STUDENT HOUSING: APLICAÇÃO PARA ALUGUEL DE QUARTOS E IMÓVEIS PARA ESTUDANTES

Artur Silvestre de Oliveira
Graduando em Engenharia de *Software* - Uni-FACEF
artur.silvestreo@gmail.com

Larissa Viana Macedo
Graduanda em Engenharia de *Software* - Uni-FACEF
larissavianamacedo5@gmail.com

Daniel Facciolo Pires
Doutor em Física Médica
daniel@facef.br

Resumo

Dentre os benefícios que a tecnologia da informação e comunicação proporciona, destacam-se as aplicações *web* e *mobile* que visam proporcionar facilidade e praticidade aos usuários. Este artigo tem como objetivo apresentar o desenvolvimento de uma aplicação *web* que auxilia estudantes na busca por moradias estudantis, especialmente, quartos vagos, e por outro lado, apoia aqueles que desejam alugar imóveis disponíveis, preferencialmente, quartos disponíveis. O usuário do tipo estudante poderá procurar por quartos ou casas para alugar, de acordo com a categoria desejada e efetuar a reserva do aluguel. O usuário do tipo proprietário pode oferecer seus quartos ou imóveis para que fiquem disponíveis para alugar, autorizando a locação. Foram utilizadas técnicas e métodos de desenvolvimento para aplicações *web*, técnicas de elicitação de requisitos e prototipação, criando artefatos da engenharia de *software*.

Palavras-chave: Aplicações *web*. Moradia de estudantes. Aluguel de quartos. Engenharia de Software

Abstract

Among the benefits that information and communication technology provide, web and mobile applications stand out, which aim at providing ease and convenience to users. This article aims at presenting the development of a Web application that helps students in their search for student accommodation, especially vacant rooms, and on the other hand, supports those who wish to rent available properties, preferably available rooms. The student type user will be able to search for rooms or houses to rent, according to the desired category, and to book the rent. The owner type user can offer their rooms or properties to be available for rent, authorizing the lease. Techniques and methods of development for Web applications were used, such as

requirements elicitation techniques and prototyping, creating software engineering artifacts.

Keywords: Web application. Student Housing. Room Rent. Software Engineering.

1 Introdução

Anualmente, estudantes se mudam para outras cidades em busca de formação acadêmica, e não conhecem ou não possuem contato com pessoas que lá residem, e consequentemente encontram dificuldades para encontrar uma casa, um apartamento ou, principalmente, um quarto para alugar. Por outro lado, existem proprietários de imóveis, que desejam locar para estudantes, ou ainda também estudantes que alugam imóveis e precisam de colegas para dividir custos, já que podem ter quartos vagos.

O cenário descrito é percebido pelos autores deste trabalho, que acompanham os desafios, dificuldades e inseguranças de pessoas que já precisaram se mudar para outra cidade para estudarem, ou até trabalharem, e não conhecem ninguém ao chegar na cidade de destino, sem conhecimento de uma boa localização, com pontos de referência, contato de responsáveis de imóveis e estado do imóvel.

Ao realizar pesquisas de aplicativos que atendessem a necessidade de estudantes que precisam alugar um quarto ou moradia, identificamos o aplicativo Airbnb, que atua como um mercado online de hospedagem para aluguel de temporada e atividades de turismo, no entanto este não é exclusivo para estudantes que procuram por um ambiente voltado para estudo.

Ao entrevistar informalmente estas pessoas, percebeu-se que um *software* pode apoiar as atividades de busca e disponibilização de vagas de quartos em imóveis de estudantes.

Neste contexto, surgem os problemas de pesquisa deste trabalho: quais as técnicas, ferramentas e metodologias da Engenharia de *Software* que permitem criar uma solução que apoie estudantes ou proprietários que precisam alugar quartos e aqueles estudantes que procuram por quartos?

Assim, o objetivo é desenvolver uma aplicação *web* para aluguéis de cômodos vazios em imóveis de estudantes que procuram e que oferecem esta vaga, na qual muitos deles se deslocam para outras cidades para estudarem e com isso precisam alugar um imóvel em bom estado, boa localização e seguro, para estudantes que possuem quartos vagos que desejam alugar para dividirem as despesas, ou proprietários que querem locar para estudantes.

Dentre as diversas funcionalidades, o sistema disponibiliza aos estudantes a opção de consultar quartos e imóveis, visualizando detalhes como os dados do responsável pelo anúncio, fotos do local, endereço com localização e as especificações, assim como poderão publicar quartos e imóveis disponíveis.

Outra funcionalidade do sistema é realizar a reserva do quarto ou imóvel, que deverá ser aprovada ou recusada pelo responsável pelo anúncio e posteriormente o estudante receberá uma notificação informando sobre o status da reserva solicitada.

O nome da aplicação foi escolhida como “*Student Housing*", que traduzido para o português significa “Moradia de estudantes”, pelo fato de ser exclusivo para estudantes que procuram quartos e imóveis para alugar ao mudarem de cidade para estudarem.

Dentre os vários benefícios do aplicativo temos: uma aplicação voltada apenas para estudantes, para divulgação de quartos e imóveis, e ainda oportunidades de estudantes ou não de obterem uma renda extra com cômodos vagos e a facilidade em entrar em contato com o responsável do imóvel.

O aplicativo proposto neste trabalho não tem a intenção de formalizar e nem judicializar contratos de aluguel, fugindo portanto do escopo do projeto. O ambiente foca em facilitar a busca e divulgação de vagas de imóvel ou quartos.

Foram elaboradas entrevistas, análise, modelagem, prototipação e implementação. Como metodologia utilizou-se os conceitos da metodologia ágil Scrum, unida a métodos e metodologias de Engenharia de *Software*, como BPMN, Diagrama de Caso de Uso, Documento de Requisitos, 5W1H, Testes, Gerenciamento de projetos e normas do PMBOK. O Figma como tecnologia de prototipação, o Canvas como padrões de projetos e a entrevista com estudantes por meio do Google *Forms*, são algumas das ferramentas utilizadas.

Para a etapa de desenvolvimento utilizou-se do *GitHub* como versionamento de código para verificarmos cada alteração durante o processo desenvolvimento, *Stack* de *Javascript* para *FrontEnd* com *ReactJS*, *BackEnd Node.js*, *PostgreSQL* com *TypeORM* para armazenar os dados, *Visual Studio Code* como editor de código e Insomnia para fazer as requests entre *Front* e *Back*.

2 Referencial Teórico do Projeto

Nesta seção aborda-se os referenciais teóricos sobre conceitos das metodologias e conceitos do desenvolvimento que foram utilizados para o planejamento da construção da aplicação.

2.1 Metodologia Ágil

“As metodologias ágeis são parte da Engenharia de *Software*, a área de conhecimento voltada para a especificação, desenvolvimento e manutenção de sistemas de *software*” (BALLE, 2011, p.13).

Na década de 1990, após a insatisfação de inúmeros desenvolvedores sobre as abordagens pesadas de engenharia de *software*, foram propostos novos métodos ágeis, segundo Sommerville (2011), que permitiram que a equipe de desenvolvimento se concentrasse no *software* em si e não na documentação e concepção.

> Métodos ágeis, universalmente, baseiam-se em uma abordagem incremental para a especificação, o desenvolvimento e a entrega do *software*. Eles são mais adequados ao desenvolvimento de aplicativos nos quais os requisitos de sistema mudam rapidamente durante o processo de desenvolvimento. Destinam-se a entregar o *software* rapidamente aos clientes, em funcionamento, e estes podem, em seguida, propor alterações e novos requisitos a serem incluídos nas iterações posteriores do sistema. Tem como objetivo reduzir a burocracia do processo, evitando qualquer trabalho de valor duvidoso de longo prazo e qualquer documentação que provavelmente nunca será usada (SOMMERVILLE, 2011, p.40).

No quadro 1 estão listados um conjunto de princípios dos métodos ágeis com base no manifesto ágil, descrito por Sommerville (2011).

Quadro 1 – Os princípios dos métodos ágeis

Princípios	Descrição
Envolvimento do cliente	Os clientes devem estar envolvidos no processo de desenvolvimento, fornecendo e priorizando novos requisitos do sistema e avaliando suas iterações.
Entrega incremental	O *software* é desenvolvido em incrementos com o cliente, especificando os requisitos a serem incluídos.
Pessoas, não processos	Reconhecer e explorar as habilidades do pessoal do desenvolvimento, de forma que eles desenvolvam suas próprias maneiras de trabalho.
Aceitar as mudanças	Deve-se ter conhecimento de que haverá mudança nos requisitos, se fazendo necessário planejar o sistema para que acomode as mudanças.
Manter a simplicidade	Simplicidade no *software* a ser desenvolvido e no processo do mesmo, trabalhando para eliminar a complexidade do sistema.

Fonte: SOMMERVILLE, 2011, p.56.

As metodologias ágeis tem enfoque nas pessoas e não em processos ou algoritmos, com o objetivo de gastar menos tempo com documentação e focar na implementação, cita Soares (2004). Além disso, as metodologias ágeis se adaptam a novos fatores que decorrem no desenvolvimento do projeto.

“Para ser realmente considerada ágil a metodologia deve aceitar a mudança ao invés de tentar prever o futuro. O problema não é a mudança em si, mesmo porque ela ocorrerá de qualquer forma. O problema é como receber, avaliar e responder às mudanças” (SOARES, 2004, online).

Metodologias ágeis podem comportar documentação, que maximizam o investimento de *stakeholders*, deve capturar informações críticas e que não são facilmente deduzíveis, afirma Balle (2011), assim como conter somente informações e objetivos o mais simples possível de fácil identificação; precisa ter público específico, facilitando o trabalho do mesmo de forma precisa e detalhada e indexados de forma eficiente, pensando em público-alvo.

2.2 Integração de Sistemas API Restful

A arquitetura Rest (*Representational State Transfer*), em português, Transferência de Estado Representacional, é uma arquitetura da Web, que consiste em princípios/regras/constraints que, quando seguidas, permitem a criação de um projeto com interfaces bem definidas, (TOTVS, 2020 online). Desta forma, permite, por exemplo, que aplicações se comuniquem.

Integração pode ser definida como sendo a combinação de conceitos que são reunidos para formarem um conceito único, e esses conceitos podem ser entendidos como sendo as API’s (*Application Programming Interface*) e os microsserviços que são combinados entre si para que se tornem um sistema. Porém, os sistemas devem possuir uma web semântica que é entendida como o compartilhamento de dados para que haja a integração de recursos (FEITOSA, 2005).

Em algumas empresas é mais difícil possuir todos os setores trabalhando juntos e, com o auxílio da integração de todos os módulos, se torna muito mais efetivo ter dados com consistência e qualidade. A integração faz com que as transações de dados entre os diferentes setores sejam mais seguras e rápidas (RICCIO, 2001).

A integração faz com que os sistemas de informações trabalhem de forma automatizada, auxiliando os sistemas na otimização da interação de informações vinda de diferentes sistemas. A interoperabilidade é algo que os sistemas devem possuir para que a integração seja mais eficiente, o que melhora a qualidade e eficiência dos processos (CUNHA, 2005).

2.2.1 Melhores Práticas para API's Restful

Segundo a IBM (2020, online), *API* é um conjunto de padrões que permitem a comunicação entre *softwares* de uma maneira menos verbosa, e as API's Rest e RestFul utilizam o padrão e protocolo http para criação de *softwares* web na qual é possível o desenvolvimento da sigla *CRUD* que deriva de palavras inglesas, em que C significa *Create* e traduzida para o portugues é Criar; R significa *Read* que traduzido para o português é Ler; U significa *Update* que traduzido para o português é Atualizar e por último o D significa *Delete* que traduzindo para o português é Deletar.

As operações do *CRUD* ajudam no desenvolvimento de *software* onde será feita a comunicação entre plataformas FrontEnd e BackEnd, em que o *CRUD* é criado pelo BackEnd, de acordo com a IBM (2021, online). Cada letra da sigla segue um padrão nas APIs RestFul, conforme apresentado abaixo:

- Letra C é o método *POST*, um recurso utilizado para realizar a criação de novas entidades ou informações;
- Letra R é o método *GET*, responsável pela busca de uma informação já criada e salva em um banco de dados, onde é possível realizar consultar por todas as informações ou caso queira buscar por um dado específico é necessário realizar a busca passando um parâmetro onde geralmente o parâmetro informando é o ID do dado, e este pode ser passado através da *Query* ou *Header*;
- Letra U é possível realizar dois métodos que executam a mesma função porém pode mostrar resultados diferentes, são eles o *PUT* e *PATCH*. Com o *PUT* é possível realizar atualizações em um dado completo, como por exemplo, na informação de cadastro de um usuário no banco de dados consta o nome e sobrenome e queira atualizar ambas as informações, neste caso utiliza-se o método *PUT*. Já no método Patch, também executa-se atualizações, porém ele realiza de forma parcial, conforme o exemplo discorrido acima, será alterado somente o sobrenome, neste caso utiliza-se o *PATCH*. No entanto, para ambos os métodos deve-se passar um parâmetro, que no exemplo que utilizou-se foi o ID do dado que desejou-se realizar a alteração;
- Letra D é o método *DELETE* que exclui um dado através de um parâmetro passado, chamado ID, pelo qual através dele é realizado a identificação do dado a ser apagado.

"O código de retorno varia, já que a primeira operação foi bem-sucedida (200 ou 204) e as chamadas subsequentes não localizaram o recurso (404 ou 410)" (IBM, 2021, online).

2.3 NodeJS

O *NodeJS* é um ambiente de execução Javascript, ou seja, é uma plataforma em que é possível criar aplicações Javascript sem a necessidade de um browser para a execução, informa Souza (2020). Tem como objetivo a criação de aplicativos de rede altamente rápidos, que sejam escaláveis e com grande volume de

conexões simultâneas, no entanto não é indicada para operações que demandem muitos recursos computacionais, como memória RAM. O seu diferencial é sua execução *single-thread*, na qual em uma única *thread* executa o código Javascript.

Dentre suas diversas vantagens, está o custo baixo e flexibilidade, permitindo aplicações multidirecionais com comunicação e troca de dados em tempo real, de acordo com Souza (2020), possibilitando assim o seu uso em empresas gigantes do mercado da tecnologia, como LinkedIn e Netflix.

2.4 React

O *React* é uma biblioteca *Javascript* de código aberto utilizado nas aplicações *front-end*. Foi lançado em 2013, desenvolvido pelo Facebook com o objetivo de criar elementos de interface reutilizáveis de forma que seja simples, com ótima performance e intuitiva, segundo Andrade (2020). Multiplataforma sobre a licença MIT e é utilizado por grandes empresas como *Facebook*, *Instagram*, *Uber*, *Spotify*, se tornando uma das principais tecnologias para desenvolvimento web no mundo. Para utilizá-lo é necessário que possua conhecimento na linguagem *Javascript*, pois a mesma é utilizada em sua base, além de conhecimentos em HTML e CSS para a construção de interfaces.

3 Contexto Empreendedor da Aplicação

Empreendedor é aquele que assume riscos e está sempre inovando, independente de possuir uma empresa própria, de acordo com Fabrete (2019, p.16) e o empreendedorismo não precisa ser aplicado somente em um novo negócio, mas sim em um já existente.

3.1 Startup

Startup é uma empresa jovem que possui um modelo de negócios repetível e escalável, com um cenário de incertezas e soluções que precisam ser desenvolvidas. Além disso, para ser uma *startup* necessita de inovação, ou seja, fugir do tradicional, de acordo com Bicudo (2021). Elas são mais frequentes na internet devido ao custo ser mais barato e de fácil propagação quando comparado com uma empresa de agronegócio, por exemplo.

3.2 Canvas

Para se empreender é necessário utilizar um modelo de negócio, geralmente utiliza-se o Canvas. Para o Sebrae (s.d., online), *Business Model* Canvas é uma ferramenta de planejamento estratégico para separar uma ideia em várias partes a fim de tornar visível um modelo de negócios viável. Ele é composto por nove blocos, cada um deles referente a um pilar do negócio e permite uma fácil compreensão e flexibilidade do modelo e ainda analisar o que pode ou não dar certo no negócio. Na Figura 1 está ilustrado o modelo Canvas desenvolvido para este projeto, seguido da descrição de cada componente contextualizado ao aplicativo Student Housing: aplicação para aluguel de quartos e imóveis para estudantes.

Figura 1 - Modelo *Canvas* do projeto

Fonte: Os autores.

1) Parcerias Principais: as escolas e faculdades para divulgarem o aplicativo aos alunos, e as imobiliárias, estudantes e proprietários de imóveis para divulgar e consultar os quartos e imóveis.

2) Atividades Principais: exibir os imóveis oferecidos pelos proprietários/estudantes e solicitar o interesse pelo aluguel de quarto/imóvel realizado pelos estudantes que estiverem cadastrados no aplicativo.

3) Proposta de valor: completar a renda ou depender financeiramente do aluguel, os estudantes terem novas vivências locais em outras cidades e auxiliar os estudantes a se locomoverem para novas cidades, a fim de que eles possam se sentir mais seguros.

4) Relacionamento com o Clientes: o aplicativo terá uma equipe para prestar assistência técnica via conexão remota, por e-mail e via whatsapp.

5) Segmentos de Clientes: estudantes que possuam idade superior a 18 anos e os pais e responsáveis de alunos menores de idade que se mudaram de cidade para estudar, assim como proprietários/estudantes que possuam quartos vagos para serem alugados e dividirem as despesas.

6) Principais Recursos: A infraestrutura de TI para hospedagem do aplicativo e internet será necessária para que os usuários se conectem no aplicativo.

7) Canais: serão utilizadas as redes sociais: Instagram e Facebook para a divulgação de anúncios, em redes sociais de escolas que fazem preparação para vestibulares e nas faculdades por meio de anúncios em portais e sites.

8) Estrutura de Custos: Os custos do projeto foram divididos em três pilares sendo eles: custos com os programadores, pois irá demandar horas trabalhadas para a construção do aplicativo; hospedagens para a aplicação web, devido ao custo com servidores em nuvens ou locais; e para a aplicação mobile custos com pagamentos das plataformas onde o aplicativo será postado, sendo *Google Play* para a versão android e a plataforma da APP Store para a versão IOS.

9) Fontes de Renda: Como retorno dos custos, será cobrada uma porcentagem do proprietário referente a cada anúncio publicado e pelos pequenos anúncios por meio do google adsense.

4 Análise do Projeto

Durante o processo de desenvolvimento do *software* é preciso realizar a análise do projeto, a fim de certificar os requisitos do sistema. Para Bezerra (2015), analisar significa “quebrar” um sistema em seus componentes e o estudo de como eles interagem entre si para que seja entendido como o sistema funciona. É realizado um estudo de forma detalhada sobre os requisitos para, a partir dele, construir os modelos para representar o sistema que será construído, a fim de obter a melhor solução para o problema. Há diversos métodos de modelagem para análise, e como o sistema será orientado a objetos, utilizou-se os diagramas da UML: diagrama de casos de uso, diagrama de classes, diagrama de estados, diagrama de atividades, dentre outros.

4.1 Engenharia de *Software*

A Engenharia de *Software* tem como objetivo focar em todos os aspectos da produção de *software*, a partir da especificação do sistema até sua manutenção, segundo Sommerville (2011), e tem a ver com a obtenção de resultados de qualidade determinado no cronograma e orçamento. Nela é utilizada uma sequência de atividades que leva à produção de um produto de *software*.

4.1.1 Elicitação de Requisitos

O levantamento de requisitos precisa envolver diferentes pessoas dentro das organizações e eles são classificados em: funcionais, não-funcionais e de domínio, segundo Sommerville (2011). No desenvolvimento do sistema Student Housing foram utilizados os requisitos funcionais e não-funcionais. Na 2 figura está ilustrado o requisito funcional Cadastrar quarto/imóvel.

O documento completo dos requisitos estão disponíveis no GITHUB (MACEDO E SILVESTRE, 2021).

Figura 1 - Requisito funcional Cadastrar quarto/imóvel

RF002 → Cadastrar quarto/imóvel + Add a view | Search | ... | New

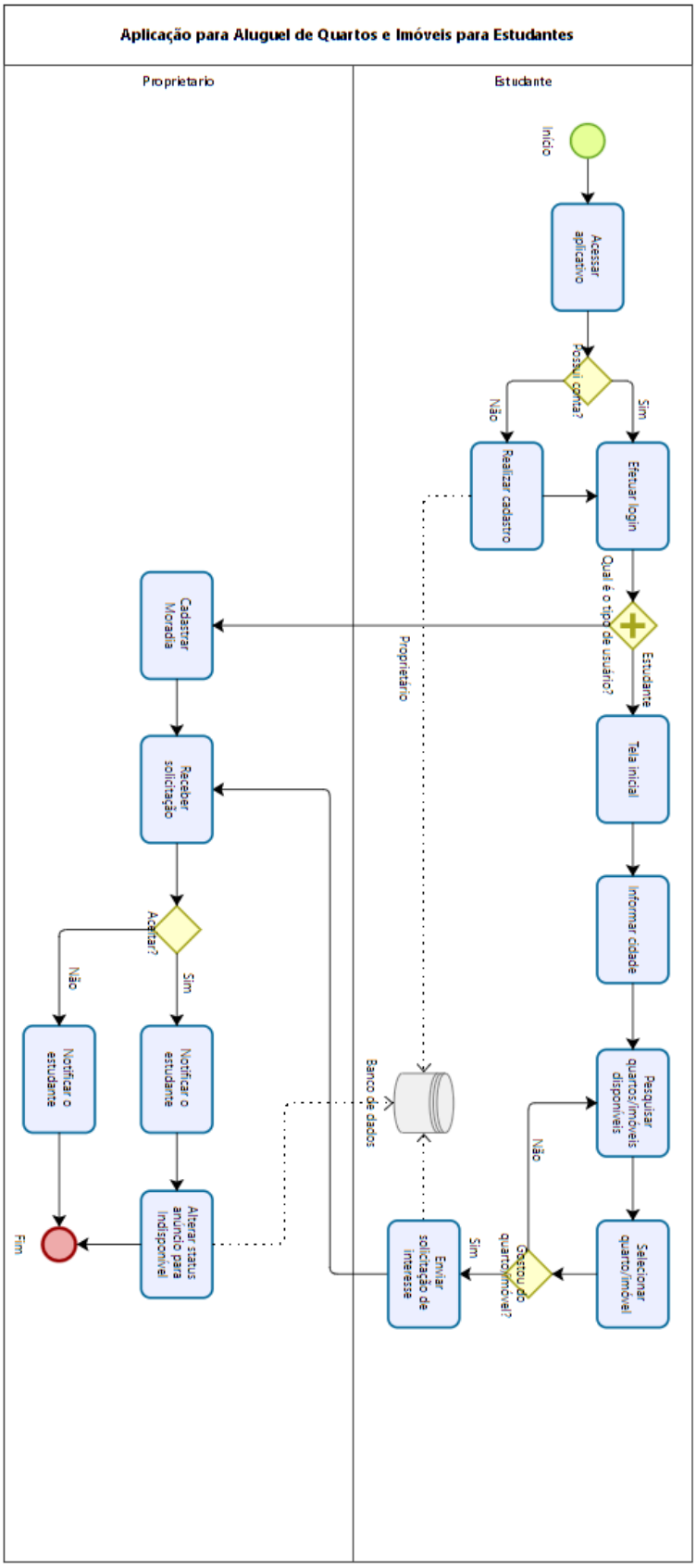
Aa Name	Tags	Column
Identificador:	RF002	
Nome:	Cadastro de quarto / imóvel	
Módulo:	Tela Cadastro (Pagina Interna aplicação)	
Data de criação:	19/05/2021	
Data da última alteração:	16/10/2021	
Versão:	1.0	
Prioridade:	Alta	
Descrição:	O sistema deverá cadastrar novos locais de quartos e imóveis vagos, através das categorias: quarto, apartamento e casa. Além disso deverá ser informado se será individual ou compartilhado.	
Informações: OPEN	Valor, tipo de imóvel: quarto, apartamento ou casa e se é individual ou compartilhado, localidade: cep, logradouro, número, bairro, cidade/uf e complemento, comodidades e imagens	
Regra de Negocio:	O sistema deverá permitir realizar o cadastro se todos os campos estiverem preenchidos de forma correta. Devem ser inseridas no mínimo cinco imagens e o endereço. O usuário deverá estar logado no sistema com o perfil de proprietário.	

Fonte: Os autores.

4.1.2 BPMN

A BPMN (*Business Process Modeling Notation*) é uma notação padrão que representa processos de negócios por meio de diagramas de processos de negócio, ou seja, descreve a lógica dos passos de um processo. É orientada para uso humano, visto que possui fácil compreensão do diagrama, mesmo em processos complexos, de acordo com Arantes (2014). O uso da modelagem é importante na automatização de processo, pois a partir dela que os processos são desenhados, além disso, é possível encontrar falhas de processo e realizar ajustes a fim de manter sua otimização. Na Figura 3 está ilustrado o BPMN do projeto Student Housing.

Figura 3 - BPMN

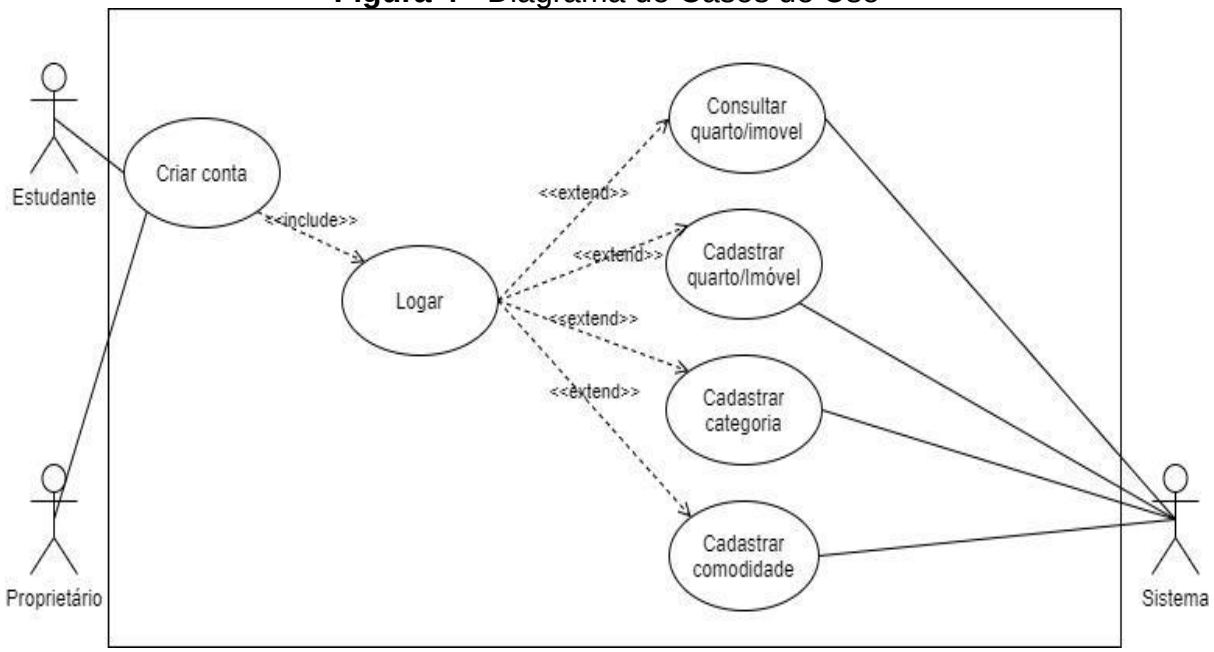


Fonte: Os autores.

4.1.3 Diagrama de Casos de Uso

Os casos de uso representam as funcionalidades do sistema identificando os atores envolvidos, de acordo com Sommerville (2011). É utilizado para apoiar a elicitação de requisitos. Na Figura 4 está ilustrado o Diagrama de Casos de Uso do projeto *Student Housing*.

Figura 4 - Diagrama de Casos de Uso



Fonte: Os autores.

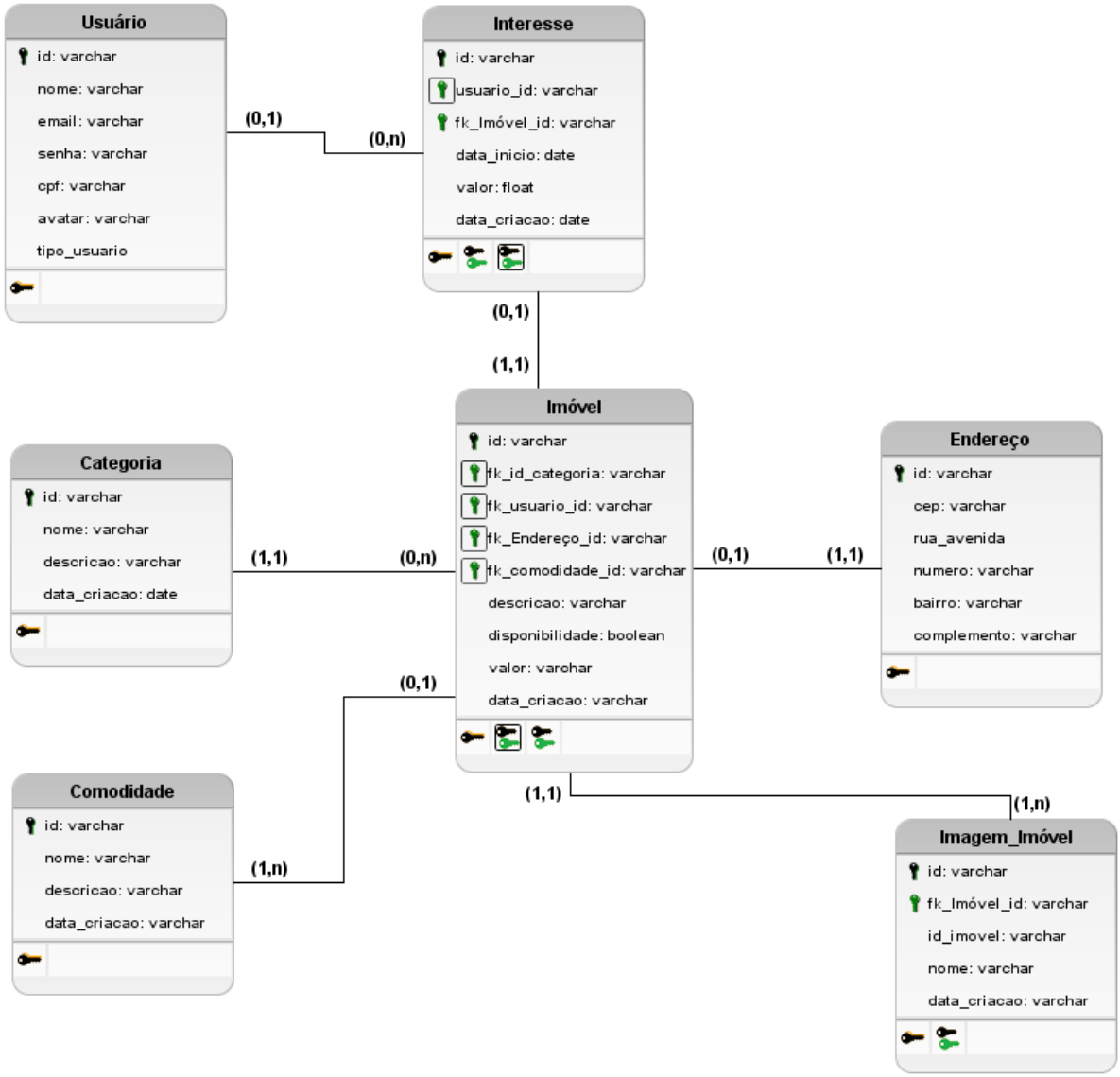
4.1.4 Modelagem de dados

A modelagem de dados na construção e manutenção de uma aplicação é imprescindível, pois com ela é possível estruturar os dados inseridos que posteriormente serão convertidos em informação, além de identificar os tipos, entidades e atributos, afirma Dias Neto (2011).

Existem três tipos de modelagem: conceitual, lógico e físico. De acordo com Miranda (2017), o modelo conceitual é responsável por determinar as entidades, que são representadas por um retângulo com seu nome escrito no centro, além de se construir o relacionamento entre as entidades que podem ser: um para muitos, muitos para muitos ou um para um. Já o modelo lógico é implementado os recursos com padrão de nomenclatura, no qual são definidas as chaves primárias e estrangeiras. E o modelo físico considera as limitações impostas pelo SGBD escolhido e, deve-se seguir os exemplos do modelo conceitual e lógico.

Na Figura 5 está ilustrado o modelo de banco de dados do projeto *Student Housing,* mostrando as entidades, os relacionamentos e tipos de dados em que foram criadas as tabelas no banco de dados escolhido, o PostGres.

Figura 5 - Modelagem de banco de dados do projeto



Fonte: Os autores.

4.1.5 Demais Artefatos de *Software*

Outros artefatos de *software*, como 5W1H, EAP e o levantamento das respostas do questionário que foi aplicado também foram desenvolvidos e estão disponíveis no GITHUB (MACEDO E SILVESTRE, 2021).

5 Desenvolvimento do Aplicativo

5.1 Telas da Aplicação

Nesta seção serão apresentadas as telas criadas no figma referente a aplicação web, em conjunto com as especificações de suas funcionalidades. A versão criada chamada *Student Housing* será destinada aos perfis de estudante e proprietário do quarto ou imóvel, e no momento do cadastro, o usuário deverá selecionar qual o seu perfil para que posteriormente seja direcionado para as telas específicas de forma que a aplicação seja segura e eficaz.

5.1.1 Landpage

O primeiro contato com a aplicação acontece por meio da tela de *Landing Page,* ilustrada na Figura 6. Nela existe uma breve apresentação da aplicação e o botão de começar agora, que direciona o usuário para a tela de login.

Figura 6 - *Landing Page*

Fonte: Os autores.

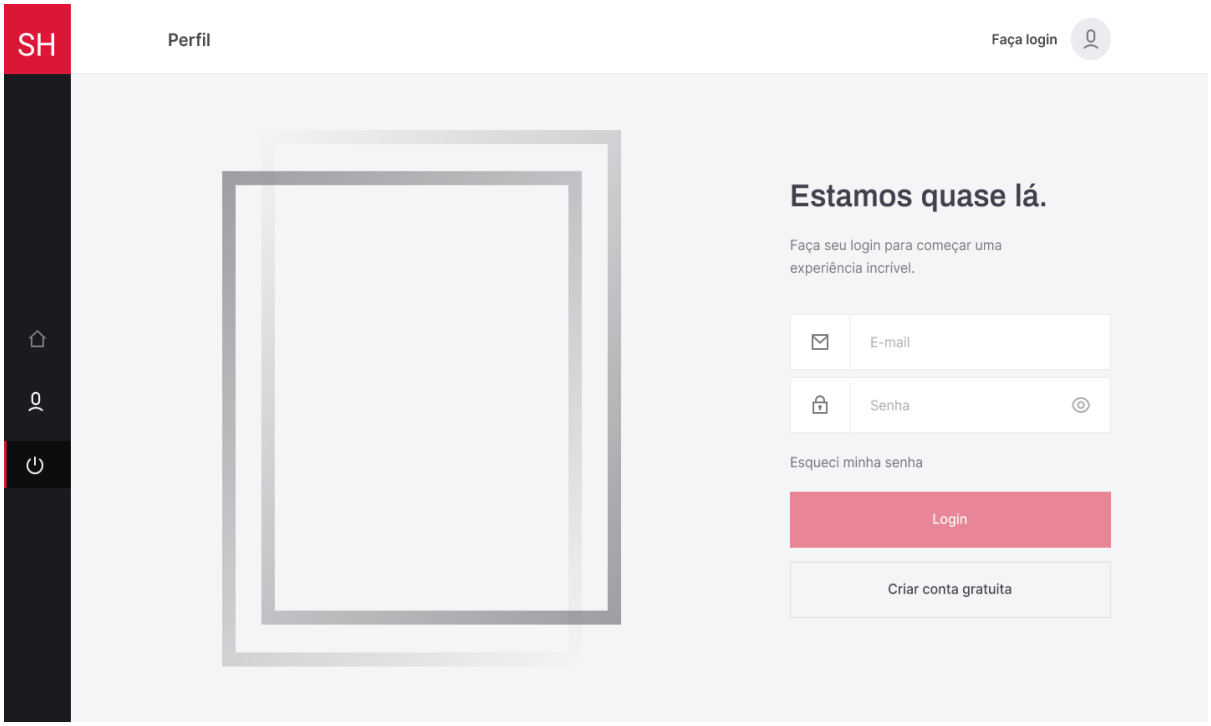
5.1.2 Login

A tela de login possui grande importância na aplicação, pois garante a segurança e privacidade ao usuário. Para ter acesso às funcionalidades, primeiramente é necessário que o estudante ou proprietário tenha um usuário previamente cadastrado na base de dados da aplicação. Na Figura 7 está ilustrada a tela de *login,* que são iguais para os dois tipos de usuários, solicitando informações básicas, como e-mail e senha cadastrados na aplicação.

5.1.3 Cadastro

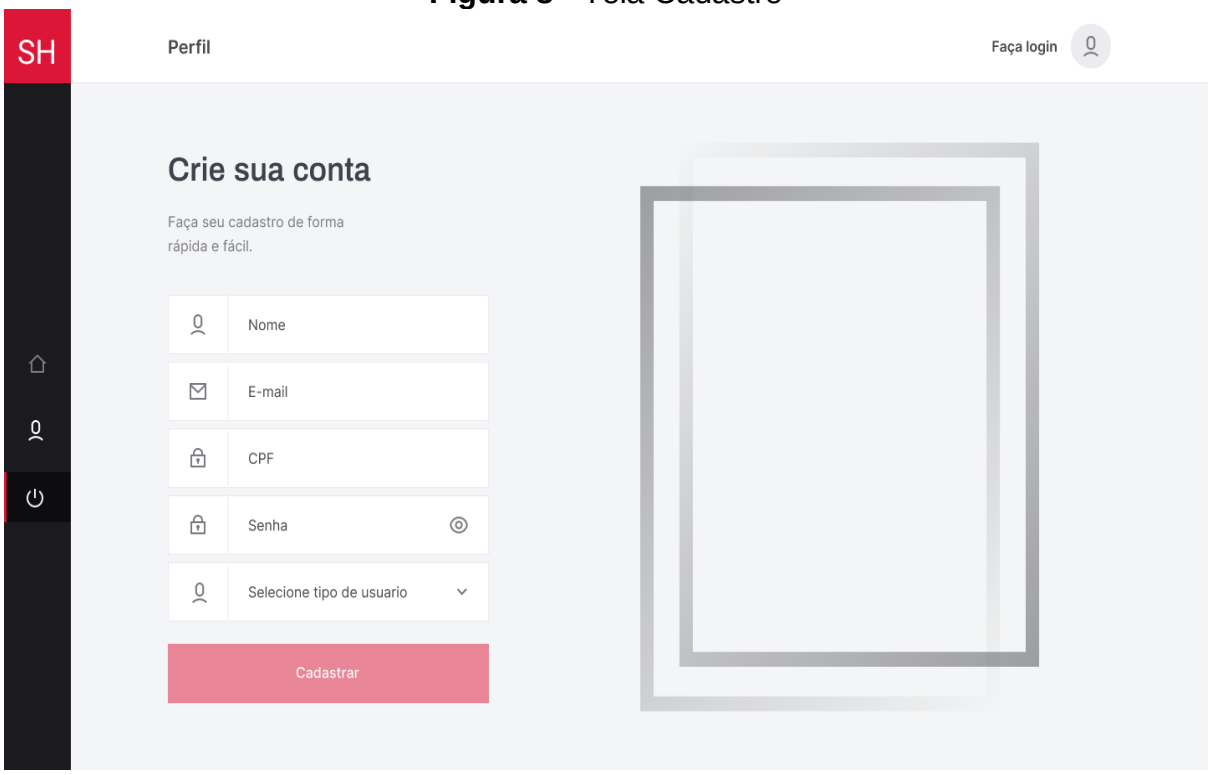
Na tela de cadastro é necessário informar nome completo, email, cpf, senha de acesso e selecionar qual é o tipo de usuário: estudante ou proprietário, conforme a Figura 8, para que o usuário tenha acesso às funcionalidades da aplicação.

Figura 7 - Tela de *Login*



Fonte: Os autores.

Figura 8 - Tela Cadastro



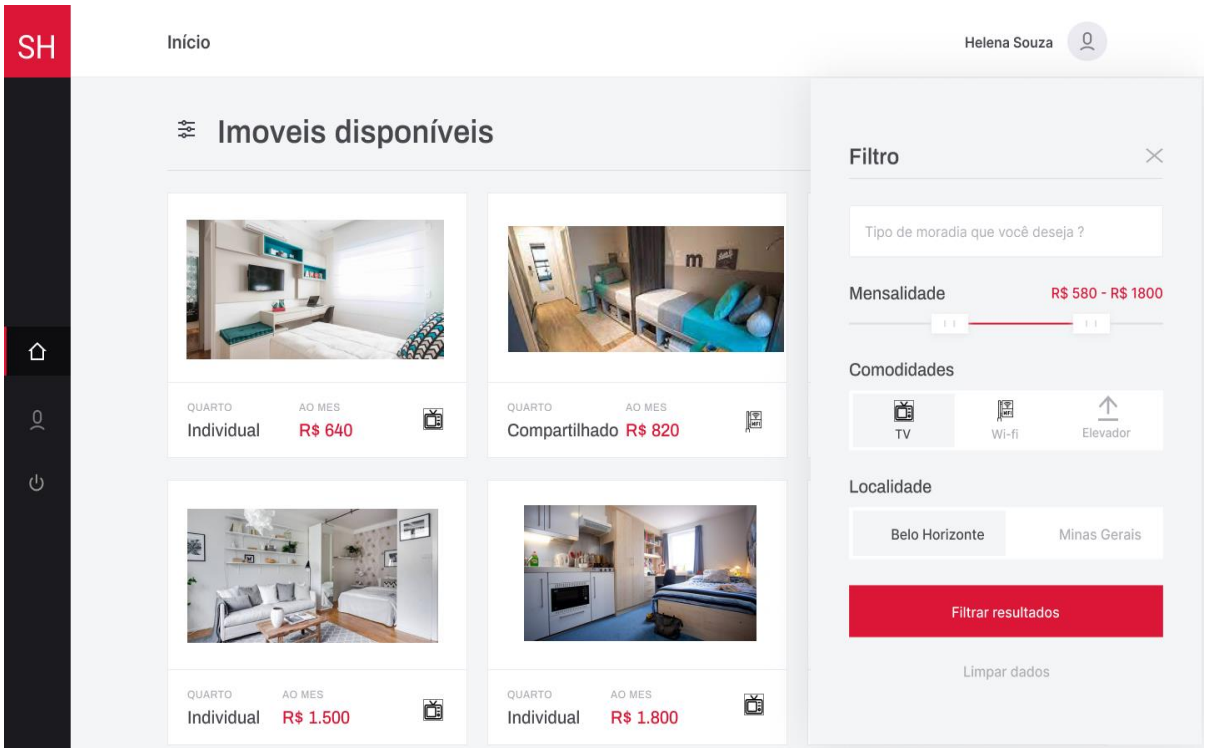
Fonte: Os autores.

5.1.4 Listagem de quartos (estudante)

O usuário que deseja alugar um quarto terá acesso aos quartos disponíveis contendo informações básicas, como tipo do quarto (individual ou compartilhado) e valor do aluguel (Figura 9).

Para melhor experiência do usuário e otimização na busca dos quartos, a opção de filtragem (Figura 9) é essencial, pois permite consultar os quartos pelo tipo de imóveis, mensalidade, localidade, além da opção de comodidades.

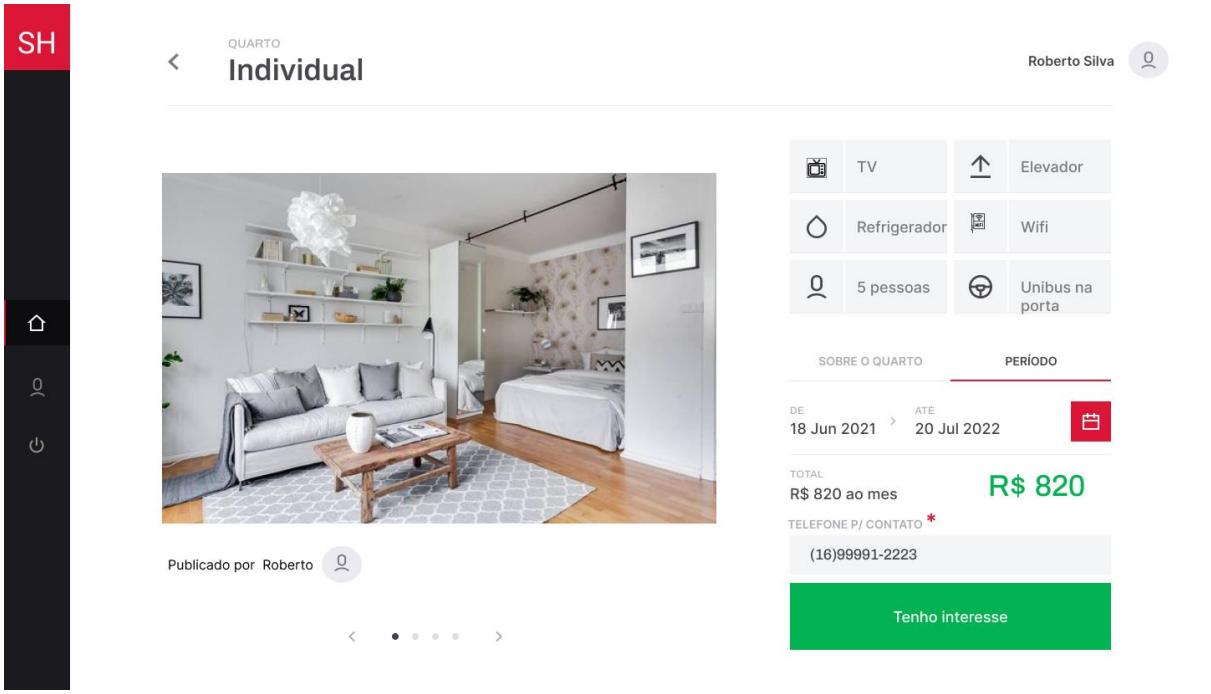
Figura 9 - Tela de consulta de quartos e imóveis



Fonte: Os autores.

Ao escolher o quarto ou imóvel, o usuário será direcionado para a tela de detalhamento (Figura 10), que contém todas as informações necessárias referentes ao imóvel, como as imagens, a descrição, tipo de imóvel, comodidades do imóvel, o período estimado do aluguel. O usuário deve informar um número de telefone para contato para que seja enviado a solicitação de interesse.

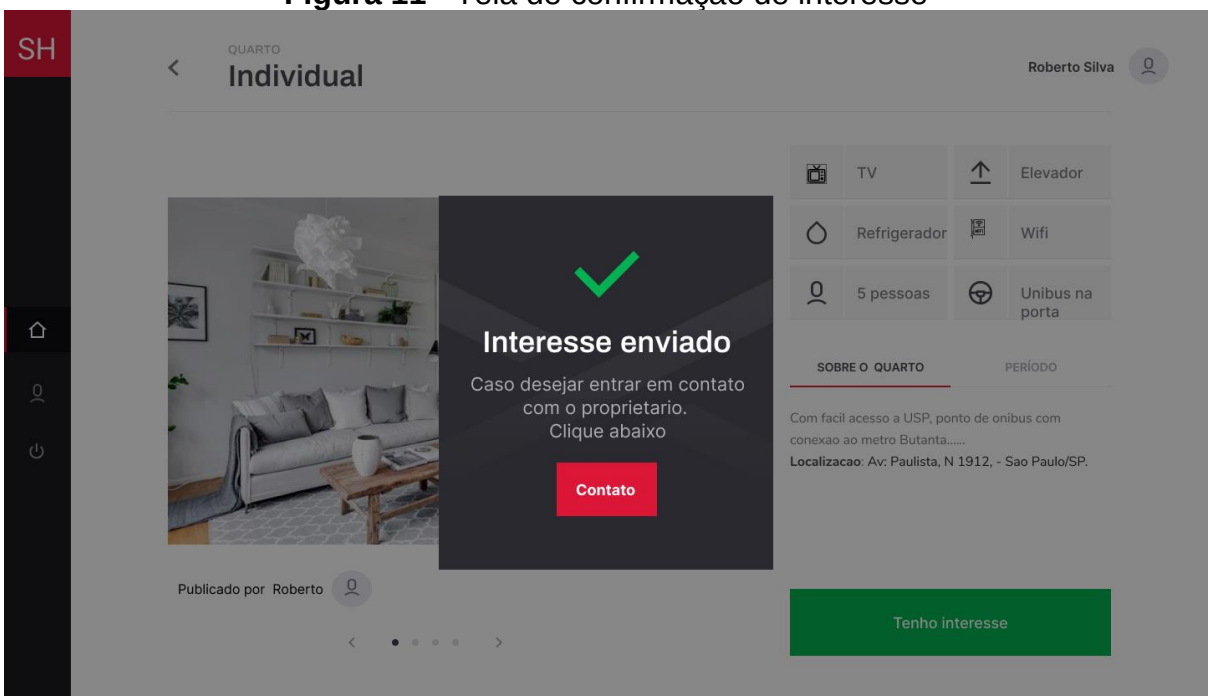
Figura 10 - Tela de quarto detalhada



Fonte: Os autores.

Ao clicar no botão Tenho interesse, uma mensagem de confirmação de interesse pelo quarto ou imóvel será enviada ao proprietário e uma mensagem aparecerá ao usuário, conforme apresentado (Figura 11).

Figura 11 - Tela de confirmação de interesse



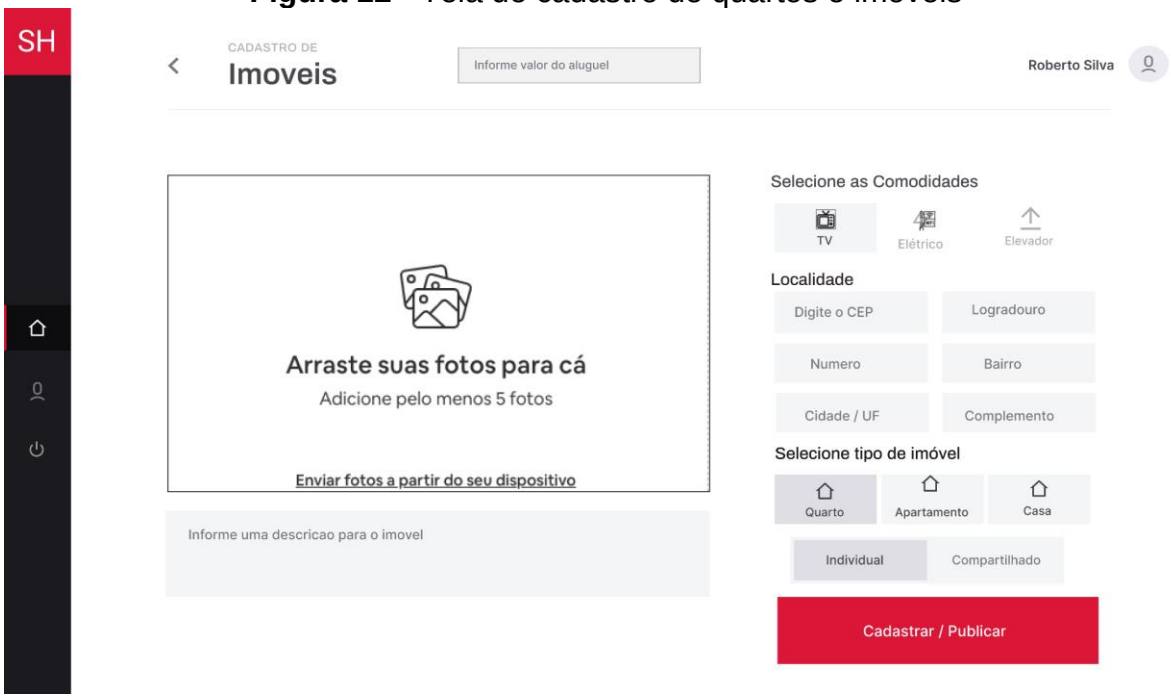
Fonte: Os autores.

5.1.5 Publicações (Proprietário)

Para o usuário que deseja cadastrar e publicar um quarto ou imóvel de sua responsabilidade para alugar, é imprescindível que o mesmo tenha se cadastrado anteriormente com o *perfil* de proprietário.

Na tela de cadastro de quartos e imóveis é necessário informar o valor, imagens, a localidade, inserir fotos do local, a descrição, selecionar as comodidades e o tipo de moradia, conforme a Figura 12.

Figura 12 - Tela de cadastro de quartos e imóveis



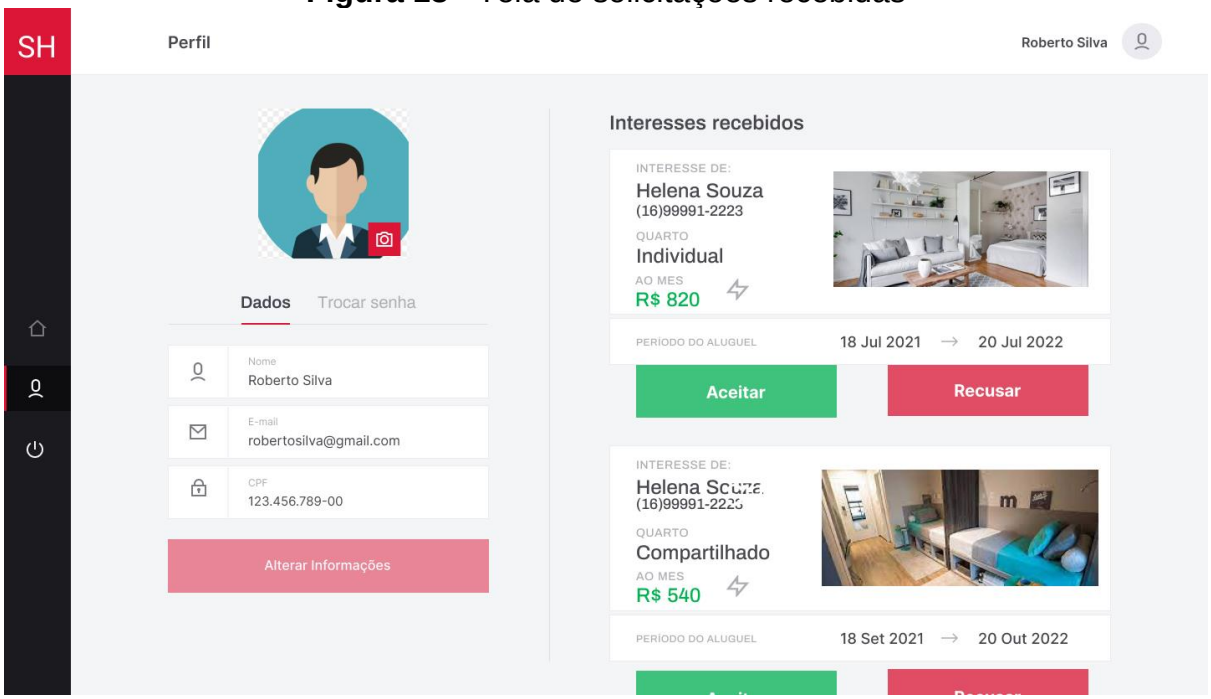
Fonte: Os autores.

5.1.6 Solicitações recebidas (proprietário)

Para o usuário consultar as solicitações recebidas, o mesmo deve estar logado com o *perfil* de proprietário.

Na tela de solicitações recebidas, são apresentadas as informações necessárias do estudante interessado, como o nome e o número de contato. O proprietário deverá aceitar ou recusar a solicitação de interesse (Figura 13).

Figura 13 - Tela de solicitações recebidas



Fonte: Os autores.

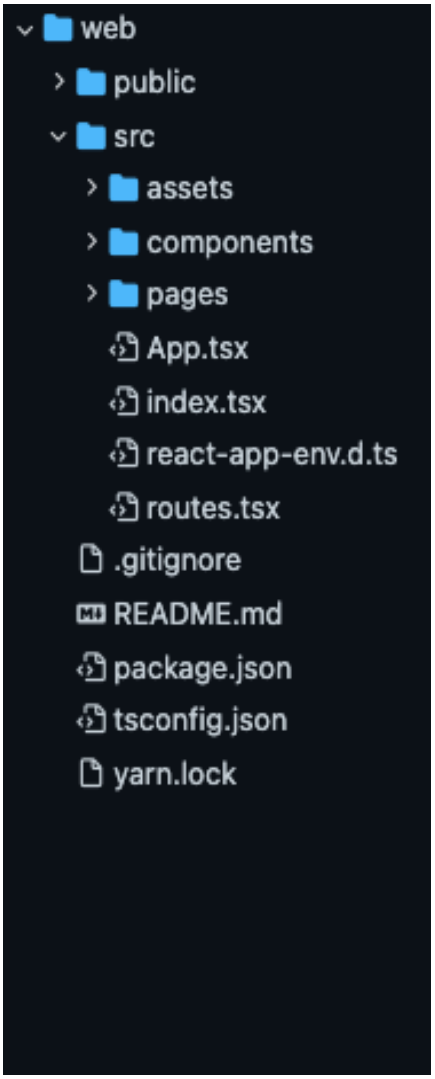
5.2 Implementação do *Front-end*

No desenvolvimento do *Front-end*, utilizou-se a linguagem *javascript* e *ReactJS* com *Typescript* para todo o desenvolvimento das páginas e componentes, no qual o *Typescript* auxiliou em como saber quais os tipos de dados são aceitos ao passar um parâmetro em uma função. Na construção da parte visual do CSS utilizou-se a *Lib Styled-Components*, pois com ela criou-se o CSS a partir do *javascript*.

5.2.1 Estrutura e organização das pastas do Front-end

Na Figura 14 está mostrado como foi dividido e estruturado o *front-end*. Como já mencionado, dividiu-se a aplicação em componentes e páginas, e os componentes foram responsáveis por pequenos blocos da aplicação utilizados em mais de uma página, a fim de facilitar na manutenção da aplicação e melhor entendimento, como por exemplo, caso futuramente seja necessário alterar parte do código que é utilizado em mais de uma página, a alteração será realizada apenas em um componente e passado para os demais pela chamada do mesmo. As páginas foram responsáveis somente pela construção da tela em si, na qual foram inseridos os componentes e outros elementos, como imagens e textos.

Figura 14 - Estrutura e organização das pastas



Fonte: Os autores.

5.2.2 Component Input

Na Figura 15 é apresentado um exemplo da construção de um componente, o *input*, com campos para criação de um novo usuário. Na linha 3 tem-se a importação do CSS. Na linha 5 utilizou-se o *Typescript* para a criação de interface, sendo passados todos os tipos de dados aceitáveis em cada campo do *input*. Em seguida, realizou-se a construção do componente como função, em que utilizou-se para passagem de parâmetros recebidos, como nome, e-mail, etc.

Figura 15 - Component Input

```
1  import React, {InputHTMLAttributes} from 'react';
2
3  import './styles.css'
4
5  interface InputProps extends InputHTMLAttributes <HTMLInputElement>{
6      name: string;
7      label: string;
8      type: string
9  }
10
11 const Input: React.FC<InputProps> = ({ label, name, type, ...rest }) => {
12     return(
13         <div className="input-block">
14             <label htmlFor={name}>{label}</label>
15             <input type={type} id={name} {...rest}/>
16         </div>
17     );
18 }
19
20 export default Input;
```

Fonte: Os autores.

5.2.3 Página de Login

Na Figura 16 estão mostrados os componentes na página de login. Na linha 1 realiza-se a importação do componente *input* e o mesmo é utilizado nas linhas 10 e 15, e cada linha é responsável por uma informação que o usuário inserir; posteriormente esta será interpretada e passada para o *back-end* realizar a leitura e, por fim, será feita a operação desejada ao usuário: acessar e logar na aplicação. Desta forma, dividem-se as responsabilidades de cada parte, além disso, utilizou-se o conceito do SOLID em toda a aplicação.

5.3 Implementação do Back-end

Para o desenvolvimento do back-end, utilizou-se a linguagem *javascript* com *NodeJS* para a criação da API *RestFul*, e também *Typescript*, pois ele auxiliou na construção de tipagens de forma mais simples e de fácil entendimento, para que, futuramente, seja possível visualizar qual o tipo de dado é recebido por uma variável. Já para a construção e conexão com o banco de dados utilizou-se o *TypeORM* com Postgres. Com ele foi possível escrever os códigos para a criação de uma nova tabela no banco usando a linguagem *javascript*, e assim podemos criar *migrations*. Além disso, o *back-end* é responsável pela criação da regra de negócio de toda a aplicação.

Figura 16 - Página Login

```
import Input from '../../components/Input';
import './styles.css'
import { Link } from 'react-router-dom';
function Landing(){
    return(
        <div id="page-landing" className="container">
            <div className="form-logon">
                <form action="">
                    <h3>Faça seu login</h3>
                    <Input
                        name = "id"
                        label = "sua ID"
                        type="text"
                    />
                    <Input
                        name = "password"
                        label = "sau senha"
                        type="password"
                    />
                    <div className="button-container">
                        <a href  ="/cars "className="button">
                            Entrar
                        </a>
                    </div>
                </form>
                <div className="links">
                    <Link to="/register">Criar uma conta</Link>
                </div>
            </div>
        </div>
    );
}
export default Landing;
```

Fonte: Os autores.

5.3.1 Criando a conexão com Banco de dados

Na Figura 17, está demonstrado o código que foi desenvolvido para realizar a conexão com o banco de dados.

Figura 17 - Criando a conexão com Banco de dados

```
import typeorm from 'TypeORM';
import path from 'path';

const db = typeORM({
    client:'postgresql',
    connection:{
        filename: path.resolve(__dirname, 'db.postgresql'),
    },
    useNullAsDefault: true,
})

export default db;
```

Fonte: Os autores.

5.3.1 Código Migration para criar a tabela Usuário

Na Figura 18 temos um exemplo da criação da migration para a tabela de usuário, na qual, de forma muito simples, criaram-se duas funções: a *UP* e a *Down*.

A função *UP* é responsável pela construção da tabela e a função a *Down* é responsável por apagar a tabela ou a ação feita anteriormente. Para criar a *migration*, as informações serem geradas no banco de dados e visualizadas em um programa, executando-se o seguinte comando através do *Prompt* de comando *typeorm* migration:create -n Nome da *Migration*.

Figura 18 - Código *migration* para criar a tabela Usuário

```
import {MigrationInterface, QueryRunner} from "typeorm";

export async function up(queryRunner: QueryRunner ){

 return typeorm.schema.createTable('users', table =>{
    table.uuid('id').primary();
    table.string('name').notNullable();
    table.string('email').notNullable();
    table.string('password').notNullable();
    table.string('cpf').notNullable();
    table.string('avatar').notNullable();
    table.string('avatar').notNullable();
    table.boolean('isOwner').notNullable();
    table.date('created_at').notNullable();
 });
}

export async function down(QueryRunner: queryrunner ){
 return queryrunner.schema.dropTable('users');
}
```

Fonte: Os autores.

6 Conclusão

Este trabalho apresentou o desenvolvimento de sistema web denominado *Student Housing* para suprir dificuldades detectadas em pesquisa exploratória informal realizada previamente. Apresentou-se a análise, a documentação e o desenvolvimento do protótipo para auxiliar estudantes que necessitam de uma aplicação para procurar quartos e imóveis para alugar, assim como, estudantes ou proprietários que desejam compartilhar e alugar quartos ou imóveis disponíveis e dividir despesas. Entendeu-se que o objetivo principal foi alcançado.

No decorrer de todo o desenvolvimento da aplicação, foi possível aplicar na prática conceitos, como as técnicas e metodologias utilizadas em Engenharia de Software. Foram elaboradas o *BPMN*, *EAP*, *5W1H*, diagrama de Caso de Uso, documentos de requisitos, além da prototipação de telas e o desenvolvimento da aplicação, que foi realizado por meio da plataforma *NodeJS* e o *framework React*.

Com o desenvolvimento do MVP (*Minimum Viable Product*), possibilitou-se colocar em prática as principais funcionalidades da aplicação, como o cadastro de usuários, cadastro de quartos e imóveis, consultas de quartos e imóveis, demonstração de interesse por um imóvel e a confirmação ou recusa da solicitação de interesse.

Uma versão *mobile* do aplicativo está em andamento e a prototipação pode ser visualizada no *github*.

Visando o aperfeiçoamento do projeto, foram levantadas algumas possíveis melhorias a fim de contribuir ainda mais com a experiência do usuário futuramente, como a implementação de um *chat* para aprimorar a comunicação entre as partes, e o compartilhamento de localização por GPS ao cadastrar e consultar quartos e imóveis. Ainda, pode-se permitir o envio de documentos contratuais a fim de formalizar o envio do contrato de aluguel.

Referências

ANDRADE, Ana Paula de. Conheça o React, biblioteca para desenvolvimento Web. Treinaweb. 2020. Disponível em: <https://www.treinaweb.com.br/blog/conheca-o-react-biblioteca-para-desenvolvimento-web>. Acesso em: 04 out. 2021.

ARANTES. Raíssa Nogueira. Introdução ao Business Process Modeling Notation (BPMN). Devmedia. 2014. Disponível em: <https://www.devmedia.com.br/introducao-ao-business-process-modeling-notation-bpmn/29892>. Acesso em: 04 out. 2021.

BALLE, Andrea Raymundo. Análise de metodologias ágeis: conceitos, aplicações e relatos sobre XP e Scrum. 2011. Trabalho de graduação. Universidade Federal do Rio Grande do Sul. Disponível em: <https://www.lume.ufrgs.br/handle/10183/31028>. Acesso em: 21 mar. 2021

BEZERRA, Eduardo. Princípios de Análise e Projeto de Sistemas com UML. 3. ed. Rio de Janeiro. Elsevier, 2015.

BICUDO, Lucas. O que é uma startup?. StartSe. 2021. Disponível em: <https://app.startse.com/artigos/o-que-e-uma-startup>. Acesso em: 04 out. 2021.

CUNHA, Francisco José Aragão Pedroza. A gestão da informação nos hospitais: importância do prontuário eletrônico na integração de sistemas de informação em saúde, 2005. Dissertação (mestrado). Universidade Federal da Bahia. Disponível em: < https://repositorio.ufba.br/ri/bitstream/ri/8174/1/Disserta%c3%a7%c3%a3o_Francisco %20Pedroza.pdf>. Acesso em: 21 mar. 2021.

DIAS NETO, Arilo Claudio. Modelagem de Dados Tutorial. Devmedia. 2011. Disponível em: <https://www.devmedia.com.br/modelagem-de-dados-tutorial/20398>. Acesso em: 18 out. 2021.

FABRETE, Teresa Cristina Lopes. Empreendedorismo. 2. ed. São Paulo: Pearson, 2019.

FEITOSA, Ailton Luiz Gonçalves. A integração entre sistemas legislativos, terminologia e web semântica na organização e representação da informação legislativa. 2005. Tese (Doutorado em Ciência da Informação). Universidade de Brasília. Disponível em: < https://repositorio.unb.br/handle/10482/34606>. Acesso em: 21 mar. 2021.

IBM. Criando microsserviços RESTful. 2020. Disponível em: <https://cloud.ibm.com/docs/cloud-native?topic=cloud-native-rest-api&locale=pt-BR>. Acesso em: 22 mar. 2021.

MACEDO, Larissa; SILVESTRE, Artur. Repositório para a disponibilização de documentos do Trabalho de Conclusão de Curso - Artur e Larissa. Disponível em: <https://github.com/LarissaVianaM/TccHousingStudent>. Acesso em: 21 out. 2021.

MIRANDA, William. Modelagem de Dados: O que é e para que serve para um DBA. PortalGSTI. 2017. Disponível em: <https://www.portalgsti.com.br/2017/02/modelagem-de-dados-o-que-e-e-para-que-serve-para-um-dba.html>. Acesso em: 18 out. 2021.

RICCIO, Edson Luiz. Efeitos da tecnologia de informação na contabilidade: estudo de casos de implementação de sistemas empresariais integrados-ERP, 2001. Tese (Livre-Docência em Contabilidade e Atuária). Universidade de São Paulo. Disponível em: <https://teses.usp.br/teses/disponiveis/livredocencia/12/tde-06122005-101802/publico//riccio.pdf>. Acesso em: 21 mar. 2021.

SEBRAE. O que é Business Model Canvas e como aplicá-lo no seu negócio?. s/d. Disponível em: <https://inovacaosebraeminas.com.br/o-que-e-business-model-canvas-e-como-aplica-lo-no-seu-negocio/>. Acesso em: 23 set. 2021.

SOARES, Michel dos Santos. Metodologias Ágeis Extreme Programming e Scrum para o Desenvolvimento de Software. Revista Eletrônica de Sistema de Informação. v.3, n.1, 2004. Disponível em: <http://www.periodicosibepes.org.br/index.php/reinfo/article/view/146>. Acesso em: 21 mar. de 2021.

SOMMERVILLE, Ian. Engenharia de Software. 9. ed. São Paulo: Pearson, 2011.

SOUZA, Ivan de. Saiba o que é Node.js, como ele funciona e como usá-lo no seu site. Rockcontent. 2020. Disponível em: <https://rockcontent.com/br/blog/node-js/>. Acesso em: 30 set. 2021.

TOTVS. Arquitetura REST: Saiba o que é e seus diferenciais. 2020. Disponível em: <https://www.totvs.com/blog/developers/rest/>. Acesso em: 16 out. 2021.