RECA
REVISTA ELETRÔNICA DE COMPUTAÇÃO APLICADA
VOL. 2 NRO. 1 2021

EUVESTIBULANDO: APLICATIVO PARA GERENCIAMENTO DE DATAS DE PROCESSOS SELETIVOS

Helil Barbosa Bergária
Graduando em Engenharia de Software – Uni-FACEF
hbbergaria@gmail.com

Pedro Lima Reis
Graduando em Engenharia de Software – Uni-FACEF
pedrolimareis11@hotmail.com

Carlos Eduardo de França Roland
Mestre em Desenvolvimento Regional – Uni-FACEF
roland@facef.br

Resumo

A época de vestibulares é muitas vezes uma experiência confusa e maçante, de forma que o vestibulando, além de estudar e se preparar para as diferentes provas, precisa também se organizar para participar de todos os processos seletivos escolhidos. Esse trabalho tem como objetivo apresentar o desenvolvimento de uma aplicação para auxiliar os vestibulandos a se organizarem em relação às datas dos vestibulares. Esse projeto permitirá que o estudante dedique seu tempo a outras atividades, como estudar e descansar. Para isso, foi realizada uma revisão literária para embasar as tecnologias e ferramentas utilizadas para a análise do problema, a criação do projeto e o desenvolvimento de um aplicativo *mobile*. A partir dessa revisão, foi criada a documentação de projeto e realizada a validação necessária para a implementação do MVP do sistema. O software possui uma interface simples e intuitiva que busca as melhores maneiras de automatizar o processo de controle de datas dos eventos dos vestibulares para o usuário.

Palavras-chave: Aplicativo. Gerenciamento de datas. Vestibulando.

Abstract

The entrance exam time is often a confusing and tedious experience, so that the entrance exam, in addition to studying and preparing for the different exams, also needs to organize to participate in all the selection processes chosen. This work aims to present the development of an application to help students organize themselves in relation to the dates of the entrance exams. This project will allow the student to dedicate their time to other activities, such as studying and resting. For this, a literature review was carried out to support the technologies and tools used to analyze the problem, create the project and develop a mobile application. From this review, the project documentation was created and the necessary validation was carried out for the implementation of the system's MVP. The software has a simple and intuitive interface that seeks the best ways to automate the process of controlling the dates of university entrance exams for the user.

Keywords: App. Date Management. Examinee.

1 Introdução

Escolher um curso, uma faculdade, estudar, se organizar em relação às datas e locais de vestibulares são algumas das muitas tarefas que um vestibulando precisa estar atento. Com esse acúmulo de afazeres fica complicado para qualquer pessoa se concentrar no que realmente importa, como estudar, descansar e se preparar para as provas de forma geral. Além disso, existem outros dois agravantes: a falta de acesso à informação e a falta de incentivo.

Em virtude disso, é comum muitos perderem os prazos para a realização das inscrições e pagamentos dos vestibulares, além de se esquecerem dos dias das provas, segundas fases e até perderem a convocação da tão sonhada matrícula na universidade.

Pensando nisso, foi realizada uma pesquisa com um grupo de pessoas, alguns já cursando uma graduação e os outros ainda passando e se preparando para os processos seletivos, a fim de validar quais são as principais dificuldades enfrentadas ao decidir por um processo seletivo para ingressar em uma faculdade.

Com base neste contexto, sucedeu-se a idealização do aplicativo EuVestibulando, com intuito de resolver essas principais questões, oferecendo uma maneira mais fácil, simples e automatizada de se gerenciar os eventos como datas, processos e etapas que devem ser concluídas.

2 Fundamentação Teórica

Nesta seção são apresentadas as ferramentas e tecnologias utilizadas para o desenvolvimento do projeto EuVestibulando.

2.1 Android

Android “é uma plataforma para tecnologia móvel completa, envolvendo um pacote com programas para celulares, já com um sistema operacional, *middleware*, aplicativos e interface do usuário" (PEREIRA, 2009, p. 3). Então, ao contrário do que muitos acreditam, o Android não é apenas um sistema operacional, mas sim uma plataforma com um conjunto de outros programas que se complementam.

De acordo com Lima (2017, p. 100) o “Android teve o seu início no ano de 2003 com o foco voltado para as câmeras digitais, porém, devido à falta de oportunidade, esse sistema mudou para o mercado de telefonia móvel. Esta plataforma teve como base o sistema operacional Linux”.

2.2 Desenvolvimento Mobile Híbrido

Will (2017, p. 14) ressalta que o “desenvolvimento de aplicações móveis se tornou uma nova oportunidade de negócio para os desenvolvedores e para as empresas, tendo em vista o crescimento exponencial do número de *smartphones* no mercado”. Assim sendo, ter um aplicativo *mobile* na atualidade se tornou uma necessidade, obrigando as empresas a acompanharem esse novo segmento de mercado para oferecer uma melhor experiência ao usuário e conseguir competir diretamente com outras empresas do ramo.

Sobre aplicações *mobile* híbridas, Prezotto e Boniati (2014, p. 73) dizem que "são aplicações que possuem como finalidade funcionar em qualquer que seja o dispositivo, sendo que para as diferentes plataformas, será utilizado o mesmo código-fonte". Com isso, o desenvolvimento de aplicativos híbridos se torna mais simples e rápido quando comparados ao desenvolvimento nativo de cada plataforma, tornando-se assim uma opção vantajosa e possivelmente mais barata para investir, principalmente considerando a manutenibilidade do projeto, que será feita em apenas um código-fonte para múltiplas plataformas.

2.3 React Native

O React Native é um *framework* criado pelo Facebook para desenvolvimento de aplicações híbridas que renderiza componentes nativos através de um código em Javascript, permitindo um desenvolvimento único para iOS e Android. Combinando soluções nativas com a biblioteca React, o *framework* disponibiliza funcionalidades que agilizam a construção da aplicação, como o *Fast Refresh*, que permite visualizar as mudanças em tempo real (FACEBOOK INC., 2021).

> Diferente de outras alternativas para aplicações mobile que utilizam *webviews* para renderizar a aplicação, o React Native invoca as APIs de renderização nativas em Objective-C, para iOS, ou Java para Android. Portanto, a aplicação será renderizada utilizando componentes nativos da plataforma (EISENMAN, 2016 *apud* RAQUEL, 2017, p. 21).

Com isso, as aplicações feitas com o *framework* tendem a ser mais fluidas e visualmente parecidas com as nativas do que quando feitas com outras ferramentas.

2.4 React

O React é uma biblioteca criada pelo Facebook para criação de interfaces de usuário. Baseada em componentes e códigos declarativos, ela utiliza o Javascript em uma sintaxe parecida com o XML, denominada JSX, que posteriormente é convertida de volta para Javascript utilizando-se o Babel, um compilador da linguagem (FACEBOOK INC., 2021).

De acordo com Signoretti (2018, p. 23):

> Aplicações construídas com esse framework são baseadas em componentes isolados e reutilizáveis que, quando combinados, podem formar interfaces de usuário mais complexas. Todos os elementos que são renderizados na tela são componentes, como botões, campos de texto, listas, etc.

Algumas grandes empresas que utilizam esse *framework* são: Americanas, Boticário, C6 Bank, iFood, Mercado Livre, Nubank, OLX, LuizaLabs e XP Investimentos (FERREIRA, 2021).

2.5 Node

Segundo Pereira (2014), o node é um *runtime environment* (ambiente de execução) de códigos, escrita em C++ e em JavaScript que são executados na *engine* V8 do Google Chrome. Essa linguagem foi desenvolvida a fim de evitar problemas de paralisação do servidor quando determinada tarefa está sendo executada, o que outras linguagens como PHP, Java, Python têm conhecido como *Blocking-Thread*.

Um dos principais recursos do Node é a utilização de um estilo de programação orientado a eventos (*event-driven*) ou assíncrono. Nesse estilo, os eventos são manipulados por funções (*callbacks*), que são chamadas quando algo a qual ela foi vinculada acontecer. Dessa forma, o processo que estiver sendo executado não irá bloquear outros processos, já que todos podem ocorrer paralelamente. Além disso, esse estilo de programação no Node é acompanhado de um *event loop*, um padrão que realiza duas principais ações em uma repetição contínua: detecção de eventos e chamada dos manipuladores vinculados ao evento. Esse *event loop* roda em apenas uma *thread* dentro de um único processo, o que tira a necessidade de estratégias para trabalhar com concorrência entre *threads* e compartilhamento de estado (TEIXEIRA, 2012).

2.6 Javascript

Criado em 1995, o Javascript é uma linguagem de programação dinâmica e multi-paradigmática, suportando os padrões de orientação a objetos, imperativo e funcional. É interpretada ou compilada pelo *JIT* (*just-in-time)*, um programa que transforma o código em *bytecode*, facilitando a execução pelo processador. Embora a linguagem seja muito conhecida como uma linguagem web, é também utilizada em ambientes sem navegador, como por exemplo com o Node.js, Apache CouchDB e Adobe Acrobat (MOZILLA *et al.*, 2017 *apud* TRAÇA, 2018).

> A sintaxe do Javascript é muito semelhante a C, C++ e Java [...]. Além disso, a instanciação de objetos com valores pré-inicializados em Javascript [...] é muito mais conveniente do que em C++ ou na linguagem de programação Java (MIKKONEN, 2007, p. 7, tradução nossa).

2.7 PostgreSQL

O PostgreSQL é um sistema de gerenciamento de banco de dados objeto-relacional extremamente poderoso, gratuito e de código aberto que teve origem em um projeto da Universidade da Califórnia em 1986 denominado POSTGRES. Ele roda em todos os principais sistemas operacionais e conta com muitos recursos como: diversos tipos de dados além dos primitivos; integridade e consistência dos dados; concorrência e alta performance; confiabilidade e recuperação de dados em desastres; segurança robusta; extensibilidade; e internacionalização, contando inclusive com ferramentas para busca de texto. Essas e muitas outras características, fizeram com que o PostgreSQL ganhasse uma forte reputação e fosse adotado por grandes empresas (THE POSTGRESQL GLOBAL DEVELOPMENT GROUP, 2021).

Segundo Ferreira e Junior (2012, p. 4):

> O PostgreSQL segue a padronização SQL [...], uma linguagem de interface para SGBD. Possui uma ferramenta texto chamada psql, para receber os

> comandos SQL e uma interface gráfica chamada PGAdmin para tornar mais fácil e intuitivo o uso do SGBD. O sistema possibilita também a manipulação dos dados armazenados através de camadas ODBC e JDBC ou com acesso nativo na linguagem C. [...]

2.8 Typescript

O Typescript, que foi criado pela Microsoft em 2012, é uma linguagem de código aberto que se baseia em Javascript, uma das ferramentas mais usadas no mundo. Essa linguagem tem a finalidade de trazer mais segurança e praticidade ao se desenvolver um código. A adoção do Typescript pode ser feita gradualmente, migrando apenas parte do código e com o tempo preparar para a conversão completa (MICROSOFT, 2021).

"O TypeScript é uma linguagem transpilada para JavaScript. O programador precisa ter em mente que em tempo de execução a linguagem que estará sendo interpretada será o JavaScript" (LOPES, p. 11).

2.9 Python

Segundo Borges (2014, p.15), Python "foi criada em 1990 por Guido van Rossum, no Instituto Nacional de Pesquisa para Matemática e Ciência da Computação da Holanda (CWI)". Apesar de ser uma linguagem relativamente nova, ela se popularizou muito nos últimos anos.

Em conformidade com Menezes (2010), Python é uma linguagem muito poderosa, podendo ser usada para administrar e desenvolver grandes sistemas, além de ser conhecida por sua facilidade em ser aprendida e utilizada. Por conta disso, o Python se tornou muito querido entre os desenvolvedores seniores e iniciantes.

3 Contexto Empreendedor do Projeto

O termo *entrepreneurship*, também conhecido como empreendedorismo, tem como fundamento desenvolver ideias a fim de consolidar pessoas e processos com o intuito de transformá-las em oportunidades de negócios bem-sucedidos (VALENCIANO SENTANIN e BARBOZA, 2005, p. 9).

3.1 Business Model Canvas

Um modelo de negócios, ou *business model*, é um importante aliado ao se desenvolver uma ideia de sucesso. De acordo com Osterwalder e Pigneur (2011, p. 15), "Modelo de Negócios é um esquema para a estratégia ser implementada através das estruturas organizacionais dos processos e sistemas". O *Business Model Canvas*, ou Quadro de Modelo de Negócios, é uma ferramenta para se organizar em 9 diferentes grupos, os pontos chave de um negócio.

Dado o modelo criado para o EuVestibulando, apresentado na Figura 1, segue o detalhamento de cada um dos grupos abordados, apontando um resumo e como se encontram no projeto.

Figura 1 — Modelo de *Business Model Canvas* do projeto

Fonte: os autores

1. Parceiros chave: são as parcerias essenciais para a efetivação da ideia. Nosso principal foco de parceria são: em instituições de ensino, como escolas e cursinhos, por terem contato direto com o vestibulando; faculdades particulares, pois muitas se mostram interessadas em parcerias para atrair novos alunos; e fundações responsáveis pela organização de vestibulares, já que elas poderiam fornecer informações diretamente para alimentação do nosso sistema;
2. Atividades chave: são as principais atividades para concretizar a proposta de valor. Em nosso caso, o grande atrativo do sistema é justamente o controle de datas e lembretes através de notificações *push*, já que isso permitirá uma maior liberdade de tempo para o vestibulando;
3. Recurso chave: são os recursos necessários para entregar os outros elementos descritos. Nesse projeto, consideramos essencial a organização (do projeto e da equipe) alinhada com o uso de tecnologia;
4. Proposta de valor: é o diferencial da empresa, aquilo que ela irá disponibilizar de valor para seus clientes. Possuímos uma premissa bem simples: desejamos disponibilizar um aplicativo gratuito que auxilie os vestibulandos na organização em relação às datas de vestibulares, de forma que eles possam gastar seu tempo com estudos, descanso, lazer e outras atividades que considerem importantes;
5. Relação com o cliente: é a maneira de se relacionar com os diferentes segmentos de clientes. Para esse projeto, planejamos oferecer um suporte via e-mail e através do próprio aplicativo, com um FAQ e uma interface intuitiva;
6. Canais: são os canais de comunicação que levam a proposta de valor até o cliente. O aplicativo *mobile* vai ser nosso único canal para isso, já que queremos simplificar o processo ao máximo para o usuário final;

7. Segmentos de mercado: são os segmentos de clientes, aqueles que serão o principal foco da empresa. O foco do EuVestibulando são os vestibulandos de qualquer idade, ou seja, todos aqueles que estão em busca de ingressar em uma faculdade;
8. Estrutura de custos: são os custos para a execução do modelo. Fora os custos de publicação nas lojas de aplicativos, precisamos manter uma equipe para oferecer suporte e uma para realizar a parte funcional;
9. Fontes de renda: são as maneiras de criar um retorno financeiro através da proposta de valor. Nosso projeto irá depender essencialmente das parcerias para isso, seja em forma de publicidade interna ou de algum contrato fechado para beneficiar ambos os lados.

4 Metodologia e análise

Para levantamento de detalhes das necessidades dos estudantes em época de vestibular e de requisitos para o aplicativo, foram realizadas entrevistas. A entrevista semiestruturada aborda o entrevistado de uma maneira distinta da entrevista estruturada: a princípio são realizadas indagações que interessam à pesquisa e após a resposta é possível abrir um amplo campo de perguntas à medida que vão sendo respondidas. Assim o entrevistado ajuda na elaboração do roteiro da pesquisa (TRIVIÑOS, 1987, p. 146).

Já que o foco desse projeto foi oferecer uma solução para os vestibulandos, buscou-se entrevistar pessoas dessa classe, procurando entender os principais problemas que eles têm em relação à sua organização para participação em processos seletivos, quais as fontes de consulta mais utilizam, quais os métodos e ferramentas usados nesses momentos, etc. Com base nessa visão foram construídas as seguintes perguntas que guiaram as entrevistas:

1. Quais vestibulares ou universidades você teve ou tem interesse?
2. Em relação a esses vestibulares, você possui alguma dúvida, dificuldade ou impedimento para a realização?
3. Você encontra dificuldade para saber quais vestibulares prestar?
4. Qual universidade ou curso escolher, qual campus, etc.?
5. Você teve/tem dificuldade em lembrar as datas dos processos seletivos?
6. Você já perdeu algum processo por esquecimento?
7. Como você se organiza perante as datas e eventuais mudanças do processo seletivo (considerando todas as etapas, como inscrição, provas, etc.)?
8. Quais fontes você consulta para saber das notícias referentes aos processos seletivos?
9. Que tipo de ferramenta você gostaria de utilizar para te acompanhar durante o processo seletivo?
10. Caso você pudesse criar um aplicativo para te auxiliar no processo de organização referente aos diferentes vestibulares prestados, quais funcionalidades você considera essenciais?
11. Como você realiza exercícios para testar seus conhecimentos?
12. Como você mantém sua rotina de estudos?
13. Possui alguma dificuldade?
14. Existe algo que gostaria de acrescentar sobre o assunto? Alguma dúvida, curiosidade, dificuldade, sugestão?

As entrevistas ocorreram de forma individual utilizando da plataforma Zoom, com duração entre 40 a 50 minutos, sendo que 5 pessoas participaram das entrevistas com as seguintes características:

1. quatro mulheres e um homem;
2. uma cursando o 3° ano do ensino médio, uma fazendo cursinho, uma recém ingressa na faculdade e duas na faculdade há mais tempo.

4.1 Personas

De acordo com Grudin e Pruitt (2002, p. 146), personas são pessoas fictícias, com nomes, idade, sexo, educação, desempenho, enfim, qualquer característica que seja relevante no dado contexto, possuindo suas próprias histórias, objetivos e tarefas. A principal vantagem de se utilizar personas é o agrupamento de diferentes características de pessoas reais, a fim de criar alguns personagens que as representem, permitindo inclusive o anonimato em entrevistas, como é o caso do EuVestibulando.

Com base nas entrevistas realizadas foram elaboradas 3 personas, apresentadas nas Tabelas 1, 2 e 3 realizando um agrupamento das principais características levantadas.

Essas personas sucederam na criação do sistema, desde as documentações até as prototipações e implementação, que são abordadas nas próximas seções.

Tabela 1 — Persona Jane

Jane — Está no 3º ano do ensino médio de uma escola particular	
Idade	18 anos
Sexo	Feminino
Objetivo	Decidir um curso e ingressar na faculdade
Universidades de interesse	- UniFACEF - Unesp - UniFRAN
Características	- Indecisa - Desorganizada - Acompanha as atualizações dos vestibulares pelas redes sociais das faculdades
Reclamações	- Possui dificuldade em se organizar perante as provas e atividades da escola ao mesmo tempo que estuda para os vestibulares - Acredita que as informações de vestibulares dadas pelas escolas são pouco acessíveis - Sente falta de orientação profissional

Fonte: os autores

Tabela 2 — Persona Brian

Brian — Está no 3º ano do ensino médio de uma escola pública	
Idade	17 anos
Sexo	Masculino
Objetivo	Conseguir ingressar direto após a conclusão do ensino médio em um curso de engenharia
Universidades de interesse	- ITA - Ufscar - USP - UniFACEF
Características	- Realiza testes online de conhecimento - Acompanha vídeo aulas no youtube
Reclamações	- Dificuldade de lembrar as datas dos vestibulares que irá prestar - Não possui tanto apoio da escola em relação aos estudos, portanto precisa estudar por conta em casa através da internet

Fonte: os autores

Tabela 3 — Persona Mary

Mary — Presta cursinho faz 3 anos	
Idade	21 anos
Sexo	Feminino
Objetivo	Ingressar numa faculdade pública de medicina
Universidades de interesse	- USP - UFSCAR - Unicamp - Unesp
Características	- Possui uma boa organização para estudos - Realiza semanalmente as provas antigas dos vestibulares que está prestando
Reclamações	- Por prestar muitos vestibulares, fica difícil acompanhar as notícias e datas de todos - Cursinhos preparatórios não fornecem todas as informações necessárias para a realização das provas - Devido ao longo período de estudos diário, fica exausta

Fonte: os autores

5 Engenharia de Software

Essa seção compreende todos os artefatos desenvolvidos em relação à documentação da Engenharia de Software que foram utilizadas no projeto. Segundo Pressman e Maxim (2016, p. 14) "é preciso fazer um esforço conjunto para compreender o problema antes de desenvolver uma solução de software". Portanto, entende-se essa etapa como essencial para atingir o objetivo do projeto e torná-lo prático e usável. Os seguintes artefatos encontram-se detalhados nessa seção: o *Business Process Model and Notation* (BPMN), o Diagrama de Caso de Uso, o Modelo Entidade-Relacionamento (MER) e o Diagrama Entidade-Relacionamento (DER) do aplicativo EuVestibulando.

5.1 BPMN

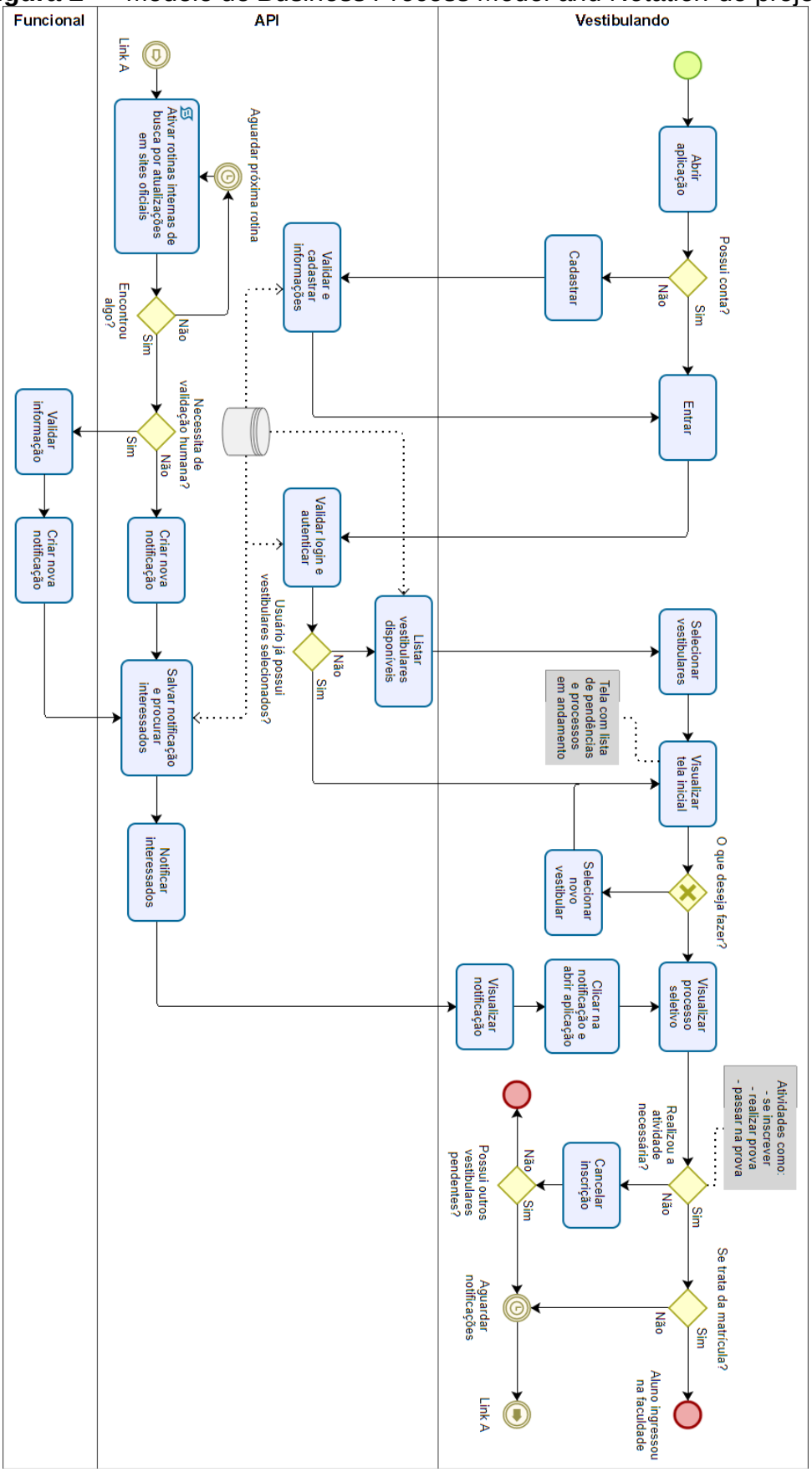
Como aponta Pizza (2012, p. 25), o *Business Process Modeling and Notation* (Notação de Modelagem de Processos de Negócio) é uma metodologia de modelagem de processos através da utilização de elementos gráficos padronizados que facilitam a compreensão, descoberta e mapeamento de processos, sejam eles de um sistema ou não, além também de permitir se repensar os fluxos a fim de otimiza-los.

O diagrama BPMN apresentado na Figura 2 começa com o usuário abrindo a aplicação. Após isso, há duas opções: caso ele não tenha uma conta na aplicação, ele realiza seu cadastro no sistema e então é redirecionado para o processo de *login*; já quando o usuário possui cadastro, ele realiza o *login*.

Em seguida é feita uma busca no banco de dados para verificar se o usuário já tem algum vestibular marcado como de interesse. Caso tenha, ele é redirecionado para a tela inicial; caso contrário são listados os vestibulares disponíveis para que ele possa selecionar aqueles que tenha interesse.

A partir desse ponto o usuário pode realizar diversas funções, como selecionar outros vestibulares, visualizar os processos seletivos, realizar as atividades de cada etapa, cancelar a inscrição ou aguardar por novas notificações dos vestibulares que está inscrito.

Figura 2 — Modelo de *Business Process Model and Notation* do projeto



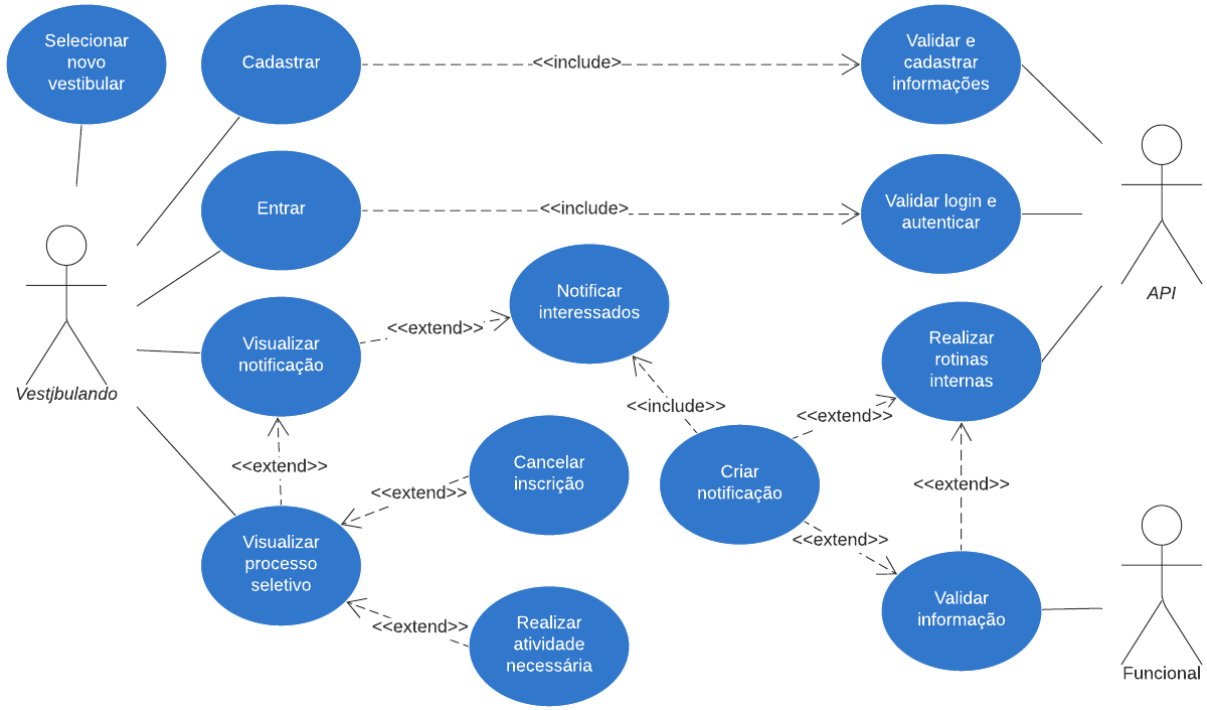
Fonte: os autores

5.2 Diagrama de Caso de Uso

O Diagrama de Caso de Uso é uma ferramenta importante para realizar o desenvolvimento de um sistema. Regis, Tavares e Leite (2015, p. 2) afirmam que “esse tipo de diagrama permite que o usuário tenha uma visão clara do que será o sistema a ser construído, detalhando todo o processo de execução”. Assim sendo, o desenvolvimento desse diagrama é fundamental para o entendimento dos processos e funcionalidades do aplicativo.

O Diagrama de Caso de Uso apresentado na Figura 3 tem o intuito de mostrar e resumir as funcionalidades, atividades e interações sistêmicas entre os atores, que são o Vestibulando, o usuário final, e o Funcional que caracteriza um membro da equipe interna do projeto.

Figura 3 — Diagrama de Caso de Uso



Fonte: os autores

5.3 Modelo Entidade-Relacionamento

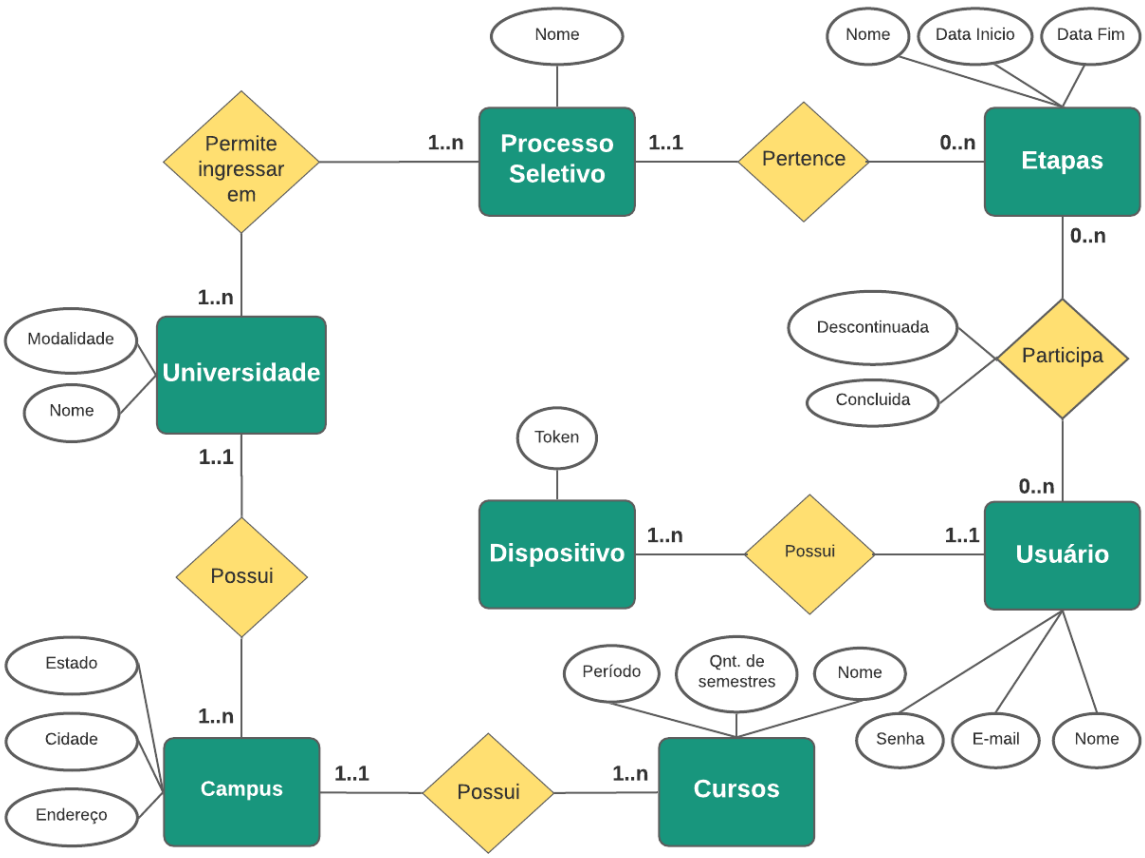
De acordo com Takai, Italiano e Ferreira (2005, p. 22), “o MER é um modelo de dados conceitual de alto-nível, ou seja, seus conceitos foram projetados para serem compreensíveis a usuários, descartando detalhes de como os dados são armazenados”. Dessa forma, esse modelo visa apenas representar as entidades com seus relacionamentos e atributos, sem levar em conta a implementação ou especificidade de um sistema de gerenciamento de banco de dados.

Para operação do aplicativo, foram definidas as entidades Cursos, Campus e Universidade que servem para estruturar as informações vinculadas à entidade Processo Seletivo que serão apresentadas na aplicação. A entidade Usuário exerce a função de manter os dados do Vestibulando, enquanto a Dispositivo gerencia os *tokens* para envio de notificações *push*.

Por fim, a entidade Processo Seletivo foi definida para agrupar, através de seus atributos e relações, todos os dados referentes aos vestibulares, estabelecendo a relação com Etapas para especificar cada passo dos processos de interesse do Usuário, que se liga ao Processo Seletivo através da relação Participa, a qual permite armazenar se o usuário concluiu a etapa ou se o processo foi descontinuado.

Na Figura 4, são apresentadas as entidades que compõem o projeto EuVestibulando, assim como seus relacionamentos e atributos.

Figura 4 — Diagrama Entidade-Relacionamento



Fonte: os autores

5.4 Demais documentações

Além dos diagramas apresentados, foram criadas outros artefatos para execução do projeto como: Levantamento e Documentação de Requisitos para descrever todas as funcionalidades do sistema; matriz *SWOT* para planejamento estratégico da solução; Mitigação de Riscos para identificar os riscos e agir de maneira preventiva em relação a eles; Estrutura Analítica do Projeto (EAP) para divisão do escopo do projeto de maneira hierárquica; Modelo lógico do banco de dados para criação do banco de dados da aplicação; e Plano 5W1H, para traçar o planejamento de forma mais abrangente.

Todos esses instrumentos foram fundamentais para o desenvolvimento de um produto consistente e podem ser visualizados na íntegra através do Github (BERGÁRIA E REIS, 2021a).

6 Implementação

Essa seção mostra os detalhes mais significativos da implementação do EuVestibulando, apresentando a prototipação de telas, a implementação *mobile* em React Native, o desenvolvimento da API em Node e o monitor de alterações em Python.

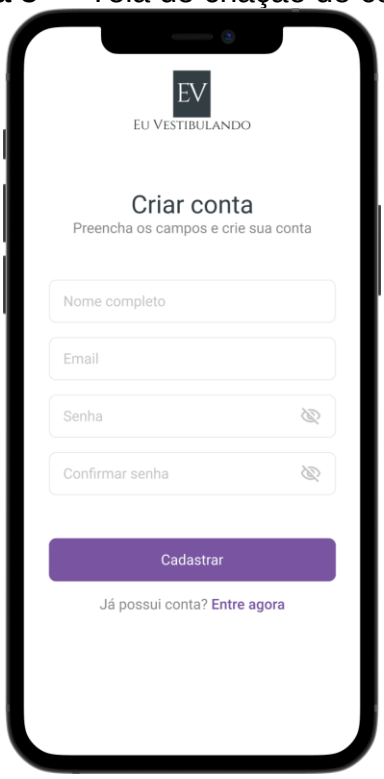
6.1 Telas do aplicativo

Nesta subseção são apresentadas as telas prototipadas para o *Minimum Viable Product* (MVP) do EuVestibulando, assim como a descrição de cada uma. Todas as telas foram concebidas com o foco em simplicidade e usabilidade.

6.1.1 Criar conta e iniciar sessão

Para o cadastro do usuário (Figura 5) são dispostos 4 campos na mesma tela: nome completo, e-mail, senha e confirmação de senha, de forma que o usuário consiga criar uma conta rapidamente. Após criar a conta, o usuário deve ser capaz de acessar o aplicativo pela tela de iniciar sessão (Figura 6).

Figura 5 — Tela de criação de conta



Fonte: os autores

Figura 6 — Tela para *login*

Fonte: os autores

6.1.2 Telas iniciais

Ao acessar a aplicação pela primeira vez, o usuário é direcionado para a tela de escolha de vestibulares (Figura 7) onde deverá ser capaz de selecionar os vestibulares que tem interesse dentre aqueles que o sistema disponibiliza. Após isso

(ou em caso de segundo acesso ou subsequentes), o usuário receberá a tela principal da aplicação (Figura 8) na qual verá um resumo dos processos em que está inscrito e poderá concluir etapas ou cancelar inscrição dos processos pendentes.

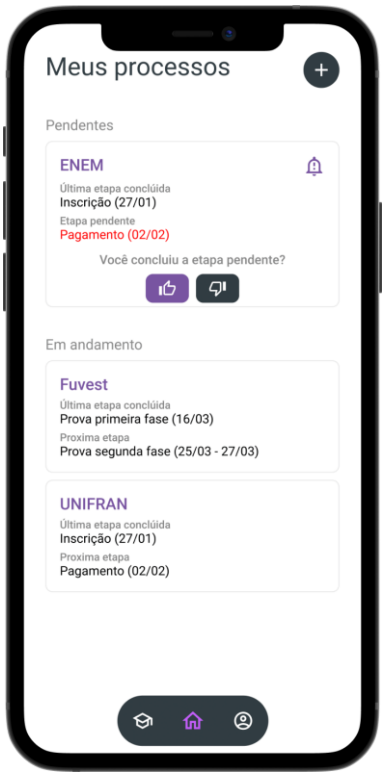
6.1.3 Detalhes do processo seletivo

Caso o usuário clique em algum processo seletivo, ele será encaminhado para a tela de detalhamento (Figura 9) podendo visualizar toda a linha do tempo das etapas até a data atual, ainda tendo a opção de concluir a etapa ou cancelar a inscrição quando houver etapa pendente.

Figura 7 — Tela de primeiro acesso

Fonte: os autores

Figura 8 — Tela inicial



Fonte: os autores

Figura 9 — Tela de detalhes do processo

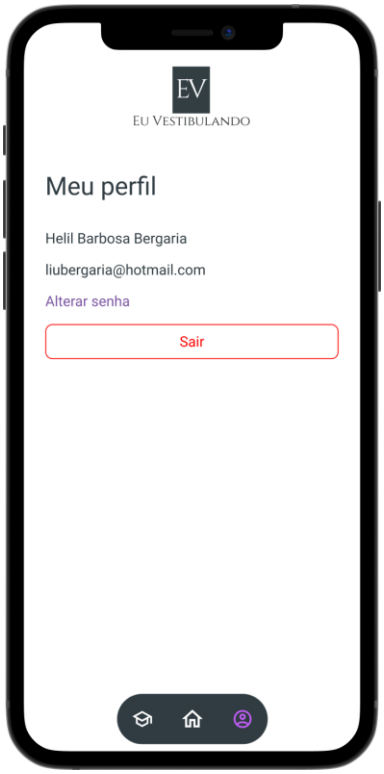
Fonte: os autores

6.1.4 Meu perfil e outros processos seletivos

Ao clicar no ícone de usuário (terceiro da esquerda para a direita) na barra de navegação inferior, é apresentada a tela de meu perfil (Figura 10) onde o vestibulando pode visualizar suas principais informações e finalizar a sessão através do botão sair.

Já ao clicar no ícone de vestibulares (primeiro da esquerda para a direita), é apresentada a tela de processos seletivos (Figura 11) permitindo visualizar os processos abertos para inscrição e os que o usuário está inscrito atualmente.

Figura 10 — Tela de perfil do usuário



Fonte: os autores

Figura 11 — Tela de listagem processos seletivos

Fonte: os autores

6.2 Implementação mobile

Para a criação do aplicativo foi utilizado o React Native com Typescript, devido a agilidade e praticidade proporcionadas para criação de uma aplicação híbrida, ou seja, que funciona para Android e iOS através do mesmo código-fonte. Em relação ao Typescript, a Microsoft (2021) define como “Javascript com sintaxe para tipagem”. Sua escolha teve como base a segurança gerada pela tipagem, que deixa os processos de codificação mais rápidos, evitando erros comuns, mas ainda mantendo funcionalidades importantes para a estrutura e performance do aplicativo, como por exemplo o assincronismo e o paradigma funcional de programação herdado do Javascript.

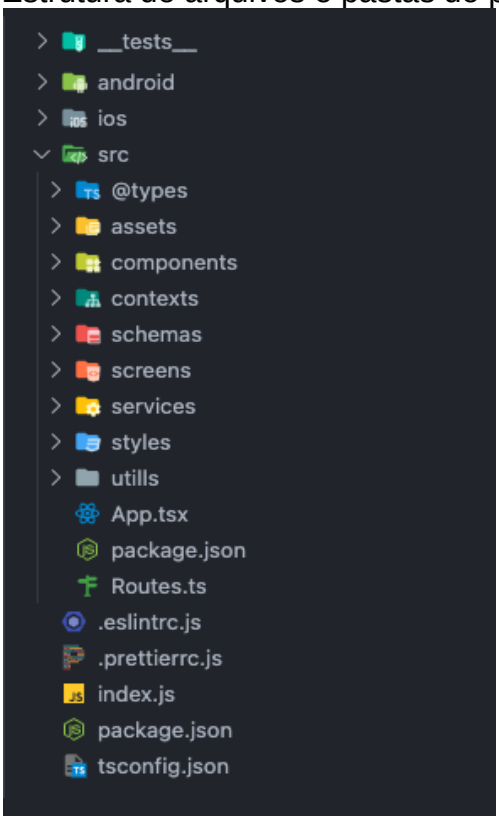
A estruturação do projeto, teve foco em conceitos como componentização e reutilização, além de manter o princípio da responsabilidade única para cada funcionalidade e arquivo com os elementos descritos a seguir e demonstrados pela Figura 12.

Elementos da estrutura do aplicativo:

1. __test__: Arquivos de testes automatizados de toda aplicação;
2. android: Projeto android nativo da aplicação;
3. ios: Projeto iOS nativo da aplicação;
4. src: Todos os arquivos de código-fonte para React Native;
    a. @types: Arquivos de tipagens globais;
    b. assets: Recursos do projeto, como imagens, vídeos ou logos;
    c. components: Todos os componentes que foram criados através da extração de pedaços de códigos reutilizáveis;

   d. contexts: Contextos para disponibilização de dados, como os de autenticação, entre toda a aplicação;
   e. schemas: Esquemas para validação de dados de diferentes formulários;
   f. screens: Conjunto de diferentes telas e seções de navegação da aplicação;
   g. services: Serviços responsáveis por buscar dados através da API;
   h. styles: Estilos globais do sistema;
   i. utils: Funções utilitárias reutilizadas entre o projeto;
   j. App.tsx: Componente principal e inicial da aplicação em React Native;
   k. Routes.ts: Arquivos para tipagens e enumeração das diferentes rotas de navegação;
5. package.json: Arquivo para listagem de dependências do projeto;
6. .eslintrc.js e .prettierrc.js: Arquivo de configuração do *linter* utilizado para manter consistência na programação;
7. index.js: Arquivo raiz da aplicação;
8. tsconfig.json: Arquivo de configuração do Typescript;

Figura 12 — Estrutura de arquivos e pastas do projeto *mobile*



Fonte: os autores

O código-fonte desse MVP *mobile* pode ser encontrado no Github (BERGARIA E REIS, 2021b).

6.2.1 Componentização

Como apresentado, o código foi todo escrito procurando sempre a reutilização de partes. Dessa forma, uma das estruturas mais comuns encontradas no projeto são os componentes, tendo por exemplo o componente de um botão, como demonstrado na Figura 13. A estrutura e sintaxe de código de componentes é praticamente a mesma vista ao se utilizar React, uma vez que o React Native roda sobre a lógica do React, alterando apenas os elementos, que ao invés de serem *tags* HTML, são componentes importados do próprio *framework*. Um fator importante para a estabilidade de utilização desses componentes é a tipagem de suas *props*, como visto na interface *IProps*.

Figura 13 — Código de componente *Button*

```
interface IProps {
  color?: ColorValue;
  containerStyle?: StyleProp<ViewStyle>;
  disabled?: boolean;
  onPress?: (event: GestureResponderEvent) => void;
  textColor?: ColorValue;
  textStyle?: StyleProp<TextStyle>;
  title: string | React.ReactNode;
}

const Button = ({
  title,
  color = Colors.primaryPurple,
  textColor = Colors.white,
  containerStyle,
  textStyle,
  onPress,
  disabled,
}: IProps): JSX.Element => {
  return (
    <TouchableWithoutFeedback disabled={disabled} onPress={onPress}>
      <View
        style={[styles.container, containerStyle, { backgroundColor: color }]}
      >
        <Text style={[textStyle, { color: textColor }]}>{title}</Text>
      </View>
    </TouchableWithoutFeedback>
  );
};
```

Fonte: os autores

6.2.2 Testes unitários de componentes

A fim de garantir uma segurança e invariabilidade dos componentes ao longo de futuras manutenções, foram desenvolvidos testes unitários de cada componente presente. Na Figura 14 realiza-se um teste de inalterabilidade de um *layout* através de *snapshots*, um tipo de teste muito comum ao se trabalhar com aspectos de *UI*. Já na Figura 15 e Figura 16, executa-se um teste de funcionalidade, verificando se o botão é capaz de ser clicado múltiplas vezes e de não ser clicado quando estiver desabilitado.

Figura 14 — Código de teste automatizado de renderização

```
it('renders correctly', () => {
  const tree = renderer.create(<Button title={'Test button'} />);

  expect(tree).toMatchSnapshot();
});
```

Fonte: os autores

Figura 15 — Código de teste automatizado de clique no botão

```
it('button click', () => {
  const mockClick = jest.fn();

  const title = 'Test button';

  const { getByText } = render(<Button title={title} onPress={mockClick} />);

  fireEvent.press(getByText(title));
  fireEvent.press(getByText(title));

  expect(mockClick.mock.results.length).toBe(2);
});
```

Fonte: os autores

Figura 16 — Código de teste automatizado de clique no botão desabilitado

```
it('disabled button click', () => {
  const mockClick = jest.fn();

  const title = 'Test button';

  const { getByText } = render(
    <Button disabled={true} title={title} onPress={mockClick} />,
  );

  fireEvent.press(getByText(title));
  fireEvent.press(getByText(title));

  expect(mockClick.mock.results.length).toBe(0);
});
```

Fonte: os autores

6.3 Desenvolvimento da API

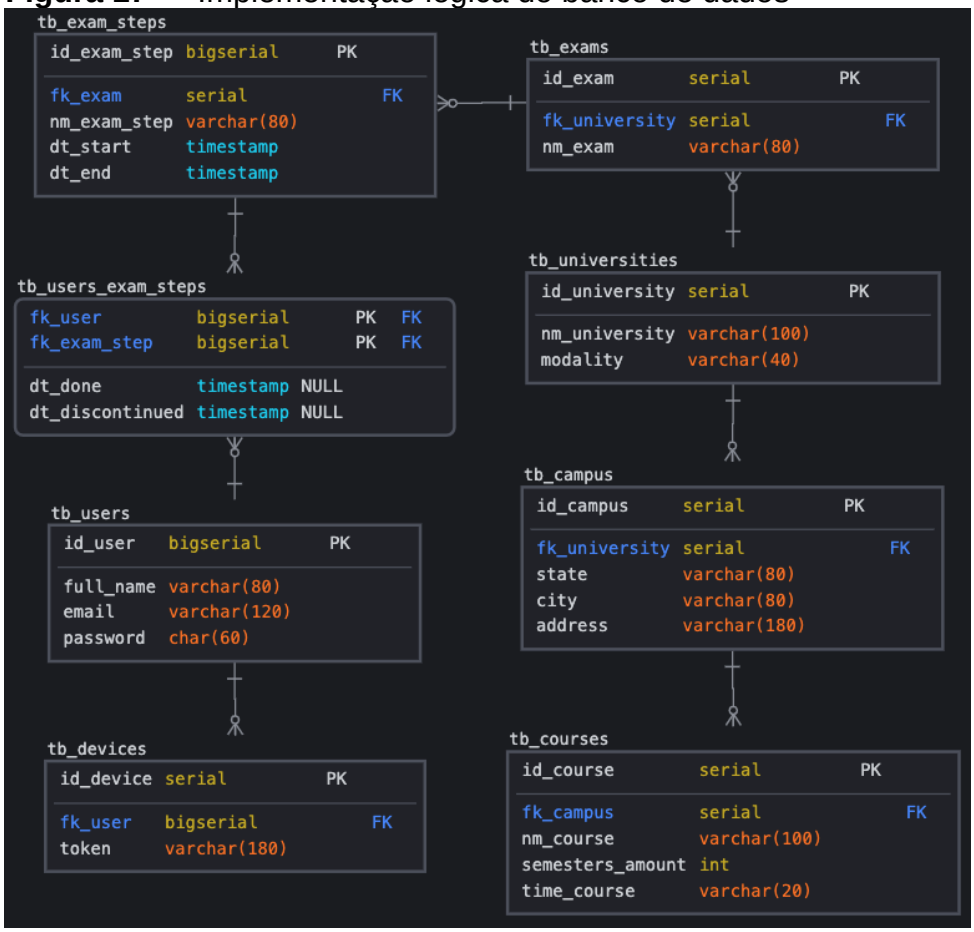
Em relação a *Application Programming Interface* (*API*), que é onde fica a maior parte das regras de negócio, armazenando e tratando todos os dados dos vestibulandos e seus processos seletivos, sucedeu o uso de um modelo MVP, bastante comum na criação de *backends*. O Node com Typescript foi a escolha de tecnologia para a criação deste módulo, pois além de permitir agilidade e simplicidade no desenvolvimento, ele possibilita uma reutilização de código entre *mobile* e *backend*, uma vez que a base das duas tecnologias é o Node com Typescript.

A codificação completa dessa *API* pode ser acessada no Github (BERGARIA E REIS, 2021c).

6.3.1 Banco de dados

O PostgreSQL foi escolhido como Sistema de Gerenciamento de Banco de Dados (SGBD), devido ao fato de ser *open source*, gratuito e altamente performático. Toda a implementação lógica e física do banco de dados foi executada tendo o modelo entidade-relacionamento apresentado anteriormente como base. A implementação lógica do banco é mostrada na Figura 17.

Figura 17 — Implementação lógica do banco de dados



Fonte: os autores

6.4 Monitor de alterações

Parte do projeto consiste na criação de um robô que irá monitorar certas páginas da internet e gerar uma notificação quando elas sofrerem alterações. Assim, foi desenvolvido um programa nomeado EVMonitor em Python, uma vez que essa linguagem oferece maiores facilidades para extrair informações de sites da internet.

Para entender o funcionamento do monitor, é importante entender o significado de uma função *hash*: “As funções hash comprimem uma cadeia de caracteres de comprimento arbitrário em uma de comprimento fixo.” (DEEPAKUMARA, DEEPAKUMARA~~HEYS~~, e VENKATESAN, 2001, tradução nossa). Com isso, foi utilizado esse conceito para diminuir o tamanho de conteúdos e agilizar as comparações, sendo esse processo mais detalhado a seguir.

Parte desta rotina pode ser visualizada na Figura 18, enquanto o código completo com todos os métodos se encontra no Github (BERGARIA E REIS, 2021d).

Figura 18 — Código de implementação do EVMonitor

```
if len(sys.argv) != 2:
    print('You should pass a link to monitor')
    sys.exit()

link =  sys.argv[1]
response = performRequest(link)
currentHash = makeHash(response)

print("\n[EVMonitor] Executing routine for", link, "\n")
time.sleep(60)

while True:
    try:
        response = performRequest(link)
        newHash = makeHash(response)

        if newHash == currentHash:
            print("[EVMonitor] No updates found on", link)
        else:
            print("\n[EVMonitor] Something change on", link, "-- notifying the team\n")
            currentHash = newHash

            notifyChanges(link)

        time.sleep(60)

    except Exception as e:
        print("[EVMonitor] An error occurred")
        print(e)
```

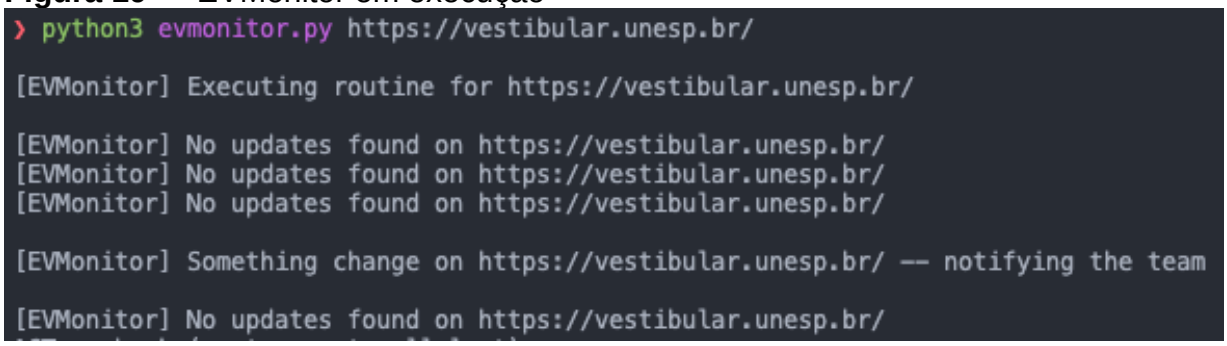
Fonte: os autores

O programa implementa um algoritmo de fluxo simples: ele recebe um *link* a observar como parâmetro da rotina; realiza uma busca pelo conteúdo retornado da página; cria um *hash* desse conteúdo; espera 60 segundos; e por fim entra em uma repetição, onde a cada 60 segundos, ele irá fazer uma nova requisição da página, criar um *hash* do conteúdo da página e verificar se esse *hash* é o mesmo do feito na busca anterior. Caso seja igual, significa que não houve alteração na página; caso seja diferente, o programa notifica as mudanças e salva o novo *hash* como o mais recente.

Optou-se por utilizar essa estratégia de criar um *hash* do conteúdo, pois isso agiliza muito as comparações, uma vez que um *hash* possui um tamanho muito menor do que uma página completa, além de evitar grande uso de memória pelo mesmo motivo.

Na Figura 19 é possível visualizar o programa em execução, utilizando a URL “https://vestibular.unesp.br/” como exemplo. Percebe-se que depois de 3 tentativas sem encontrar novas atualizações, o programa encontrou alguma alteração na 4ª vez e gerou a notificação. Após isso, o programa continuou o processo com o novo *hash*, não notificando a mesma mudança duas vezes.

Figura 19 — EVMonitor em execução



Fonte: os autores

7 Considerações finais

O objetivo do desenvolvimento do MVP do projeto, mostrado no decorrer do artigo, foi concluído utilizando-se de técnicas de Metodologia Ágil com diferentes artefatos da Engenharia de Software, que foram desenvolvidos e aplicados em cada etapa da solução. Com isso, os vestibulandos poderão fazer o acompanhamento de datas, provas, segundas chamadas e convocações dos seus processos seletivos com mais segurança, sabendo que o aplicativo irá lhes auxiliar nesse gerenciamento.

Ao decorrer da execução do projeto, foram observadas melhorias e novas funcionalidades que serão implementadas em futuras versões do aplicativo, como o desenvolvimento de um módulo *web* para administração do sistema, e soluções mais automatizadas para todo o fluxo funcional. O intuito é sempre entregar uma experiência segura, simples e confiável a todos os usuários, permitindo que eles se preocupem com outras tarefas de maior importância.

Referências

BERGÁRIA, Helil Barbosa; REIS, Pedro Lima. **Repositório de demais documentações construídas para o projeto**. 2021a. Disponível em: <https://github.com/EuVestibulando/documents>. Acesso em: 17 out. 2021.

BERGÁRIA, Helil Barbosa; REIS, Pedro Lima. **Repositório do código-fonte do aplicativo mobile**. 2021b. Disponível em: <https://github.com/EuVestibulando/mobile-app>. Acesso em: 17 out. 2021.

BERGÁRIA, Helil Barbosa; REIS, Pedro Lima. **Repositório do código-fonte da API**. 2021c. Disponível em: <https://github.com/EuVestibulando/api>. Acesso em: 17 out. 2021.

BERGÁRIA, Helil Barbosa; REIS, Pedro Lima. **Repositório do código-fonte do monitor EVMonitor**. 2021d. Disponível em: <https://github.com/EuVestibulando/evmonitor>. Acesso em 17 out. 2021.

BORGES, Luiz Eduardo. **Python para desenvolvedores: aborda Python 3.3**. Novatec Editora, 2014.

DEEPAKUMARA, Janaka; HEYS, Howard M.; VENKATESAN, R. **FPGA implementation of MD5 hash algorithm.** In: Canadian Conference on Electrical and Computer Engineering 2001. Conference Proceedings (Cat. No. 01TH8555). IEEE, 2001. p. 919-924.

FERREIRA, Erick Rodrigues; JÚNIOR, Sergio M. Trad. **Análise de desempenho de bancos de dados**. Universidade Presidente Antônio Carlos, Barbacena, 2012.

FERREIRA, Gabriel. **Repositório que mostra empresas e projetos que utilizam React no Brasil**. 2021. Disponível em: <https://github.com/react-brasil/empresas-que-usam-react-no-brasil>. Acesso em: 21 mar. de 2021.

FILHO, Wilson de Pádua Paula. **Engenharia de Software: fundamentos, métodos e padrões**. 2000. Disponível em: <http://aulasprof.6te.net/Arquivos_Aulas/07-Proces_Desen_Soft/Livro_Eng_Soft_Fund_Met_Padroes.pdf>. Acesso em: 7 set. de 2021.

GRUDIN, Jonathan; PRUITT, John. **Personas, Participatory Design and Product Development: An Infrastructure for Engagement**. 2002. Disponível em: <http://ojs.ruc.dk/index.php/pdc/article/viewFile/249/241>. Acesso em: 7 set. de 2021.

LIMA, Welton Dias. **Android e a influência do sistema operacional Linux**. Tecnologias em Projeção, v. 8, n. 1, p. 100-111, 2017.

LOPES, Jean de Oliveira. **PHP ou TypeScript: uma comparação de duas linguagens para web pelas suas características.** 2017 .Disponível em <http://atom.poa.ifrs.edu.br/uploads/r/biblioteca-clovis-vergara-marques-4/a/6/4/a64df95ef48f25c7704fee48b159f27d21a5d279cdd30f890671ed16c8c6421d/Jean_de_Oliveira_Lopes.pdf>. Acesso em: 15 abr. de 2021.

MENEZES, Nilo Ney Coutinho. **Introdução à Programação com Python.** 2010. Disponível em: <https://s3.novatec.com.br/capitulos/capitulo-9788575222508.pdf>. Acesso em: 10 out. de 2021

MICROSOFT. **TypeScript: Typed JavaScript at Any Scale**. 2021 .Disponível em: <https://www.typescriptlang.org>. Acesso em: 10 abr. de 2021.

MIKKONEN, Tommi; TAIVALSAARI, Antero. **Using JavaScript as a real programming language**. 2007.

OSTERWALDER, Alexander; PIGNEUR, Yves. **Business Model Generation: Inovação em Modelos de Negócios**. Starlin Alta Editora. 2011. Acesso em: 5 set. de 2021.

PEREIRA, Caio Ribeiro. **Aplicações web real-time com Node.js**. São Paulo: Casa do Código, 2014. p. 37.

PEREIRA, Lucio Camilo Oliva; DA SILVA, Michel Lourenço. **Android para desenvolvedores.** Brasport, 2009.

PIZZA, William Roque. **A metodologia Business Process Management (BPM) e sua importância para as organizações**. 2012. Disponível em: <http://www.fatecsp.br/dti/tcc/tcc00084.pdf>. Acesso em: 12 jul. de 2021

PRESSMAN, Roger S.; MAXIM, Bruce R. **Engenharia de software uma abordagem profissional**. 8º Edição. AMGH Editora Ltda. 2016. Acesso em: 7 set. de 2021.

PREZOTTO, Ezequiel. **Estudo de frameworks multiplataforma para desenvolvimento de aplicações mobile híbridas**. 2014. Disponível em: < https://docplayer.com.br/16198154-Estudo-de-frameworks-multiplataforma-para-desenvolvimento-de-aplicacoes-mobile-hibridas.html>. Acesso em: 5 fev. de 2021.

RAQUEL, Lucas Gomes. **Dita Ofertas: uma aplicação para reproduzir ofertas de produtos no rádio do carro**. 2017. Disponível em: <http://dsc.inf.furb.br/arquivos/tccs/monografias/2017_1_lucas-gomes_monografia.pdf>. Acesso em: 5 fev. de 2021.

REACT. **React - A JavaScript library for building user interfaces**. 2021. Disponível em: <https://reactjs.org>. Acesso em: 12 mar. de 2021.

REACT NATIVE. **React Native - Learn once, write anywhere**. 2021. Disponível em: <https://reactnative.dev>. Acesso em: 12 mar. de 2021.

REGIS, Luiz Felipe Muniz; TAVARES, Cássio Jardim; LEITE, Frederico do Carmo; SILVA, Wilton Ribeiro. **Diagramas de caso de uso para criação de um software de custo de produção de uma lavoura**. 2015. Disponível em: <https://ifgoiano.edu.br/ceic/anais/files/papers/20682.pdf>. Acesso em: 10 set. de 2021.

SIGNORETTI, Gabriel Lucas Albuquerque Maia. **Visualização de dados em um contexto de veículos inteligentes**. 2018. Disponível em: <https://monografias.ufrn.br/jspui/bitstream/123456789/8125/1/visualizacaodedadosemumcontexto_TCC.pdf>. Acesso em: 5 fev. de 2021.

TAKAI, Osvaldo Kotaro; ITALIANO, Isabel Cristina; FERREIRA, João Eduardo. **Introdução a Banco de Dados**. 2005. Disponível em: <https://www.ime.usp.br/~jef/apostila.pdf>. Acesso em: 21 set. de 2021.

TEIXEIRA, Pedro. **Professional Node.js: building Javascript based scalable software**. John Wiley & Sons, 2012.

THE POSTGRESQL GLOBAL DEVELOPMENT GROUP. **PostgreSQL: About.** Disponível em: <https://www.postgresql.org/about/>. Acesso em: 10 mar. de 2021.

TRAÇA, Tiago Ribeiro. **Ferramenta de software para auxiliar a promoção da saúde em ambientes universitários**. 2018. Disponível em: <http://riut.utfpr.edu.br/jspui/bitstream/1/10761/1/DV_COENS_2018_2_10.pdf>. Acesso em: 10 mar. de 2021.

TRIVIÑOS, Augusto Nibaldo Silva. **Introdução à pesquisa em ciências sociais**. Editora Atlas S.A. 1987. Acesso em: 14 set. de 2021.

VALENCIANO SENTANIN, Luis Henrique; BARBOZA, Reginaldo José. **Conceitos de empreendedorismo**. 2005. Disponível em: <http://faef.revista.inf.br/imagens_arquivos/arquivos_destaque/CvfACUcZOtmMWBx_2013-4-26-12-25-36.pdf>. Acesso em: 14 set. de 2021.

WILL, Jean Carlos. **Aplicativo mobile para busca de restaurantes**. 2017. Disponível em: <http://repositorio.roca.utfpr.edu.br/jspui/bitstream/1/8166/1/PB_COADS_2017_1_05.pdf>. Acesso em: 5 fev. de 2021.