DAS Departamento de Automação e Sistemas
CTC Centro Tecnológico
UFSC Universidade Federal de Santa Catarina

Desenvolvimento de um SaaS para imobiliarias

Relatório submetido à Universidade Federal de Santa Catarina
como requisito para a aprovação da disciplina:
DAS 5511: Projeto de Fim de Curso

Michelle Moreira de Araújo

Florianópolis, Junho de 2019

Desenvolvimento de um SaaS para imobiliarias

Michelle Moreira de Araújo

Esta monografia foi julgada no contexto da disciplina

DAS 5511: Projeto de Fim de Curso

e aprovada na sua forma final pelo

Curso de Engenharia de Controle e Automação

Prof. Romulo Silva de Oliveira

Banca Examinadora:

Rodrigo Holz Schüler / Jungle Devs
Orientador na Empresa

Prof. Romulo Silva de Oliveira
Orientador no Curso

Prof. Hector Bessa Silveira
Responsável pela disciplina

Cleber Jorge Amaral, Avaliador

Igor Althoff Vidal, Debatedor

Miguel Budag Becker, Debatedor

Agradecimentos

Gostaria de agradecer à Jungle Devs, por me dar a oportunidade de fazer parte da empresa e de trabalhar no projeto Woliver.

Agradecimentos também à Antonio Adalberto Duarte Júnior e Andrio Frizon por me acompanharem em todo o processo.

A Vinicius de Morais Justo por ser o mentor na empresa, bem como Augusto Lazzarotto de Lima, Felipe de Alencar Pinheiro e Rodrigo Holz Schüler, por auxiliar nos momentos de dúvida e ajudando em todo o processo de aprendizagem dentro da empresa.

A todos os demais membros da equipe pelo convívio diário, auxílio e troca de experiências.

E meus pais pelo apoio incondicional dado a todo e qualquer projeto em que me envolvesse.

Resumo

Este documento é referente ao período de estágio e desenvolvimento do Projeto de Fim de Curso na empresa Jungle Devs. Será relacionada a problemática do projeto e as atividades desenvolvidas com o contexto do curso de Engenharia de Controle e Automação. Será apresentada, brevemente, a empresa Jungle Devs e a metodologia empregada para desenvolvimento de projetos dentro da empresa. Objetiva apresentar o funcionamento e contribuições à um Software as a Service (SaaS) para imobiliárias. O produto permite a busca de imóveis para locação, bem como o agendamento de visita à estes imóveis. A plataforma possui integração com APIs de terceiros, como Google Maps, Google Tag Manager, Google Analytics, Here Maps, AccountKit, BLiP Chat, entre outras. Serão listadas as tecnologias utilizadas e implementações, bem com os resultados obtidos. Serão apresentadas as atividades desenvolvidas dentro do projeto, bem como listadas e explicadas as ferramentas e linguagens de programação utilizadas para o desenvolvimento da plataforma. Serão apresentados os resultados obtidos durante o período citado, e será feita uma análise crítica dos resultados, avaliando o que foi feito.

HTML, CSS, JavaScript, ReactJS, Software, Desenvolvimento Web, Integração de Sistemas, Front-End, White-Label, SaaS, Software como Serviço.

Abstract

This document refers to the internship period and development of the End of Course Project at the company Jungle Devs. It will be presented the problematic of the project and the activities accomplished in the context of the course Control and Automation Engineering. It will present the company Jungle Devs, its Academy program and the methodology used in the development of projects inside the company. It aims to present the functioning and contributions to a software as a service (SaaS) for real estate agencies. The service allows the user to search for rental properties, as well as allowing the user to schedule a visit to those properties. The product integrates APIs such as Google Maps, Google Tag Manager, Google Analytics, Here Maps, Accountkit, BLiP Chat, and others. The document will present the technologies used, implementations and results. It will present the activities performed in the projects, as well as list and explain the tools and programming languages used. It will present the results obtained during the internship period, and a critical analysis of the results and what was accomplished.

Keywords: HTML, CSS, JavaScript, ReactJS, Software, Web Development, API, Front-end, White-Label, SaaS, Software as a Service.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

SaaS - Software as a Service

SSR - Server Side Rendering

CSR - Client Side Rendering

Sumário

1 Introdução

O objetivo deste primeiro capítulo é contextualizar o escopo do trabalho realizado no período de desenvolvimento do Projeto de Fim de Curso na empresa Jungle Devs, apresentando as problemáticas envolvidas as quais as atividades realizadas procuram solucionar, assim como a motivação e justificativa para escolha do tema.

1.1 Localização do assunto no contexto do curso

O escopo das atividades realizadas na empresa, relacionadas ao desenvolvimento de software, está fortemente ligado à área de informática, que é uma das três principais áreas oferecidas no curso.

Para realizar as tarefas e desenvolver a plataforma, utilizou-se conhecimentos relacionados à técnicas de programação, que foram adquiridos em disciplinas como "Introdução à Informática para Automação- (DAS5334), “Fundamentos da Estrutura da Informação” – (DAS5102), “Informática Industrial” – (DAS5305) e “Metodologia para Desenvolvimento de Sistemas” – (DAS5312).

Além destas, foram, também, utilizados alguns conceitos desenvolvidos em disciplinas optativas, como alguns conceitos de “Fundamentos de Sistemas de Banco de Dados” – (INE5213) e “Integração de Sistemas Corporativos” – (DAS5941).

1.2 Motivação

A demanda por soluções tecnológicas para a área imobiliária vem crescendo nos últimos anos, com os dispositivos móveis ganhando destaque nas ferramentas mais utilizadas. Tendo isto em vista, cada vez se faz mais necessário inovar e gerar meios mais rápidos e simples para solucionar problemas, e diminuir burocracias na hora de alugar um imóvel.

Com a chegada de imobiliárias digitais no Brasil [1], imobiliárias tradicionais precisam se reinventar e inovar para continuarem relevantes no mercado.

Com isto em mente, este trabalho de PFC propõe o desenvolvimento de uma plataforma web para imobiliárias, na forma de um Software como Serviço (*Software as a Service*, SaaS). A plataforma foi ao ar temporariamente na imobiliária Brognoli para validação, e, atualmente, está no ar como uma plataforma genérica customizável (*White-label*, seção 3.6) para a imobiliária Terraz, ainda em processo de desenvolvimento.

A plataforma tem como objetivo facilitar o processo de locação de imóveis residenciais, aprimorando o sistema de busca por imóveis através de filtros e facilitando o

agendamento de visita à imóveis. A plataforma também tem com objetivo simplificar e automatizar o processo de contratos, entretanto esta parte não estará contemplada neste documento, tendo em vista que ainda está em fase de implementação. Será melhor comentada na seção de perspectivas futuras, ao final do documento.

A plataforma também exerce a responsabilidade de integrar os sistemas (APIs de terceiros) que compõe os requisitos da aplicação, realizando também a integração com o back-end e com um banco de dados com configurações das imobiliárias, utilizado no SaaS.

1.3 Objetivos

O objetivo geral deste PFC foi o desenvolvimento de uma plataforma web para imobiliárias.

Este documento irá focar no desenvolvimento da parte de *front-end* da plataforma, juntamente com a integração com o *back-end* e os serviços de terceiros utilizados pela mesma.

1.4 Estrutura do Documento

Este documento está estruturado da seguinte maneira:

No capítulo 2, é apresentada a empresa em que o trabalho foi realizado, bem como o processo de desenvolvimento (metodologia de projeto) adotado tanto pela empresa quanto para a realização deste projeto, bem como o programa de estágios (chamado Academy, seção 2.2) dentro da empresa.

No capítulo 3, são apresentados os conceitos básicos utilizados para a formalização do projeto. São listadas e explicadas as linguagens de programação utilizadas no projeto, bem como algumas bibliotecas fundamentais para o desenvolvimento da mesma. São apresentados conceitos utilizados ao longo do trabalho, bem como algumas ferramentas utilizadas para o desenvolvimento do mesmo.

No capítulo 4, apresenta-se a problemática do assunto, bem como a proposta da plataforma. São levantados os requisitos funcionais para a plataforma.

Já no capítulo 5, é apresentado o desenvolvimento do trabalho. É neste capítulo que está explicitado tudo o que foi feito e que é pertinente ao desenvolvimento deste PFC, como as atividades desenvolvidas, apresentados as diferentes *views* e componentes que compõe a aplicação, bem como as fases de desenvolvimento, entre a criação da plataforma para uma única imobiliária, transformação da mesma em uma *White-label*, transformação da plataforma de Client Side Rendering para Server Side Rendering (seções 3.8 e 3.9), e, por fim, a transformação da plataforma em um SaaS.

No capítulo 6 serão apresentados os resultados obtidos neste projeto, analisando o cumprimento dos requisitos funcionais levantados no capítulo 4, bem como uma análise de utilização da plataforma.

No capítulo 7, serão apresentadas as conclusões do projeto, e serão levantadas perspectivas futuras para o mesmo. Também, será abordado neste capítulo o que ainda está atualmente em desenvolvimento na aplicação, mas que não foi contemplado neste documento.

2 A empresa

O objetivo desta seção é fazer uma breve introdução à empresa Jungle Devs apresentando um pouco dos processos da empresa e metodologias utilizadas. O programa Academy também será apresentado nesta seção.

A Jungle Devs é uma empresa dedicada ao desenvolvimento de produtos de software com foco em tecnologias Web e Mobile, incluindo as áreas de *design*, front-end, *back-end* e *DevOps*.

A empresa foi oficialmente fundada em fevereiro de 2018, por ex-alunos dos curso de Engenharia de Controle e Automação da UFSC, com o objetivo de se consolidar na área de desenvolvimento de software e criar o programa Academy. A empresa é sediada em Florianópolis e tem como missão mudar a forma como as pessoas aprendem desenvolvimento de software e dar condições para que alunos de graduação trabalhem em empresas no Brasil e no mundo.

A motivação para abrir a empresa veio da necessidade de novos profissionais no setor de tecnologia. Então, tendo em vista a capacitação de pessoas ao fornecer experiência prática em projetos, o foco da empresa é no desenvolvimento de pessoas.

Atualmente, a Jungle Devs conta com parcerias em empresas no Brasil, Estados Unidos, África do Sul, Austrália e Polônia.

Figura 1 – Logo da Jungle Devs

2.1 Processo de Desenvolvimento

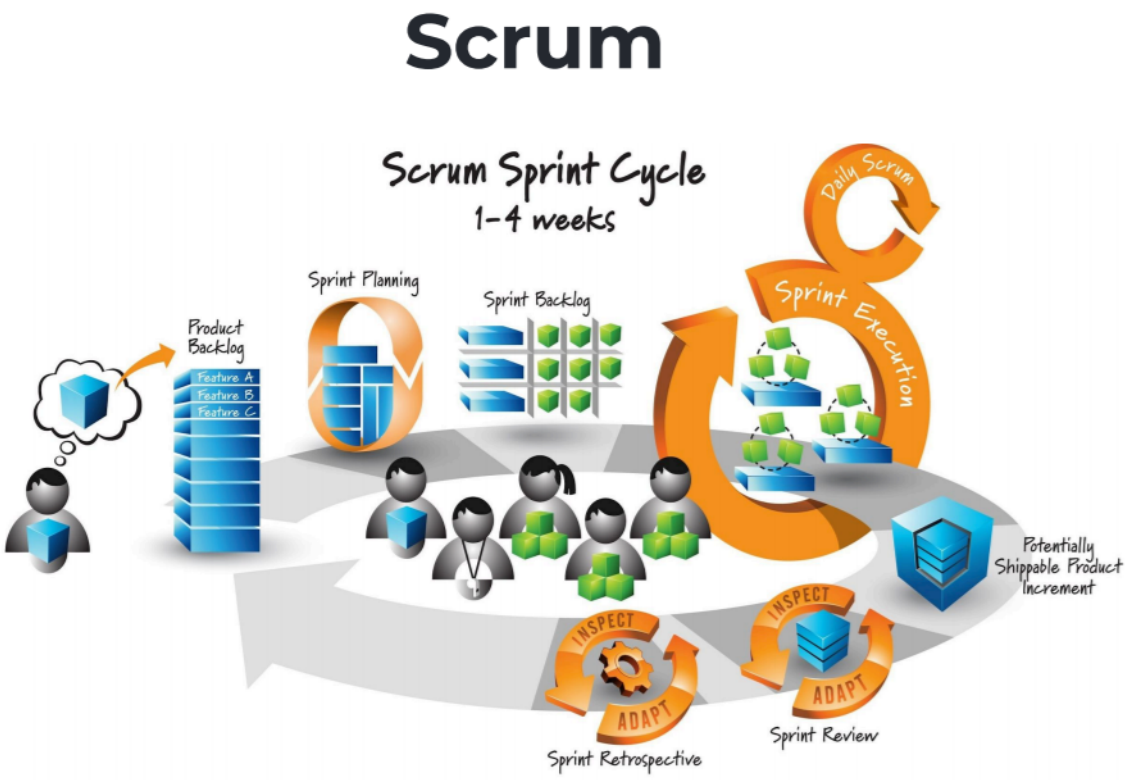
O processo de desenvolvimento de produtos na empresa se dá através de métodos ágeis [2], mais especificamente o framework Scrum [3]. Ao contrário do método tradicional em cascata, em que se deve especificar todos os requisitos do projeto para o prosseguimento

para as próximas etapas, sem que se alterem os requisitos pré definidos, os métodos ágeis aceitam alterações no projeto, sendo realizado em iterações que duram poucos dias (de 2 à 4 semanas), onde pode-se receber feedback constantemente, aumentando a qualidade do produto.

Na empresa, baseando-se no conceito de Scrum, divide-se os projetos em ciclos de duas semanas chamados *sprints*, onde um conjunto de atividades deve ser executado. As funcionalidades que deverão ser implementadas no projeto são organizadas em uma lista de tarefas, chamada *backlog*. À partir disto, alocam-se os membros da equipe para cada tarefa, onde os mesmos devem classificar as tarefas de acordo com nível de complexidade. Ao início de cada Sprint, é realizado o *Sprint Planning* (planejamento da Sprint), onde o Project Owner (PO) prioriza os itens do *backlog* de acordo com a complexidade das tarefas e, então, os membros da equipe decidem quais atividades serão implementadas durante aquela Sprint.

Todos os dias acontece uma reunião (chamada *Stand Up Meeting*) com a equipe, onde os desenvolvedores explicam o que foi realizado no dia anterior, e o que deve ser feito no dia que se inicia, fazendo com que toda a equipe saiba o andamento dos projetos. O fluxo do projeto pode ser visto na figura 2.

Figura 2 – Scrum - Ciclo de Sprint



Fonte: Innolution, LLC and Kenneth S. Rubin

A Jungle Devs divide a equipe em Tribos, que são similares a um time de Scrum. Neste time, deve-se ter as habilidades e ferramentas necessárias para desenvolver os projetos atribuídos à tribo, incluindo design, desenvolvimento, testes e o lançamento do projeto. Cada tribo tem um *Manager* de projeto, responsável pelo fluxo de trabalho e entregas, pelos desenvolvedores e pelo projeto em si.

A empresa faz uso de controle de versionamento (Git) dos projetos, utilizando o serviço Github, que será comentado no próximo capítulo. Desta forma, após ser concluída uma parte do código, outros desenvolvedores podem avaliá-lo e propor alterações.

2.2 Programa Academy

O programa Academy tem o objetivo de ser um primeiro contato com o ambiente de desenvolvimento profissional, onde pode-se aprender e reforçar conceitos de desenvolvimento voltados para a área de interesse, seja este *front-end*, *back-end*, mobile, web e design.

O programa tem duração de um ano, onde inicialmente foca-se em aprender os conceitos, linguagens de programação e ferramentas que são utilizadas na empresa e em empresas de desenvolvimento em geral, e o estagiário progressivamente se envolve mais nos projetos reais da empresa.

Os participantes do programa contam com a ajuda de tutores, que sanam dúvidas e auxiliam no seu crescimento.

São passados desafios no início do projeto, para desenvolver os conceitos básicos das linguagens empregadas. São também realizados testes de qualidade do produto (Quality Assurance) em projetos reais da empresa, onde o objetivo é procurar *bugs* nos projetos e reportá-los para que sejam reparados. Isto garante maior contato dos participantes do programa com projetos reais.

Com o passar do tempo, com os conceitos mais concretizados, os integrantes do programa começam a participar de projetos reais, desenvolvendo *features*, criando e lançando os projetos.

3 Conceitos

Antes de serem apresentados os problemas, soluções propostas e desenvolvimento mais a fundo, é necessário fazer uma revisão dos conceitos, práticas e tecnologias utilizadas. O propósito desse capítulo é apresentar brevemente assuntos pertinentes a execução das atividades durante o período do estágio.

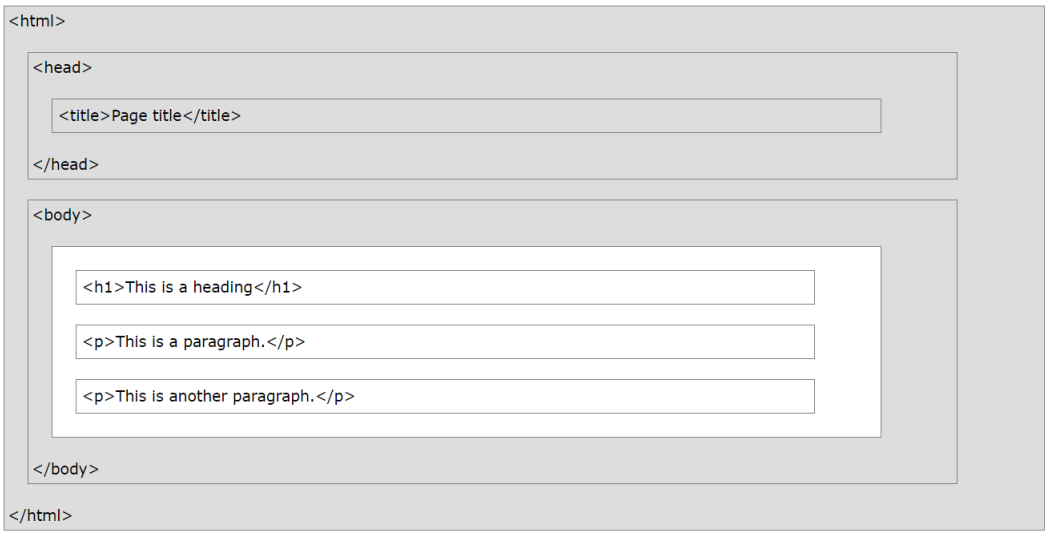
3.1 HTML

A linguagem Hypertext Markup Language [4] (HTML), é utilizada para desenvolvimento de websites. É de fácil entendimento tanto para programadores quanto para máquinas. Baseia-se em marcação, onde marcam-se os elementos para mostrar as informações que deverão ser exibidas na página.

Utilizam-se elementos para construir as páginas e estes são representados por tags. As tags dão nome as partes do conteúdo, como parágrafos, tabelas e títulos (paragraphs, tables e headings). Os navegadores não mostram as tags HTML, mas as utilizam para renderizar o conteúdo da página.

Pode-se observar um exemplo de utilização de HTML na figura 3.

Figura 3 – Exemplo HTML



Fonte: W3schools.com - Tutorial HTML.

3.2 CSS

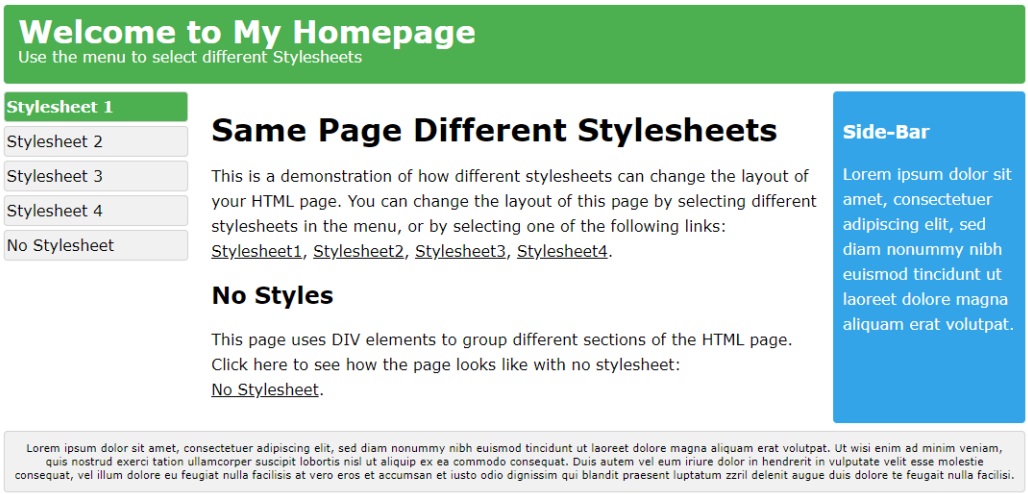
A linguagem de programação CSS (Cascading Style Sheets, ou Folha de Estilo em Cascatas) [5] é utilizada para descrever como elementos HTML deverão ser mostrados na tela. Pode controlar o layout de diversas páginas da web ao mesmo tempo.

A linguagem CSS também possibilita variações no display para telas responsivas, como variações de tamanho de tela e dispositivos.

Ela foi criada para solucionar o problema que os desenvolvedores web tinham em adicionar estilos para suas páginas web em HTML, principalmente quando tinha-se um site muito grande com diversas páginas, em que se tinha que adicionar o estilo para cada página individualmente. Então, a linguagem CSS removeu a formatação de estilo dos arquivos HTML, que passou a ser feita nos arquivos CSS.

Resumidamente falando, o HTML é reponsável pela estrutura da página, enquanto que o CSS determina o estilo, como cores, tamanhos, animações, entre outros. Isto pode ser observado nas figuras 4 e 5, onde, com o mesmo arquivo HTML para estruturar a página, são aplicados dois arquivos CSS diferentes, que alteram o layout da mesma.

Figura 4 – Exemplo CSS - Estilo 1

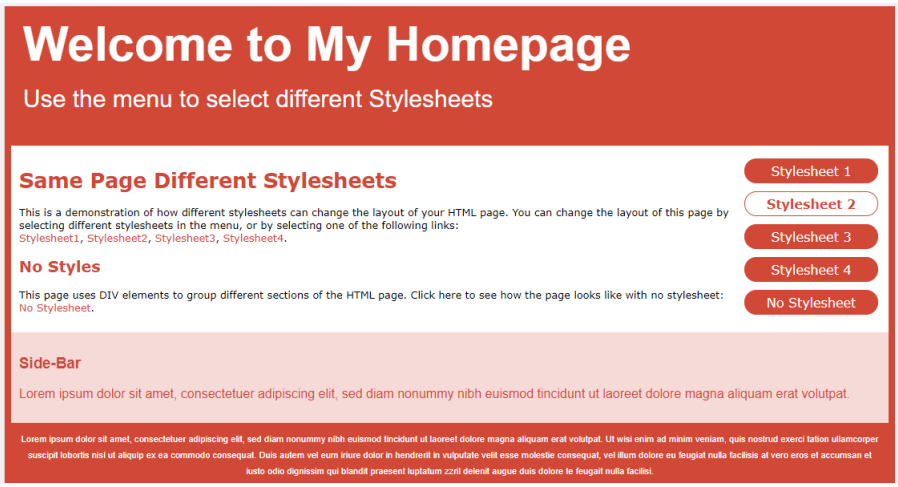


Fonte: W3schools.com - Tutorial CSS.

3.3 JavaScript

JavaScript [6] é uma linguagem de programação interpretada, que permite implementar itens complexos em páginas web (como, por exempli, botões e suas ações de clique), como conteúdo que se atualiza em determinado intervalo de tempo, mapas interativos,

Figura 5 – Exemplo CSS - Estilo 2



Fonte: W3schools.com - Tutorial CSS.

gráficos animados, entre outros. O JavaScript interage com o HTML e CSS e controla-os para tornar a página dinâmica.

Atualmente é a principal linguagem para programação client-side para navegadores web através de tecnologias como Angular JS, React e Vue Js e também utilizada no lado do servidor (Node.js, MongoDB e outras).

Ao carregar uma página web em um navegador, executa-se o código da página (HTML, CSS e JavaScript) dentro de um ambiente de execução. O arquivo JavaScript é executado pelo motor de renderização do navegador somente após os arquivos HTML e CSS terem sido renderizados. O arquivo JavaScript irá, então, modificar dinamicamente estes arquivos para atualizar a interface do usuário (Figura 6).

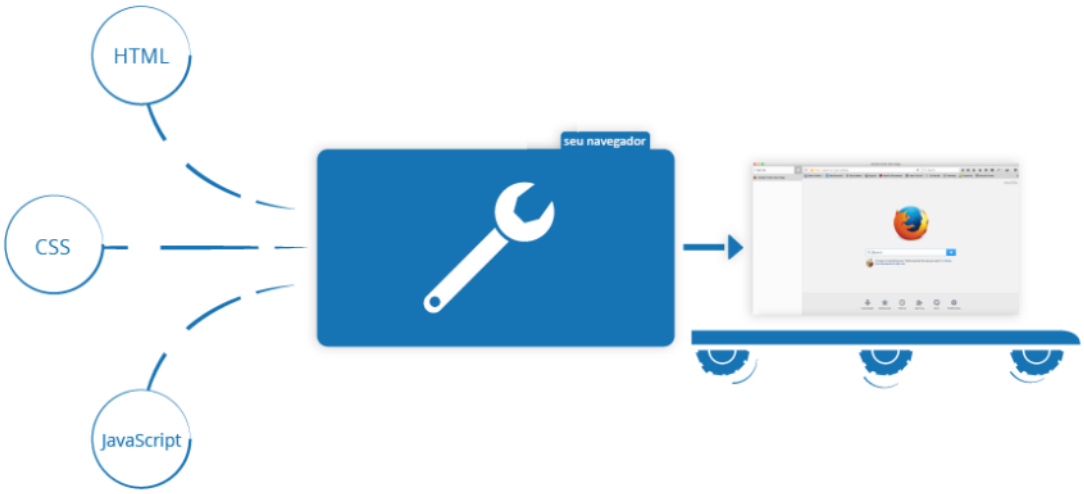
3.4 ReactJS

ReactJS [7] é uma biblioteca JavaScript declarativa, eficiente e flexível para criação de interfaces de usuário (UI). Ou seja, ReactJS não é um framework, mas uma coleção de funcionalidades relacionadas que podem ser utilizadas para resolver problemas específicos, como a criação de interfaces reutilizáveis.

O ReactJS é responsável pela criação de interfaces de usuário, organizando o conteúdo para o usuário final.

Esta biblioteca surgiu em 2011 com o Facebook, depois passou a ser utilizada pelo Instagram, até ser finalmente aberta à comunidade em 2013, o que colaborou bastante para a sua popularização.

Figura 6 – HTML, CSS e JavaScript no navegador



Fonte: Developer Mozilla.

A biblioteca já é referência no desenvolvimento *front-end*, consolidando-se como uma das mais utilizadas pelos desenvolvedores atualmente.

O ReactJS separa o código em pequenas partes que se comportam como componentes reutilizáveis. Enquanto que no HTML todo o código está condensado em um único arquivo, com ReactJS cada parte é um arquivo (componente) separado.

Isto pode ser observado na figura 7, em que cada parte especificada com um número é um componente diferente, que compõem uma única página.

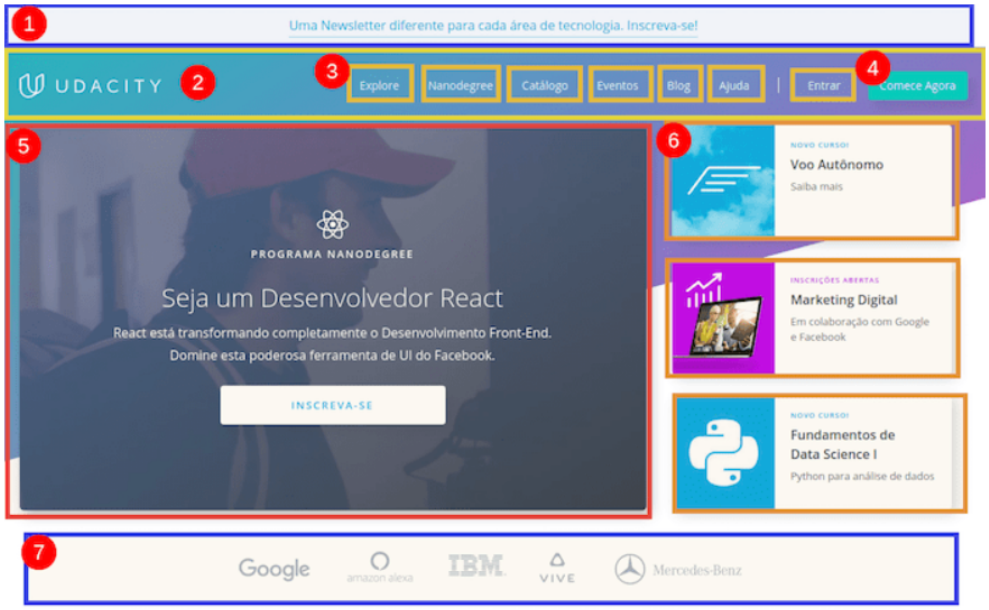
3.5 Redux

O Redux [8] é uma implementação da arquitetura Flux (arquitetura que, juntamente com o framework React, é usada para construir aplicações web que funcionem de forma reativa), que foi criada pelo Facebook. Ela soluciona o problema de compartilhamento de estados entre componentes, tornando-a unidirecional. Isto pode ser observado na figura 8 à seguir.

Observa-se que o Redux simplifica a evolução de estados de uma aplicação quando têm-se múltiplos estados e muitos componentes que deseja-se controlar. Assim, tira-se a responsabilidade de guardar o estado em cada componente, passando esta responsabilidade para uma *Store* centralizada.

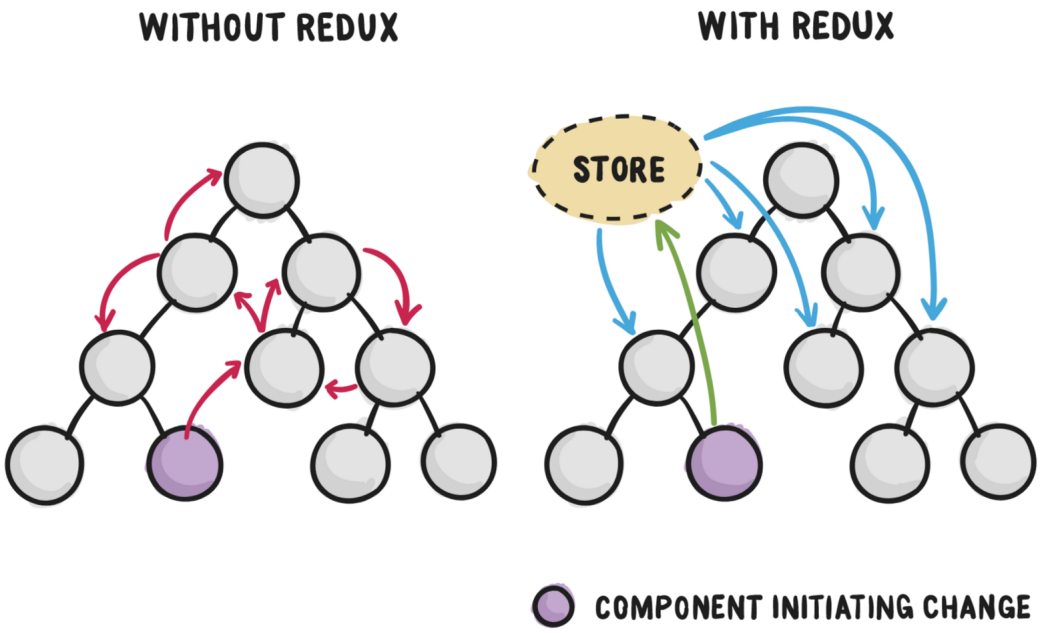
Na figura 9 à seguir, é possível ver como funciona o fluxo de uma evolução de estado com Redux.

Figura 7 – Componentes ReactJs em uma página web



Fonte: Blog da Udacity.

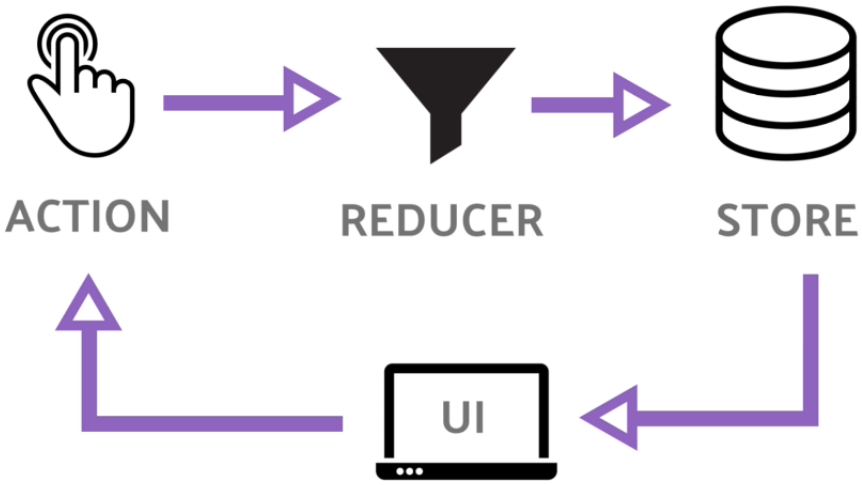
Figura 8 – Arquitetura Flux - com e sem Redux



Fonte: Medium.

A utilização do Redux depende de 4 partes. Estas são a *Store*, *Actions*, *Reducers* e as Conexões dos componentes ao Redux.

Figura 9 – Evolução de Estados com Redux



Fonte: Medium.

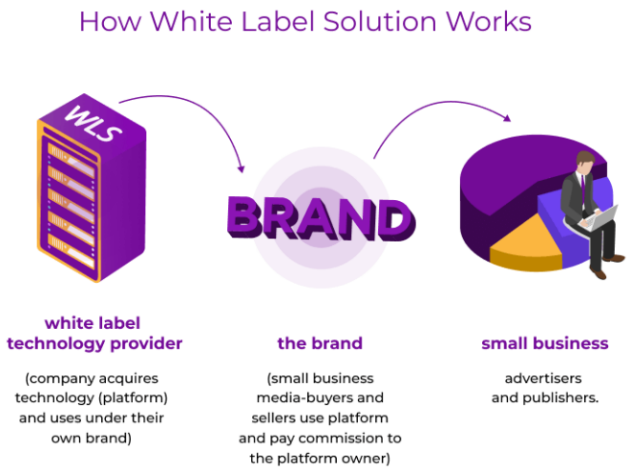
A Store é um container que armanzena o estado da aplicação. É imutável, então nunca se altera. As Actions são fontes de informação enviadas da aplicação para a Store. São funções que ativam os Reducers. Reducers são responsáveis por receber as informações e tratá-las para que sejam enviadas à Store. E os componentes devem se inscrever à evolução dos estados da Store ou disparar eventos para evoluí-la.

3.6 White-Label

Uma plataforma White-Label [9] é um produto de software já pronto, desenvolvido por uma empresa, onde uma outra aplica a sua marca em cima.

Ou seja, a plataforma está pronta para uso, e uma empresa cliente apenas aplica sua marca, alterando cores, logos, ícones e alguns componentes de texto para que a plataforma fique alinhada com a sua marca.

Figura 10 – Funcionamento de White label.



Fonte: Smarty Ads.

3.7 Software as a Service (SaaS)

Software as a service (Saas, ou Software como serviço [10]) é quando um sistema é comercializado como um serviço, ao invés de um produto. Ou seja, não é necessaria a instalação de nenhum software, mas utiliza-se as aplicações através da internet.

O software pode ser utilizado de qualquer lugar, por qualquer dispositivo, desde que se tenha acesso à internet.

O cliente não compra o produto, mas paga pela assinatura por utilizar o serviço.

Um software SaaS pode ser desde uma aplicação como um e-mail pessoal, até soluções para gerir operações em uma empresa.

Ao utilizar um SaaS, o contratante não precisa se preocupar em pagar licenças de softwares, nem com contatar um time de engenharia para desenvolver a aplicação, e pode focar inteiramente na atividade-fim da empresa. Isto acaba por reduzir gastos.

As principais características de um SaaS estão listadas à seguir:

- Acesso ao serviço totalmente através da internet;
- Correções e atualizações do serviço são feitas pelo fornecedor do sistema;
- Integração com APIs de terceiros;
- Pagamento por uso e não por licença;
- Gerenciamento da aplicação é centralizado.

3.8 Client Side Rendering

Client Side Rendering [11] se refere à carregar o conteúdo no browser através de JavaScript. Ou seja, ao invés de se receber todo o conteúdo através do HTML do site, o usuário recebe um HTML "cru", juntamente com o JavaScript que irá renderizar o site através do navegador.

O *request* inicial irá retornar um arquivo HTML. Então, o lado do cliente (client side) irá chamar os *end points* das APIs correspondentes. Os dados são armazenados em um estado e são cacheados (*chached*).

Há alguns pontos negativos à se considerar quando utiliza-se aplicações Client Side. O conteúdo não será totalmente rendezirado até que a página seja totalmente carregada no navegador.

Além disto, a aplicação não é carregada até que todo o JavaScript necessário seja baixado pelo navegador. Ou seja, caso o usuário tenha uma conexão lenta com a internet (por exemplo, usando dispositivos móveis), pode ser que o carregamento inicial seja relativamente lento.

A arquitetura de uma aplicação Client Side Rendering pode ser observada na figura 11 abaixo.

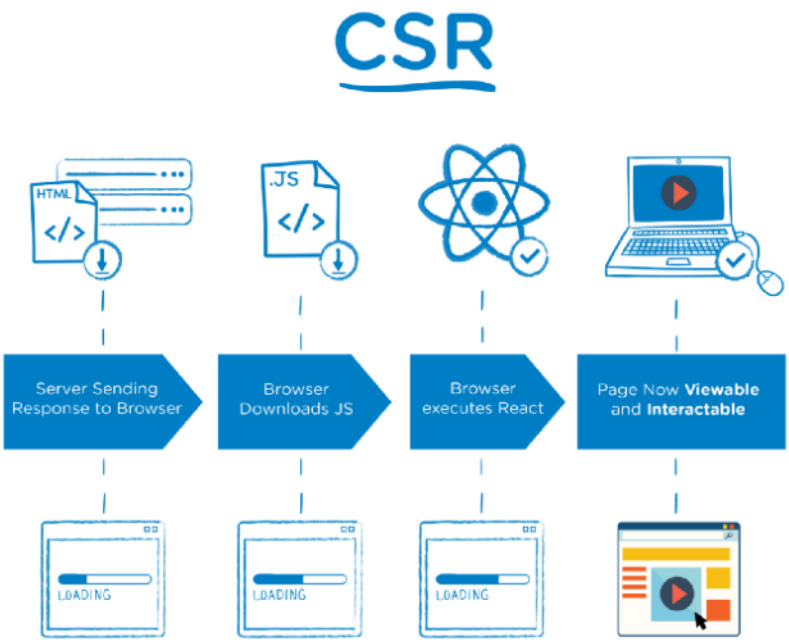
3.9 Server Side Rendering

Server Side Rendering (SSR) [12] é o processo de renderizar todos os arquivos JavaScript a CSS de um site como estáticos do lado do servidor.

O objetivo disto é fazer com que o tempo de carregamento do site seja reduzido, e possibilitar que o site seja totalmente indexável por SEO's (*Search Engine Optimization*) e Crawlers.

Com o SSR, é possível entregar todo o conteúdo significamente para o usuário quase imediatamente. Isto de dá porque o HTML e seus principais assets (como por exemplo imagens e ícones) são carregados em um mesmo arquivo e entregue ao cliente.

Figura 11 – Estrutura de Client Side Rendering



Fonte: Medium.

Logo, pulam-se algumas etapas do download dos assets, e o browser deve somente carregar os components criados e fluxos que ficam em um arquivo reduzido. É possível observar a arquitetura do Server Side Rendering na figura 12.

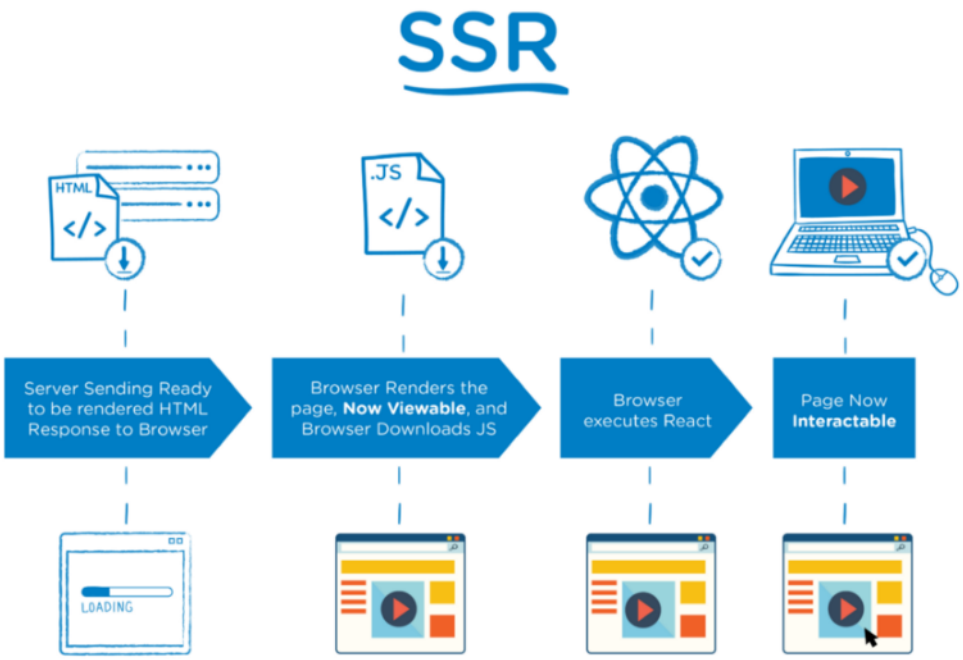
3.10 Github

Primeiramente, precisa-se explicitar o conceito de Git. Git [13] é um sistema de controle de versionamento de arquivos. Utilizando ele, pode-se desenvolver projetos onde diversos desenvolvedores podem contribuir simultaneamente, alterando e criando novos arquivos, sem que se sobre escreva arquivos antigos.

Já o Github [14] é uma serviço web que oferece funcionalidades extras aplicadas ao git. Ele permite aos desenvolvedores hospedarem seus projetos. Permite também seguir projetos de outros desenvolvedores, e comentar sobre os mesmos.

O serviço oferece acesso gratuíto limitado e planos pagos que oferecem maior controle sobre o código fonte. É um serviço que armazena o material na nuvem, podendo ser acessado de qualquer lugar.

Figura 12 – Estrutura de Server Side Rendering



Fonte: Medium.

3.11 Jira

Jira [15] é uma ferramenta que permite o monitoramento de tarefas e acompanhamento de projetos em um único lugar.

Nele, é possível criar projetos, que agrupam tarefas que podem ser organizadas de diferentes formas, de acordo com as necessidades. É possível criar diferentes tipos de projetos, com diferentes tipos de metodologias ágeis usadas, podendo criar-se um quadro de Kanban ou Scrum, por exemplo.

3.12 CircleCI

CircleCI [16] é uma ferramenta de integração continua, que integra com sistemas de versionamento contínuo (como o GitHub) e roda automaticamente uma série de passos toda vez que ele detecta alguma mudança no repositório.

Normalmente, uma *build* no CircleCI consiste em instalar as dependências, testar e lançar o aplicativo.

4 O projeto

Esta seção tem como objetivo apresentar a plataforma web para imobiliárias, Woliver, expondo suas funcionalidades principais, retificando seus requisitos gerais, funcionais e não funcionais, e demonstrar o que a plataforma se propõe a fazer.

A plataforma foi idealizada pela imobiliária Brognoli, com o objetivo de inovar e gerar meios mais rápidos e simples para solucionar o problema de locação de imóveis. Objetivou-se, então, o desenvolvimento de um Software as a Service , ou seja, é um serviço de aluguéis residenciais, onde cada imobiliária aplicará sua marca em cima da plataforma, para sua customização (como nome, logo, cores, imagens, serviços, cartilha de imóveis,...).

Nesta plataforma, é possível fazer a busca de imóveis residenciais para aluguel, aplicando-se filtros na busca. É possível ver as informações do imóvel, como valores, número de quartos, garagens, área, descrição, imagens, videos e até mesmo mapas, onde é possível simular trajetos até o imóvel. É possível, também, fazer agendamentos para visitações nos imóveis, onde escolhe-se data e hora para visitação.

Para se entender melhor a motivação para a criação desta plataforma, faz-se necessário entender o contexto de locação de imóveis atual. Cada vez mais se faz necessário inovar e gerar meios mais rápidos e simples para solucionar problemas. A plataforma tem como objetivo tornar a experiência dos usuários buscando um imóvel mais rápida, encontrando resultados mais precisos para suas necessidades e com menos burocracia para agendamento de visitas. Ao mesmo tempo, torna mais fácil e rápida e implementação de uma plataforma para as imobiliárias, podendo utilizar uma plataforma pronta, apenas utilizando seus serviços e aplicando sua marca à ela. Serão coletados dados dos usuários dentro da aplicação para se ter mais insumos para fechar negócios com potenciais clientes.

O processo de agendamento de visitas a imóveis era feito por telefone ou e-mail, ou indo diretamente na agência, somente sendo feito entrando em contato com um agente imobiliário. Então, têm-se o objetivo de pular uma etapa deste processo, onde não mais se requer que o cliente entre em contato com um agente imobiliário, e ele mesmo pode agendar sua visita através da plataforma.

Ressaltando que, neste projeto, o levantamento de requisitos funcionais e não funcionais da especificação foram feitos pelo cliente, bem como o design das páginas da plataforma.

4.1 Busca de Imóveis

O usuário poderá escolher critérios de busca, através de filtros pré definidos dentro da plataforma. Após escolher os filtros de interesse, o sistema deve listar os imóveis que combinem com esta busca.

Os resultados desta busca podem se dar de duas maneiras. Resultados totais, que podem ser observados na tabela 1 e na figura 13, e os resultados exatos e resultados parciais.

Tabela 1 – Requisito Funcional - Listar imóveis na busca

F1: Listar imóveis na busca	
Descrição: O sistema deve listar os imóveis que combinem com a busca que o usuário fez	
Requisitos Não-Funcionais	
Nome	Restrição
Imóvel	Deve constar o título do imóvel, imagens, descrição, número de garagens, quartos e banheiros do mesmo

Caso o usuário seja redirecionado para a página de busca B, explicada na seção 5.2.3, os resultados irão aparecer entre resultados exatos (tabela 2), que correspondem exatamente à busca feita, e resultados parciais (tabela 3), que corresponde apenas parte da busca feita. O design de ambos aparece na figura 14.

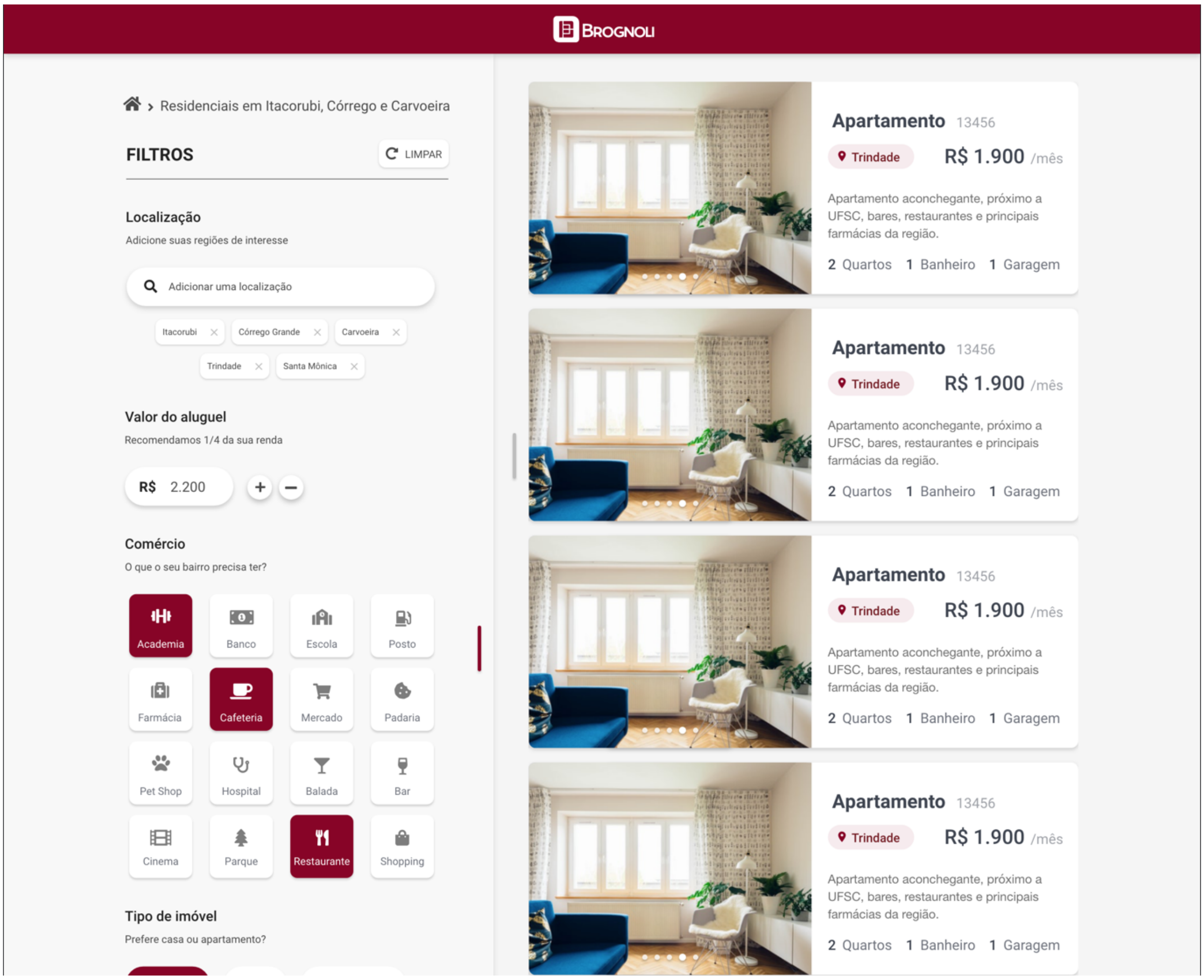
Tabela 2 – Requisito Funcional - Listar resultados de imóveis exatos na busca

F2: Listar imóveis exatos na busca	
Descrição: O sistema deve listar os imóveis que combinem exatamente com a busca que o usuário fez	
Requisitos Não-Funcionais	
Nome	Restrição
Resultado Imóvel Exatdo	Deve constar o título do imóvel, imagens, descrição, número de garagens, quartos e banheiros do mesmo

Tabela 3 – Requisito Funcional - Listar resultados de imóveis parciais na busca

F3: Listar imóveis parciais na busca	
Descrição: O sistema deve listar os imóveis que combinem parcialmente com a busca que o usuário fez	
Requisitos Não-Funcionais	
Nome	Restrição
Resultado Imóvel Parcial	Deve constar o título do imóvel, imagens, descrição, número de garagens, quartos e banheiros do mesmo

Figura 13 – Design para tela de resultados totais



Fonte: Zeplin, Jungle Devs.

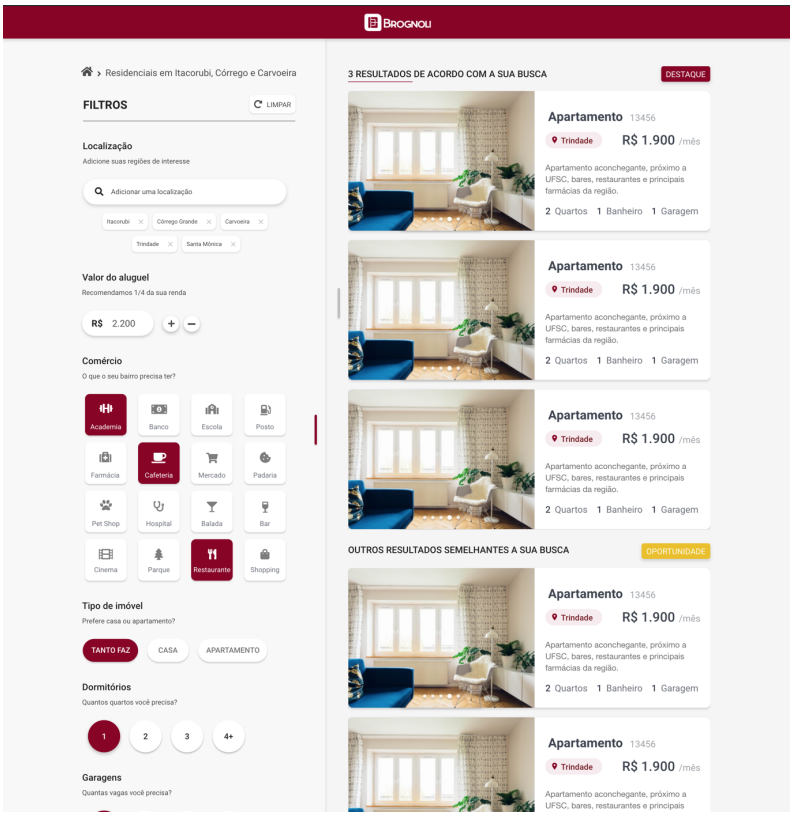
4.2 Imóvel

Após selecionar um imóvel, o usuário é redirecionado para outra página, contendo informações do mesmo. A descrição deste requisito está na tabela 4 e o design da página pode ser visto na figura 15.

Tabela 4 – Requisito Funcional - Informações do imóvel

F4: Informações do imóvel	
Descrição: O sistema deve apresentar as informações do imóvel	
Requisitos Não-Funcionais	
Nome	Restrição
Imóvel	Deve constar o título do imóvel, imagens, endereço, descrição, caracterísicas (número de garagens, quartos e banheiros, e área) id do imóvel, preços, mapas e video

Figura 14 – Design para tela de resultados exatos e parciais



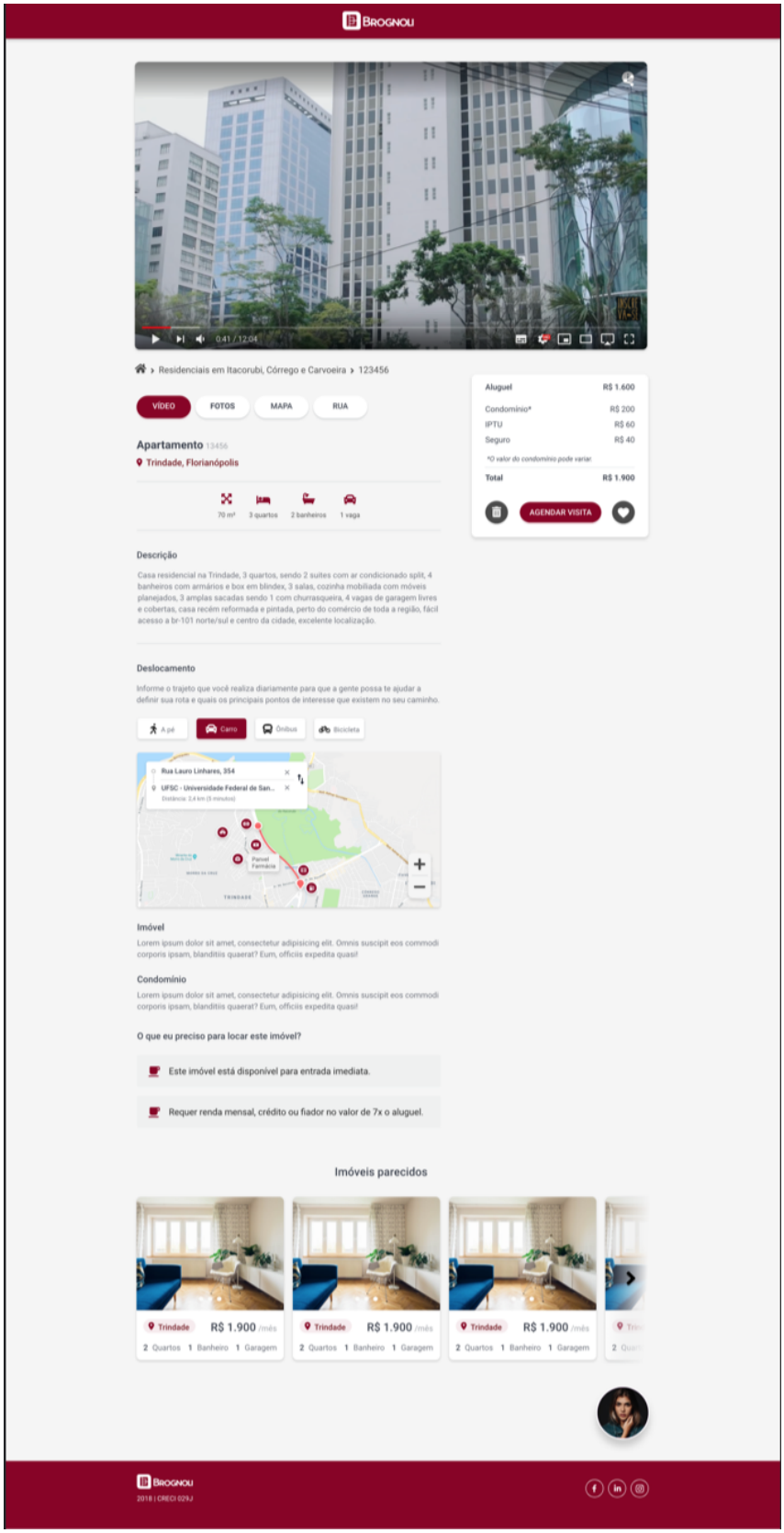
Fonte: Zeplin, Jungle Devs.

O usuário também tem a opção de adicionar o imóvel selecionado à lista de imóveis que gostou ou à lista de imóveis que não gostou. Ambos os requisitos funcionais podem ser vistos nas tabelas 5 e 6 respectivamente.

Tabela 5 – Requisito Funcional - Adicionar imóvel à lista gostei

F5: Adicionar imóvel à lista gostei	
Descrição: O usuário deve poder adicionar um imóvel à lista gostei	
Requisitos Não-Funcionais	
Nome	Restrição
Adicionar imóvel à lista gostei	Deve adicionar o imóvel à lista de imóveis que o usuário gostou

Figura 15 – Design para tela de detalhes do imóvel



Fonte: Zeplin, Jungle Devs.

Tabela 6 – Requisito Funcional - Adicionar imóvel à lista não gostei

F6: Adicionar imóvel à lista não gostei	
Descrição: O usuário deve poder adicionar um imóvel à lista não gostei	
Requisitos Não-Funcionais	
Nome	Restrição
Adicionar imóvel à lista não gostei	Deve adicionar o imóvel à lista de imóveis que o usuário não gostou

4.3 Agendamento

Um outro requisito funcional do sistema é que o usuário deve ser capaz de realizar o agendamento de uma visita a um imóvel, selecionando data e hora para tal. Este requisito é visto na tabela 7.

Tabela 7 – Requisito Funcional - Agendar visita à imóvel

F7: Agendar visita à imóvel	
Descrição: O usuário deve poder agendar uma visita à um imóvel	
Requisitos Não-Funcionais	
Nome	Restrição
Agendar Visita	Deve permitir agendar uma visita à um imóvel, mediante o usuário selecionar data e hora para visitação

O design da tela para a seleção de data e hora do fluxo de agendamento pode ser observado na figura 16 abaixo.

4.4 Usuário

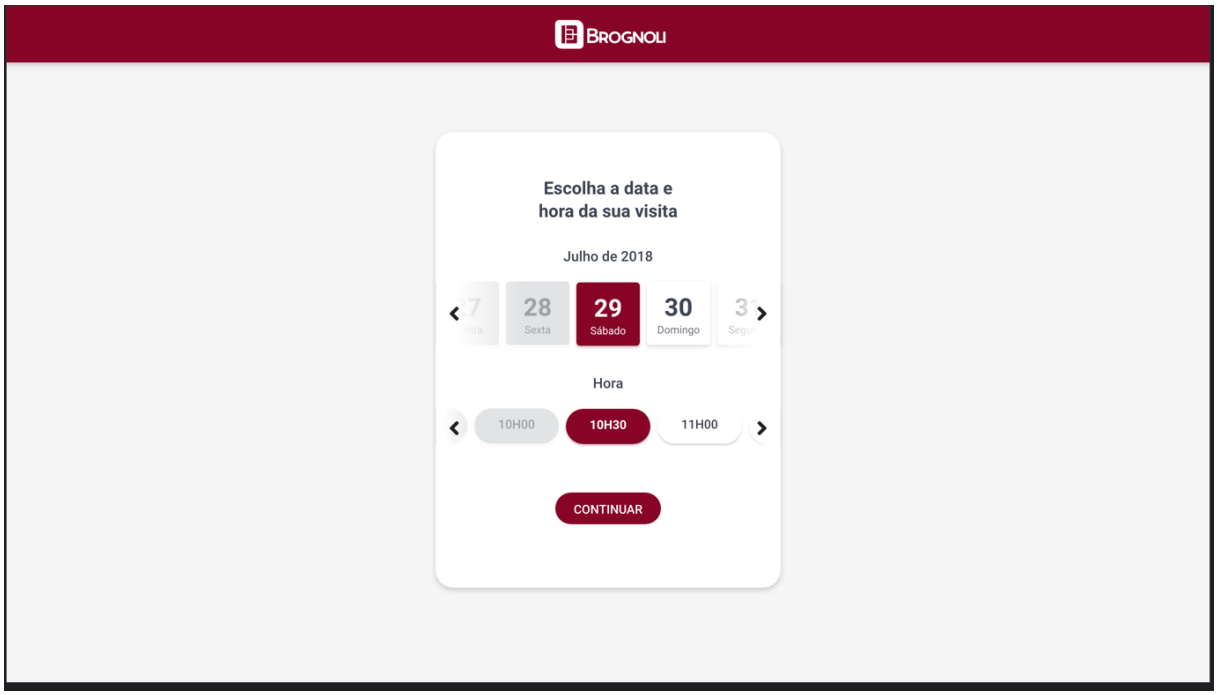
O sistema deve, também, permitir que o usuário faça um cadastro. O cadastro conta com nome, email e CFP do usuário. O requisito funcional pode ser visto na tabela 8.

Tabela 8 – Requisito Funcional - Criar cadastro do usuário

F8: Criar cadastro do usuário	
Descrição: O usuário deve poder criar um cadastro na plataforma	
Requisitos Não-Funcionais	
Nome	Restrição
Criar cadastro	Deve permitir ao usuário criar um cadastro na plataforma

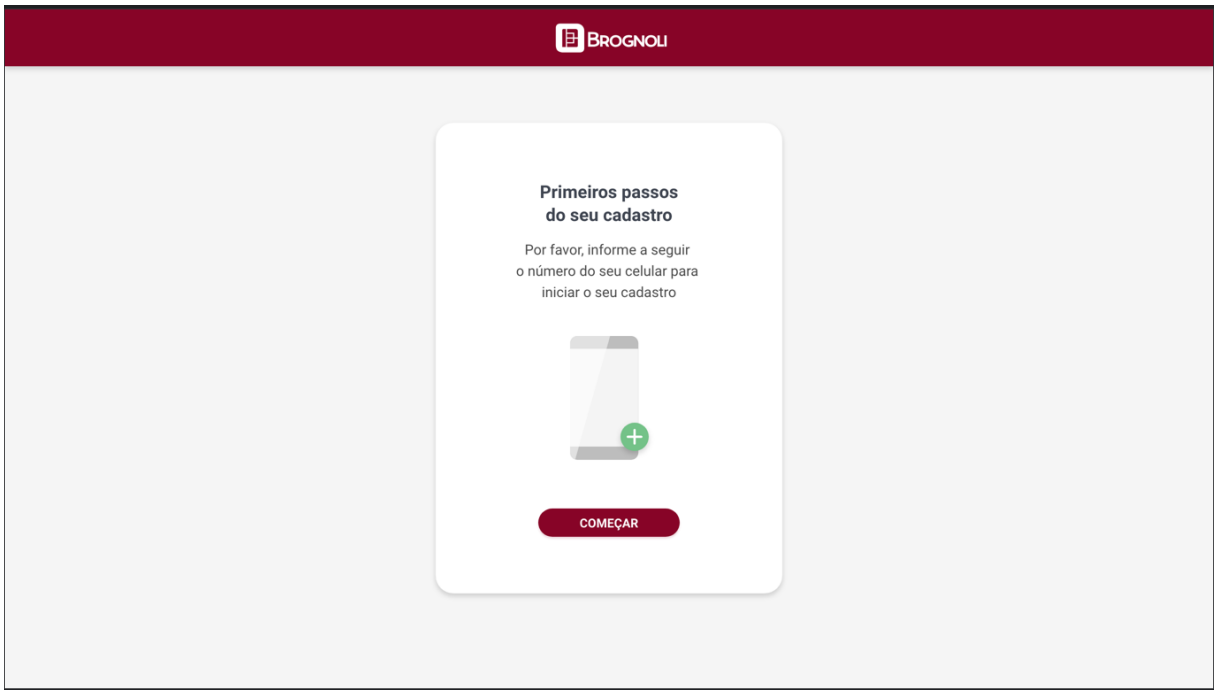
Nas figuras 17 e 18 observa-se o design para as telas responsáveis pelo fluxo de cadastramento de usuário.

Figura 16 – Design para tela de seleção de data e hora para agendamento



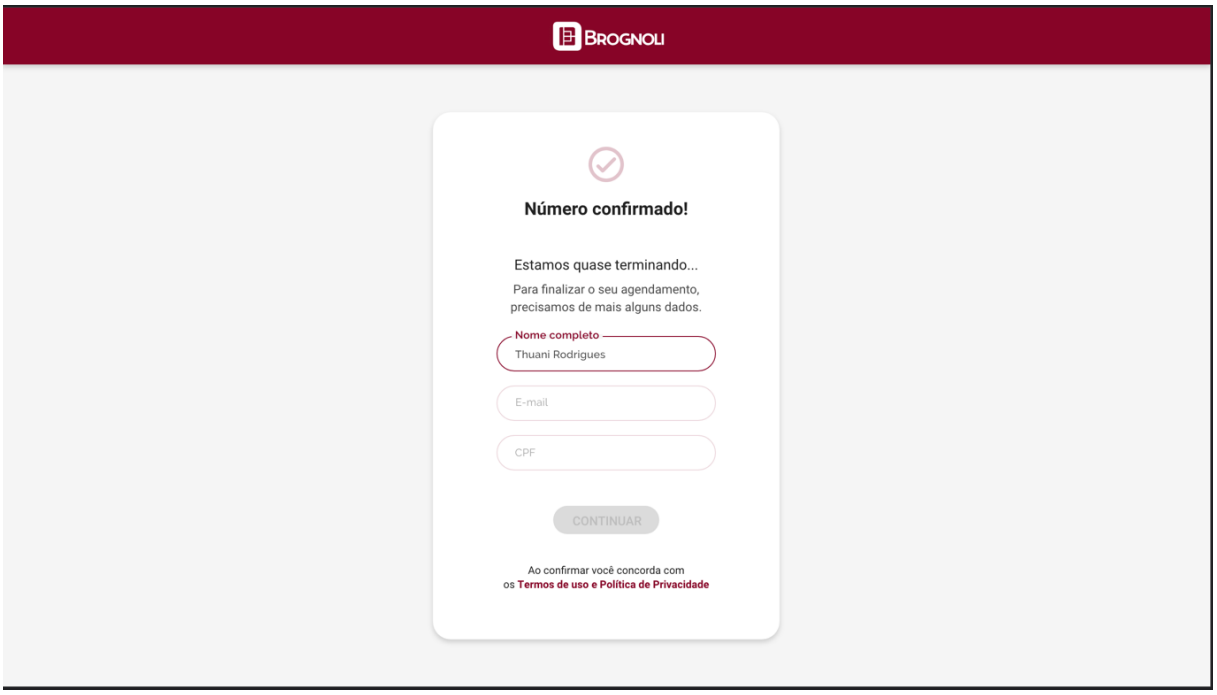
Fonte: Zeplin, Jungle Devs.

Figura 17 – Design para tela de Cadastro do usuário (AccountKit)



Fonte: Zeplin, Jungle Devs.

Figura 18 – Design para tela de Cadastro do usuário: dados do usuário



Fonte: Zeplin, Jungle Devs.

5 Atividades Desenvolvidas

Dando continuidade, com os processos e ferramentas apresentados respectivamente nos capítulos 2 e 3 e com o sistema definido na seção anterior, tem-se agora o contexto adequado para apresentar as tarefas realizadas no projeto em si, sendo assim, essa seção tem como objetivo apresentar as atividades desenvolvidas pela estagiária no projeto, durante o período das atividades.

Vale a pena ressaltar que as atividades desenvolvidas durante o período de estágio tiveram foco em desenvolvimento *front-end*, sendo feita apenas a integração com o *back-end*. O *back-end* foi desenvolvido por um engenheiro de software da empresa e sua integração com o *front-end* foi feita utilizando protocolo HTTP [17]. O formato dos dados transferidos é JSON, utilizando os métodos GET, POST e PATCH (para retornar informações, para criar informações e para atualizar informações, respectivamente).

5.1 Componentes

Foram criados alguns componentes que são reutilizados através da plataforma. Estes, estão explicitados abaixo.

5.1.1 Botões

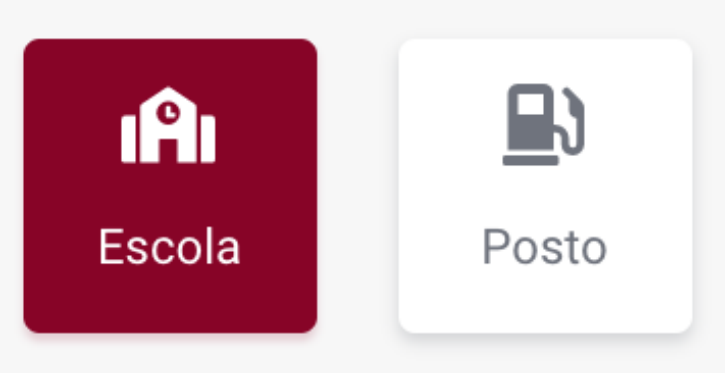
Foi criado o componente Botão. Este apresenta diversas cores, que na aplicação chamamos de temas, e com diversos formatos (redondo, quadrado ou com ícone). Foi criado um componente reutilizável para tal. Na figura 19 abaixo, é possível observar o botão com tema padrão (do meio) e dois botões com ícone.

Figura 19 – Componente: Botão

Fonte: Jungle Devs.

Já na figura 20, é possível observar o botão quadrado. Na esquerda, o botão está selecionado, na direita não está selecionado.

Figura 20 – Componente: Botão Quadrado



Fonte: Jungle Devs.

5.1.2 Entradas de Texto

Foi criado, também, um componente reutilizável de Entradas de texto (*input*). Foram criadas três inputs diferentes, explicitadas à seguir.

5.1.2.1 Entrada de texto padrão

As entradas de texto padrão consistem apenas nas inputs de texto com uma label. Ela apresenta diferentes estados, para quando está preenchida e quando está vazia. Apresenta também um estado de erro. Estes, estão representados à seguir, na figura 21, onde o componente supeior é o design padrão e o segundo é o componente em estado de erro.

Figura 21 – Componente: Entrada de texto padrão

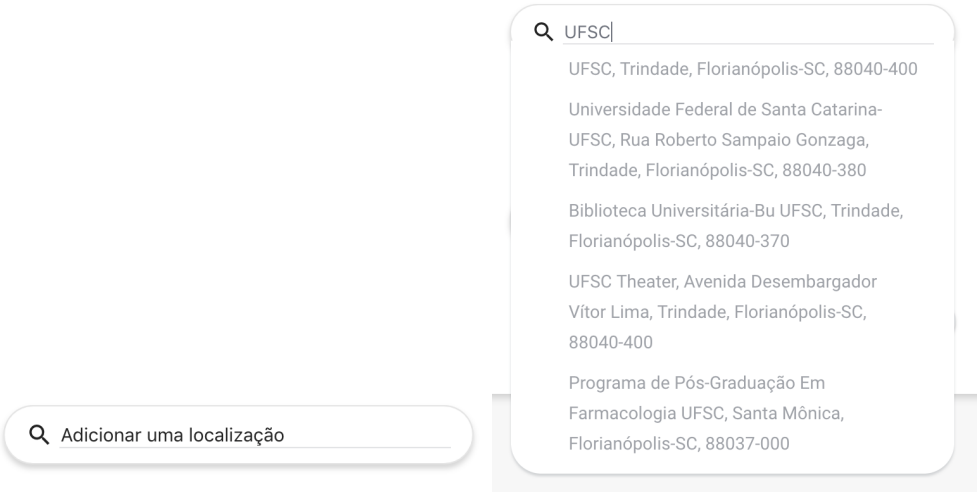
Fonte: Jungle Devs.

Estes componentes estão presentes no quarto passo de agendamento, que será explicado nas seções seguintes.

5.1.2.2 Entrada de texto de localização

As entradas de texto de localização estão presentes na tela de filtros, na página inicial e no mapa na tela de detalhes do imóvel. Elas tem o design um pouco diferente das inputs padrão, apresentando um ícone, bem como um menu dropdown que aparece ao escrever algo. Isto pode ser observado na figura 22.

Figura 22 – Componente: Entrada de texto de localização



Fonte: Jungle Devs.

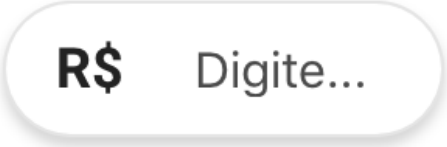
Esta entrada de texto tem integração com a API Here (de mapas e geolocalização), utilizada na busca por localizações, que será explicada na seção 5.3.3.

5.1.2.3 Entrada de texto numérica

O componente de entrada de texto numérica está presente nas seções de busca de imóveis, contidas na página inicial e na página de filtros de busca.

Ela tem o design parecido com a input de localização, mas aceita apenas valores numéricos inteiros. Observa-se este componente na figura 23 à seguir.

Figura 23 – Componente: Entrada de texto numérica



Fonte: Jungle Devs.

5.1.3 Componentes de Ícones

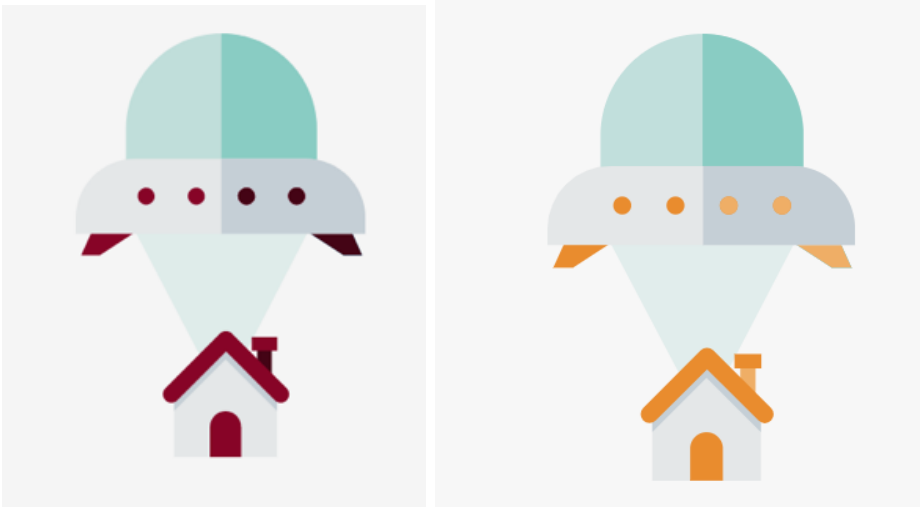
Devido a plataforma ser customizável e ser possível alterar as cores da mesma, alguns arquivos SVG (de imagens) também tem suas cores alteradas.

Devido à natureza do projeto, foram criados, então, componentes para tais imagens, sendo possível alterar sua cor para a cor da marca da imobiliária.

Este componentes foram feitos pegando o código que gera as imagens (SVG), e transformando-as em um componente react. Assim, é possível passar propriedades para este componente, sendo possível alterar a cor.

Os componentes podem ser observados nas figuras que seguem.

Figura 24 – Componente de Ícone: Abdução



Fonte: Jungle Devs.

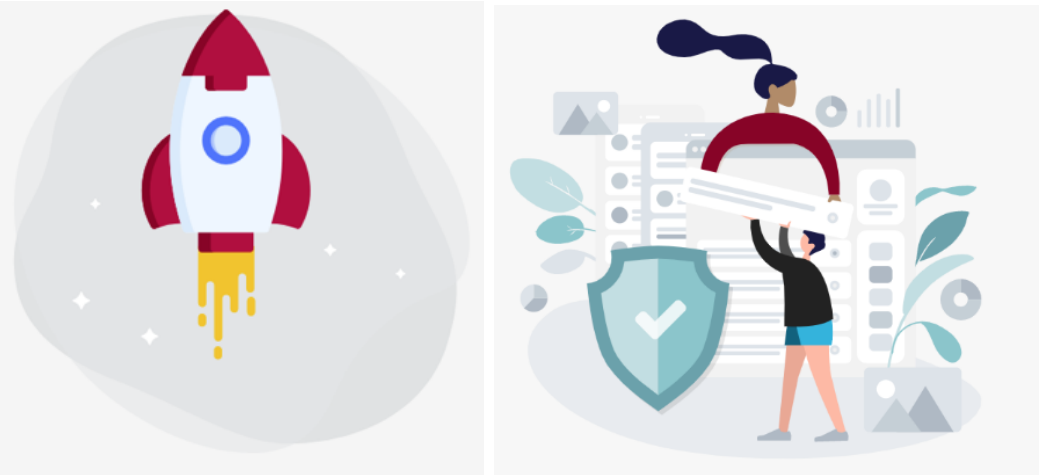
Na figura 24 é possível observar a mudança de cor no componente. Os demais componentes de ícone seguem o mesmo padrão.

Figura 25 – Componentes de Ícone: Coração e Localização

Fonte: Jungle Devs.

Além destes, também têm os marcadores que estarão presentes no mapa da api Here (explicada na seção 5.3.3).

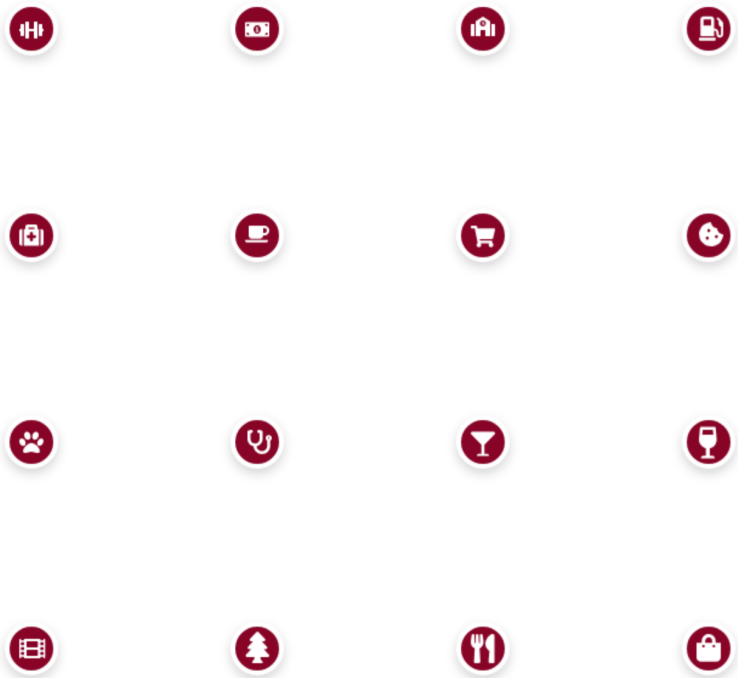
Figura 26 – Componentes de Ícone: Espaçonave e Privacidade



Fonte: Jungle Devs.

São feitos do mesmo jeito que os componentes anteriores, mas ao invés de transformá-los em um componente React, são Strings, em que se passa a cor como uma variável. É possível observar estes marcadores na figura 27 à seguir.

Figura 27 – Marcadores do mapa



Fonte: Jungle Devs.

5.2 Views

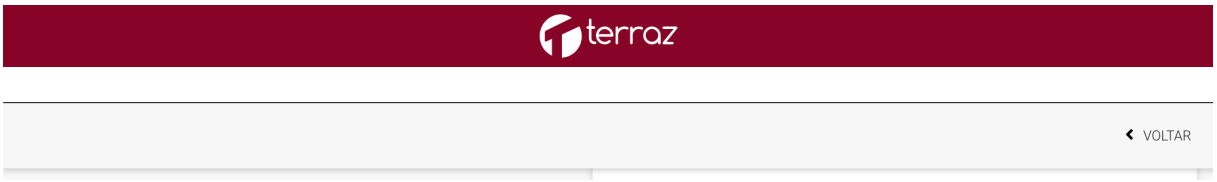
Foram desenvolvidas diferentes telas no projeto, chamadas *Views*. Nelas estão presentes as informações do aplicativo, componentes, e a integração com APIs e com o *back-end*.

Todas as *Views* foram desenvolvidas com interfaces tanto para plataformas móveis (mobile), tanto quanto para desktop. Ou seja, as *views* são responsivas, com design diferente para mobile e desktop.

5.2.1 App

A view App é uma view que contém todas as outras da plataforma. É nela que estão instanciados o cabeçalho (*header*, figura 28), o rodapé (*footer*, figura 29), o banner e o modal de cookies (figuras 30 e 31, respectivamente).

Figura 28 – Componente: Header. Tema padrão e tema de política de privacidade



Fonte: Jungle Devs.

Figura 29 – Componente: Footer

Fonte: Jungle Devs.

Para os cookies, na versão desktop temos um modal e na versão mobile temos um banner. O aviso de cookies aparece na primeira vez que o usuário entra em uma view diferente da página inicial. Uma vez que se fecha o aviso ou faz alguma ação na plataforma, o aviso é fechado e não aparece novamente.

Já o banner, aparece somente na página inicial. O banner aparece somente na versão mobile. Uma vez que se fecha este banner, ele aparece novamente ao se recarregar a página.

Figura 30 – Componente: Cookies Modal (desktop) e Banner (Mobile)

Fonte: Jungle Devs.

Figura 31 – Componente: Banner

Fonte: Jungle Devs.

A integração com a API BLiP Chat é feita nesta view. Seus detalhes serão explicados na seção 5.3.6.

Além disto, esta view também conecta-se ao *back-end*, tendo integração aos seguintes reducers:

- User: é no reducer de user que têm-se algumas informações como token de autenticação (AccountKit), propriedades que o usuário gosta e não gosta, se abriu modais, entre outros;
- Modal: para abrir e fechar o modal do BLiP Chat;
- Search: Salva resultados da busca e filtros aplicados;
- Server: Para detectar se o dispositivo usado é mobile ou não;
- SaaS: Busca informações da imobiliária no banco de dados do SaaS.

Esta integração é feita utilizando a biblioteca *React Redux*, que foi explicada no capítulo 3.

5.2.2 Página Inicial (Landing)

Na Página inicial da plataforma é o primeiro contato dos usuários com a plataforma. Esta contém algumas informações básicas sobre a imobiliária, contém um carrossel com imóveis em uma dada região padrão, e é possível realizar a busca por imóveis para alugar.

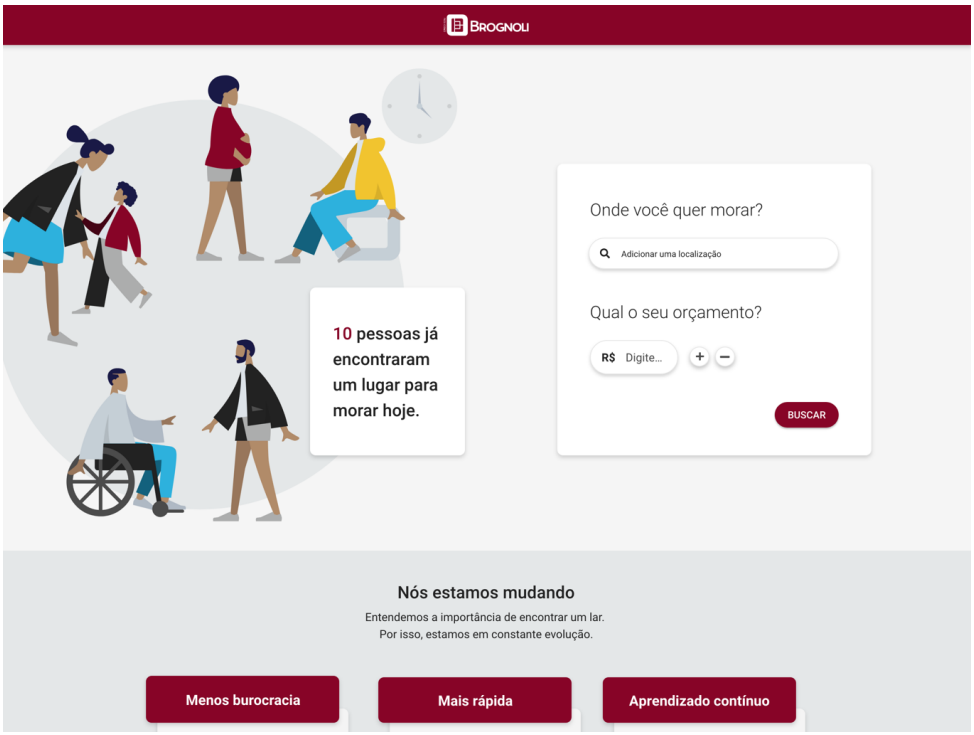
Nesta *view*, é possível usar como filtro de busca de imóveis apenas a localização e o valor do aluguel.

Na figura 32 é possível observar a parte superior da *Landing*, que contém a parte de busca. Já na figura 33, é possível observar o carrossel de imóveis.

Esta *View* também tem integração com o *back-end*, conectando-se ao reducer SaaS e *Highlights*.

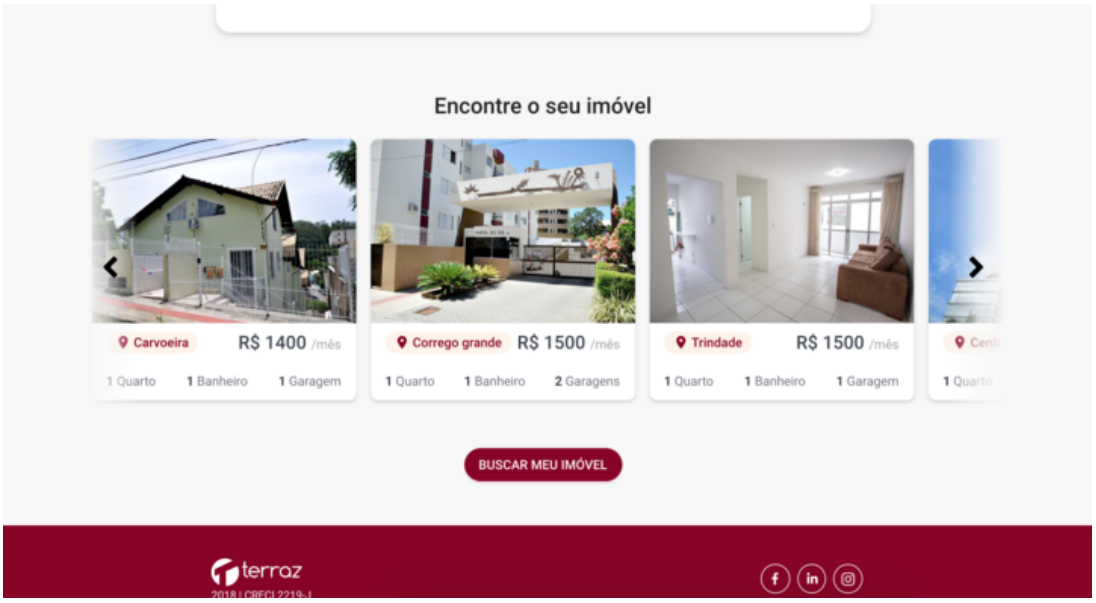
- SaaS: para buscar as informações da marca, como cores, imagens, e alguns textos;
- Highlights: são os imóveis mostrados no carrossel.

Figura 32 – View: Landing



Fonte: Jungle Devs.

Figura 33 – View: Landing



Fonte: Jungle Devs.

5.2.3 Página de Busca

A *view* de busca é onde se tem o resultado da busca por imóveis. Ela conta com uma seção onde tem os filtros para busca e outra que contém a lista de resultados.

As opções disponíveis para filtro estão apresentadas na lista à seguir:

- Valor do aluguel, onde entra-se com um valor numérico inteiro, que varia entre R$300 e R$30000;
- Localização, sendo uma *input* de texto, com integração com a API Here para busca por bairros, ruas e estabelecimentos. Pode-se pesquisar mais de uma localização por vez;
- Comércio, onde é possível selecionar estabelecimentos que o usuário deseja que tenha perto do imóvel;
- Tipo de imóvel (casa, apartamento ou tanto faz);
- Dormitórios, onde é possível selecionar entre 1 e 4;
- Garagens, onde escolhe-se quantas vagas de garagem devem ter os imóveis;
- Área do imóvel, em metros quadrados. É possível selecionar área mínima e máxima para o imóvel;
- Mobília, para buscar imóveis mobiliados ou não;
- Animais, para buscar imóveis que aceitam ou não animais.

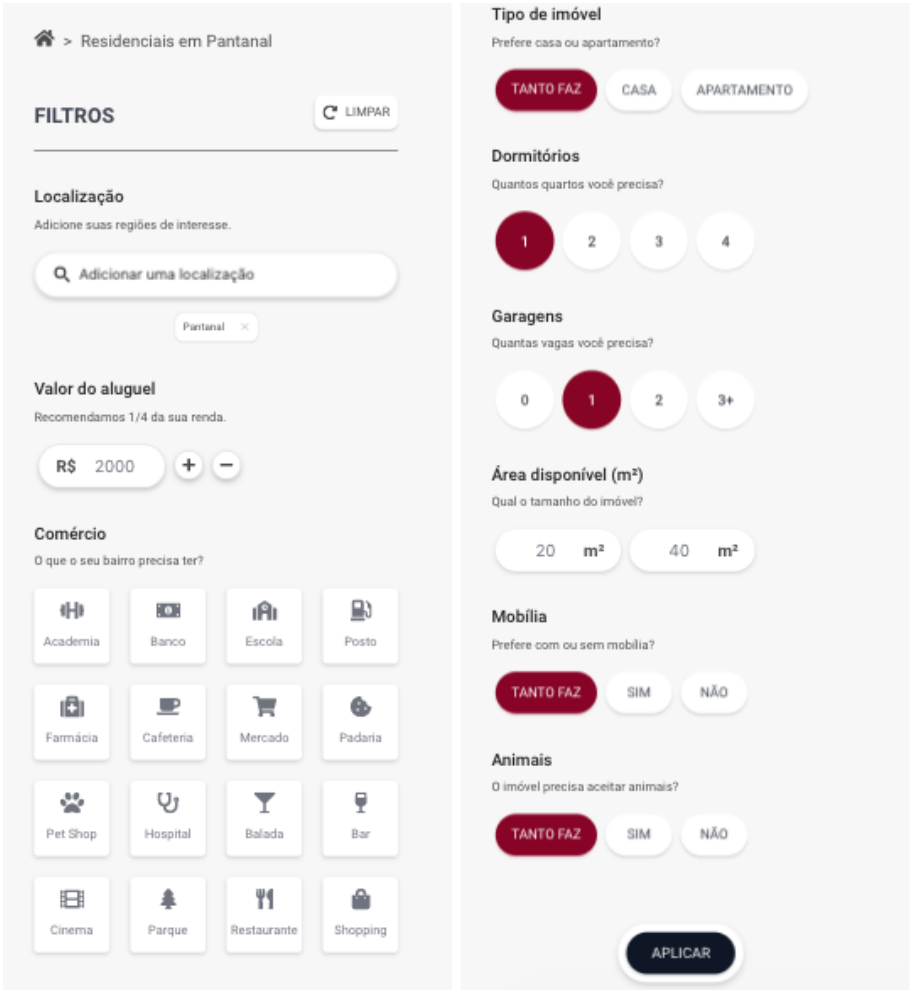
É possível observar a parte de filtros da *View* na figura 34.

A seção de filtros é apresentada de forma diferente para a plataforma desktop e para mobile. Para desktop, a seção de filtros e de resultados aparece lado a lado (figura 35. Já na versão mobile (figura 36), a seção de filtros aparece como um modal, ao selecionar um botão.

A seção de resultados apresenta os imóveis retornados pela busca, de acordo com os filtros selecionados. Ela consiste em uma lista apresentando *cards* dos imóveis, que contém um carrossel com imagens do imóvel, bairro, valor do aluguel, número de quartos, banheiros e garagens do imóvel. Na *view* para desktop, o *card* também apresenta uma breve descrição do imóvel. Ao clicar em um dos *cards* de imóvel, o usuário é redirecionado para a tela de detalhes do mesmo, que será comentada na seção 5.2.4.

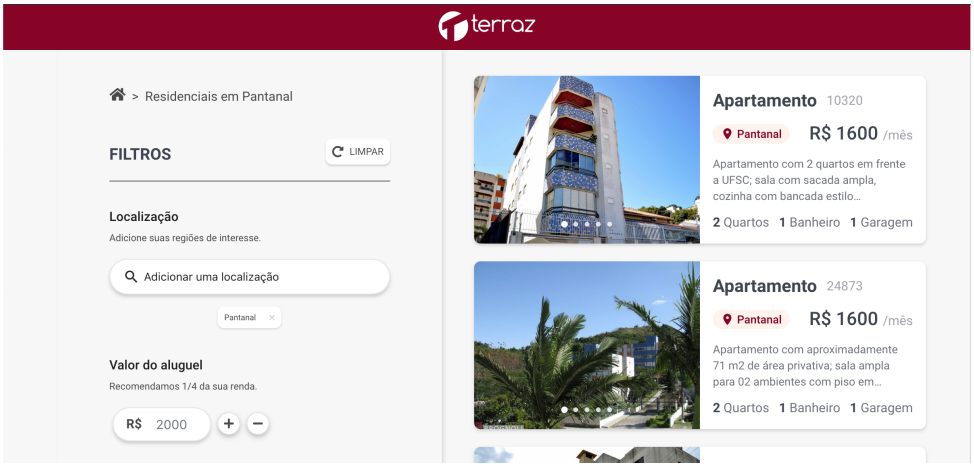
A *view* de busca apresenta dois formatos diferentes. Tinha-se interesse em testar diferentes tipos de resultados para a busca, à fim de avaliar se mostrando os resultados de

Figura 34 – View: Filtros



Fonte: Jungle Devs.

Figura 35 – View: Busca - Filtros - Desktop



Fonte: Jungle Devs.

Figura 36 – View: Busca - Modal de Filtros

Fonte: Jungle Devs.

maneira diferente para os usuários faria com que o número de agendamentos de visitas aumentasse.

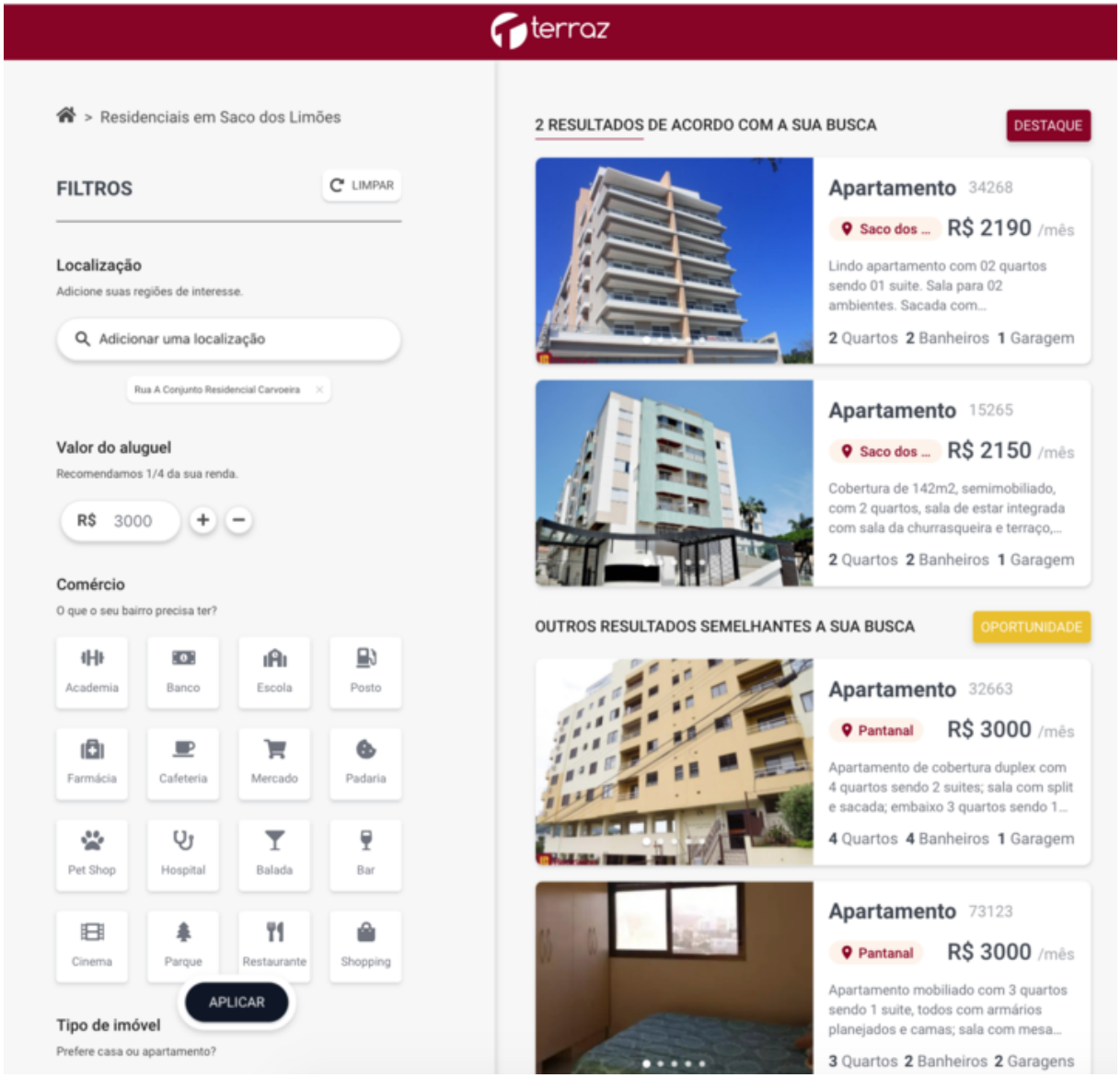
Para tal, foi implementado um teste A/B nesta tela de busca. Testes A/B tem o objetivo de testar formatos diferentes da interface, para analisar o impacto da mesma sobre a plaraforma. Isto significa que, ao entrar na plataforma pela primeira vez, o usuário seria redirecionado para uma das duas telas de busca aleatoriamente. Foi definido inicialmente que 80% dos usuários seriam redirecionados para a tela de busca B e o resto para a A. Depois de algum tempo, alteraram para que 50% dos usuários fosse redirecionado cada uma das diferentes buscas. Tal teste teve como finalidade testar se, ao apresentar os resultados de formas diferentes, teriam-se mais ou menos usuários finalizando agendamentos.

A busca A consiste em uma tela de busca que apenas lista todos os resultados

retornados pelo *back-end*. Mostra todos os *cards* de imóveis sem diferenciação. Ela está sendo mostrada na figura 35 à cima.

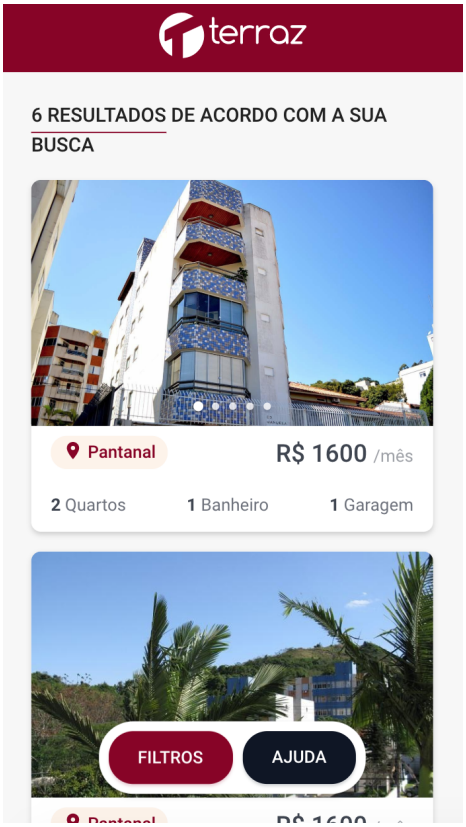
Já a busca B consiste em mostrar os *cards* de imóveis, separando em em duas seções: os resultados exatos e os resultados parciais. Isto significa que, na parte superior da busca, aparecem os imóveis que tem as características iguais às dos filtros aplicados. Já na lista de resultados parciais, estão listados todos os imóveis que tem uma combinação parcial com os filtros aplicados (como ter a mesma localização aplicada no filtro, mas número de garagens diferente, por exemplo).A versão desktop pode ser vista na figura 37 e a versão mobile pode ser vista na figura 38.

Figura 37 – View: Busca - Resultados B



Fonte: Jungle Devs.

Figura 38 – View: Busca - Resultados B (Mobile)



Fonte: Jungle Devs.

Caso a busca não retorne nenhum imóvel exato, aparece uma mensagem dizendo que não foi possível encontrar estes imóveis, e são listados, abaixo, os resultados parciais da busca. Isto pode ser visto na figura 43.

5.2.4 Página de Detalhes do Imóvel

A página de detalhes do imóvel apresenta as informações do mesmo.

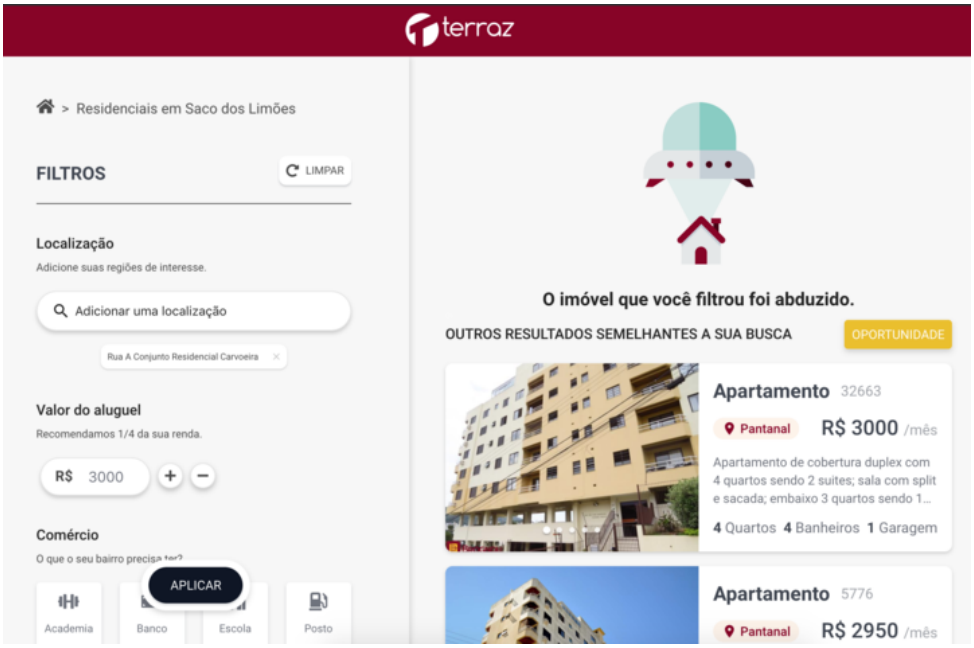
Na parte superior, é possível selecionar algumas opções, como vídeo do imóvel, fotos, que são apresentadas em um carrossel de imagens, um mapa que mostra apenas a localização do imóvel e a rua do imóvel.

Têm-se, então, o título do imóvel, bem como o seu número de *id* e o endereço do mesmo.

Aparecem as informações de preço, como aluguel, condomínio, IPTU, seguro e valor total. Esta parte aparece diferente para a versão mobile e a versão desktop da plataforma.

Para a versão mobile (figura 44), estas informações aparecem logo abaixo do endereço do imóvel. Já para a versão desktop, aparece como um *sticker*, no lado direito da

Figura 39 – View: Busca - Resultados B (Mobile)

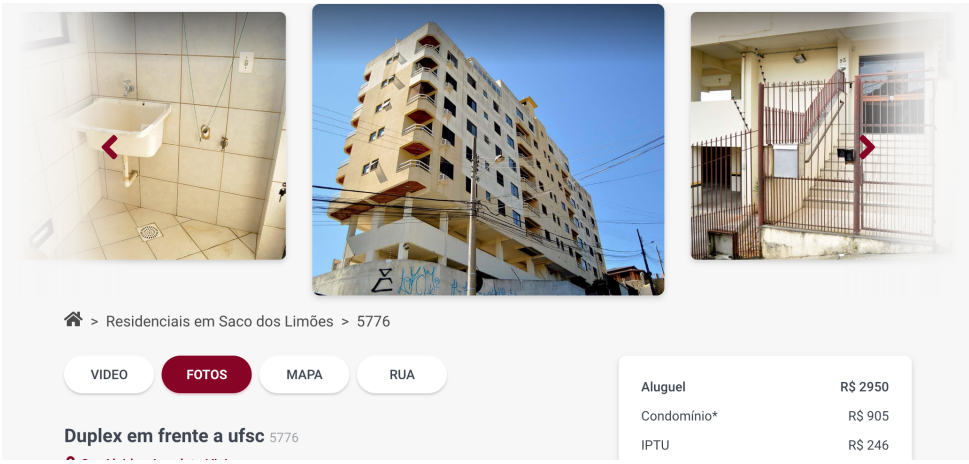


Fonte: Jungle Devs.

Figura 40 – View: Detalhes do Imóvel - Video

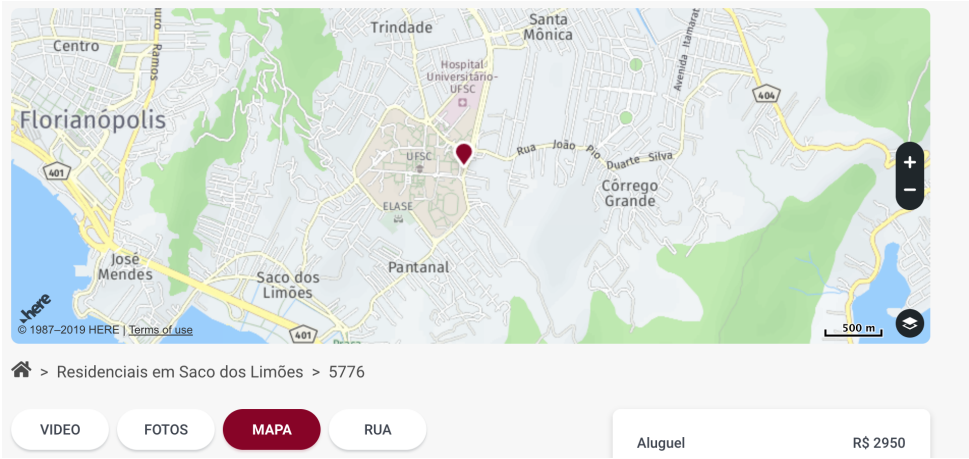
Fonte: Jungle Devs.

Figura 41 – View: Detalhes do Imóvel - Imagens



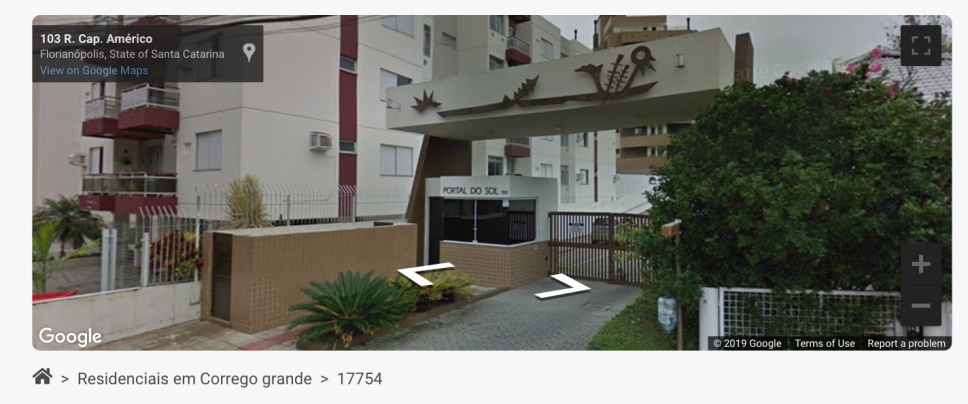
Fonte: Jungle Devs.

Figura 42 – View: Detalhes do Imóvel - Mapa



Fonte: Jungle Devs.

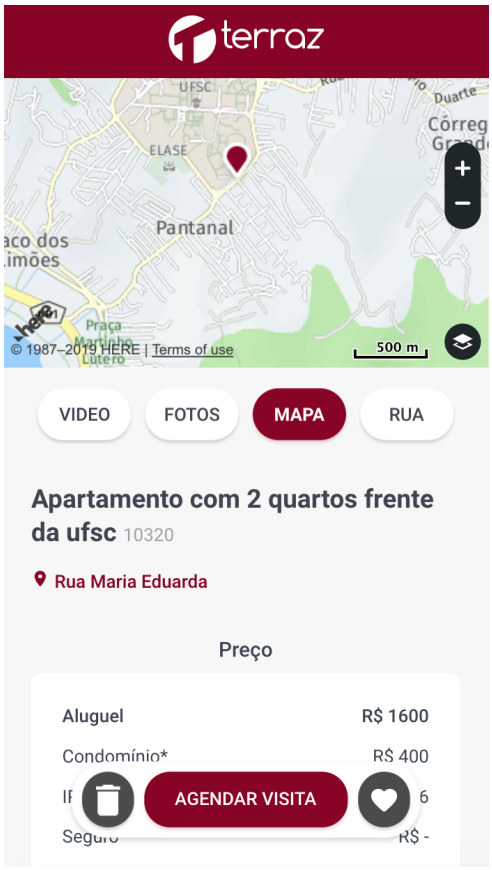
Figura 43 – View: Detalhes do Imóvel - Rua



Fonte: Jungle Devs.

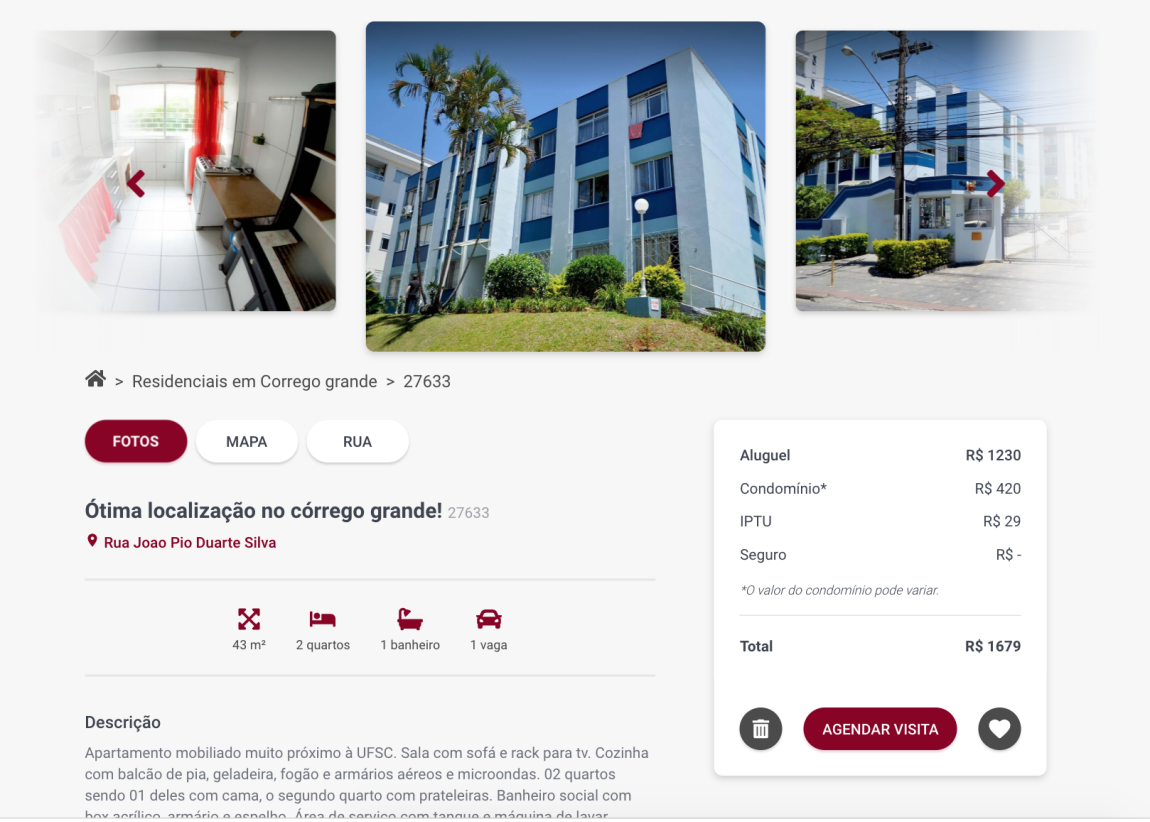
página. Isto significa que quando o usuário rola para baixo na página, estas informações rolam junto (figura 45).

Figura 44 – View: Detalhes do Imóvel Mobile



Fonte: Jungle Devs.

Figura 45 – View: Detalhes do Imóvel



Fonte: Jungle Devs.

Tem-se, então, a descrição do imóvel, contendo várias informações do mesmo, e as características, informando ao usuário a área do imóvel, número de quartos, banheiros e garagens.

Em seguida, têm-se uma seção de deslocamento, que consiste em um mapa, mostrando a localização do imóvel. Caso o usuário tenha selecionado estabelecimentos no filtro de busca, estes também aparecerão no mapa, em um raio de 1km do imóvel.

O usuário também tem a opção de fazer uma busca por trajetos neste mapa, preenchendo a *input* de busca com uma localização. O mapa, então, calcula uma rota entre o imóvel e o localização. É possível alternar o ponto de partida e chegada da rota, entre o imóvel e a localização, e alterar o meio de transporte, como sendo de carro (default), a pé, ônibus ou bicicleta.

Este mapa consiste em uma API chamada Here API, que está explicada na seção 5.3.3. Observa-se o mapa na figura 46.

Em seguida, têm-se informações necessárias para a locação do imóvel, que são padrão de imobiliária para imobiliária.

Figura 46 – View: Detalhes do Imóvel - Mapa

Fonte: Jungle Devs.

No final da *view* é apresentado um carrossel de imóveis parecidos, que são imóveis com características semelhantes ao imóvel da página em questão.

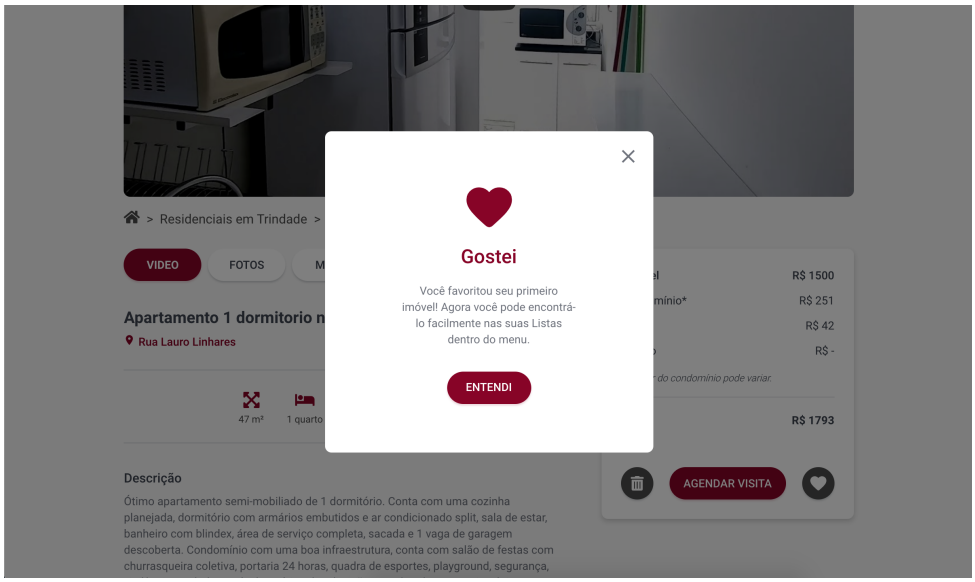
Por fim, têm a parte onde é possível fazer agendamento, gostar ou não gostar do imóvel.

Na versão desktop, esta seção fica no *sticker*, juntamente com os valores do imóvel. Já na versão mobile, esta seção fica como um *sticker* permanente na parte inferior da tela. O botão com ícone de lata de lixo adiciona o imóvel para a lista de "não gostei". Já o botão com ícone de coração irá adicionar o imóvel para a lista de "gostei". Na primeira vez que o usuário seleciona uma destes dois botões, abre-se um modal explicando qual a função dos mesmos (figuras 48 e 47). Ainda não está implementado na plataforma, mas ter-se-há uma rota onde o usuário poderá ver os imóveis que selecionou como "gostei".

É possível saber se é a primeira vez que o usuário seleciona estes botões, pois fica salvo nos *cookies* do navegador. Caso o usuário limpe os cookies, esta informação será perdida e os modais aparecerão novamente.

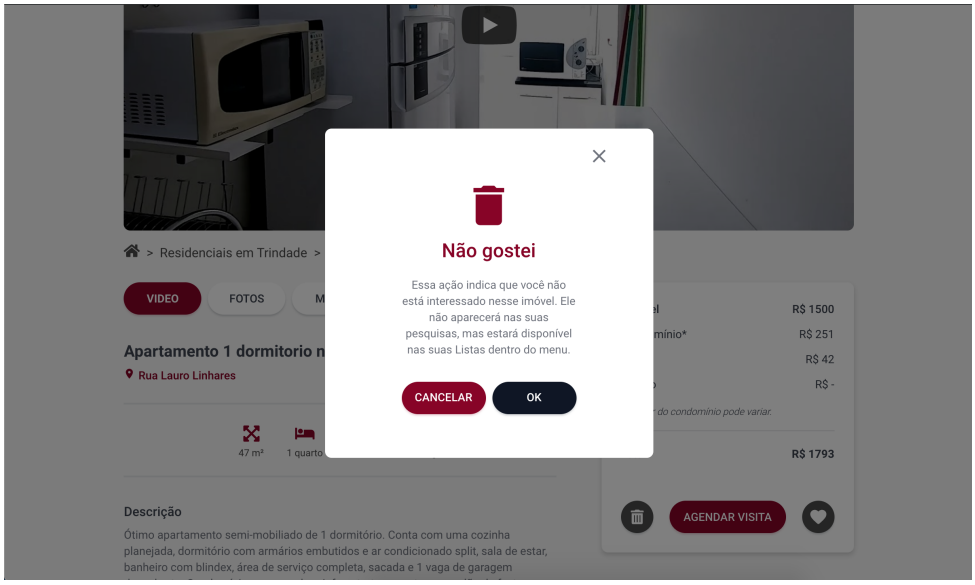
Se o usuário selecionar o botão de Agendar visita, o usuário é redirecionado para o fluxo de agendamento, que será explicitado na seção a seguir.

Figura 47 – View: Detalhes do Imóvel - Modal Gostei



Fonte: Jungle Devs.

Figura 48 – View: Detalhes do Imóvel - Modal Não Gostei



Fonte: Jungle Devs.

5.2.5 Páginas de Agendamento

O fluxo de agendamento consiste em selecionar uma data e horário para agendamento do imóvel. Ele consiste de 5 *views* diferentes, que são explicadas a seguir.

5.2.5.1 Agendamento: passo 1

No primeiro passo de agendamento, o usuário deve escolher uma data e um horário para a visita. São mostrados dois carrosséis com botões, um para a data e outro para hora (figura 49, imagem a esquerda versão desktop, direta versão mobile).

No primeiro, para data, aparece no topo o mês e ano de acordo com a data no botão que está centralizado no carrossel. Nos botões do carrossel, aparecem as datas disponíveis para agendamento. Não é possível agendar para dias anteriores ao atual, nem dias específicos definidos pela imobiliária.

O mês e ano, bem como os dias no carrossel são calculados utilizando uma biblioteca Javascript chamada moment.js [18]. Nela, é possível manipular, validar e mostrar datas e horários formatados. Ou seja, utiliza-se esta biblioteca para saber o dia da semana, para poder desabilitar as datas e horários onde a imobiliária não permita agendamento.

Após o usuário selecionar uma data e um horário, o botão de continuar é habilitado, para navegar para o próximo passo.

5.2.5.2 Agendamento: passo 2

O segundo passo do agendamento consiste em mostrar as informações do agendamento novamente. Ou seja, o usuário deve revisar se o imóvel e a data e horário selecionados estão corretos. Observa-se na figura 50.

Ao confirmar os detalhes do agendamento, o usuário é redirecionado para o próximo passo.

5.2.5.3 Agendamento: passo 3

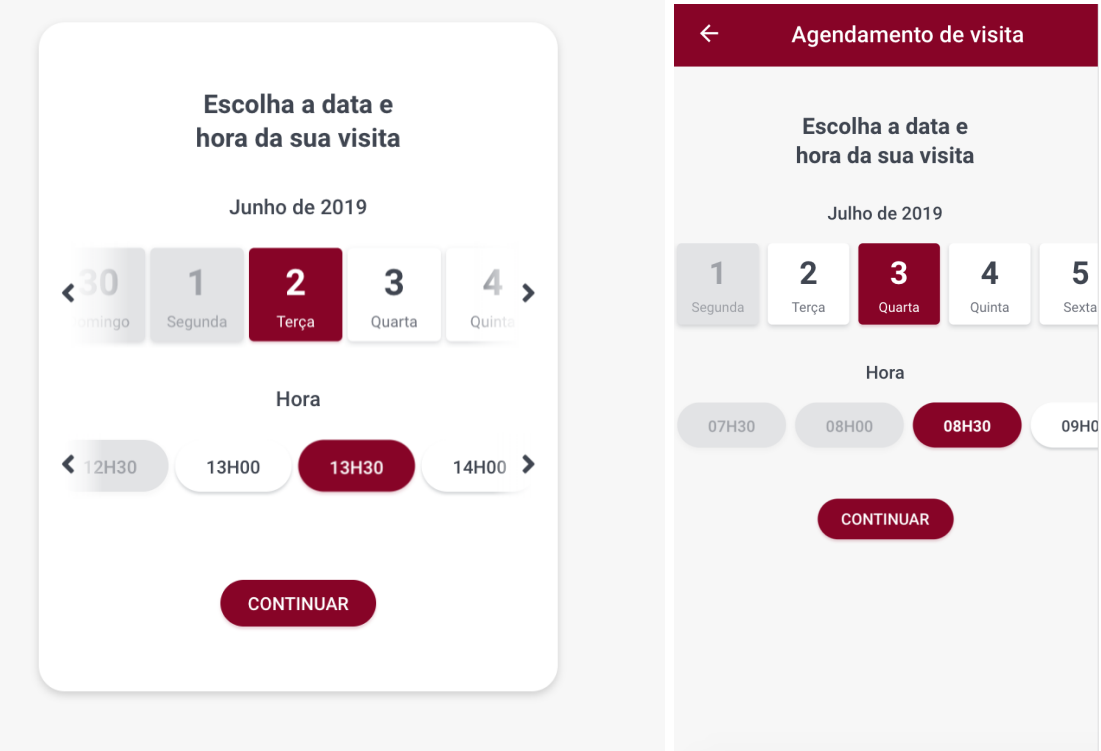
O terceiro passo de agendamento consiste no início do cadastro do usuário na plataforma, fazendo a validação do seu número de telefone (figura 51). Para tal, utiliza-se a API Accountkit, do Facebook. O funcionamento da API será explicado na seção 5.3.5.

Nesta *view*, caso o usuário já tenha feito validação pela Accountkit, ele será automaticamente redirecionado para o passo 4.

5.2.5.4 Agendamento: passo 4

Já no quarto passo, a plataforma irá pegar os dados do usuário de nome, email e CPF para o agendamento.

Figura 49 – View: Agendamento - Passo 1



Fonte: Jungle Devs.

Ao confirmar os dados, o usuário é redirecionado para o último passo.

Oberva-se a *view* na figura 52.

5.2.5.5 Agendamento: passo 5

No quinto passo, o usuário pode dar um *feedback* para a plataforma, avaliando o serviço entre 1 e 5 estrelas, bem como enviar um comentário sobre o processo de busca e agendamento de imóveis.

É possível também adicionar a data e horário do agendamento a calendários, como Apple Calendar, Google, Outlook e Yahoo.

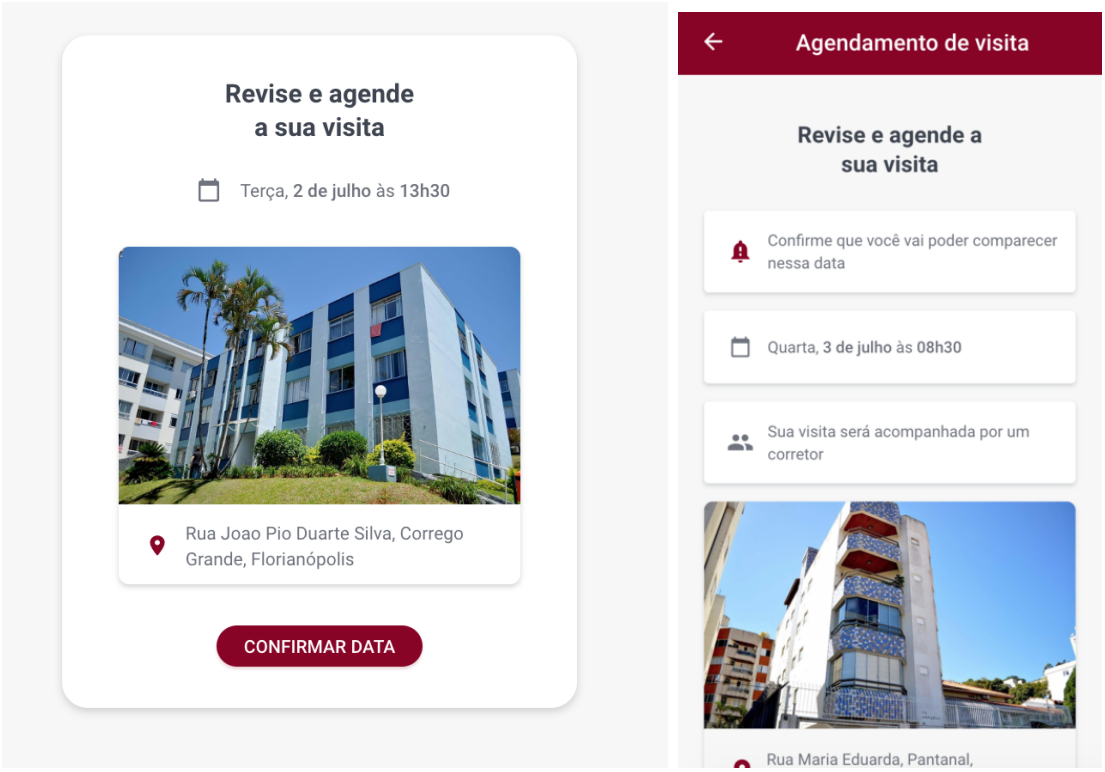
Para adicionar a data a um calendário, foi utilizada a biblioteca *react add to calendar* [19], que faz isto automaticamente, fornecendo apenas a data e horário.

A *view* pode ser vista na figura 53.

5.2.6 Termos de Uso e Política de Privacidade

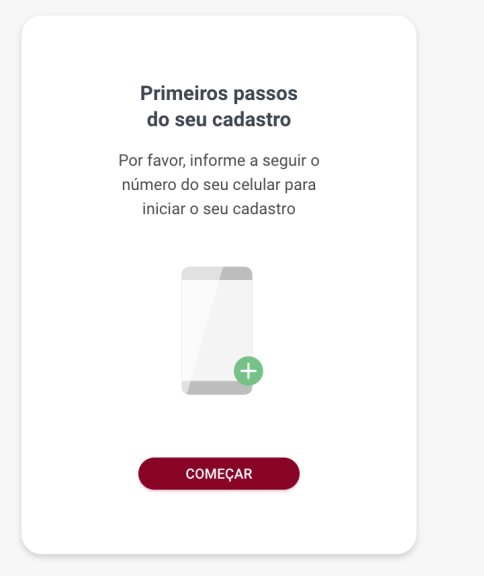
Para ambas as *views* de Política de privacidade e Termos de uso, foi utilizado um componente base para mostrar as informações.

Figura 50 – View: Agendamento - Passo 2



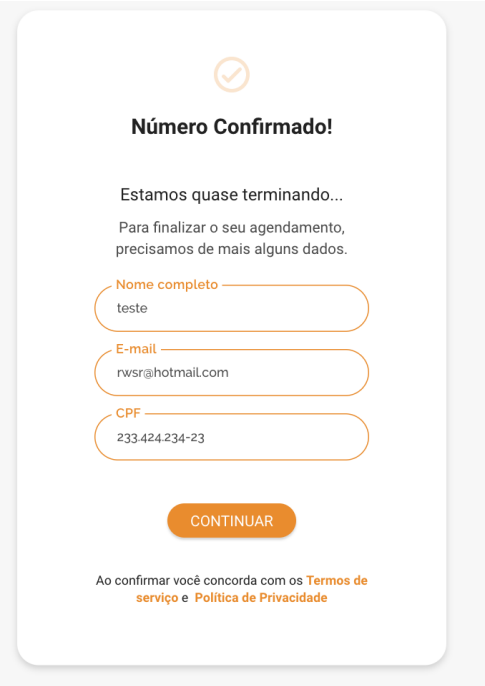
Fonte: Jungle Devs.

Figura 51 – View: Agendamento - Passo 3



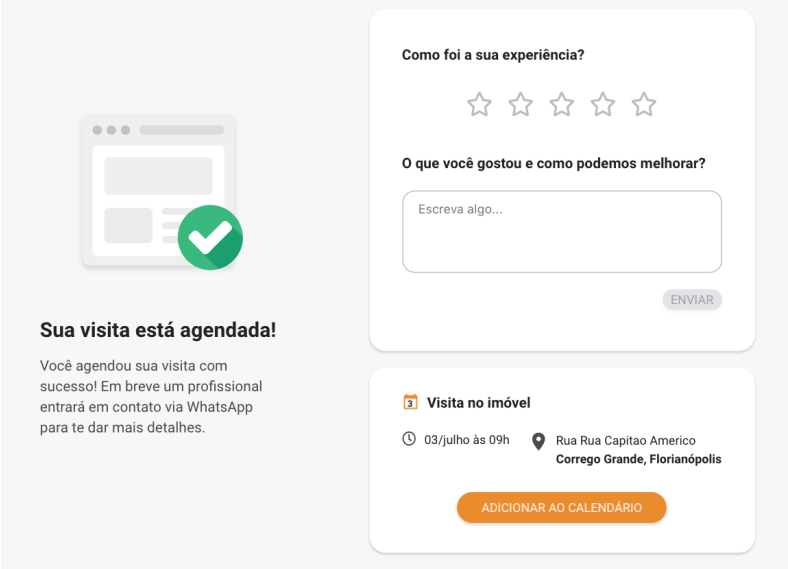
Fonte: Jungle Devs.

Figura 52 – View: Agendamento - Passo 4



Fonte: Jungle Devs.

Figura 53 – View: Agendamento - Passo 5



Fonte: Jungle Devs.

Este componente base consiste em um *wireframe*, onde coloca-se o título das seções e seus respectivos textos, e gera-se a *view*.

Na versão desktop das *views*, tem-se um menu de navegação lateral, onde pode-se rolar a página para a seção de interesse.

Observa-se na figura 54 a *view* para a Política de privacidade, na versão desktop e na figura 55 a *view* para os Termos de uso, versão mobile.

Figura 54 – View: Política de Privacidade

Fonte: Jungle Devs.

5.3 Integração com APIs de terceiros

Nesta seção, serão apresentadas e explicado a integração com APIs de terceiros na plataforma.

5.3.1 Google Tag Manager

O Google Tag Manager é uma ferramenta da Google, que torna possível instalar diversos serviços em websites, sem haver a necessidade de alterar o código da aplicação.

A integração desta API com a plataforma, foi feita utilizando uma biblioteca chamada React GTM Module [20], que inicializa o Google Tag Manager na plataforma, sendo necessário apenas o código de id da API, que é respectiva de cada imobiliária.

Após isto, o resto da configuração é feita pela própria imobiliária através da plataforma do Google Tag Manager [21].

Figura 55 – View: Política de Privacidade Mobile

Fonte: Jungle Devs.

5.3.2 Google Analytics

Google Analytics é uma ferramenta de análise da Google, que tem o intuito de monitorar o comportamento do usuário no site. É possível obter relatórios detalhados sobre a navegação pelas páginas da plataforma, armazenando dados.

A integração da API com a plataforma foi feita utilizando uma biblioteca javascript chamada React Ga [22], onde apenas inicializa-se a API utilizando o código de id da imobiliária. O resto da configuração da API é feita pela própria imobiliária, na plataforma da aplicação, [23].

5.3.3 Here Maps

A API Here [24] é uma api para mapas e localização. A API é utilizada na plataforma na parte de buscas por imóvel e nos mapas na *view* de detalhes de imóveis.

Para a busca de imóveis, utiliza-se a API para buscar os endereços que o usuário seleciona na *input* de localização. Ao digitar na *input*, a API utiliza o serviço de geocoding para buscar endereços de acordo com o que foi digitado. É possível filtrar localizações para

limitar a busca. No caso, filtra-se através das cidades onde a imobiliária atua. Assim, a API irá retornar bairros, ruas, cidades e estabelecimentos, que são utilizadas no filtro da busca.

Já nos mapas da tela de detalhes dos imóveis, utilizam-se os serviços de mapeamento, rotas e geocoding. Têm-se, pelo *back-end* as coordenadas do imóvel, que são então fornecidas à API. Com isto, instancia-se o mapa na *view* fornecendo as coordenadas. Então, cria-se um marcador para indicar a posição do imóvel no mapa.

Caso o usuário tenha selecionado estabelecimentos na busca, utiliza-se o serviço de geocoding para buscar as coordenadas de tais estabelecimentos, utilizando o serviço de Places do API, que retorna estabelecimentos no entorno de uma coordenada. No caso, utilizam-se as coordenadas do imóvel e um raio de 1km e filtram-se os estabelecimentos retornados de acordo com os filtros de busca de imóveis. Têm-se, então, as coordenadas dos estabelecimentos, que aparecem no mapa com marcadores.

Também utiliza-se o serviço de rotas do API, onde geram-se rotas do imóvel até alguma localização e vice-versa, no intuito de o usuário poder fazer rotas do imóvel até seu trabalho, por exemplo. Para tal, utiliza-se o serviço de geocoding explicado acima para buscar uma localização para gerar a rota, que o usuário preenche numa *input*, e gera-se a rota.

Para o funcionamento da API, é necessário um código Id e um Code relativo à imobiliária.

5.3.4 Google Maps

A API do Google Maps, assim como a API Here (seção 5.3.3), serve para mapas e localização.

Inicialmente, esta API foi implementada antes da API Here, fazendo a mesma função, de busca de localização, mapas e rotas. A plataforma foi ao ar por alguns dias, como uma alternativa para o site da imobiliária Brognoli. Entretanto, após algumas horas de uso, a API excedeu o limite de textitrequests, e parou de funcionar, impossibilitando a busca por imóveis. Ao analisar a API, constatou-se que o custo de utilização seria muito alto para a quantidade de *requests* que uma imobiliária grande como a Brognoli tem. Além disto, aumentar o número de *requests* que a API aceita é complicado, tendo em vista que o pagamento pela utilização da API no Brasil é feito através de terceiros.

Logo, buscou-se APIs alternativas e optou-se por utilizar a API Here na plataforma.

Entretanto, ainda utiliza-se o serviço StreetView da API do Google Maps [25]. Este fica na tela de detalhes do imóvel. Seu funcionamento é parecido com o funcionamento da API Here, onde, através das coordenadas do imóvel, instancia-se o mapa, e inicializa a

função streetview.

Para o funcionamento da API, é necessário um código Id relativo à imobiliária.

5.3.5 AccountKit

O AccountKit [26] é uma API do Facebook que faz autenticação sem senha. A autenticação é feita através de um código de identificação, enviado através de SMS ou WhatsApp para números de celular. Tal código é de uso único.

Ao clicar no botão de começar no terceiro passo de agendamento, chama-se a API, que abre uma nova janela, onde o usuário preenche seu número de telefone e seleciona se quer receber o código por SMS ou WhatsApp. O usuário é redirecionado para uma nova tela onde ele deve preencher o código recebido e confirmar. Ao fazer isso, sua autenticação está feita, e o usuário pode prosseguir com os outros passos de agendamento, sem ser necessário criar uma conta com senha para utilizar a plataforma.

Para o funcionamento da API, faz-se necessário que as imobiliárias tenham uma Id da API. Ao entrar na plataforma da API, é possível configurar as cores da imobiliária na janela que abre na hora de fazer a autenticação.

5.3.6 BLiP Chat

O API do BLiP Chat [27] consiste em uma interface na qual o usuário visualiza mensagens enviadas e recebidas de um chat bot.

Assim como as demais APIs, é necessário ter uma id da API para que a mesma funcione, respectiva a cada imobiliária.

Esta API é instanciada, na versão mobile, no rodapé (*footer*), através de um botão que abre a interface da mesma. Já na versão desktop, a interface da API é aberta através de um botão flutuante.

A configuração das respostas do chatbot é feita pelas imobiliárias.

5.3.7 Bitrix

Assim como a API BLiP Chat, a Bitrix [28] também é uma API de chat bot. Ela é instanciada através de um *snippet* com a id da imobiliária injetado diretamente no HTML da plataforma, que já instancia um botão flutuante para abrir a interface da API. Atualmente, esta API de chatbot está apenas sendo utilizada na plataforma da imobiliária Terraz, sendo a BLiP Chat a API padrão.

A configuração das respostas do chatbot é feita pela imobiliária.

5.3.8 Mixpanel

A API Mixpanel [29] é uma API que analisa o comportamento do usuário em sites e aplicativos.

É possível mandar eventos com mensagens, de acordo com o comportamento do usuário na plataforma. Ou seja, qual botão foi selecionado, qual busca foi feita, navegou para qual página, entre outros.

O intuito da utilização desta API é para averiguar se as *features* da plataforma estão sendo utilizadas, e qual a relevância das mesmas, bem como alguns fluxos dentro da plataforma.

É necessário que cada imobiliária tenha um código id para a utilização da API. Então, inicializa-se a API na plataforma, e cria-se funções para enviar as mensagens de acordo com a ação. Estas mensagens podem, então, ser analisadas na plataforma da Mixpanel.

5.4 White-Label

A plataforma havia sido feita, até então, para uma única imobiliária (Brognoli). Então, todas as cores, imagens e informações eram da mesma, não sendo possível automaticamente alterar para outra imobiliária.

Para transformar a plataforma em uma white label, foi necessário transformar tudo aquilo que iria ser customizado por imobiliárias em variável.

Inicialmente, buscou-se todos os lugares em que se utilizava a cor padrão da marca e substituiu por uma variável (no caso, *primary-color*). Então, analisou-se quais outras cores poderiam ser customizadas, e alterou todas as instâncias delas por variáveis. Esta parte foi feita em arquivos do tipo css, onde é possível utilizar variáveis , como *var(–primary-color)*.

Após isto, alteraram-se imagens utilizadas pela imobiliária, como a logo, e as imagens utilizadas nos botões da api BLipChat (seção 5.3.6), por variáveis. Estas alterações foram feitas em arquivos JavaScript.

Criou-se, então, um arquivo de configuração, que continha os valores destas variáveis. Inicialmente, continuou-se utilizando as configurações da Brognoli para testar se as alterações estavam funcionando. Ao validar que a criação de variáveis funcionava, foram criados templates, utilizando um módulo chamado *HandlebarsJS* [30]. Os templates eram arquivos JavaScript, que consistiam de uma string, contendo o código do arquivo que seria gerado, onde as variáveis eram instanciadas utilizando duplas chaves, como $\{\{primaryColor\}\}$. Foram criados três templates. Um para configurar as variáveis utilizadas em arquivos JavaScript, outra para arquivos CSS e uma última para gerar o arquivo HTML do projeto,

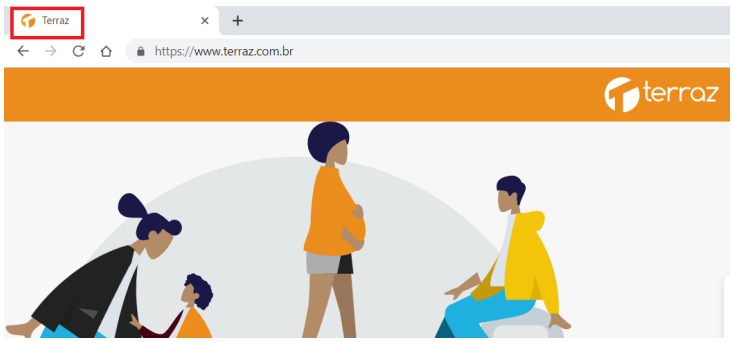
onde se fazia necessário alterar o nome do aplicativo, no caso, o nome da imobiliária, e a cor.

Foram criados, então, arquivos de configuração do tipo JSON, onde tinham-se todas as informações da imobiliária que seriam transformadas em variáveis (logo, nomes, cores, entre outros).

Para gerar tais arquivos através dos templates, criou-se um novo arquivo JavaScript chamado setup, que, compila os templates utilizando o módulo *HandlebarsJS*, trocando as variáveis do template pelos valores correspondentes no arquivos JSON de configuração, e gerando, então, os arquivos.

Este arquivo setup também era responsável por copiar as imagens de favicon (ícones que aparecem ao lado do título do website na aba do navegador) da imobiliária, da pasta de configuração da mesma, para a pasta public do projeto, onde elas devem ser instanciadas. Estas imagens consistem na imagem que aparece no navegador, do tipo .ico. Isto pode ser observado na figura 56.

Figura 56 – Favicon



Fonte: Jungle Devs.

Então, ao abrir o projeto, deveria-se rodar o arquivo setup especificando o caminho que continha os arquivos de configuração da imobiliária de interesse, e só então, rodar o projeto em si.

Automatizou-se este processo utilizando uma plataforma chamada *CircleCI*, onde se selecionava o arquivo da imobiliária de interesse no momento em que se lançava a plataforma, rodando o arquivo setup e gerando os arquivos de configuração. O projeto tem um arquivo chamado config.yml, onde é configurado o fluxo de lançamento da plataforma, e pode-se chamar comandos como o necessário para rodar o arquivo setup, entre outros essenciais para o funcionamento da aplicação.

Utilizar o *CircleCI* também resolve o problema de utilizar diferentes id de APIs de terceiro, já que cada imobiliária tem suas próprias ids, pois é possível instanciar variáveis de ambiente que são substituídas pelos valores configurados na plataforma

CircleCI. Por exemplo, para a id do Google Analytics, tinha uma variável de ambiente chamada REACT_APP_GA_ID, que no momento em que se lançava a plataforma para determinada imobiliária, iria pegar o valor salvo da id da imobiliária na plataforma CircleCI e substituir pela respectiva id. Ou seja, a imobiliária terraz teria a variável com o nome REACT_APP_GA_ID_TERRAZ, por exemplo.

5.5 Server Side Rendering

Para transformar a aplicação em um Server Side Rendering, foi necessário fazer alterações no projeto, já que inicialmente era *Client Side Rendering*.

Para isto, alterou-se os arquivos de configuração do webpack [31], criando arquivos de webpack para production, server e vendor.

Então, criou-se um arquivo chamado server, que utilizando o módulo Express [32], inicializa o servvidor com algumas configurações iniciais, como os cookies.

Primeiro, se inicializa a aplicação e criam-se as rotas dentro do aplicativo. Cria-se o estado inicial através de informações provenientes de cookies e de alguns dados do backend, como, por exemplo, informações do imóvel, para o caso da view de detalhes do imóvel.

Com o server inicializado e funcionando, criou-se, então, um componente chamado SEO (*Search Engine Optimization*), que irá definir as informações das *views* para serem exibidas. Para isto, foi utilizada uma API chamada react Helmet

Este componente foi instanciado nas views App, para conter as informações básicas da imobiliária, nas views de Busca, para aparecer quais bairros estavam na busca, e na view de detalhes do imóvel, mostrando o título, descrição e imagem do imóvel. Isto pode ser visto na figura 57 à baixo.

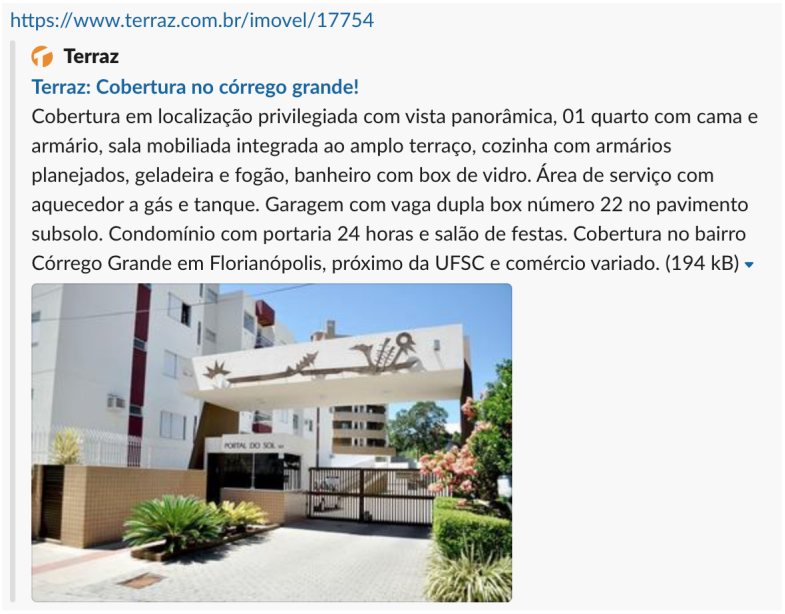
5.6 Software as a Service (Saas)

Transformando-se a plataforma em um Saas, não há mais necessidade para os arquivos de configuração de variáveis, nem rodar o arquivo de setup.

As informações de configuração das imobiliárias agora é proveniente de um banco de dados. Tal banco de dados, bem como sua configuração e instanciamento no server não serão contemplados neste documento, tendo em vista que sua implementação foi feita por um outro engenheiro da equipe.

Com tais informações no banco de dados, alterou-se o projeto para, ao invés de buscar os valores das variáveis da imobiliária nos arquivos de configuração, estes agora estavam vindo do *reducer*, chamado saas. Ou seja, diferentemente da white label, onde, caso se alterasse algum valor de variável para configuração, teria-se que lançar a plataforma

Figura 57 – Compartilhamento de link de imóvel



novamente, com as informações em um banco de dados sendo integradas em tempo real à plataforma, no momento em que fosse alterado o valor da variável no banco de dados, já altera na plataforma.

Ou seja, em todos os lugares do projeto em que eram utilizados as variáveis de configuração, alterou-se para buscar as informações integrando com o banco de dados.

Para tal, foi criado um Modelo do reducer Saas, para organizar as informações que seriam utilizadas.

Isto também alterou de onde vinham as ids para as APIs de terceiros. Agora elas não são mais fornecidas pelo CircleCI, mas estão no banco de dados da imobiliária.

5.7 Acessibilidade

Um último ponte de interesse para a plataforma era torná-la acessível para todos os usuários.

Acessbilididade na web significa que pessoas com deficiência podem navegar livre e facilmente pela web [33].

Desde 2016, existe uma lei no Brasil chamada de Lei Brasileira de Inclusão que requer que todos os sites sejam acessíveis.

Tendo em vista que a plataforma tem foco no aluguel de imóveis, é de interesse atingir o maior número possível de pessoas.

Existem alguns padrões a serem seguidos para facilitar a acessibilidade em páginas

web. Um destes padrões é adicionar *labels* e *alts* em botões, imagens e entradas de texto, para que usuários com deficiência visual possam navegar pela plataforma utilizando programas de voz (*Voice Over*).

Há também ARIA ou WAI Aria (Accessible Rich Internet Applications, Web Acessibility Initiative) [34], onde é possível utilizar aria-label em componentes, para especificar sua função, ou aria-hidden, para esconder algun componente de dispositivos de navegação por som (voice Over).

Outro padrão a ser seguido é manter o *Outline* em botões, links e entradas de texto, para pessoes com deficiência motora poderem navegar na plataform através do teclado. O *outine* foi desabilitado na plataforma para eventos de Mouse, mas aparece quando navegando pelo teclado (pela tecla *Tab*, por exemplo). Isto pode ser observado na figura 58, onde observa-se o *Outline* azul ao redor do botão na *Header*.

Figura 58 – Acessibilidade - Outline

Uma ferramenta utilizada para testar a acessibilidade na plataforma foi utilizando a *Audits*, que é uma ferramenta do Google Chrome que avalia o nível de acessibilidade de uma página web. Utilizou-se a ferramenta para obter notas em cada *view*, e torna-la acessível, seguinto as sugestões que aparecem no resultado da avaliação.

O foco da acessibilidade na plataforma foram usuários com deficiências motora e visual. Vale ressaltar que pessoas com deficiências auditivas também tem dificuldade em utilizar websites e aplicativos, uma vez que estas se comunicam através da linguagem Libras, que tem uma estrutura diferente da lingua portuguesa. Deve-se levar em conta que muitas pessoas com tal deficiência não dominam completamente a língua portuguesa,assim tendo dificuldade para compreender textos escritos em português. Entretanto, para tal, existem ferramentas e aplicativos externos que traduzem textos do português para a linguagem de sinais Libra, de forma que não é necessário nenhuma adaptação da plataforma para tal. Um aplicativo utilizado para fazer esta tradução é o Hand Talk [35], que utiliza uma interface onde um mascote (Hugo, feito em animação) faz os gestos da linguagem Libras para o usuário, traduzindo, então, o conteúdo.

6 Análise dos Resultados

Até o momento em que este relatório foi redigido, a plataforma foi lançada temporariamente como alternativa para o site da imobiliária Brognoli, mas já saiu do ar. Foi lançada oficialemente, também, como uma White-Label, para a imobiliária Terraz (www.terraz.com.br).

6.1 Análise dos Requisitos Funcionais

Será analisado o cumprimento dos requisios inicialmente levantados no capítulo 4 deste documento.

O sistema eficientemente faz a busca por imóveis de acordo com os filtros.

De todos os filtros apresentados, alguns ainda não estão implementados, pois faltam as informações nos imóveis das imobiliárias. No caso, não estão sendo aplicados os filtros de tipo de imóvel, nem quantidade de animais permitidos.

Entretanto, tanto a busca tipo A como B funcionam.

O requisito de buscar as informações de um imóvel específico também foi cumprido, mostrando todas as informações necessárias. É possível adicionar um imóvel à lista de "gostei"e à lista de "não gostei".

O requisito de agendamento de visita também foi cumprido, com todo o fluxo de agendamento funcionando. Este facilitou o processo de agendamento de visitas, que era feito por agentes imobiliários, que tinham que entrar em contato com os clientes para agendar a visita.

O requisito de cadastro de usuário também foi cumprido. Após a validação do usuário pela API AccountKit, o sistema pede informações de nome, e-mail e CPF do usuário para terminar o cadastro e concluir o agendamento. Com estas informações, a imobiliária tem informações sobre o usuário.

6.2 Análise de utilização

Quando a plataforma foi ao ar temporariamente como um link alternativo do site da imobiliária Brognoli, foi possível validar brevemente a plataforma, tendo em vista que alguns usuários utilizaram a plataforma para fazer buscas, e alguns chegaram a concluir o fluxo de agendamento de visita ao imóvel.

Com o site no ar, foi possível perceber que utilizar a API do Google Maps para

fazer as buscas por endereços e mostrar os mapas na tela de detalhes do imóvel teria um custo muito alto. Por este motivo, a API foi substituída nestes pontos pela API Here, que tem funcionalidade similar e custo reduzido em 4 vezes. A *feature* de mostrar a rua do imóvel não tem muitos acessos, logo, não faz muitos requerimentos a API. Então, optou-se por manter utilizando a API do Google Maps para esta funcionalidade.

Após a plataforma ir ao ar como uma *White-Label*, através da imobiliária Terraz, foi possível validar melhor a plataforma.

Entretanto, analisando os acessos e a movimentação dos usuários dentro da plataforma, observou-se que em torno de 70% dos usuários apenas entra na página inicial do aplicativo, sem realizar nenhuma busca ou navegar para outras páginas.

Para tentar entender o motivo disto, adicionou-se a ferramenta *Hotjar* [36], que rastreia os movimentos de mouse do usuário. Até o momento de redação deste documento, ainda não foram levantados motivos, mas pensa-se em atualizar o design da página inicial para torná-la mais amigável e fácil de navegar.

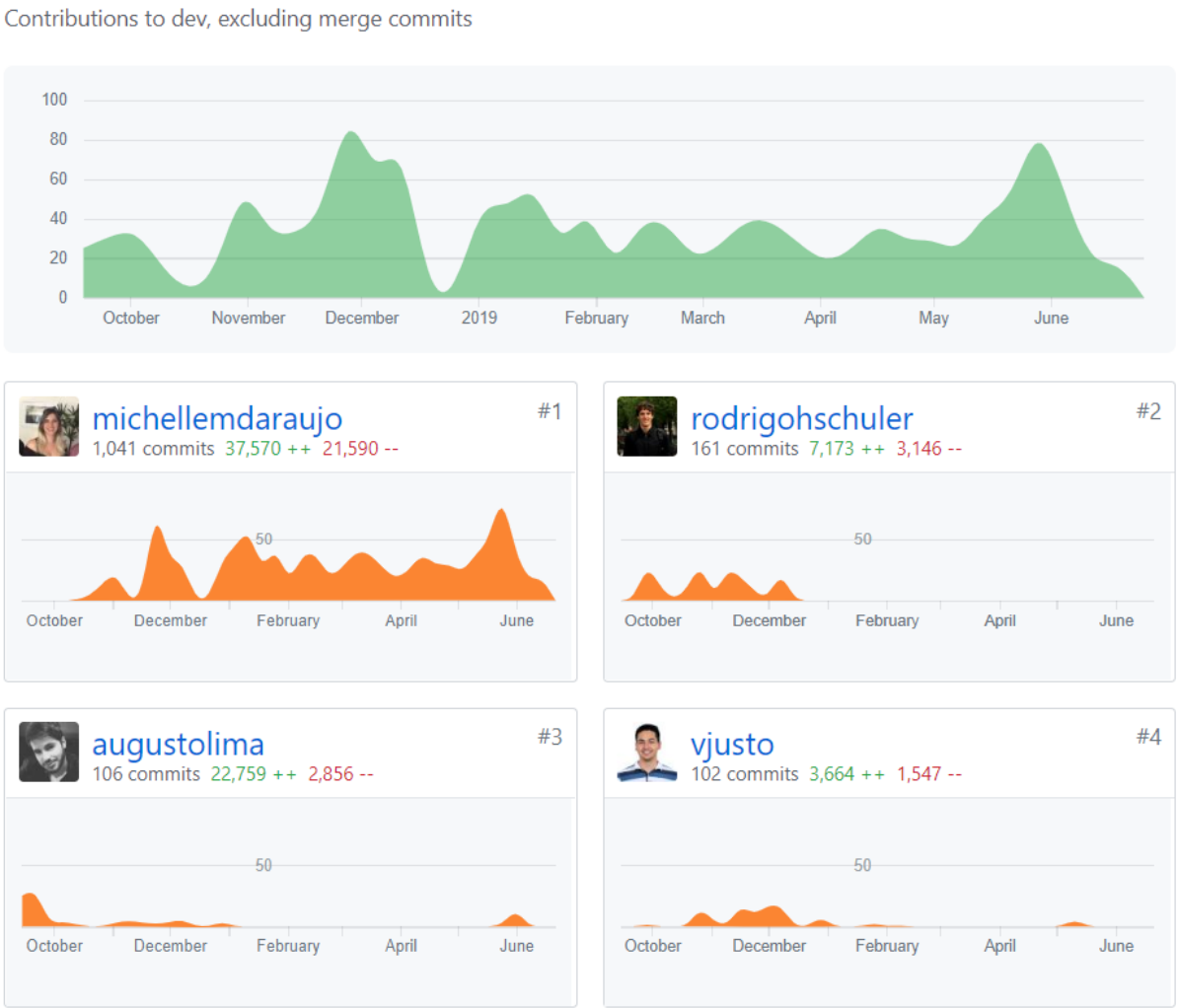
6.3 Validação do Programa Academy

Pelo fato de tanto a empresa Jungle Devs como o programa Academy terem pouco mais de um ano, o desenvolvimento deste projeto funcionou como forma de validação para o programa. A autora deste documento foi a primeira pessoa a concluir o programa Academy voltado para Web.

Além de validar o programa Academy, o trabalho apresentou bons resultados referentes à outras etapas do programa. Isto pode ser observado através da contribuição prática da autora para o projeto. Analisando os dados disponíveis no repositório do projeto para a parte de *front-end* na empresa, conclui-se que a autora teve contribuições significativas para o projeto. Oberva-se, na figura 59, que a autora foi responsável por 1041 commits no repositório.

Conclui-se, então, que as etapas iniciais do programa foram importantes e efetivas no processor de aprendizagem de conceitos práticos de projeto.

Figura 59 – Contribuições para o repositório do projeto



7 Conclusões e Perspectivas

O objetivo deste projeto foi o desenvolvimento da parte de front end de um SaaS para imobiliárias. Fazendo uso de métodos ágeis para desenvolvimento de projetos, foi possível desenvolver a plataforma até chegar em um ponto em que a mesma pudesse ser lançada oficialmente por uma imobiliária. Através da integração com o back end e com APIs de terceiros, foi possível implementar as funcionalidades essenciais do projeto.

As contribuições da autora para o projeto foram de suma importância para o desenvolvimento da plataforma. A participação neste projeto e o trabalho realizado dentro da empresa serviram de grande aprendizado. Foram desenvolvidas diversas habilidades técnicas que serão úteis para a vida profissional, bem como habilidades pessoas, como trabalho em grupo, gerenciamento de tempo, entre outros.

Ainda não é possível medir com exatidão o impacto desta plataforma no setor imobiliário, tendo em vista que só foi oficialmente ao ar em uma imobiliária. Porém, é de grande valia a possibilidade de se ter um serviço para imobiliárias digital, integrando diversos serviços e permitindo a busca e agendamento de visitas à imóveis.

O projeto ainda está longe de ser concluído, tendo em vista que está em desenvolvimento a parte de acesso do usuário e o fluxo de contratos automatizados na plataforma.

Até o momento de redação deste documento, já foram implementadas algumas *views* do usuário, como uma *view* mostrando os agendamentos realizados pelo usuário.

Como perspectiva futura, seria interessante utilizar métodos de inteligência artificial e machine learning para aprimorar a recomendação de imóveis para os usuários, baseado nas interações dos usuários com a plataforma.

Por fim, vale ressaltar que a autora enfrentou diversos desafios ao longo deste projeto. Por começar com integrações em APIs, implementando diversas funcionalidades das mesmas, como as integrações feitas com as APIs do Google Maps e Here.

Além do mais, transformar a plataforma em uma white label proporcionou a busca de conhecimentos para a criação do script de setup para aplicar a marca de imobiliária na plataforma, bem como a utilização de templates para geração de arquivo.

Também foi necessário adquirir conhecimentos sobre Server Side Rendering, tendo em vista que esta foi a primeira vez que a autora participou de um projeto que utiliza esta tecnologia, já que os outros projetos eram em Client Side Rendering.

Além disto, este foi o primeiro projeto que a autora participou em que tanto as tomadas de decisões quando o design da plataforma são feitos por um cliente externo

à empresa. Isto apresenta alguns desafios na comunicação, entre alinhar objetivos e até mesmo responder dúvidas relacionadas ao projeto.

Conhecimentos adquiridos no decorrer do curso de Engenharia de Controle e Automação foram de suma importância para o trabalho realizado, como a disciplinas de metodologia de projetos e disciplinas que envolviam o desenvolvimento de código para conclusão de projetos. Uma outra disciplina que ajudou muito para o entendimento do problema foi a disciplina de Integração de sistemas corporativos, uma vez que foi feita a integração com diversas APIs externas, e uma vez que o próprio sistema consiste em um SaaS.

Entretanto, seria interessante o curso prover disciplinas alternativas como optativas, mais voltadas para a área de informática e desenvolvimento de software e aplicativos, uma vez que todas as linguagens de programação utilizadas neste projeto foram aprendidas dentro da empresa, e não no escopo do curso. Um outro ponto interessante seria introduzir os alunos à diferentes metodologias de desenvolvimento de projeto, como métodos ágeis, já que estão cada vez mais sendo utilizadas dentro de empresa, em contrapartida com o método cascata que é a mais ressaltada no contexto da disciplina de Metodologia para Desenvolvimento de Sistemas.

Por fim, a empresa Jungle Devs proporcionou um ambiente ideal para o aprendizado e reforço de conceitos anteriormente adquiridos, fornecendo todo o apoio necessário para o desenvolvimento, tanto do projeto quanto do desenvolvimento pessoal da autora.

Referências

1 INOVAçãO imobiliárias. Disponível em: <https://exame.abril.com.br/negocios/dino/startup-mineira-benvenuto-muda-a-experiencia-na-compra-e-venda-de-imoveis-dino89092912131/>. Citado na página 17.

2 MéTODOS Ágeis. Disponível em: <http://www.metodoagil.com/metodos-ageis/>. Citado na página 21.

3 WHAT is Scrum. Scrum. Disponível em: <https://www.scrum.org/resources/what-is-scrum>. Citado na página 21.

4 O básico: O que é HTML?. Tableless. Disponível em: <https://tableless.com.br/o-que-html-basico/>. Citado na página 25.

5 TUTORIAIS. w3schools. Disponível em: <w3schools.com>. Citado na página 26.

6 O que é JavaScript?. Developer Mozzila. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Learn/JavaScript/>. Citado na página 26.

7 REACT: o que é e como funciona essa ferramenta?. Udacity. Disponível em: <https://br.udacity.com/blog/post/react-o-que-e-como-funciona>. Citado na página 27.

8 INICIANDO com Redux em 9 passos. Medium. Disponível em: <https://medium.com/reactbrasil/iniciando-com-redux-c14ca7b7dcf>. Citado na página 28.

9 SMARTY Ads. Disponível em: <https://smartyads.com/blog/what-does-white-label-mean-in-business/>. Citado na página 31.

10 MEU positivo. Disponível em: <https://www.meupositivo.com.br/panoramapositivo/saas/>. Citado na página 31.

11 CLIENT Side Rendering. Disponível em: <http://storylens.com/@manjunath/server-side-vs-client-side-rendering---an-in-depth-overview>. Citado na página 32.

12 SERVER Side Rendering. Disponível em: <https://medium.com/@baphemot/whats-server-side-rendering-and-do-i-need-it-cb42dc059b38>. Citado na página 32.

13 WHAT is Git. Atlassian. Disponível em: <https://br.atlassian.com/git/tutorials/what-is-git>. Citado na página 33.

14 O que é GitHub. Oficina da Net. Disponível em: <https://www.oficinadanet.com.br/post/14791-o-que-github>. Citado na página 33.

15 JIRA. Atlassian. Disponível em: <https://br.atlassian.com/software/jira>. Citado na página 34.

16 CLIENT Side Rendering. Disponível em: <https://circleci.com/>. Citado na página 34.

17 UMA visão geral do HTTP = MDN Web Docs. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Overview>. Citado na página 43.

18 MOMENT.JS. Disponível em: <https://momentjs.com/>. Citado na página 63.

19 REACT add to calendar. Disponível em: <https://github.com/jasonsalzman/react-add-to-calendar>. Citado na página 64.

20 REACT GTM. Disponível em: <https://github.com/alinemorelli/react-gtm#readme>. Citado na página 67.

21 GOOGLE Tag Manager. Disponível em: <https://www.google.com/intl/pt-BR/tagmanager/>. Citado na página 67.

22 REACT GA. Disponível em: <https://github.com/react-ga/react-ga>. Citado na página 68.

23 GOOGLE Analytics. Disponível em: <https://analytics.google.com/analytics/web/provision/?authuser=0#/provisiona>. Citado na página 68.

24 HERE API Documentation. Disponível em: <https://developer.here.com/documentation>. Citado na página 68.

25 GOOGLE Maps Documentation. Disponível em: <https://developers.google.com/maps/documentation/?hl=pt-brn>. Citado na página 69.

26 ACCOUNT Kit Documentation. Disponível em: <https://developers.facebook.com/docs/accountkit/>. Citado na página 70.

27 BLIP Chat API. Disponível em: <https://help.blip.ai/hc/pt-br>. Citado na página 70.

28 BITRIX API. Disponível em: <https://www.bitrix24.com/apps/dev.php>. Citado na página 70.

29 PLATAFORMA Mixpanel. Disponível em: <https://mixpanel.com/>. Citado na página 71.

30 HANDLEBARS js. Disponível em: <https://handlebarsjs.com/>. Citado na página 71.

31 WEBPACK - ReactJS.net. Disponível em: <https://reactjs.net/bundling/webpack.html>. Citado na página 73.

32 EXPRESS. Disponível em: <https://expressjs.com/pt-br/>. Citado na página 73.

33 ACESSIBILIDADE na Web - Handtalk. Disponível em: <http://blog.handtalk.me/acessibilidade-na-web/>. Citado na página 74.

34 WAI ARIA - W3. Disponível em: <https://www.w3.org/WAI/standards-guidelines/aria/>. Citado na página 75.

35 HAND Talk. Disponível em: <https://www.handtalk.me/>. Citado na página 75.

36 HOTJAR. Disponível em: <https://www.hotjar.com/>. Citado na página 78.