Plataforma de gestão de arrendamento baseada numa arquitetura assíncrona orientada a serviços

Nuno Ricardo Pinto Barreira - a37553

Projeto apresentado à Escola Superior de Tecnologia e de Gestão de Bragança para obtenção do Grau de Mestre em Informática.

Trabalho orientado por:

Prof. Doutor Paulo Alves

Prof. Doutor José Eduardo Fernandes

Bragança

Dezembro de 2023

Plataforma de gestão de arrendamento baseada numa arquitetura assíncrona orientada a serviços

Nuno Ricardo Pinto Barreira - a37553

Projeto apresentado à Escola Superior de Tecnologia e de Gestão de Bragança para obtenção do Grau de Mestre em Informática.

Trabalho orientado por:

Prof. Doutor Paulo Alves

Prof. Doutor José Eduardo Fernandes

Bragança

Dezembro de 2023

Dedicatória

Dedico este trabalho a todas as pessoas que estiveram envolvidas em toda a minha educação, especialmente no decorrer de todo este processo e que me ajudaram de forma indireta na produção deste resultado, mas sobretudo aos meus pais que sempre me apoiaram em todo o decorrer da minha vida e que são as pessoas que me fizeram ser a pessoa que sou hoje.

Agradecimentos

Agradeço de forma especial aos meus pais e avós que de alguma forma estiveram sempre envolvidos e sempre demonstraram preocupação e valor neste processo e me ajudaram a formar na minha educação. Um obrigado especial também aos meus amigos e à minha namorada, Andreia, que sempre demonstrou um grande apoio e foi um pilar para o desenvolvimento deste trabalho e de todo o processo para me tornar mestre. Concluo por agradecer aos professores Paulo Alves e José Eduardo Fernandes por toda a ajuda, empenho e transmissão de conhecimentos no decorrer deste trabalho.

Resumo

O RentaRoom é uma inovadora aplicação web concebida com o objetivo principal de simplificar e aprimorar significativamente o processo de arrendamento de quartos, oferecendo uma solução simples tanto para proprietários quanto para arrendatários. Num mundo cada vez mais digital, a necessidade de centralizar informações e tornar o gerenciamento de arrendamentos mais eficiente é essencial.

Uma das principais características do RentaRoom é a sua interface de utilizador moderna e amigável, construída utilizando o React, uma biblioteca bem conhecida do Javascript. Isto permite que tanto proprietários quanto arrendatários possam navegar e interagir com a plataforma de forma fácil e intuitiva, sem a necessidade de conhecimento técnico avançado. A simplicidade da interface desta aplicação proporciona uma experiência agradável e sem complicações aos seus utilizadores.

Além disso, toda a lógica de negócio por trás do RentaRoom é alimentada pelo FastAPI com Motor. A utilização destas tecnologias garante um desempenho ágil e confiável, essencial para lidar com as operações envolvidas na gestão de arrendamentos. Os proprietários assim podem gerir as suas propriedades, documentos, despesas e contratos de forma eficiente, enquanto os arrendatários têm acesso fácil às informações relevantes e podem comunicar de forma direta com os proprietários. Além de esta aplicação simplificar a gestão de arrendamentos, esta oferece também uma funcionalidade adicional de criação de tickets de reparação. Isto permite que os arrendatários relatem rapidamente problemas com a propriedade, facilitando a manutenção e o reparo oportuno, tornando a experiência de moradia mais satisfatória e o processo mais rápido e acessível para todos os envolvidos.

Em resumo, o RentaRoom é uma aplicação web que visa otimizar e modernizar o

processo de arrendamento de quartos através de uma interface amigável, tecnologia recente e recursos adicionais como a gestão de documentos e tickets de reparação, tornando desta forma o arrendamento de propriedades numa experiência mais eficiente e tranquila para proprietários e arrendatários.

Palavras-chave: Arrendamento Imobiliário, FastApi, React, WebApp

Abstract

RentaRoom is an innovative web application designed with the main objective of significantly simplifying and improving the room rental process, offering a simple solution for both owners and tenants. In an increasingly digital world, the need to centralize information and make rental management more efficient is essential.

One of RentaRoom's main features is its modern and user-friendly user interface, built using React, a well-known Javascript library. This allows both owners and tenants to navigate and interact with the platform easily and intuitively, without the need for advanced technical knowledge. The simplicity of this application's interface provides a pleasant and uncomplicated experience for its users.

In addition, all the business logic behind RentaRoom is powered by FastAPI with Engine. The use of these technologies guarantees agile and reliable performance, which is essential for handling the operations involved in managing rentals. Landlords can thus manage their properties, documents, expenses and contracts efficiently, while tenants have easy access to relevant information and can communicate directly with landlords. Not only does this application simplify tenancy management, it also offers an additional feature of creating repair tickets. This allows tenants to quickly report problems with the property, facilitating timely maintenance and repair, making the living experience more satisfying and the process faster and more accessible for everyone involved.

In summary, RentaRoom is a web application that aims to optimize and modernize the room rental process through a user-friendly interface, the latest technology and additional features such as document management and repair tickets, thus making property rental a more efficient and smooth experience for owners and tenants.

Keywords: Real Estate Rental, FastApi, React, WebApp

Conteúdo

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

1.1 Enquadramento

No processo de arrendamento imobiliário é importante haver uma comunicação e gestão de processos entre o proprietário e o arrendatário. A solução para a desmaterialização destes processos é uma aplicação que simplifique e facilite a gestão a longo prazo de um arrendamento de quartos, de forma a que tanto o proprietário como o arrendatário tenham acesso a informação de despesas, recibos, contratos.

1.2 Objetivos

Este projeto tem como principal objetivo a facilitação da gestão e do processo de arrendamentos criando uma plataforma onde tanto o arrendatário como o proprietário possam ter acesso a toda a informação do arrendamento, centralizando assim todas as interações entre os mesmos num só local. Esta é uma plataforma com uma interface gráfica de utilizador desenvolvida em React, uma arquitetura assíncrona orientada a serviços desenvolvida em FastApi com Motor e uma camada de base de dados NoSQL desenvolvida em MongoDB. O objetivo é desenvolver uma plataforma web baseada numa arquitetura assíncrona orientada a serviços para gestão de arrendamento de quartos que suporta a gestão de contratos, mensagens internas, tickets de reparação, fracionamento de despesas

de água, luz, gás e internet, pagamento de rendas e emissão de recibos.

1.3 Estrutura do Documento

O desenvolvimento desta dissertação foi estruturado em seis capítulos, podendo estes ser descritos da seguinte forma :

- O capítulo dois representa uma pesquisa que foi feita de forma a descrever os conceitos, objetivos e inovações no setor imobiliário, tendo principal foco na gestão dos mesmos. Também neste capítulo são apresentadas soluções de gestão de propriedades já encontradas no mercado.

- O capítulo três contém uma breve abordagem de várias ferramentas e tecnologias que são relevantes para o desenvolvimento desta dissertação.

- O capítulo quatro tem como principal objetivo detalhar os requisitos para o desenvolvimento da aplicação, bem como mostrar as abordagens feitas para a estruturação da modelação da solução desenvolvida.

- No capítulo cinco são apresentados detalhes da solução final, explicando o desenvolvimento implementado na parte do backend e do frontend, onde é possível visualizar a interface gráfica desenvolvida.

- De forma a finalizar, no último capítulo são abordadas as conclusões que se retiram do desenvolvimento de todos os processos que envolveram a elaboração desta dissertação, incluindo também possíveis melhorias que poderiam ser adicionadas futuramente à aplicação desenvolvida.

Capítulo 2

Estado de Arte

Este capítulo tem como principal objetivo introduzir vários conceitos relacionados com o impacto de soluções de gestão de arrendamentos na indústria imobiliária. Na primeira secção serão abordados assuntos no âmbito das aplicações de gestão de propriedades de forma a proporcionar uma melhor compreensão dos objetivos dos mesmos. Na segunda secção são abordadas as vantagens e impactos destas soluções de forma a dar uma perceção de como estas poderão afetar de forma positiva a gestão de propriedades. Na terceira secção é apresentada uma visão geral de novas tecnologias que têm sido aplicadas nesta indústria e como as mesmas proporcionam avanços para este ramo. Por fim, na quarta secção são apresentadas várias soluções presentes no mercado e explicadas as vantagens e desvantagens de cada uma delas, proporcionando assim uma visão geral do estado atual das aplicações de gestão de arrendamentos.

2.1 Objetivos da Gestão de Propriedades

Desde o crescimento de investimentos por parte de particulares nos inícios de 1950 no ramo imobiliário [1], que a gestão imobiliária tem sido cada vez mais requisitada. Numa gestão de propriedades, vários pontos têm de ser tidos em conta, quando se visa o objetivo da mesma. Segundo Francis K. Loo [2] os objetivos desta gestão passam pela eficiência dos processos de gestão, minimizando com isto os seus custos. Também de forma a

evitar custos associados, uma especial atenção é tida em conta na qualidade dos serviços, limitando assim a possibilidade de um maior número de avarias futuras. Visando ainda a maximização financeira, a valorização da renda é um dos pontos a ser tido em conta. Por fim, um dos principais focos passa também por manter uma boa relação com os inquilinos, dependendo muito do sucesso da gestão feita nesta relação.

Na gestão de propriedades, estas podem ser separadas em duas diferentes visões: operacional e estratégica. A primeira tem o foco no curto prazo, preocupando-se mais com a garantia de qualidade nos processos diários, como tarefas associadas a tickets de reparação e submissão de despesas. A segunda visa de uma forma geral todas as decisões a longo prazo, focando-se mais nas decisões estratégicas das propriedades, de forma a que o impulsionamento financeiro seja tido em conta [3].

Inicialmente, a gestão de propriedades tinha como objetivo apenas o foco nos processos operacionais do ambiente físico da propriedade. No entanto, hoje em dia, a gestão destas propriedades tende a ser cada vez mais complexa com o aumento do acesso à tecnologia, evoluindo para uma incorporação de gestão de processos, gestão social, automatização, análise de dados e potencialização financeira [4]. A simplificação e automatização de tarefas proporciona assim uma maior agilidade e menos carga operacional no gestor da propriedade, possibilitando que este se foque mais na visão estratégica do seu negócio [5].

2.2 Impacto da Tecnologia na Gestão de Imóveis

As últimas décadas têm sido acompanhadas pela constante inovação e crescimento de aplicação de novas tecnologias em várias indústrias. Estes avanços tecnológicos e a maior facilidade de acesso a dispositivos tecnológicos por parte da população tem permitido que haja uma adoção ainda maior na forma como as indústrias desenvolvem soluções aplicadas aos mesmos [6]. A indústria imobiliária é uma das indústrias onde a revolução tecnológica tem vindo a ter impacto [7]. A gestão de processos de arrendamento desempenha um papel importante nesta indústria ao permitir que administradores de propriedades possam gerir as suas propriedades de forma eficiente. No entanto, esta gestão pode tornar-se

complicada ao adotar processos manuais na execução de tarefas exigidas por essa mesma gestão. O aumento de propriedades e de inquilinos aumenta exponencialmente a dificuldade de gestão [8]. Neste caso, informações sobre as propriedades e sobre os inquilinos necessitam de ser guardadas de forma organizada, como também é necessário assegurar que as despesas dos inquilinos são pagas e pedidos de reparação são solucionados. É aqui que a adoção de softwares de gestão de arrendamentos vem facilitar estes processos, sendo estes cada vez mais utilizados hoje em dia ao proporcionar uma maior eficácia, eficiência dessas tarefas e facilitando o acesso à informação, trazendo um maior valor para o proprietário, reduzindo os seus custos e tempo de gestão [3]. Entre todos os processos de gestão de arrendamentos, esses softwares possibilitam a centralização de toda a informação, facultando uma filtragem e acesso a grandes quantidades de informações, gestão de despesas, envio de faturas, comunicação com inquilinos, gestão de atividades de manutenção, elaboração de relatórios e automatização de tarefas [3]. A implementação destas funcionalidades facilitam e alavancam a eficiência dos envolvidos no processo, o que consequentemente traz um enorme valor para todos os envolvidos no processo de gestão de propriedades, tanto para quem faz esta gestão, como para quem usufrui do espaço gerido [4].

Resumindo, a tecnologia tem um papel importante na gestão de propriedades ao reduzir os custos de quem a gere, promovendo um maior valor e expondo áreas que podem ter lugar para melhoria, produzindo assim uma maior eficiência de gestão.

2.3 Tendências Tecnológicas no Mercado Imobiliário

Como já foi abordado nas secções anteriores, a introdução de um software de gestão de propriedades é crucial para a gestão de uma propriedade ao promover uma maior eficiência nas mesmas [9]. Nos últimos anos têm vindo a surgir várias abordagens de introdução de novas tecnologias e novas funcionalidades na indústria imobiliária.

De entre todas as implementações que foram surgindo neste ramo podemos destacar a

inclusão de inteligência artificial, blockchain, cloud computing, análise de big data, implementação de softwares em soluções móveis, utilização de tecnologias de realidade virtual, iot, entre outras. Um breve desenvolvimento e impacto de algumas destas tecnologias neste setor irá ser abordado nos seguintes parágrafos.

2.3.1 Soluções Baseadas em Cloud

As soluções cloud têm sido cada vez mais adotadas na última década em todas as áreas de desenvolvimento de implementações tecnológicas, devido ao facto de estas possibilitarem o aumento da sua escalabilidade, flexibilidade e redução de custos [10], [11]. As soluções que adotem por este tipo de computação, para além de não necessitarem de uma componente física, como servidores e armazenamentos , o que por si só já incorre em elevados custos de hardware e energia, podem também usufruir de um alívio financeiro ao não necessitarem de alocar recursos para a gestão de segurança destas infra-estruturas, nem manutenção das mesmas [10]. Estas soluções de computação em cloud trazem uma maior conveniência para aqueles que delas usufruem, proporcionando assim um fácil e rápido acesso a aplicações, dados, servidores e serviços de uma forma simples, direta e sem grandes custos nem grande esforço para a gestão da mesma [7]. Estas implementações possibilitam que pequenas empresas possam implementar soluções com baixos custos, ao permitir a alocação de dados em segurança e de forma escalável, desenvolver aplicações com este tipo de computação sem a necessidade de executar tarefas que se poderão tornar dispendiosas, estando estes serviços facilmente acessíveis através da internet [12].

Estas soluções trazem alguns benefícios quando aplicadas numa solução de gestão de propriedades, ao proporcionarem o acesso a funcionalidades ou informações que estão rapidamente disponíveis através de dispositivos com acesso à internet. Assim, aplicações de gestão de propriedades poderão estar acessíveis em qualquer dispositivo, não necessitando que sejam instaladas localmente no dispositivo, bem como os dados, que em vez de estarem alocados num armazenamento fixo, estarão acessíveis em qualquer local com ligação à internet.

Um caso de uma implementação de um software de gestão de propriedades com uma solução baseada em cloud foi abordada por Ikuomola A. J. and Asefon M.P [8]. Neste foi desenvolvida uma plataforma móvel utilizando tecnologias como Ionic, angular, laravel e MySql, estando a informação alocada numa nuvem, possibilitando que os dados estejam acessíveis de forma eficiente. Nesta aplicação estão presentes várias funcionalidades que tiram partido da computação em cloud, bem como do uso de uma solução móvel, como localização de propriedades através de uma integração com google map view, um sistema de pagamentos de forma a que transações dentro de cada propriedades possam ser feitas, autenticação, notificações, geração de relatórios tendo em conta as atividades de um utilizador, um chat para comunicação, pesquisa por registos, entre outros. Estando estas funcionalidades e informações todas disponíveis através da internet, torna muito mais eficiente e cómodo o acesso às mesmas, podendo ser possível executá-las e acedê-las em qualquer lado.

Em suma pode-se afirmar que soluções baseadas neste tipo de computação possuem uma vantagem em grande escala com o acesso facilitado às informações através de qualquer dispositivo, em qualquer localização, possibilitando uma maior eficiência e resposta de gestão [4].

2.3.2 Soluções Móveis

A adoção de soluções móveis refere-se a implementações que sejam focadas em distribuir um software que possa ser acessível através de dispositivos que ofereçam uma grande portabilidade, como tablets e smartphones. Este tipo de soluções, em comparação com as convencionalmente desenvolvidas para computadores, destacam-se pela sua capacidade de proporcionar aos utilizadores uma maneira de instalar e aceder a estas aplicações através de um dispositivo que o acompanha independentemente da localização para onde este pretender deslocar-se, estando assim acessível em todo o lado. Para além destas vantagens que o utilizador final pode usufruir, este tipo de desenvolvimento proporciona também algumas vantagens para os programadores e empresas, que desta forma podem

implementar as suas soluções com menos custos e de forma mais rápida [13]. No mundo das aplicações móveis, o facto de estas terem um mercado centralizado com todas as aplicações para que os utilizadores possam pesquisar pelas mesmas e fazer o download de forma facilitada, também é uma vantagem comparando com a forma usual de pesquisa e download de um software para computador. As soluções móveis conseguem ainda integrar funcionalidades que aproveitando capacidades que normalmente os computadores não possuem e em contrapartida a maioria dos dispositivos móveis atuais possuem, sendo estes o barómetro, giroscópio, gps, acelerômetro, sensores de proximidade, entre outros [14].

O mercado das soluções móveis expandiu-se de forma muito rápida com grande interesse nas vantagens que estes dispositivos móveis proporcionam aos utilizadores. Toda esta expansão iniciou-se com a introdução do primeiro smartphone em 2007, lançado pela Apple, e até aos dias de hoje este mercado expandiu-se de tal forma que 60.89% de todo o tráfego de internet é acedido através de dispositivos móveis [15].

Tendo as vantagens da utilização de soluções móveis em conta, é possível afirmar que estas proporcionam uma grande vantagem na gestão de propriedades, possibilitando uma resposta rápida a necessidades [4], ao poder executar funcionalidades a partir de um dispositivo que estes podem aceder em qualquer parte do mundo de forma muito cómoda, como pagamento de despesas e análise de dados.

2.3.3 Inteligência Artificial e Big Data

A inteligência artificial e big data são conceitos que têm tido um exponencial interesse e aplicação nos vários setores ao longo dos últimos anos [16]. Este interesse tem crescido devido às várias vantagens que estas potencializam num negócio que envolva tratamento de grandes quantidades de dados, proporcionando que as análises e processamento eficiente destes dados se traduzam em estratégias e previsões, e consequentemente em ganhos financeiros futuros [17].

A big data refere-se à coleta de um grande volume de dados e à consequente análise

dos mesmos, possibilitando assim que se possam encontrar padrões nestas análises. A inteligência artificial, em particular machine learning, tira grande partido destas grandes coletas e análises de dados. Estes algoritmos tiram o partido de aprender através de grandes coleções de dados, fazendo com estes previsões daquilo que poderá suceder no futuro, ao avaliar vários fatores que seriam impossíveis de serem avaliados por um humano [17].

Este processamento e análise de dados potencializa o setor imobiliário ao proporcionar a melhoria na avaliação de decisões por parte de empresas. Para além destas vantagens, isto possibilita que os gestores gastem menos tempo a analisar e interpretar quantidades enormes de dados e se foquem noutras atividades relacionadas com a gestão [4].

2.3.4 Internet of Things

A definição de Iot(Internet of things) refere-se à interconexão da internet com dispositivos, como sensores, que possibilitam a obtenção de informações. Os vários sensores que podem ser utilizados podem agregar alguns dados relativamente à temperatura, luz, movimento, humidade, níveis de oxigénio, entre outros [4]. Este tipo de tecnologias permite que todo o tipo de condições físicas sejam traduzidas para dados, possibilitando também a inclusão da aplicação de machine learning neste tipo de tecnologias [17].

Estes tipos de tecnologias podem ser aplicados na gestão de propriedades, de forma a tornar as propriedades inteligentes. A adoção que estas tecnologias na manutenção e gestão de uma propriedade podem ter um impacto positivo ao possibilitar que operações se tornem mais eficientes e automatizadas, aliado ao facto de estas poderem também reduzir custos financeiros com a redução de energia [7].

2.4 Soluções Existentes no Mercado

À medida que o setor imobiliário continua a evoluir, estas aplicações desempenham um papel fundamental na simplificação e eficiência da administração de propriedades arrendadas. Cada aplicação tem as suas vantagens, focando estas em diferentes recursos e

funcionalidades de forma a colmatar as várias necessidades de proprietários, inquilinos e profissionais do mercado imobiliário. Ao analisar esses exemplos, é possível destacar as diferentes qualidades e focos de cada aplicação de forma a tentar proporcionar uma experiência de alta qualidade para todas as partes envolvidas. Entre as várias soluções do mercado destacam-se as seguintes :

- Rentila: [18] Esta solução surgiu em meados de 2011, em Paris, por Olivier Goy, um empreendedor que tinha como objetivo trazer uma solução para o mercado que pudesse simplificar o quotidiano dos proprietários e inquilinos no mercado imobiliário. Esta aplicação de gestão de propriedades oferece uma variedade de recursos para proprietários e inquilinos. As características que mais se destacam são a facilidade de criação e a gestão de contratos de arrendamento de forma eficiente. Permite também a facilidade no rastreamento e organização dos pagamentos de arrendamento. Oferece ainda a conveniência de armazenar documentos relacionados às propriedades de forma digital. Esta aplicação simplifica a gestão de contratos e pagamentos de despesas, tornando o processo mais eficiente para proprietários e inquilinos.

- Gesar: [19] Gesar é uma plataforma de gestão imobiliária que abrange desde a administração de propriedades até à contabilidade e a interação com inquilinos. As características que mais se destacam incluem a automatização de tarefas rotineiras, como lembretes de pagamento e manutenção. Oferece ainda a capacidade de criar relatórios detalhados de acordo com as necessidades específicas do propritário. Tem ainda algumas funcionalidades que previgiliam a comunicação eficiente com inquilinos.

- VE-Imóveis: [20] VE-Imóveis é uma plataforma que oferece funcionalidades que permitem a gestão, incluindo uma interação fácil e eficiente entre proprietários e inquilinos, bem como entre vendedores e possíveis compradores das propriedades.

- Buildium: [21] Buildium é uma aplicação que proporciona gestão de propriedades que oferece uma ampla gama de funcionalidades. As características que se destacam

nesta solução são o acompanhamento e gestão de informações detalhadas sobre inquilinos bem como a facilitação de processos de contabilidade relacionadas com os arrendamentos e as suas despesas [22]. Possui também recursos que possibilitam uma comunicação eficaz com inquilinos e proprietários.

Cada uma dessas aplicações possui características diferentes que podem ser vantajosas para diferentes tipos de proprietários e gestores de propriedades. Estas podem colmatar diferentes necessidades dependendo do perfil do gestor, trazendo estas aplicações uma vasta oferta para os proprietários, tendo estes a possibilidade de escolher a solução que mais se adequa às suas necessidades.

Capítulo 3

Tecnologias de desenvolvimento

Neste capítulo irão ser abordadas as tecnologias de desenvolvimento utilizadas na construção desta aplicação web, bem como uma revisão geral das tecnologias no mercado nesta área da tecnologia.

3.1 Evolução da WEB

As aplicações Web, representam uma evolução significativa na forma como interagimos com a internet e com a tecnologia num todo. A maneira como utilizadores interagem hoje em dia com as várias aplicações web deve-se a à grande evolução das tecnologias que foram aparecendo nas últimas décadas para o desenvolvimento das mesmas [23]. Nos primórdios da internet, na década de 1990, a maioria das páginas que estavam presentes na web eram estáticas e raramente o utilizador interagia com as mesmas. Esta era foi denominada por WEB 1.0, a primeira versão da web, inventada por Tim Burners-Lee, e é conhecida também por ser a fase na qual os utilizadores estavam limitados a apenas procurar e ler informação das páginas [24].

A WEB 2.0 surgiu em 2004, com a queda da WEB 1.0 devido à pouca interação do utilizador com as páginas. Esta foi introduzida por Dale Dougherty como a versão WEB da escrita e leitura de informação e da web que visava uma participação maior por parte dos utilizadores [25]. O sucesso desta versão da web deveu-se à possibilidade de interação

dos utilizadores com as páginas, promovendo assim um grande interesse por parte dos mesmos, de programadores e também de empresas [26]. A WEB 2.0 proporcionou interfaces mais responsivas, conteúdo dinâmico e promoveu a colaboração dos utilizadores. Estas novas funcionalidades, possibilitaram o aparecimento de novas aplicações atualmente conhecidas como MySpace e Youtube que foram ganhando grande relevância, bem como blogs e wikis [24]. Esta versão ficou ainda marcada pelo desenvolvimento de um javascript asíncrono, proporcionado com o desenvolvimento através de javascript e AJAX, fazendo com que as aplicações web pudessem alterar dados visualizados na página, sem a necessidade de atualizar a mesma [27], [28].

O surgimento da Web 3.0 aconteceu em 2006 por John Markoff, definida como web semântica, e tinha como propósito fazer com que a web pudesse ser interpretada não só pelos humanos mas também pelos próprios computadores. A ideia centralizava-se na estruturação de dados, possibilitando assim melhores pesquisas, automatização e integração de toda a web. Esta evolução proporcionou que hoje em dia, utilizadores pudessem ter uma melhor experiência ao interagir com as aplicações web, estando estas completamente presentes no seu quotidiano.

3.2 API

Numa aplicação web é necessária a comunicação entre a parte visual, que irá apresentar a informação que o utilizador poderá visualizar, e a base de dados que possui essa informação. As APIs (Aplication programming interface) fazem a ponte entre estes dois extremos. Estas interfaces são responsáveis por definirem certas regras de forma a que estes dois extremos se comuniquem de forma efetiva, fazendo uso do protocolo http. Em termos práticos, estas interagem de forma a enviar um pedido por parte do cliente para o servidor através de http, informando ao servidor o método http a ser usado e os recursos a ser passados. De seguida, após o processamento dos recursos por parte do servidor, retornar a resposta do pedido para o cliente [29].

Estas interfaces foram primeiramente referidas em 1968, por Frank S. Greatorex e

Ira W. Cotton na publicação de um artigo intitulado "Data Structures and techniques for remote computer graphics" [30] e posteriormente num artigo de CJ. Date, em 1974, com a publicação do seu artigo "The Relational and Network Approaches: Comparison of the Application Programming Interface" [31]. Este evento teve um grande impacto na tecnologia, proporcionando uma grande expansão no desenvolvimento e surgimento de APIs e sua utilização em massa no setor tecnológico [32], [33].

De forma a definir quais as regras e estruturas entre as interações destas interfaces [34], foram surgindo vários protocolos, sendo o REST API e SOAP API dos mais populares atualmente e serão abordados nas próximas duas sub-secções [35].

3.2.1 REST API

O Rest (Representational State Transfer) foi introduzido pela primeira vez por Roy Fielding em 2000, onde este realizou uma abordagem com o Rest, como sendo um estilo arquitetural de forma a definir algumas regras para os serviços web [36]. Na sua dissertação, vários pontos foram definidos de forma a descrever as delimitações às quais serviços web com estilo de arquitetura Rest deveriam seguir [29], [36], [37]:

- "Code-on-demand" - Esta delimitação é opcional e tem como principal objetivo promover uma maior extensibilidade, ao promover o envio de execução de código através do servidor para o cliente Rest.

- Sistema por camadas - Como o nome indica, esta delimitação tem como função aplicar um sistema composto por várias camadas, onde cada uma dessas camadas é responsável pelo seu papel, de forma a obter vantagens em termos de encapsulamento de serviços, bem como tirar vantagem de um sistema mais escalável. Apesar destas vantagens, estas camadas fazem com que exista uma maior lentidão na maneira como os dados são processados.

- Interface uniforme - Esta interface uniforme que é referida tem como objetivo definir que apenas uma linguagem é definida para fazer a comunicação entre o servidor e o

cliente.

- Cache - Esta restrição tem como finalidades possibilitar que respostas possam ser armazenadas em cache, de forma a promover uma maior velocidade na obtenção de dados por parte do cliente. Estas respostas podem assim ser cacheable, sendo assim armazenadas na cache do cliente, ou não cacheable dependendo do que for definido na mesma. Ao afirmar-se ser cached, esta irá conter também a informação de até quando a mesma ficará disponível em cache.

- Sem estado - Esta é outra delimitação referida por Roy e tem como objetivo definir que cada uma das comunicações feitas entre cliente e servidor é independente. Isto faz com que cada comunicação tenha os dados necessários para finalizar com sucesso a mesma, proporcionando uma melhor fiabilidade, visibilidade de informação de cada pedido e resposta e também a escalabilidade de comunicações.

- Cliente - Servidor - Esta restrição proporciona uma separação relacional entre o cliente e o servidor, fazendo com que as mudanças e novas funcionalidades de um não interfira com o estado operacional do outro.

3.2.2 SOAP API

O SOAP (Simple Object Access Protocol) é um protocolo usado para estas interações de serviços web, que usa xml como estrutura para a troca de dados num sistema entre consumidores e provedores [35], [38]. Este protocolo tem como propriedades a ausência de estado e a unidirecionalidade das suas trocas de mensagens entre nós. No SOAP existem três tipos principais de nós [38]:

- Remetente - Envia a mensagem SOAP.

- Destinatário - Recebe a mensagem SOAP e processa a mesma de forma a enviar a resposta.

- Intermediário - Os intermediários são ambos o remetente e o destinatário onde o mesmo recebe as mensagens remetidas a ele e envia a resposta para o destinatário.

Tendo em conta a estrutura que o SOAP usa para as mensagens presentes nesta comunicação, estas podem ser separadas pelo envelope que define o início e fim da mensagem, definindo assim os limites da mesma e identificando a mensagem como SOAP. Nestas mensagens está também presente o cabeçalho que apresenta informação adicional como autenticação, linguagem, entre outros. O corpo da mensagem também faz parte da estrutura da mensagem ao conter os dados necessários do pedido ou resposta, estando este estruturado em XML. Por fim temos o elemento Fault que indica as anomalias que foram encontradas na solicitação do pedido [38].

3.2.3 Comparação RestAPI vs SoapAPI

Estas dois tipos de serviços têm as suas vantagens e desvantagens no que diz respeito à comunicação e às restrições que definem cada uma. Por um lado o Rest, ao contrário do SOAP, não exige uma estrutura rígida, pelo que é importante que programadores definam convenções de nomenclaturas e um design organizado de forma a que o uso destas API Rest seja mais fácil e consistente [39]. Este tem também várias vantagens no que diz respeito à facilidade de uso, eficiência, e rapidez [39], [40]. Também quanto aos tipos de dados que suporta na resposta a pedidos, este ganha ao SOAP, que apenas suporta XML [38]. O SOAP por sua vez destaca-se pela sua segurança, e por consequência possui uma maior adaptabilidade no setor do negócio. No entanto esta segurança exige que hajam requisitos como uma estrutura rígida, que levam à presença de algumas desvantagens em comparação com o REST, como a necessidade de mais recursos computacionais e de uma maior largura de banda [39].

3.2.4 HTTP

O Hypertext Transfer Protocol (HTTP) é um protocolo de comunicação que permite a transferência de dados entre navegadores web e servidores web. Este protocolo desempenha um papel crucial na interação entre o utilizador e a web, ao permitir que este aceda a sites, envie formulários, visualize imagens, transmita vídeos e realize praticamente todas as atividades na internet.

O HTTP é um protocolo sem estado, o que significa que cada comunicação entre um browser e um servidor é completamente independente das solicitações anteriores [41].

Nos cabeçalhos HTTP estão presentes parâmetros adicionais que são enviadas juntamente com a solicitação ou resposta HTTP. Estas fornecem metadados sobre a solicitação ou resposta na qual podem incluir informações sobre a autenticação, tipo de conteúdo, cache, idioma. Os cabeçalhos são importantes nestes pedidos de forma a possibilitar uma comunicação entre o servidor e o cliente de forma a que ambos percebam como processar os dados do pedido [41].

O HTTPS foi criado para proteger a privacidade e a segurança dos dados enviados pela web. O HTTPS, uma extensão do HTTP, usa criptografia para proteger os dados durante a transmissão. Este tipo de segurança é necessária tendo em conta que informações confidenciais como senhas e dados de cartão de crédito são enviados através destes pedidos. Cada resposta HTTP é acompanhada por um código de estado do pedido, indicando o resultado do mesmo. Estes códigos contêm 3 dígitos e são divididos em categorias. Quando um código de estado começa por um dígito 2 então este indica que o pedido foi executado com sucesso, um dígito 3 refere-se a um redirecionamento, um dígito 4 para erros do que ocorreram no lado do cliente e um dígito 5 para erros que ocorreram do lado do servidor. De todos os códigos existentes, os que mais se destacam pela sua maior utilidade e conveniência em aplicações Web são os seguintes [41] :

Tabela 3.1: Códigos de Resposta HTTP

Código	**Descrição**
200	OK: O pedido ocorreu com sucesso, e o servidor devolveu os dados pedidos.
201	Criado: O pedido foi cumprido e resultou na criação de um novo recurso.
204	Sem Conteúdo: O servidor processou o pedido com sucesso, no entanto não existe conteúdo adicional para enviar na resposta.
400	Pedido Inválido: O pedido do cliente é inválido.
401	Não Autorizado: É necessária autenticação, e as credenciais fornecidas encontram-se inválidas ou não são fornecidas.
403	Proibido: O cliente tentou fazer um pedido que o cliente está proibido a aceder.
404	Não Encontrado: O recurso solicitado não existe não foi encontrado.
500	Erro Interno do Servidor: Ocorreu um erro interno do servidor.
503	Serviço Indisponível: O servidor está temporariamente indisponível.

O HTTP usa métodos para determinar o que deve ser feito em um recurso que é reconhecido por uma URL. A tabela a seguir mostra os métodos [42]:

Tabela 3.2: Métodos HTTP

Método	**Descrição**
GET	Solicita um recurso específico.
POST	Envia dados que serão processados por um recurso específico.
PUT	Atualiza um recurso específico ou cria um novo se ele não existir.
DELETE	Remove um recurso específico.
HEAD	Solicita apenas os cabeçalhos de resposta de forma a que o corpo da resposta não seja recebido.
OPTIONS	Descreve quais as opções de comunicação para o recurso.
PATCH	Aplica modificações parciais a um recurso.
CONNECT	Estabelece uma comunicação para um dado servidor.
TRACE	Executa um loop de forma a fazer diagnóstico ao longo do caminho para o recurso de destino.

Apesar destes métodos todos os que são convenientemente mais utilizados são o GET, POST, PUT, DELETE, OPTIONS e PATCH.

3.3 Python

O python é uma linguagem de programação introduzida em meados de 1991 por Guido Van Rossum, tendo como objetivo a facilidade de compreensão e utilização, sendo hoje em dia uma das principais escolhas dos programadores [43].

De acordo com a análise feita em 2022 pela IEEE Spectrum's [44], constatou-se que o python é a linguagem mais popular entre todas as linguagens de programação existentes. As aplicações desta linguagem em casos reais são ainda algumas, sendo as mais relevantes o uso desta linguagem por parte da Netflix para o processamento de dados de forma a analisar o histórico de visualizações dos utilizadores, possibilitando assim sugestões personalizadas. Também outros gigantes do mundo da tecnologia como a Google e o Facebook usam a mesma para o desenvolvimento do Youtube e do Instagram, respetivamente [45]. Com a sua evolução, foi começando a ser maioritariamente adotada nas áreas de inteligência artificial, machine learning e ciência de dados, devido ao seu grande número de bibliotecas, que permitem a fácil construção de aplicações nestas áreas [46]. Para além da sua aplicação nestas áreas de desenvolvimento, esta linguagem também é bastante requisitada para a implementação de APIs, através das várias frameworks existentes no mercado. Entre estas, as mais conhecidas são o Flask, o Django e o FastApi.

3.4 FastAPI

A framework de python, fastAPI , criada em 2018 e suportada pelo python 3.6 [47], tem tido alguma relevância durante os últimos anos, sendo uma das frameworks de python para desenvolver APIs mais utilizadas, juntamente com flask e Django [48]. Esta framework veio ganhando espaço no mercado devido a uma das grandes necessidades que não era colmatada pelas frameworks já referidas, a velocidade [49]. Esta é conhecida não só pela sua rapidez de implementação e processamento devido às suas capacidades assíncronas, mas também devido à sua capacidade de escalabilidade num projeto. Outras vantagens que esta framework também possibilita é a validação automática dos dados dos pedidos

e respostas com recurso a sugestões na digitação promovidas pelo python, de forma a minimizar os erros em execução, permitindo também assim manter uma correta integridade dos dados. O fastAPI proporciona também injeção de dependências, que permite aos programadores uma mais fácil gestão destas mesmas dependências e também uma forma de reutilização de código [50].

Quanto às outras frameworks para o desenvolvimento de API em python já referidas, o django foi a que surgiu primeiro, no ano 2005 e tem como principais vantagens proporcionar uma grande escalabilidade, customização e ter por defeito tudo o que o programador necessita para a criação da API. Mais tarde, em 2010, a outra framework intitulada de Flask surgiu com algumas principais qualidades como a sua leveza, robustez e facilidade com que o programador tem em criar uma API do zero [51]. Qualquer uma destas frameworks seria uma ótima escolha para o desenvolvimento desta dissertação, no entanto a escolhida foi o fastAPI, pela facilidade de implementação e rapidez.

3.5 HTML e CSS

Desde o começo da existência dos websites que o HTML (HyperText markup language) e o CSS (Cascading style sheets) andam lado a lado no desenvolvimento dos mesmos. O HTML tem como finalidade estruturar o documento apresentado pelo site através de tags e propriedades. Estas tags são apresentadas num esquema de árvore correspondendo posteriormente ao que irá aparecer visualmente ao utilizador [52]. Já o CSS é denominada por ser a folha de estilos da página, tendo como principal objetivo a formatação visual do documento estruturado pelo HTML, aplicando estilos ao mesmo e tornando-o dinâmico em termos de aspeto [53].

3.5.1 Evolução do HTML e CSS

O HTML aparece pela primeira vez em 1990 onde sofre uma grande quantidade de revisões e melhorias nos primeiros 5 anos onde pelo meio é lançada a 1ª versão HTML 1.0 com a intenção de partilhar informação que poderia ser interpretada pelos navegadores web, no

entanto nesta altura a linguagem não evoluiu devido ao pouco interesse demonstrado por parte dos programadores no desenvolvimento Web. Em 1995 é lançada uma nova versão, sendo esta denominada por HTML 2.0, tendo principais novas funcionalidades como elementos estruturais para parágrafos, cabeçalhos e listas ordenadas e não ordenadas. Ainda no mesmo ano foi lançado uma nova versão , o HTML 3.0 que implementou características melhoradas que permitiam ao programador desenvolver uma interface web mais poderosa, mas que no entanto perdia alguma performance. Em 1997 foi lançada a versão HTML 4.0 que visava entre algumas novas funcionalidades, incluir o uso de scripts numa página em HTML. Por fim, a última versão principal HTML 5.0 é lançada em 2008 oferecendo desta vez, entre muitas outras funcionalidades novas, a possibilidade de os programadores incluírem áudio e vídeo nas suas páginas e outras tags que são ainda bastante usadas hoje em dia [54].

Quanto ao CSS, este surgiu em 1994 por Håkon Wium Lie e o seu propósito foi dar vida às páginas criadas em HTML de forma a estes poderem ter estilos dinâmicos. Entretanto em 1996 o CSS é pela primeira vez lançado tendo como denominação o nome de CSS 1 e posteriormente em 1998 é publicada uma nova versão CSS 2 que proporciona aos programadores a oportunidade de adaptar as suas páginas a diferentes dispositivos como impressoras, portáteis, etc. Com o CSS 3 foram introduzidas novas funcionalidades para simplificar o desenvolvimento como a divisão do CSS em vários módulos, o que permitiu aos programadores ter uma maior facilidade em gerir o CSS em projetos em larga escala. Nesta versão também foram lançadas outras funcionalidades como a possibilidade de desenvolver melhores animações, melhor facilidade de lidar com bordas, entre outras [55]. Com esta evolução veio também o aparecimento de frameworks que disponibilizavam ao programador estilos já pré-definidos, como é o caso do BootStrap e Tailwind, sendo estes bastante utilizados. Para além destas frameworks foram surgindo pré-processadores que implementaram no CSS uma forma de usar variáveis, intercalamento e lógica, como por exemplo o SASS e LESS [56].

3.6 Javascript

Nos tempos atuais o desenvolvimento web não é só construído a partir de páginas estáticas em HTML e CSS como era nos primórdios da internet. Com o tempo os programadores começaram a introduzir scripts no HTML de uma página com o objetivo que esta pudesse usufruir de alguma lógica, de forma a tornar a página mais dinâmica e interativa, sendo hoje uma parte essencial no desenvolvimento web.

O JavaScript surge em 1995 por Brendan Eich onde foi inicialmente lançado em versões Beta do Netscape Navigator 2.0 . O javascript proporcionou a aplicação de scripts aplicada às páginas web, permitindo assim a manipulação dinâmica da DOM (Modelo de Objeto de Documentos) e mais tarde com as tecnologias AJAX(Asynchronous JavaScript and XML) a manipulação de dados em segundo plano [57], [58].

Para além destas aplicações do javascript na web, este começou a ser aplicado para soluções de desenvolvimento back-end. Uma das frameworks que ganhou bastante relevância foi o NodeJS que possibilita ao programador construir APIs utilizando o javascript. Este foi um evento que proporcionou uma grande revolução na maneira como foi apartir daí a percepção dos programadores com o javascript, possibilitando apartir deste momento que tanto o desenvolvimento da parte do servidor, como o desenvolvimento da parte do cliente fossem possíveis com o javascript [58].

3.7 React

O React é uma biblioteca do javascript criada por Jordan Walke, tendo ficado open-source em 2013, juntado-se assim a várias bibliotecas/frameworks presentes em javascript como Angular, JQuery, Vue, Meteor, entre outros [59]. De acordo com o site “State of Javascript” [60] a estatística de uso de frameworks/bibliotecas front-end do javascript, o react é a mais usada ao longo dos últimos 6 anos, estando em segundo lugar o Angular e o Vue.

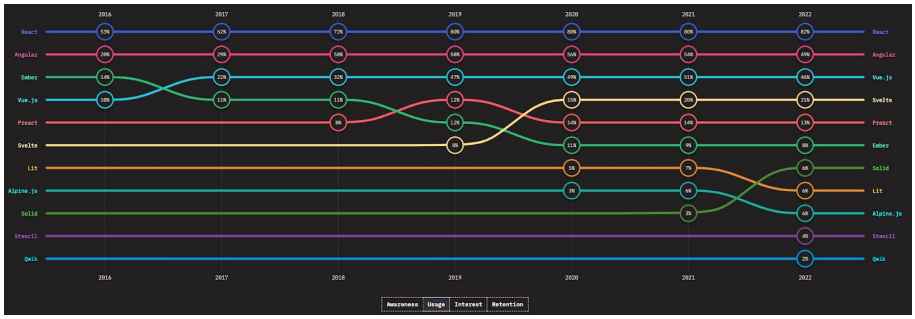

Figura 3.1: Uso de Frameworks front-end do javascript segundo o site “State of Javascript”.

O react destaca-se pelas várias propriedades que possui. Entre elas podem destacar-se que o react implementa solução declarativa, possibilitando assim que o programador possa mudar estados dos componentes, fazendo o react o resto do trabalho, atualizando de forma dinâmica a DOM à medida que estes estados mudam [61], [62]. Isto é possível tendo em conta que o react faz uso de uma DOM virtual. Esta DOM virtual irá manter em memória aquilo que seria a estrutura da DOM de acordo com os estados presentes nos componentes, e quando estes estados mudarem, a DOM virtual fará uma comparação com a anterior representação da mesma, e de seguida a DOM real é sincronizada com a DOM virtual, atualizando apenas os estados necessários [63]. Isto faz com que haja uma grande melhoria de performance [61], [62], fazendo com que uma aplicação web seja baseada em apenas numa só página, as chamadas aplicações SPA (Single Page Aplication).

Adicionalmente vale a pena referir que o react usa jsx de forma a que o programador defina qual a estrutura de componentes a ser apresentada na DOM, sendo esta uma extensão do javascript. Desta maneira, a lógica de negócio e a estrutura gráfica podem aliar-se no mesmo ficheiro, tornando o jsx numa poderosa ferramenta para os programadores [62], [63].

Esta biblioteca tem sido cada vez mais usada, estando inclusive presente no desenvolvimento de várias aplicações bem conhecidas como Facebook, Netflix, Discord e Instagram [64].

3.8 Material UI

Para além da importância das funcionalidades e interação com os componentes visuais na parte gráfica de uma aplicação web, é relevante que haja uma boa apresentação na maneira como os componentes são visualizador pelo utilizador. O Material UI é mesmo isso que proporciona ao programador. Este é uma biblioteca de componentes do react, que permite a utilização de componentes já pré feitos, possibilitando uma mais rápida implementação da intereface gráfica por parte do programador, sem ter que abdicar de interface gráfica agradável e moderna [65]. Esta biblioteca de componentes segue as diretrizes do Google Material Design, promovendo assim uma experiência consistente para quem a usa [66].

O Material UI proporciona uma vasta quantidade de componentes como botões, alertas, gráficos, paginação, entre outros, possibitando assim uma melhor experiência de desenvolvimento, e menos esforço na implementação da mesma.

3.9 MongoDB

O MongoDB é uma base de dados não relacional (NOSQL) cujo modelo de dados é orientado a documentos. Isto significa que em vez de armazenar dados nas convencionais tabelas, adicionando uma linha a cada inserção de informação, esta tecnologia possui coleções em vez de tabelas, na qual estas possuem vários documentos JSON de forma a guardar os vários dados de um registo [67], como se pode observar pela seguinte ilustração:

Figura 3.2: Estrutura de um documento de dados em MongoDB.

Como pode ser visualizado na figura acima, cada documento tem o seu identificador único, e o resto do documento é mapeado através de pares de nomes de propriedades e os seus valores, podendo estes valores serem strings, números, arrays, datas ou mesmo documentos JSON [68]. Estas bases de dados NOSQL foram desenvolvidas de forma a que grande quantidade de dados pudessem ser processadas, não tendo que obedecer a uma estrutura de dados mais rígida. Isto permite que grandes quantidades de dados possam ser acedidos, analisados e processados em tempo real, permitindo assim uma maior eficiência, escalabilidade e flexibilidade no uso destas abordagens [67].

Para além das várias vantagens, o MongoDB destaca-se também pelos seus vários tipos de indexação, como indexação secundária, geoespacial e de texto, permitindo assim uma maior rapidez na procura de dados por parte de execução de queries. Destaca-se também pela possibilidade de executar operações em vários documentos numa só transação, permitindo assim execução de operações mais complexas e eficientes em apenas uma transação [69].

Esta tecnologia está a ser usada por várias soluções de grandes empresas no mercado tecnológico. De entre várias implementações, pode-se destacar o uso de MongoDB no eBay para o armazenamento de grandes dados como perfis e histórico de atividades, AWS da amazon, Uber para armazenar dados relacionados com os utilizadores que usufruem

da aplicação, Cisco, entre outros [70].

Capítulo 4

Análise e Modelação do Sistema

O processo de análise e modelação é essencial para perceber como certa aplicação irá ser implementada, tentando assim escolher quais irão ser as ferramentas utilizadas para o desenvolvimento da solução, criar mockups de forma a visualizar um esboço daquilo que será a aplicação, fazer um levantamento de requisitos com o objetivo de entender quais são as necessidades que precisarão de ser colmatadas e organizar estes objetivos de forma a podermos criar um solução de forma escalável, bem pensada e corretamente executada.

Neste capítulo será abordada toda a parte da análise e modelação. Na secção 4.1 serão demonstrados alguns dos esboços criados para ter uma visão de como seria a interface gráfica da aplicação. A segunda secção 4.2, tem como objetivo representar os requisitos funcionais e não funcionais levantados, mostrando-os visualmente num diagrama de casos de uso. Nesta secção serão também descritas todas as user stories identificadas de forma a organizar e visualizar em melhor pormenor os requisitos funcionais do sistema a ser desenvolvido. Na secção 4.3 irá ser abordada a modelação da base de dados, mostrando todas as tabelas e propriedades de casa entidade envolvida. Por fim, na última secção deste capítulo estarão demonstradas todas as atividades que os utilizadores intervenientes poderão fazer, estando este visualizado num diagrama de atividades.

4.1 Requisitos Funcionais e não Funcionais

Para o desenvolvimento de uma aplicação é fundamental que haja uma ponte entre o que irá ser desenvolvido e as necessidades e expectativas do cliente para com a solução final. Com este intuito, entender, coletar e aplicar os requisitos dos utilizadores da aplicação é de extrema importância para o desenvolvimento da mesma [71]. Existem várias maneiras de obter estes requisitos. Entendimento dos casos de uso, entrevistas, grupos de foco, experiências são alguns dos exemplos para esta finalidade [71].

Um dos principais diagramas usados para a especificação de requisitos é o diagrama de UC [72]. Os UC, possibilitam a visualização gráfica das interações entre os atores e as funcionalidades que necessitam de desempenhar ao interagirem com a aplicação[72][73], conseguindo desta forma visualizar graficamente.

Com este intuito, foram desenvolvidas primeiramente, um diagrama de casos de uso da aplicação a ser desenvolvida, levantando assim as possíveis necessidades dos arrendatários e dos proprietários.

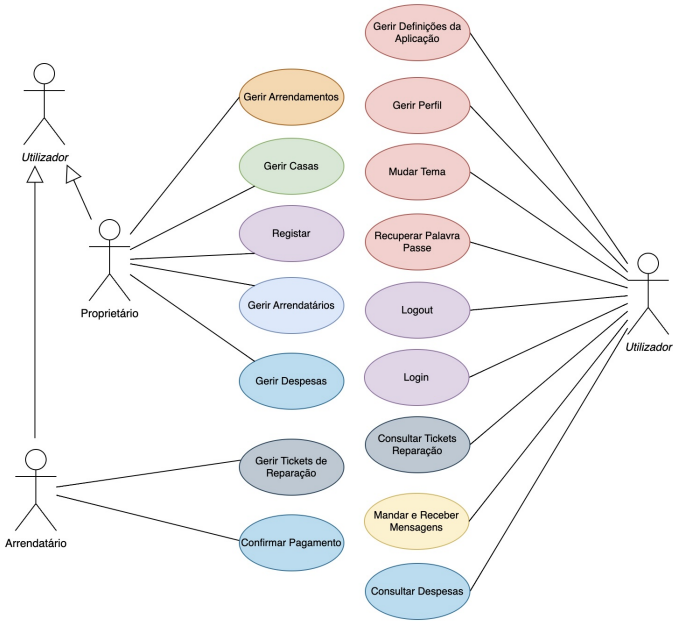

Figura 4.1: Diagrama de Casos de Uso.

Estas funcionalidades foram retiradas tendo em conta os objetivos finais da aplicação que tinham como visão a gestão de contratos, mensagens internas, tickets de reparação, gestão de despesas, pagamentos de despesas e emissão de recibos. As necessidades dos utilizadores intervenientes foi construída tendo em conta estes objetivos, refletindo assim na contruação do diagrama de casos de uso em cima apresentado. De forma a entrar em mais detalhe e poderem estes requisitos traduzirem-se num desenvolvimento ágil foi necessário traduzir os mesmo para user stories.

A técnica das user stories é uma forma de captar estas necessidades e facilitar o entendimento durante o desenvolvimento do que é pretendido [74]. As user stories são coletadas a partir de ações que o utilizador pretende desempenhar e compiladas em breves descrições da perspectiva do utilizador [75]. Estas descrições mostram as funções que o sistema ou produto deve atender [76], que incluem coisas além das características técnicas, como coisas que retornam valor ao utilizador [77].

O principal objetivo desta técnica é trazer independência para cada uma das funcionalidades, fazendo com que cada uma tenha o seu desenvolvimento, permitindo que sejam testadas de forma autónoma, de maneira a garantir que estas cumpram com os requisitos. Para além disso possibilita que o desenvolvimento de cada user story seja moldado consoante o feedback dos stakeholders, de forma a que haja mudanças dos requisitos tendo em conta as necessidades dos mesmos. A acrescentar a estes benefícios, destaca-se também a estimabilidade que é atribuida a cada user story, de forma a perceber a dificuldade das mesmas e o valor que esta técnica traz para o desenvolvimento de um projeto [78]. Tendo em conta estes benefícios, foram desenvolvidas as User Stories da plataforma, de forma a poder usufruir das vantagens das mesmas. Estas podem ser visualizadas nas seguintes tabelas:

USU1	Fazer Login
USU2	Fazer Logout
USU3	Recuperação de conta
USU4	Visualizar as Configurações
USU5	Mudar Palavra Passe
USU6	Mudar Tema da Aplicação
USU7	Editar Informações de perfil
USU8	Editar Configurações de Notificações
USU9	Consultar Despesas
USU10	Visualizar Despesa
USU12	Enviar Mensagens
USU13	Visualizar Mensagens
USU14	Consultar Recibos
USU15	Visualizar Tickets de Reparação
USU16	Consultar Arrendamentos
USU17	Ver Detalhes de Arrendamento

Tabela 4.1: User Stories de Arrendatários e Proprietários

USP1	Criar Casa
USP2	Editar Casa
USP3	Eliminar Casa
USP4	Consultar Casas
USP5	Visualizar Casa
USP6	Criar Arrendatário
USP7	Editar Arrendatário
USP8	Eliminar Arrendatário
USP9	Consultar Arrendatários
USP10	Visualizar Arrendatário
USP11	Criar Arrendamento
USP12	Editar Arrendamento
USP13	Eliminar Arrendamento
USP14	Consultar Arrendamentos
USP15	Visualizar Arrendamento
USP16	Terminar Arrendamento
USP17	Emissão de Recibos
USP18	Fazer Registo
USP19	Mudar estado Tickets de Reparação
USP20	Mudar estado Despesa

Tabela 4.2: User Stories relativas ao Proprietário

USA1	Pagar despesa
USA2	Criação de Tickets de Reparação

Tabela 4.3: User Stories relativas ao Arrendatário

Com as US definidas foi possível perceber melhor o desenvolvimento a ser feito, possibilitando também uma melhor organização na distribuição de tarefas pelas várias etapas de desenvolvimento, dando também uma melhor percepção daquilo que já foi implementado e o que ainda falta.

4.1.1 Requisitos não funcionais

- Interface gráfica: O sistema deve ser simples e objetivo para o cliente.
- Base de dados: Deve ser utilizado MongoDB.
- Metodologia: Deve-se utilizar uma metodologia ágil para o desenvolvimento da aplicação web.
- Plataforma: O sistema deve ser Web.
- Linguagem de programação: Deve ser utilizado a linguagem JavaScript para o desenvolvimento do front-end e Python para o back-end.

4.2 Modelação de Dados

Posteriormente ao levantamento de requisitos do sistema, iniciou-se a construção daquilo que seria a modelação de dados. Sendo o MongoDB a base de dados escolhida para o desenvolvimento da aplicação, foi então desenvolvida a modelação numa solução que possibilita posteriormente o deploy do script do modelo da base de dados para o MongoDB, criado assim as coleções e as devidas propriedades. Tendo isto em conta foi desenvolvido a seguinte modelação:

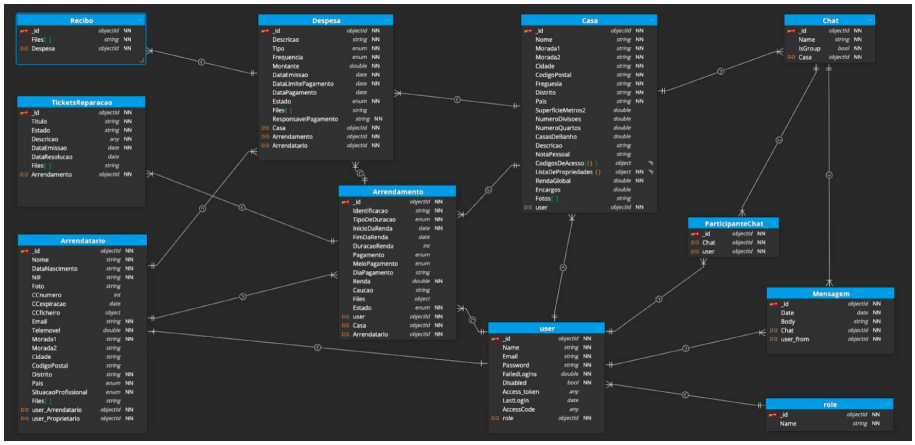

Figura 4.2: Modelação de dados.

Para a construção desta modelação foram identificadas primeiramente as entidades envolvidas nos requisitos da aplicação. Com isto foi possível identificar as seguintes:

- Casa - Existe a necessidades de guardar todos os dados respectivos das casas dos proprietários. Nesta entidade temos referência para o id do utilizador, sendo esta apenas para o utilizador com role de proprietário. Esta contém informações sobre:
  - Lista de Propriedades;
  - Códigos de Acesso;
- Utilizador - Esta entidade é essencial de forma a identificar as propriedades do utilizador, quer seja o proprietário ou o arrendatário, de forma a este poder identificar-se (login) na plataforma e a interagir com a mesma.
- Role - Esta entidade é necessária de forma a identificar quais as funcionalidades que um certo utilizador pode aceder ou executar. Este poderá ser um admin, Proprietário ou Arrendatário.
- Arrendatário - Esta entidade contém as informações de um arrendatário que foi previamente criado por um proprietário. Situação Profissional: Estudante / trabalhador / trabalhador-estudante / outro

- Arrendamento - Com a necessidade de haver uma gestão de arrendamentos e gestão de contratos, esta entidade tem como objetivo colmatar a necessidade de agregar essa informação.
  - Estado: Ativo / Inativo
  - Tipo de duração: Definida / Indefinida
  - Pagamento: Mensal / Bimensal / Trimestral / Semestral / Anual / Único
  - Meio de pagamento: Cartão de Crédito / Dinheiro / Transferência Bancária / MBWay / Débito Direto
- Tickets de Reparação - Esta agrega todas as informações acerca de um pedido de reparação por parte de um arrendamento
- Despesa - Esta agrega toda a informação relativa a despesas associadas a um arrendamento.
  - Tipo: Renda / Luz / Água /Gás / Caução / Outro
  - Frequência: Única / Mensalmente / Trimestralmente / Semestralmente / Anualmente
  - Estado: Pago / Aguarda Pagamento / Atrasado
- Recibo - Esta tem como objetivo guardar os recibos associados a uma despesa.
- Chat - Com a necessidade de comunicação dentro da aplicação, esta entidade tem como objetivo identificar vários chats existentes dentro da aplicação, estando esta associada diretamente a uma casa.
- ParticipanteChat - Tem como objetivo agregar os participantes de um determinado Chat
- Mensagem - Esta entidade tem como função guardar as mensagens enviadas por um certo participante para um certo chat.

4.3 Arquitetura do Design de Desenvolvimento

Antes do processo de desenvolvimento da API, é importante que esta seja pensada, organizada e bem construída desde início, de forma a proporcionar uma melhor leitura e manutenção e escalabilidade do código [79]. De forma a ir de encontro a uma solução que correspondesse a estes parâmetros, foi desenvolvida a seguinte estrutura demonstrada na figura:

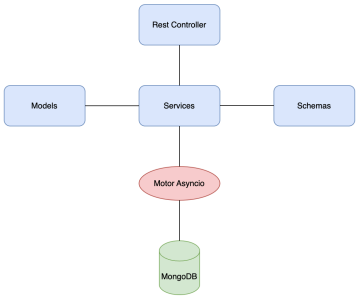

Figura 4.3: Estrutura da API.

Esta estrutura foi desenvolvida tendo em conta várias camadas. Cada uma destas camadas tem a sua função:

- Rest Controller: A sua função é receber os pedidos tendo em conta as rotas definidas no mesmo, fazer uma comunicação com os serviços necessários e posteriormente receber informação dos serviços e enviar a resposta.
- Services: Esta camada é a responsável pela manipulação de dados e toda a lógica de negócio envolvida na aplicação. Esta será também a camada que terá um contacto direto com todas as outras camadas.
- Models: A camada dos Models tem como objetivo estruturar as entidades usadas e as suas propriedades na aplicação, de forma a poder instanciar objetos em toda a aplicação.
- Schemas: A função dos Schemas é definir uma estrutura de dados, de forma a ser possível definir qual será a estrutura de uma resposta ou de um pedido.

- Motor Asyncio e MongoDB: O motor asyncio será responsável pelos pedidos assíncronos ao MongoDB, sendo este último o responsável pela alocação de dados da aplicação.

4.4 Repositório e Versionamento

Para o desenvolvimento de uma aplicação de software, é extremamente necessário nos dias de hoje que haja uma plataforma que possibilite a alocação dos ficheiros do mesmo, fazendo uma gestão de versão para que o seu desenvolvimento seja mais facilitado, principalmente no desenvolvimento em grupo.

São várias as plataformas que oferecem o serviço de alocação do repositório remoto, como GitHub, GitLab, Bitbucket, entre outros. Estes tem uma relação com o git, que é um gerenciador de versionamento, fazendo com que todo o código seja possível de ser gerido ao possuir um histórico de mudanças nos ficheiros.

Tendo isto em conta, foi necessário escolher qual dos repositórios faria sentido servir de repositório remoto para o projeto. O escolhido foi o github devido à sua fama e grande comunidade, bem como as suas funcionalidades e integração com o git.

Figura 4.4: Logo do GitHub.

4.5 Metodologia do Desenvolvimento

As metodologia Agile são abordagens de desenvolvimento flexíveis e de fácil adaptação em situações em que há frequentes mudanças nas necessidades, permitindo assim que resultados sejam mais rapidamente fornecidos aos clientes. Estas abordagens preconizam a divisão do desenvolvimento em ciclos breves ou iterações que duram apenas algumas semanas, tendo estas a nomenclatura de sprints, permitindo que os clientes recebam, ao final de cada ciclo, versões que agreguem valor às operações da empresa [80]. Os programadores podem assim adaptar-se às mudanças nos requisitos no início de cada ciclo, para além de receberem feedback contínuo dos clientes, o que reduz os riscos do projeto. Na seguinte figura podemos ver uma ilustração daquilo que é na prática uma metodologia Scrum Agile.

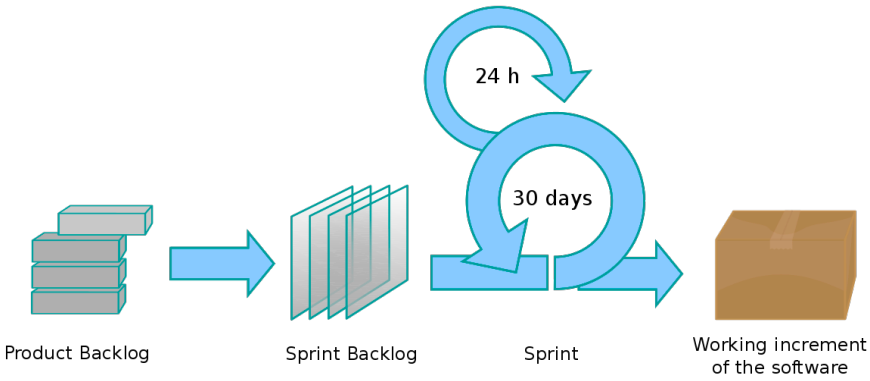

Figura 4.5: Metodologica Scrum.

De forma a organizar o desenvolvimento do projeto foi incorporada então esta metodologia ágil no início do projeto. Esta metodologia seguia os padrões daquilo que era o Scrum, tendo como planeamento o espaço de tempo de Sprints de 2 semanas, tendo ao final dessa Sprint uma reunião de forma a rever o trabalho feito e a levantar dificuldades ou bloqueios que foram encontradas. Estes planeamentos de sprints foram desenvolvidos numa plataforma que potencia este tipo de metodologias. Esta aplicação chama-se Notion, e para além de ser um programa de notas, tem também muitas outras utilidades

como a criação de tabelas, visualizações de gráficos e imensos templates com várias funcionalidades, incluindo Scrum, facilitando bastante o seu desenvolvimento [81]. De forma a incorporar esta metodologia agile no Notion, começou-se por anotar todas as User Stories num backlog. Após isso para a criação das sprints, eram criadas tarefas associadas às User Stories a que pertenciam. De seguida era feita uma avaliação do tempo que iria demorar a tarefa e com isso adicionava-se posteriormente à sprint em questão.

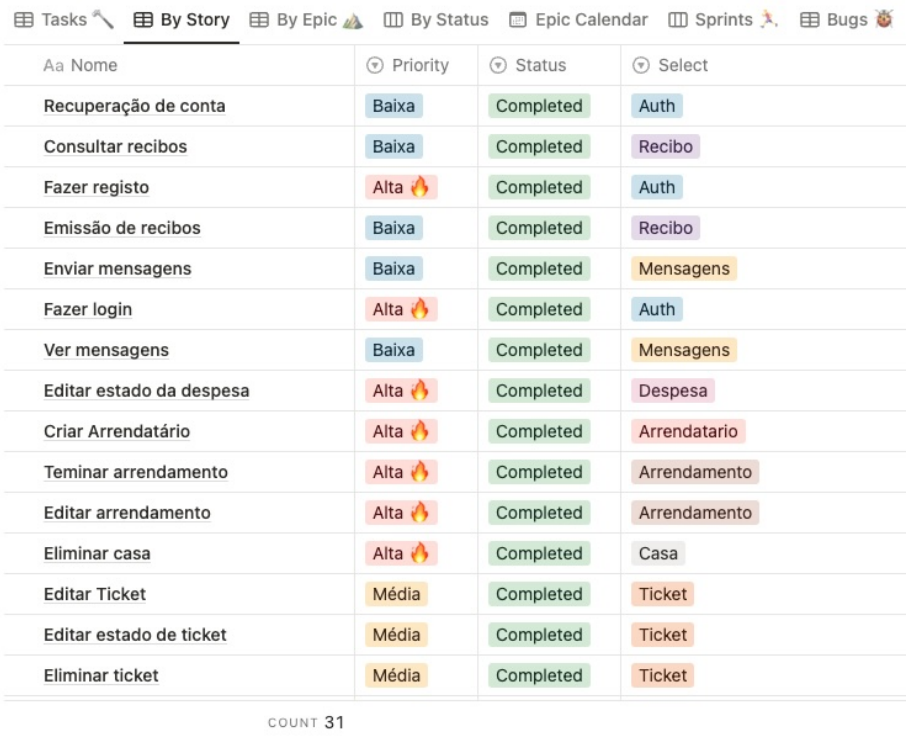

Figura 4.6: User Stories

Para cada sprint as tarefas possuem 3 tipos de status de forma a organizar as tarefas que ainda não tinham sido abordadas na sprint, as que estavam em curso e as que já estavam finalizadas.

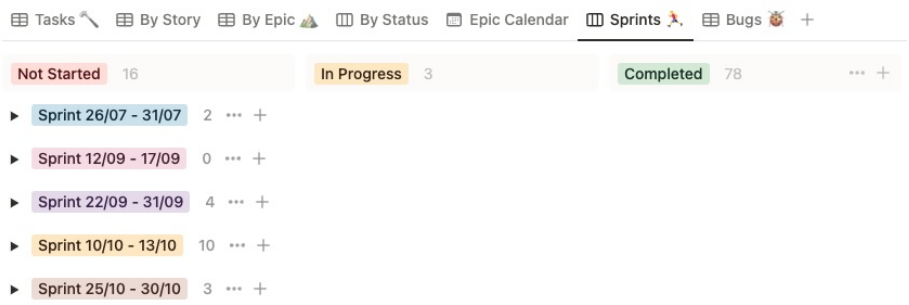

Figura 4.7: Sprints

4.6 Mockups

Numa aplicação Web, uma das partes mais fulcrais é a interface com que o utilizador irá interagir (UI), tendo esta um impacto direto na qualidade de apresentação e adoção de uma aplicação por parte de um utilizador. No desenvolvimento de uma aplicação, é essencial antes de passar para a implementação da solução, traçar qual será o aspeto visual da aplicação de forma também a identificar eventuais problemas, que de outra forma, poderiam ser apenas identificados já no desenvolvimento da aplicação. Inicialmente foi necessário decidir entre as vastas aplicações presentes no mercado, qual é que seria a mais indicada para a implementação dos mockups. O figma, uma aplicação web que possibilita a criação de interfaces gráficas online, foi a escolhida para esta tarefa. Este dispõe de uma plataforma simples e prática para o desenvolvimento das mesmas. Esta foi a tecnologia escolhida para o desenvolvimento do esboço de diversos ecrãs. Um dos primeiros ecrãs a serem desenhados no figma foi o ecrã onde o proprietário poderá visualizar todas as propriedades que já introduziu na plataforma, como se pode ver na seguinte figura:

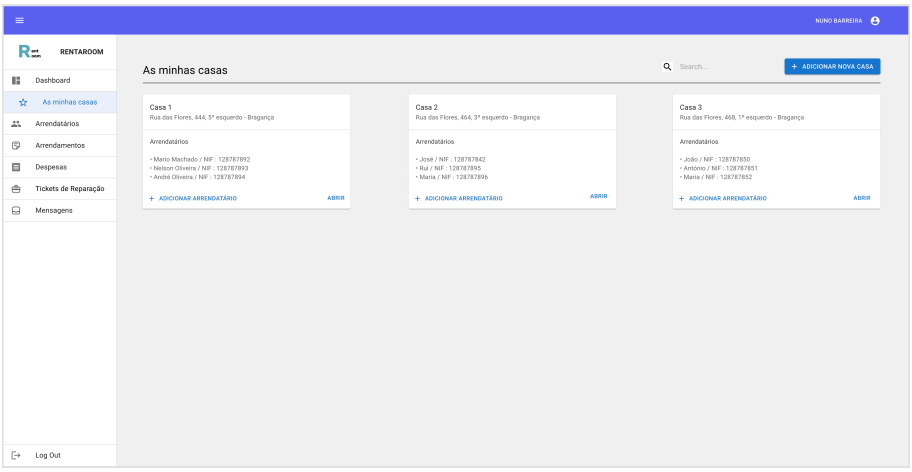

Figura 4.8: Mockup do ecrã de visualizar todas propriedades

Neste mockup foi também possível projetar como seria o aspeto geral da aplicação, como a palette de cores da aplicação e a definição de como seria o aspeto visual da navegação entre páginas. Os tons de azul, cinza e branco constituem o conjunto de cores escolhidas para satisfazer a pallete de cores da aplicação. Outro ecrã relevante é um exemplo daquilo que serão os ecrãs de visualização das várias despesas, arrendamentos, arrendatários e tickets de reparação. Como em todas esta páginas o principal objetivo é a visualização de todas os resultados de cada uma destas entidades, então todos estes ecrãs serão muito idênticos. O exemplo do esboço destes ecrãs está presente na figura (4.9), onde é possível visualizar o ecrã onde o proprietário consegue visualizar todos os arrendatários que já adicionou à plataforma.

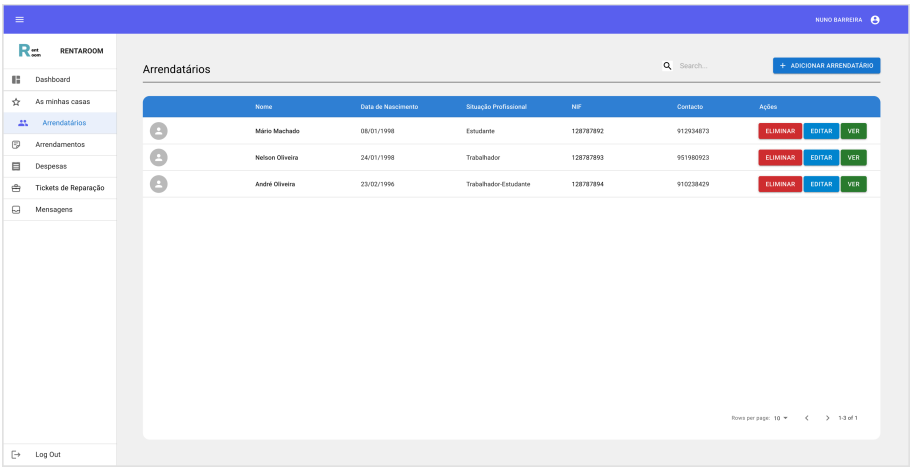

Figura 4.9: Visualização de todos os arrendatários.

Neste esboço o proprietário consegue visualizar algumas informações acerca do arrendatário, bem como realizar várias ações sobre estes arrendatários, como por exemplo eliminar, editar as informações do mesmo ou até ver todas as informações do arrendatário. É possível também visualizar que foi esboçado uma paginação e uma barra de busca de forma a poder otimizar a visualização de dados na aplicação e filtrar os resultados. Foi também adicionado um botão com o objetivo do proprietário poder adicionar novos arrendatários. Um ecrã que foi também esboçado foi o de criação de um arrendatário (Figura 4.10), que se assemelha aos outros ecrãs de criação de outras entidades também.

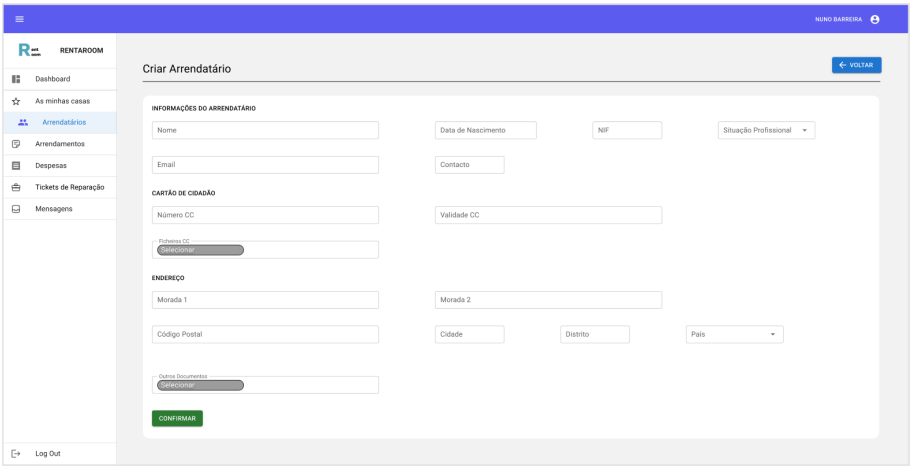

Figura 4.10: Criação de um arrendatário.

Neste ecrã o objetivo é o utilizador poder preencher todos os campos tendo em conta as propriedades da entidade arrendatário, de forma a conseguir criar um arrendatário. Foi possível neste ecrã definir a forma como iriam aparentar os vários tipos de inputs e a forma como os formulários de criação da aplicação iriam ser construídos. Por fim, outro ecrã relevante é o de visualização de um arrendatário já criado. A visualização deste ecrã é idêntica ao de criação, no entanto é possível visualizar os documentos adicionados, bem como todos os campos preenchidos com informação.

Capítulo 5

Desenvolvimento

Tendo todos os requisitos sido levantados, definidos e organizados, iniciou-se a etapa seguinte de implementação da aplicação. Neste capítulo será abordada toda a parte de desenvolvimento de software da aplicação de gestão de arrendamentos, tendo por objetivo descrever os passos necessários para a conclusão da solução final.

Para que o desenvolvimento desta aplicação fosse possível, seria necessário escolher um IDE (Integrated Development Environment) de modo a simplificar e aumentar a produtividade do programador. Estas interfaces permitem aos programadores editar o código fonte de uma aplicação, dar build a soluções, fazer debug à aplicação e instalar extensões que irão colmatar necessidades que o programador tenha, tornando assim o seu trabalho mais produtivo [82]. A IDE escolhida para o desenvolvimento deste projeto foi o VSCode devido ao seu suporte a várias linguagens, ter uma grande comunidade, personalização e ferramentas [83]. A escolha desta ferramenta teve em consideração as tecnologias que iam ser usadas para a implementação, que seriam o Python e o React, tendo este IDE suporte e algumas ferramentas de integração para estas duas tecnologias [84].

5.1 Desenvolvimento do Back-End

O passo seguinte passou pela criação das API Rest com a tecnologia Fast API. Para isso foi necessária a instalação do python. A versão instalada do python foi a 3.10, sendo esta no começo do projeto, a versão mais recente. Atualmente a versão mais recente é a 3.12 estando esta ainda em pre-release [85]. No entanto, para a utilização do FastApi apenas seria necessária uma versão acima do python 3.6 [86].

5.1.1 Setup

A este ponto, de forma a iniciar o desenvolvimento da parte de back-end, era necessário a criação da solução base. O FastApi já disponibiliza uma solução que serve como um esboço para o desenvolvimento. Foi necessário correr então o seguinte comando:

```
pip install "fastapi[all]"
```

Com a execução deste comando, são instaladas as dependências opcionais e funcionalidades necessárias para uma solução inicial da API [87]. De entre todas as dependências que irão ser instaladas com este comando, é de destacar a presença do uvicorn que irá servir como um servidor local (Asynchronous Server Gateway Interface (ASGI)), possibilitando desta forma que os pedidos aos endpoints definidos na API sejam escutados [88]. Concluído este processo, é apenas necessário executar o comando:

```
uvicorn main:app --reload
```

Depois de executado, a API estará já pronta para escutar pedidos. No comando, o argumento que lhe é passado - "–reload" - possibilita ao utilizador que o código seja re-compilado a cada mudança guardada que haja na solução.

5.1.2 Lista de EndPoints

Para os pontos de acesso da Rest Api foi criada uma estrutura dos URL de acesso, de forma a facilitar a compreensão do real objetivo do pedido. Com isto definiu-se que os

URL de acesso começavam pela entidade da funcionalidade que se queria desempenhar com a chamada do pedido. Após o separador, o URL teria também a ação que seria suposto o pedido desempenhar, relativamente à Entidade anteriormente definida, seguida pelo parâmetro da identificação da identidade em questão. Resumidamente, a estrutura do URL pode-se definir pela seguinte expressão:

```
/Entidade/Ação/{Parametro}
```

Já tendo a estrutura dos URL's dos endpoints definida, passou a ser implementada toda a parte de back-end. Finalizada a implementação da aplicação, a representação final de todos os web services pode ser apresentada da seguinte forma:

Utilizador

GET	user/getAll	Obter todos os utilizadores
GET	user/getUserById/{id}	Obter um utilizador através do Id do mesmo
GET	user/getUserByEmail/{email}	Obter um utilizador através do email do mesmo
POST	user/confirmAccessCode	Confirmar que o código que é passado como parâmetro corresponde ao utilizador do email que é passado como parâmentro.
POST	user/createUser	Criar um novo utilizador
PUT	user/updateUser	Atualizar propriedades do utilizador
PUT	user/updateUserPassword	Atualizar palavra-passe do utilizador
DELETE	user/delete/{id}	Eliminar um utilizador através do Id do mesmo

Role

GET	role/getAll	Obter todos os roles
GET	role/getRoleById{id}	Obter um role através do Id do mesmo
GET	role/getRoleByName/{name}	Obter um role através do nome do mesmo
POST	role/createRole	Criar um novo role
PUT	role/updateRole/{id}	Atualizar as propriedades de um novo role
DELETE	role/delete/{id}	Eliminar um role através do Id do mesmo

Arrendatário

GET	arrendatario/getAll	Obter todos os arrendatários
GET	arrendatario/getArrendatarioById/{id}	Obter arrendatário através do Id do mesmo
GET	arrendatario/getArrendatarioByName/{name}	Obter arrendatário através do nome do mesmo
POST	arrendatario/createArrendatario	Criar um arrendatário
PUT	arrendatario/updateArrendatario	Atualizar as informações de um arrendatário
DELETE	arrendatario/delete/{id}	Eliminar um arrendatário através do Id do mesmo
GET	arrendatario/getArrendatarioByCasa/{casaId}	Obter todos os arrendatários de uma casa
GET	arrendatario/getArrendatarioByProprietario/{proprietarioId}	Obter todos os arrendatários associados a um proprietário
GET	arrendatario/getArrendatarioByUser/{userId}	Obter o arrendatário associado a um utilizador
GET	arrendatario/getArrendatarioName/{id}	Obter o nome do arrendatário através do seu Id
GET	arrendatario/getArrendatariosSemArrendamento	Obter os arrendatários que não estão associados a um arrendamento
GET	arrendatario/getArrendatariosRelations/{userId}	Obter as relações com outras entidades de um certo utilizador
GET	arrendatario/getArrendatariosByCasaSimpleProperties/{casaId}	Obter as principais propriedades dos arrendatários pertencentes a uma casa

Arrendamento

GET	arrendamento/getAll	Obter todos os arrendamentos
GET	arrendamento/getArrendamentoById/{id}	Obter arrendamento através do Id do mesmo
GET	arrendamento/getArrendamentoByName/{name}	Obter arrendamento através do nome do mesmo
POST	arrendamento/createArrendamento	Criar um novo arrendamento
PUT	arrendamento/updateArrendamento	Atualizar as propriedades de uma arrendamento
PUT	arrendamento/terminarArrendamento/{arrendamentoId}	Terminar um arrendamento ao mudar o estado do mesmo
DELETE	arrendamento/delete/{id}	Eliminar um arrendamento através do Id do mesmo
GET	arrendamento/getArrendamentoByArrendatario/ {id}	Obter o arrendamento de um arrendatário
GET	arrendamento/getArrendamentosByProprietario/ {proprietarioId}	Obter todos os arrendamentos associados a um proprietário
GET	arrendamento/getArrendamentosByArrendatario/ {arrendatarioId}	Obter todos os arrendamentos associados a um arrendatário
GET	arrendamento/getArrendatarioByArrendamento/ {arrendamentoId}	Obter o arrendatário de um certo arrendamento
GET	arrendamento/getArrendamentosByCasa/casaId	Obter todos os arrendamentos associados a uma certa casa
GET	arrendamento/getArrendamentosByCasaSimpleProperties/ {casaId}	Obter as principais propriedades dos arrendamentos pertencentes a uma casa

Casa

GET	casa/getAll	Obter todas as casa
GET	casa/getCasaById/{id}	Obter casa através do Id da mesma
GET	casa/getCasasByProprietario/{userId}	Obter todas as casas que estão associadas a um proprietário
PUT	casa/updateCasa	Atualizar as propriedades de uma casa
DELETE	casa/delete/{id}	Eliminar um casa através do Id da mesma
POST	casa/createCasa	Criar uma nova casa
POST	casa/createCodigosAcesso	Adicionar códigos de acesso a uma casa
POST	casa/deleteCodigosAcesso	Eliminar códigos de acesso a uma casa

Despesa

GET	despesa/getAll	Obter todas as despesas
GET	despesa/getDespesaById/{id}	Obter despesa através do Id da mesma
GET	despesa/getDespesasByCasa/{casaId}	Obter todas as despesas associadas a uma casa
GET	despesa/getDespesasByCasaByArrendatario/{arrendatarioId}	Obter todas as despesas associadas a uma arrendatário
GET	despesa/getDespesasByCasaByProprietario/{proprietarioId}	Obter todas as despesas associadas a um proprietário
PUT	despesa/updateDespesa	Atualizar as propriedades de uma despesa
DELETE	despesa/delete/{id}	Eliminar uma despesa
POST	despesa/createDespesa	Criar uma despesa
POST	despesa/confirmarDespesa	Confirmar uma despesa

Mensagem

GET	mensagem/getAll	Obter todas as mensagens
GET	mensagem/getMensagemById/{id}	Obter mensagem através do Id da mesma
GET	mensagem/getAllMessagesFromChat/{chatId}	Obter todas as mensagens de um certo chat
PUT	mensagem/updateMensagem	Atualizar as informações de uma mensagem
DELETE	mensagem/delete/{id}	Eliminar uma mensagem
DELETE	mensagem/deleteMensagemFromCasa/{casaId}	Eliminar todas as mensagens do chat associado a uma casa
POST	mensagem/createMensagem	Criar uma nova mensagem

Recibo

GET	recibo/getAll	Obter todos os recibos
GET	recibo/getReciboById/{id}	Obter recibo através do seu Id
GET	recibo/getAllRecibosFromDespesa/{despesaId}	Obter todos os recibos de uma certa despesa
PUT	recibo/updateRecibo	Atualizar propriedades de um certo recibo
DELETE	recibo/delete/{id}	Eliminar um recibo

Tickets de Reparação

GET	ticket/getAll	Obter todos os tickets
GET	ticket/getTicketById/{id}	Obter ticket através do seu id
GET	ticket/getAllTicketsFromArrendatario/{arrendatarioId}	Obter todos os tickets associados a uma arrendatário
GET	ticket/getAllTicketsFromProprietário/{proprietarioId}	Obter todos os tickets associados a uma proprietário
PUT	ticket/updateTicket	Atualizar propriedades de um ticket
DELETE	ticket/delete/{id}	Eliminar um ticket através do Id
POST	ticket/createTicket	Criar um novo ticket

Chat

GET	chat/getAll	Obter todos os chats
GET	chat/getChatById/{id}	Obter chat através do seu id
GET	chat/getAllChatsFromUser/{userId}	Obter todos os chats associados a um utilizador
PUT	chat/updateChat	Atualizar propriedades de um chat
DELETE	chat/delete/{id}	Eliminar um chat através do Id
DELETE	chat/deleteFromCasa	Eliminar chat associado a uma casa
POST	chat/createChat	Criar um novo chat

Participante Chat

GET	participanteChat/getAll	Obter todos os participantes de chats
GET	participanteChat/getParticipanteChatById/{id}	Obter participante de um chat através do seu id
PUT	participanteChat/participanteChat	Atualizar propriedades de um participante de um chat
DELETE	participanteChat/delete/{id}	Eliminar participante de um chat através do Id
DELETE	participanteChat/deleteFromCasa	Eliminar participante de um chat associado a uma casa
POST	participanteChat/createparticipanteChat	Eliminar participante de um chat associado a uma casa

Autenticação

GET	autenticacao/verificarToken	Fazer a verificação de que um token de um certo utilizador se encontra válido
GET	autenticacao/utilizadorLogado	Verificar qual o utilizador logado
POST	autenticacao/login	Gera um token de autenticação que é guardado nas propriedades do utilizador e retornado na resposta
POST	autenticacao/recuperarPassword	Gera uma nova password para um certo utilizador e envia email para o mesmo

Durante o desenvolvimento foi possível observar estes endpoints numa interface gráfica chamada swagger como se pode verificar por um excerto destes serviços na figura 5.1.

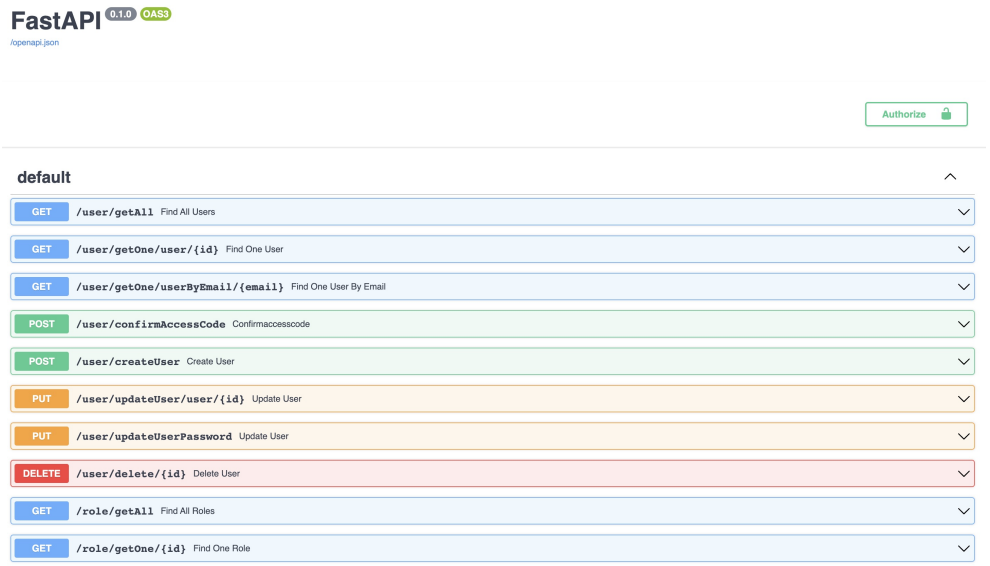

Figura 5.1: Representação dos endpoints na ferramenta Swagger.

Este programa é um open source que disponibiliza ao programador, uma vasta quantidade de ferramentas, de forma a este poder testar, e aceder às especificações de cada serviço da Rest API. Para além da ferramenta disponibilizada através da interface gráfica, o Swagger possui também outras funcionalidades como geração de testes, documentação automática e geração de código [89].

5.1.3 Paginação

A paginação é um aspeto importante numa aplicação, de forma a proporcionar uma melhor performance. Esta melhoria de performance deve-se ao facto de limitar o número de dados obtidos, em vez de obter todos os valores de uma só vez. Desta forma, os dados mostrados nas tabelas da aplicação são feitos à medida que são necessários, e não todos de uma só vez, prevenindo desta forma que o pedido demore demasiado tempo e evitando informação a mais que possivelmente será desnecessária, tornando a paginação numa funcionalidade essencial [90], [91].

Tendo estas melhorias na aplicação em consideração, foi implementado um middleware na API com o objetivo de promover uma paginação modular, de forma a que todos os pedidos pudessem de maneira genérica ter uma estrutura que consiga lidar com a paginação pedida. Este middleware irá fazer o seu trabalho ao atuar quando um pedido HTTP é recebida pela aplicação, antes de chegar à rota específica. Este middleware quando recebe o pedido irá examinar os parâmetros do mesmo de forma a determinar as informações da paginação, como número da página atual e quantidade de itens por página. Após lidar com esses dados, o middleware chama a rota principal que lida com a solicitação específica. Tendo em conta o lado do front-end, este envia informação relativamente à paginação da seguinte maneira:

```
http://exemplo.com/api/dados?skip=${skip}&limit=${limit}&filtro=${filtro}
```

No pedido é enviada a informação sobre o número de itens que se quer obter através do "limit" que é enviado, o "skip" que diz respeito ao número de itens que queremos ignorar no serviço. Este "skip" é calculado através do número da página atual em que o utilizador se encontra e o número de itens a mostrar na tabela. E por fim é enviado no pedido os filtros pelo qual o pedido irá filtrar nos resultados a enviar.

No lado do pedido, este obtém o "limit", o "skip" e os filtros e aplica na chamada da query ao mongoDB através dos métodos presentes no motor, de "skip()" que vai saltar o número de itens a ter em conta, "limit()" que irá limitar o número de itens a retornar e o "find(filtros)" que irá filtrar os resultados tendo em conta os filtros aplicados. Para além

disto, é enviado no corpo da resposta, juntamente com os resultados obtidos, o número de resultados totais sem a paginação.

5.1.4 Desenvolvimento da Autenticação

Com o intuito de possibilitar a autenticação dos utilizadores na aplicação, foi necessário criar um sistema que fizesse a criação da autenticação e validação da mesma. Para que esta autenticação de um utilizador fosse possível foram implementados três serviços.

O primeiro serviço tem como objetivo fazer a criação do utilizador, enquadrando-se este, no caso de uso do utilizador fazer o registo na aplicação. Para este serviço foi usada a biblioteca "passlib" de forma a dar hash à password introduzida pelo utilizador, com o objetivo de guardar de forma segura a password do utilizador na base de dados.

O segundo terá a função de validar as credenciais do utilizador e caso estas estejam corretas, criar um token JWT de forma a que este seja enviado como resposta do serviço para o front-end, sendo este de seguida alocado na cache do browser. Este serviço é executado caso o utilizador tente efetuar o login na aplicação, não tendo um JWT Token válido na cache do browser. De maneira a ir de encontro à solução pretendida foi criado este serviço que recebe o email e password do utilizador. De seguida este serviço valida se o utilizador está ativo e se o número de tentativas de login falhados é menor que três. A seguir a estas validações, caso se verifiquem válidas, é chamada uma função, criada com o intuito de fazer a validação das credenciais do utilizador. Esta função irá pegar no email introduzido para fazer o login e verificar se existe algum registo com esse email. De seguida, é usada a mesma biblioteca utilizada para fazer hash da password, com o objetivo desta vez de fazer a comparação da password introduzida pelo utilizador para fazer login, com a guardada na base de dados, estando esta última hashed por essa mesma biblioteca. Após esta validação se demonstrar correta, é gerado um JWT Token usando o algoritmo "HS256" e uma secret key presente no serviço, contendo este token informação da data de expiração do mesmo e o email e role do utilizador. De seguida, este token é guardado na base de dados, associado ao utilizador em questão e enviado como resposta

do serviço para o front-end, sendo este alocado na cache do browser.

O terceiro serviço tem como objetivo validar se o JWT Token de um utilizador encontra-se de acordo com o registado na base de dados. Este serviço foi implementado com a finalidade de colmatar as necessidades do caso de uso de o utilizador ter um token válido. Após o utilizador tentar aceder à aplicação, o sistema irá validar se o utilizador tem efetivamente um token guardado na cache do browser. Caso tenha, irá verificar a validade do mesmo. Se este se encontrar dentro do limite de expiração então é chamado este terceiro serviço. Este serviço vai-se encarregar de comparar o token guardado na cache do browser do utilizador com aquele que está associado ao seu email na base de dados, de forma a comprovar que este é aquele que foi originalmente criado pelo sistema. Ao comprovar que o mesmo é autêntico, o serviço enviará uma resposta válida, de forma a o utilizador poder aceder a todas as funcionalidades da aplicação.

5.1.5 Sistema de Envio de Emails

Uma das necessidades de desenvolvimento desta aplicação era o envio de emails. De forma a colmatar esta necessidade, foi criado um script em python, usando uma biblioteca entitulada de "smtplib", que possibilita o envio de emails através do protocolo SMTP (Simple Mail transfer Protocol). Este possibilita que emails possam ser enviados através de servidores que enviam as informações pretendidas para a rota solicitada com o conteúdo desejado [92]. Empresas como a Microsoft e Gmail possibilitam o uso destes servidores, tendo uma limitação de emails que podem ser enviados, na versão gratuita. O servidor da Microsoft Outlook foi usado para implementar esta funcionalidade. Com o servidor escolhido, foi necessário criar uma conta no Outlook que serviria como email de remetente. Posteriormente foi apenas necessário configurar os campos do servidor, email e password como parâmetros para a biblioteca "smtplib". Esta possibilita após esta configuração, o envio de emails, através da função "sendemail", passando como propriedades o corpo da mensagem, o email do remetente e o email do destinatário.

5.2 Desenvolvimento do Front-End

Neste capítulo será apresentado o aspeto final da parte gráfica da aplicação. Serão analisados em mais detalhe cada um dos ecrãs desenvolvidos, demonstrando a abordagem feita em cada um dos mesmos com as suas devidas ilustrações.

O layout abordado para a aplicação foi orientado tendo em conta o que foi previamente desenvolvido nos mockups já descritos no capítulo da Modelação (Cap 4.). Este layout teve como principal preocupação a facilidade de utilização da aplicação por parte do utilizador, usando navegações objetivas. O layout aplicado tem também em conta a preocupação com a utilização de cores neutras de forma a facilitar a visualização dos dados.

5.2.1 Ecrã de Login

Numa aplicação orientada a utilizadores, a imposição de uma página que possibilite ao utilizador fazer login, registar-se e recuperar a password é inquestionável. Para alcançar esse objetivo, foram criadas várias páginas para atender à necessidade de proporcionar ao utilizador essas funcionalidades.

Primeiramente foi criada a página de login que possibilita ao utilizador inserir o seu email e a password já registadas na aplicação de forma a aceder às outras funcionalidades da aplicação. Caso o utilizador introduza corretamente os seus dados, o sistema irá guardar no browser do utilizador um token JWT, contendo informações de login do utilizador, sendo este guardado também na base de dados. Este token será então usado para validar a cada mudança de rota dentro da aplicação, se o utilizador ainda poderá aceder à aplicação, validando a data de expiração do JWT token e comparando o JWT Token do browser do utilizador com o que está associado a esse utilizador na base de dados. Desta maneira é possível garantir que só um token gerado e com data válida seja usado para aceder à aplicação.

Na ação de login correto por parte do utilizador, o sistema irá obter informação do utilizador como o nome, token, email e role, e irá guardar estas informações num contexto. Este contexto no react possibilita que informações sejam guardadas e acedidas

globalmente. Desta forma, as informações do utilizador poderão ser obtidas em cada página da aplicação, possibilitando assim que seja feita a validação do que pode estar visível ou não para o utilizador em cada página, tendo em conta o seu role. Este contexto foi desenvolvido com um hook do react entitulado de "useContext". Após isto, o sistema irá reencaminhar o utilizador para a página inicial da aplicação (figura 5.8) onde poderá executar todas as suas funcionalidades. Na eventualidade de o utilizador fazer o logout da aplicação, o token guardado no browser é eliminado, deixando assim o utilizador de ter acesso às funcionalidades da aplicação, sendo diretamente reencaminhado para esta página de login. Este ecrã não só permite ao utilizador efetuar o login, mas também possibilita o registo do mesmo na aplicação, bem como a recuperação da palavra-passe caso seja esquecida. Este ecrã pode visualizar-se na figura 5.2.

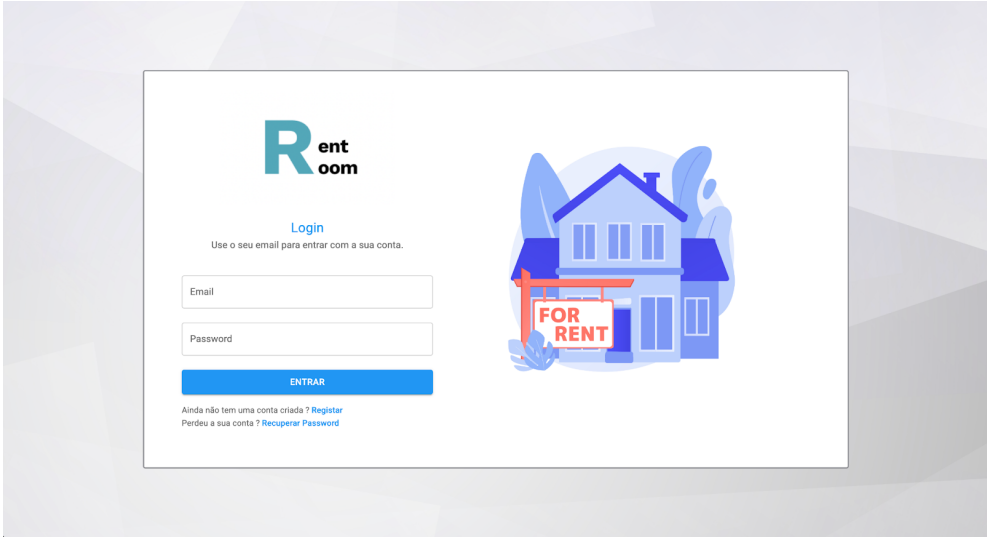

Figura 5.2: Ecrã de login.

Caso os dados inseridos pelo utilizador estejam incorretos, então o acesso à aplicação não será concedido. A informação de acesso negado é demonstrada através de um alerta por baixo do botão "Entrar", como se pode visualizar na figura 5.3.

Caso o utilizador introduza incorretamente a password por 3 vezes consecutivas, então o utilizador será bloqueado e a mensagem de erro será exatamente a informá-lo disso.

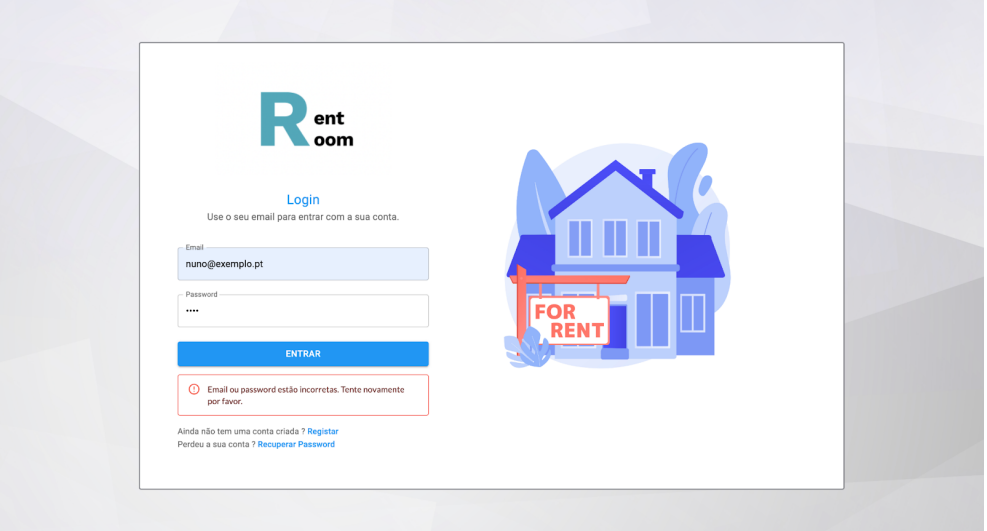

Figura 5.3: Erro nos dados introduzidos.

5.2.2 Ecrã de Recuperação da Password

O utilizador é reencaminhado para o ecrã de recuperação de password através da página de login, ao clicar no botão "Recuperar Password". Neste ecrã o utilizador poderá inserir o seu email. Ao fazê-lo, o sistema irá gerar uma nova palavra-passe e de seguida enviará um email para o endereço fornecido.

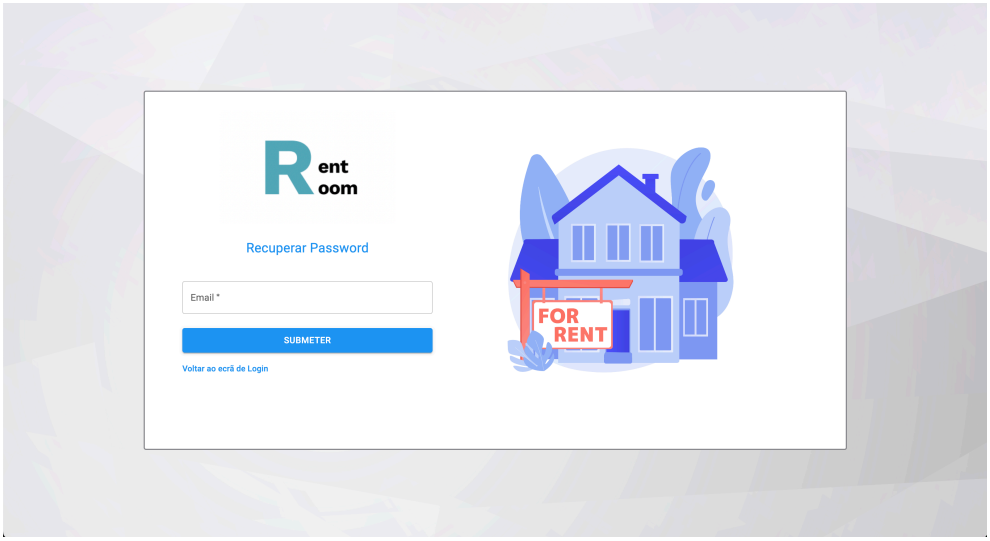

Figura 5.4: Ecrã de recuperação de password.

5.2.3 Ecrã de Registo

Para o caso de uso de o utilizador ainda não estar registado na plataforma e o queira fazer, foi criado um ecrã onde este poderá fazer o seu registo. Este ecrã pode ser acedido através do ecrã de login, ao pressionar o botão "Registar". O utilizador será reencaminhado para o ecrã (figura 5.5), onde é possível preencher os seus dados de forma a fazer esse registo.

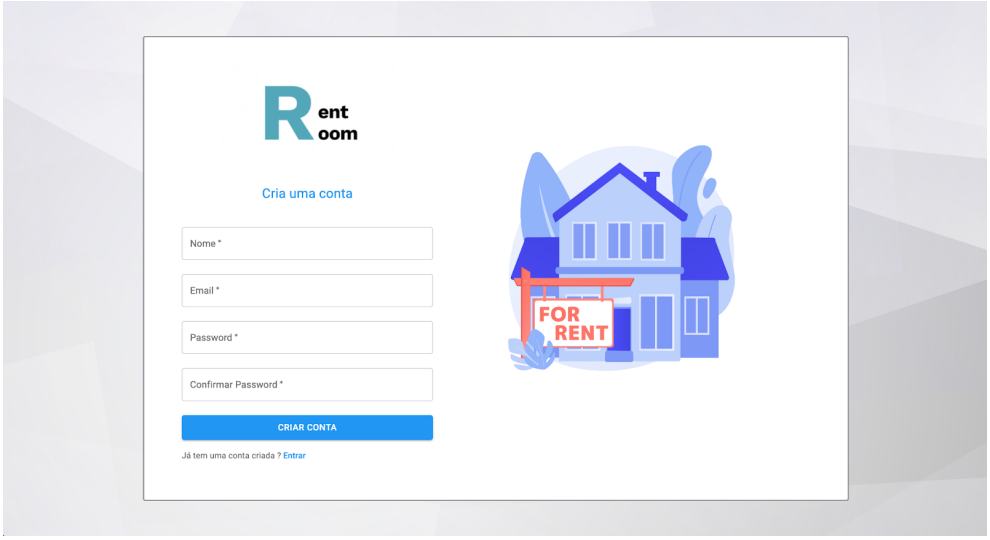

Figura 5.5: Ecrã de Registo.

Após o utilizador preencher os dados do formulário de registo e pressionar o botão "Criar Conta" presente no ecrã da figura 5.5, caso todos os dados sejam validados corretamente, aparecerá um alerta de sucesso, de forma a informar o utilizador que um email com um código de acesso foi enviado para o email fornecido, como é possível visualizar na figura 5.6.

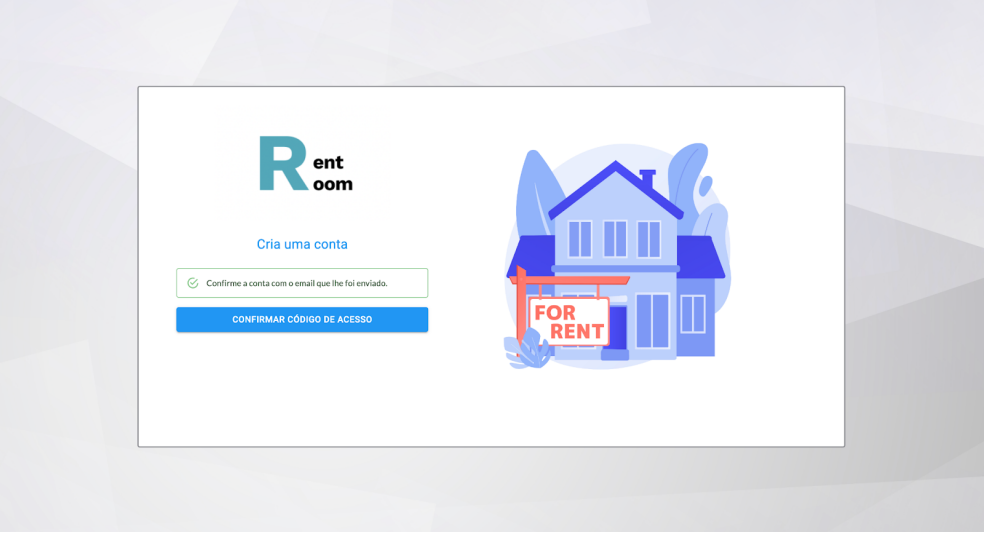

Figura 5.6: Confirmação de Email enviado.

Recebendo o código de acesso no email, o utilizador poderá confirmar o mesmo ao clicar no botão "Confirmar Código de Acesso" presente no ecrã da figura 5.6. Ao fazê-lo, o utilizador será reencaminhado para um ecrã onde poderá validar esse código de acesso (figura 5.7). Na eventualidade de o utilizador introduzir o email e código de acesso correspondente corretos, este será reencaminhado para a páginal inicial da aplicação (figura 5.8), tendo acesso a todas as funcionalidades presentes no seu role.

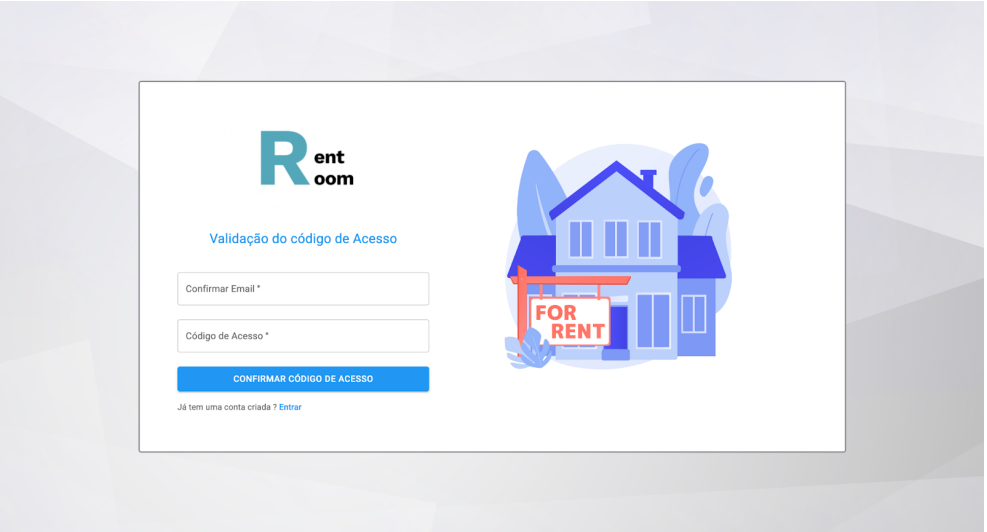

Figura 5.7: Validação do código de confirmação de email.

5.2.4 Página Inicial

Tendo o utilizador iniciado sessão, este irá ser primeiramente reencaminhado para a página inicial da aplicação. Esta página tem como principal função fornecer uma visão geral daquilo que poderá ter mais relevância para o utilizador, sendo ele proprietário ou arrendatário.

De forma a ir de encontro com as necessidades anteriores, foram criadas para esta página 3 divisões. A primeira, entitulada de "Acessos Rápidos", fornece ao utilizador a possibilidade de aceder a funcionalidades de maneira mais acessível e rápida, clicando apenas num botão para exercer essas funcionalidades. Desta forma o proprietário poderá adicionar uma casa, arrendatário, arrendamento ou despesa de forma muito mais rápida (figura 5.8). No caso do arrendatário, este terá apenas permissão para adicionar tickets de reparação através destes acessos rápidos. A segunda, corresponde aos tickets de reparação e o seu estado e a terceira às despesas associadas ao utilizador. Ambas as divisões foram criadas com a intenção de fornecer informação relevante de forma direta ao utilizador, obtendo este esta informação sem ter de aceder aos ecrãs respetivos aos tickets ou às despesas. Assim o utilizador poderá de forma rápida verificar se as últimas despesas ou tickets já mudaram de estado ou não.

Para além destas informações rápidas, a plataforma possui uma barra de navegação rápida, de forma a que seja possível o utilizador navegar para os vários ecrãs tendo em conta as funcionalidades que pretende executar. Nesta barra de navegação é possível navegar para a página "Perfil" onde o utilizador poderá mudar informações relativas ao seu perfil, mudar a palavra-passe, mudar de tema e mudar preferências de notificações. As outras navegações possibilitam ao utilizador executar funcionalidades relativas às propriedades que possui, arrendamentos, arrendatários, despesas, tickets e mensagens dentro da aplicação. Para além destas funcionalidades, no canto suporior direito o utilizador poderá terminar sessão ao pressionar o botão de "Logout". No canto superior esquerdo o utilizador poderá ainda pressionar o logo da aplicação intitulado de "RentaRoom", navegando sempre para a página inicial.

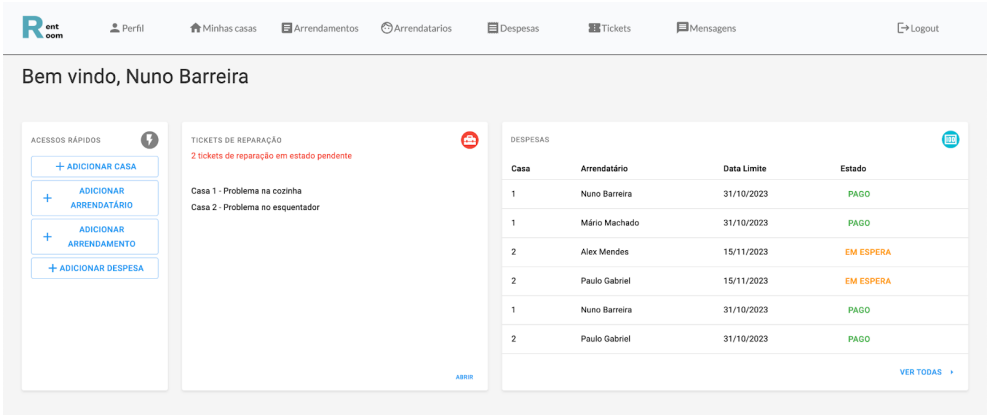

Figura 5.8: Página Inicial.

As páginas desta aplicação foram implementadas de forma a que fossem responsivas, podendo adaptar-se a vários tipos de largura e altura de diferentes tipos de ecrã como é possível ver na figura 5.9, onde todos os componentes do ecrã inicial, incluindo a barra de navegação, se adaptam à largura da janela do browser. Com esta funcionalidade, a interação da aplicação, tanto em dispositivos móveis como dispositivos como computadores, será otimizada, possibilitando ao utilizador uma melhor experiência, independentemente do seu dispositivo.

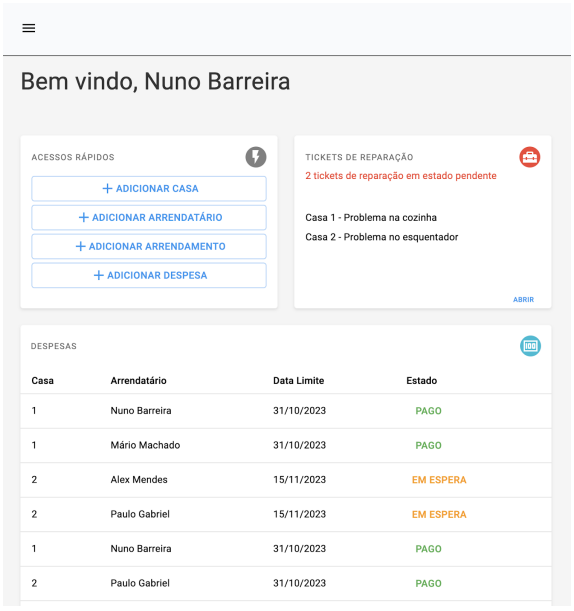

Figura 5.9: Dashboard com Navbar colapsada.

De forma a que esta responsividade fosse possível em ecrãs pequenos, foi necessário criar um comportamento diferente para a barra de navegação de forma a que esta se adapte a um ecrã com menor largura. Com o objetivo de ultrapassar este obstáculo foi necessário criar estilos visuais diferentes dependendo da largura do ecrã. Desta forma, caso a largura da janela do browser do utilizador possuísse mais de 1200px, a barra de navegação teria estilos aplicados de forma a que esta fosse mostrada horizontalmente como na figura 5.8. Caso contrário, a barra passaria para a lateral, podendo o utilizador colapsar a mesma como é possível verificar na figura 5.9, ou então expandi-la (figura 5.10), de forma a poder interagir com os componentes de navegação dela. Para além do ajustamento da barra de navegação, os componentes também se ajustam consoante a largura do ecrã. Para esta finalidade, os componentes da biblioteca de componentes "MUI" possuem uma propriedade que possibilita definir qual o comportamento dos mesmos e o espaço que estes deverão ocupar, tendo em conta o tamanho do ecrã.

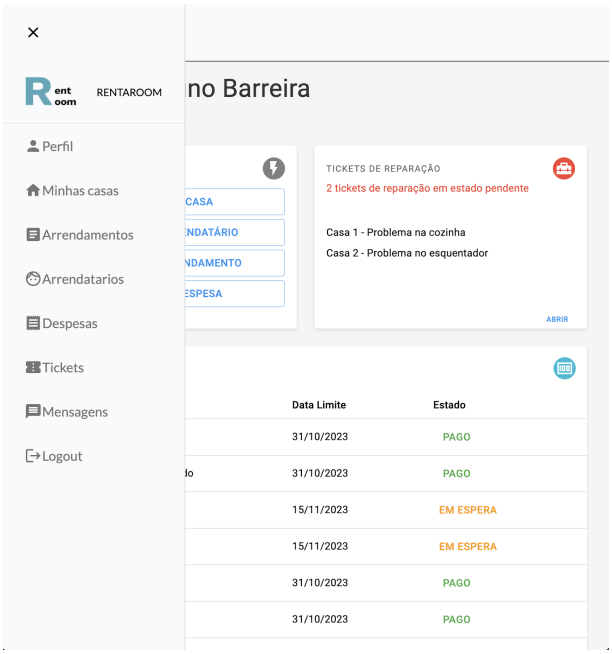

Figura 5.10: Dashboard com Navbar expandida.

5.2.5 Configurações do Utilizador

O ecrã de configurações do utilizador pode ser acedido através do botão "Perfil" da barra de navegação. Este ecrã está divido em 4 secções separadas. A primeira diz respeito às informações do utilizador, onde este poderá mudar a sua foto de perfil e visualizar as suas informações registadas na aplicação.

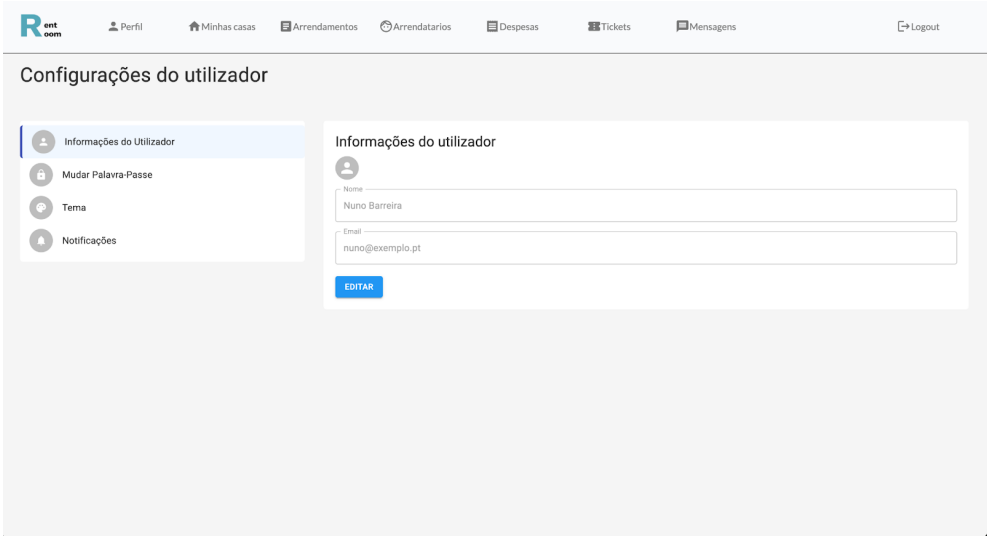

Figura 5.11: Informações do Proprietário.

Na segunda secção o utilizador tem a possibilidade de mudar a sua palavra-passe (figura 5.12). Para esta funcionalidade, este precisa de introduzir primeiramente a sua palavra-passe atual. De seguida necessita de introduzir duas vezes a nova palavra-passe de forma a garantir que as mesmas estejam corretamente definidas. De seguida o sistema verifica se as palavras passe novas são iguais. Caso isto se verifique, então o sistema irá validar se a palavra-passe atual introduzida corresponde à que está guardada na base de dados. Sendo isto verdadeiro, uma notificação aparecerá de forma a informar que a palavra-passe foi corretamente mudada.

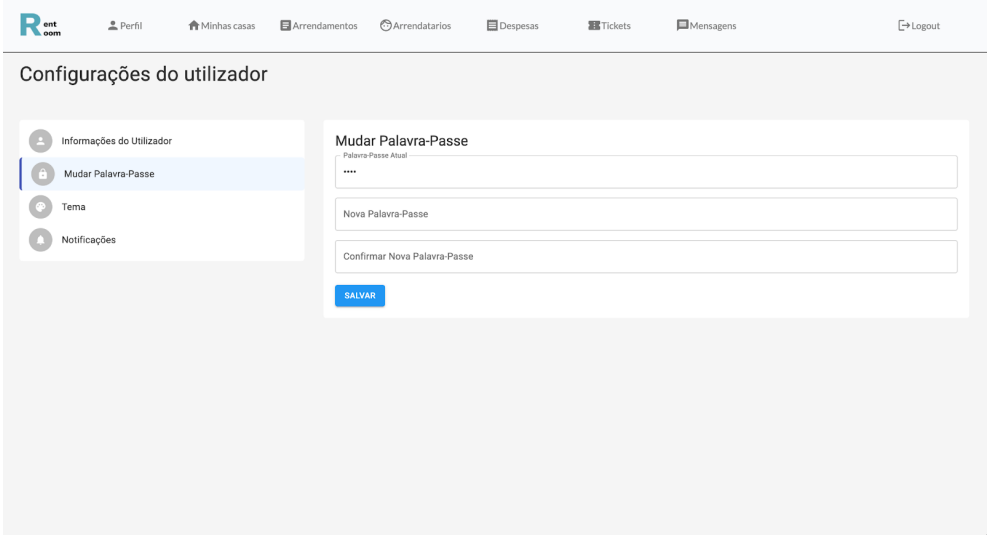

Figura 5.12: Mudar palavra-passe.

Na terceira secção da página de configurações do utilizador, é disponibilizada ao utilizador a possibilidade de mudança de tema (figura 5.13). Nesta secção é apenas apresentado um botão que proporciona o alternamento em tempo real do tema de toda a aplicação. Por defeito, a aplicação é apresentada com o tema claro, sendo esta apresentada pelo botão de alternamento na posição da esquerda. No entanto ao clicar no botão, este mover-se-á para a direita, habilitando o tema escuro.

A identificação do tema é também ela guardada num contexto. Desta forma é possível adequar todos os componentes de acordo com o tema escolhido pelo utilizador.

De forma a que o tema escolhido pelo utilizador seja refletida permanentemente para o dispositivo que este esteja a aceder, a informação sobre o tema escolhido é guardada na cache do browser. Desta forma, sempre que o utilizador acede à aplicação através desse mesmo browser, o tema escolhido será aplicado de forma imediata.

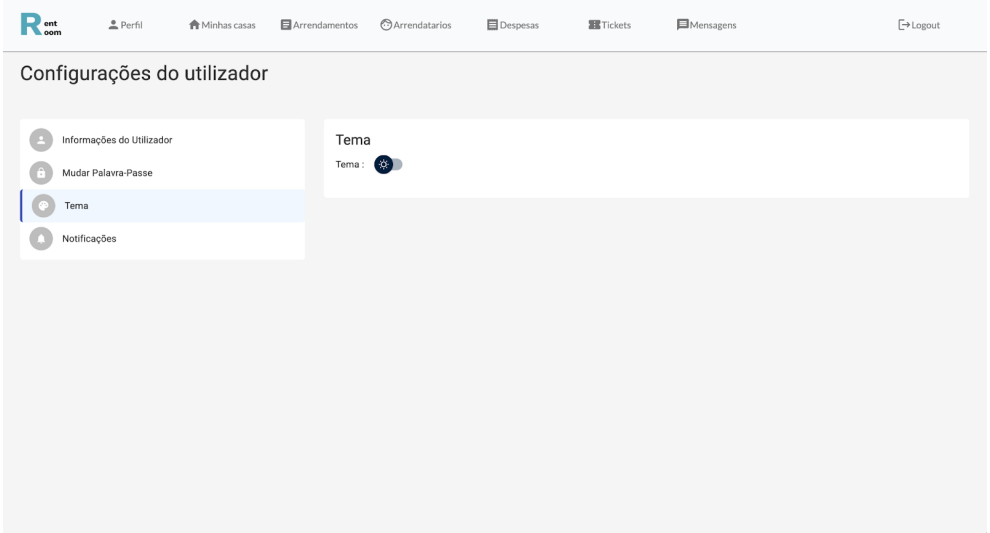

Figura 5.13: Tema.

Por fim, na última secção o utilizador poderá definir as suas preferências quanto às notificações da plataforma (figura 5.14). Este poderá definir se pretende que as notificações da aplicação e as que são enviadas por email sejam mostradas ou ocultadas. Estas notificações referem-se à mudança de estados de tickets ou despesas.

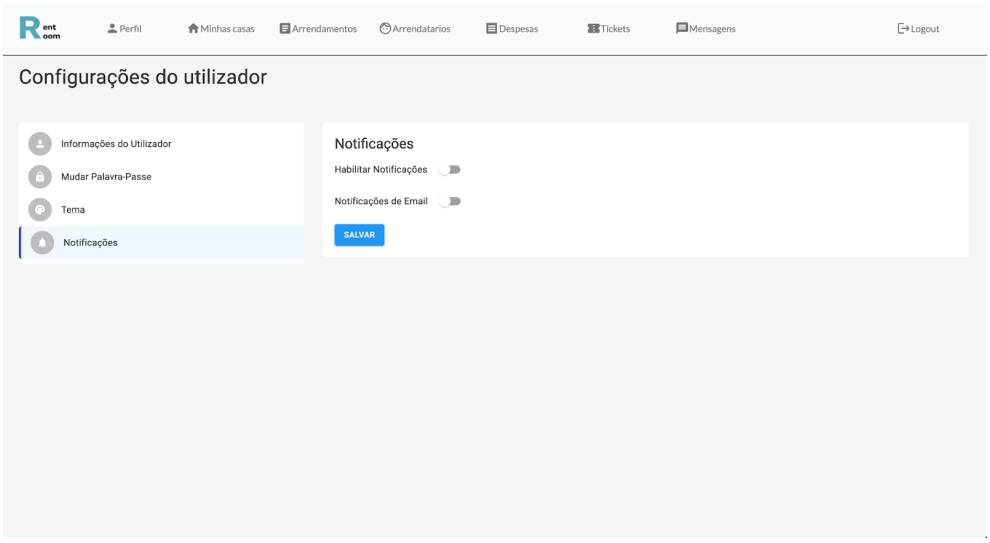

Figura 5.14: Configuração das notificações.

5.2.6 Consultar Propriedades

O utilizador, possuindo o role de proprietário, poderá aceder às funcionalidades associadas às suas propriedades através do botão "Minhas Casas", presente na barra de navegação. Será então reencaminhado para a página principal desta funcionalidade que é a da consulta das suas propriedades, onde visualizará todas as propriedades que já registou na aplicação.

Neste ecrã o utilizador poderá ainda desempenhar várias funções relativas às suas propriedades. A de consultar as despesas de uma certa casa é uma delas, sendo este reencaminhado para a página de consulta de despesas (figura 5.28), aplicando o filtro da casa selecionada, de forma a só mostrar resultados relativos às despesas da casa em questão. O proprietário poderá ainda adicionar um arrendatário a uma das suas casas. Selecionando esta funcionalidade, o proprietário será reencaminhado para o ecrã de criação de arrendamento (figura 5.26), onde o campo da casa estará pré-preenchido com a casa selecionada previamente. Da mesma forma se comporta a funcionalidade de adicionar despesa, reencaminhando o proprietário para o registo de despesa (figura 5.33), pré-selecionando a casa escolhida.

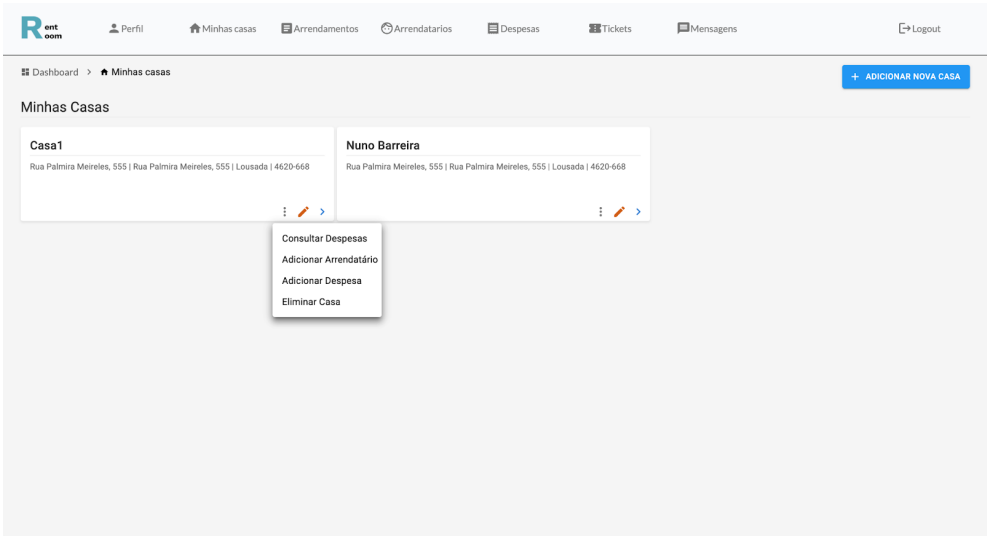

Figura 5.15: Consulta das propriedades do proprietário.

No ecrã de consulta de propriedades, o proprietário possui também a funcionalidade de poder visualizar os detalhes de uma propriedade ao pressionar o botão com uma seta azul, sendo este o que está mais à direita em cada uma das caixas associadas a cada propriedade (figura 5.15).

Clicando o proprietário no botão anteriormente referido, este será então reencaminhado para o ecrã que se poderá visualizar na figura 5.16. Neste ecrã é o proprietário que poderá obter algumas informações sobre a propriedade em questão, como por exemplo o nome, propriedades da casa, arrendatários associados à casa, descrição, renda e morada. Para além disto destaca-se também a visualização das fotos da propriedade na parte esquerda do ecrã, podendo o proprietário visualizar todas as fotos clicando nos botões para mover para a imagem seguinte ou para a anterior. Destaca-se também o componente que ocupa a maior parte do ecrã. Aqui o utilizador poderá consultar os seus ganhos e as suas perdas, estando estas diretamente ligadas com as despesas registadas. Este componente incorporado pela biblioteca "react-chartjs-2", dá a possibilidade de o propritário visualizar o fluxo de gastos e ganhos através de uma vista geral e de uma vista mais pormenorizada, dividida por meses.

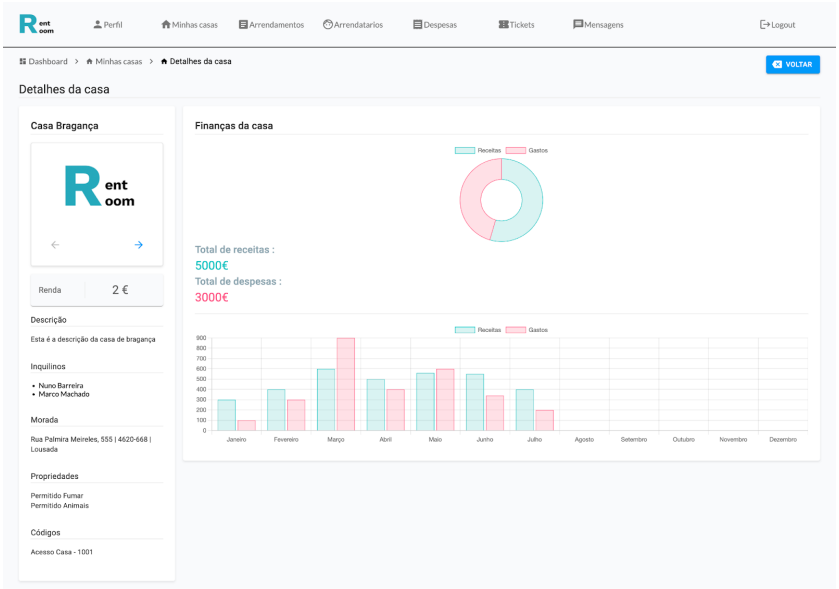

Figura 5.16: Visualização informação de propriedade.

No ecrã de consultas das propriedades (figura 5.15), é possível o utilizador adicionar uma nova propriedade ao clicar no botão "Adicionar Nova Casa", que o redirecionará para o ecrã da figura 5.17. Neste ecrã o proprietário encontra um formulário com as propriedades presentes na entidade "Casa". Para além dos campos mais comuns como o nome da propriedade, renda, características da propriedade, é possível também adicionar imagens da propriedade e também códigos de acesso que sejam necessários na propriedade. Quando o proprietário confirma o formulário de criação da propriedade, o sistema irá verificar que todas as informações da propriedade foram inseridas corretamente, através da validação dos campos. Após esta validação de que todos os campos estão conforme o expectável, o sistema irá então criar esta casa na base de dados e criará um chat referente a esta casa, que poderá ser acedido no ecrã da figura 5.37.

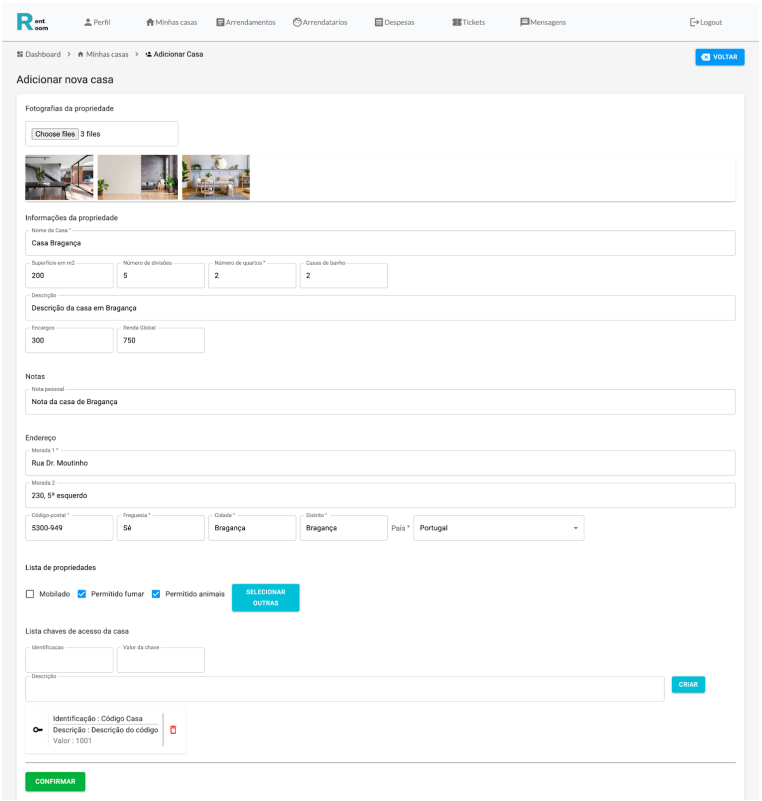

Figura 5.17: Registo de propriedade.

5.2.7 Consultar Arrendatários

O proprietário poderá usufruir das funcionalidades relacionadas com a entidade "Arrendatários" ao selecionar esse botão na barra de navegação. Ao pressionar neste botão o utilizador será reencaminhado para o ecrã (figura 5.18) onde é possível consultar todos os arrendatários registados até ao momento.

Neste ecrã, representado na figura 5.18, o proprietário tem a possibilidade de executar funcionalidades como a de eliminar um arrendatário. Neste caso o proprietário encontrará um modal de confirmação de eliminação do arrendatário, um ecrã que se assemelha com o da figura 5.25. Se o proprietário confirmar a eliminação do arrendatário, o sistema irá validar se o arrendatário está a ser referenciado em algum arrendamento. Na eventualidade de esta condição se verificar, então uma notificação irá ficar visível, informando que a ação não é possível porque o arrendatário está associado a um arrendamento ativo. Caso contrário, o utilizador será eliminado da base de dados, bem como o seu acesso como utilizador.

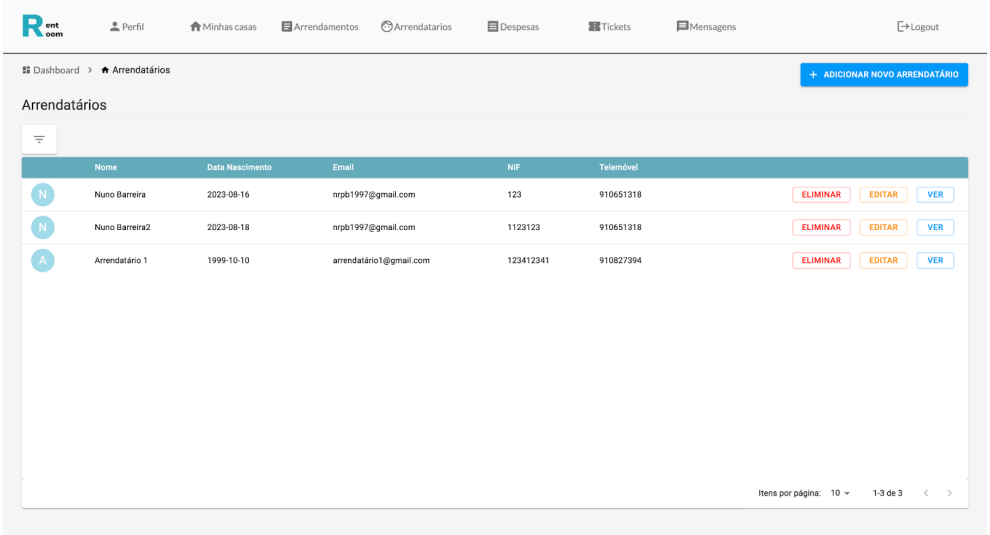

Figura 5.18: Consulta de arrendatários.

Neste ecrã de consulta de arrendatários está presente um ícone de filtragem que ao interagir com o mesmo, este expande e colapsa, mostrando os campos de possivelmente filtragem ou ocultando os mesmos respetivamente. Estes filtros presentes na figura 5.18 e

5.19 possibilitam ao utilizador filtrar os resultados da tabela de registos de arrendatários, de acordo com os parâmetros submetidos, podendo estes ser o nome do arrendatário, o email, NIF ou telemóvel. O proprietário ao aplicar os filtros, um novo pedido será feito à API, contendo os novos parâmetros, sendo os resultados novos obtidos mostrados de seguida na tabela.

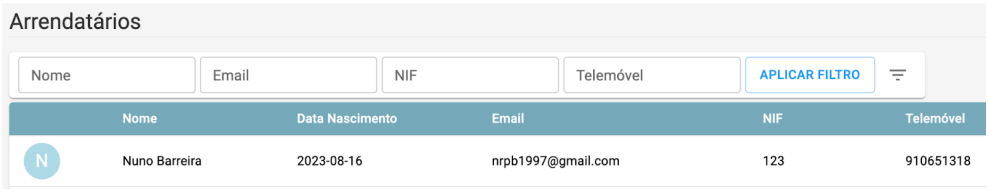

Figura 5.19: Filtros presentes na consulta de arrendatários.

5.2.8 Criar Arrendatário

No ecrã da figura 5.18 é possível visualizar um botão de criação de um novo arrendatário. Ao pressionar neste, o proprietário é reencaminhado para o ecrã da figura 5.20, onde está presente um formulário com as propriedades que o arrendatário possui. No formulário é possível o proprietário definir as informações pessoais do arrendatário a ser criado, endereço principal do mesmo, bem como adicionar documentos que possam ser relevantes associar a este arrendatário.

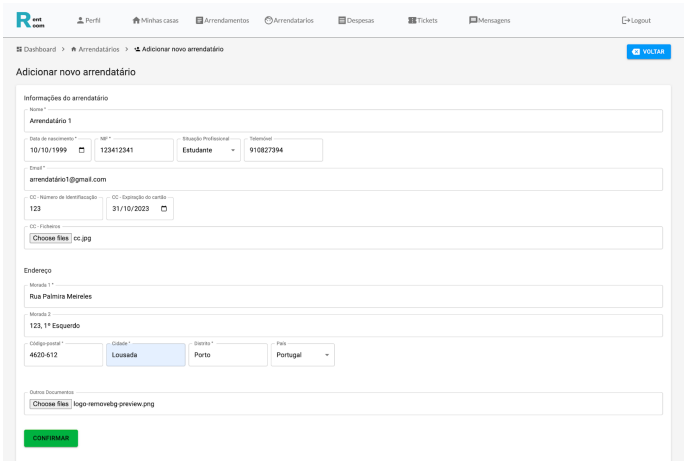

Figura 5.20: Registo de arrendatário.

No momento em que o formulário é confirmado, o sistema irá verificar se todos os campos obrigatórios foram preenchidos e fará uma validação de todos os campos. Após a validação o arrendatário é criado. Ao ser criado, o sistema irá criar também um utilizador para possibilitar o acesso do arrendatário à aplicação, gerando também uma password para o mesmo. De seguida um email é enviado pelo sistema para o email do arrendatário criado, contendo este informação sobre o acesso do mesmo à plataforma (figura 5.21).

Figura 5.21: Email enviado ao arrendatário criado.

5.2.9 Ver Arrendatário

Após a criação do arrendatário, o proprietário é reencaminhado novamente para o ecrã de consulta de arrendatários. Neste ecrã poderá ainda visualizar as informações do mesmo. Ao fazê-lo este irá visualizar o ecrã da figura 5.22, sendo possível obter todas as informações submetidas anteriormente acerca do arrendatário, bem como visualizar e fazer download dos ficheiros associados. Ao editar o arrendatário, o proprietário poderá editar todas as informações do arrendatário com a exceção do email, bem como remover e adicionar ficheiros.

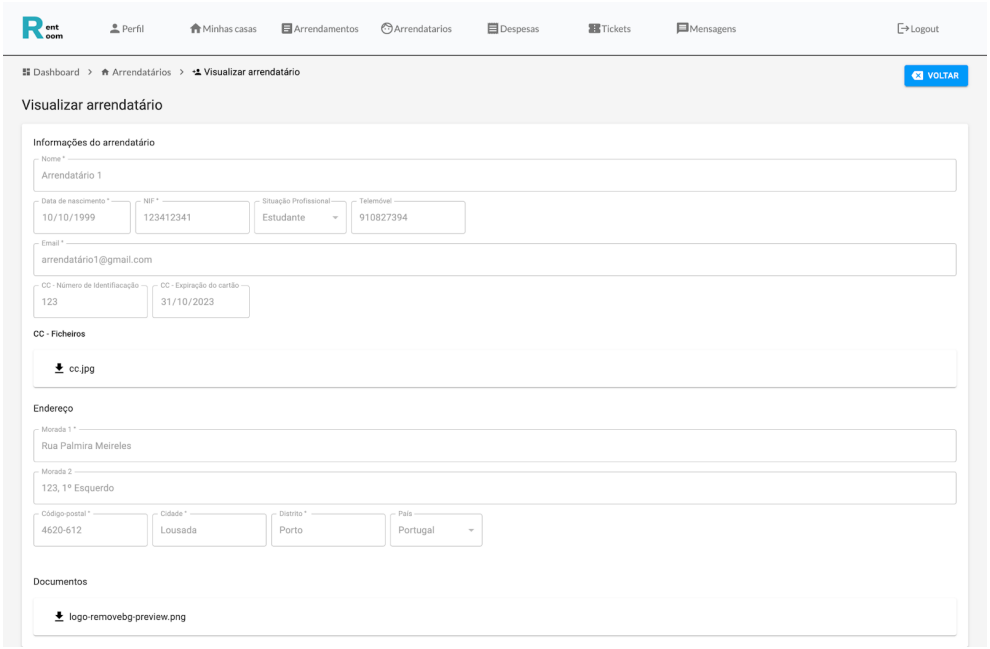

Figura 5.22: Visualizar informação de um arrendatário.

5.2.10 Consultar Arrendamentos

Na navegação é ainda possível selecionar o botão "Arrendamentos" que encaminhará o utilizador (tanto o proprietário como o arrendatário têm acesso a este ecrã) para o ecrã da figura 5.23. Este ecrã é semelhante aos outros de consulta. Neste é possível o proprietário consultar todos os arrendamentos no qual poderá executar as suas funcionalidades de eliminação, criação e visualização. Na tabela tanto o proprietário como os arrendatários poderão ver informações reduzidas dos arrendamentos associados aos mesmos, bem como o seu estado.

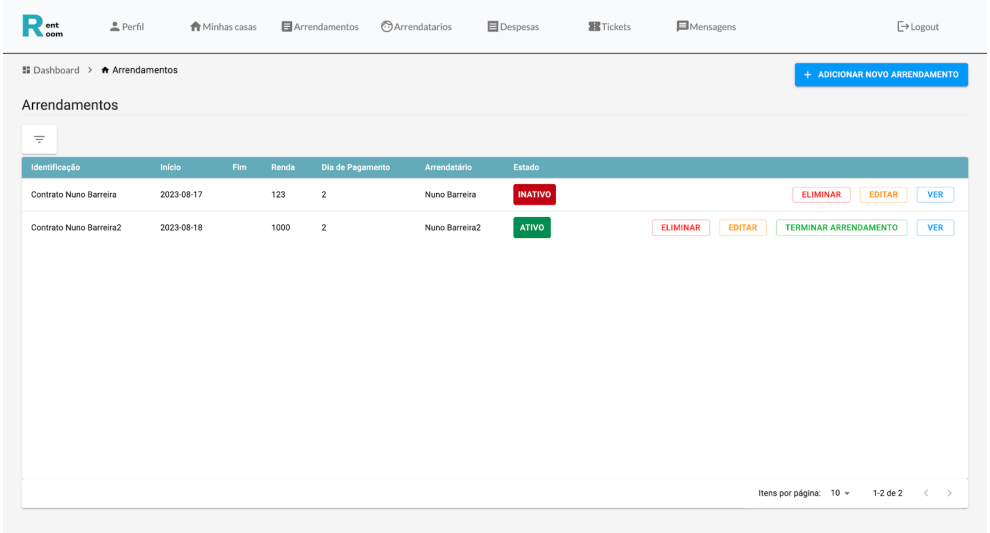

Figura 5.23: Consulta de arrendamentos.

Esta tabela de consulta possui também uma filtragem, possibilitando assim que o utilizador faça uma seleção dos arrendamentos que quer visualizar. Esta filtragem, presente na figura 5.24, tem como parâmetros a identificação do arrendamento e o estado do arrendamento, apresentando assim a tabela, resultados de acordo com os valores de filtragem do utilizador.

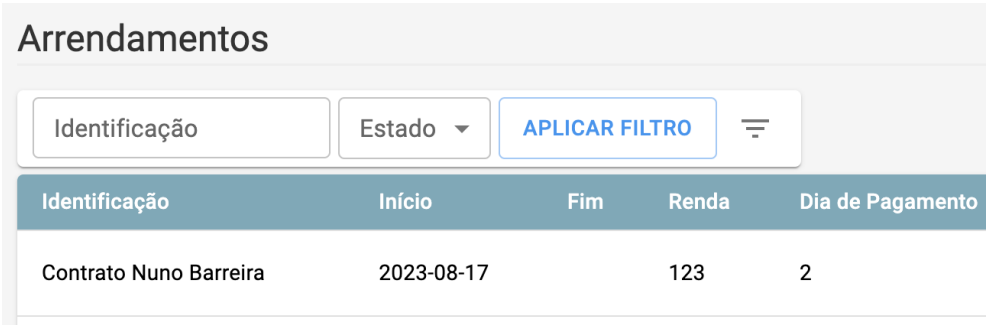

Figura 5.24: Filtros na consulta de arrendamentos.

Para além das funcionalidades básicas de criação, remoção e edição, presente em todas as tabelas de consulta, o proprietário pode ainda executar uma outra funcionalidade na tabela de consulta dos arrendamentos. Este poderá terminar um arrendamento específico, onde lhe aparecerá um modal de confirmação (figura 5.25), de forma a confirmar a intenção da ação. Após esta confirmação, o estado do arrendamento irá mudar para inativo,

deixando este de estar associado ativamente à casa associada. Este estado tem influência por exemplo na criação de uma despesa, onde é possível escolher o arrendatário alvo para a despesa. Caso o arrendamento esteja inativo, o arrendatário associado não irá aparecer disponível para alvo de uma despesa.

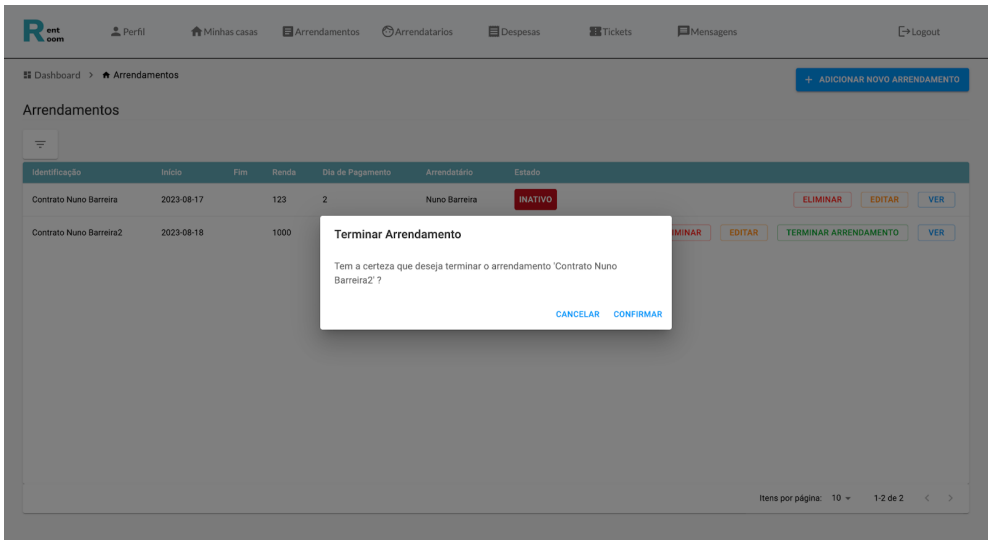

Figura 5.25: Modal de confirmação de ação.

5.2.11 Criar Arrendamento

Através do ecrã de consulta de arrendamentos, o proprietário poderá criar um novo arrendamento ao pressionar o botão no canto superior direito da página (figura 5.23) onde navegará para o ecrã de registo de um novo arrendamento, visualizado na figura 5.26. Neste ecrã de registo, o proprietário poderá definir o tipo de duração de contrato, podendo este ser determinado ou indeterminado. Caso seja indeterminado, este só terá de definir o início do mesmo. Caso seja determinado, para além do início do arrendamento, o proprietário necessitará de definir a data de finalização do contrato. Para além destas informações, outras como recorrência de pagamento, método de pagamento, dia de pagamento, renda e caução precisam de ser definidas. No final do formulário, ainda é necessário que o proprietário defina qual a propriedade destinada ao arrendamento e quem é o arrendatário associado. Neste campo de arrendatário, aparecerão apenas os

arrendatários criados que ainda não tenham nenhum contrato de arrendamento definido. Aliado a estas informações, documentos poderão ser também anexados ao arrendamento a ser criado, como é possível visualizar no final do formulário. Na conclusão do registo de um arrendamento, este irá ser validado pelo sistema e criado na base de dados. Consequentemente, o sistema irá adicionar o arrendatário ao chat da casa em questão, tendo o arrendatário assim acesso ao chat da casa, podendo este enviar e visualizar mensagens.

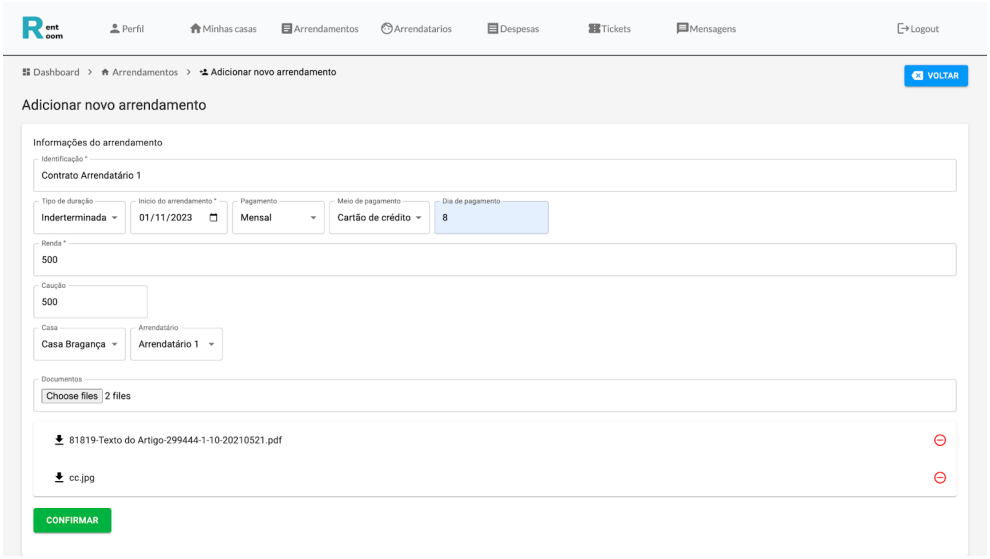

Figura 5.26: Registo de um arrendamento.

5.2.12 Visualizar Arrendamento

Após a criação de arrendamentos, estes poderão ser visualizados através do ecrã de consulta (figura 5.23) que reencaminhará o utilizador para os detalhes todos de um arrendamento presente na figura 5.27. Para além da visualização de todas as informações, o utilizador poderá ainda fazer o download dos ficheiros anexados ao arrendamento.

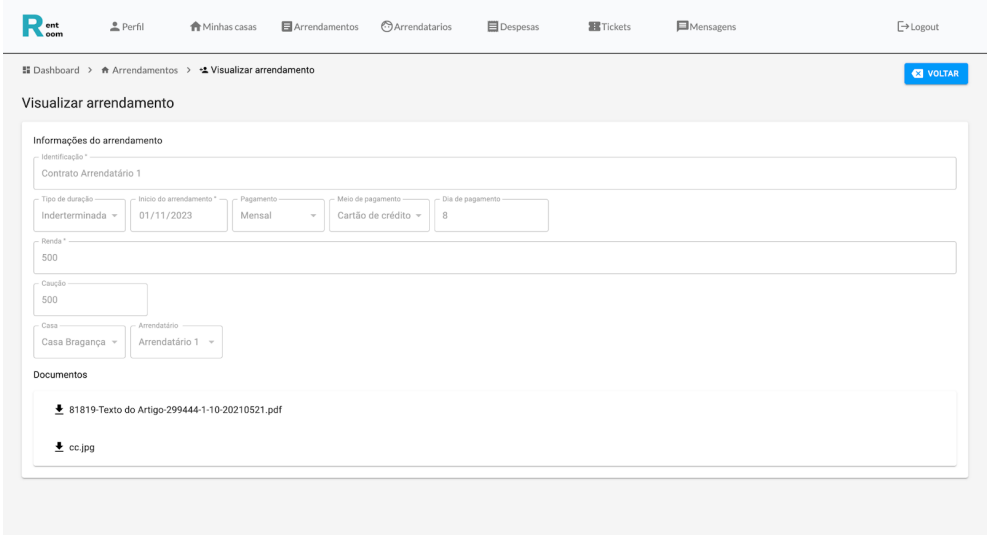

Figura 5.27: Visualização de informação de um arrendamento.

5.2.13 Consultar Despesas

O ecrã de consulta de despesas é similar aos restantes ecrãs de consulta anteriormente referidos e pode ser visualizado pela figura 5.28. Este dá uma visão geral de todas as consultas associadas ao utilizador, quer seja ele o criador da despesa ou o alvo da despesa. Na tabela é possível ver algumas informações básicas das despesas registadas como a data de emissão, descrição, a casa associada à despesa, o arrendatário associado à despesa caso exista, o tipo de despesa, valor, data limite para pagamento e o estado da despesa. Para além disto é ainda visível se a despesa é um gasto para o utilizador logado ou se é uma receita, através da primeira coluna da tabela.

Para além das ações básicas que podem ser executadas pelo proprietário, como eliminar e editar uma despesa, este poderá também confirmar as mesmas através do botão "Confirmar Pagamento". Esta ação muda o estado da despesa para "Pago".

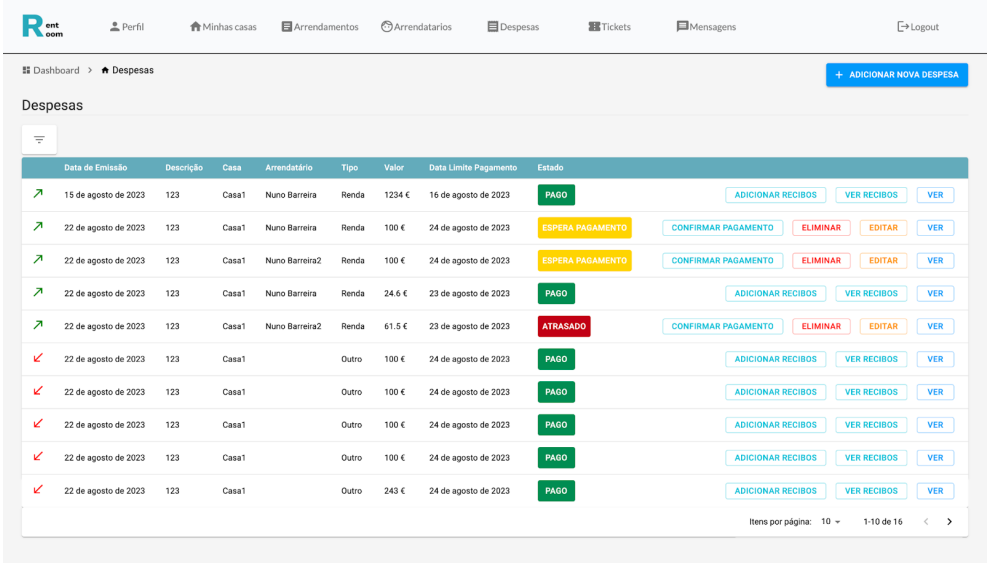

Figura 5.28: Consulta das despesas.

5.2.14 Filtros

Como já foi dito, de forma similar aos outros ecrãs de consulta, este também possui um sistema de filtragem de resultados a serem mostrados. Nesta barra de filtragem o utilizador pode definir vários parâmetros de filtragem. As propriedades que podem ser filtradas, como a figura 5.29 demonstra, são a casa, arrendatário, tipo e estado. No caso do tipo e do estado, estas propriedades podem ser filtradas pelas opções limitadas que as mesmas possuem, enquanto que a casa e arrendatário são campos de escrita livre.

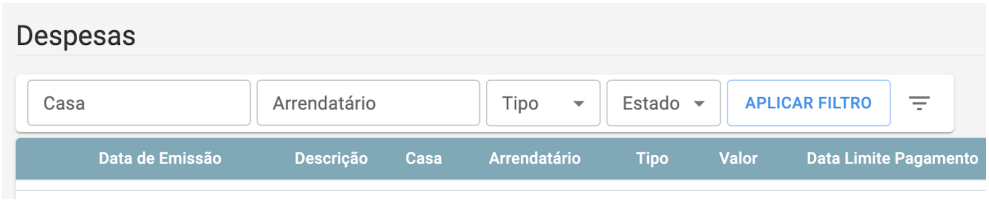

Figura 5.29: Filtros na consulta das despesas.

5.2.15 Paginação

Para além da filtragem, todas as páginas de consulta presentes na plataforma, possuem um sistema de paginação de forma a possibilitar que na eventualidade de existirem muitos registos, neste exemplo de despesas, sejam apenas pedidos à base de dados uma quantidade limitada dos mesmos, permitindo assim que a manipulação destes dados seja mais fácil e que a velocidade de resposta e visualização de dados seja mais eficiente e rápida.

Neste sistema de paginação, é possível definir o limite de dados que queremos que sejam mostrados na tabela. Para além desta limitação de resultados é possível navegar entre páginas, caso hajam mais resultados totais do que o limite escolhido, como é exemplificado na figura 5.30, onde o total de registos é 16, no entanto os itens por página escolhidos é de 10. Neste exemplo é possível verificar que atualmente o utilizador está na página 1 e que o botão para passar para a página seguinte está habilitado.

No caso exemplificado na figura 5.30 verifica-se o caso em que o número de itens por página escolhido é maior que o número de resultados, portanto os botões de navegação entre páginas estão desabilitados.

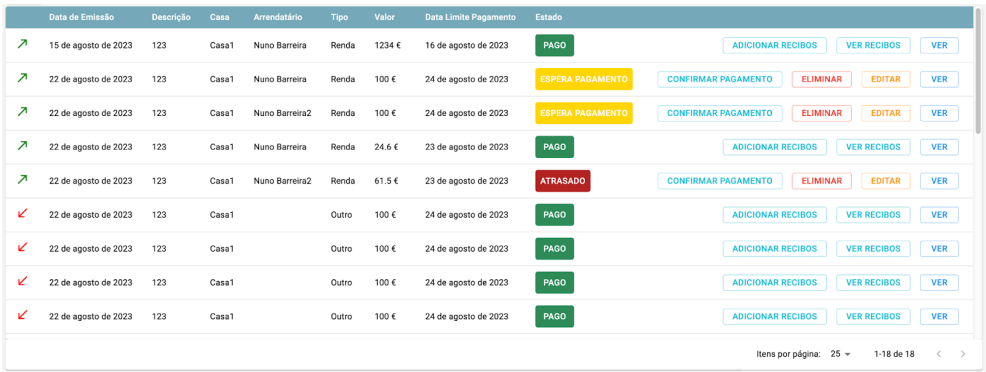

Figura 5.30: Paginação numa consulta de despesa.

A funcionalidade de adicionar um recibo a uma despesa está também ela presente em todos os registos na tabela de despesas que estejam no estado “Pago”. Ao executar esta funcionalidade, um modal aparecerá no ecrã como é exemplificado na figura 5.31. Neste modal o proprietário poderá adicionar os ficheiros respetivos e associa-los à despesa em questão.

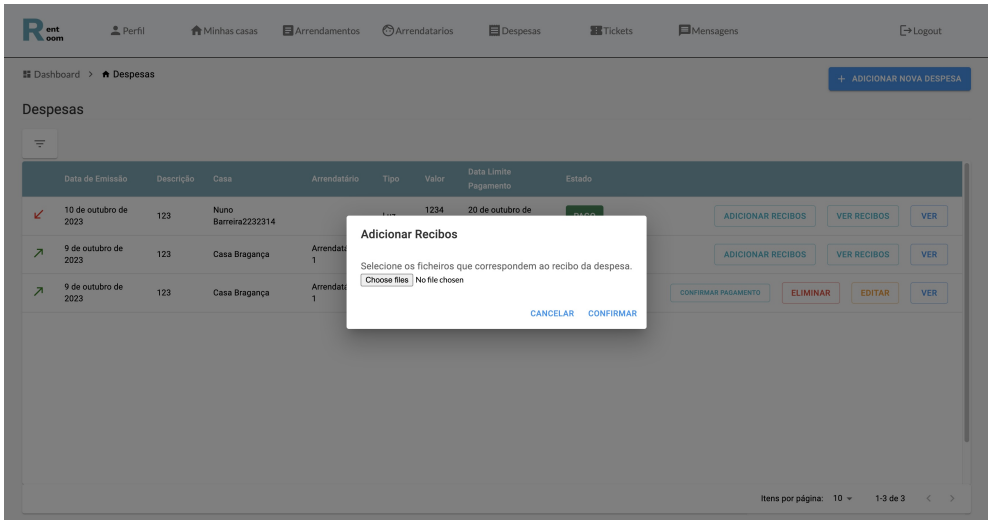

Figura 5.31: Adicionar recibo a uma despesa.

Após associados recibos a uma despesa paga, o utilizador poderá visualizar as mesmas através da tabela de consulta de despesas ao pressionar o botão "Ver Recibos". Após essa ação, o modal para ver os recibos é aberto e o utilizador poderá fazer o download de todos estes recibos (figura 5.32).

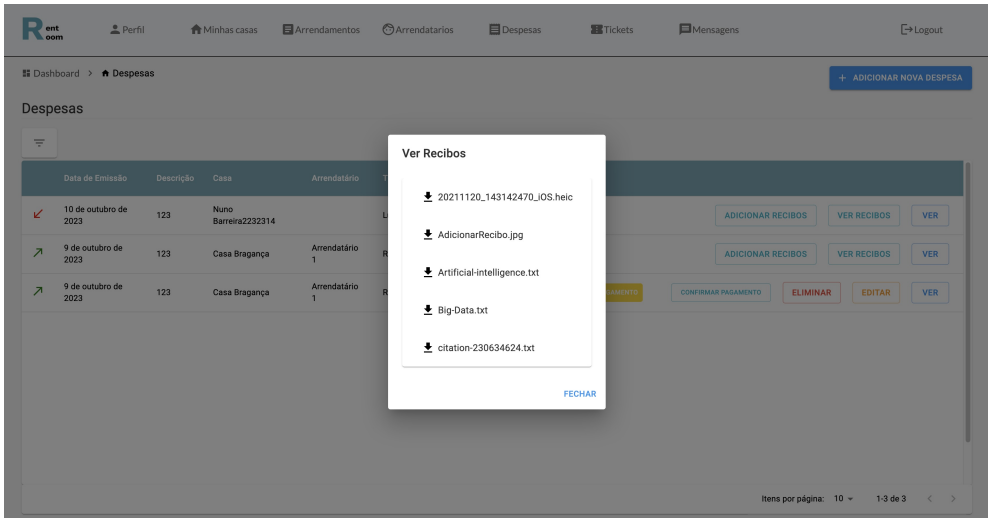

Figura 5.32: Visualização de recibos.

5.2.16 Adicionar Despesa

No ecrã de consulta de despesas (figura 5.28) o proprietário tem ainda a funcionalidade de criar uma despesa nova ao pressionar o botão presente no canto superior direito da página. Isto irá redirecionar o proprietário para o ecrã de registo de uma nova despesa, exemplificado na figura 5.33, onde este poderá especificar as propriedades relacionadas com a despesa. No formulário o proprietário terá de definir no primeiro campo qual a propriedade associada à despesa. Neste campo aparecerão para selecionar apenas as casas já criadas. No campo seguinte este poderá definir o responsável pelo pagamento da despesa. Neste campo de seleção, o proprietário tem 3 opções:

- Casa: Esta opção visa a divisão da despesa a ser criada pelos vários elementos da casa já selecionada. Ao selecionar esta opção, uma caixa aparecerá em baixo, como é possível visualizar na figura 5.33, contendo as percentagens correspondentes a dividir pelos arrendatários disponíveis na casa selecionada no primeiro campo. No exemplo, é possível verificar que existem dois arrendatários na casa selecionada "Casa1", onde a percentagem da despesa correspondente a cada arrendatário é de 50%. Este valor percentual é automaticamente atribuido ao dividir o total percentual pelo número de arrendatários disponíveis na casa.

- Proprietário: Selecionando esta opção o proprietário registará uma despesa para si próprio, não contando esta como uma receita no ecrã de visualização de uma propriedade, mas sim como uma despesa associada ao mesmo.

- Arrendatário: Ao selecionar esta opção o proprietário estará a criar uma despesa, tendo como responsável como pagamento pela despesa um arrendatário da casa selecionada. Na seleção desta opção, um campo aparecerá por baixo deste, onde o utilizador poderá selecionar um arrendatário pertencente à casa previamente selecionada.

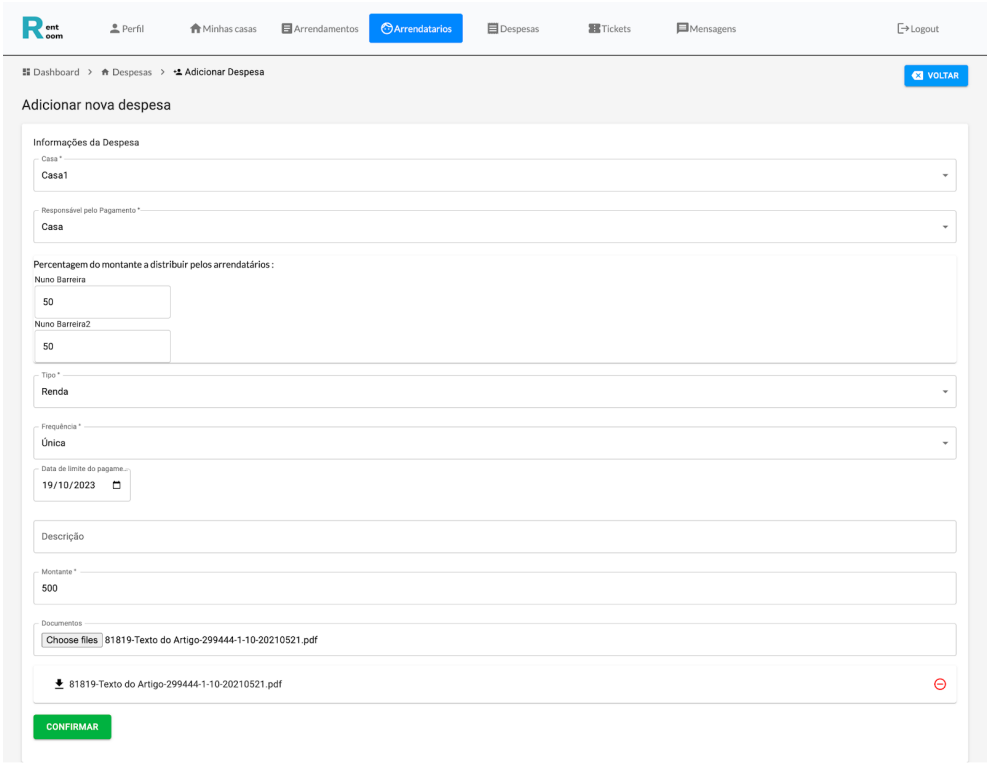

Figura 5.33: Registo de uma despesa.

Ao ser confirmada a criação desta despesa o sistema irá validar o correto preenchimento de todos os campos. Após esta validação o sistema criará a despesa na base de dados com o estado "Espera Pagamento", exceto quando o responsável pelo pagamento é o próprio proprietário. Neste caso o estado da despesa é criado como "Pago". No caso de o responsável ser o arrendatário, o estado da despesa apenas mudará caso o proprietário confirme o pagamento da despesa, mudando o estado para "Pago", ou a data limite de pagamento seja ultrapassada, passando a despesa para o estado "Atrasada". Na ocorrência de o proprietário ter optado pela opção da divisão da despesa pela casa, escolhendo no campo do responsável da despesa a opção "Casa", então o sistema irá dividir o montante da despesa pelos arrendatários da casa, tendo em conta a percentagem escolhida para cada um dos arrendatários e irá criar uma despesa para cada um deles. No exemplo da figura 5.33 o montante escolhido para a divisão das despesas pelos arrendatários é de 500 e o número de arrendatários é de 2, sendo que cada um irá pagar 50% do montante escolhido.

Ao confirmar esta criação de despesa, é possível observar através da figura 5.34 que, conforme esperado, foram criadas duas despesas, uma para cada um dos arrendatários associados à casa escolhida "Casa1". O montante para cada uma dessas despesas é então 250 que corresponde aos 50% do montante total escolhido para dividir.

Figura 5.34: Divisão de despesas por 2 arrendatários.

5.2.17 Consultar Tickets de Reparação

Uma funcionalidade da aplicação seria também a de consulta de tickets de reparação. Esta funcionalidade pode também ser acedida através da barra de navegação. No ecrã de consulta de todos os tickets de reparação, também à semelhança com os restantes ecrãs de consulta, é possível ver numa tabela toda a informação de todos os tickets registados (figura 5.35). Os dados referentes a cada ticket são a data de criação, título, descrição, casa associada ao ticket, data de resolução caso já tenha sido resolvida, e o estado do ticket. Para além da tabela com a paginação já presente em todas as outras tabelas de consulta, é possível também filtrar os resultados desta tabela. Nesta tabela é possível o utilizador filtrar os resultados pela casa associada ao ticket de reparação, estado do ticket de reparação e também a casa associada ao ticket de reparação.

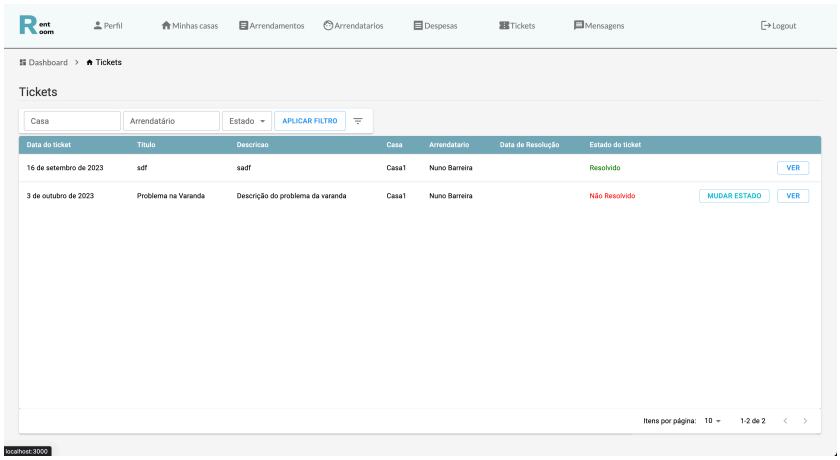

Figura 5.35: Consulta dos tickets de reparação.

O arrendatário poderá ainda criar um ticket de reparação através do botão presente no ecrã de consulta de tickets. Caso o faça, este será reencaminhado para o ecrã de criação de tickets de reparação, que se pode visualizar na figura 5.36. Neste formulário o arrendatário poderá preencher as informações do problema que está a enfrentar, escrevendo um breve título, uma descrição sobre a reparação que é necessária que seja feita e adicionando os ficheiros que possam ser necessários. Ao confirmar a criação do ticket, o sistema irá criar na base de dados o mesmo com a informação introduzida e com o estado "Não Resolvido". O estado dos tickets apenas mudará para "Resolvido" quando o proprietário confirmar a resolução de um ticket na tabela de consulta de tickets (figura 5.35)

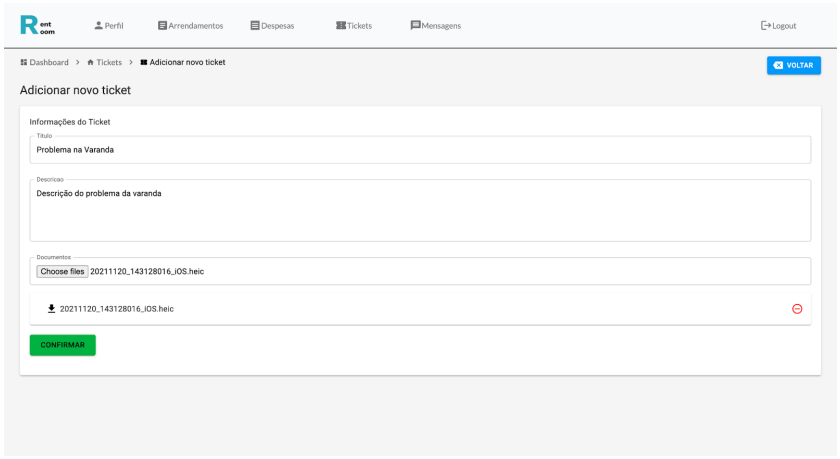

Figura 5.36: Criação de ticket de reparação.

5.2.18 Chats

Outra funcionalidade pretendida para o desenvolvimento da aplicação era a comunicação entre o proprietário e os vários elementos de uma casa. Para esta necessidade foi desenvolvido um chat que permite esta mesma comunicação. Estes chats são gerados na criação de uma casa como já foi referido anteriormente neste capítulo. À medida que arrendamentos são criados para uma casa, os arrendatários associados são associados como participantes ao chat a que a casa corresponde.

Os chats podem ser acedidos através da barra de navegação, sendo o utilizador reencaminhado para o ecrã visível na figura 5.37. Na coluna presente na parte esquerda desta página, o utilizador poderá aceder ao chat que pretende visualizar, e em tempo real, as mensagens do respetivo chat aparecerão no componente que é possível ser observado na parte direita do ecrã. No componente à direita neste ecrã, o utilizador poderá visualizar todas as mensagens enviadas pelos vários arrendatários ou proprietário pertencentes à casa associada ao chat. As mensagens que aparecem com um fundo verde e na parte direita da janela pertecem ao utilizador logado, enquanto que as com fundo cinzento e mais à esquerda, pertencem aos restantes utilizadores. Nestas mensagens pode visualizar-se o conteúdo da mensagem, a hora a que foi enviada e o nome do utilizador que a enviou. Ainda neste componente o utilizador conseguirá escrever uma mensagem através da parte inferior e enviar a mesma ao clicar no botão “Enviar”. Após esta ação, a mensagem irá ser guardada na base de dados, podendo ser visualizada por todos os utilizadores que tiverem acesso ao chat em questão.

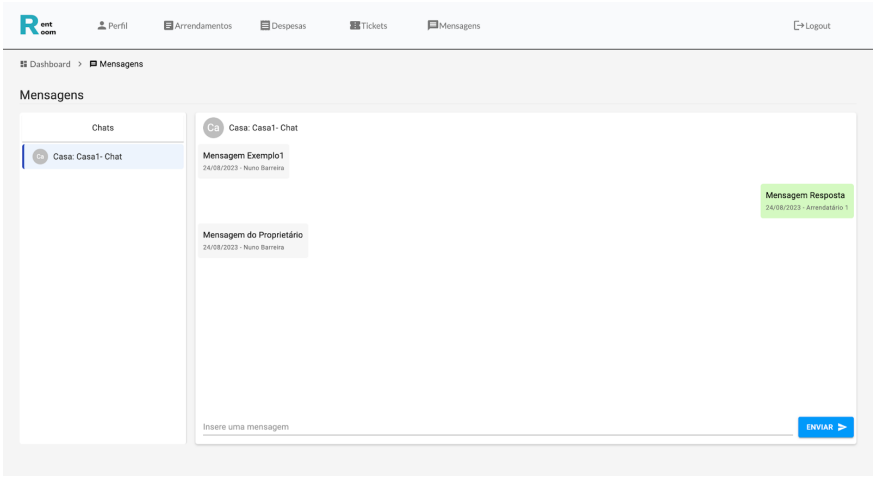

Figura 5.37: Chat.

5.2.19 Tema Escuro

De forma a melhorar a visibilidade da aplicação, dependendo das preferências e condições de ambiente em que o utilizador se encontra, foram criados dois temas. O tema claro, estando este selecionado de maneira pré-definida, e o tema escuro, que é possível selecionar nas configurações do utilizador, na secção "Tema", presente na figura 5.13 com o tema claro aplicado, e na figura 5.38 com o tema escuro aplicado.

Após aplicado o tema escuro, toda a aplicação irá reagir de forma a mostrar os componentes com a paleta de cores definida para o tema escuro.

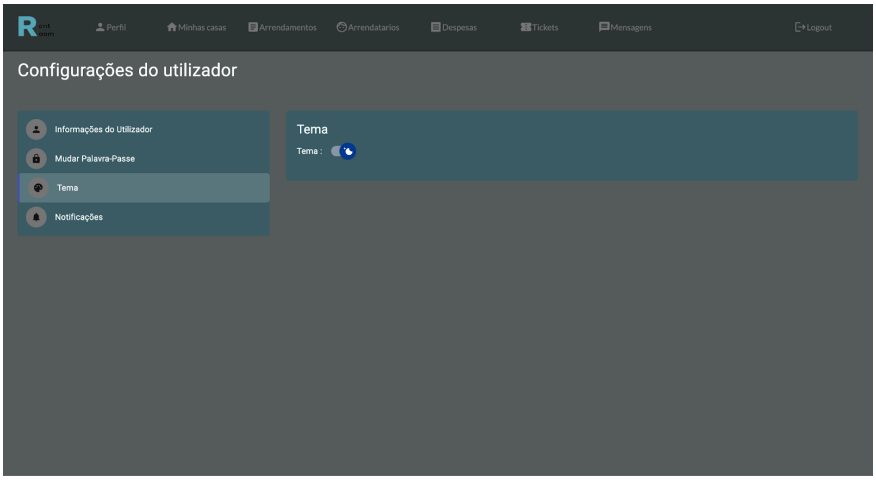

Figura 5.38: Ativação de tema escuro.

Aplicando a paleta de cores do tema escuro à pagina inicial, esta ficará com o aspeto da figura 5.39. Esta pode ser então comparada com o mesmo ecrã com o tema claro aplicado, presente na figura 5.8. É possível verificar que os vários tons de cinzento e branco presentes no tema claro, são mudados para tons de preto, cinzento escuro e azul escuro. Quanto aos componentes que representam texto, no tema claro estes são maioritariamente pretos e cinzento escuro de forma a contrastar com os fundos claros, enquanto no tema escuro estes são brancos ou cinzento claro.

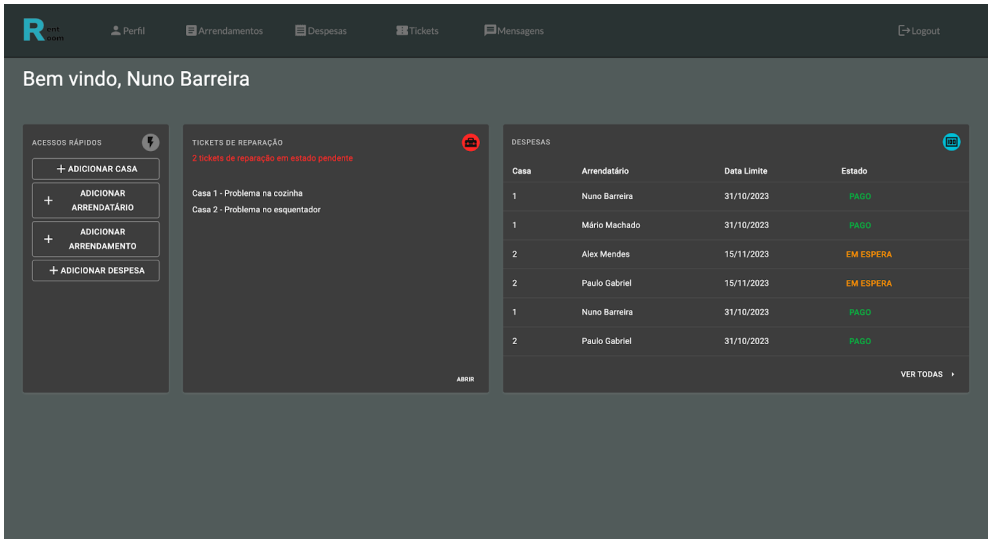

Figura 5.39: Página inicial com tema escuro aplicado.

Nos ecrãs de consulta é notável a diferença entre o tema escuro aplicado (figura 5.40) e o tema claro (figura 5.35). Os botões azuis do tema claro passam a ser brancos de forma a dar um maior contraste, aumentando assim a visibilidade dos mesmos. Os componentes respetivos à tabela e aos filtros em vez de possuirem branco como cor de fundo, ficam pretos e o fundo do ecrã fica com um tom cinzento escuro azulado.

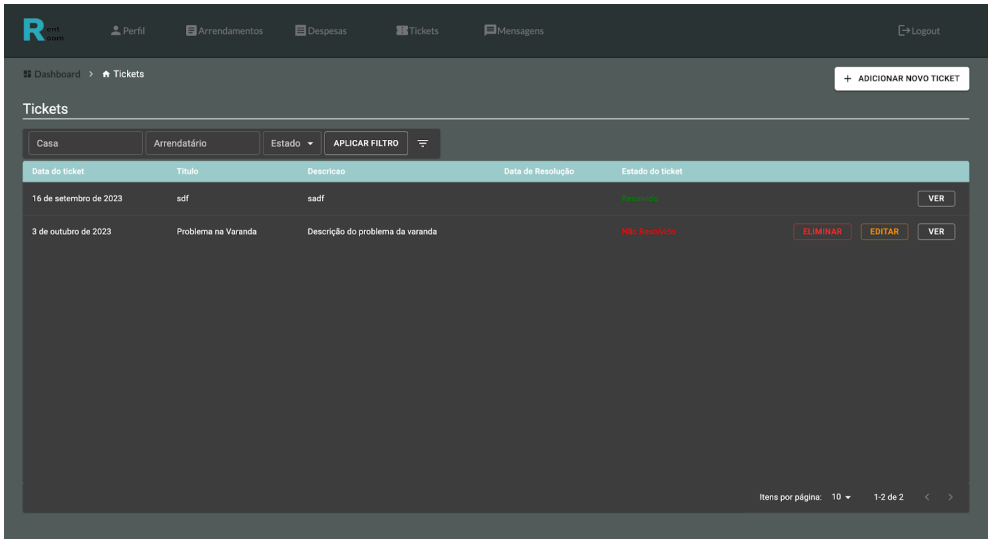

Figura 5.40: Consulta de registos com tema escuro aplicado.

A mesmo mudança pode ser verificada nos ecrãs de criação de entidades. A comparação pode ser verificada ao comparar o tema claro aplicado ao ecrã de criação de tickets de reparação na figura 5.36 e o tema escuro aplicado na figura 5.41.

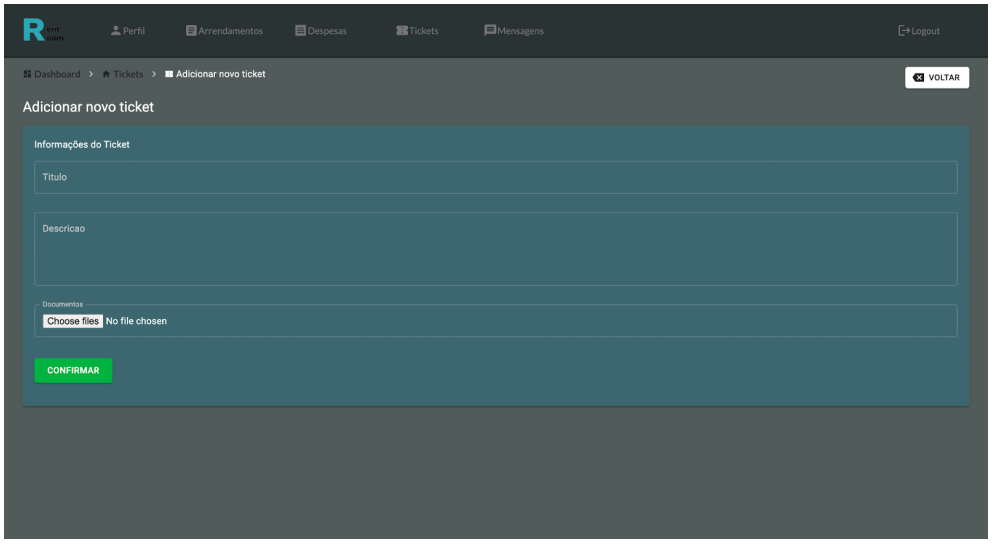

Figura 5.41: Formulário de criação com tema escuro aplicado.

5.2.20 Notificações

Durante o desenvolvimento da aplicação foram também implementadas notificações com a finalidade de indicar se certa ação do utilizador foi bem sucedida ou não. Estas notificações podem ser visualizadas na figura 5.42 e 5.43. A primeira figura diz respeito na qual uma é mostrada quando alguma operação não é executada com sucesso. Neste caso a mudança da palavra-passe. Na segunda figura, temos uma notificação que informa o utilizador que a ação que tentou executar ocorreu corretamente.

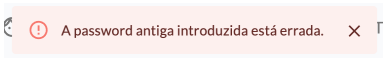

Figura 5.42: Notificação de ocorrência inválida.

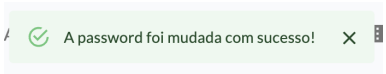

Figura 5.43: Notificação de ocorrência válida.

Capítulo 6

Conclusões

Neste relatório foi apresentado o processo de modelação e desenvolvimento de uma aplicação de gestão de arrendamentos com a implementação de funcionalidades que foram propostas no início da mesma.

Como foi abordado neste artigo, existem bastantes necessidades na gestão de uma propriedade, tanto por parte de inquilinos, como proprietários. A aplicação desenvolvida permite que estas necessidades sejam colmatadas, proporcionando assim uma maior eficiência e rapidez nos processos de gestão de arrendamentos. Com esta aplicação o proprietário tem a possibilidade de gerir informações da sua propriedade, fazer uma gestão de despesas de forma automática, permitindo ter um registo das mesmas. Este também poderá atribuir uma divisão de despesas pelos arrendatários de uma casa. Para além disto o proprietário tem também a possibilidade de associar os recibos a estas despesas, gerir arrendatários e as suas informações assim como gerir os arrendamentos dos mesmos. Por parte do arrendatário, este poderá usufruir de criação de tickets, facilitando assim este processo com o proprietário, permitindo uma maior fluidez do mesmo. A aplicação tem ainda funcionalidades como envio de mensagens, suporte de temas e envio de notificações.

Esta aplicação foi desenvolvida através do React, permitindo o desenvolvimento da parte visual, com a qual o utilizador irá interagir. Foi implementado também uma Api em FastAPI de forma a criar os serviços necessários, interagindo este com a base de dados MongoDB que armazena todos os dados da aplicação.

De forma a melhorar o que foi implementado neste trabalho, com o objetivo de tornar a solução final mais aprimorada, cobrindo mais necessidades dos proprietários e inquilinos, várias funcionalidades adicionais poderiam ser adicionadas à solução. Entre várias implementações possíveis, é de destacar a inclusão de um sistema de transações monetárias de forma a que fosse possível o pagamento de despesas dentro da aplicação. Isto proporcionaria uma forma mais rápida e cómoda de os pagamentos serem executados. Outra possível funcionalidade seria a implementação de um sistema que possibilitasse a geração automática de recibos quando uma despesa fosse paga, tendo em conta todas as informações guardadas acerca do pagamento. O agendamento de despesas e uma maior estatística sobre as informações de receitas e despesas seria também uma melhoria para a aplicação. De forma a concluir, é possível afirmar que o desenvolvimento deste projeto proporcionou um aumento de conhecimento de várias tecnologias e conceitos que serão uma mais valia para um futuro profissional.

Bibliografia

[1] Bvc-holdings. (), URL: https://bvc-holdings.com/a-brief-history-of-real-estate-investing-in-the-last-100-years/.

[2] F. K. Loo, «A Guide to Effective Property Management in Hong Kong», *Hong Kong University Press*, 1994.

[3] e. J. G. Halvitigala Dulani, «THE USE OF PROPERTY MANAGEMENT SOFTWARE IN RESIDENTIAL PROPERTY MANAGEMENT», 2014.

[4] D. Warburton, «The Role of Technology in the Real Estate Industry», 2016.

[5] «How Technology Is Reshaping the Future of Property Management», 2023.

[6] O. B. O. e. A. O. Babajide, «BARRIERS TO ICT DEPLOYMENT IN THE NIGERIAN REAL ESTATE PRACTICE», *FULAFIA JOURNAL OF SCIENCE AND TECHNOLOGY 4*, 2018.

[7] U. S. e. M. M. Mohammed Jibrin; M. Bello, «A MODEL FOR INTEGRATED SMART REAL ESTATE», 2018.

[8] A. Ikuomola, «A Secured Mobile Cloud-Based House Rental Management System», 2020.

[9] Y. L. L. C. Y. Q. W. S. W. Sh, «A Property Management System Using WebGIS», 2023.

[10] J. P. Pires, «A adoção de cloud nas organizações», 2022.

[11] G. A. S. S. F. Y. A. e A.s George., «The Impact of Cloud Hosting Solutions on IT Jobs: Winners and Losers in the Cloud Era», 2023.

[12] Salesforce. (), URL: https://www.salesforce.com/products/platform/best-practices/cloud-computing/.

[13] Digitalturbine. (), URL: https://www.digitalturbine.com/blog/mobile-marketing/the-rise-of-mobile-how-mobile-apps-have-changed-our-lives/.

[14] Internetsociety. (2015), URL: https://www.internetsociety.org/resources/doc/2015/global-internet-report-2015/.

[15] Explodingtopics. (), URL: https://explodingtopics.com/blog/mobile-internet-traffic.

[16] «Big data, artificial intelligence, machine learning and data protection», 2017.

[17] J. Conway, «Artificial Intelligence and Machine Learning: Current Applications in Real Estate», 2018.

[18] Rentila. (), URL: https://www.rentila.pt/.

[19] Gesar. (), URL: https://webteam.pt/gesar-software-gestao-de-arrendamento/.

[20] Ve-Imoveis. (), URL: https://www.ve-imoveis.pt/.

[21] Buildium. (), URL: https://www.buildium.com/.

[22] J. M. A. Patzi, «Sistema de Administración de Propiedades Proyecto de Software Libre Wakodi», 2010.

[23] e. N. S. Agarwal Vibhor, «"Way back then": A Data-driven View of 25+ years of Web Evolution», 2022.

[24] M. Kariyawasam, «The Evolution of the Web: How Application Development Changed as a Result», 2020.

[25] S. Aghaei, «Evolution of the World Wide Web: From Web 1.0 to Web 4.0», 2012.

[26] S. Murugesan, «Understanding Web 2.0», 2007.

[27] K. S. K. S. U. Irfan, «Crawling Ajax-based Web Applications: Evolution and State-of-the-art. Malaysian Journal of Computer Science.», pp. 31. 35–47, 2018.

[28] S. Batra, «AJAX - Asynchronous Java Script and XML», 2006.

[29] e. A. S. Ranga Virender, «API Features Individualizing of Web Services: REST and SOAP», 2019.

[30] F. S. G. Ira W. Cotton, «Data structures and techniques for remote computer graphics», 1968.

[31] C. Date, «The Relational and Network Approaches: Comparison of the Application Programming Interface», 1974.

[32] T. Data, «Short History of API | From Cabinet to Big BOOM», 2022.

[33] «Short History of API | From Cabinet to Big BOOM», 2022.

[34] (), URL: `https://konghq.com/learning-center/api-management/different-api-types-and-use-cases`.

[35] e. A. S. Ranga Virender, «API Features Individualizing of Web Services: REST and SOAP», 2019.

[36] R. T. Fielding, «Architectural Styles and the Design of Network-based Software Architectures», 2000.

[37] (), URL: `https://www.mulesoft.com/pt/resources/api/what-is-rest-api-design`.

[38] H. F. e Erenis Ramadani, «Web Services: A Comparison of Soap and Rest Services», 2018.

[39] e. E. R. Halili Festim, «Web Services: A Comparison of Soap and Rest Services», 2018.

[40] L. J. O. Felipe, «Design and development of a rest-based web service platform for applications integration», 2010.

[41] M. contributors, rel. téc., 2023.

[42] X. Q. V.-C. Nguyen e K. Richta, «Domain Specific Language Approach on Model-driven Development of Web Services», 2014.

[43] A. B. H. of Python. (), URL: https://exyte.com/blog/a-brief-history-of-python.

[44] S. Cass, «Top Programming Languages 2022», 2022.

[45] E. Fetisov, «Top 30 Companies That Use Python for Success and Profit», 2023.

[46] e. D. G. Rayhan Abu, «The Rise of Python: A Survey of Recent Research», 2023.

[47] fastapi. (), URL: https://fastapi.tiangolo.com/.

[48] L. S. Vailshery, «Most used web frameworks among developers worldwide, as of 2023», 2023.

[49] O. Treasure. (). An Introduction to Using FastAPI, URL: https://refine.dev/blog/introduction-to-fast-api/#introduction.

[50] K. Shahid, 2023.

[51] J. Sandy, «Choosing between Django, Flask, and FastAPI», 2021.

[52] mozilla. (), URL: https://developer.mozilla.org/pt-BR/docs/Web/HTML.

[53] ——, (), URL: https://developer.mozilla.org/pt-BR/docs/Learn/Getting_started_with_the_web/CSS_basics.

[54] w3. (), URL: https://www.w3.org/People/Raggett/book4/ch02.html.

[55] ——, (), URL: https://www.w3.org/Style/CSS20/history.html.

[56] «Anna Monus», 2023.

[57] mozilla. (), URL: https://developer.mozilla.org/pt-BR/docs/Glossary/JavaScript.

[58] R. Souza. (). JavaScript Everywhere — Parte 1: História Do JavaScript, URL: https://medium.com/@rafaelsouzaim/javascript-everywhere-ato-1-hist%C3%B3ria-do-javascript-f26bea74b6d0.

[59] E. P. Alex Banks, «Learning React: Modern Patterns for Developing React Apps», 2020.

[60] stateofjs. (), URL: https://2022.stateofjs.com/en-US/libraries/front-end-frameworks/.

[61] geeksforgeeks. (), URL: https://www.geeksforgeeks.org/what-are-the-advantages-of-react-js/.

[62] R. P. e Archana N Mahajan, «ReactJS: A Modern Web Development Framework», 2020.

[63] reactjs. (), URL: https://legacy.reactjs.org/docs.

[64] T. Stoyko, «Best Examples of Apps Written in React.js», 2023.

[65] (), URL: https://flatlogic.com/blog/bootstrap-vs-material-ui-which-one-to-use-for-the-next-web-app/.

[66] A. Sirotka, «What is Material UI?», 2022.

[67] R. S. e Anand Jain e Priyanka Patil e Mahesh Patil, «Mongo DB GUI Operation Using Python», 2015.

[68] B. K. e Peter Bakkum e Shaun Verch e Douglas Garrett e Tim Hawkins, «MongoDB in Action», 2016.

[69] T. C.-O. e Florin Radulescu e Alexandru Boicea e Ion Bucur, «Performance Evaluation for CRUD Operations in Asynchronously Replicated Document Oriented Database», 2015.

[70] S. Borkar, «Why use Mongo DB?», 2023.

[71] ROZENFELD, «Gestão de Desenvolvimento de Produtos: uma referência para a melhoria do processo», 2006.

[72] G. C. O. Emmanuel Sävio Silva Freire e M. E. de Sousa Gomes, «Analysis of open-source CASE tools for supporting software modeling process with UML», pp. 51–60, 2018.

[73] U. M. L. UML. (2017), URL: https://www.omg.org/spec/UML/.

[74] CAMPESE, «ESTUDO DOS MÉTODOS DE UCD», 2015.

[75] M. COHN, «Advantages of User Stories for Requirements Why User Stories», 2004.

[76] M. J. REES, «A Feasible User Story Tool for Agile Software Development», 0–8, 2002.

[77] P. D. OGLIO, «Uma Ferramenta para Gerenciamento de Requisitos em Projetos Baseados em Extreme Programming Resumo», 2006.

[78] S. Z. I. L. C. J. Costa., «USER STORIES: QUEM, QUANDO E COMO DEVE SER USADO?», pp. 810–18, 2017.

[79] K. Somvanshi. (2023). Structuring a FastAPI App: An In-Depth Guide, URL: https://medium.com/@ketansomvanshi007/structuring-a-fastapi-app-an-in-depth-guide-cdec3b8f4710.

[80] V. F. DANTAS, «Uma Metodologia para o Desenvolvimento de Aplicações Web num Cenário Global», 112 f, 2003.

[81] Notion. (2023), URL: https://www.notion.so/use-case/scrum-board.

[82] C. A. Team. (2019). What is an IDE., URL: https://www.codecademy.com/article/what-is-an-ide.

[83] O. Ossadon. (2018). The Ultimate VSCode Setup for Front End/JS/React, URL: https://medium.com/productivity-freak/the-ultimate-vscode-setup-for-js-react-6a4f7bd51a2.

[84] D. Khmelenko. (2022). This Is Why Developers Choose Visual Studio Code, URL: https://medium.com/geekculture/this-is-why-developers-choose-visual-studio-code-f3df266f2dae.

[85] Python. (), URL: https://www.python.org/downloads/.

[86] L. França, «Como criar uma API em Python com FastAPI e primeiras impressões sobre o Framework», rel. téc., 2020.

[87] FastAPI. (), URL: https://fastapi.tiangolo.com/tutorial/.

[88] Uvicorn. (), URL: https://www.uvicorn.org/.

[89] Swagger. (), URL: https://swagger.io/.

[90] K. Tewouda. (2023). FastAPI and pagination, URL: https://lewoudar.medium.com/fastapi-and-pagination-d27ad52983a.

[91] Atlassian. (), URL: https://developer.atlassian.com/server/confluence/pagination-in-the-rest-api/.

[92] M. Clemente, «Servidor SMTP: o que é, como funciona e como configurar», 2019.