CENTRO ESTADUAL DE EDUCAÇÃO TECNOLOGICA PAULA SOUZA

FACULDADE DE TECNLOGIA DE INDAIATUBA

DR. ARCHIMEDES LAMMOGLIA

CURSO DE TECNOLOGIA EM

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

CARLOS EDUARDO FIORI DOS SANTOS

Desenvolvimento de uma aplicação mobile para o sistema de gerenciamento acadêmico da Fatec utilizando web scraping

Indaiatuba
Novembro 2024

CENTRO ESTADUAL DE EDUCAÇÃO TECNOLOGICA PAULA SOUZA

FACULDADE DE TECNOLOGIA DE INDAIATUBA

DR. ARCHIMEDES LAMMOGLIA

CURSO DE TECNOLOGIA EM

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

CARLOS EDUARDO FIORI DOS SANTOS

Desenvolvimento de uma aplicação mobile para o sistema de gerenciamento acadêmico da Fatec utilizando web scraping

Trabalho de Graduação apresentado por Carlos Eduardo Fiori dos Santos como pré-requisito para a conclusão do Curso Superior de Tecnologia em Gestão Empresarial, da Faculdade de Tecnologia de Indaiatuba, elaborado sob a orientação do Prof. Felipe do Espirito Santo.

Indaiatuba
Novembro 2024

CENTRO ESTADUAL DE EDUCAÇÃO TECNOLOGICA PAULA SOUZA

FACULDADE DE TECNOLOGIA DE INDAIATUBA

DR. ARCHIMEDES LAMMOGLIA

TECNOLOGIA EM

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

CARLOS EDUARDO FIORI DOS SANTOS

Banca Avaliadora:

Felipe do Espirito Santo	Orientador
João Manoel de Campos	Área
Magali Berçante	Convidado

Data da defesa: 03/12/2024

RESUMO

Em meio a crescente evolução tecnologia, observa-se o aumento significativo do uso de smartphones em todos os setores da sociedade. Devido a essa intensificação, houve uma grande expansão no setor de aplicativos móveis que trazem diversas funcionalidades para todos os tipos de usuário, as mais variadas instituições buscam criar essas aplicações para seus usuários. Pode-se destacar universidades que trazem essa tecnologia para suporte à gestão acadêmica, facilitando o uso do sistema ao prover funções e informações que podem ser acessadas com maior agilidade, facilidade e usabilidade. Contudo, não são todas essas instituições que podem fornecer essa tecnologia, como é o caso da Fatec, tendo em vista os diversos problemas relacionados a fato de ser uma instituição pública. Face ao contexto, o atual trabalho tem como objetivo disponibilizar as informações institucionais dos alunos da Fatec de forma simples e eficaz para os dispositivos móveis. Entretanto há alguns obstáculos relacionados ao acesso à base de dados da instituição, que não fornece uma forma de acesso oficial. Portanto, o objetivo desse trabalho é o desenvolvimento de um aplicativo móvel para Android e um *web scraping* do sistema de gerenciamento acadêmico da Fatec, o SIGA, que será acessado por meio de uma *API* configurada no ambiente cloud e irá retornar informações do aluno. Tendo o aplicativo operante e consumindo os dados provenientes do *web scraping*, poderá ser de grande utilidade para os alunos da instituição, uma vez que poderá acessar suas informações acadêmicas de forma ágil, garantindo uma visão positiva para a imagem da entidade por contribuir para sua diversidade tecnológica.

Palavras-chave: Aplicativo; Mobile; *web scraping; AWS*; SIGA; *Fatec*.

ABSTRACT

Amid the growing technological evolution, there is a significant increase in the use of smartphones in all sectors of society. Due to this intensification, there has been a great expansion in the mobile application sector, bringing various functionalities to all types of users, with the most diverse institutions seeking to create these applications for their users. Universities can be highlighted as they bring this technology to support academic management, facilitating the use of the system by providing functions and information that can be accessed with greater agility, ease, and usability. However, not all institutions can provide this technology, as is the case with Fatec, due to various problems related to being a public institution. In this context, the current work aims to provide Fatec students' institutional information simply and effectively for mobile devices. However, there are some obstacles related to accessing the institution's database, which does not provide an official access method. Therefore, the objective of this work is to develop a mobile application and web scraping for Fatec's academic management system, SIGA, which will be accessed through an API configured in the cloud environment and will return student information. Having the application operational and consuming data from the web scraping can be of great utility for the institution's students, as it will allow them to access their academic information quickly, ensuring a positive view of the entity's image by contributing to its technological diversity.

Keywords: Application; Mobile; web scraping; AWS; SIGA; Fatec.

LISTA DE TABELAS

Tabela 1 – Tecnologias nativas de cada plataforma. 16
Tabela 2 – Ferramentas e tecnologias utilizadas por cada projeto. 26
Tabela 3 – Funcionalidades disponíveis em aplicativos do mercado. 28

LISTA DE FIGURAS

Figura 1: Renderização de um componente em React e React Native 17
Figura 2 - Demonstração de *workflow*.....21
Figura 3: Fluxograma do acesso ao SIGA.....30
Figura 4: Diagrama de caso de uso representando o processo de autenticação do usuário no sistema 31
Figura 5: Diagrama de caso de uso representando o processo de visualização de dados 32
Figura 6: Diagrama de caso de uso representando o processo de visualização de dados offline.....33
Figura 7: Diagrama da arquitetura do projeto 35
Figura 8: Tela de login do aplicativo 37
Figura 9: Tela de login com mensagem de erro, "Preencha todos os campos" 38
Figura 10: Tela de login com notificação informando que o login é inválido..39
Figura 11: Tela de login com notificação informando que as credenciais estão sendo verificadas 40
Figura 12: Tela de login com notificação informando que o dispositivo está sem conexão com a internet.....41
Figura 13: Tela de login com mensagem informando que o login e senha não conferem 42
Figura 14: Tela de login com notificação informando problema com o servidor 43

Figura 15: Tela de login pedindo autenticação nativa para acessar usuário já logado 44
Figura 16: Tela de login com notificação e sucesso após autenticação nativa 45
Figura 17: Tela de login com notificação informando que a autenticação nativa foi cancelada 46
Figura 18: Tela de carregamento padrão.....47
Figura 19: Tela padrão notificando erro ao carregar os dados 48
Figura 20: Tela inicial do aplicativo, parte 1.....49
Figura 21: Tela inicial, parte 2.....49
Figura 22: Tela incial, parte 3 50
Figura 23: Tela inicial com acordeão e horários aberto, parte 3 51
Figura 24: Tela inicial com acordeão de notas parciais expandida, parte 3...51
Figura 25: Tela inicial com acordeão de históricos expandido, parte 3 52
Figura 26: Tela inicial com acordeão de perfil expandido, parte 3.....52
Figura 27: Tela inicial, parte 4.....53
Figura 28: Tela inicial, parte 5.....53
Figura 29: Tela de horário com acordeão fechado 54
Figura 30: Tela de horários com acordeão expandido.....55
Figura 31: Tela de horário sem conexão com a internet.....56
Figura 32: Tela de notas parciais, parte 1.....57
Figura 33: Tela de notas parciais sem conexão com a internet.....58
Figura 34: Tela de histórico com acordeões fechados 59
Figura 35: Tela de histórico com acordeão expandido 59
Figura 36: Tela de histórico sem conexão com a internet.....60
Figura 37: Tela de perfil.....61
Figura 38: Tela de login notificando a saída da conta 62
Figura 39: Tela da carteirinha do aluno 63
Figura 40: Tela de recuperação de senha 64
Figura 41: Organização criada para o projeto no GitHub.....65
Figura 42: Padronização de branches dentro do repositório do aplicativo.....66
Figura 43: Estrutura de arquivos do back-end 68
Figura 44: Visualização do painel de roles dentro da AWS 69
Figura 45: Diretório do back-end com a pasta terraform expandido 70

Figura 46: Arquivo variables.tf 71
Figura 47: Arquivo terraform.tfvars da pasta de dev 71
Figura 48: Arquivo terraform.tfvars da pasta de qual 72
Figura 49: Arquivo terrafrom.tfvars da pasta de prod 72
Figura 50: Arquivo provider.tf 72
Figura 51: Arquivo main.tf 73
Figura 52: Arquivo backend.tf 74
Figura 53: Arquivo lambda-data-user.tf 75
Figura 54: Painel de log de execução do lambda na AWS 76
Figura 55: Arquivo api-gateway.tf - parte 1 77
Figura 56: Arquivo api-gateway.tf 78
Figura 57: Diretório do back-end com a pasta lambdas e get-data-user expandidas 79
Figura 58: Diretório do back-end com as pastas .github e workflows expandida 80
Figura 59: Diagrama do processo da execução de CI/CD no repositório do back-end 81
Figura 60: Diretório do projeto do aplicativo do SIGA 82
Figura 61: Diagrama do processo da execução de CI/CD no repositório do aplicativo (front-end) 83

LISTA DE ABREVIATURAS E SIGLAS

API - Application Programming Interface
AWS – Amazon Web Service
CAPTCHA - Completely Automated Public Turing test to tell Computers and Humans Apart
CI/CD – Continuous Integration and Continuous Deployment
Fatec – Faculdade de Tecnolgia
FGV – Fundação Getulio Vargas
HTML - Hypertext Markup Language
HTTP – Hypertext Transfer Protocol
IaC – Infrastructure as code
IaaS – Infrastructure as a service
JSON – JavaScript Object Notation
LGPD – Lei Geral de Proteção de Dados
NoSQL – Not Only Structured Query Language
REST – Representational State Transfer
S3 – Simples Storage Service
SDK – Software Development Kit
SIGA – Sistema de Gerenciamento Acadêmico
SOAP – Simple Object Access Protocol
SQL – Structured Query Language
XML – Extensible Markup Language

SUMÁRIO

1. INTRODUÇÃO 12
2. FUNDAMENTAÇÃO TEÓRICA 14
2.1 Sistema operacional e aplicativos móveis 14
2.1.1 iOS 15
2.1.2 Android 15
2.2 Tecnologia de desenvolvimento de aplicativos móveis 15
2.2.1 Nativo 16
2.2.2 *WebApp* 16
2.2.3 Híbrido 17
2.3 *Web Scraping* 18
2.4 *API – Application Programming Interface* 19
2.5 Versionamento de software 19
2.6 *CI/CD – Continuous Integration and Continuous Deployment* 20
2.7 *IaC – Infrastructure as Code* 21
2.8 *IaaS – Infrastructure as a Service* 22
2.9 Trabalhos acadêmicos relacionados 23
2.9.1 Aplicativo para apresentação de boletim escolar com extração de dados do SUAP através do *Web Scraping*: estudo de caso no IFSP (SILVA, 2019) 23
2.9.2 PUC-Rio Mobile – Aplicativo de gerenciamento estudantil para os alunos da PUC-Rio (AGUIAR, 2022) 24
2.9.3 SIGAA Mobile – O caso de sucesso da ferramenta de gestão acadêmica na era da computação móvel (FILHO; AQUINO; ROSA, 2013) 24
2.9.4 Interface mobile para a plataforma SIGAA (MASCARENHAS, 2022) 24
2.9.5 Jambo: coleta de dados com web scraping (TELES; SILVA, 2021) 25
2.10 Comparação entre referências acadêmicas 25
2.11 Aplicativos de mercado relacionados 26
2.11.1 SIGAUFMG (2024) 26
2.11.2 SIGAA Mobile (2024) 27

2.11.3 ACADESC (2024) – Gestão Escolar 27
2.11.4 UniCampus (2024) 27
2.11.5 Portal Aluno UFRJ (2024) 27
2.12 Comparação com as referências de mercado 28
3. METODOLOGIA 29
3.1 Casos de uso 29
3.1.1 Diagrama de caso de uso 30
3.2 Testes realizados 33
4. RESULTADO E DISCUSSÕES 34
4.1 Telas do aplicativo 36
4.1.1 Tela de Login 36
4.1.2 Tela de carregamento de dados 46
4.1.3 Tela de erro no carregamento de dados 47
4.1.4 Tela inicial 48
4.1.5 Tela de horários 54
4.1.6 Tela de notas parciais 56
4.1.7 Tela de histórico 58
4.1.8 Tela de perfil 60
4.1.9 Tela de recuperação de senha 63
4.2 Repositórios do projeto 64
4.2.1 Padronização dentro do GitHub 65
4.3 Configurações gerais 66
4.3.1 Node_modules 66
4.3.2 Package.json 67
4.3.3 Package-lock.json 67
4.3.4 Tsconfig.json 67
4.3.5 .gitignore 68
4.4 SIGA app back-end 68
4.4.1 Amazon Web Service (AWS) 69

4.4.2 Terraform 69
4.4.3 Lambdas 78
4.4.4 GitHub Workflows 79
4.5 SIGA app front-end 81
4.5.1 GitHub Workflows 82
4.6 Limitações e riscos 83
4.7 Propostas de melhoria futura 84
5. CONCLUSÃO 86
REFERÊNCIAS 88

1. INTRODUÇÃO

O desenvolvimento de aplicativos móveis vem sendo cada vez mais frequente para sistemas utilizados na web ou desktop, pois garantem uma certa facilidade e portabilidade para as pessoas, sendo possível acessar qualquer funcionalidade em qualquer lugar a todo tempo. Esse fato também se deve ao crescente uso de dispositivos móveis, como é mostrado na pesquisa da CNN Brasil em 2022, onde é constatado que mais de 92 milhões de usuário acessam a internet pelo celular, correspondente 62% dos 149 milhões usuários da internet no Brasil (PORTO, 2023). entretanto, o avanço iminente da tecnologia e a democratização dos smartphones, a tendência é que esse número de usuários aumente, ampliando o uso e importante dos aplicativos móveis nos diversos aspectos da vida cotidiana, inclusive na esfera acadêmica.

Dentro do contexto acadêmico, essa demanda por aplicativos se mantém, levando em consideração que os universitários necessitam de acesso rápido e prático às informações sobre o curso, notas, horários e demais atividades. Por conta disso, estão surgindo diversos aplicativos de instituições acadêmicas, como o SUAP Mobile, disponibilizado para o sistema acadêmico SUAP do Instituto Federal de São Paulo, que conta com as funcionalidades sugeridas acima para maior integração dos usuários (IFRN, 2017). Ao oferecer essa conveniência, o aplicativo não só promove uma experiência mais fluida e concisa, mas também contribui para otimização dos processos administrativos e de comunicação, influenciando também na popularidade da instituição, tendo em vista que ela apoia e investe em recursos tecnológicos para melhor convivência.

A instituição de ensino superior Fatec disponibiliza para seus estudantes o SIGA, sistema de gestão acadêmica, que proporciona funções e informações relacionadas à faculdade, sendo de suma importância para a sua comunidade acadêmica, contudo, seu sistema está disponível apenas na plataforma web e não realizar uma adaptação para os celulares, têm sido uma grande barreira para os alunos, quando se trata de facilidade no acesso e visualização de dados. Essa limitação compromete a experiência do usuário no que diz respeito a acessibilidade, gerando descontentamento quando há a necessidade de um rápido acesso para consultar alguma informação.

Tendo em vista esse cenário, foi identificado uma oportunidade de desenvolvimento de um aplicativo móvel para o SIGA para suprir essa necessidade, contudo, algumas limitações em relação ao acesso dos dados presentes na plataforma foram identificadas, pois não é possível realizar o acesso direto à base de dados devido à segurança que a instituição deve ter, além de não ser disponibilizado nenhuma forma oficial de consumo desses dados, como um *web service*.

Portanto o objetivo deste trabalho é desenvolver um aplicativo móvel para Android focado nos estudantes da Fatec, onde irá mostrar os dados acadêmicos, podendo ser acessado com ou sem acesso à internet. Além disso, com a limitação identificada, será desenvolvida uma aplicação para extrair dos dados contidos dentro do SIGA a partir de um navegador, o chamado *web scraping* ou raspagem de dados, que ficará disponível dentro de um ambiente na nuvem do Amazon Web Service (AWS) e poderá ser acessado por uma API configurada dentro do serviço, alimentando o aplicativo criado.

Então, serão abordadas todas as etapas fundamentais durante o desenvolvimento deste trabalho, separando por capítulos. O capítulo 2, abordara os conceitos e ferramentas fundamentais usadas e fará uma comparação de outros trabalhos acadêmicos e do mercado relevantes para tema. Em seguida, no capítulo 3, será apresentado a metodologia utilizada durante o trabalho, os casos de uso definidos e como foram realizados os testes das funcionalidades. No capítulo 4, será apresentado todos os artefatos desenvolvidos, bem como seus pontos principais, incluindo os processos de integração e entrega contínua, *web scraping*, utilização de serviços na nuvem e telas do aplicativo, também serão abordados os detalhes sobre como foram realizados os testes, as melhorias futuras do aplicativo e os possíveis riscos e limitações do software desenvolvido. Finalmente, no capítulo 5, serão feitas as considerações finais, refletindo se o objetivo proposto foi atingido, as limitações dentro do desenvolvimento e as possibilidades para trabalhos futuros.

2. FUNDAMENTAÇÃO TEÓRICA

Nessa seção será abordada os temas relevantes utilizados durante o desenvolvimento deste trabalho, fazendo-se de amplo entendimento as tecnologias e ferramentas aplicadas. Será mostrado o conceito dos seguintes tópicos:

- Sistema operacional e aplicativos móveis;
- Tipos de desenvolvimento de aplicativos móveis;
- *Web Scraping;*
- *API - Application Programming Interface;*
- Versionamento de software;
- *CI/CD – Continuous Integration and Continous Delivery;*
- *IaC – Infraestructure as Code;*
- *IaaS – Infraestructure as a Service;*

Na sequência será mostrado os trabalhos acadêmicos coerentes a este projeto, salientando suas peculiaridades, ferramentas utilizadas e os seus resultados. Alinhado a isso, também serão apresentados produtos disponíveis no mercado relacionados a esse trabalho, apresentando seus objetivos e tecnologias.

2.1 Sistema operacional e aplicativos móveis

Para que possa compreender todo o mecanismo por traz de um celular faz-se necessário o conhecimento do conceito de sistema operacional, que, nada mais é do que um conjunto de instruções executadas pelo processador embutido no equipamento, além disso sua principal função é fazer com que os aplicativos usados pelos usuários possam ser executados em paralelo com todo os outros periféricos e hardware existentes no aparelho, é correto dizer que ele é o intermediador entre ambos (MACHADO; MAIA, 2007). A partir disso, é possível fazer o uso de aplicativos móveis que são programados utilizando alguma linguagem de programação que visa sanar as necessidades do usuário. Atualmente há dois principais sistemas operacionais predominantes no mercado global, sendo eles o IOS desenvolvido pela empresa Apple e o Android atualmente suportado e atualizado pela Google.

2.1.1 iOS

O IOS é um sistema operacional suportado e criado pela Apple e ele é um software fechado, ou seja, somente a empresa tem acesso para utilizá-lo. A linguagem de programação nativa do sistema é o Swift, também criado pela empresa (APPLE, 2014). O desenvolvimento de aplicativos para serem utilizados em seu sistema operacional é necessário fazer o uso de kit de desenvolvimento chamado iOS SDK juntamente com a interface de desenvolvimento Xcode que é disponibilizado pela própria empresa do sistema operacional, Apple. Entretanto, esse kit de desenvolvimento só pode ser utilizados por um dispositivo fabricado pela companhia. É importante salientar que, até o presente momento, os aplicativos desenvolvidos para esse sistema operacional ficam disponíveis apenas em sua loja de aplicativos oficial, chamado de App Store.

2.1.2 Android

Android é um sistema operacional criado pela aliança de empresas *Open Handset Alliance*, que é tem como paradigma o código aberto, ou seja, seu código fonte pode ser livremente modificado para ser usado nos mais diversos locais com diferente objetivos. A interface de desenvolvimento nativa desse sistema é o Android Studio que utiliza o Android SDK e possibilita o uso da linguagem de programação Java e Kotlin. Os aplicativos desenvolvidos para este sistema operacional pode ser encontrados dentro da loja de aplicativos padrão Google Play Store, porém, como se trata de um código aberto, as empresas que fazem dispositivos direcionados para o Android, podem criar a loja de aplicativos especifica para seus produtos, o que aumenta a possibilidade da fonte de acesso aos aplicativos.

2.2 Tecnologia de desenvolvimento de aplicativos móveis

Existem diversas ferramentas e linguagens de programação que pode ser utilizado para desenvolver um aplicativo móvel, entretanto, elas fazem uso de um tipo de desenvolvimento específicos que afetam a linguagem de desenvolvimento que é utilizada, o kit do desenvolvedor que corresponde a um conjunto de recursos

facilitadores para criação de um aplicativo ou programa e a metodologia de programação (PINHEIRO, 2020).

2.2.1 Nativo

O tipo nativo de desenvolvimento da aplicação, varia de acordo com a especificação dada pelo fabricante do sistema operacional, como os exemplos apresentados na sessão anterior sobre os sistemas operacionais, definindo os processos e tecnologias a serem abordados, tais como as linguagens de programação, *SDK*, entre outros fatores (SILVA, 2014). Tal aproximação tem suas vantagens, uma delas é o acesso a todas as funcionalidades nativas presentes no celular.

Tabela 1 – Tecnologias nativas de cada plataforma.

Sistema operacional	Janeiro/2021	SDK	Ambiente de desenvolvimento	Loja de Aplicativos
iOS	Swift	iOS SDK	Xcode	App Store
Android	Java/Kotlin	Android SDK	Android Studio	Google Play Store

Fonte: Pinheiro (2020).

2.2.2 *WebApp*

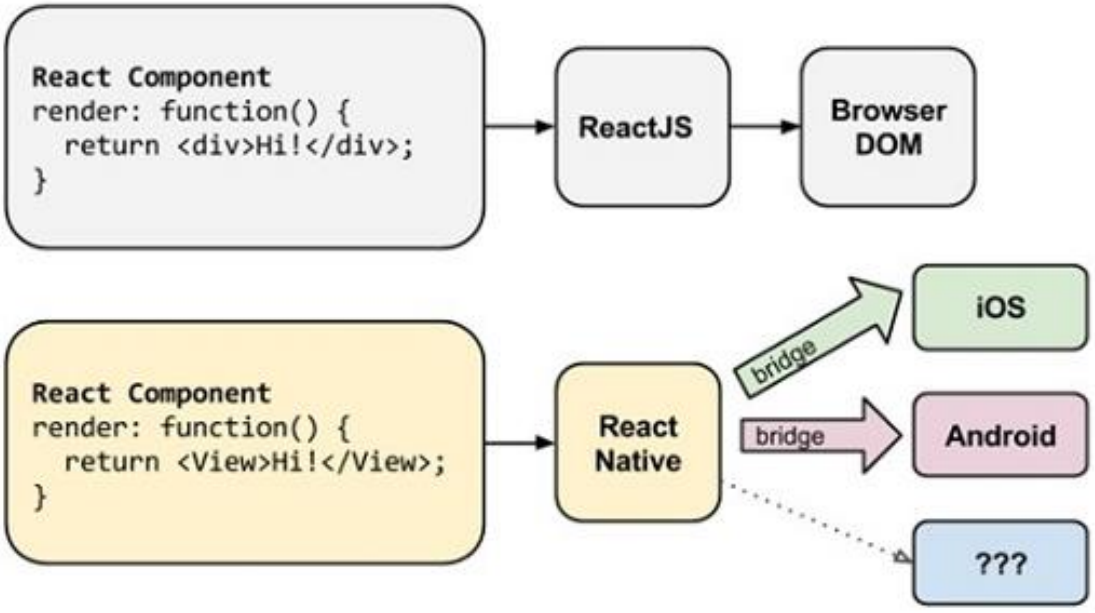
Dentro do formato *WebApp*, é feito com o uso de tecnologias para desenvolvimento de um site, tais como o *Javascript, HTML e CSS*. Portanto, para esse software é uma página otimizada para se comportar como um aplicativo, e para ser utilizado dentro de um dispositivo móvel, é necessário que um navegador web seja iniciado e então execute o conteúdo (PINHEIRO, 2020). Esse navegador fará a simulação de uma aplicativo, sem que seja necessário baixá-lo, já que será executado por uma página na web. Uma desvantagem desse tipo de desenvolvimento é que ela fica limitada ao que diz respeito ao uso de funcionalidades nativas dos dispositivos móveis, tais como notificações, câmera e sistema de arquivo. Porém ele pode ser utilizado em sistemas operacionais diferentes sem que seja necessário refatoração do código.

2.2.3 Híbrido

O modelo híbrido diz a respeito a qualquer linguagem de programação que pode ser traduzida pelo sistema operacional do celular por meio de interpretadores e se transformar em um aplicativo nativo do próprio dispositivo. Esse, diferente do formato de *WebApp*, consegue utilizar as funcionalidades nativas por não serem usados no navegador. Portando, de acordo com o escopo do aplicativo desenvolvido, podem ter vantagem no que diz respeito ao uso dos recursos e todas as funcionalidades nativas do dispositivo (PINHEIRO, 2020).

Uma das principais tecnologias disponíveis que realizam essa técnica é o React Native, um framework de código aberto criado pelo Meta Platforms, Inc. Ele foi desenvolvido com um propósito simples de poupar tempo no desenvolvimento de um aplicativo móvel que suportasse tanto o iOS quanto o Android, além de que ele teve como base o React que já era uma tecnologia existente para desenvolvimentos de aplicações web, portanto a empresa aproveitou essa ferramenta para utilizar no React Native (DANIELSSON, 2016). Eles chamam essa abordagem de "*learn once, write anywhere"*, ou seja, quando se aprende a desenvolver em React, também se aprende a desenvolver em React Native .

Figura 1: Renderização de um componente em React e React Native



Fonte: Danielsson (2016).

A figura 1 ilustra como a renderização do mesmo componente funciona para React e React Native. No primeiro caso, vemos que o componente em React criado utilizando a linguagem de marcação *HTML*, em seguida passa do ReactJS para o Browser DOM onde é integrado ao navegador na estrutura de elementos *HTML*. No segundo caso vemos que há um componente em React utiliza tags parecidas com elementos *HTML*, mas não são, para mapear elementos nativos em cada sistema. Em seguida passa pelo React Native para ir até os sistemas operacionais mobiles por meio de *bridges* que permite a renderização de APIs das respectivas linguagens de cada sistema operacional, como o iOS e o Android (DANIELSSON, 2016).

2.3 *Web Scraping*

O *Web scraping,* também chamado de extração web ou raspagem de dados, é uma técnica utilizada para realizar a extração de dados que estão disponíveis em páginas da *web* por um navegador ou utilizando-se o *Hypertext Transfer Protocol (HTTP)* para realizar uma requisição e obter o conteúdo da página, em seguida os dados são processados e filtrados para que sejam salvos no sistema de arquivos local ou em uma base de dados, posteriormente sendo utilizado para algum propósito (ZHAO, 2017).

O método mais simples, mas trabalhoso para adquirir os dados de uma página é indicando o nome dos elementos *HTML* que contém os dados nos quais seja deseja adquirir, de forma manual, porém essa forma pode apresentar diversas desvantagens, isso porque em sites *web* esses elementos são gerados dinamicamente ter podem ter seus identificadores alterados, invalidando todo seu mapeamento de elementos.

Em contrapartida, é possível utilizar ferramentas chamadas de seletores automáticos, mas ainda assim é necessário realizar a configuração dessas ferramentas para que ela saiba o que procurar e siga uma lógica para isso. A vantagem é que ela vai se adaptar as gerações dinâmicas feitas pelas páginas *web* e não quebrar o fluxo de extração de dados.

Uma das ferramentas mais conhecidas e usadas para esse processo é o Puppeteer, uma biblioteca de código aberto do NodeJs que proporciona uma API de alto nível para controlar navegadores baseados em Chromium (BERTOLINO; FARIA; LAGO; SEMINI, 2024)

2.4 *API – Application Programming Interface*

Uma *API*, também chamada de interface de programação de aplicação, é uma ferramenta utilizada para realizar a comunicação entre sistemas diferentes, dessa forma os desenvolvedores podem fazer programas que se conectam diretamente com um banco de dados ou uma plataforma que disponibiliza dados para consumi-los ou manipulá-los (CASTANHA; SANTOS, 2023). Portanto é uma ferramenta importante para otimização e viabilização de funcionalidades que seriam realizadas de forma manual, fornecendo uma ampla utilização na integração entre as mais diversas aplicações existentes.

No contexto de *API*, há dois padrões importantes que ditam como a comunicação será realizada, o *REST* (*Representational State Transfer)* e o *SOAP* (*Simple Object Access Protocol*). O *REST* é um estilo de arquitetura que funciona a partir da interação entre um cliente e um servidor, onde o cliente realiza uma chamada e o servidor retorna uma resposta. Ele utiliza o protocolo *HTTP* como base, empregando métodos como *GET*, *POST, PUT* e *DELETE (*ANSHU; VIRENDER, 2019). Apesar do *REST* ser muito flexível, um dos seus princípios é a ausência de estado, o que significa que o servidor não mantém nenhuma informação sobre o estado da interação com o cliente entre as requisições, fazendo-se necessário que todas as requisições tenham as informações necessárias para serem processadas.

Em contrapartida, *SOAP* é um protocolo que vai ditar como as *APIs* podem interagir umas com as outras ou com outros clientes. Ele é baseado em *XML*, uma linguagem que estrutura dados de forma organizada e permite que os dados sejam lidos e interpretados por diferentes sistemas. O *SOAP* se caracteriza por ser rígido, pois, ao contrário do *REST,* ele impõe um conjunto de comunicação estrita que deve ser seguida, o que oferece maior segurança e controle, mas torna o protocolo mais complexo e menos flexível.

2.5 Versionamento de software

O versionamento do software é uma prática realizada para controlar as mudanças realizadas no desenvolvimento do software, podendo ser separado por módulos e entregas nos diferentes ramos do projeto para que possa ser testado

separadamente cada alteração, como a parte da interface do usuário e o *back-end* que é o processamento realizado pelo sistema (DIAS; FELLER; MOTTA, 2014).

Essa abordagem é importante, pois garante que caso haja um defeito em algum ponto do progresso no desenvolvimento, é possível reverter a última versão em que o software estava funcionando normalmente, além disso para ambientes corporativos que contam com maior número de contribuintes, é possível realizar o monitoramento e controle de entregas e mudanças feitas por funcionários separadamente.

Há diversas ferramentas que implementam esse conceito e a que é mais utilizada e se destacou globalmente foi o *git*, que é um sistema de controle de versão em que é possível criar um repositório localmente que seja totalmente funcional para o trabalho offline ou remoto, posteriormente sincronizam com o servidor para juntar a cópia do servidor com a local (JACOBS; CASEY; KAIM, 2023). Os principais benefícios em utilizar o *git* está no fato de que a maioria das interfaces de desenvolvimento tem suporte interno a ele. Outras vantagens são que seu código é aberto, ele conta com um forte suporte da comunidade, possibilita o uso de funcionalidade de solicitações *pull* para que a alteração feita seja discutida antes de ser incorporada e permite a configuração de fluxos de trabalho para configurar políticas de versionamento, garantindo que cada alteração atenda a um requisito antes de ser concluída.

2.6 *CI/CD – Continuous Integration and Continuous Deployment*

Continuous integration and continous deployment ou integração continua e entrega continua é um conceito dentro do desenvolvimento de software com o objetivo de automatizar e otimizar os processos de criação, teste, integração e entrega de software, sem que seja preciso se preocupar a todo momento com a implementação do software (ALE, 2020).

A integração continua diz respeito a integração das mudanças que ocorrem em um código, de forma que novas alterações sejam acopladas a um repositório remoto onde essa nova mudança será testada e construída, esse conceito está diretamente alinhado com o versionamento do código. Após isso a entrega continua é aplicada, implementando automaticamente as mudanças do código em um ambiente, seja de desenvolvimento, qualidade ou produção (VALTANEN, 2023).

Ainda na implementação continua, podemos defini-la como um conjunto de processos automatizados, nessa etapa pode ser definidos testes, *builds* e entregas. Esses processos podem ser chamados de *workflow* ou fluxo de trabalho, por onde o código alterado passa (MUKHERJEE, 2024).

Figura 2 - Demonstração de *workflow*

Fonte: What is CI/CD 101 | All You Need To Know (2014)

A figura 2 ilustra um processo de workflow de um software, primeiramente é realizado o desenvolvimento do código, e então é realizado o *commit*, onde essa alteração é enviada para o repositório remoto. Em seguida ele entra na primeira *pipeline* onde esse código é testado e é realizado o *build*. Por fim ele entra na última *pipeline* que faz com que as alterações sejam revisadas e enviadas para produção (RAJKUMAR, 2024).

2.7 *IaC – Infrastructure as Code*

Infrastructure as Code ou infraestrutura como código surgiu como solução para a crescente demanda por escalabilidade e automação de ambientes de nuvem, pois ele traz a possibilidade de gerenciamento da criação de todos os recursos de infraestrutura com práticas de desenvolvimento moderno, como versionamento de software anteriormente citado e testes automatizados (SMITH, 2013).

A infraestrutura como código na prática consiste vai ser composta pelo uso de código fonte para projetar os recursos a serem usados. Nele contém scripts, automações, modelos e parâmetros de configuração que devem ser seguidos para a criação desses recursos. Pode-se dizer que o seu principal objetivo é similar a abordagem do desenvolvimento de software, utilizando linguagem de programação

ou ferramentas para desenvolver soluções adaptáveis e reutilizáveis (NOLL; PRETTO, 2020).

Entre os benefícios de utilizar *IaC* está a repetibilidade e consistência, já que ao utilizarmos código para prover toda a infraestrutura do software, a chance de erros manuais diminui, além de que a mesma configuração pode ser utilizada para múltiplos ambientes, garantindo consistência (SMITH, 2013). Paralelo a isso, escalabilidade e agilidade também é contemplada, uma vez que podemos facilmente alterar as configurações de nossa infraestrutura ou até aumentá-la com uma simples alteração no código (NOLL; PRETTO, 2020). Outra importante vantagem é que quando há falhas de segurança e desastres, onde é possível reconstruir todo o ambiente com *backups* e o próprio código da infraestrutura, reduzindo ao máximo o tempo em que o sistema fica inativo (SMITH, 2013).

Uma das ferramentas que seguem esse conceito e é amplamente utilizada é o Terraform, desenvolvido pela HashiCrop, de código aberto. Nela é possível definir recursos computacionais que sejam em servidores locais ou na nuvem de forma em que a sintaxe do código utilizado para essa ferramenta seja altamente declarativa e de fácil interpretação, sua linguagem é chamada de HashiCorp Configuration Language (HCL). Por meio dessa linguagem é possível definir quais recursos devem ser criados, bem como suas características, o que devem fazer ou como vão interagir entre si (HOWARD, 2022).

2.8 *IaaS – Infrastructure as a Service*

Infrastructure as a service ou infraestrutura como serviço se refere a computação em nuvem, na qual um provedor disponibiliza por meio de servidores, que ele será responsável pela manutenção e atualização da parte do hardware, recursos como máquinas virtuais, armazenamento, firewalls e dispositivos de redes. Além de prover esses recursos, o *IaaS* também fica responsável por fazer os gerenciamentos da infraestrutura dos serviços, no que diz respeito ao provisionamento, alocação, mapeamento de requisitos, modelagem, estimativa e adaptação de recursos computacionais. Um dos principais benefícios de se utilizar esse serviço é a escalabilidade que se pode ter de forma otimizada e eficiente, isso porque os recursos são fornecidos sob demanda e não é necessário ter a preocupação com a parte da infraestrutura de *hardware*.

A Amazon Web Services (AWS) é uma das pioneiras no campo de IaaS, ela oferece diversos serviços em nuvem que abrangem infraestrutura, plataformas e software. Sua plataforma fornece recursos como armazenamento, computação, banco de dados e inteligência artificial, disponíveis sob um modelo de pagamento por uso (AWS, 2024).

Nesse trabalho serão utilizados cinco serviços disponibilizados pela AWS, sendo a Lambda, API Gateway, S3, DynamoDB e CloudWatch. Lambda é um serviço em computação que executa um código em um container isolado dentro de uma máquina virtual que contém memoria, armazenamento e CPU, pode-se entender com função da Lambda o conjunto dos fatores do código, configuração e dependências (SBARSKI; KROONENBURG, 2017). Ela pode ser invocada e pode invocar outros serviços em paralelo, com um vasto leque de possibilidades para ser utilizado. A AWS (2024) diz que o Amazon API Gateway é um serviço que disponibiliza a criação, monitoramento, proteção e publicação de APIs do tipo REST de forma simplificada, e a partir dela chamar outro serviço, como uma Lambda. O S3 ou *Amazon Simple Storage* Service é um serviço de armazenamento de objetos, pode ser arquivos, dados, imagens ou áudios, altamente escalável e seguro, ele foi projetado para armazenar e proteger grandes volumes de dados (AWS, 2024). Seguindo o mesmo raciocínio o *DynamoDB* é um banco de dados *NoSQL* (*Not Only SQL*) sem servidor com baixa latência (abaixo de 10 milissegundos), ou seja, com o tempo de resposta baixo e totalmente gerenciado, ele suporta tabelas de qualquer tamanho e prove alta disponibilidade e desempenho, ele conta com o design chamado de chave-valor ou documento (AWS, 2024). Por fim o *CloudWatch* realiza o monitoramento dos recursos e aplicações na AWS em tempo real, fazendo a coleta de métricas para análise e desempenho, a partir dele é possível criar alarmes, notificações e ajustes automáticos com base nos recursos e em limites configurados, como uso de CPU ou disco (AWS, 2024).

2.9 Trabalhos acadêmicos relacionados

2.9.1 Aplicativo para apresentação de boletim escolar com extração de dados do SUAP através do *Web Scraping*: estudo de caso no IFSP (SILVA, 2019)

Este trabalho mostrou um estudo de caso de uma aplicação existente, que por meio da técnica de *Web Scraping*, extrai dados do sistema acadêmico do IFSP, SUAP, para mandá-los a uma API feita em *JavaScript* e consumi-los em um aplicativo mobile para Android (SILVA, 2019). O sistema foi publicado no play store para que possa ser utilizado em sua versão beta.

2.9.2 PUC-Rio Mobile – Aplicativo de gerenciamento estudantil para os alunos da PUC-Rio (AGUIAR, 2022)

De acordo com Aguiar (2023), o PUC-Rio Mobile visa oferecer informações relevantes do sistema acadêmico SAU para os alunos, de forma a facilitar suas rotinas acadêmicas e acesso às informações da instituição por meio de um app para celular. O aplicativo foi desenvolvido em Flutter e o Firebase, contando com dados simulados, pois não foi possível realizar a integração com o sistema SUAP. Dessa forma foi obtido um aplicativo com funções limitadas e sem integração com o sistema da instituição, mas com grande potencial para otimização e desenvolvimento de mais funções.

2.9.3 SIGAA Mobile – O caso de sucesso da ferramenta de gestão acadêmica na era da computação móvel (FILHO; AQUINO; ROSA, 2013)

Considerando a grande utilização de smartphones e a facilidade que eles provêm, este projeto visa o desenvolvimento de uma aplicação móvel para o SIGA da UFRN, partindo da escolha de funcionalidades e o envolvimento dos usuários foi levantado os principais requisitos funcionais (FILHO; AQUINO; ROSA, 2013). Assim foi desenvolvido o aplicativo que integra com o sistema SIGAA existente e provê uma interface gráfica, com um design participativo e garantindo usabilidade. Como resultado o sistema oferece diversas funcionalidades e está disponível nas plataformas digitais para baixá-lo.

2.9.4 Interface mobile para a plataforma SIGAA (MASCARENHAS, 2022)

Com o objetivo de tornar o aceso ao SIGAA mais prático, eficiente e melhorar a interação entre os alunos e professores da instituição IFAM esse projeto tem como objetivo desenvolver uma interface mobile para o SIGAA utilizando (MASCARENHAS,

2022). Foi identificado as principais funcionalidades que o sistema deveria ter e depois foi realizado o protótipo utilizando a ferramenta Figma. Como conclusão foi desenvolvido um aplicativo Android, embora sem a integração do *back-end* com o sistema da instituição, ficando como melhoria para o projeto.

2.9.5 Jambo: coleta de dados com web scraping (TELES; SILVA, 2021)

Esse trabalho fala sobre o desenvolvimento de uma feramente de pesquisa na internet de informações voltadas para o ambiente acadêmico utilizando a metodologia de *web scraping* com a feramente Jambo (Teles; Silva, 2021). Foram levantando requisitos funcionais e não funcionais para desenvolvimento de uma aplicação, onde pode-se destacar a importância da engenharia de software e que o aplicativo desenvolvido se demonstrou funcional e versátil.

2.10 Comparação entre referências acadêmicas

Tabela 2 – Ferramentas e tecnologias utilizadas por cada projeto.

	Linguagem de Programação	Framework	Conexão Sistema Acadêmico	Banco de dados
SIGA Fatec Mobile	Typescript	React Native	API Gateway (AWS), Web Scraping	Local
Interface mobile SIGAA (MASCARENHAS, 2022)	Java			Assets local
PUC-Rio Mobile (AGUIAR, 2022)	Dart	Flutter		Firebase
SIGAA Mobile (FILHO; AQUINO; ROSA, 2013)	Objective-C	PhoneGap	*Web Services RESTful*	PostgreSQL
SUAP Mobile (SILVA, 2019)	JavaScript	React Native	API, Web scraping	

Fonte: Elaborado pelo autor.

2.11 Aplicativos de mercado relacionados

2.11.1 SIGAUFMG (2024)

O aplicativo SIGA UFMG foi criado com o objetivo de simplificar o acesso dos alunos às informações acadêmicas, necessitando apenas do login e senha no sistema *minhaUFMG*. Sua versão inicial oferece funcionalidades como consulta de matrículas,

notas e planos de estudo, além de notificações em tempo real para alterações acadêmicas. Em constante desenvolvimento, o app pretende incluir novas funcionalidades, como suporte à pós-graduação.

2.11.2 SIGAA Mobile (2024)

O SIGAA Mobile tem como objetivo fornecer acesso rápido às funcionalidades do sistema para discentes e docentes, proporcionando facilidade e praticidade no gerenciamento acadêmico. Com recursos como novo visual, integração com Google Agenda, acesso offline e acompanhamento de bibliotecas, o aplicativo promove uma experiência agradável e eficiente, facilitando o planejamento e acompanhamento das atividades acadêmicas.

2.11.3 ACADESC (2024) – Gestão Escolar

O aplicativo Acadesc é um aplicativo de gestão escolar que facilita a comunicação entre escola, pais, alunos e professores, melhorando a participação dos pais na vida escolar de seus filhos e otimizando a gestão pedagógica. O aplicativo fornece acesso às informações do ACADESC - Software de Gestão Escolar, incluindo boletim, avaliações, agenda escolar e gestão financeira, conforme o perfil do usuário selecionado, promovendo uma experiência prática e acessível para todos os envolvidos na comunidade escolar.

2.11.4 UniCampus (2024)

O aplicativo UniCampus oferece um portal do aluno e portal administrativo atendendo instituições como *UniCampus*, UnitPlus e UnitBrasil. No portal do aluno, os usuários podem consultar notas, financeiro, salas, utilizar o chat e receber notificações. Já no portal administrativo, há acesso a painéis acadêmicos e financeiros.

2.11.5 Portal Aluno UFRJ (2024)

O Portal Aluno UFRJ é um aplicativo dedicado a fornecer serviços fundamentais para os alunos, incluindo emissão de documentos, inscrição em disciplinas, carteirinha digital, visualização de salas de aula e acompanhamento de requerimentos. Através desta plataforma, os estudantes podem acessar com facilidade e praticidade uma variedade de recursos importantes para sua vida acadêmica, otimizando sua experiência na universidade.

2.12 Comparação com as referências de mercado

Tabela 3 – Funcionalidades disponíveis em aplicativos do mercado.

	Mobile Android	Mobile IOS	Autenticação com a conta da instituição	Offline	Boletim	Informações do curso	Calendário	Carteirinha digital	Integração SIGA FATEC
SIGA Fatec Mobile	X	X	X	X	X	X	X	X	X
IFSP Conecta (2024)	X		X		X	X	X		
SIGAUFMG (2024)	X	X	X		X	X			
SIGAA Mobile (2024)	X	X	X	X	X	X	X		
ACADESC (2024)	X	X	X		X	X	X		
UniCampus (2024)	X	X	X		X	X			
Portal Aluno UFRJ (2024)	X	X	X			X		X	

Fonte: Elaborado pelo autor

3. METODOLOGIA

Esse trabalho faz-se o uso da pesquisa aplicada, que consiste no foco em problemas imediatos presentes em instituições, grupos ou organizações para que seja realizado uma identificação de problemas, diagnósticos e busca de soluções por meio de objetivos práticos e aquisição de novos conhecimentos. Portanto a pesquisa aplicada vai reunir tarefas para coletar, selecionar e processar dados para se gerar impacto (FLEURY; WERLANG, 2017).

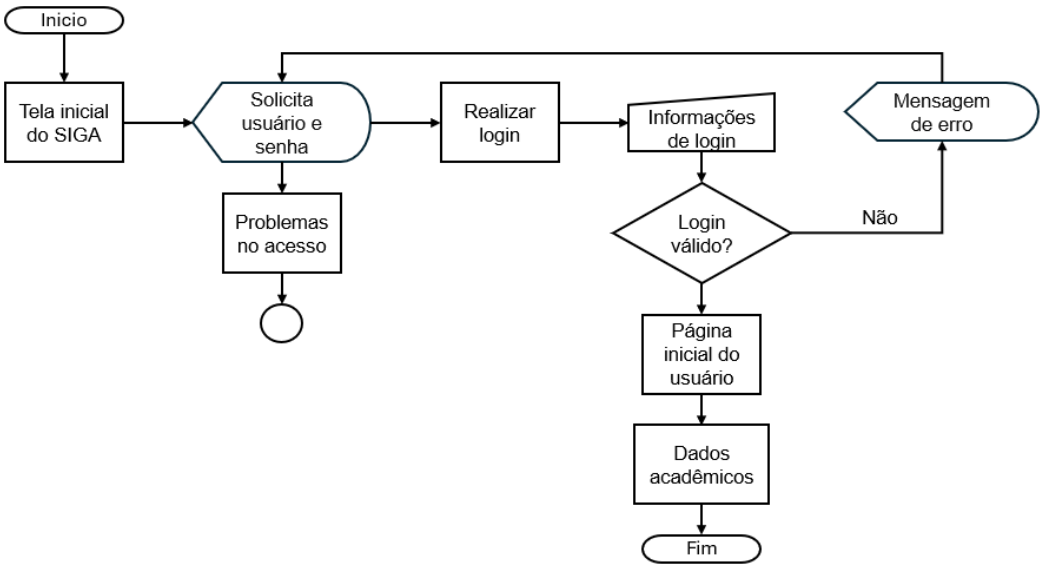
O objetivo imediato deste estudo é desenvolver um software móvel que disponibilize dados acadêmicos dos estudantes, facilitando o acesso a essas informações pelo celular. O sistema será projetado com uma arquitetura em nuvem, integrando práticas de implementação e entrega contínua, além de versionamento de software. Essas tecnologias permitirão a extração e disponibilização eficiente de dados do sistema de gerenciamento acadêmico diretamente no aplicativo.

3.1 Casos de uso

Na etapa de planejamento e desenvolvimento do aplicativo, foi fundamental identificar e descrever os casos de uso, que representam as principais interações entre os usuários e o aplicativo. Eles foram elaborados com o objetivo descrever todas as funcionalidades contempladas nesse aplicativo, além permitem representá-los de forma estruturada, facilitando tanto o entendimento do escopo do projeto quanto o alinhamento com os objetivos propostos. Para o presente trabalho, foram considerados como atores principais o aluno, que acessará as informações acadêmicas, o aplicativo e o sistema SIGA, que será responsável por fornecer os dados necessários.

Antes de definir o caso de uso, foi estabelecido fluxo realizado pelos usuários no SIGA. Esse processo é essencial, pois o aluno precisará passar pela autenticação para acessar todas as funcionalidades disponíveis no sistema. Dado que essa etapa é de extrema importância e não sofrerá alterações no contexto do aplicativo, ela será representada por meio de um fluxograma, para ilustrá-la de forma clara e objetiva.

Figura 3: Fluxograma do acesso ao SIGA



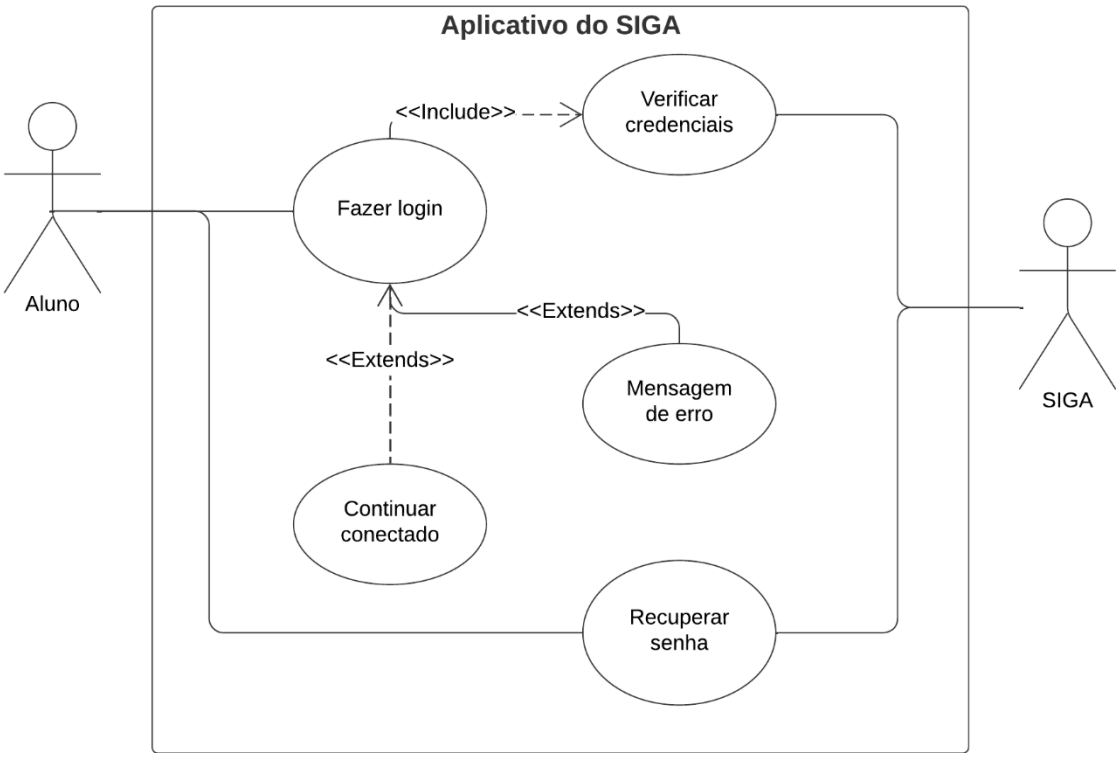
Fonte: Elaborado pelo autor

Na figura acima é possível visualizar de forma clara e sucinta o processo de acesso as informações dentro do siga, onde o usuário deve fornecer seu usuário e sua senha e caso seja correta, ele poderá acessar suas informações, caso contrário, será apontado um erro e terá de reinserir seus dados novamente. O outro processo que não foi totalmente apresentado dentro desse fluxograma, foi a seção para resolver problemas no acesso, que conta com redefinição de senha e identificação do usuário. O motivo pelo qual não foi mostrado se deve pelo fato que ele não será contemplado no aplicativo, pois para realizar esses processos, o SIGA exige que o usuário faça uma verificação por *CAPTCHA* e isso seria inviável para solicitar por meio de um *web scraping*, portanto não foi utilizado no escopo do projeto.

3.1.1 Diagrama de caso de uso

Nesta seção, são apresentados os diagramas de caso de uso elaborados para o aplicativo do SIGA, com o objetivo de representar graficamente as principais funcionalidades do sistema. Esses diagramas destacam os atores principais, suas respectivas ações e os relacionamentos entre as funcionalidades do sistema, permitindo uma análise clara dos fluxos.

Figura 4: Diagrama de caso de uso representando o processo de autenticação do usuário no sistema

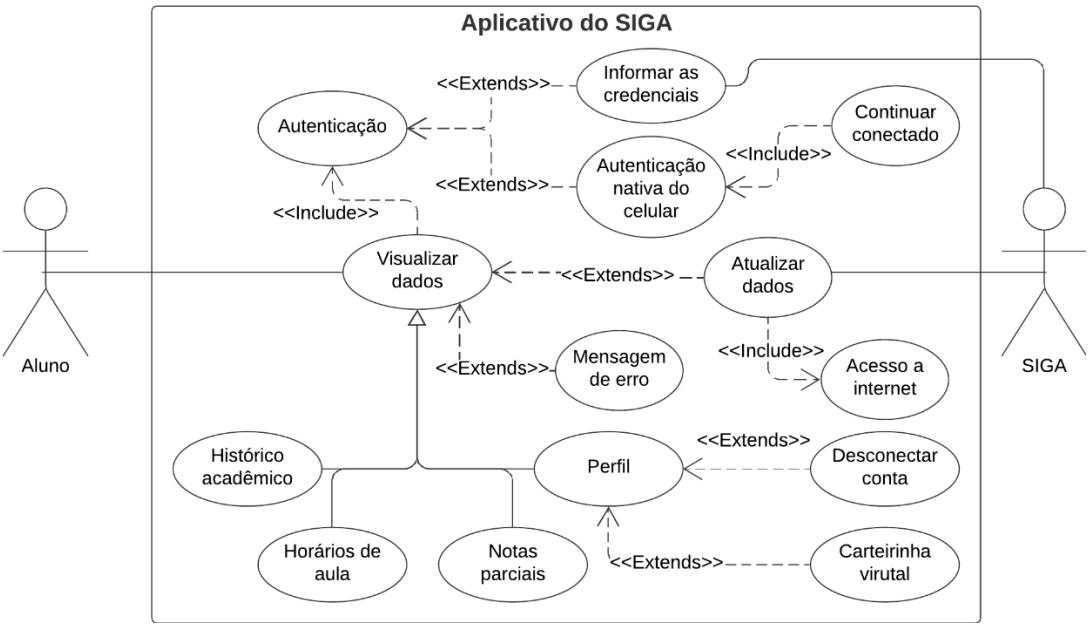


Fonte: Elaborado pelo autor

No diagrama apresentado acima é possível identificar o primeiro caso de uso, a autenticação básico do usuário que acontece quando é realizado o login. As credenciais fornecidas são verificadas pelo SIGA e se qualquer tipo de erro ocorrer, tais como credenciais incorretas, qualquer falha na comunicação com o SIGA ou sem acesso à internet, será mostrado uma mensagem do erro ao usuário. Também é importante comentar que o aplicativo oferece a possibilidade de manter a conta conectada, o que possibilita o uso de funcionalidades como o acesso offline e a utilização das credenciais nativas do celular para autenticação.

Por fim, a opção de recuperação de senha, como mencionado anteriormente, não é uma funcionalidade implementada nesse aplicativo, portanto, o usuário será redirecionado para o site do SIGA quando acessado no aplicativo.

Figura 5: Diagrama de caso de uso representando o processo de visualização de dados

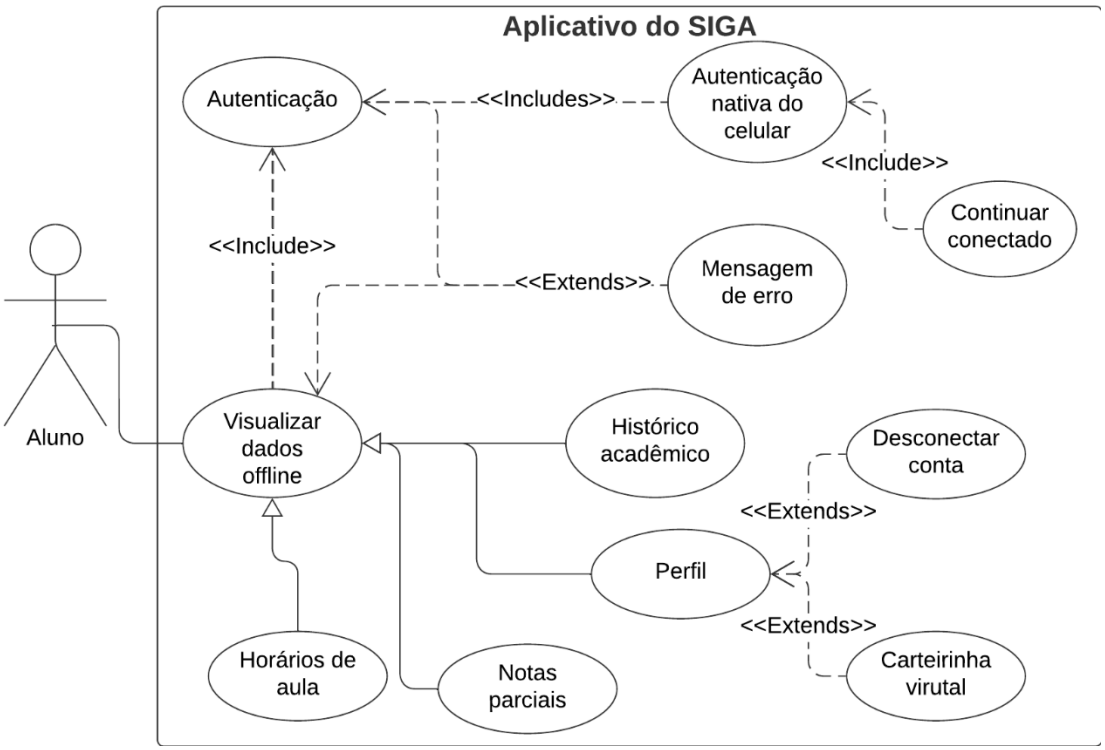


Fonte: Elaborado pelo autor

Na figura acima, é possível visualizar o caso de uso relacionado à visualização de dados dentro do aplicativo. Para ter acesso às informações, é necessário que o usuário esteja autenticado em sua conta, caso já tenha uma autenticação de outra sessão utilizando a opção de continua conectado, ele terá de usar a autenticação nativa do celular, como mencionado anteriormente, senão irá realizar a autenticação com as credenciais do SIGA.

A visualização de dados é dividida em quatro seções: histórico Acadêmico, horários de Aula, notas Parciais e perfil. No Perfil, o aluno tem a opção de desconectar sua conta e visualizar a carteirinha virtual, além de ter os dados de seu perfil. Caso o aluno já tenha estado conectado, os dados exibidos serão os armazenados da sessão anterior. Caso contrário, as informações serão atualizadas diretamente do SIGA. Além disso, o aluno tem a funcionalidade de atualizar os dados, trazendo as informações mais recentes do SIGA.

Figura 6: Diagrama de caso de uso representando o processo de visualização de dados offline



Fonte: Elaborado pelo autor

O diagrama acima representa um caso de uso parecido com o anterior, tendo como diferença a visualização de dados offline, ou seja, sem conexão com a internet. Para seguir esse fluxo é obrigatório que o usuário tenha se autenticado em uma sessão anterior e optado por manter o login, portanto a autenticação offline será sempre realizada utilizando as credenciais nativas do dispositivo. Se não houver nenhuma conta previamente autenticada será exibido uma mensagem de erro. Além disso, os dados mostrados serão aqueles que armazenados localmente, e a opção de atualização dos dados não estará disponível. Caso ocorra algum erro de não houver dados salvos localmente, o aplicativo apresentará uma mensagem informando ao usuário.

3.2 Testes realizados

Os testes realizados no aplicativo foram conduzidos pelo autor, fazendo o uso de suas próprias credenciais para o acesso ao SIGA, com o objetivo de validar o funcionamento correto das funcionalidades definidas e garantir que as informações exibidas sejam as mesmas da fonte original.

No primeiro momento foi verificado se o processo de login funcionava corretamente, tanto com as credenciais de usuário e senha para autenticação do SIGA quanto utilizando a abordagem de autenticação nativa do celular, dessa forma garantindo que o usuário fosse identificado sem erros.

Em seguida, foram realizados testes para validar a funcionalidade de visualização de dados com e sem conexão à internet. Primeiramente garantindo que após a efetuação do login, sem acessar com a opção de continuar conectado, o sistema realizava a busca pelos dados mais atualizados do SIGA, verificando-se também o uso da funcionalidade de atualizar os registro. Para o cenário offline, foi verificado se o aplicativo acessava corretamente as informações armazenadas no dispositivo e se mostrava uma mensagem caso não houvesse dados locais. Em ambos os casos, foi feito a checagem das informações apresentadas com as disponíveis no site do SIGA.

Também foram verificados os cenários de erros gerais que incluem falta de conexão com a internet para realização de alguma função que depende da mesma, qualquer falha durante a chamada da API para o siga e credenciais invalidas para acesso ao SIGA. Para conseguir fazer essas verificações, foram necessários alguns métodos para forçar essas situações, tais como desligar a internet para tentar executar o login e verificar seu comportamento mesmo depois de autenticado e forçar a atualização dos dados a todo momento. Isso possibilitou que todas as mensagens de erro fossem acessadas e validadas para que sejam exibidas de forma clara e concisa.

Todos os testes apresentados anteriormente, foram realizados de forma manualmente, seguindo todos os casos de uso. E para determinar se os testes realizados ocorreram conforme as expectativas e atenderam com sucesso, foi levado em consideração a natureza do teste. Para os que se tratava de trazer informações do apresentá-los, passavam os que correspondiam corretamente com os dados presentes dentro do site do SIGA e para os outros esperava-se que uma mensagem correspondente e objetiva do problema apresentado fosse mostrada na tela.

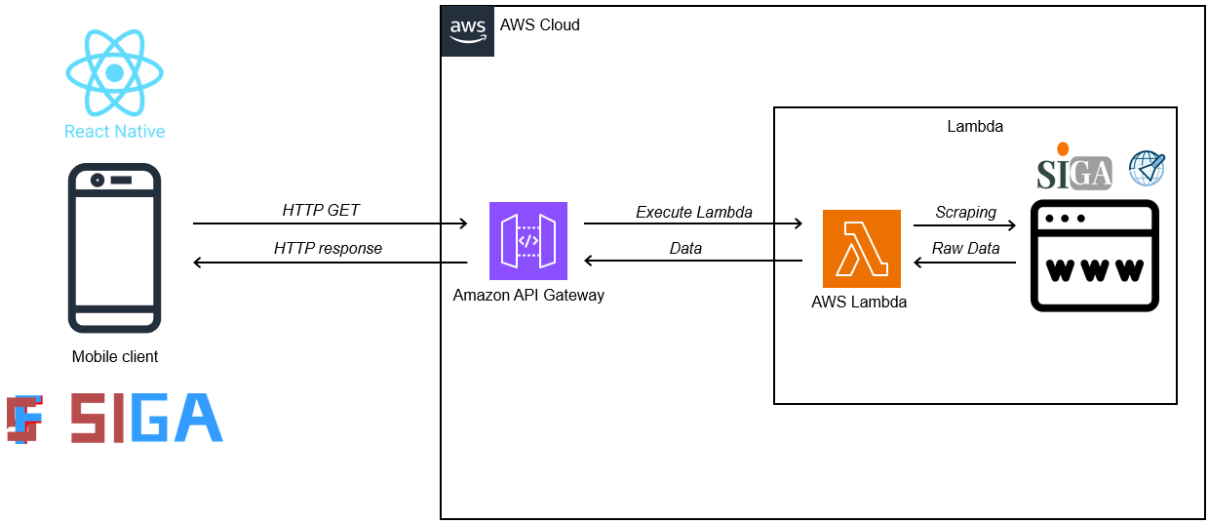
4. RESULTADO E DISCUSSÕES

Diante da limitação de compatibilidade e falta de um aplicativo móvel do sistema de gerenciamento acadêmico da Fatec, este trabalho propôs e desenvolveu

uma solução para superar esses problemas. A proposta inclui um aplicativo que consome dados de uma API baseada em *web scraping*, oferecendo praticidade no acesso às informações acadêmicas e preenchendo essa importante lacuna tecnológica.

Em uma visão geral da arquitetura do projeto, podemos dividir em três principais momentos, sendo o primeiro referente ao aplicativo no qual apresenta as informações, o segundo é a requisição feita para solicitar os dados e o terceiro é o processamento feito para resgatar esses dados dentro da página do SIGA. A ilustração a seguir representa exatamente o que foi explicado:

Figura 7: Diagrama da arquitetura do projeto



Fonte: Elaborado pelo autor.

Na figura acima é possível visualizar o fluxo de como o sistema funciona, começando pelo *Mobile client* ou celular que está com o aplicativo do SIGA, posteriormente o app faz requisições HTTP para uma API que está dentro do ambiente da AWS. Por sua vez, quando a API recebe qualquer requisição GET, ela executa a lambda responsável por conter o código que realiza o *web scraping* dentro do navegador no site do SIGA. Quando a extração dos dados é concluída, a função lambda resgata essas informações e as transforma para ser retornada em uma resposta HTTP, na qual a API recebe e envia para o aplicativo que fez a solicitação. Em seguida o app é responsável por ler esses dados vindos da reposta e manipular como bem queira.

4.1 Telas do aplicativo

Nesta seção será apresentado as telas do aplicativo conforme o fluxo do sistema, iniciando pela tela de login e seguindo para as funcionalidades subsequentes, conforme o usuário navega através das opções e interações disponibilizadas pela aplicação. Também será apresentado as exceções aos erros mapeados e não mapeados dentro do aplicativo.

4.1.1 Tela de Login

Essa vai ser a tela que o usuário vai acessar quando abrir o aplicativo pela primeira vez. Então, a partir desse ponto será possível realizar duas ações para navegar para outra tela, sendo a primeira clicando no texto sublinhado que está no rodapé da tela escrito “Esqueceu sua senha?” ou preenchendo os dois campos de texto na tela com o usuário e senha para acessar o sistema.

Figura 8: Tela de login do aplicativo

Fonte: Elaborado pelo autor.

4.1.1.1 Verificação dos campos

O sistema faz dois tipos de verificações dos campos de login e senha quando o botão de entrar é clicado, e depois realiza a chamada para a API se não falhar nas verificações. Primeiro é verificado se os campos não estão vazios, pois é obrigatório que os dois estejam preenchidos, depois é feito uma verificação se a quantidade de caracteres fornecidos no login é de 11 caracteres, já que isso é um padrão para acessar o SIGA.

Figura 9: Tela de login com mensagem de erro, "Preencha todos os campos"

Fonte: Elaborado pelo autor.

Na figura anterior é possível visualizar a mensagem que aparece para o usuário caso ele pressione o botão sem antes preencher os campos.

Figura 10: Tela de login com notificação informando que o login é inválido

Fonte: Elaborado pelo autor

Já na figura acima, é mostrado uma notificação informando o usuário que o Login está inválido, visto que o que foi digitado não contempla o padrão de 11 caracteres.

4.1.1.2 Enviando credenciais

Quando o usuário preenche corretamente os campos necessários e clica no botão de entrar, ele é notificado que sua solicitação está sendo verificada.

Figura 11: Tela de login com notificação informando que as credenciais estão sendo verificadas

Fonte: Elaborado pelo autor

4.1.1.3 Sem conexão com a internet

Caso o dispositivo do usuário esteja sem conexão com a internet e ele tente realizar o login sem ter acessado previamente com uma conexão estável com a internet e escolhido para continuar conectado, será mostrado uma notificação avisando que não há conexão com a internet, portanto nenhuma ação será realizada.

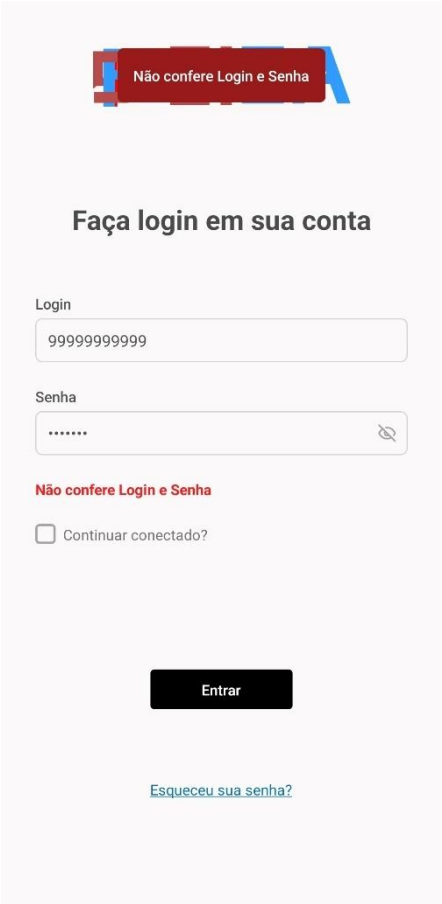
Figura 12: Tela de login com notificação informando que o dispositivo está sem conexão com a internet

Fonte: Elaborado pelo autor

4.1.1.4 Credenciais incorretas

Caso a verificação das credenciais passe pelas verificações, é enviado uma requisição para a API que vai retornar os dados básicos do usuário caso seja válido, se não ela vai retornar uma mensagem de erro dizendo que o login e senha não estão corretos.

Figura 13: Tela de login com mensagem informando que o login e senha não conferem



Fonte: Elaborado pelo autor.

4.1.1.5 Prevenção de erros na API

Como a obtenção de informações depende de uma conexão estável com a internet para fazer a requisição a API e do site do SIGA estar funcionando corretamente, é importante que ocorra uma série de verificações para que caso não seja possível concluir o processamento como esperado, seja retornado uma mensagem informando que houve problema da parte da API para frente.

Figura 14: Tela de login com notificação informando problema com o servidor

Fonte: Elaborado pelo autor.

4.1.1.6 Continuar conectado

Quando o usuário já fez um login bem-sucedido no aplicativo anteriormente e selecionou a opção de “Continuar conectado?”, nas próximas vezes que ocorrer o acesso no app, será solicitado que o usuário se autentique utilizando a forma nativa padrão definida pelo celular que está utilizando. Essa função vai estar sempre disponível, mesmo que o dispositivo esteja sem conexão com a internet, basta que o usuário já tenha feito login em outro momento e selecionou a opção para manter login. Caso essa autenticação nativa seja cancelada e o usuário não autentique, o aplicativo apaga todas as informações desse usuário e entende que foi solicitado para fornecer as credenciais novamente e desconecta esse usuário. Se a autenticação nativa for bem-sucedida, será possível acessar o aplicativo normalmente e consultar os dados previamente guardados no último login, sem que necessite de internet.

Figura 15: Tela de login pedindo autenticação nativa para acessar usuário já logado

Fonte: Elaboração pelo autor.

A figura anterior foi retirada de um celular que tem a autenticação padrão nativa como a digital, portanto foi essa a solicitada. A partir dela é feito a verificação se o usuário poderá prosseguir com a sessão ou não.

Figura 16: Tela de login com notificação e sucesso após autenticação nativa

Fonte: Elaborado pelo autor.

A figura 16 mostra quando a autenticação nativa foi concluída com sucesso, portanto o acesso a conta anteriormente conectado é permitido.

Figura 17: Tela de login com notificação informando que a autenticação nativa foi cancelada

Fonte: Elaborado pelo autor.

A figura acima mostra o cenário onde o usuário cancelou a autenticação nativa padrão, e, portanto, é solicitado que ele se autentique utilizando as credenciais de login e senha novamente para acessar o aplicativo.

4.1.2 Tela de carregamento de dados

Em todas as telas, após o login, em que há a requisição de dados do usuário para a API seja para trazer os dados no primeiro acesso ou porque o usuário aperte o botão de recarregar informações disponível no cabeçalho da tela, ela irá entrar no modo de carregamento padrão para todas.

Figura 18: Tela de carregamento padrão

Fonte: Elaborado pelo autor.

4.1.3 Tela de erro no carregamento de dados

Assim como a tela de carregamento de dados padrão, há uma tela de erro que será utilizada para todas as telas que envolvem o carregamento de dados caso não seja possível resgatar essas informações na API, por qualquer motivo.

Figura 19: Tela padrão notificando erro ao carregar os dados

Fonte: Elaborado pelo autor.

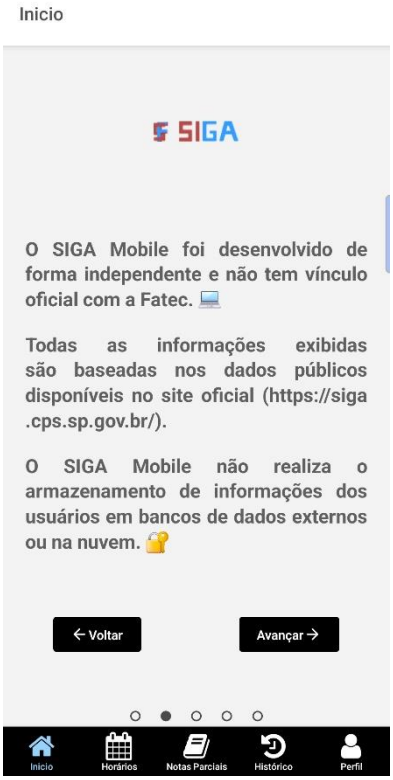
4.1.4 Tela inicial

Após o login ser efetuado com sucesso, a primeira tela a ser mostrada vai ser a tela inicial. Nessa tela serão mostradas informações sobre o aplicativo, falando um pouco do projeto, informando que o aplicativo não armazena as informações dos usuários, mostrando o que está disponível dentro dele e disponibilizando um link para acessar detalhes técnicos do desenvolvimento do app.

Figura 20: Tela inicial do aplicativo, parte 1

Fonte: Elaborado pelo autor

Figura 21: Tela inicial, parte 2



Fonte: Elaborado pelo autor.

Figura 22: Tela incial, parte 3

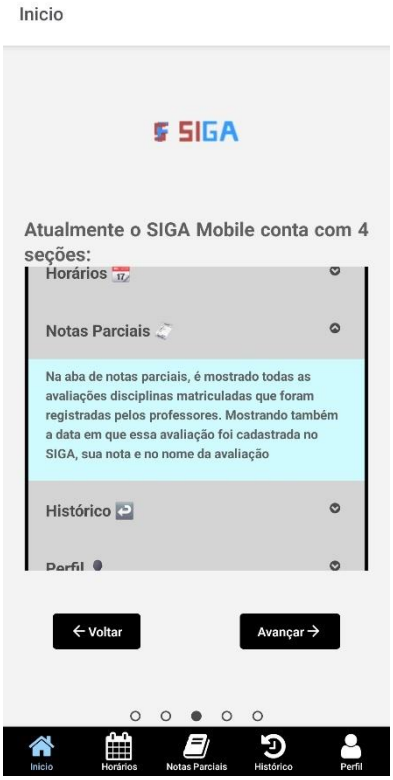
Fonte: Elaborado pelo autor

A figura acima apresenta uma informando as 4 seções de dados disponíveis no SIGA separados por acordeões, e cada um vai fornecer uma explicação sobre cada seção.

Figura 23: Tela inicial com acordeão e horários aberto, parte 3

Fonte: Elaborado pelo autor.

Figura 24: Tela inicial com acordeão de notas parciais expandida, parte 3

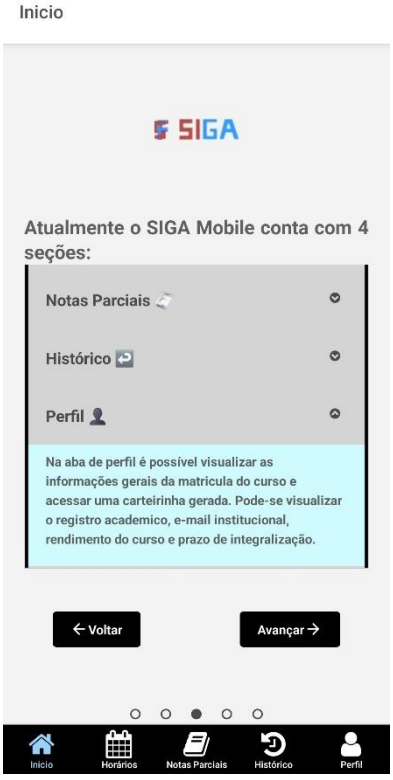


Fonte: Elaborado pelo autor

Figura 25: Tela inicial com acordeão de históricos expandido, parte 3

Fonte: Elaborado pelo autor

Figura 26: Tela inicial com acordeão de perfil expandido, parte 3



Fonte: Elaborado pelo autor

Figura 27: Tela inicial, parte 4

Fonte: Elaborado pelo autor

Figura 28: Tela inicial, parte 5

Fonte: Elaborado pelo autor.

4.1.5 Tela de horários

A seguir, será apresentada a tela de horários, onde o usuário pode visualizar as aulas programadas para a semana, incluindo o nome da disciplina, o horário e a turma correspondente. Na parte superior, há um acordeão que permite consultar todas as matérias em que o aluno está matriculado, exibindo suas siglas e a duração de cada aula.

Figura 29: Tela de horário com acordeão fechado

Fonte: Elaborado pelo autor.

Figura 30: Tela de horários com acordeão expandido

Fonte: Elaborado pelo autor.

4.1.5.1 Sem conexão com a internet

Caso o usuário acesse no aplicativo sem conexão com a internet, mas tenha acessado sua conta anteriormente e escolhido manter o login, uma mensagem será exibido no lugar do ícone de recarregar o conteúdo, informando o usuário da perda de conexão.

Figura 31: Tela de horário sem conexão com a internet

Fonte: Elaborado pelo autor

4.1.6 Tela de notas parciais

Nessa tela será apresentada as notas parciais do usuário, onde o aluno pode visualizar as notas atribuídas a cada disciplina matriculada no atual semestre. São exibidas informações sobre a nota obtida, a data da última atualização e o nome da avaliação correspondente.

Figura 32: Tela de notas parciais, parte 1

Fonte: Elaborado pelo autor.

4.1.6.1 Sem conexão com a internet

Nessa tela também será exibido as informações previamente salvas no celular, caso o aluno tenha acessado sua conta anteriormente e escolhido manter o login, será informado ao usuário a perda de conexão.

Figura 33: Tela de notas parciais sem conexão com a internet

Fonte: Elaborado pelo autor.

4.1.7 Tela de histórico

A tela de histórico está organizada por semestres em formato de acordeões, que podem ser expandidos para exibir detalhes sobre o período cursado. Para cada disciplina, são apresentadas a média final, frequência, quantidade de faltas e uma observação sobre a situação do aluno na matéria. Além disso, ao lado da sigla da disciplina, um *badge* indica o status atual, como aprovado, reprovado ou está em curso, oferecendo uma visão clara e organizada do desempenho acadêmico ao longo do curso.

Figura 34: Tela de histórico com acordeões fechados

Fonte: Elaborado pelo autor

Figura 35: Tela de histórico com acordeão expandido

Fonte: Elaborado pelo autor.

4.1.7.1 Sem conexão com a internet

Para a tela de histórico, também será possível visualizar os dados sem conexão com a internet caso o usuário tenha logado previamente, indicando a falta de internet.

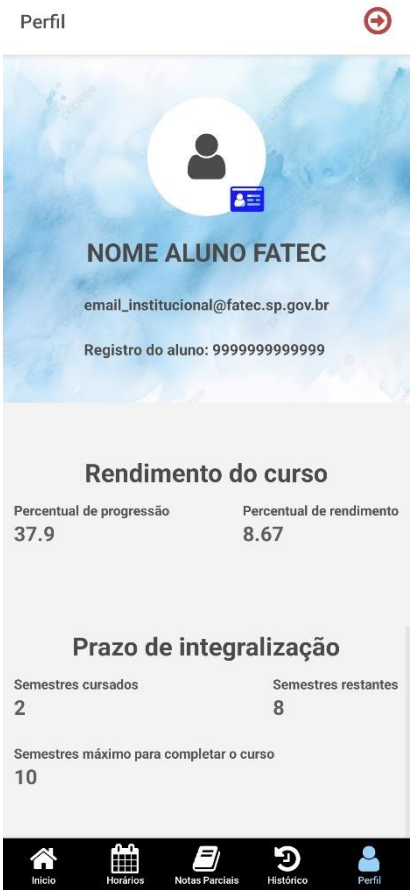
Figura 36: Tela de histórico sem conexão com a internet

Fonte: Elaborado pelo autor.

4.1.8 Tela de perfil

A tela de perfil exibe informações acadêmicas do usuário. No topo, apresenta a imagem de perfil vinculada ao SIGA, acompanhada do nome, e-mail acadêmico e registro do aluno. Mais abaixo, fornece dados sobre o progresso no curso, incluindo informações sobre rendimento acadêmico e prazo de integralização.

Figura 37: Tela de perfil



Fonte: Elaborado pelo autor

No cabeçalho ao lado direito, há um ícone que pode ser clicado e efetuado a saída do usuário do aplicativo, desconectando sua foto e retornado para a tela de login.

Figura 38: Tela de login notificando a saída da conta

Fonte: Elaborado pelo autor

4.1.8.1 Carteirinha de informações

O aplicativo oferece uma carteirinha com informações do aluno, acessível por meio de um ícone localizado ao lado direito da foto de perfil. Ao clicar no ícone, o usuário é direcionado para uma nova tela onde pode visualizar os detalhes da carteirinha, que fica disponível sem conexão com a internet.

Figura 39: Tela da carteirinha do aluno

Fonte: Elaborado pelo autor.

4.1.9 Tela de recuperação de senha

Na tela de recuperação de senha que pode ser acessada a partir da tela Login, tem a função de levar o usuário até a página de recuperação de senha do site do SIGA, portanto, essa funcionalidade em si, não está no escopo desse projeto.

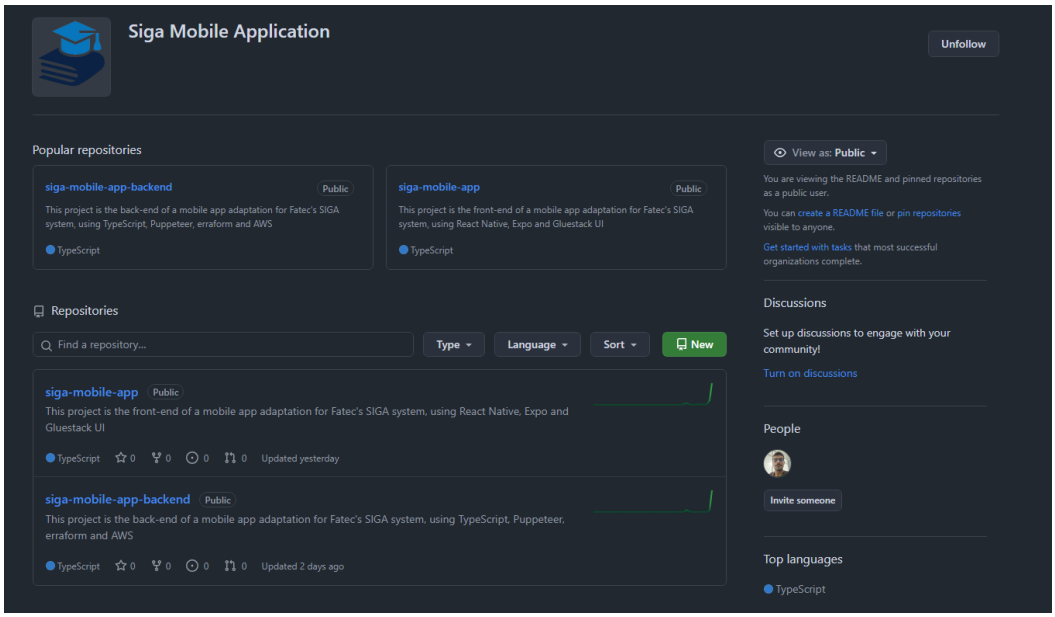
Figura 40: Tela de recuperação de senha

Fonte: Elaborado pelo autor.

4.2 Repositórios do projeto

Para obter uma melhor organização do projeto foi realizado a criação de uma organização no GitHub, ela permite nos permite centralizar os repositórios, facilitar a colaboração e garantir a escalabilidade do desenvolvimento. Com ela, foi possível organizar o código, documentações e scripts de forma estruturada, além de gerenciar permissões de acesso de maneira eficiente. Essa abordagem também garante continuidade ao projeto, já que por ser público, ele permite que qualquer um possa sugerir alguma mudança ou alteração, contribuindo para uma abordagem profissional.

Figura 41: Organização criada para o projeto no GitHub

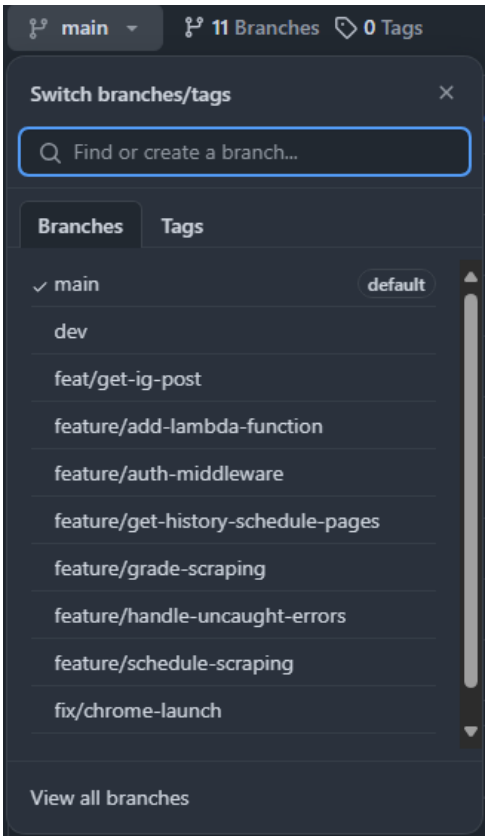


Fonte: GitHub (2024)

4.2.1 Padronização dentro do GitHub

Foi implementada uma padronização no uso de *branches*, ou ramificações, no GitHub. A *branch main* é utilizada como produção e é a padrão, enquanto a *branch qual* centraliza as mudanças em desenvolvimento, e é por onde são realizados testes mais avançados com todas as funcionalidades ao mesmo tempo. Todas as outras *branches* criadas são consideradas de desenvolvimento, fora as duas citadas, e utilizam um prefixo que indica a finalidade ou tarefa que estão realizando, como por exemplo ‘feature/nome-da-funcionalidade’ ou ‘fix/nome-do-correção’. Esse padrão garante organização e clareza no fluxo de trabalho dos repositórios.

Figura 42: Padronização de branches dentro do repositório do aplicativo



Fonte: GitHub (2024)

4.3 Configurações gerais

Dentro de ambos as pastas do back-end e do front-end, temos alguns arquivos padrão que desempenham um importante papel para o funcionamento e organização do projeto.

Outro detalhe essencial é que, para todo o projeto, estamos utilizando o npm (Node Package Manager) como gerenciador de pacotes do NodeJs. Ele nos permite instalar e gerenciar bibliotecas e ferramentas JavaScript de forma eficiente. Com o npm, controlamos as versões das dependências, garantindo consistência no desenvolvimento. Além disso, ele facilita o compartilhamento de pacotes e a automação de tarefas do projeto, promovendo um fluxo de trabalho organizado e garantindo qualidade no ciclo de desenvolvimento.

4.3.1 Node_modules

Esse diretório é gerado pelo npm e nele contém todas as dependências baixadas que nosso projeto necessita para poder rodas. Ele sempre será criado automaticamente a partir de outros dois arquivos chamados package.json e do package-lock.json, que guardam exatamente as bibliotecas e configurações necessárias para baixar sempre os mesmos arquivos dentro do node_modules.

O node_modules fica fora do controle de versionamento do projeto, isso porque ele é extremamente grande e isso tornaria o repositório lento e nada prático, além de que ele pode ser reinstalado localmente com as mesmas dependências a partir dos outros dois arquivos.

4.3.2 Package.json

Ele é o arquivo central e necessário dentro de qualquer projeto que utilizar Node.js, nele é possível encontrar informações e metadados relacionados ao projeto, tais como nome do projeto e versão. Ele também armazena todas as dependências necessárias para a execução e desenvolvimento do projeto, além de fornecer scripts pré-definidos que ajuda na automação de tarefas como teste, *build* e *deploy*.

4.3.3 Package-lock.json

O package-lock.json é gerado automaticamente quando utilizamos o npm para instalar as dependências do nosso projeto, garantindo que as versão de cada dependência e sub dependência sejam exatamente as mesma para todos as ambientes, isso promove consistências para o desenvolvimento, evitando atualizações inesperadas de pacotes.

4.3.4 Tsconfig.json

Esse arquivo vai disponibilizar as configurações do TypeScript que define a forma de compilação dos arquivos, como o destino do JavaScript gerado, inclusão ou exclusão de arquivos no projeto e configurações para facilitar a depuração e a integração com ferramentas.

Ele será sempre necessário em projetos TypeScript para personalizar o comportamento da transpilação, que é a conversão de uma linguagem para outra, e garantir que o código SIGA as diretrizes estabelecidas.

4.3.5 .gitignore

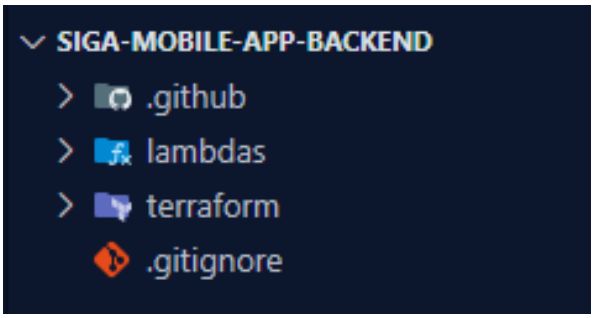
Este arquivo define quais arquivos ou diretórios o Git deve ignorar quando for enviar todas as alterações feitas localmente para o repositório remoto. É essencial para evitar o versionamento de arquivos desnecessários, como arquivos temporários, logs e dependências armazenadas em node_modules. Isso reduz o tamanho do repositório e mantém o foco em arquivos críticos para o projeto, além de garantir agilidade dentro dos ambientes.

4.4 SIGA app back-end

O back-end é a parte responsável por fazer toda a lógica necessária para o funcionamento do nosso sistema, nesse contexto ela se refere a todo o processo que ocorre desde a API até o extrator de dados do SIGA.

Para analisarmos como foi desenvolvido o back-end do aplicativo e como utilizamos e integramos com a AWS, é importante destacar como os arquivos estão estruturados dentro da raiz do projeto.

Figura 43: Estrutura de arquivos do back-end



Fonte: Elaborado pelo autor

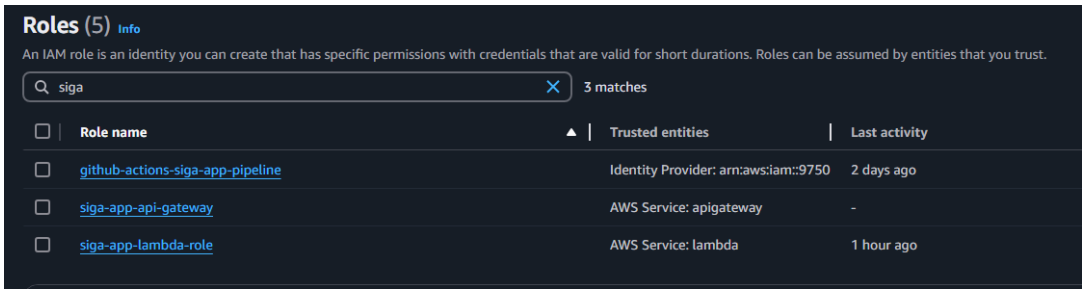
Na figura acima, a forma como esses arquivos estão organizados foi realizada para mantermos um padrão no desenvolvimento, dessa forma quando precisamos

alterar alguma configuração especifica ou acessar algum arquivo específico dentro do back-end, saberemos exatamente onde ela está.

4.4.1 Amazon Web Service (AWS)

Para implementar o Terraform na AWS, foi necessário criar uma conta na plataforma, utilizando o plano de nível gratuito para viabilizar o desenvolvimento sem custos adicionais. Foi configurada uma role específica, com permissões adequadas para que o Terraform pudesse acessar e gerenciar os recursos da AWS de forma segura e automatizada, além da criação de roles para os recursos que foram utilizados, como o Lambda e o API Gateway. Além disso, foi necessário integrar a organização do GitHub com a conta AWS, de forma que os *workflows* do GitHub Actions realizassem *deploys* e gerenciassem os recursos da infraestrutura conforme definido no código do projeto.

Figura 44: Visualização do painel de roles dentro da AWS

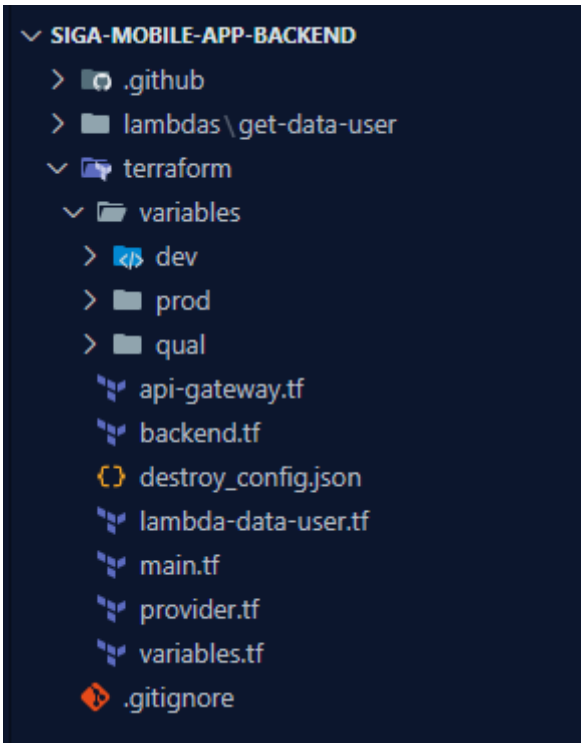


Fonte: AWS (2024)

4.4.2 Terraform

Dentro desse diretório estão todos os arquivos terraform utilizados no *back-end*, eles são utilizados para definirmos e gerenciarmos nossa infraestrutura e recursos dentro de AWS.

Figura 45: Diretório do back-end com a pasta terraform expandido



Fonte: Elaborado pelo autor

Como é possível visualizar na figura mostrada acima, a estrutura está organizada de tal forma que os recursos que serão criados estão divididos por arquivos e funções principais para a criação desses recursos. Isso também contempla o uso de variáveis que auxiliam durante a criação para gerenciar para qual ambiente esses recursos estão sendo criados ou alterados.

4.4.2.1 Variables.tf

Como o próprio nome do arquivo já fornece um indicativo do seu propósito, ela vai ser encarregada de estabelecer variáveis ou constantes que podem ser usadas em todos os outros arquivos Terraform. Se criarmos uma constante, teremos que passar seu valor fixo, mas se criarmos uma variável temos que indicaremos o tipo dela e quando formos executar o Terraform, dizer qual valor essas variáveis vão assumir, poder meio de outro arquivo e então dinamicamente ele as incorpora, isso é realizado durante a execução da Workflow do GitHub.

Figura 46: Arquivo variables.tf

Fonte: Elaborado pelo autor.

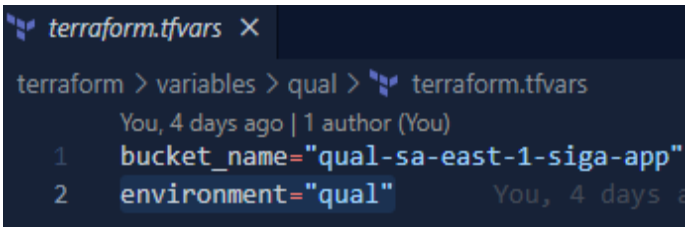
A figura anterior apresenta a criação das dos ‘locals’ que são as contantes e já recebe um valor predefine, no caso o nome do aplicativo, e as duas outras variáveis representando o nome do bucket no s3 que será acessado ou criado e qual o ambiente para qual será executado o Terraform, cada qual com seu tipo, ‘string’.

Em paralelo com essas variáveis, o Terraform disponibiliza outra extensão auxiliar de arquivos para trabalhar junto com as variáveis, são os ‘.tfvars’, no qual irão definir valores que poderão ser incorporados dinamicamente. Nesse caso, foi criado três arquivos, cada um contendo seu valor para as variáveis, seguindo o padrão de criar três ambientes, desenvolvimento (dev), qualidade (qual) e produção (prod).

Figura 47: Arquivo terraform.tfvars da pasta de dev

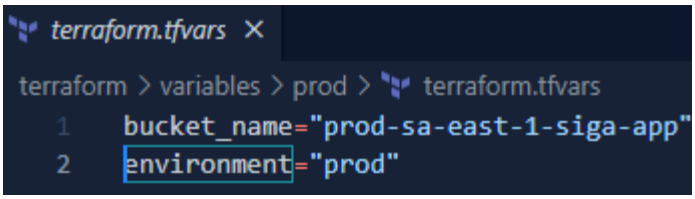
Fonte: Elaborado pelo autor.

Figura 48: Arquivo terraform.tfvars da pasta de qual



Fonte: Elaborado pelo autor.

Figura 49: Arquivo terrafrom.tfvars da pasta de prod



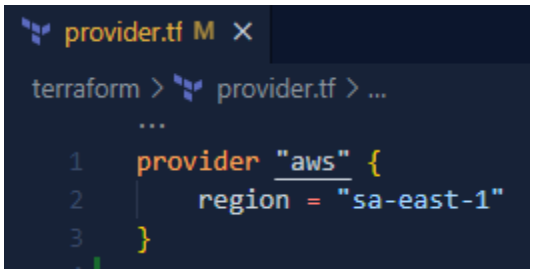
Fonte: Elaborado pelo autor.

As três figuras anteriores mostram as variáveis que são passadas para cada ambiente, os diferenciando para que não utilizem os mesmos recursos. Para que posteriormente os outros arquivos Terraform acessem essas variáveis e constates, basta que sinalizem qual o escopo, '*locals'* ou '*variable'*, acrescentar um ponto e o nome, por exemplo 'locals.environment'.

4.4.2.2 Provider.tf

Este arquivo é responsável por sinalizar qual o provedor de serviços de nuvem que iremos utilizar e em qual região geográfica nós iremos utilizar para criar nossos recursos. Essas informações são muito importantes para dizer para orientar o Terraform de como ele deve seguir na sua execução.

Figura 50: Arquivo provider.tf



Fonte: Elaborado pelo autor.

4.4.2.3 Main.tf

Dentro do main.tf será definido recursos que serão utilizados para a auxílio na manutenção e atualização do código Terraform em si e dos recursos que serão atualizados ou criados dentro da AWS, porque ele armazena qual o estado desses objetos e sempre que uma requisição for realizada, ele vai comparar os estados nesse armazenamento e atualizá-lo.

Figura 51: Arquivo main.tf

Fonte: Elaborado pelo autor.

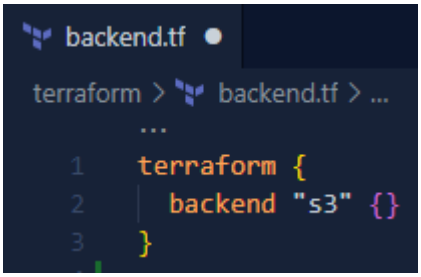
Observando a figura anterior, é possível identificar que há algumas palavras que estão laranja, elas indicam que vamos declarar um novo objeto. Quando é colocado '*resource'*, estamos dizendo para o Terraform que vamos declarar um novo recurso, em seguida ele espera que seja indicado qual o recurso existente na AWS e nome para esse recurso, já o 'data', ele vai fazer algo semelhante, mas ao invés de declara um novo recurso, ele está resgatando objetos que já estão previamente criados dentro da AWS e colocando dentro de uma variável. Por exemplo na linha 1 a 4 é declarado um recurso 'aws_s3_bucket' com o nome 'bucket', isso permite que ele possa ser referenciado por outros recursos para fazer alguma ação. Podemos ver uma referência ao recurso citado na linha 9, onde ele fala o recurso e o nome dele, em seguida pega qual o identificador desse recurso, o 'id'.

Ainda na figura 51, os recursos criados nesse Terraform serão utilizados para gerenciamento de estado da nossa infraestrutura, portanto o recurso ‘bucket’ será utilizado para guardar toda estrutura e configurações dos outros recursos, o ‘dist’ declarado na linha 6 realiza uma operação em cada objeto do ‘bucket’ de verificação de integridade. Os outros dois ‘data’ estão obtendo as roles, ou funções, já criadas dentro da AWS que serão utilizadas posteriormente para conceder permissão para nossos recursos de executar ações especificas.

4.4.2.4 Backend.tf

O backend.tf é extremamente importante, pois ele tem a responsabilidade de notificar que utilizaremos a gerência de estado e definir onde iremos armazenar todo o estado do nosso Terraform, além de realizar o bloqueio de estado para evitar que não ocorra conflito caso haja a modificação do estado ao mesmo tempo.

Figura 52: Arquivo backend.tf



Fonte: Elaborado pelo autor.

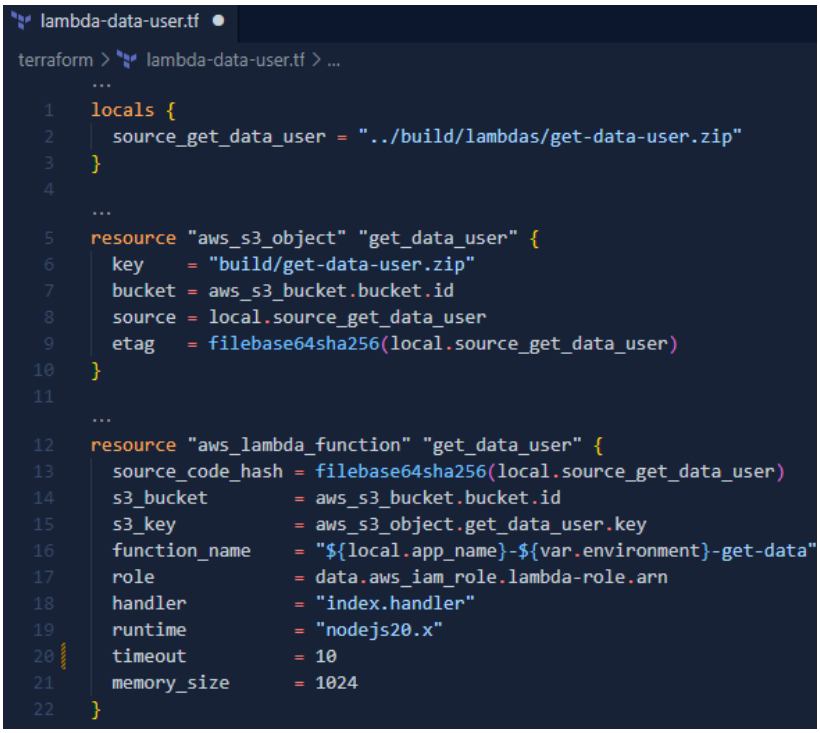
Ne caso estamos apenas indicando que iremos utilizar a gerência de estados do Terraform, isso é necessário, pois caso contrário ele não vai realizar o bloqueio de estão e nem o armazenamento. A configuração e o apontamento para qual lugar iremos armazenar os estados é indicado posteriormente dentro dos workflows do GitHub.

4.4.2.5 Lambda-data-user.tf

Essa seção será encarregada de criar a função lambda que irá realizar o *web scraping* e retornar os dados. A ideia é que ela pegue todo código fonte que foi compactado em uma pasta, armazene dentro de um s3 bucket para poder utilizar esse

código dentro da AWS, em seguida crie a lambda e ela assume todo código que foi armazenado anteriormente.

Figura 53: Arquivo lambda-data-user.tf

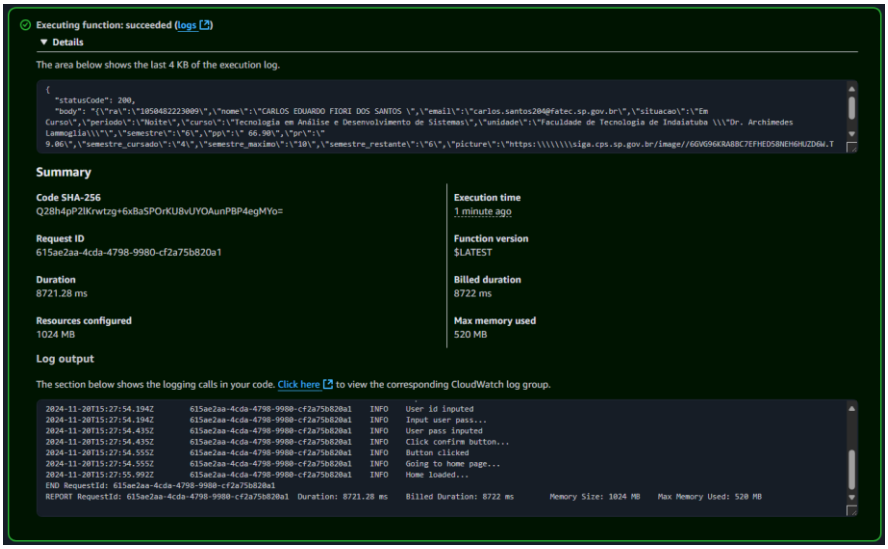


Fonte: Elaborado pelo autor.

Na figura acima podemos visualizar a criação dos recursos relacionados a lambda, primeiro é definido uma constante como ‘locals’ para apontar onde vai estar o código compactado, em seguida um recurso para acessar esse arquivo criado e por último um recurso para a criação do lambda é feita, apontando para o objeto anteriormente declarado pelos campos ‘s3_bucket’ e ‘s3_key’. Nos parâmetros passados para a lambda, podemos verificar várias referências a recursos criados anteriormente em outras lambdas, como a role que foi definida dentro do arquivo main.tf, e permite que a lambda execute e faça sua função. Outras características importantes definidas aqui é o ‘runtime’, que diz qual a versão do node que vamos rodar o nosso código, o ‘timeout’, para definir o tempo limite em segundos que esse lambda tentara executar o código e o ‘memory_size’ que atribui a quantidade de memória em megabytes que a lambda vai ter. Esses atributos foram pensados de acordo com a função que esse lambda vai ter e a versão compatível do Node com o que o código foi escrito. A tamanha da memória foi pensada com base nas bibliotecas e o que esse lambda terá de fazer, uma vez que o código em si não é grande, mas

ele precisa que um navegador seja aberto para executar o scraping e isso acaba demandando um pouco mais de memória. É importante ressaltar que o aumento da memória também impacta proporcionalmente a quantidade de CPU, o que melhora a capacidade computacional disponível (AWS, 2024)

Figura 54: Painel de log de execução do lambda na AWS



Fonte: AWS (2024)

A figura acima monstra uma execução de teste desse lambda, e nas informações da execução, podemos visualizar a quantidade de memória que essa execução utilizou. Além dessa requisição, também foi pensado na performance que essa função deveria ter, ao realizarmos um web scraping, é demandado uma alta carga de processamento, portanto é importante que a CPU seja o suficiente para manter o tempo de execução baixo.

4.4.2.6 Api-gateway.tf

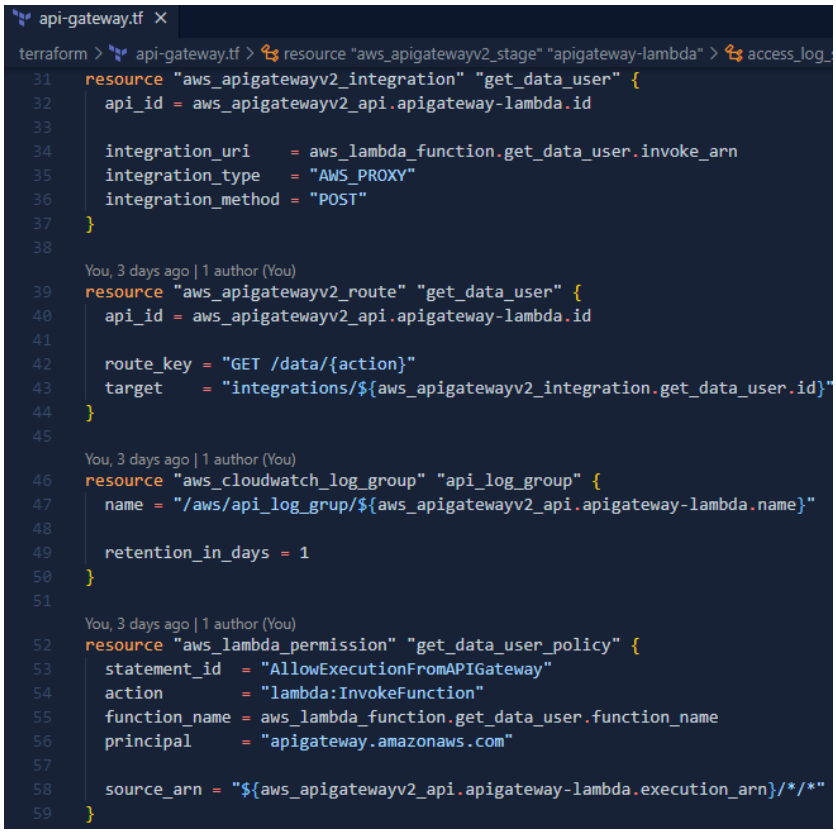
O código desse arquivo configura uma API Gateway HTTP para encaminhar solicitações para uma função Lambda usando uma integração do tipo proxy, que utiliza solicitações HTTP. Ele também habilita a geração de logs no CloudWatch para monitorar o tráfego e dá as permissões necessárias para que a API Gateway invoque a Lambda.

Figura 55: Arquivo api-gateway.tf - parte 1

Fonte: Elaborado pelo autor.

Na figura anterior é mostrado a criação do API Gateway, definindo seu nome e o tipo de protocolo que vai utilizar, no caso HTTP. Em seguida é criado um recuso dentro da lambda chamado de ‘stage’ ou ‘fase’ que serve para criarmos versões da API em operação, e dentro dela está sendo passado algumas configurações relacionadas ao nome do stage, a qual API Gateway ela se refere, ela realiza *deploy* automático para pegar as atualização feitas na API e por fim uma configuração dos logs gerados, que serão monitorados pelo CloudWatch que foi definido no ‘destination_arn’ e é feito um formato personalizado em JSON de quais informações esse monitoramento irá acessar.

Figura 56: Arquivo api-gateway.tf



Fonte: Elaborado pelo autor.

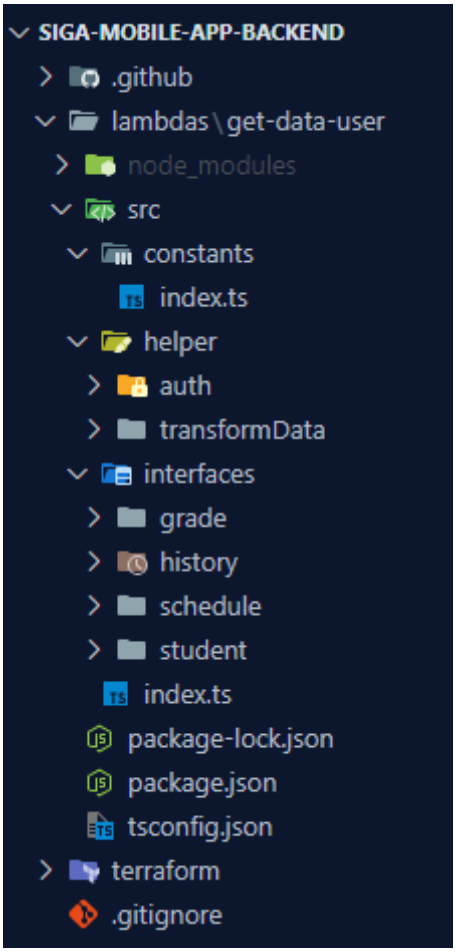
Na figura acima podemos ver o código configurando a integração entre o API Gateway e a função lambda que definimos anteriormente, por meio de uma integração ‘proxy’. Logo em seguida, é utilizado um recurso para configurar a rota em que a API irá disponibilizar para receber solicitações e enviar para nossa Lambda, estamos definindo que para a rota ‘/data/{action}’ que recebe o parâmetro por rota ‘{action}’ iremos receber solicitações do tipo ‘GET’. Dentro do lambda iremos utilizar esse parâmetro da rota para determinar quais informações iremos extrair do SIGA. Na sequência é criado um recurso do CloudWatch, definindo seu nome e quantos dias irá reter os logs registrados. Por fim no recurso final, estamos atribuindo as permissões necessárias para a API Gateway poder invocar a Lambda e passar a requisição HTTP.

4.4.3 Lambdas

Neste diretório, vamos construir o código da nossa função lambda, e para melhor organização dos arquivos, foi utilizado uma arquitetura limpa para o Node.js,

realizando uma separação por função dentro da aplicação. Podemos visualizar essa estrutura dentro da pasta ‘get-data-user’ no ‘src’, que vai dividir as funções auxiliares na qual detém de regras que ajudam na função principal, as interfaces que vão conter as entidades utilizadas nessa aplicação e vão definir seus atributos e tipos, as constantes que definem dados imutáveis e globais na aplicação e por fim as o arquivo principal que vai utilizar todas as outras funções, o ‘index.ts’. Em um escopo anterior na pasta ‘get-data-user’ temos os arquivos que vão realizar a configuração da nossa aplicação, como comentado anteriormente nas configurações gerais.

Figura 57: Diretório do back-end com a pasta lambdas e get-data-user expandidas



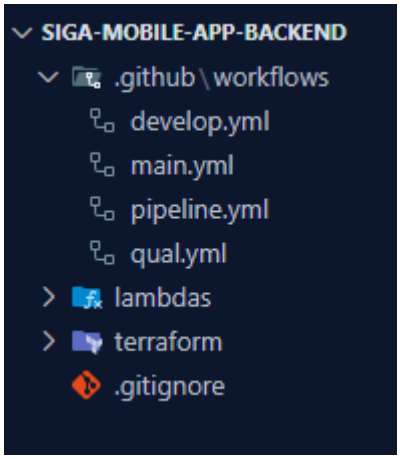
Fonte: Elaborado pelo autor.

4.4.4 GitHub Workflows

O GitHub Workflows é uma funcionalidade do GitHub Actions que permite automatizar processos de desenvolvimento, como testes, *deploys* e *builds*. Ele usa arquivos de configuração YML para definir ações que são acionadas por eventos no

repositório, como *commits* ou *pull requests*. Ele foi utilizado no *back-end* desse projeto como um uma forma de implementar a integração contínua (CI) e entrega contínua (CD), isso porque toda vez que é realizado alguma alteração no projeto e essa alteração vai para o repositório remoto, é chamado um workflow que realiza o build do projeto, garantindo que o código não está com falhas, e executando comandos do Terraform para fazer o *deploy* da alteração dentro da AWS.

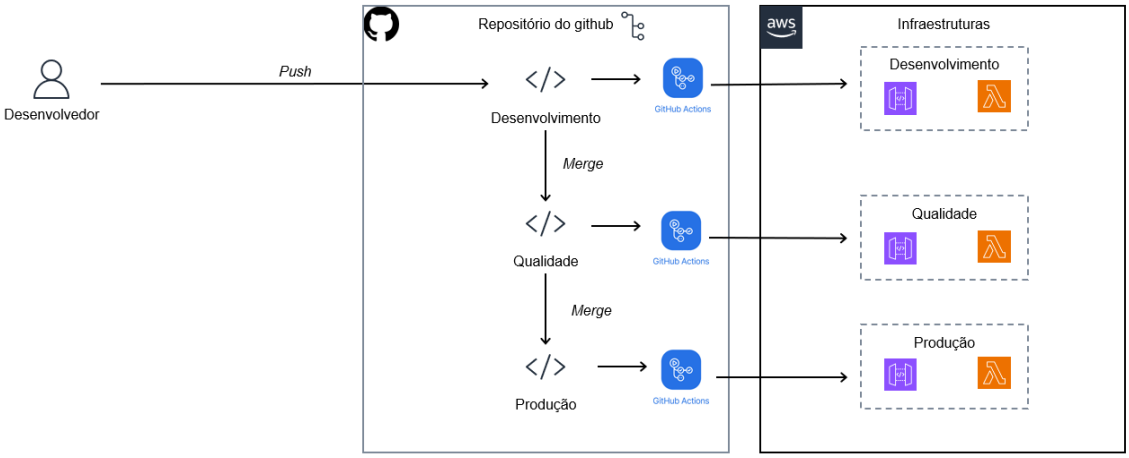
Figura 58: Diretório do back-end com as pastas .github e workflows expandida



Fonte: Elaborado pelo autor.

A figura anterior apresenta a estrutura utilizada para fazer o uso dos *actions* do GitHub, o arquivo principal que vai realizar toda verificação, build e *deploy*, será o ‘main.yml’. Porém não é ele que será chamado quando uma Branch no GitHub for alterada, e sim os outros arquivos relativos aos ambientes de qualidade (*qual*), desenvolvimento (*develop*) e produção (*main*), na qual passam parâmetros específicos para cada ambiente.

Figura 59: Diagrama do processo da execução de CI/CD no repositório do back-end



Fonte: Elaborado pelo autor

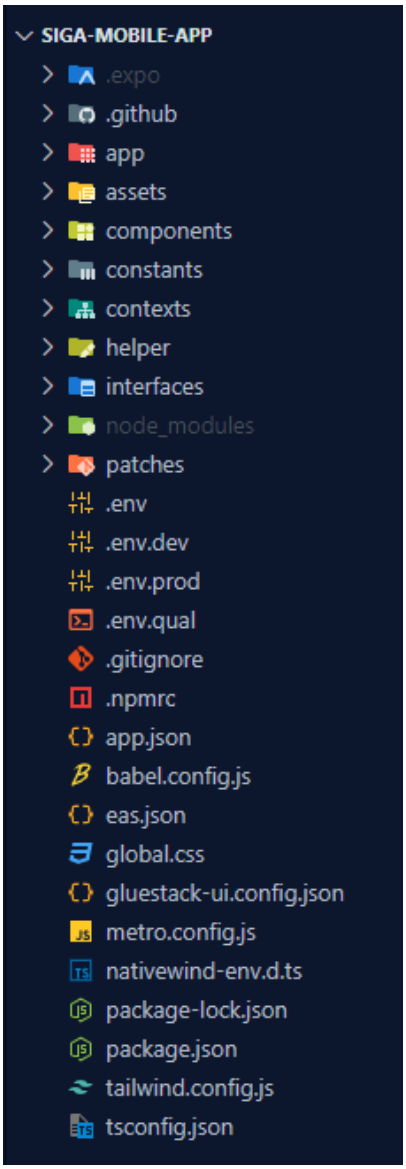
4.5 SIGA app front-end

A organização do projeto segue o padrão estrutural comentado anteriormente para facilitar o desenvolvimento e a manutenção. As pastas principais incluem app, que contém o código principal do aplicativo dividido em subcomponentes, sendo o *assets*, para recursos estáticos como imagens, e pastas como *constants*, *contexts*, *helpers*, *components* e *interfaces* para centralizar valores globais, gerenciamento de estado, funções utilitárias e componentes funcionais.

Os arquivos .env são utilizados para definir variáveis de ambiente específicas para desenvolvimento (.env.dev), produção (.env.prod) e testes de qualidade (.env.qual), garantindo flexibilidade e segurança ao isolar a URL da API.

Outros arquivos, como app.json (configuração do Expo), tsconfig.json (configuração TypeScript), e tailwind.config.js (estilização), são usados para personalizar aspectos técnicos e visuais do projeto.

Figura 60: Diretório do projeto do aplicativo do SIGA

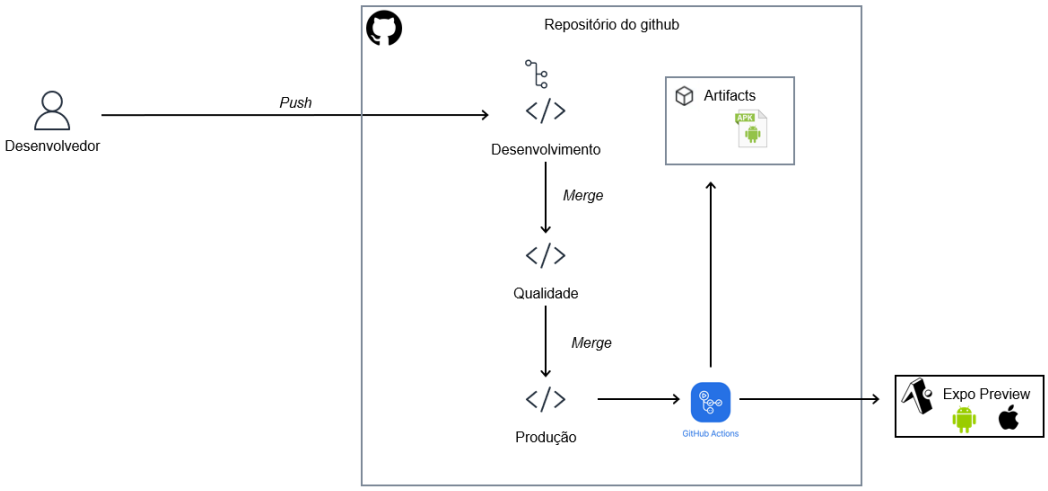


Fonte: Elaborado pelo autor.

4.5.1 GitHub Workflows

O fluxo de trabalho do GitHub nesse repositório é configurado para ser acionado apenas quando houver alterações na *branch main*, alinhando-se ao padrão do projeto de manter esta *branch* como o ambiente de produção. Esse *workflow* automatiza a geração de um APK do aplicativo e configura uma pré-visualização global no Expo. Essa configuração permite que qualquer usuário tenha acesso e utilize a versão mais atual do aplicativo diretamente, caso tenha o aplicativo do Expo no celular.

Figura 61: Diagrama do processo da execução de CI/CD no repositório do aplicativo (front-end)



Fonte: Elaborado pelo autor

4.6 Limitações e riscos

Durante o desenvolvimento do projeto foram identificados algumas limitações e possíveis riscos que podem futuramente afetar o atual funcionamento do aplicativo, tanto em questões técnicas quanto em relação a parte legal da manipulação de dados privados que o aplicativo realiza. É interessante pontuar que todas essas preocupações estão relacionadas a manipulação do ambiente externo ao aplicativo, ou seja, o uso do *web scraping*.

A forma como a implementação do *web scraping* foi realizado neste projeto utiliza-se de componentes presentes dentro da estrutura HTML, onde em um primeiro momento foi identificado como as informações são apresentadas no SIGA e mapeado quais são as principais elementos HTML que contém as informações requisitadas para serem usadas no aplicativo e então foi constatado todos os atributos ‘name’ de cada elemento para então ser usados na aplicação que realiza o *web scraping*. O problema surge quando nos sites mais atuais surgiu um conceito chamado de *single page application* (SPA) que resumidamente vai gerar atributos dinâmicos a cada acesso, o que significa que o atributo ‘name’ dos elementos anteriormente mapeados se tornam inválido a cada acesso ao site, resultando em erros e falhas no script de obtenção dos dados.

Dentro do site do SIGA, esse conceito não foi incorporado, portanto até o presente momento em que este trabalho foi desenvolvido, o *web scraping* funciona

em sua plena capacidade. Entretanto, é por esse motivo que isso é um risco, já que qualquer alteração no site que possa afetar ou modificar os atributos de cada elemento presente no SIGA, poderá interromper o funcionamento normal da aplicação.

Outro ponto importante é com relação a possível bloqueio que o servidor do SIGA possa fazer a fonte do *web scraping*, pensando que em um cenário onde diversos usuários utilizarão o aplicativo ao mesmo tempo e fazendo múltiplas requisições, o SIGA pode considerar como algum tipo de ataque para sobrecarregar seu servidor e por medidas de segurança, bloquear a origem dessas solicitações, fazendo o *web scraping* incapaz e inutilizável.

Por fim temos umas das questões mais importantes que aflige o *web scraping*, a lei geral de proteção de dados (LGPD). Embora a LGDP não bloqueie diretamente o uso do *web scraping*, ela impõe algumas exigências sobre o uso de dados pessoais, como o consentimento sobre a coleta dos dados, finalidades legitimas, apenas dados estritamente necessários ao objetivo, transparência com relação ao uso desses dados e que o responsável pelo tratamento dos dados garanta que eles sejam manipulados com medidas de segurança adequadas para protegê-los contra acessos não autorizados (GOVERNO FEDERAL, 2024). A aplicação desenvolvida confere com estes requerimentos, uma vez que é transparente qual o propósito do uso dos dados do SIGA, apenas para visualização dos estudantes da Fatec, e é evidenciado na tela inicial da aplicação que nenhum dado do usuário é salvo em qualquer outro lugar senão o próprio dispositivo móvel. Além de que para uso desse aplicativo, é necessário que o usuário aceite que seus dados sejam manipulados apenas para seu próprio benefício.

4.7 Propostas de melhoria futura

Visando entregar as melhores funcionalidades, entender a visão dos usuários e abranger o maior número de estudantes possíveis, algumas propostas serão apontadas para serem desenvolvidas futuramente dentro do aplicativo.

A primeira e uma das mais importantes é o estudo voltado para a usabilidade e experiencia do usuário que o aplicativo entrega. Tendo em vista que os testes realizados na aplicação móvel foram em sua totalidade feitos pelo autor deste trabalho, não consolidando uma base de usabilidade forte para o projeto desenvolvido. Portanto para essa proposta, em um primeiro momento o ideal seria utilizar de

técnicas e conceitos existentes para se basear e definir uma interface de usuário mais detalhada no que diz respeito a experiencia do usuário, como as 10 heurísticas de Nielsen. Elas consistem em pilares utilizados durante o desenvolvimento do design de uma interface, servindo como um tipo de guia para uma posterior avaliação do usuário (RODRÍGUEZ, 2022). Sendo assim, em um segundo momento é essencial que haja o engajamento e experiencia do usuário no uso do aplicativo, permitindo que ele seja disponibilizado para testes e por meio de formulários seja coletado as percepções e feedbacks dos usuários. Garantindo que os principais requisitos de quem irá utilizar o aplicativo, sejam atendidos e torne o uso ainda mais fluído.

Outro ponto de melhoria para futuras implementações, que agregaria um grande valor ao aplicativo e expandiria o número de usuários seria disponibilizar o app para dispositivos da Apple que utilizam o sistema operacional iOS. Essa funcionalidade não tem tanta complexidade e não requer tantas ações, levando em consideração que no framework em que o projeto foi desenvolvido (React Native) disponibiliza o suporte para ambos sistemas operacionais, tanto Android e iOS, e está habilitado para este aplicativo, entretanto algumas medidas teriam que ser tomadas para garantir a mesma experencia para o usuários de Iphone, pois algumas funcionalidades e componentes podem ter diferentes formas de serem usadas para comportar ambos sistemas, então cabe realizar testes para verificar. Não foi possível implementar com garantia nesse momento, por motivo de não ser possível realizar os testes necessário no dispositivo iOS, que, portanto, ficará para o futuro.

Além das propostas de melhorias mencionadas, também é fundamental considerar a implementação de novas funcionalidades dentro do aplicativo afim de aumentar o escopo do projeto, abordar todas os processos presentes no site do SIGA e garantir que todas as necessidades dos usuários sejam atendidas, tendo uma maior retenção no engajamento do app. Essas propostas incluem disponibilizar as funções de solicitação de documentos dentro do SIGA, tais como a revisão de notas, faltas ou exame geral, atestado de matrículas e outros que estão disponíveis dentro site. Outra melhoria com o objetivo de fornecer detalhes sobre cada disciplina no atual semestre, incluindo os detalhes sobre cada cursos, como o plano de aulas, as avaliações mais detalhadas (mostrando a fórmula e os nomes descritivos das avaliações), materiais disponibilizados pelos professores e as bibliografias.

5. CONCLUSÃO

A partir do desenvolvimento deste projeto, foi possível realizar a criação de uma aplicação móvel e um web scraping para acessar e exibir os dados presentes dentro do SIGA da Fatec, podendo assim já ser implantando para uso dos alunos. Neste processo, foram aplicados conceitos e práticas relacionados à organização de código, alinhados ao versionamento de software de forma eficiente, o que proporcionou um ambiente de desenvolvimento limpo e funcional. Isso, aliado ao uso de tecnologias modernas na nuvem, por meio da Amazon Web Services (AWS), utilizando-se dos principais serviços disponibilizados, garantiu um alto nível tecnológico e agilidade ao projeto.

A implementação de workflows automatizados no GitHub também trouxe grande qualidade ao aplicativo desenvolvido, uma vez que os fluxos de trabalho realizam testes e fazem a implementação das aplicações, facilitando futuras melhorias e expansões do projeto.

Com todas essas implementações, o projeto proporcionou uma ampla gama de conhecimentos relevantes ao autor, além de favorecer o aprofundamento no contexto acadêmico, unindo conceitos teóricos e práticos.

Portanto, os resultados obtidos contemplam os requisitos definidos no escopo do projeto que inclui suas funções básicas de visualização de dados acadêmicos e estar pronto para ser usado. Entretanto, alguns pontos de melhoria foram apontados para serem trabalhados em futuras iterações, com o objetivo de estender as funcionalidades e o escopo do projeto. Esses pontos são fundamentais, pois consideram as percepções e as reais necessidades do público-alvo do aplicativo, obtidas por meio de testes e experimentações. Busca-se, assim, proporcionar uma experiência do usuário mais fluida, intuitiva e alinhada às suas expectativas, tornando o aplicativo mais relevante.

Como sugestões para trabalhos futuros, é importante dedicar esforços à otimização do tempo de resposta do processo de web scraping, caso essa técnica continue sendo utilizada, considerando as diversas variáveis envolvidas. Métodos como a implementação de uma máquina virtual no ambiente cloud, responsável por gerenciar sessões abertas de usuários, podem ser explorados. Nesse modelo, sempre que alguém acessa o aplicativo, uma sessão é aberta e mantida por um limite de inatividade, agilizando a busca por dados.

Além disso, funcionalidades extras fora do escopo atual do site da instituição podem ser implementadas, como um sistema de gerenciamento de agenda dentro do próprio aplicativo. Essa funcionalidade garantiria maior conveniência ao usuário, que não precisaria alternar entre diferentes aplicativos no celular constantemente.

REFERÊNCIAS

AGUIAR, Camila P. PUC-Rio Mobile – Aplicativo de gerenciamento estudantil para os alunos da PUC-Rio. PUC-RJ, Rio de Janeiro, 2022.

ANSHU, Soni; VIRENDER, Ranga. API Features Individualizing of Web Services: REST and SOAP. IJTIEE, 2019.

APLICATIVO ACADESC – Gestão Escolar. Disponível em: <https://play.google.com/store/apps/details?id=acadesc.com.br.acadesc>. Acesso em: 21 maio 2024.

APLICATIVO PORTAL ALUNO UFRJ. Disponível em: <https://play.google.com/store/apps/details?id=br.ufrj.SIGA.mobile>. Acesso em: 21 maio 2024.

APLICATIVO UNICAMPUS. Disponível em: <https://play.google.com/store/apps/details?id=com.unitplus>. Acesso em: 21 maio 2024.

APLICATIVO SIGAUFMG. Disponível em: <https://play.google.com/store/apps/details?id=br.ufmg.ddp.cecom.academico.SIGAufmg>. Acesso em: 21 maio 2024.

APLICATIVO SIGAA Mobile. Disponível em: <https://play.google.com/store/apps/details?id=br.ufrn.sinfo.SIGAambeta>. Acesso em: 21 maio 2024.

APLICATIVO SUAP Mobile. Disponível em: <https://play.google.com/store/apps/details?id=br.edu.ifto.suap.mobile>. Acesso em: 21 maio 2024.

APPLE. Apple WWDC 2014. Youtube, Estados Unidos, 2014.

AWS. AWS. Disponível em: <https://aws.amazon.com/>. Acesso em: 3 nov. 2024.

BERTOLINO, Antonia; FARIA, João; LAGO, Patricia; SEMINI, Laura. Quality of Information and Communications Technology. QUATIC, Italy, 2024.

BOJINOV, Valentin. RESTful Web API Design with Node.js. 2.ed. Mumbai: Pack, 2016.

CASTANHA, Rafael G.; SANTOS, Francielle F. BRAPCI. Explorer: um novo ambiente web para análises bibliométricas a partir da Base de Dados Referencial de Artigos de Periódicos em Ciência da Informação. RDBCI, Campinas, 2023.

CONFIGURAR MEMÓRIA PARA UMA FUNÇÃO DO LAMBDA. Disponível em: <https://docs.aws.amazon.com/pt_br/lambda/latest/dg/configuration-memory.html>. Acesso em: 3 nov. 2024.

DANIELSSON, William. React Native application development: a comparison between native Android and React Native. Linköping: Linköpings Universitet, 2016.

DANTAS, Renan F. B.; DINIZ, Michely C.; GALHARDO, Cristiane X. Uso de Aplicativos Móveis Desenvolvidos por Universidades Federais como Suporte à Gestão Acadêmica. Revista de Administração, Sociedade e Inovação, Rio de Janeiro, 2023.

DIAS, Henrique; FELLER, Nadjia; MOTTA, Thiago. Desenvolvimento Colaborativo e Integrado de Sistemas: Caso de Sucesso no CPD-UFRGS. Universidade Federal do Rio Grande do Sul, Rio Grande do Sul, 2014.

FILHO, Itamir B.; AQUINO, Gibeon; ROSA, José G. S. SIGAA Mobile – O caso de sucesso da ferramenta de gestão acadêmica na era da computação móvel. UFRN, Rio Grande do Norte, 2013.

FLEURY, Maria; WERLANG, Sergio. Pesquisa aplicada: conceitos e abordagens. FGV-EASP; FGV-EPGE, 2017.

GOVERNO FEDERAL. Guia da Lei Geral de Proteção de Dados – LGPD. Disponível em: <https://www.gov.br/governodigital/pt-br/privacidade-e-seguranca/guias/guia_lgpd.pdf>. Acesso em: 3 nov. 2024.

HOWARD, Michael. Terraform — Automating infrastructure as a service. Portland State University, Portland, 2022.

IFRN. SUAP Mobile: versão para plataformas móveis é disponibilizada. Disponível em: <https://portal.ifrn.edu.br/campus/reitoria/noticias/suap-mobile-versao-para-plataformas-moveis-e-disponibilizada/>. Acesso em: 21 maio 2024.

JACOBS, Mike; CASEY, Liz; KAIM, Ed. O que é o Git?. Disponível em: <https://learn.microsoft.com/pt-br/devops/develop/git/what-is-git>. Acesso em: 3 nov. 2024.

JÚNIO, Sávio. Uma Breve Introdução à Arquitetura Limpa com Node.js. Disponível em: <https://www2.decom.ufop.br/terralab/uma-breve-introducao-a-arquitetura-limpa-com-node-js/>. Acesso em: 3 nov. 2024.

MACHADO, Francis B.; MAIA, Luiz Paulo. ARQUITETURA DE SISTEMAS OPERACIONAIS. 4.ed. Rio de Janeiro: LTC, 2007.

MANVI, Sunilkumar; GOPAL, Shyam. Resource management for infrastructure as a service (IaaS) in cloud computing: a survey. Department of Computer Science and Engineering. Reva Institute of Technology and Management, Bangalore, 2013.

MASCARENHAS, Rodrigo P. INTERFACE MOBILE PARA A PLATAFORMA SIGAA. IFSP Amazonas, Manaus, 2022.

MEIRELLES, Fernando de Souza. Uso de TI no Brasil: País tem mais de dois dispositivos digitais por habitante. FGVcia, Brasil, 2023.

MUKHERJEE, Juni. Continuous delivery pipeline 101. Atlassian, 2024.

NARENDAR, Ale. Integrating performance testing into CI/CD pipelines for test automation. Journal of Scientific and Engineering Research, 2020.

NOLL, Jones Luis; PRETTO, Fabrício. Implementação de infraestrutura como código para provisionamento e deploy de aplicações. Univates, Lajeado, 2020.

PINHEIRO, J. S. Análise do desenvolvimento de aplicativos mobile nativos e multiplataforma. UFSC, Florianópolis, 2020.

PORTO, Douglas. Mais de 92 milhões de brasileiros acessam a internet apenas pelo celular. Disponível em: <https://www.cnnbrasil.com.br/tecnologia/mais-de-92-milhoes-de-brasileiros-acessam-a-internet-apenas-pelo-celular-diz-pesquisa/#goog_rewarded>. Acesso em: 21 maio 2024.

RAJKUMAR. What is CI/CD?. Disponível em: <https://www.softwaretestingmaterial.com/what-is-ci-cd/>. Acesso em: 3 nov. 2024.

RODRÍGUEZ, Ashley. O que são Heurísticas de Nielsen e como aplicá-las em UX. Disponível em: <https://rockcontent.com/br/blog/heuristicas-de-nielsen/>. Acesso em: 25 nov. 2024.

SIGA FATEC. Disponível em: <https://SIGA.cps.sp.gov.br/aluno/login.aspx>. Acesso em: 21 maio 2024.

SIGA Mobile App. Disponível em: <https://github.com/SIGA-Mobile-Application>. Acesso em: 17 nov. 2024.

SILVA, Marcelo M.; SANTOS, Marilde T. P. Os Paradigmas de Desenvolvimento de Aplicativos para Aparelhos Celulares. UFSCAR, Brasília, 2016.

SILVA, Thiago C.; BARION, Michele C. Aplicativo para apresentação de boletim escolar com extração de dados do SUAP através do Web Scraping: estudo de caso no IFSP. IFSP, Hortolândia, 2019.

SMITH, Stephen. Test-Driven Infrastructure with Chef: Bring behavior-driven development to infrastructure as code. 2.ed. O'REILLY, 2011.

TELES, Cario M.; SILVA, Lucas A. da. Jambo: coleta de dados com Web scraping. FATEC Americana, Americana, 2021.

VADAPALLI, Sricharan. Continuous delivery, integration, and deployment with DevOps: rapid learning solution. Packt, Birmingham, 2018.

VALTANEN, Valtteri. Improving the maintainability and developer experience of Terraform code. Faculty of Information Technology and Communication Sciences, 2023.

ZAMPETTI, Fiorella; GEREMIA, Salvatori; BAVOTA, Gabriele; PENTA, Massimiliano. CI/CD Pipelines Evolution and Restructuring: A Qualitative and Quantitative Study. ICSME, Luxemburg, 2021.

ZHAO, Bo. Web scraping. Oregon State University, Estados Unidos, 2017.