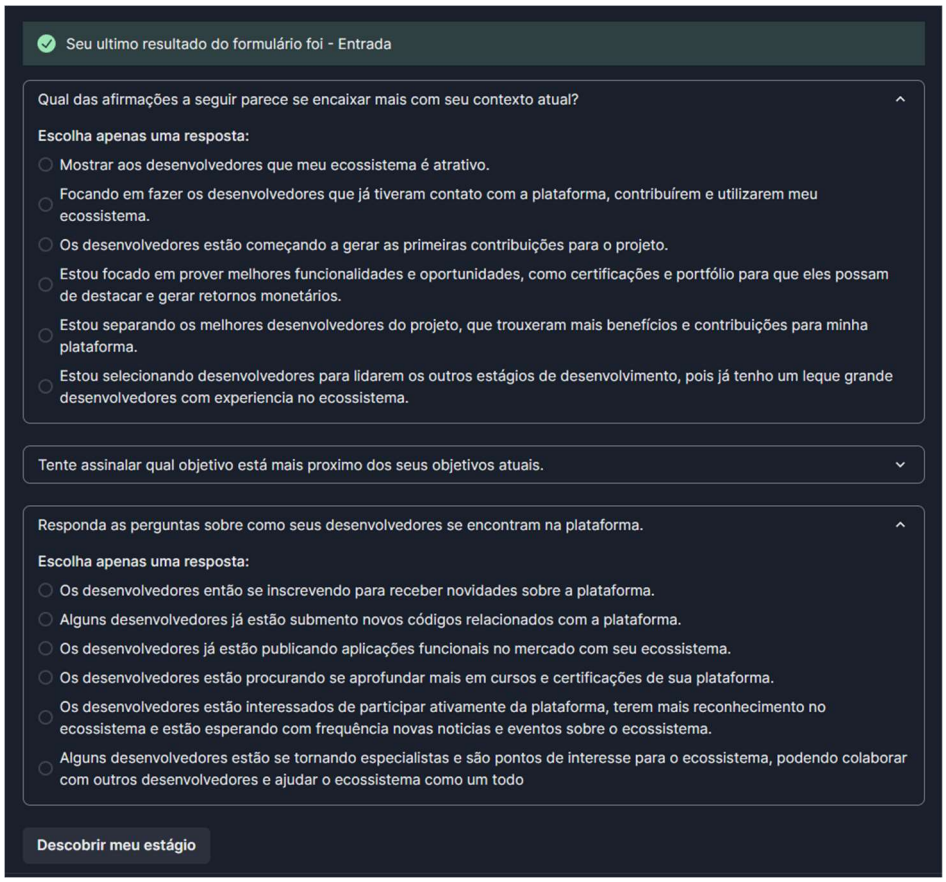
Figura 33 – Exemplo de lições aprendidas da página do modelo




Fonte: De autoria própria

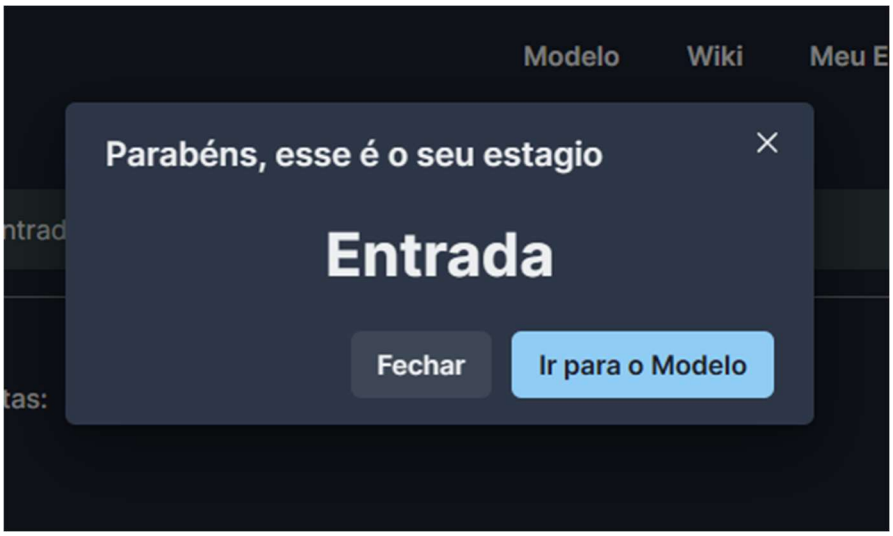
Após os usuários conhecerem um pouco sobre o modelo, eles podem acessar a página “Meu Estágio” (Figura 34). Essa página foi criada para ajudar os usuários a encontrarem mais rapidamente o estágio em que se encontram. A página contém diversas perguntas que contabilizam uma pontuação para cada um dos estágios, entretanto esse detalhe não fica visível ao usuário. Ao responder todas as perguntas, será exibido para o usuário o resultado do estágio está com maior somatória de pontos (Figura 35). Além dessa funcionalidade, a página também exibe o resultado da última resposta que o usuário deu ao formulário.

Figura 34 – Formulário da página “Meu Estágio”




Fonte: De autoria própria

Figura 35 – Modal de exibição do resultado do estágio




Fonte: De autoria própria

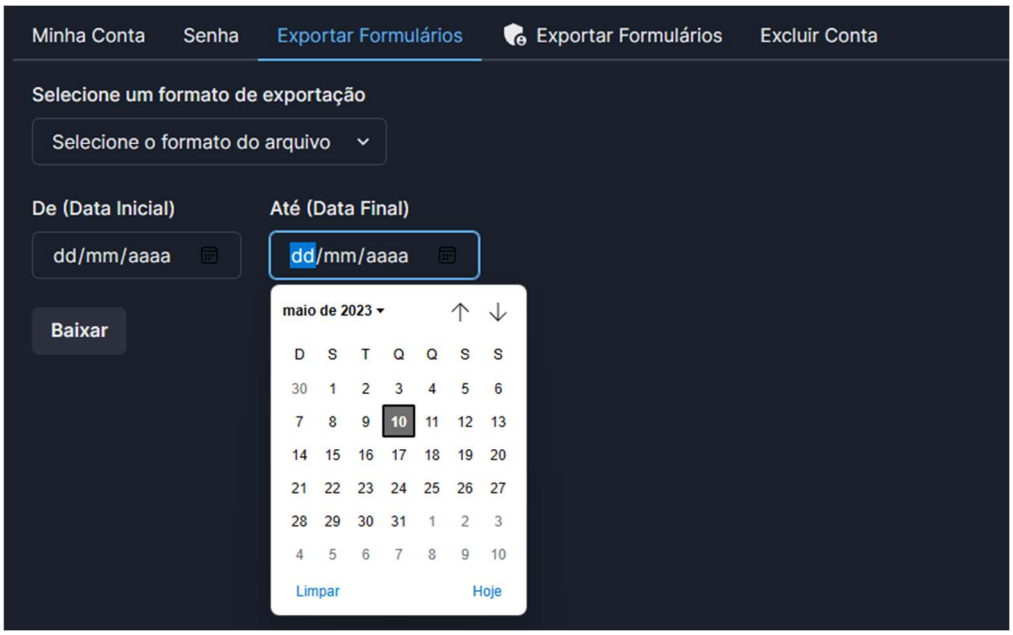
Outra funcionalidade e possivelmente a mais importante para a ferramenta é a página de formulários, que possibilita o usuário a responder formulários de pesquisa sobre o modelo (Figura 36). Essa página só pode ser acessada por usuários logados na plataforma, já que o formulário é associado ao usuário que o respondeu. Posteriormente ao envio do formulário, o usuário pode exportar os formulários para consultá-los (Figura 37).

Figura 36 - Formulários de pesquisa da página “Formulários”


Fonte: De autoria própria

Além disso, o usuário pode escolher o formato no qual será exportado o formulário dentre as opções JSON e CSV. Para exportar os formulários é necessário selecionar a data inicial e final dos formulários que serão exportados.

Figura 37 – Página de exportação de formulários privada

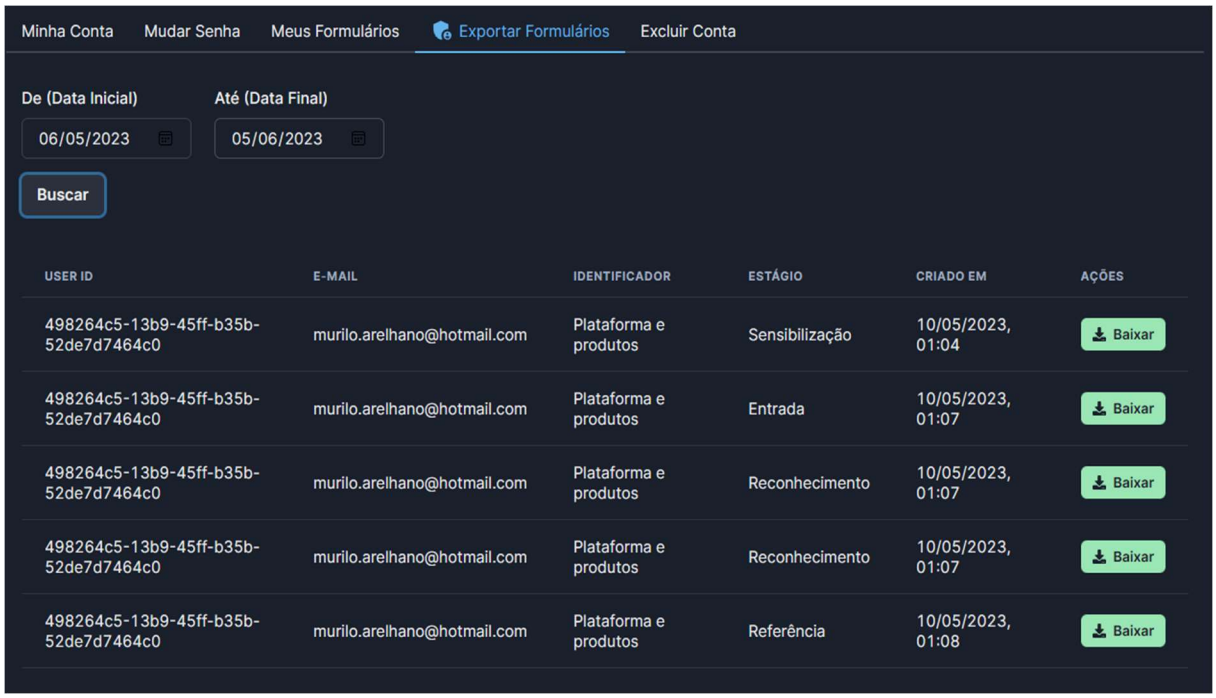



Fonte: De autoria própria

### 4.2.2 Visão do usuário administrador

Outro foco da aplicação é o administrador do sistema ter acesso aos formulários, para poder analisar as respostas enviadas pelo usuário. Para isso foi construído um sistema de papéis na aplicação que regra que apenas usuários com papel de administrador podem acessar uma tela de download dos formulários enviados, como mostrado na Figura 38. É necessário que o administrador selecione um intervalo de data, do qual será baixado os formulários.

Figura 38 – Página do administrador de download de formulários




Fonte: De autoria própria

# 5 CONCLUSÃO

Esse capítulo apresenta as considerações finais acerca desse trabalho, além de se sugestões sobre funcionalidade para continuidade e refinamento da aplicação Go DevRel.

## 5.1 Considerações finais

Esse trabalho teve como objetivo apresentar as funcionalidades da aplicação web Go DevRel, ferramenta que ajudará os profissionais de DevRel. Compartilhando conhecimento, lições aprendidas e coletando novos *feedbacks* para melhoria contínua da do modelo. Também foram apresentados detalhes de implementação e implantação da aplicação, além das decisões tomadas ao decorrer do projeto.

Tendo em vista que os benefícios que o profissional de *Developer Relations* traz para as empresas são inúmeros. Incluindo a construção de uma base de usuários leais e engajados, a identificação de novas oportunidades de negócios, a melhoria da qualidade do produto e a ampliação da visibilidade da marca. É de extrema importância melhorar esse processo e consequentemente melhorar muito o

desempenho das comunidades de desenvolvedores e desses profissionais pelo mundo.

As escolhas das ferramentas usadas durante o trabalho estão diretamente ligadas com tendencias do mercado, por exemplo, a grande adoção do Node.js pela comunidade de desenvolvedores. A definição de uma arquitetura sólida e padronizada também ajudara novos desenvolvedores a compreender mais rapidamente o projeto e dar continuidade. Além da facilidade de implantação e padronização de uma infraestrutura baseada em container, vai ajudar também em uma implantação mais simplificada do sistema.

## 5.2 Trabalhos Futuros

Todo o projeto de software passa por processos de melhoria contínua em sua estrutura, melhorias corretivas de problemas e melhorias adaptativas, por exemplo. Entretanto para este projeto ainda existem algumas funcionalidades que não foram implementadas a tempo. Dentre elas estão:

- Adicionar retornos visuais e confirmação para envio dos formulários. Quando o usuário finalizar o envio de um formulário é interessante que esse usuário saiba quais formulários já foram enviados naquele dia.
- Exibir no dashboard dados sobre a quantidade de formulários já enviados por estágio e informações adicionais da conta, como data de criação e outras métricas relevantes sobre os formulários.
- Exibir métricas das respostas da comunidade de forma pública, para usuários poderem ter parâmetros para comparar suas respostas ou até analisar as necessidades mais relatadas por outros usuários, sem comprometer a identidade dos usuários.
- Na tela inicial, salvar os e-mails dos usuários que se cadastrarem para receber notificações sobre atualizações no site ou no modelo.
- Implementar diferentes tipos de login em por outras plataformas, como Google, Microsoft e GitHub, usando o padrão *OpenId Connect*.
- Implementar um questionário dinâmico com uma estrutura que armazene as questões em um banco de dados e que possam ser mudadas.

- Refinar e melhorar os formulários a fim de coletar feedbacks mais refinados que sejam mais úteis ao decorrer do desenvolvimento do modelo.
- Permitir que apenas usuários com e-mail validado enviem formulários.

**REFERÊNCIAS**

Fontão, A., Cleger-Tamayo, S., Wiese, I. & Pereira dos Santos, R., 2021. A Developer Relations (DevRel) Model to Govern Developers in Software Ecosystems. Em: *Journal of Software: Evolution and Process.* s.l.:s.n., p. 35.

Fontão, A. d. L., 2019. *DevGo: um modelo para governança de desenvolvedores em ecossistema de software móvel a partir de developer relations.* Manaus: s.n.

Gill, P. S., 2022. *"What is Developer Relations?".* [Online] Available at: https://www.commudle.com/blogs/what-is-developer-relations

Martin, R. C., 2017. *Clean Architecture: A Craftsman's Guide to Software Structure and Design.* s.l.:Pearson Education. Inc.

Oliveira, R. et al., 2021. Developer relations (devrel) roles: An exploratory study on practitioners' opinions. Em: *Proceedings of the XXXV Brazilian Symposium on Software Engineering.* s.l.:s.n., p. 363–367.

Santos, R. & Viana, D., 2016. Ecossistemas de Software no Desenvolvimento de Plataformas para Web, Redes Sociais e Multimídia. Em: *Anais do XXII Simpósio Brasileiro de Sistemas Multimídia e Web (Vol. 3): Minicursos.* s.l.:s.n.

Thengvall, M., 2018. *The Business Value of Developer Relations.* San Francisco, California, USA: Apress.