



Escola Superior de Tecnologia e Gestão

Politécnico de Coimbra

Licenciatura em Engenharia Informática

Relatório de Projeto

Gestor de Alojamentos

Muaiad Mhd Fahd Al Hadad

Oliveira do Hospital, fevereiro de 2024

Honestidade Intelectual

Eu, Muaiad Mhd Fahd Al Hadad, estudante n.º 2020130486 da Licenciatura em Engenharia informática, declaro que o relatório de estágio/trabalho de projeto intitulado gestão de alojamentos é original e que, ao longo da sua elaboração, não pratiquei plágio ou qualquer forma de falsificação de conteúdo. O trabalho de projeto resulta do meu próprio trabalho, sendo reconhecidas todas as fontes utilizadas por se encontrarem devidamente citadas no corpo do texto e identificadas na secção de referências bibliográficas. Assumo ter plena consciência de que a prática de plágio - utilização como sendo criação ou prestação sua de obras, ideias, afirmações, dados, imagens ou ilustrações de outra autoria, no todo em parte, sem o adequado reconhecimento explícito - constitui, no âmbito académico, grave falta ética e desonestidade intelectual, tendo como consequência a anulação do trabalho apresentado, para além de poder constituir crime de violação dos direitos de autor e infração disciplinar.

Mais declaro que tomei conhecimento integral do Código de Ética e Conduta do Instituto Politécnico de Coimbra e demais regulamentos aplicáveis e que foram respeitadas as orientações recebidas quanto à pseudonimização ou anonimização de dados pessoais ou organizacionais.

Oliveira do Hospital, 12 de fevereiro de 2024

Nome completo: Muaiad Mhd Fahd Al Hadad

Assinatura: (digital)

Agradecimentos

Gostaríamos de expressar a nossa sincera gratidão a todos os professores da Escola Superior de Tecnologia e Gestão de Oliveira do Hospital (ESTGOH) que contribuíram de maneira significativa para o nosso percurso académico. O seu conhecimento, orientação e apoio foram fundamentais para o nosso desenvolvimento pessoal e profissional.

Em particular, queremos estender os nossos agradecimentos especiais ao Prof. Luis Veloso, ao Prof. Marco Veloso e ao Prof. Nuno Gil. As suas instruções, conselhos e incentivo foram essenciais para o sucesso deste projeto e para a nossa formação como engenheiros de informática.

Agradecemos por compartilharem connosco o seu profundo conhecimento, paixão pela área e dedicação ao ensino. As suas contribuições não apenas nos ajudaram a adquirir habilidades técnicas, mas também nos inspiraram a alcançar os nossos objetivos com determinação e excelência.

Que este simples gesto de gratidão expresse a nossa profunda admiração e respeito por vocês, que continuemos a manter uma relação de aprendizado mútuo e crescimento ao longo das nossas trajetórias profissionais.

Com os nossos mais sinceros agradecimentos,

Muaiad Mhd Fahd Al Hadad.

Resumo

Este projeto desenvolveu um sistema de gestão de alojamento para a Escola Superior de Tecnologia e Gestão de Oliveira do Hospital (ESTGOH), com o objetivo de facilitar a busca, reserva e comunicação entre alunos e senhorios. Utilizando HTML, JavaScript, PHP e uma base de dados, a plataforma permite aos alunos procurar quartos disponíveis, comunicar-se com os senhorios e efetuar reservas eficientemente. A metodologia envolveu o desenvolvimento iterativo do sistema, integrando “feedback” dos utilizadores para garantir usabilidade e funcionalidade adequadas. Os resultados incluem uma plataforma intuitiva e responsiva, melhorando significativamente a experiência dos utilizadores na busca por alojamento. Conclusões relevantes destacam a importância de soluções tecnológicas para otimizar processos académicos e a necessidade contínua de adaptação às demandas dos alunos.

Palavras-chave

Alojamento, Gestão, Escola, Estudantes, Plataforma.

Abstract

This project developed a housing management system for the Oliveira do Hospital School of Technology and Management (ESTGOH), aiming to facilitate the search, reservation, and communication between students and landlords. Using HTML, JavaScript, PHP, and a database, the platform allows students to search for available rooms, communicate with landlords, and make reservations efficiently. The methodology involved iterative development of the system, integrating user feedback to ensure adequate usability and functionality. Results include an intuitive and responsive platform, significantly improving users' experience in housing search. Relevant conclusions highlight the importance of technological solutions to streamline academic processes and the ongoing need to adapt to students' demands.

Keywords

Housing, Management, School, Students, Platform.

Índice

1. Índice

Honestidade Intelectual	i
Agradecimentos	ii
Resumo	iii
Abstract	iv
Lista de Figuras	vii
Lista de Tabelas	ix
Lista de Acrónimos	x
2. Introdução.....	1
3. Estado da Arte	2
4. Objectivos e Metodologias	4
4.1. Ferramentas e Tecnologias	4
4.2. Planeamento	5
4.3. Diagramas.....	6
4.3.1. Fluxogramas.....	6
4.3.2. Caso de use	10
4.3.3. Classes UML	11
4.3.4. ER (Base Dados):	13
5. Trabalho Desenvolvido	13
5.1. Página inicial.....	13
5.1.1. Front-end	13
5.1.2. Back-end	20
5.1.3. Resultado	25
5.2. Detalhe	29
5.2.1. Front-end	29
5.2.2. Back-end	32
5.2.3. Resultado:	33
5.3. Login	36
5.3.1. Front-end	36
5.3.2. Back-end	38
5.3.3. Resultado	39
5.4. Registrar	40
5.4.1. Front-end	40
5.4.2. Back-end	41
5.4.3. Resultado	43
5.5. Email enviados.....	44
5.5.1. Back-end	44
5.5.2. Resultado	46
5.6. Validação código	47
5.6.1. Back-end	47
5.6.2. Resultado	49
5.7. Recuperar a Password.....	50
5.7.1. Back-end	50

5.7.2.	Resultado	51
5.8.	Gestor área pessoal.....	52
5.8.1.	Front-end	52
5.8.2.	Back-end	54
5.8.3.	Resultado	56
5.9.	Adicionar	57
5.9.1.	Back-end	57
5.10.	Profile	58
5.10.1.	Back-end	58
5.10.2.	Resultado	60
5.11.	Back-end remover utilizador.....	62
5.12.	Back-end consultar anúncio	63
5.13.	Back-end aprovar e reprovar anúncio.....	65
5.14.	Back-end mudar estado de um anúncio	66
5.15.	Senhorio área pessoal	68
5.15.1.	Back-end	68
5.15.2.	Resultado	69
5.16.	Editar anúncio	70
5.16.1.	Back-end	70
5.16.2.	Resultado	73
5.17.	Adicionar anúncio.....	76
5.17.1.	Front-end	76
5.17.2.	Back-end	79
5.17.3.	Resultado	82
5.18.	Requisitos Implementados.....	84
5.18.1.	REQUISITOS.....	84
5.18.2.	Requisitos planeados	87
5.18.3.	Principais motivos para eventuais desvios (Sprint 1)	87
5.18.4.	Principais motivos para eventuais desvios (Sprint 2)	87
5.18.5.	Lista dos diagramas.....	87
6.	Conclusões	88
6.1.	Forças	89
6.2.	Limitações	89
6.3.	Trabalho Futuro.....	90
7.	Referências.....	92
7.1.	Lista de Referência	92

Lista de Figuras

Figure 1-Fluxograma de Autenticação.....	6
Figure 2- Fluxograma de Filtragem de anúncios	7
Figure 3- Fluxograma de Gestor na aprovação/reprovação de anúncios	8
Figure 4- Fluxograma de Registro de senhorios/alunos na plataforma.....	9
Figure 5-diagrama caso de use	10
Figure 6-Classes UML.....	11
Figure 7-ER.....	13
Figure 8- cabeçalho da página com login e sem.....	14
Figure 9-Java Script de menu de utilizador	15
Figure 10- Lógica de mostra os utilizadores	16
Figure 11-filtragem	18
Figure 12-As ofertas.....	19
Figure 13-ecuperar os dados das propriedades imobiliárias para exibição na página inicial.....	20
Figure 14- exibir a página inicial do site	22
Figure 15-Filtragem	24
Figure 16-Cabeçalho com utilizador e sem	25
Figure 17-filtrar.....	26
Figure 18-resultado de filtragem.....	26
Figure 19-Página 1	27
Figure 20-Página 2	28
Figure 21-fotos de propriedade	29
Figure 22-específicos da propriedade	30
Figure 23-Map	31
Figure 24-recuperar os detalhes de uma casa específica	32
Figure 25-Resultado de detalhe	35
Figure 26-login form	36
Figure 27-toast.....	37
Figure 28-controller de login	38
Figure 29-login error.....	39
Figure 30-form registar.....	40
Figure 31-controler registar.....	41
Figure 32-registar sem error.....	43
Figure 33-registar com email existe	43
Figure 34-controlar email	45
Figure 35-email de verifique.....	46
Figure 36-controlar ativação de utilizador	47
Figure 37-controlar reenviar código de ativação	48
Figure 38-codigo mal formato	49
Figure 39-codigo invalido	49
Figure 40-controlar Forgot Password.....	50
Figure 41-Recuprar password.....	51
Figure 42-email com password	52
Figure 43-controlar de página principal de gestor	54
Figure 44-lista utilizadores	56

Figure 45-lista dos anúncios	56
Figure 46-lista dos pedidos.....	56
Figure 47-pesquisar na lista.....	57
Figure 48-controlar add gestor.....	57
Figure 49-dados do utilizador.....	58
Figure 50-update dados de utilizador	59
Figure 51-profile	60
Figure 52-editar porfile.....	61
Figure 53-remover utilizador	62
Figure 54-consultar anúncio	64
Figure 55-Aprovar e reprovar anúncio	65
Figure 56-mudar estado de um anúncio	66
Figure 57-controlar área pessoal senhorio	68
Figure 58-lista das propriedades	69
Figure 59-enviar dados de anúncio para página de edit	70
Figure 60-submeter editar.....	72
Figure 61-editar	75
Figure 62-Html adicionar anúncio	77
Figure 63-controlar adicionar anúncio	81
Figure 64-Adicionar anúncio.....	84

Lista de Tabelas

Tabela 1- Requisitos de Autenticação e Sessão	84
Tabela 2- Requisitos Geral.....	85
Tabela 3- Requisitos Gestor.....	85
Tabela 4- Requisitos Senhorio	86
Tabela 5-Requisitos Aluno	86
Tabela 6- Requisitos NÃO FUNCIONAIS.....	87
Tabela 7-Lista dos diagramas	88

Lista de Acrónimos

ER Modelo Entidade-Relacionamento

JS JavaScript

2. Introdução

A crescente demanda por soluções eficientes de gestão de alojamento, especialmente em ambientes académicos como a Escola Superior de Tecnologia e Gestão de Oliveira do Hospital (ESTGOH), tem sido evidente nos últimos anos. Nesse contexto, este projeto de final de curso em Engenharia de Informática surge para enfrentar esse desafio, desenvolvendo um sistema abrangente e intuitivo para gestão de alojamentos, direcionado especificamente para os alunos desta instituição.

O cerne deste projeto é facilitar o processo de busca, reserva e comunicação entre os estudantes e os senhorios, proporcionando uma solução centralizada e eficaz. Utilizando uma combinação de linguagens de programação “web”, como HTML, JavaScript e PHP, e integrando uma base de dados para armazenamento e recuperação de informações críticas, busca-se oferecer uma plataforma que atenda às necessidades específicas dos utilizadores.

A plataforma permitirá aos alunos buscar quartos disponíveis de acordo com critérios específicos, como localização, preço e comodidades, além de possibilitar a comunicação direta com os senhorios, facilitando negociações e esclarecimento de dúvidas antes da reserva. Garantindo a integridade dos dados e a disponibilidade dos quartos selecionados, evitados conflitos de reservas, o sistema visa proporcionar uma experiência fluida e transparente para os utilizadores.

Os senhorios terão acesso a um painel de controle dedicado para gerir as suas propriedades, atualizar informações e responder às consultas dos alunos. A segurança e a privacidade dos dados dos utilizadores serão prioridades essenciais durante todo o desenvolvimento do projeto, em conformidade com as regulamentações de proteção de dados.

Em resumo, este projeto oferecerá uma solução abrangente e eficiente para a gestão de alojamentos, promovendo uma experiência amigável e intuitiva por meio de uma “interface” “web” moderna e responsiva, atendendo às necessidades específicas dos alunos da ESTGOH.

3. Estado da Arte

Neste estado da arte, examinamos duas plataformas líderes no mercado de acomodações para estudantes: Uniplaces e Student at Home. Ambas as plataformas pretendem facilitar a busca e reserva de acomodações para estudantes universitários, embora possam ter diferenças significativas em termos de recursos, alcance geográfico e modelo de negócios. Abaixo está uma análise detalhada de cada plataforma:

Uniplaces:

Descrição: Uniplaces é uma plataforma “online” estabelecida que conecta estudantes universitários a acomodações em várias cidades ao redor do mundo. A plataforma tem crescido significativamente, tornando-se uma escolha popular entre estudantes internacionais em busca de moradia durante os estudos.

Características:

“Interface” amigável e intuitiva que permite aos utilizadores pesquisar e filtrar acomodações com base nas preferências.

Opções de pagamento seguras e processamento de reservas “online”.

Avaliações e comentários de propriedades por outros utilizadores para auxiliar na tomada de decisão.

Suporte ao cliente dedicado para resolver quaisquer problemas relacionados à reserva ou à estadia.

Student at Home:

Descrição: Student at Home é outra plataforma “online” que oferece acomodações para estudantes universitários. A plataforma pode ter um foco regional específico ou oferecer recursos adicionais para estudantes em comparação com outras plataformas.

Características:

Oferece uma variedade de opções de moradia para estudantes, desde quartos individuais até apartamentos completos.

Pode incluir recursos adicionais específicos para estudantes, como guias de vida estudantil, informações sobre universidades locais e eventos estudantis.

Opções de reserva e pagamento “online”, embora difiram em termos de funcionalidades específicas em comparação com outras plataformas.

Comparação:

Ambas as plataformas pretendem principal fornecer uma experiência de busca e reserva de acomodações simplificada para estudantes universitários. No entanto, existem algumas diferenças notáveis entre elas. Enquanto Uniplaces pode ter um alcance global mais amplo e uma base de utilizadores mais estabelecida, Student at Home pode

oferecer recursos adicionais específicos para estudantes, dependendo do mercado em que opera.

Conclusão:

Tanto Uniplaces quanto Student at Home desempenham um papel vital no mercado de acomodações para estudantes, oferecendo opções convenientes e confiáveis para estudantes universitários em busca de moradia durante os estudos. Com a contínua evolução do mercado e as demandas em constante mudança dos estudantes, essas plataformas continuarão a se adaptar e inovar para atender às necessidades em constante mudança dos utilizadores.

Este estado da arte fornece uma visão abrangente das duas plataformas, destacando características distintas e papel no mercado de acomodações para estudantes.

Ao examinar o estado da arte, podemos observar que há uma variedade de soluções disponíveis no mercado que abordam aspectos semelhantes aos propostos neste projeto. No entanto, a proposta deste projeto é oferecer uma solução específica e personalizada para atender às necessidades dos alunos da ESTGOH, integrando funcionalidades de busca, reserva e comunicação numa plataforma única e intuitiva.

4. Objectivos e Metodologias

Objetivos:

Desenvolver uma plataforma “web” abrangente para gestão de alojamentos, direcionada especificamente para os alunos da Escola Superior de Tecnologia e Gestão de Oliveira do Hospital (ESTGOH).

Facilitar o processo de busca, reserva e comunicação entre os alunos e os senhorios, proporcionando uma solução centralizada e eficaz para as necessidades de alojamento dos estudantes.

Integrar funcionalidades de busca avançada, reserva de quartos e comunicação direta entre alunos e senhorios, garantindo uma experiência fluida e transparente para os utilizadores.

Metodologias:

Levantamento de Requisitos: Realizar entrevistas cliente para identificar as necessidades e preferências relativamente ao sistema de gestão de alojamentos.

Projeto e Desenvolvimento: Utilizar metodologias ágeis, como Scrum, para o desenvolvimento iterativo da plataforma, com entregas incrementais e ‘feedback’ contínuo dos utilizadores.

Implementação: Utilizar uma combinação de linguagens de programação “web”, incluindo HTML, JavaScript e PHP, para desenvolver a “interface” do utilizador e a lógica de negócios da plataforma.

Integração de Banco de Dados: Implementar um banco de dados para armazenar e recuperar informações críticas, como detalhes dos quartos, reservas e dados dos utilizadores.

Testes e Validação: Realizar testes de usabilidade, funcionais e de segurança para garantir a qualidade e a robustez da plataforma antes do lançamento.

Implantação e Manutenção: Implementar a plataforma num ambiente de produção e fornecer suporte contínuo, atualizações e manutenção conforme necessário.

4.1. Ferramentas e Tecnologias

Para o desenvolvimento deste projeto, serão utilizadas as seguintes ferramentas e tecnologias:

Laravel: Um “framework” de desenvolvimento “web” em PHP, conhecido por sua elegância, eficiência e facilidade de uso. O Laravel oferece recursos poderosos para o desenvolvimento rápido de aplicações “web”, incluindo gestão de rotas, ORM (Object-Relational Mapping), autenticação de utilizadores e muito mais.

MySQL: Um sistema de gestão de banco de dados relacional amplamente utilizado, que oferece desempenho confiável, escalabilidade e suporte a uma variedade de recursos avançados. O MySQL será utilizado para armazenar e recuperar dados críticos do sistema, como detalhes dos quartos, informações de utilizadores e histórico de reservas.

HTML, CSS e JavaScript: as linguagens fundamentais para o desenvolvimento de “interfaces” “web”. O HTML será utilizado para estruturar o conteúdo da página, o CSS para estilização e o JavaScript para adicionar interatividade e dinamismo à “interface” do utilizador.

Git: Um sistema de controlo de versão distribuído, essencial para o trabalho colaborativo e a gestão de código-fonte durante o desenvolvimento do projeto. O Git permite rastrear alterações no código, colaboração entre membros da equipa e a reversão para versões anteriores, se necessário.

Essas ferramentas e tecnologias foram selecionadas com base na minha capacidade de atender aos requisitos do projeto e fornece uma base sólida para o desenvolvimento de uma plataforma robusta e eficiente de gestão de alojamentos. O uso do Laravel e MySQL, em particular, proporcionará um ambiente de desenvolvimento seguro e escalável, enquanto o uso de HTML, CSS, JavaScript garantirá uma experiência de utilizador moderna e intuitiva. O Git será utilizado para garantir o controlo de versão.

4.2. Planeamento

O planeamento do projeto será realizado conforme as seguintes etapas:

Definição de Requisitos: Durante esta fase, serão identificados e documentados os requisitos funcionais e não funcionais da plataforma de gestão de alojamentos. Isso incluirá a determinação das funcionalidades necessárias, como busca de quartos, reserva, comunicação entre alunos e senhorios, além de requisitos de segurança e privacidade de dados.

Análise e “Design”: após a definição dos requisitos, será realizada uma análise detalhada para identificar as melhores práticas de “design” e arquitetura para o sistema. Isso incluirá a criação de diagramas de fluxo de dados, diagramas de classes e outros artefactos de “design” para orientar o desenvolvimento.

Desenvolvimento Iterativo: O desenvolvimento será realizado iterativamente, com entregas incrementais ao longo do tempo. Isso permitirá a rápida prototipagem e validação das funcionalidades, bem como a incorporação contínua do “feedback” dos utilizadores.

Testes e Depuração: Após cada iteração de desenvolvimento, serão realizados testes rigorosos para garantir a qualidade e a estabilidade do sistema. Isso incluirá testes de unidade, integração e aceitação, bem como testes de segurança e desempenho.

Implantação e Lançamento: Uma vez que o desenvolvimento e os testes estejam concluídos, o sistema será implantado num ambiente de produção e lançado para os

utilizadores finais. Isso será acompanhado por atividades de monitoramento e suporte para garantir que o sistema funcione conforme o esperado.

O planeamento será conduzido de forma ágil e flexível, permitindo ajustes conforme necessário ao longo do ciclo de vida do projeto. Isso garantirá que a plataforma de gestão de alojamentos seja entregue no prazo e atenda às expectativas dos utilizadores finais.

4.3. Diagramas

4.3.1. Fluxogramas

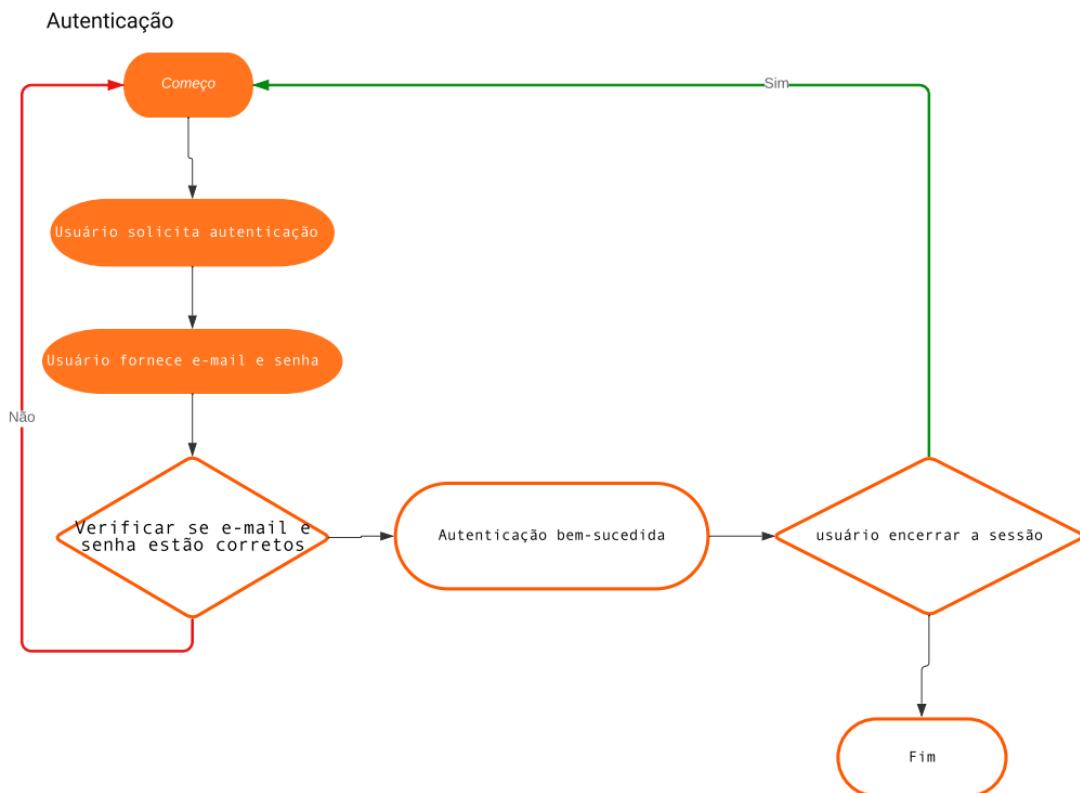


Figure 1-Fluxograma de Autenticação

Este fluxograma representa o processo de autenticação do utilizador quando ele tenta fazer “login” no sistema utilizando o seu e-mail e senha.

Início: O processo começa quando o utilizador solicita a autenticação, geralmente clicando num botão de “login”.

Utilizador fornece e-mail e senha: O utilizador digita o e-mail e senha nos campos apropriados.

Verificar se e-mail e senha estão corretos: O sistema verifica se o e-mail e a senha fornecidos correspondem a uma conta válida no banco de dados.

Autenticação bem-sucedida ou falha: se as credenciais estiverem corretas, o utilizador é autenticado com sucesso e tem acesso ao sistema. Caso contrário, a autenticação falha e o utilizador é notificado do erro.

Opção para o utilizador encerrar a sessão: após autenticado, o utilizador pode escolher encerrar a sessão, o que o desconecta do sistema.

Fim: O fluxo termina aqui.

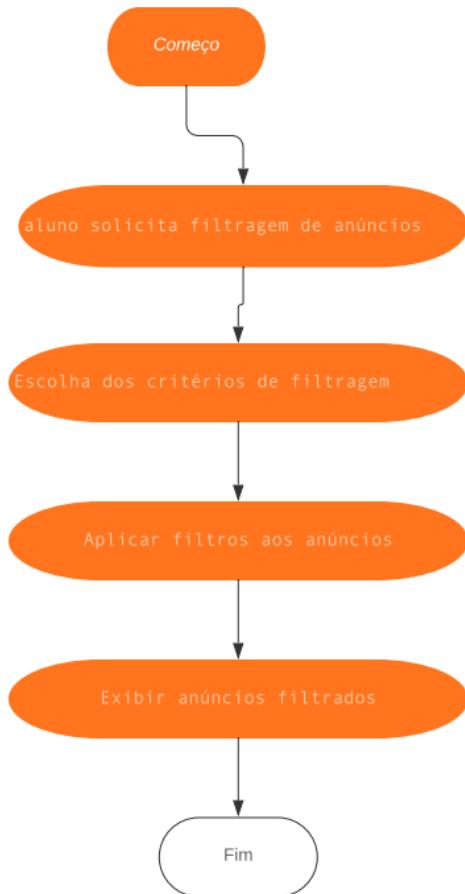


Figure 2- Fluxograma de Filtragem de anúncios

Este fluxograma representa o processo de filtragem de anúncios por gestores e alunos.

Início: O processo começa quando um gestor ou aluno solicita a filtragem de anúncios.

Escolha dos critérios de filtragem: O utilizador escolhe os critérios pelos quais deseja filtrar os anúncios, como distância à ESTGOH, valor da renda, número de quartos etc.

Aplicar filtros aos anúncios: O sistema aplica os critérios de filtragem selecionados aos anúncios disponíveis.

Exibir anúncios filtrados: O sistema exibe os anúncios que correspondem aos critérios de filtragem selecionados pelo utilizador.

Fim: O fluxo termina aqui.

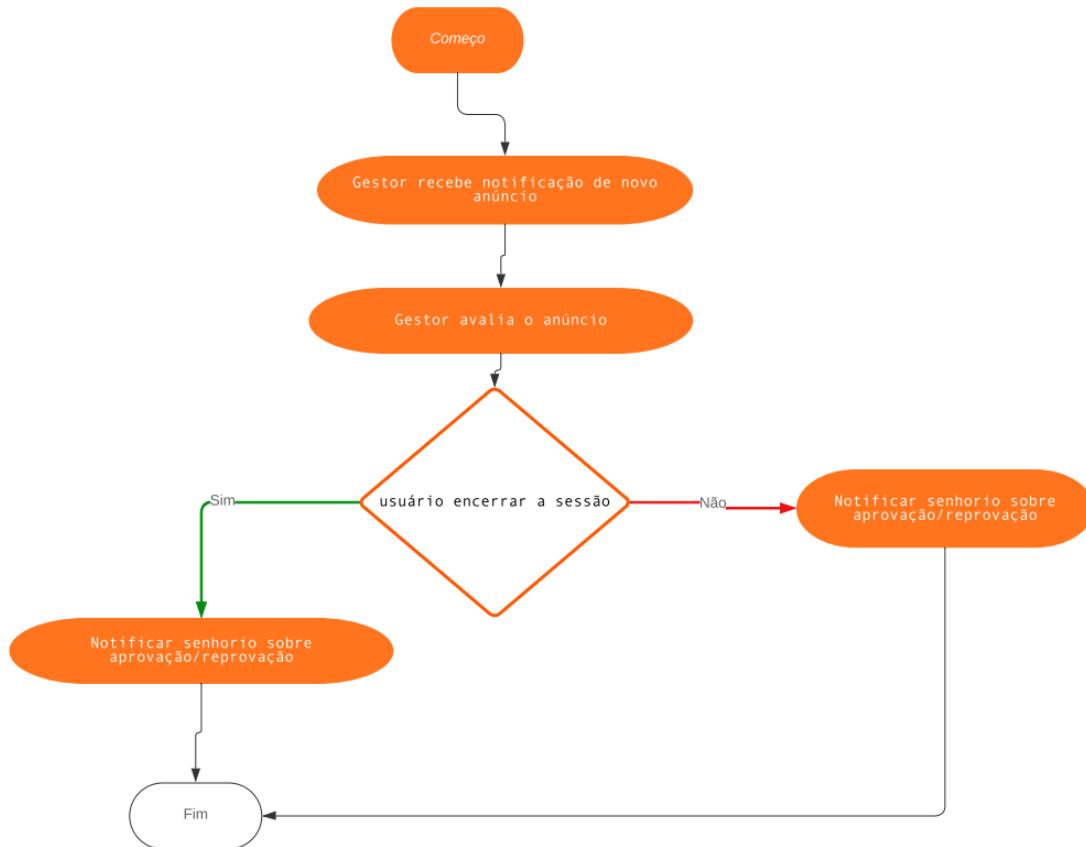


Figure 3- Fluxograma de Gestor na aprovação/reprovação de anúncios

Este fluxograma representa o processo de avaliação de anúncios por um gestor, com a opção de aprovar ou reprovavar os anúncios.

Início: O processo começa quando um novo anúncio é submetido à plataforma e um gestor recebe uma notificação sobre isso.

Gestor avalia o anúncio: O gestor revê o anúncio e decide se ele deve ser aprovado ou reprovado.

Anúncio aprovado ou reprovado: Se o gestor aprovar o anúncio, ele é publicado na plataforma. Se for reprovado, o anúncio não é publicado.

Notificar senhorio sobre aprovação/reprovação: O gestor notifica o senhorio sobre a decisão tomada em relação ao seu anúncio.

Fim: O fluxo termina aqui.

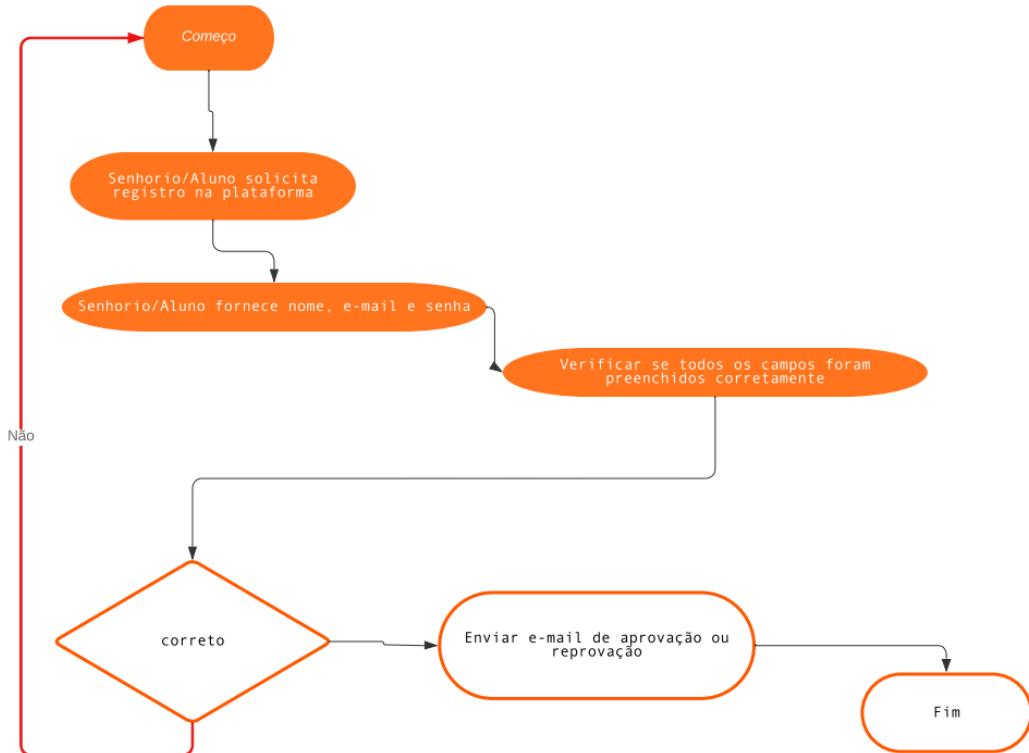


Figure 4- Fluxograma de Registro de senhorios/alunos na plataforma

Este fluxograma representa o processo de registo de um novo senhorio na plataforma.

Início: O processo começa quando um senhorio decide se registar na plataforma.

Senhorio fornece nome, e-mail e senha: O senhorio preenche os campos necessários para o registo, incluindo nome, e-mail e senha.

Verificar se todos os campos foram preenchidos corretamente: O sistema verifica se todas as informações necessárias foram fornecidas adequadamente pelo senhorio.

Registo bem-sucedido ou falho: se todas as informações estiverem corretas, o registo é bem-sucedido e o senhorio pode a cessar a plataforma. Caso contrário, o registo falha e o senhorio é notificado do erro.

Enviar e-mail de aprovação ou rejeição: O sistema envia um e-mail ao senhorio informando se o registo foi aprovado ou rejeitado.

Fim: O fluxo termina aqui.

4.3.2. Caso de use

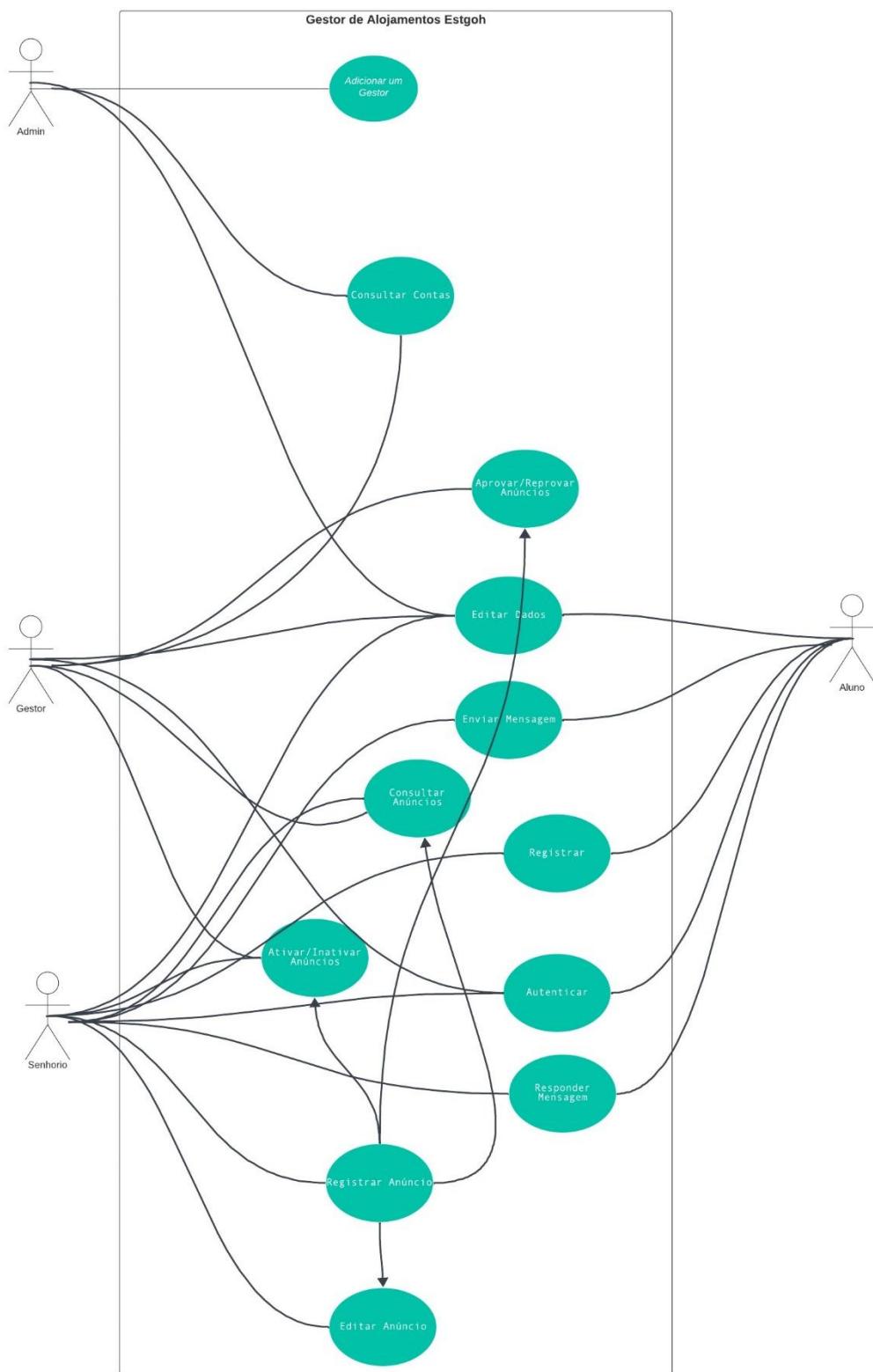


Figure 5-diagrama caso de use

Gestor: Este ator é responsável pela gestão dos anúncios dentro do sistema. Eles podem aprovar, reprovar, ativar, inativar anúncios, consultar anúncios, consultar contas, editar dados e responder mensagens.

Senhorio: Representa os proprietários de imóveis que desejam anunciar suas propriedades no sistema. Eles podem registrar anúncios, editar anúncios existentes, inativar e ativar anúncios, consultar anúncios, editar seus dados e responder mensagens.

Aluno: Este ator representa os usuários que estão procurando por acomodações. Eles têm permissão para consultar anúncios, enviar mensagens e responder mensagens.

Administrador: Este é o ator responsável pela gestão geral do sistema. Eles têm autoridade para realizar tarefas administrativas, como aprovar anúncios, consultar anúncios, consultar contas, editar dados de usuários, além de enviar e responder mensagens, se necessário.

4.3.3. Classes UML

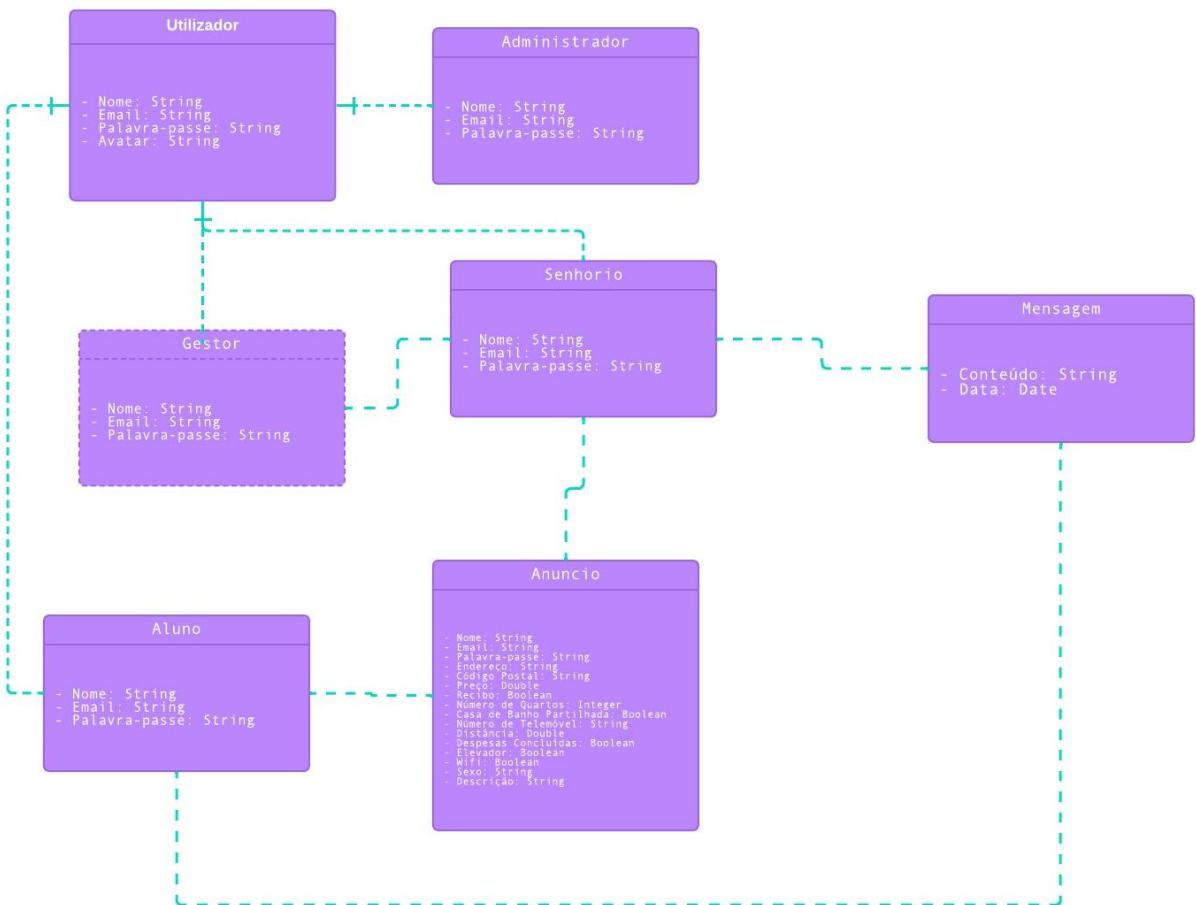


Figure 6-Classes UML

Classes:

Utilizador: Esta classe representa um utilizador genérico no sistema. Ela possui os atributos comuns a todos os tipos de utilizadores, como Nome, Email, Palavra-passe e Avatar.

Administrador: É uma subclasse de utilizador e representa um administrador do sistema. Herda os atributos de utilizador.

Gestor: Também é uma subclasse de utilizador e representa um gestor. Herda os atributos de utilizador.

Senhorio: Outra subclasse de utilizador, representa um senhorio. Além dos atributos herdados de utilizador, possui atributos adicionais relacionados aos detalhes da propriedade que ele oferece para aluguer.

Anúncio: Esta classe representa um anúncio de aluguer de propriedade. Possui atributos relacionados às características do anúncio, como Distância à ESTGOH, Valor da Renda, Número de Quartos e Sexo.

Aluno: Subclasse de Utilizador, representa um aluno. Herda os atributos de utilizador.

Associações:

Gestor gerencia Senhorio: Indica que um gestor pode gerir vários senhorios. Esta associação é representada por uma relação de composição ("1" para muitos).

Senhorio possui Anúncio: Indica que um senhorio pode possuir vários anúncios de propriedades para aluguer. Esta também é uma relação de composição.

Senhorio recebe Mensagem: Indica que um senhorio pode receber várias mensagens de alunos interessados nas propriedades que ele oferece para aluguer. Esta é uma associação simples ("1" para muitos).

Aluno envia Mensagem: Similar à relação anterior, indica que um aluno pode enviar várias mensagens para senhorios relativamente aos anúncios de propriedades. Também é uma associação simples.

Classes Adicionais:

Mensagem: Esta classe representa uma mensagem enviada entre um aluno e um senhorio relativamente a um anúncio de propriedade. Possui atributos como Conteúdo e Data para armazenar o conteúdo da mensagem e a data em que foi enviada.

Em resumo, o diagrama mostra a estrutura básica do sistema de aluguer de propriedades, incluindo os diferentes tipos de utilizadores, relacionamentos e as entidades relacionadas aos anúncios e mensagens.

4.3.4. ER (Base Dados):

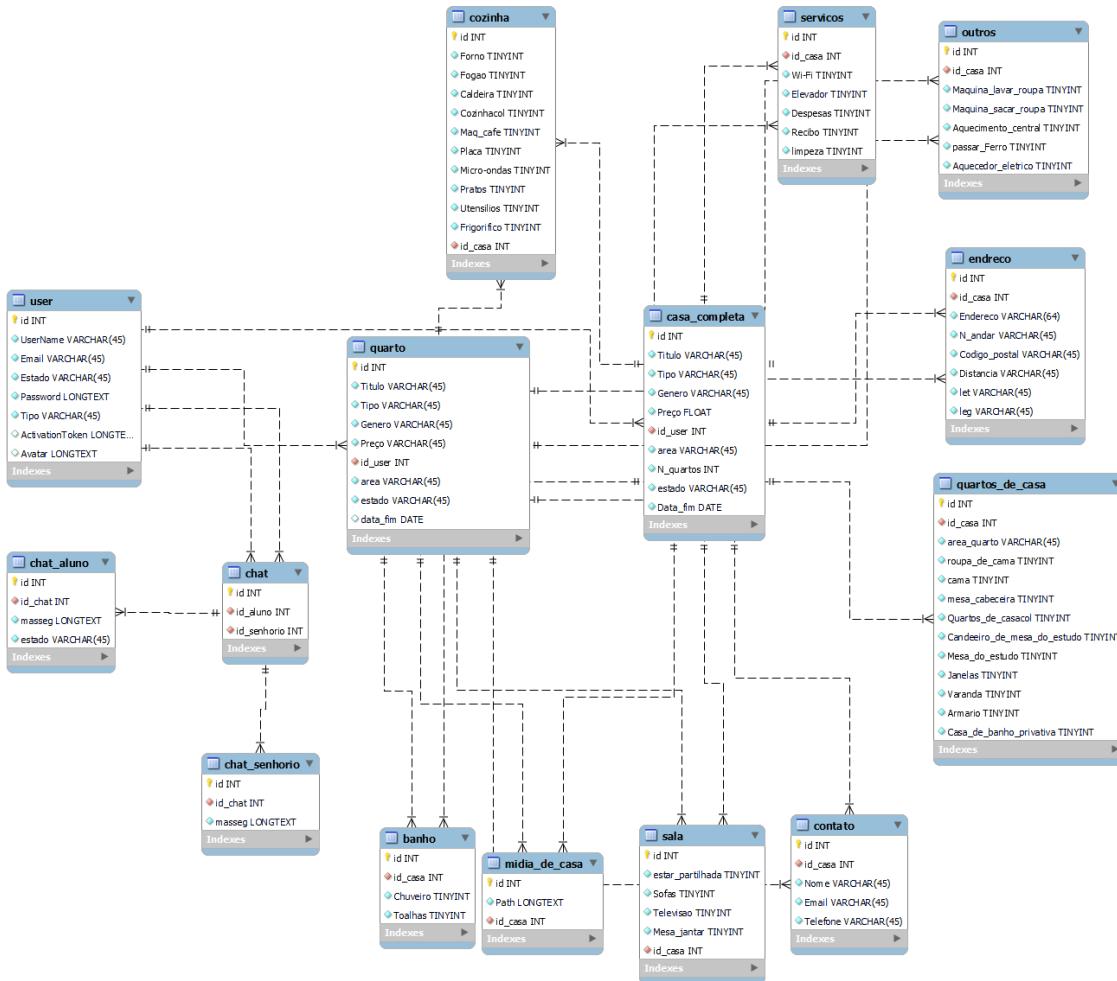


Figure 7-ER

5. Trabalho Desenvolvido

Foi utilizado e todas as páginas o modelo de ESTGOH.

5.1. Página inicial

5.1.1. Front-end

Aqui nesta parte vamos explicar como foi implementado o front-end de página inicial:

```

<div class="top-row">
    <div class="inner">
        <a class="main-logo small" href="https://www.estgoh.ipc.pt">
            
            <div style="..." class="title-sm-main">Escola Superior de <br>Tecnologia e Gestão<br>Politécnico de Coimbra</div>
        </a>
        @if($DadosUser!=null)
            <ul class="site-options">
                <li class="dropdown-menu-user-div-container">
                    @foreach($DadosUser as $user)
                        
                        <i class="fa fa-inverse">{{$user->Email}}</i>
                    @endforeach
                    <a class="dropdown-menu-user"><i class="fa fa-angle-down fa-inverse"></i></a>
                </li>
            </ul>
        @else
            <ul class="site-options">
                <li><a class="js-layerSearchToggle"><i class="fa fa-search fa-2x fa-inverse"></i></a></li>
                <li class="dropdown-menu-user-div-container">
                    <a class="dropdown-menu-user"><i class="fa fa-user fa-2x fa-inverse"></i></a>
                    <a class="dropdown-menu-user"><i class="fa fa-angle-down fa-inverse"></i></a>
                </li>
            </ul>
        @endif
    </div>

```

Figure 8- cabeçalho da página com login e sem

Divisões e classes CSS: A primeira div (`<div class="top-row">`) serve como contiver principal para a linha superior da barra de navegação. Dentro dela, há outra div (`<div class="inner">`) que pode ser usada para centralizar ou agrupar elementos.

Logo e Título: Dentro da `<div class="inner">`, há um link (`<a>`) que provavelmente representa o logotipo da página ou do site. Ele contém uma imagem (``) e um título (`<div class="title-sm-main">`) que descreve a instituição ou o propósito da página.

Condicional PHP (@if): O código PHP entre `@if` e `@else` verifica se há dados de utilizador disponíveis (`$DadosUser`). Dependendo disso, diferentes elementos HTML são renderizados. Se houver dados de utilizador, será provavelmente exibido um menu suspenso com informações do utilizador e um ícone de seta para baixo (`<i class="fa fa-angle-down fa-inverse"></i>`). Caso contrário, serão exibidos outros elementos, como um ícone de busca (`<i class="fa fa-search fa-2x fa-inverse"></i>`) e um ícone de utilizador (`<i class="fa fa-user fa-2x fa-inverse"></i>`).

Iteração sobre os dados do utilizador: se houver dados de utilizador disponíveis, o código PHP utiliza um loop `foreach` para iterar sobre esses dados. que ele exibe o avatar do utilizador e o endereço de e-mail (`{{$user->Email}}`).

Ícones FontAwesome: Os ícones utilizados (`<i class="fa fa-..."></i>`) são da biblioteca FontAwesome, que fornece uma ampla variedade de ícones vetoriais.

Geralmente, este trecho de código é responsável por exibir o cabeçalho da página, incluindo o logotipo, título, elementos de navegação e informações do utilizador, dependendo do contexto da autenticação do utilizador.

```

<script>
    document.addEventListener("DOMContentLoaded", function() {
        var userDropdown = document.querySelector('.dropdown-menu-user-div-container');
        var dropdownMenu = document.querySelector('.dropdown-menu-user-div');

        userDropdown.addEventListener('click', function(e) {
            e.stopPropagation();
            dropdownMenu.classList.toggle('show');
        });

        // Fechar o menu quando clicar fora dele
        document.addEventListener('click', function() {
            dropdownMenu.classList.remove('show');
        });

        // Evitar que o menu feche quando se clica dentro dele
        dropdownMenu.addEventListener('click', function(e) {
            e.stopPropagation();
        });
    });

</script>

```

Figure 9-Java Script de menu de utilizador

Evento 'DOMContentLoaded': esse evento é acionado quando todo o conteúdo do DOM foi carregado. Isso significa que o “script” será executado assim que a estrutura HTML estiver pronta para ser manipulada.

Seleção de Elementos: O código seleciona dois elementos do DOM:

userDropdown: Representa o contiver do menu suspenso.

dropdownMenu: Representa o próprio menu suspenso.

Evento de Clique no contêiner do menu suspenso: quando o contiver do menu suspenso é clicado, o evento de clique é acionado. Dentro desse evento, é chamado e.stopPropagation(), o que impede que o evento se propague para os elemento pai, e então é alternada a classe “show” no menu suspenso. Isso provavelmente mostra ou esconde o menu, dependendo do seu estado atual.

Evento de Clique fora do menu suspenso: esse evento é adicionado ao documento inteiro. Quando qualquer parte do documento é clicada, o menu suspenso é fechado. Isso é feito removendo a classe “show” do menu suspenso.

Evento de Clique no menu suspenso: esse evento é adicionado ao próprio menu suspenso. Quando o menu suspenso é clicado, o evento é acionado. Dentro dele, e.stopPropagation() é chamado para evitar que o evento se propague para os elemento pai. Isso garante que o menu não seja fechado quando se clica dentro dele.

Em resumo, esse “script” JavaScript adiciona funcionalidades de exibição e ocultação ao menu suspenso, permitindo que ele seja aberto ao clicar num determinado elemento e

fechado ao clicar fora dele, mas não fechado ao clicar no próprio menu suspenso. Isso proporciona uma melhor experiência de utilizador ao interagir com o menu.

```
<div class="dropdown-menu-user-div">
    @if( isset($DadosUser) && $DadosUser!=null)
        @foreach($DadosUser as $user)
            @if($user->Tipo=='senhorio')
                <ul>
                    <li><a href="/Senhorio/Profile"><i class="fa fa-indent"></i> Profile</a></li>
                    <li><a href="/Senhorio"><i class="fa fa-sign-out"></i>Área pessoal</a></li>
                    <li><a href="/logout"><i class="fa fa-sign-out"></i> Logout</a></li>
                </ul>
            @endif
            @if($user->Tipo=='gestor')
                <ul>
                    <li><a href="gestor/profile"><i class="fa fa-indent"></i> Profile</a></li>
                    <li><a href="/gestor"><i class="fa fa-sign-out"></i>Área pessoal</a></li>
                    <li><a href="/logout"><i class="fa fa-sign-out"></i> Logout</a></li>
                </ul>
            @endif
            @if($user->Tipo=='Aluno')
                <ul>
                    <li><a href="gestor/profile"><i class="fa fa-indent"></i> Profile</a></li>
                    <li><a href="/gestor"><i class="fa fa-sign-out"></i>Área pessoal</a></li>
                    <li><a href="/logout"><i class="fa fa-sign-out"></i> Logout</a></li>
                </ul>
            @endif
        @endforeach
    @else
        <ul>
            <li><a href="/login"><i class="fa fa-sign-in"></i> Login</a></li>
            <li><a href="/register"><i class="fa fa-user-plus"></i> Registrar</a></li>
        </ul>
    @endif
</div>
```

Figure 10- Lógica de mostra os utilizadores

Divisão e classe CSS: A div com a classe dropdown-menu-user-div representa provavelmente o contiver do menu suspenso.

Condicional PHP (@if): Assim como no trecho anterior, este código PHP verifica se existem dados de utilizador disponíveis (\$DadosUser) e se não estão vazios. Se os dados estiverem presentes, o código itera sobre esses dados usando um “loop” foreach.

Condições IF no “loop”: no “loop” foreach, o código verifica o tipo de utilizador (\$user->Tipo) e exibe opções de menu específicas com base nesse tipo. Se o utilizador for do tipo "senhorio", por exemplo, serão exibidas opções relacionadas a senhorios, como perfil, área pessoal e logout. O mesmo ocorre para os tipos de utilizador "gestor" e "Aluno".

Ligações de navegação: Para cada tipo de utilizador, são exibidas ligações de navegação () para diferentes partes do sitio “web”, como perfil, área pessoal e logout. Cada ligação é acompanhada por um ícone da biblioteca FontAwesome.

Condição ELSE: Se não houver dados de utilizador disponíveis, o código exibe opções de menu padrão para utilizadores não autenticados, como “login” e registo.

Geralmente, este trecho de código é responsável por exibir um menu suspenso dinâmico que se adapta ao tipo de utilizador autenticado, exibindo opções específicas para cada tipo de utilizador, ou opções padrões para utilizadores não autenticados.

```
<form action="/inicio" method="POST">
    @csrf
    <div class="center_home_1 clearfix">
        <div class="center_home_1i clearfix">
            <div class="col-sm-4">
                <div class="center_home_1i1 clearfix">
                    <input name="preco" placeholder="Preço" class="form-control" type="text">
                </div>
            </div>
            <div class="col-sm-4">
                <div class="center_home_1i1 clearfix">
                    <input name="destance" placeholder="Distância por Metros" class="form-control" type="text">
                </div>
            </div>
            <div class="col-sm-4">
                <div class="center_home_1i1 clearfix">
                    <select name="Tipo" class="form-control" name="property">
                        <option value="Tipo">Tipo de Propriedade</option>
                        <option value="Casa">Casa</option>
                        <option value="Apartamento">Apartamento</option>
                        <option value="Estúdio">Estúdio</option>
                    </select>
                </div>
            </div>
        </div>
    </div>

<div class="center_home_1i clearfix">
    <div class="col-sm-4">
        <div class="center_home_1i1 clearfix">
            <select name="n_quaros" class="form-control" name="beds" id="beds">
                <option value="n_quaros">Quartos</option>
                <option value="1">1</option>
                <option value="2">2</option>
                <option value="3">3</option>
                <option value="4">4</option>
                <option value="5">5</option>
                <option value="6">6</option>
                <option value="7">7</option>
                <option value="8">8</option>
                <option value="9">9</option>
                <option value="10">10</option>
            </select>
        </div>
    </div>
    <div class="col-sm-4">
        <div class="center_home_1i1 clearfix">
            <select name="Sexo" class="form-control">
                <option value="Sexo">Género</option>
                <option value="Masculino">Masculino</option>
                <option value="Feminino">Feminino</option>
                <option value="M/F">feminino e masculino</option>
            </select>
        </div>
    </div>
</div>
```

option

The `<option>` HTML ↗ element is used to define an item contained in a `<select>`, an `<optgroup>`, or a `<datalist>` element. As such, `<option>` can represent menu items in popups and other lists of items in an HTML document.

Supported by: Chrome, Chrome Android, Edge, Firefox, Opera 15, Safari 4, Safari iOS 3.2

By Mozilla Contributors ↗, CC BY-SA 2.5 ↗ 'option' on developer.mozilla.org ↗

```

<div class="col-sm-4">
    <div class="center_home_111 clearfix">
        <select name="Sexo" class="form-control" name="beds" id="beds">
            <option value="Sexo">Gênero</option>
            <option value="Masculino">Masculino</option>
            <option value="Feminino">Feminino</option>
            <option value="M/F">feminino e masculino</option>
        </select>
    </div>
</div>
<div class="col-sm-4">
    <div class="center_home_111 clearfix">
        <button style="..." class="mgt text-center"><a class="button_1 block mgt"><i class="fa fa-filter"></i> Filtrar</a></button>
    </div>
</div>

```

Figure 11-filtragem

Seção e classes CSS: O formulário está envolto numa seção com o ID "center" e as classes "center_list" e "clearfix". Essas classes são provavelmente usadas para estilização CSS e layout.

Container e linhas: Na seção, há um contêiner (`<div class="container">`) e uma linha (`<div class="row">`). Esses elementos podem ser usados para organizar o conteúdo numa grade, provavelmente usando um 'framework' CSS como Bootstrap.

Formulário: Na linha, há um formulário (`<form>`) com um atributo `action` definido para "/inicio" e um método `method` definido como "POST". Isso indica que os dados do formulário serão enviados para o URL "/inicio" usando o método POST.

Campos de entrada e seleção: O formulário contém vários campos de entrada e seleção:

Um campo de entrada para o preço (`<input name="preco" ...>`).

Um campo de entrada para a distância em metros (`<input name="destance" ...>`).

Um menu suspenso para selecionar o tipo de propriedade (`<select name="Tipo" ...>`).

Um menu suspenso para selecionar o número de quartos (`<select name="n_quaros" ...>`).

Um menu suspenso para selecionar o gênero (`<select name="Sexo" ...>`).

Botão de envio: Há um botão de envio no final do formulário, estilizado para ocupar 100% da largura do seu contêiner. Este botão será provavelmente usado para enviar o formulário após o utilizador ter selecionado os critérios de filtro.

Ícone do filtro: O botão de envio contém um ícone de filtro (um ícone FontAwesome representado pela classe "fa fa-filter").

Geralmente, este formulário permite que os utilizadores filtrem propriedades imobiliárias com base em critérios como preço, distância, tipo de propriedade, número de quartos e gênero. Ao submeter o formulário, os dados serão enviados para "/inicio" para processamento adicional.

```

<div class="popular_1 text-center clearfix">
    <div class="col-sm-12">
        <h1 class="mgt"> Lista de Propriedades</h1>
        <p>Encontre quarto com melhor classificação para você.</p>
    </div>

    @if(isset($DataQuartoAtive) && $DataQuartoAtive !=null)
    <div class="popular_2 clearfix">
        <h2 class="mgt">Renda por Quartos</h2>
        <hr>
        @foreach($DataQuartoAtive as $QuartoAtive)
            <div class="col-sm-5">
                <div class="popular_21 clearfix">
                    <div class="popular_211 clearfix">
                        <a href="/inicio/{{$QuartoAtive->idnow}}/quarto"></a> </div>
                    <div class="popular_212 clearfix">
                        <h5 class="mgt"><a href="/inicio/{{$QuartoAtive->idnow}}/quarto">{{ $QuartoAtive->Tipo }}</a></h5>
                    </div>
                </div>
                <div class="popular_213 clearfix">
                    <h5 class="mgt"><a href="/inicio/{{$QuartoAtive->idnow}}/quarto">{{ $QuartoAtive->Titulo }}</a></h5>
                    <h4 class="col_1">{{ $QuartoAtive->Preço }}€ / <span class="col_2">Month</span></h4>
                    <h6>@if($QuartoAtive->Genero =='Masculino')<i class="fa fa-male col_2"></i>@elseif($QuartoAtive->Genero =='Feminino')<i class="fa fa-female col_2"></i>@endif {{ $QuartoAtive->Genero }} <span< i class="fa fa-user col_2"></i><{{ $QuartoAtive->area }}> m</span></h6>
                    <h5 class="inline"><a class="button_1" href="/inicio/{{$QuartoAtive->idnow}}/quarto"><i class="fa fa-info-circle"></i> Detalhes</a></h5>
                    <h5 class="inline"><a class="button_1" href="detail.html"><i class="fa fa-heart" style="color: #ccc;"></i></a></h5>
                </div>
            </div>
        @endforeach
    </div>

```

Figure 12-As ofertas

Seção e classes CSS: A seção tem o “ID” “popular” e as classes “container” e “row”, que provavelmente são usadas para estilização CSS e layout.

Título e descrição: Na seção, há um elemento h1 com o título “Lista de Propriedades” e um parágrafo (

) com uma descrição “Encontre quarto com melhor classificação para você.”

Condicional PHP (@if): Há duas estruturas condicionais @if que verificam se há dados de propriedades ativas disponíveis. Se houver, esses dados são iterados usando um loop foreach. Para cada propriedade ativa, são exibidas informações como tipo, título, preço, género, área e botões para detalhes e favoritos.

Botões de paginação: No final da seção, há um bloco de código PHP que gera uma lista de botões de paginação usando um loop for. Cada botão de página é um link que aponta para uma página específica.

Geralmente, esta seção exibe uma lista de propriedades imobiliárias, incluindo quartos e casas, com informações como tipo, título, preço, género, área e botões para detalhes e favoritos. Também incluem botões de paginação para navegar entre as diferentes páginas de resultados.

5.1.2. Back-end

```

public function GetPageInicial($pagina = 1){
    $resultadosPorPagina = 8;
    $quantoporQuarto=4;
    $quantoporCasa=4;
    $indiceInicial = ($pagina - 1) * $resultadosPorPagina;
    if(session( key: 'ActivationToken')!=null){
        $token=session( key: 'ActivationToken');
        $user = DB::table( table: 'user')->where( column: 'ActivationToken', $token)->get();
        $totalQuartos = DB::table( table: 'quarto')->where( column: 'estado', operator: 'Ativo')->count();
        $totalCasas = DB::table( table: 'casa_completa')->where( column: 'estado', operator: 'Ativo')->count();
        if($totalCasas<4){
            $quantoporQuarto=8-$totalCasas;
        }
        if($totalQuartos<4){
            $quantoporCasa=8-$totalQuartos;
        }
        $QuartosAtive =DB::table( table: 'quarto')
            ->join( table: 'banho', first: 'banho.id_quarto', operator: '=', second: 'quarto.id')
            ->join( table: 'contato', first: 'contato.id_quarto', operator: '=', second: 'quarto.id')
            ->join( table: 'cozinha', first: 'cozinha.id_quarto', operator: '=', second: 'quarto.id')
            ->join( table: 'endereco', first: 'endereco.id_quarto', operator: '=', second: 'quarto.id')
            ->join( table: 'quartos_de_casa', first: 'quartos_de_casa.id_quarto', operator: '=', second: 'quarto.id')
            ->join( table: 'sala', first: 'sala.id_quarto', operator: '=', second: 'quarto.id')
            ->join( table: 'servicos', first: 'servicos.id_quarto', operator: '=', second: 'quarto.id')
            ->join( table: 'outros', first: 'outros.id_quarto', operator: '=', second: 'quarto.id')
            ->leftJoin( table: 'midia_de_casa', function ($join) {
                $join->on('midia_de_casa.id_quarto', '=', 'quarto.id')
                    ->whereRaw('midia_de_casa.id = (SELECT id FROM midia_de_casa WHERE id_quarto = quarto.id ORDER BY id ASC LIMIT 1)');
            })
            ->where( column: 'quarto.estado', operator: 'Ativo')
            ->select( columns: 'midia_de_casa.*','quarto.id as idnow','quarto.*', 'banho.*', 'contato.*', 'cozinha.*', 'endereco.*'
            , 'quartos_de_casa.*'
            , 'sala.*', 'servicos.*', 'outros.*')
            ->skip($indiceInicial)
    }
    ->take($quantoporQuarto)
    ->get();
    $CasaAtive =DB::table( table: 'casa_completa')
        ->join( table: 'banho', first: 'banho.id_casa', operator: '=', second: 'casa_completa.id')
        ->join( table: 'contato', first: 'contato.id_casa', operator: '=', second: 'casa_completa.id')
        ->join( table: 'cozinha', first: 'cozinha.id_casa', operator: '=', second: 'casa_completa.id')
        ->join( table: 'endereco', first: 'endereco.id_casa', operator: '=', second: 'casa_completa.id')
        ->join( table: 'sala', first: 'sala.id_casa', operator: '=', second: 'casa_completa.id')
        ->join( table: 'servicos', first: 'servicos.id_casa', operator: '=', second: 'casa_completa.id')
        ->join( table: 'outros', first: 'outros.id_casa', operator: '=', second: 'casa_completa.id')
        ->leftJoin( table: 'midia_de_casa', function ($join) {
            $join->on('midia_de_casa.id_casa', '=', 'casa_completa.id')
                ->whereRaw('midia_de_casa.id = (SELECT id FROM midia_de_casa WHERE id_casa = casa_completa.id ORDER BY id ASC LIMIT 1)');
        })
        ->where( column: 'casa_completa.estado', operator: 'Ativo')
        ->select( columns: 'midia_de_casa.*','casa_completa.id as idnow','casa_completa.*', 'banho.*', 'contato.*', 'cozinha.*', 'endereco.*'
        , 'sala.*', 'servicos.*', 'outros.*')
        ->skip($indiceInicial)
        ->take($quantoporCasa)
        ->get();

    $totalResultados = $totalQuartos + $totalCasas;
    $totalPaginas = ceil( num: $totalResultados / $resultadosPorPagina);
    return view( view: 'inicio_index', ['DataCasaAtive'=>$CasaAtive, 'DataQuartoAtive'=>$QuartosAtive, 'DadosUser'=>$user
    , 'totalPaginas'=>$totalPaginas,
    'pagina' => $pagina]);
}

```

Figure 13-ecuperar os dados das propriedades imobiliárias para exibição na página inicial

Este código PHP é responsável por recuperar os dados das propriedades imobiliárias para exibição na página inicial do site:

Métodos Utilizados:

GetPageInicial(\$pagina): Este método é responsável por recuperar os dados das propriedades imobiliárias para exibição na página inicial do site web. Ele recebe o número da página como parâmetro e executa consultas ao banco de dados para obter

os dados dos quartos ativos e das casas completas. Além disso, ele implementa a lógica de paginação para exibir um número específico de resultados por página.

Técnicas Utilizadas:

Consulta SQL Complexa: O código utiliza consultas SQL complexas que envolvem várias junções e subconsultas para recuperar dados das tabelas relacionadas. Isso permite recuperar informações detalhadas sobre os quartos e casas, incluindo características como banheiros, cozinhas, serviços, etc.

Paginação: A técnica de paginação é implementada para dividir os resultados em várias páginas, facilitando a navegação do utilizador. A lógica de paginação calcula o número total de páginas com base no número de resultados e no número desejado de resultados por página.

Lógicas Utilizadas:

Lógica de Recuperação de Dados: O método GetPageInicial utiliza uma lógica para recuperar apenas as propriedades imobiliárias ativas do banco de dados. Isso é feito por condições de filtragem nas consultas SQL, garantindo que apenas os quartos e casas ativos sejam exibidos na página inicial.

Lógica de Paginação: A lógica de paginação calcula o índice inicial para a consulta com base no número da página e no número de resultados por página. Além disso, ela determina o número total de páginas necessárias para exibir todos os resultados disponíveis.

Verificação de Sessão e Controle de Acesso: O código inclui uma lógica para verificar se há uma sessão ativa do utilizador. Se houver uma sessão ativa, determinados parâmetros de exibição podem ser ajustados com base nas informações do utilizador logado.

Considerações Finais:

Esses métodos, técnicas e lógicas são fundamentais para garantir que a página inicial do sítio web apresente de forma eficiente e organizada as propriedades imobiliárias disponíveis. A utilização de consultas SQL complexas, lógica de paginação e verificação de sessão contribuem para uma experiência do utilizador mais fluida e personalizada.

```

public function index($pagina = 1){
    $resultadosPorPagina = 8;
    $indiceInicial = ($pagina - 1) * 4;
    if(session('key') == ActivationToken){=}
    $token = session('key');
    $user = DB::table('user')->where('ActivationToken', $token)->get();
    $QuartosAtive = DB::table('quarto')
        ->join('banho', first: 'banho.id_quarto', operator: '=', second: 'quarto.id')
        ->join('contato', first: 'contato.id_quarto', operator: '=', second: 'quarto.id')
        ->join('cozinha', first: 'cozinha.id_quarto', operator: '=', second: 'quarto.id')
        ->join('endereco', first: 'endereco.id_quarto', operator: '=', second: 'quarto.id')
        ->join('quartos_de_casa', first: 'quartos_de_casa.id_quarto', operator: '=', second: 'quarto.id')
        ->join('sala', first: 'sala.id_quarto', operator: '=', second: 'quarto.id')
        ->join('servicos', first: 'servicos.id_quarto', operator: '=', second: 'quarto.id')
        ->join('outros', first: 'outros.id_quarto', operator: '=', second: 'quarto.id')
        ->leftJoin('midia_de_casa', function ($join) {
            $join->on('midia_de_casa.id_quarto', '=', 'quarto.id')
            ->whereRaw("midia_de_casa.id = (SELECT id FROM midia_de_casa WHERE id_quarto = quarto.id ORDER BY id ASC LIMIT 1)");
        })
        ->where('quarto.estado', operator: 'Ativo')
        ->select('midia_de_casa.*', 'quarto.id as idnow', 'quarto.*', 'banho.*', 'contato.*', 'cozinha.*', 'endereco.*',
            'quartos_de_casa.*',
            'sala.*', 'servicos.*', 'outros.*')->get();
    $CasaAtive = DB::table('casa_completa')
        ->join('banho', first: 'banho.id_casa', operator: '=', second: 'casa_completa.id')
        ->join('contato', first: 'contato.id_casa', operator: '=', second: 'casa_completa.id')
        ->join('cozinha', first: 'cozinha.id_casa', operator: '=', second: 'casa_completa.id')
        ->join('endereco', first: 'endereco.id_casa', operator: '=', second: 'casa_completa.id')
        ->join('sala', first: 'sala.id_casa', operator: '=', second: 'casa_completa.id')
        ->join('servicos', first: 'servicos.id_casa', operator: '=', second: 'casa_completa.id')
        ->join('outros', first: 'outros.id_casa', operator: '=', second: 'casa_completa.id')
        ->leftJoin('midia_de_casa', function ($join) {
            $join->on('midia_de_casa.id_casa', '=', 'casa_completa.id')
            ->whereRaw("midia_de_casa.id = (SELECT id FROM midia_de_casa WHERE id_casa = casa_completa.id ORDER BY id ASC LIMIT 1)");
        })
        ->where('casa_completa.estado', operator: 'Ativo')
        ->select('midia_de_casa.*', 'casa_completa.id as idnow', 'casa_completa.*', 'banho.*', 'contato.*', 'cozinha.*', 'endereco.*',
            'sala.*', 'servicos.*', 'outros.*')->get();
    $totalQuartos = $QuartosAtive->count();
    $totalCasas = $CasaAtive->count();
    $totalResultados = $totalQuartos + $totalCasas;
    $totalPaginas = ceil($totalResultados / $resultadosPorPagina);
    $quantoporQuarto = 4;
    $quantoporCasa = 4;
    if ($totalCasas < 4) {
        $quantoporQuarto = 8 - $totalCasas;
    }
    if ($totalQuartos < 4) {
        $quantoporCasa = 8 - $totalQuartos;
    }
    $QuartosPaginados = $QuartosAtive->skip(max(0, $indiceInicial))->take($quantoporQuarto);
    $CasasPaginadas = $CasaAtive->skip(max(0, $indiceInicial))->take($quantoporCasa);

    return view('inicio\index', [
        'DataCasaAtive' => $CasasPaginadas,
        'DataQuartoAtive' => $QuartosPaginados,
        'totalPaginas' => $totalPaginas,
        'DadosUser' => $user,
        'pagina' => $pagina
    ]);
}

```

Figure 14- exibir a página inicial do site

Métodos Utilizados:

index(\$pagina): Este método é responsável por exibir a página inicial do site, que inclui uma lista de propriedades imobiliárias disponíveis para locação. Ele recebe o número da página como parâmetro para implementar a paginação dos resultados.

Técnicas Utilizadas:

Consulta SQL Complexa: O código utiliza consultas SQL complexas que envolvem várias junções e subconsultas para recuperar dados detalhados sobre as propriedades imobiliárias. Isso inclui características como banheiros, cozinhas, salas, serviços, etc.

Paginação: A técnica de paginação é implementada para dividir os resultados em várias páginas, facilitando a navegação do utilizador. A lógica de paginação calcula o número total de páginas com base no número de resultados e no número desejado de resultados por página.

Lógicas Utilizadas:

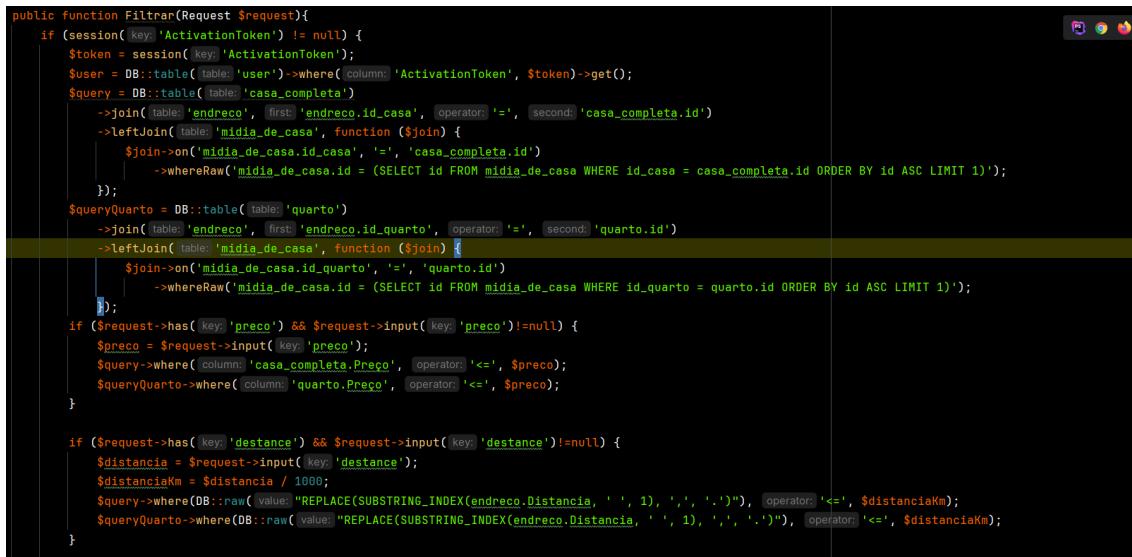
Lógica de Recuperação de Dados: O método index utiliza consultas ao banco de dados para recuperar dados sobre os quartos e casas disponíveis para locação. Ele verifica se há uma sessão ativa do utilizador para personalizar a exibição com base nas informações do utilizador logado.

Lógica de Paginação: A lógica de paginação calcula o índice inicial para a consulta com base no número da página e no número de resultados por página. Além disso, ela determina o número total de páginas necessárias para exibir todos os resultados disponíveis.

Ajuste de Quantidade de Resultados por Página: O código inclui uma lógica para ajustar a quantidade de resultados por página com base no número total de quartos e casas disponíveis. Isso garante que o número ideal de resultados seja exibido em cada página, considerando o número total de resultados disponíveis.

Considerações Finais:

Esses métodos, técnicas e lógicas são cruciais para garantir que a página inicial do sítio web apresente de forma eficiente e organizada as propriedades imobiliárias disponíveis para locação. A utilização de consultas SQL complexas, lógica de paginação e verificação de sessão contribuem para uma experiência do utilizador mais fluida e personalizada.



```

public function Filtrar(Request $request){
    if ($request->has('ActivationToken') != null) {
        $token = $request->input('ActivationToken');
        $user = DB::table('user')->where('column', 'ActivationToken', $token)->get();
        $query = DB::table('casa_completa')
            ->join('table: endreco', 'first: endreco.id_casa', 'operator: '=', 'second: casa_completa.id')
            ->leftJoin('table: midia_de_casa', function($join) {
                $join->on('midia_de_casa.id_casa', '=', 'casa_completa.id')
                    ->whereRaw('midia_de_casa.id = (SELECT id FROM midia_de_casa WHERE id_casa = casa_completa.id ORDER BY id ASC LIMIT 1)');
            });
        $queryQuarto = DB::table('quarto')
            ->join('table: endreco', 'first: endreco.id_quarto', 'operator: '=', 'second: quarto.id')
            ->leftJoin('table: midia_de_casa', function($join) {
                $join->on('midia_de_casa.id_quarto', '=', 'quarto.id')
                    ->whereRaw('midia_de_casa.id = (SELECT id FROM midia_de_casa WHERE id_quarto = quarto.id ORDER BY id ASC LIMIT 1)');
            });
        if ($request->has('preco') && $request->input('key: preco')!=null) {
            $preco = $request->input('key: preco');
            $query->where('column: casa_completa.Preco', 'operator: '<=', $preco);
            $queryQuarto->where('column: quarto.Preco', 'operator: '<=', $preco);
        }

        if ($request->has('key: distancia') && $request->input('key: distancia')!=null) {
            $distancia = $request->input('key: distancia');
            $distanciaKm = $distancia / 1000;
            $query->where(DB::raw("REPLACE(SUBSTRING_INDEX(endreco.Distancia, ' ', 1), ',', '.')"), 'operator: '<=', $distanciaKm);
            $queryQuarto->where(DB::raw("REPLACE(SUBSTRING_INDEX(endreco.Distancia, ' ', 1), ',', '.')"), 'operator: '<=', $distanciaKm);
        }
    }
}

```

```

if ($request->has( key: 'Tipo' ) && $request->input( key: 'Tipo')!="Tipo") {
    $tipo = $request->input( key: 'Tipo');
    $query->where( column: 'casa_completa.Tipo', operator: '=', $tipo);
    $queryQuarto->where( column: 'quarto.Tipo', operator: '=', $tipo);
}

if ($request->has( key: 'n_quartos' ) && $request->input( key: 'n_quartos')!="n_quartos") {
    $n_quartos = $request->input( key: 'n_quartos');
    $query->where( column: 'casa_completa.N_quartos', operator: '<=', $n_quartos);
    $queryQuarto->where( column: 'quarto.N_quartos', operator: '<=', $n_quartos);
}

if ($request->has( key: 'Sexo' ) && $request->input( key: 'Sexo')!="Sexo") {
    $sexo = $request->input( key: 'Sexo');
    $query->where( column: 'casa_completa.Genero', operator: '=', $sexo);
    $queryQuarto->where( column: 'quarto.Genero', operator: '=', $sexo);
}

$query->where( column: 'casa_completa.estado', operator: '=', value: 'Ativo');
$queryQuarto->where( column: 'quarto.estado', operator: '=', value: 'Ativo');

$resultadosQuarto = $queryQuarto->select( columns: 'midia_de_casa.*','quarto.*','quarto.id as idnow')->get();
$resultadosCasa = $query->select( columns: 'midia_de_casa.*','casa_completa.*','casa_completa.id as idnow')->get();

return view( view: 'inicio\index',[ 'DadosUser' => $user,'resultadosCasa' => $resultadosCasa,'resultadosQuarto'=>$resultadosQuarto]);

```

Figure 15-Filtragem

Esta função filtrar é responsável por filtrar propriedades imobiliárias e quartos disponíveis com base nos critérios especificados pelo utilizador e exibir os resultados na página inicial do sitio web. Aqui está uma explicação passo a passo do que o código faz:

Verificação do “Token” de Ativação de Sessão:

Primeiro, verifica se há um token de ativação de sessão armazenado. Isso provavelmente é usado para identificar o utilizador logado.

Consulta ao Banco de Dados:

A função realiza consultas ao banco de dados para recuperar os detalhes das propriedades e quartos disponíveis.

As consultas são elaboradas para selecionar informações relevantes das tabelas casa_completa (para propriedades) e quarto (para quartos).

As consultas utilizam junções internas (join) e externas (leftJoin) para garantir que todas as informações necessárias sejam obtidas.

Filtragem com base nos Parâmetros da Solicitação:

Os filtros são aplicados às consultas com base nos parâmetros fornecidos na solicitação (Request).

Os parâmetros considerados incluem preço máximo (preco), distância máxima (destance), tipo de propriedade (Tipo), número de quartos (n_quartos) e sexo (Sexo).

Se os parâmetros estiverem presentes na solicitação e não forem nulos, eles são aplicados às consultas para filtrar os resultados.

Condições de Estado:

As consultas também incluem condições para garantir que apenas propriedades e quartos com estado "Ativo" sejam retornados.

Recuperação de Resultados:

Os resultados das consultas são armazenados em variáveis, como \$resultadosCasa para propriedades e \$resultadosQuarto para quartos.

Retorno da Visualização:

Por fim, os resultados são passados para uma visualização chamada 'inicio\index' para serem exibidos na página inicial do sitio web.

Além dos resultados, também são passadas informações sobre o utilizador logado, obtidas anteriormente com base no "token" de ativação da sessão.

Essa função desempenha um papel fundamental na exibição dinâmica de propriedades e quartos com base nos critérios de pesquisa do utilizador, melhorando a experiência do utilizador ao procurar por acomodações.

5.1.3. Resultado

Figure 16-Cabeçalho com utilizador e sem

Preço

2000

Quartos

Feminino

Tipo de Propriedade

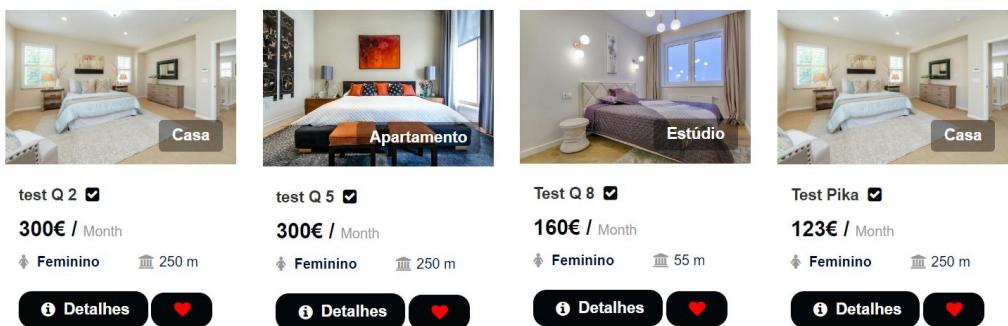
Filtrar

Figure 17-filtrar

Lista de Propriedades Filtrada

Encontre quarto com melhor classificação para você.

Renda por Quarto



Renda por Casa

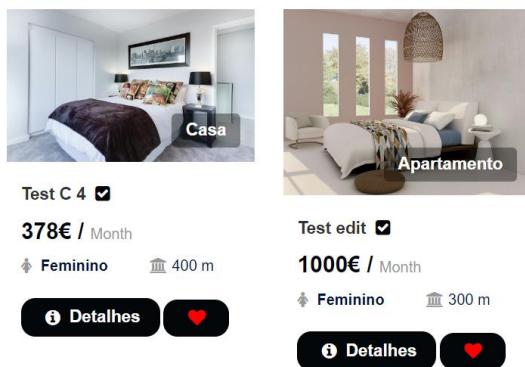


Figure 18-resultado de filtragem

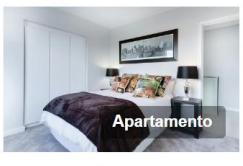
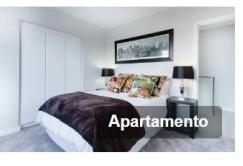
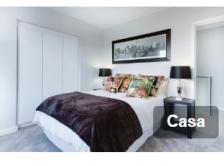
Lista de Propriedades

Encontre quarto com melhor classificação para você.

Renda por Quartos

 Casa test Q 1 <input checked="" type="checkbox"/> 200€ / Month  Masculino  250 m Detalhes 	 Casa test Q 2 <input checked="" type="checkbox"/> 300€ / Month  Feminino  250 m Detalhes 	 Casa test Q 3 <input checked="" type="checkbox"/> 200€ / Month  Masculino  250 m Detalhes 	 Apartamento test Q 4 <input checked="" type="checkbox"/> 300€ / Month  Masculino  5555 m Detalhes 
---	--	---	---

Renda por Casa

 Estúdio Test C 1 <input checked="" type="checkbox"/> 300€ / Month  M/F  250 m Detalhes 	 Apartamento Test C 2 <input checked="" type="checkbox"/> 600€ / Month  M/F  250 m Detalhes 	 Apartamento Test C 3 <input checked="" type="checkbox"/> 390€ / Month  Masculino  250 m Detalhes 	 Casa Test C 4 <input checked="" type="checkbox"/> 378€ / Month  Feminino  400 m Detalhes 
---	---	---	---

1 2 3

Figure 19-Página 1

Renda por Casa



Apartamento

Teste C 5

124€ / Month

 M/F  123 m

Detalhes 



Apartamento

Teste C 6

1500€ / Month

 M/F  250 m

Detalhes 



Apartamento

Teste C 7

300€ / Month

 M/F  250 m

Detalhes 



Casa

Teste C 8

200€ / Month

 Feminino  250 m

Detalhes 

1 **2** 3

Figure 20-Página 2

5.2. Detalhe

5.2.1. Front-end

Este trecho de código HTML é responsável por exibir detalhes de uma propriedade imobiliária na página de detalhes do imóvel.

Em resumo, esse trecho de código cria uma página detalhada para cada propriedade imobiliária, exibindo todas as informações relevantes e facilitando a visualização e o contato para possíveis interessados.

```
<section id="center" class="clearfix center_detail" style="...>
    <div class="center clearfix">
        <div id="carousel" class="carousel slide carousel-fade" style="...>
            <div class="carousel-inner" style="...>
                @if($PhotoCasa !=null)
                    @php $slideNo = 0 @endphp
                    @foreach($PhotoCasa as $photoquarto)
                        @if($slideNo == 0)
                            <div data-slide-no="{{ $slideNo }}" class="item carousel-item active">
                                
                                
                <span class="fa fa-angle-left kb_icons" aria-hidden="true"></span>
                <span class="sr-only">Previous</span>
            </a>
            <!-- Right-Button -->
            <a class="right carousel-control kb_control_right" href="#carousel" role="button" data-slide="next">
                <span class="fa fa-angle-right kb_icons" aria-hidden="true"></span>
                <span class="sr-only">Next</span>
            </a>
        </div>
    </div>
</section>
```

Figure 21-fotos de propriedade

O código começa com um elemento `<div>` que contém um carousel de imagens, permitindo que os utilizadores vejam várias fotos da propriedade. Isso é implementado usando a biblioteca Bootstrap Carousel.

As imagens são obtidas dinamicamente a partir do objeto `$PhotoCasa`, e para cada imagem, é criado um elemento `<div>` com a classe `item carousel-item`. A primeira imagem recebe a classe adicional `active` para ser exibida inicialmente.

```

<div class="row">
    @if($dadosCasaAtive !=null)
        @foreach($dadoscasaAtive as $dadoscasa)
            <div class="list_detail_1 clearfix">
                <div class="col-sm-16">
                    <div class="list_detail_11 clearfix">
                        <div class="list_detail_111 clearfix">
                            <h3 class="mgt">{{ $dadoscasa->Preco}}€ / <span class="span_1">Por mês</span> <span class="span_2">Aluguel</span></h3>
                            <p><i class="fa fa-map-marker"></i> {{ $dadoscasa->Endereco}}, {{ $dadoscasa->Codigo_postal}}</p>
                        </div>
                        <div class="list_detail_112 clearfix">
                            <h4 class="mgt">Informações da propriedade</h4>
                            <hr>
                            <div class="list_detail_112i clearfix">
                                <div class="col-sm-3 space_left">
                                    <h5 class="mgt">Distância</h5>
                                    <h6 class="col_1">{{ $dadoscasa->Distancia}}</h6>
                                </div>
                                <div class="col-sm-3 space_left">
                                    <h5 class="mgt">Sexo</h5>
                                    <h6 class="col_1">{{ $dadoscasa->Genero}}</h6>
                                </div>
                                <div class="col-sm-3 space_left">
                                    <h5 class="mgt">Área total do quarto</h5>
                                    <h6 class="col_1">{{ $dadoscasa->area}}</h6>
                                </div>
                                <div class="col-sm-3 space_left">
                                    <h5 class="mgt">Número de Quartos</h5>
                                    <h6 class="col_1">{{ $dadoscasa->N_quartos}}</h6>
                                </div>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>
    <div class="home_inner_i clearfix">
        <div class="col-sm-3 space_left">
            <h5 class="mgt"><i class="icon roupa-cama col_1">
                @if($dadosquartocasa->roupa_de_cama==0)
                    <i class="icon no"></i>
                @endif
            </i> roupa de cama </h5>
        </div>
        <div class="col-sm-3 space_left">
            <h5 class="mgt"><i class="icon cama col_1">
                @if($dadosquartocasa->cama==0)
                    <i class="icon no"></i>
                @endif
            </i> cama</h5>
        </div>
        <div class="col-sm-3 space_left">
            <h5 class="mgt"><i class="icon mesa-cama col_1">
                @if($dadosquartocasa->mesa_cabeceira==0)
                    <i class="icon no"></i>
                @endif
            </i> mesa cabeceira </h5>
        </div>
        <div class="col-sm-3 space_left">
            <h5 class="mgt"><i class="icon lambada col_1">
                @if($dadosquartocasa->Candeeiro_de_mesa_do_estudo==0)
                    <i class="icon no"></i>
                @endif
            </i> Candeeiro</h5>
        </div>
    </div>

```

Figure 22-específicos da propriedade

São exibidos os detalhes específicos da propriedade.

O loop @foreach percorre cada propriedade em \$DadosCasaAtive e exibe informações como preço, endereço e outras características, como distância, sexo, área total, número de quartos, etc.

Se houver informações de quartos específicos (\$DadosQuartoCasa), elas são exibidas em um loop adicional.

Descrição, Cozinha, Sala e Banheiro:

A descrição da propriedade é exibida num parágrafo.

As características da cozinha, sala e casa de banho são exibidas em listas, mostrando quais recursos estão disponíveis na propriedade.

```
<div class="list_detail_ll2 clearfix">
    <h4 class="mgt">Local</h4>
    <hr>
    <script>
        var letValue = "{{ $dadoscasa->let }}";
        letValue = letValue.replace(/\([ ]/g, '');
        var coordinates = letValue.split(',');
        var lat = parseFloat(coordinates[0]);
        var lng = parseFloat(coordinates[1]);
        function initMap() {
            const myLatLng = { lat: 40.35827117617252, lng: -7.8569972177657075 };
            const map = new google.maps.Map(document.getElementById("map"), {
                zoom: 15,
                center: myLatLng,
            });
            const icon_escola = "https://img.icons8.com/emoji/55/school-emoji.png";
            var estgoh = { lat: 40.361008858094266, lng: -7.861192556524104 };
            new google.maps.Marker({
                position: estgoh,
                icon:icon_escola,
                map: map,
                title: "Hello World!",
            });
            const icon_casa = "https://img.icons8.com/plasticine/75/order-delivered.png";
            var casa = { lat:lat,lng:lng };
            new google.maps.Marker({
                position: casa,
                icon:icon_casa,
                map: map,
                title: "Hello World!",
            });
        }
    </script>
    <div id="map" style="...></div>
```

Figure 23-Map

Um mapa do Google é exibido, mostrando a localização da propriedade e de pontos de interesse próximos, como escolas.

As coordenadas da propriedade são obtidas do objeto \$dadoscasa e usadas para marcar a sua localização no mapa.

5.2.2. Back-end

```

if ($session['ActivationToken'] != null) {
    $token = session('ActivationToken');
    $user = DB::table('user')->where('ActivationToken', $token)->get();
    $CasasAtive = DB::table('casa_completa')
        ->join('banho', 'first', 'banho.id_casa', 'operator', '=', 'second', 'casa_completa.id')
        ->join('table' 'contato', 'first', 'contato.id_casa', 'operator', '=', 'second', 'casa_completa.id')
        ->join('table' 'cozinha', 'first', 'cozinha.id_casa', 'operator', '=', 'second', 'casa_completa.id')
        ->join('table' 'endreco', 'first', 'endreco.id_casa', 'operator', '=', 'second', 'casa_completa.id')
        ->join('table' 'sala', 'first', 'sala.id_casa', 'operator', '=', 'second', 'casa_completa.id')
        ->join('table' 'servicos', 'first', 'servicos.id_casa', 'operator', '=', 'second', 'casa_completa.id')
        ->join('table' 'outros', 'first', 'outros.id_casa', 'operator', '=', 'second', 'casa_completa.id')
        ->join('table' 'user', 'first', 'user.id', 'operator', '=', 'second', 'casa_completa.id_user')
        ->where('column', 'casa_completa.id', '$id')
        ->select('columns', 'contato.Email as EmailQuarto', 'user.*', 'cozinha.Micro-ondas as Micro', 'casa_completa.id as idnow', 'casa_completa.*', 'banho.*', 'contato.*', 'sala.*', 'servicos.*', 'outros.*', 'servicos.Wi-Fi as wifi')->get();
    if (!$CasasAtive){
        abort(404, message: 'Bad Request.');
    }
    $PhotoCasa =DB::table('casa_completa')
        ->join('table' 'midia_de_casa', 'first', 'midia_de_casa.id_casa', 'operator', '=', 'second', 'casa_completa.id')
        ->where('column', 'casa_completa.id', '$id')
        ->select('columns', 'midia_de_casa.*')->get();
    $QuartosCasa = DB::table('casa_completa')
        ->join('table' 'quartos_de_casa', 'first', 'quartos_de_casa.id_casa', 'operator', '=', 'second', 'casa_completa.id')
        ->where('column', 'casa_completa.id', '$id')
        ->select('columns', 'quartos_de_casa.*')->get();
    return view('view_inicio/detalhe_casa', ['DadosUser' => $user, 'DadosCasaAtive' => $CasasAtive, 'PhotoCasa' => $PhotoCasa, 'DadosQuartoCasa' => $QuartosCasa]);
}

```

Figure 24-recuperar os detalhes de uma casa específica

Essa função GetPageDetalheCasa é responsável por recuperar os detalhes de uma casa específica com base no seu ID e exibir esses detalhes na página de detalhes da casa. Aqui está uma explicação do que o código faz:

Verificação do Token de Ativação de Sessão:

Verifica se há um token de ativação de sessão armazenado para identificar o utilizador logado.

Consulta ao Banco de Dados:

Realiza várias junções (join) entre diferentes tabelas para recuperar todos os detalhes relevantes da casa, como informações de contato, detalhes da cozinha, do casa de banho, da sala, serviços oferecidos, etc.

Esses detalhes são recuperados com base no ID da casa fornecido como parâmetro para a função.

Verificação da Existência da Casa:

Verifica se a casa com o ID fornecido existe no banco de dados. Se não existir, retorna um erro 404.

Recuperação de Fotos da Casa:

Recupera as fotos da casa com base no seu ID.

Recuperação dos Quartos da Casa:

Recupera os quartos disponíveis na casa com base no seu ID.

Retorno da Visualização:

Retorna uma visualização que exibe os detalhes da casa, as fotos e os quartos disponíveis.

Se um utilizador estiver logado, os detalhes do utilizador são passados para a visualização. Caso contrário, é passado null para indicar que nenhum utilizador está logado.

Essa função é crucial para exibir informações detalhadas sobre uma casa específica na página do sítio web, fornecendo aos utilizadores uma visão abrangente dos serviços e comodidades oferecidos pela propriedade.

5.2.3. Resultado:



The screenshot shows a real estate listing for a bedroom. At the top, there's a header with the text "sonhos Alojamentos Estgoh" and "Politécnico de Coimbra". Below the header, there's a large image of a bedroom. The room features a double bed with orange and blue pillows, a painting of red flowers above the bed, and two lamps on either side. To the left of the bed, there are two vertical artworks depicting birds. A window with blue curtains is on the right. Below the image, the price is listed as "1000€ / Por mês" with a "Aluguel" button. The address is "Rua Almeida Garrett, Oliveira do Hospital, 3400-080". Under the heading "Informações da propriedade", there's a table with the following data:

Distância	Sexo	Área total do quarto	Número de Quartos
2,0 km	Feminino	300	3

Informações da Quarto Nº 1

Área total do quarto

30

- | | | | |
|---|---|---|------------------------------------|
| <input type="checkbox"/> roupa de cama | <input type="checkbox"/> cama | <input type="checkbox"/> mesa cabeceira | <input type="checkbox"/> Candeeiro |
| <input checked="" type="checkbox"/> Mesa do estudo | <input checked="" type="checkbox"/> Janelas | <input checked="" type="checkbox"/> Varanda | <input type="checkbox"/> Armário |
| <input checked="" type="checkbox"/> Casa de banho privativa | | | |

Informações da Quarto Nº 2

Área total do quarto

20

- | | | | |
|---|---|--|---|
| <input checked="" type="checkbox"/> roupa de cama | <input checked="" type="checkbox"/> cama | <input checked="" type="checkbox"/> mesa cabeceira | <input checked="" type="checkbox"/> Candeeiro |
| <input checked="" type="checkbox"/> Mesa do estudo | <input checked="" type="checkbox"/> Janelas | <input checked="" type="checkbox"/> Varanda | <input checked="" type="checkbox"/> Armário |
| <input checked="" type="checkbox"/> Casa de banho privativa | | | |

Descrição

Test edit

Cozinha

- | | | | |
|---|---|---|---|
| <input checked="" type="checkbox"/> Forno | <input checked="" type="checkbox"/> Fogão | <input checked="" type="checkbox"/> Caldeira de água | <input checked="" type="checkbox"/> Máquina de café |
| <input checked="" type="checkbox"/> Placa | <input checked="" type="checkbox"/> Micro-ondas | <input checked="" type="checkbox"/> Pratos e talheres | <input checked="" type="checkbox"/> Utensílios de cozinha |
| <input checked="" type="checkbox"/> Frigorífico | | | |

Sala

- | | | | |
|--|---|---|---|
| <input checked="" type="checkbox"/> Área de estar partilhada | <input checked="" type="checkbox"/> Sofás | <input checked="" type="checkbox"/> Televisão | <input checked="" type="checkbox"/> Mesa de jantar com cadeiras |
|--|---|---|---|

Casa de banho

Chuveiro Toalhas

Outros

Máquina de lavar roupa Máquina de sacar roupa Aquecimento central máquina passar Ferro

Aquecedor elétrico

Serviços

Wi-Fi Elevador Despesas incluídas Recibo

limpeza

Local

Informação de contacto

Nour Pika
[Test edit](#) / aeadhadad19990@gmail.com

[Conectar](#)

Figure 25-Resultado de detalhe

5.3. Login

5.3.1. Front-end

```
<div class="container-main">
  <div class="container-login">
    <div class="form-box-login ">
      
      <form action="/login" method="POST">
        @csrf
        <div class="input-group-login ">
          <label for="Email">Email</label>
          <input type="text" id="Email" name="Email" required>
        </div>
        <div class="input-group-login ">
          <label for="password">Password</label>
          <input type="password" id="password" name="password" required>
          <a href="/Forgot"><i class="fa fa-send"></i> Forget Password</a>
        </div>
        <button type="submit">Login</button>
      </form>
    </div>
  </div>
</div>
```

Figure 26-login form

Esse trecho de código HTML representa um formulário de “login”. Ele permite que os utilizadores insiram o seu endereço eletrónico e senha para fazer “login”. Após preencher os campos, os utilizadores podem clicar no botão “Login” para enviar os dados do formulário para uma rota especificada, geralmente no backend do aplicativo “web”, onde o servidor processará as informações e autenticará o utilizador. A ligação “Forgot Password” permite que os utilizadores solicitem uma redefinição de senha caso tenham esquecido a senha.

```
<div style="...>
  @if(session('Error')!=null)
    <div class="toast">
      <div class="toast-content" style="...>
        <i class="fa fa-solid fa-remove error"></i>
        <div class="message">
          <span class="text text-1">Error</span>
          <span class="text text-2">{{session('Error')}}</span>
        </div>
      </div>
      <i class="fa fa-solid fa-close close"></i>
      <div class="progress error active"></div>
    </div>
  @endif
</div>
```

Figure 27-toast

Este código HTML exibe uma caixa de mensagem de erro de forma dinâmica na interface do utilizador. Ele verifica se há uma variável de sessão chamada 'Error' definida. Se essa variável estiver definida e não for nula, uma caixa de mensagem de erro será exibida, contendo um ícone de erro, uma mensagem de erro e uma barra de progresso para indicar que a mensagem está a ser exibida. O utilizador pode fechar a caixa de mensagem de erro clicando num ícone de error.

5.3.2. Back-end

```

▲ MuaiadHadad
public function login(Request $request)
{
    $Email = $request->input('key: 'Email');
    $password = $request->input('key: 'password');

    $user = DB::table('user')->where('column: 'Email', $Email)->first();

    if (!$user || !password_verify($password, $user->Password)) {
        return back()->with(['Error' => 'O e-mail ou palavra passa está incorreto!']);
    }

    if($user->Estado == "Ativo"){
        $activationToken = Str::random('length: 60');
        DB::table('user')->where('column: 'Email', $user->Email)->update([
            'ActivationToken' => $activationToken
        ]);

        session(['ActivationToken' => $activationToken]);
        session(['tipo_usuario' => $user->Tipo]);
        session(['Email' => $user->Email]);
        if ($user->Tipo == 'aluno') {
            return redirect('to: '/aluno');
        }
        if ($user->Tipo == 'senhorio') {
            return redirect('to: '/Senhorio');
        }
        if ($user->Tipo == 'gestor') {
            return redirect('to: '/gestor');
        }
    }elseif ($user->Estado == "Desativo"){
        return back()->withErrors(['message' => 'Conta não está ativa']);
    }elseif ($user->Estado != "Desativo" && $user->Estado != "Ativo"){
        return redirect('to: '/validation)->with('Email', $Email);
    }
}

```

Figure 28-controller de login

Este trecho de código PHP é responsável por lidar com o processo de login de utilizadores no sistema. Aqui está uma explicação passo a passo do que o código faz:

Obtenção dos Dados de Login:

O método login recebe uma solicitação (Request) que contém os campos de entrada Email e password, representando o e-mail e a senha inseridos pelo utilizador no formulário de login.

Verificação do Utilizador:

É feita uma consulta ao banco de dados para buscar um utilizador com o e-mail fornecido. Se não for encontrado nenhum utilizador correspondente, ou se a senha fornecida não coincidir com a senha armazenada no banco de dados (após ser

criptografada com password_hash), uma mensagem de erro é retornada, informando que o e-mail ou a senha estão incorretos.

Verificação do Estado do Utilizador:

Se o utilizador for encontrado no banco de dados e estiver com o estado "Ativo", um token de ativação é gerado aleatoriamente usando Str::random(60) e é atualizado no registro do utilizador no banco de dados. Em seguida, esse token é armazenado na sessão do utilizador.

Além disso, algumas informações do utilizador, como tipo de utilizador e e-mail, são armazenadas na sessão.

Dependendo do tipo de utilizador (aluno, senhorio ou gestor), o redireccionamento é feito para a página apropriada após o login.

Tratamento de Contas Desativadas:

Se o utilizador estiver com o estado "Desativo", é retornado um redireccionamento com uma mensagem informando que a conta não está ativa.

Tratamento de Outros Estados de Conta:

Se o estado do utilizador não for nem "Desativo" nem "Ativo", é feito um redireccionamento para uma página de validação com o e-mail do utilizador na sessão, possivelmente para ativação ou confirmação de e-mail.

Em resumo, esse código gerencia o processo de login de utilizadores, garantindo que apenas utilizadores com contas ativas acessem o sistema e redirecionando-os para a página apropriada após o login. Ele também lida com casos em que a conta está desativada ou requer validação adicional.

5.3.3. Resultado

The screenshot shows a login form for the Politécnico de Coimbra. The form includes fields for 'Email' (containing 'aeadhadad1999@gmail.com') and 'Password' (containing '.....'). Below the form is a 'Login' button. To the right of the form is a red error message box with a close button. The message reads: 'Error' followed by 'O e-mail ou palavra passa está incorreto!' (The email or password is incorrect!).

Figure 29-login error

5.4. Registrar

5.4.1. Front-end

```

<div class="container-main">
    <div class="container-login">
        <div class="form-box-login">
            
            <form action="/registrar" method="post" >
                @csrf
                <div class="input-group-login">
                    <label for="username">Username</label>
                    <input type="text" id="username" name="username" required>
                </div>
                <div class="input-group-login">
                    <label for="email">Email</label>
                    <input type="email" id="email" name="email" required>
                </div>
                <div class="input-group-login">
                    <label for="password">Password</label>
                    <input type="password" id="password" name="password" required>
                </div>
                <div class="input-group-login">
                    <label for="password_confirmation">Re-password</label>
                    <input type="password" id="password_confirmation" name="password_confirmation" required>
                </div>
                <div style="text-align: center">
                    <button type="submit">Registrar</button>
                </div>
            </form>
        </div>
    </div>
</div>

```

Figure 30-form registrar

Este trecho de código HTML representa um formulário de registo de utilizador. Aqui está uma explicação de cada parte do código:

Estrutura do Formulário:

O formulário está contido numa estrutura de divs com as classes container-main, container-login e form-box-login, que podem ser usadas para aplicar estilos de layout específicos.

Imagem:

Uma imagem é exibida acima do formulário. A URL da imagem é definida como <https://comum.rcaap.pt/retrieve/104938>.

Campos do Formulário:

Existem quatro campos no formulário:

Username: Um campo de texto onde os utilizadores podem inserir o seu nome de utilizador.

Email: Um campo de texto onde os utilizadores podem inserir o seu endereço de e-mail.

Password: Um campo de senha onde os utilizadores podem inserir a sua senha.

Repassword. Um campo de senha onde os utilizadores devem confirmar a sua senha, digitando-a novamente.

Botão de Envio:

Um botão "Registar" é fornecido para enviar o formulário. Quando clicado, o formulário será submetido para a rota /registrar usando o método HTTP POST.

Validação dos Campos:

Todos os campos do formulário são marcados com o atributo required, o que significa serem obrigatórios. Isso garante que os utilizadores não possam enviar o formulário sem preencher todos os campos obrigatórios.

Este formulário é usado para coletar informações de registo de utilizadores, como nome de utilizador, e-mail e senha. Uma vez preenchido e enviado, os dados do formulário podem ser processados por um servidor "web" para criar uma conta de utilizador no sistema.

5.4.2. Back-end

```
1 usage  ▾ MuaiadHadad
public function registrar(Request $request){
    // 200 -> foi registrado
    // 400 -> já existe o email
    $estado = strtoupper(Str::random( length: 6));

    $emailExists = DB::table( table: 'user')->where( column: 'Email', $request->email)->exists();

    if($emailExists) {
        return view( view: 'User\Register')->with('Code','400');
    }
    $tipoUtilizador = strpos($request->email, needle: '@alunos.estgoh.ipc.pt') !== false ? 'aluno' : 'senhorio';
    DB::table( table: 'user')->insert([
        'UserName' => $request->username,
        'Email' => $request->email,
        'Password' => bcrypt($request->password),
        'Estado' => $estado,
        'Tipo' => $tipoUtilizador,
        'Avatar'=> 'avatars/User-avatar.svg.png'
    ]);
    $this->enviarEmail( titulo: 'Codigo de ativação de sua conta', $request->username , $estado,$request->email);

    return view( view: 'User\Register')->with('Code', '200');
}
```

Figure 31-controler registrar

Este trecho de código PHP está relacionado ao processo de registro de utilizadores a sistema. Aqui está uma explicação das principais partes:

Geração de Estado Aleatório:

A variável `$estado` é definida como uma sequência aleatória de 6 caracteres em maiúsculas. Isso é feito usando o método `Str::random(6)` para gerar uma string aleatória e `strtoupper` para convertê-la para maiúsculas.

Verificação de E-mail Existente:

O código verifica se o e-mail fornecido já existe na tabela de utilizadores. Isso é feito consultando o banco de dados usando o método `where` e o método `exists`. Se o e-mail já existir, o código retorna uma visão específica com o código de status '400', indicando um erro.

Determinação do Tipo de Utilizador:

Com base no domínio do e-mail fornecido, o código determina se o usuário é um aluno ou um senhorio. Se o e-mail contiver `@alunos.estgoh.ipc.pt`, o utilizador será considerado um aluno; caso contrário, será considerado um senhorio.

Inserção no Banco de Dados:

Se o e-mail não existir na base de dados, os detalhes do utilizador (nome de utilizador, e-mail, senha, estado, tipo e avatar padrão) são inseridos na tabela de utilizadores usando o método `insert` do Laravel.

Envio de E-mail de Ativação:

Após a inserção bem-sucedida no banco de dados, um e-mail de ativação é enviado para o novo utilizador. O método `enviar Email` é chamado com os detalhes necessários para enviar o e-mail de ativação. Presumivelmente, esse método é responsável por enviar um e-mail para o utilizador com um código de ativação.

Retorno da Resposta:

Finalmente, uma visão específica é retornada com o código de status '200', indicando um registo bem-sucedido.

Esse código controla o processo de registo de utilizadores, garantindo que apenas e-mails únicos sejam registados, gerando um estado aleatório para a conta e enviando um e-mail de ativação para o utilizador recém-registado.

5.4.3. Resultado

Username
aeadhadad19990@gmail.com

Email

Password
.....

Re-password

Registrar

Success
Foi registado com sucesso!

Aluno
Se você um aluno tem de registrar com o vosso endereço eletrónico de escola "A2020...@alunos.estgoh.ipc.pt"!

Figure 32-registar sem erro

Username
aeadhadad19990@gmail.com

Email

Password
.....

Re-password

Error
Este e-mail já existe!

Aluno
Se você um aluno tem de registrar com o vosso endereço eletrónico de escola "A2020...@alunos.estgoh.ipc.pt"!

Figure 33-registar com email existe

5.5. Email enviados

5.5.1. Back-end

```
class EstadoEmail extends Mailable
{
    use Queueable, SerializesModels;

    2 usages
    public $subject;
    public $contentData;
    2 usages
    public $viewName;

    /**
     * Create a new message instance.
     *
     * @return void
     */
    4 usages  ↳ MuaiadHadad
    public function __construct($subject, $contentData, $viewName)
    {
        $this->subject = $subject;
        $this->contentData = $contentData;
        $this->viewName = $viewName;
    }

    /**
     * Build the message.
     *
     * @return $this
     */
    no usages  ↳ MuaiadHadad
    public function build()
    {
        return $this->subject($this->subject)
            ->view($this->viewName)
            ->with($this->contentData);
    }
}
```

```

/**
 * Build the message.
 *
 * @return $this
 */
no usages  ▲ MuaiadHadad
public function build()
{
    return $this->subject($this->subject)
        ->view($this->viewName)
        ->with($this->contentData);
}

```

Figure 34-controlar email

Esta classe EstadoEmail é uma Mailable do Laravel, o que significa que ela é responsável por construir e enviar e-mails no framework Laravel. Daremos uma olhada nas partes principais da classe:

Propriedades Públicas:

`$subject`: Armazena o assunto do e-mail.

`$contentData`: Armazena os dados de conteúdo do e-mail.

`$viewName`: Armazena o nome da visão (template) a ser usada para renderizar o e-mail.

Método Construtor:

`__construct($subject, $contentData, $viewName)`: Recebe o assunto do e-mail, os dados de conteúdo e o nome da visão como parâmetros e os atribui às propriedades correspondentes da classe.

Método `build()`:

Este método é chamado para construir o e-mail.

`subject($this->subject)`: Define o assunto do e-mail.

`view($this->viewName)`: Define a visão (template) a ser usada para renderizar o e-mail.

`with($this->contentData)`: Passa os dados de conteúdo para a visão do e-mail, permitindo que sejam acessados na visão como variáveis.

Essa classe é uma forma conveniente de estruturar e-mails no Laravel. Ela permite separar a lógica de construção e envio de e-mails do restante da aplicação, seguindo o conceito de responsabilidade única. Isso facilita a manutenção e o desenvolvimento de funcionalidades relacionadas ao envio de e-mails.

5.5.2. Resultado

BACK TO LIST		Delete	Source
	Alojamentos de Estgoh alojamentoestgoh@outlook.com	Date: 14-04-2024 20:30:23	
Subject: Código de ativação de sua conta			
Alojamento Estgoh			
<p style="text-align: center;">Verifique seu e-mail</p>  <p>Politécnico de Coimbra</p> <p>Olá Test, Obrigado por se juntar à nossa plataforma! Para ativar a sua conta, por favor utilize o código de ativação abaixo:</p> <p style="text-align: center;">Código de Ativação: 6YBCHZ</p> <p>Por favor, insira este código na página de ativação da sua conta. Se tiver alguma dúvida ou precisar de assistência, não hesite em contactar-nos.</p>			

Figure 35-email de verifique

5.6. Validação código

5.6.1. Back-end



```

1 usage  ~ MuaiadHadad
public function validateCode(Request $request){
    $codigo = $request->input('CODIGO');
    $email = $request->input('Email');

    $user = DB::table('user')->where('Estado', $codigo)->where('Email', $email)->first();

    if ($user) {
        DB::table('user')->where('Email', $email)->update(['Estado' => 'Ativo']);
        return redirect('/Login')->with('success', 'Código validado com sucesso! Você pode fazer login agora.');
    } else {
        return back()->with([
            'Error' => 'Código de validação inválido.',
            'Email' => $email
        ]);
    }
}

```

Figure 36-controlar ativação de utilizador

Este método validateCode é responsável por validar o código de ativação enviado por e-mail durante o processo de registo. Aqui está uma explicação detalhada do que ele faz:

Parâmetros do Método:

\$request: objeto que contém os dados da requisição HTTP, como os dados enviados pelo formulário.

Recuperação dos Dados:

\$codigo = \$request->input('CODIGO'): Obtém o código de ativação enviado pelo usuário.

\$email = \$request->input('Email'): Obtém o endereço de e-mail associado ao código de ativação.

Consulta ao Banco de Dados:

DB::table('user')->where('Estado', \$codigo)->where('Email', \$email)->first(): Consulta a tabela de usuários procurando por um registro que corresponda ao código de ativação e ao endereço de e-mail fornecido.

Validação do Resultado:

Se um utilizador for encontrado com o código de ativação e o endereço de e-mail correspondentes, isso significa que o código é válido.

Atualiza o estado do utilizador para "Ativo" no banco de dados.

Redireciona o utilizador para a página de login com uma mensagem de sucesso.

Se nenhum utilizador for encontrado com os dados fornecidos, significa que o código de ativação é inválido.

Retorna o utilizador à página anterior (provavelmente a página de registo) com uma mensagem de erro.

Esse método desempenha um papel importante no processo de registo de utilizadores, garantindo que apenas códigos de ativação válidos sejam aceites para ativar as contas dos utilizadores.

```
public function ReenviarCode(Request $request, $email){
    $estado = strtoupper(Str::random(6));
    $update=DB::table('user')->where('Email', $email)->update(['Estado' => $estado]);
    if($update){
        $this->enviarEmail( titulo: 'Codigo de ativação de sua conta', $estado,$email);
        return back()->with([
            'success'=> 'Código reenviado com sucesso para ' . $email,
            'Email' => $email
        ]);
    }else{
        return back()->with([
            'Error' => 'Não foi possível reenviar',
            'Email' => $email
        ]);
    }
}
```

Figure 37-controlar reenviar código de ativação

Este método ReenviarCode é responsável por reenviar o código de ativação para um determinado endereço de e-mail. Aqui está uma explicação passo a passo do que ele faz:

Parâmetros do Método:

\$request: objeto que contém os dados da requisição HTTP.

\$email: O endereço de e-mail para o qual o código de ativação será reenviado.

Geração de um Novo Código de Ativação:

\$estado = strtoupper(Str::random(6)): Gera um novo código de ativação aleatório com 6 caracteres em maiúsculas.

Atualização do Código de Ativação no Banco de Dados:

DB::table('user')->where('Email', \$email)->update(['Estado' => \$estado]): Atualiza o código de ativação para o novo valor gerado no banco de dados para o utilizador correspondente ao endereço de e-mail fornecido.

Envio do E-mail com o Novo Código:

Se a atualização for bem-sucedida:

Chama o método enviarEmail para enviar um e-mail com o novo código de ativação.

Retorna para a página anterior (provavelmente a página de registro) com uma mensagem de sucesso, indicando que o código foi reenviado com sucesso para o endereço de e-mail especificado.

Se a atualização falhar:

Retorna para a página anterior com uma mensagem de erro informando que não foi possível reenviar o código.

Este método permite que os utilizadores solicitem um novo código de ativação caso o anterior tenha sido perdido ou não recebido. Isso garante uma experiência suave durante o processo de registo.

5.6.2. Resultado

Código de validação

Enviar código novamente

! Faça corresponder o formato pedido.
O código deve conter 6 caracteres alfanuméricos.

Figure 38-código mal formato

Código de validação

Enviar código novamente

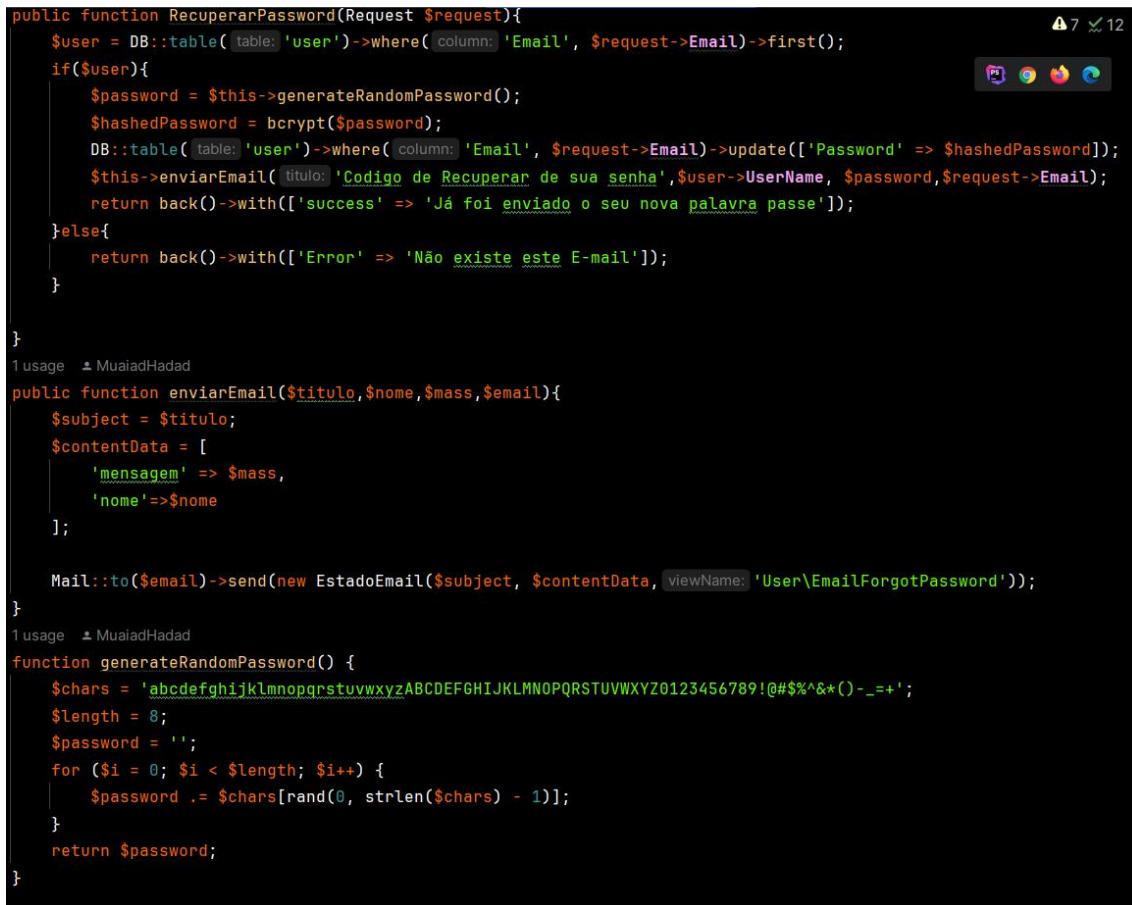
Validar

Error
Código de validação inválido.

Figure 39-código inválido

5.7. Recuperar a Password

5.7.1. Back-end



```

public function RecuperarPassword(Request $request){
    $user = DB::table('user')->where('Email', $request->Email)->first();
    if($user){
        $password = $this->generateRandomPassword();
        $hashedPassword = bcrypt($password);
        DB::table('user')->where('Email', $request->Email)->update(['Password' => $hashedPassword]);
        $this->enviarEmail(['titulo' => 'Codigo de Recuperar de sua senha', 'user' => $user, 'password' => $password, 'request' => $request]);
        return back()->with(['success' => 'Já foi enviado o seu nova palavra passe']);
    }else{
        return back()->with(['Error' => 'Não existe este E-mail']);
    }
}

1 usage  ± MuaiadHadad
public function enviarEmail($titulo, $nome, $mass, $email){
    $subject = $titulo;
    $contentData = [
        'mensagem' => $mass,
        'nome' => $nome
    ];
    Mail::to($email)->send(new EstadoEmail($subject, $contentData, ['viewName' => 'User\EmailForgotPassword']));
}
1 usage  ± MuaiadHadad
function generateRandomPassword() {
    $chars = 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789!@#$%^&*()_-+=';
    $length = 8;
    $password = '';
    for ($i = 0; $i < $length; $i++) {
        $password .= $chars[rand(0, strlen($chars) - 1)];
    }
    return $password;
}

```

Figure 40-controlar Forgot Password

Este código implementa a funcionalidade de recuperação de senha. Aqui está o que cada parte faz:

Método RecuperarPassword:

Primeiro, ele procura no banco de dados por um utilizador com o e-mail fornecido.

Se encontrar o utilizador:

Gera uma nova senha aleatória usando o método generateRandomPassword.

Hashes a nova senha.

Atualiza a senha do utilizador no banco de dados com a nova senha gerada.

Chama o método enviarEmail para enviar um e-mail ao utilizador com a nova senha.

Retorna para a página anterior (provavelmente a página de recuperação de senha) com uma mensagem de sucesso.

Se não encontrar o utilizador, retorna para a página anterior com uma mensagem de erro informando que o e-mail não existe.

Método enviarEmail:

Este método recebe um título, um nome, uma mensagem e um endereço de e-mail como argumentos.

Constrói o conteúdo do e-mail com o título, a mensagem e o nome.

Usa o objeto Mail para enviar o e-mail para o endereço fornecido, utilizando a classe EstadoEmail para renderizar o e-mail.

Método generateRandomPassword:

Este método gera uma senha aleatória de 8 caracteres, utilizando letras maiúsculas, minúsculas, números e alguns caracteres especiais.

Geralmente, esse código permite que os utilizadores solicitem a recuperação de senha e recebam um e-mail com uma nova senha temporária, permitindo que eles cessem a sua conta e posteriormente a alterem para uma senha da sua escolha.

5.7.2. Resultado



Figure 41-Recuperar password

Código de Recuperar de sua senha [Caixa de entrada](#)

 Alojamentos de Estgoh
para mim ▾

Recuperar a sua palavra passe

 **Politécnico
de Coimbra** [!\[\]\(27d57349fd95fd6b6bc001c630632c0f_img.jpg\)](#)

Caro João pinto,

Espero que esta mensagem o encontre bem.

Detectamos recentemente atividades suspeitas na sua conta, por motivos de segurança, decidimos redefinir a sua palavra-passe. Por favor, utilize a nova palavra-passe abaixo para aceder à sua conta:

Nova Palavra-passe: ^QWXC6#K

Pedimos desculpa pelo inconveniente, mas a segurança da sua conta é a nossa principal prioridade. Recomendamos que altere a sua palavra-passe periodicamente e evite partilhá-la com terceiros.

Se precisar de qualquer assistência adicional, não hesite em contactar-nos.

Cumprimentos,
Alojamento Estgoh

Figure 42-email com password

5.8. Gestor área pessoal

5.8.1. Front-end

interface de utilizador para um sistema de gestão de utilizadores, anúncios e pedidos. Aqui está uma visão geral do que ele faz:

Toast Messages:

As mensagens de toast são exibidas para fornecer feedback ao utilizador sobre operações realizadas, como erros ou sucesso.

As mensagens de erro e sucesso são exibidas em elementos <div> posicionados à direita da tela.

A toast messages contêm ícones para indicar o tipo de mensagem (erro ou sucesso), bem como o texto da mensagem.

Seleção de Lista:

Uma seção de seleção de lista permite que o utilizador escolha entre diferentes tipos de listas, como lista de utilizadores, lista de anúncios ou lista de pedidos.

A seleção é feita via um menu suspenso.

Listas:

Existem três seções de listas representadas por <section> com classes específicas (module--list).

Cada seção de lista corresponde a um tipo específico de lista: lista de utilizadores, lista de anúncios ou lista de pedidos.

Cada seção de lista contém uma tabela (<table>) que exibe os dados relevantes.

Para cada tipo de lista, as colunas da tabela exibem informações como ID, nome, endereço, proprietário, data de expiração, preço, estado e opções de ação (por exemplo, remover, ativar/desativar, consultar).

As linhas da tabela são preenchidas com os dados obtidos da variável \$utilizadores, \$Casa, \$Quartos, \$DataQuartoPending e \$DataCasaPending, dependendo do tipo de lista sendo exibida.

Para cada item na lista, são fornecidas opções de ação, como ativar/desativar um utilizador ou anúncio, remover um utilizador ou anúncio, ou consultar detalhes adicionais.

Funcionalidade Interativa:

As seções de lista e as mensagens de toast ter funcionalidades interativas ativadas em resposta a ações do utilizador, como selecionar uma lista específica, pesquisar itens ou executar ações nas listas (como ativar/desativar ou remover utilizadores/anúncios).

Geralmente, “interface” de utilizador funcional para um sistema de gestão administrativa. Ele fornece uma maneira intuitiva para os administradores interagirem com e gerenciarem utilizadores, anúncios e pedidos no sistema.

5.8.2. Back-end

```

2 usages ▲ MuaiadHadad
public function GetPageGestor(){
    if(session( key: 'tipo_usuario')=="gestor"){
        $token=session( key: 'ActivationToken');
        $user = DB::table( table: 'user')->where( column: 'ActivationToken', $token)->get();
        $utilizadores = DB::table( table: 'user')->get();
        $QuartosPending =DB::table( table: 'quarto')
            ->join( table: 'banho', first: 'banho.id_quarto', operator: '=', second: 'quarto.id')
            ->join( table: 'contato', first: 'contato.id_quarto', operator: '=', second: 'quarto.id')
            ->join( table: 'cozinha', first: 'cozinha.id_quarto', operator: '=', second: 'quarto.id')
            ->join( table: 'endereco', first: 'endereco.id_quarto', operator: '=', second: 'quarto.id')
            ->join( table: 'quartos_de_casa', first: 'quartos_de_casa.id_quarto', operator: '=', second: 'quarto.id')
            ->join( table: 'sala', first: 'sala.id_quarto', operator: '=', second: 'quarto.id')
            ->join( table: 'servicos', first: 'servicos.id_quarto', operator: '=', second: 'quarto.id')
            ->join( table: 'outros', first: 'outros.id_quarto', operator: '=', second: 'quarto.id')
            ->join( table: 'user', first: 'user.id', operator: '=', second: 'quarto.id_user')
            ->where( column: 'quarto.estado', operator: 'pending')
            ->select( columns: 'user.*', 'quarto.id as idnow', 'quarto.*', 'banho.*', 'contato.*', 'cozinha.*', 'endereco.*'
                , 'quartos_de_casa.*'
                , 'sala.*', 'servicos.*', 'outros.*')->get();
        $CasaPending =DB::table( table: 'casa_completa')
            ->join( table: 'banho', first: 'banho.id_casa', operator: '=', second: 'casa_completa.id')
            ->join( table: 'contato', first: 'contato.id_casa', operator: '=', second: 'casa_completa.id')
            ->join( table: 'cozinha', first: 'cozinha.id_casa', operator: '=', second: 'casa_completa.id')
            ->join( table: 'endereco', first: 'endereco.id_casa', operator: '=', second: 'casa_completa.id')
            ->join( table: 'sala', first: 'sala.id_casa', operator: '=', second: 'casa_completa.id')
            ->join( table: 'servicos', first: 'servicos.id_casa', operator: '=', second: 'casa_completa.id')
            ->join( table: 'outros', first: 'outros.id_casa', operator: '=', second: 'casa_completa.id')
            ->join( table: 'user', first: 'user.id', operator: '=', second: 'casa_completa.id_user')
            ->where( column: 'casa_completa.estado', operator: 'pending')
            ->select( columns: 'user.*', 'casa_completa.id as idnow', 'casa_completa.*', 'banho.*', 'contato.*', 'cozinha.*', 'endereco.*'
                , 'sala.*', 'servicos.*', 'outros.*')->get();
        $Quartos =DB::table( table: 'quarto')
            ->join( table: 'banho', first: 'banho.id_quarto', operator: '=', second: 'quarto.id')
            ->join( table: 'contato', first: 'contato.id_quarto', operator: '=', second: 'quarto.id')
            ->join( table: 'cozinha', first: 'cozinha.id_quarto', operator: '=', second: 'quarto.id')
            ->join( table: 'endereco', first: 'endereco.id_quarto', operator: '=', second: 'quarto.id')
            ->join( table: 'quartos_de_casa', first: 'quartos_de_casa.id_quarto', operator: '=', second: 'quarto.id')
            ->join( table: 'sala', first: 'sala.id_quarto', operator: '=', second: 'quarto.id')
            ->join( table: 'servicos', first: 'servicos.id_quarto', operator: '=', second: 'quarto.id')
            ->join( table: 'outros', first: 'outros.id_quarto', operator: '=', second: 'quarto.id')
            ->join( table: 'user', first: 'user.id', operator: '=', second: 'quarto.id_user')
            ->select( columns: 'user.*', 'quarto.id as idnow', 'quarto.*', 'banho.*', 'contato.*', 'cozinha.*', 'endereco.*'
                , 'quartos_de_casa.*'
                , 'sala.*', 'servicos.*', 'outros.*')->get();
        $Casa =DB::table( table: 'casa_completa')
            ->join( table: 'banho', first: 'banho.id_casa', operator: '=', second: 'casa_completa.id')
            ->join( table: 'contato', first: 'contato.id_casa', operator: '=', second: 'casa_completa.id')
            ->join( table: 'cozinha', first: 'cozinha.id_casa', operator: '=', second: 'casa_completa.id')
            ->join( table: 'endereco', first: 'endereco.id_casa', operator: '=', second: 'casa_completa.id')
            ->join( table: 'sala', first: 'sala.id_casa', operator: '=', second: 'casa_completa.id')
            ->join( table: 'servicos', first: 'servicos.id_casa', operator: '=', second: 'casa_completa.id')
            ->join( table: 'outros', first: 'outros.id_casa', operator: '=', second: 'casa_completa.id')
            ->join( table: 'user', first: 'user.id', operator: '=', second: 'casa_completa.id_user')
            ->select( columns: 'user.*', 'casa_completa.id as idnow', 'casa_completa.*', 'banho.*', 'contato.*', 'cozinha.*', 'endereco.*'
                , 'sala.*', 'servicos.*', 'outros.*')->get();
        return view( view: 'Page_Gestor\Gestor', [ 'utilizadores' => $utilizadores, 'Data'=>$user, 'DataCasaPending'=>$CasaPending,
            'DataQuartoPending'=>$QuartosPending, 'Quartos'=>$Quartos, 'Casa'=>$Casa]);
    }else{
        abort( code: 403, message: 'Acesso não autorizado.');
    }
}

```

Figure 43-controlar de página principal de gestor

Essa função GetPageGestor é responsável por renderizar a página do gestor, exibindo informações relevantes para utilizadores com permissão de gestor. Aqui está uma visão geral do que ela faz:

Verificação de Permissão:

A função verifica se o tipo de utilizador na sessão é "gestor".

Se não for um gestor, retorna um erro 403 (Acesso não autorizado).

Recuperação de Dados:

Se o utilizador for um gestor, a função continua a recuperar os dados relevantes para exibição na página do gestor.

O token de ativação do utilizador é recuperado da sessão.

Os dados do utilizador, incluindo os dados do utilizador com base no token de ativação, são recuperados do banco de dados.

Os dados de diferentes tipos de propriedades (quartos e casas) são recuperados separadamente do banco de dados. Existem três conjuntos de dados diferentes:

QuartosPending: Quartos que estão pendentes de aprovação.

CasaPending: Casas completas que estão pendentes de aprovação.

Quartos: Todos os quartos disponíveis.

Casa: Todas as casas completas disponíveis.

Renderização da Página:

Os dados recuperados são passados para a visão Page_Gestor\Gestor para renderização.

Os dados de utilizadores, quartos pendentes, casas pendentes, quartos e casas são passados como variáveis para a visão.

Retorno de Erro para Acesso Não Autorizado:

Se o utilizador não for um gestor, a função chama abort(403, 'Acesso não autorizado.'), que retorna um erro 403 (Acesso não autorizado).

Essa função é responsável por garantir que apenas gestores autorizados tenham acesso à página do gestor e exibam as informações relevantes para eles.

5.8.3. Resultado

ID	Nome	Email	Estado(Ativar/Inativar)	Remover
32	Muaiad hadad	aeadhadad19990@gmail.com	Ativo	
39	João pinto	aeadhadad5@gmail.com	Ativo	
40	Nour Pika	noush244@gmail.com	Ativo	
42	joao f	a2020132536@alunos.estgoh.ipc.pt	Ativo	
43	Test	bixako1865@kravify.com	Ativo	

Figure 44-lista utilizadores

ID	Nome	Endereço	Proprietário	Data Expirou	Preço	Estado(Inativar/Reativar)	Consultar
8	Test C 8	rua dr.carlos pinto	João pinto	2025-04-08	200€	Ativo	
11	Test	rua dr.carlos pinto	João pinto	2025-04-10	600€	Ativo	
12	Test edit	Rua Almeida Garrett , Oliveira do Hospital	Nour Pika	2025-04-10	1000€	Ativo	
1	test Q 1	coimbra	João pinto	2025-04-07	200€	Ativo	
2	test Q 2	rua dr.carlos pinto	João pinto	2025-04-07	300€	Ativo	
3	test Q 3	rua dr.carlos pinto	João pinto	2025-04-07	200€	Ativo	
4	test Q 4	Rua Almeida Garrett , Oliveira do Hospital	João pinto	2025-04-07	300€	Ativo	
5	test Q 5	Rua Almeida Garrett , Oliveira do Hospital	João pinto	2025-04-07	300€	Ativo	

Figure 45-lista dos anúncios

ID	Nome	Endereço	Proprietário	Data Expirou	Preço	Consultar
1						
2						
3						
4						
5						
6						
7						
8						
9						
10						

Figure 46-lista dos pedidos



The screenshot shows a table titled "Lista de Utilizadores". The columns are ID, Nome, Email, Estado(Ativar/Inativar), and Remover. There are two rows of data:

ID	Nome	Email	Estado(Ativar/Inativar)	Remover
39	João pinto	aeadhadad5@gmail.com	Ativo	
42	joao f	a2020132536@alunos.estgoh.ipc.pt	Ativo	

Figure 47-pesquisar na lista

5.9. Adicionar

5.9.1. Back-end

```
1 usage  ± MuaiadHadad
public function Adcionargestor(Request $request){
    $username = $request->username;
    $email = $request->Email;
    $userExists = DB::table('user')->where('Email', $email)->exists();
    if ($userExists) {
        return redirect()->back()->with('error', 'Este e-mail já está em uso. Por favor, escolha outro.');
    }
    $password = $this->generateRandomPassword();
    $hashedPassword = bcrypt($password);
    DB::table('user')->insert([
        'UserName' => $request->username,
        'Email' => $request->Email,
        'Password'=>$hashedPassword,
        'Estado'=>'Ativo',
        'Tipo'=>'gestor',
        'Avatar'=> 'avatars/User-avatar.svg.png'
    ]);
    $this->enviarEmail( titulo: 'Bem-vindo à Plataforma de Alojamento da ESTGOH!', $request->username , $password,$request->Email);
    return redirect()->back()->with('success', 'Gestor adicionado com sucesso!');
}
```

Figure 48-controlar add gestor

Esta função Adcionargestor é responsável por adicionar um novo gestor ao sistema. Aqui está uma visão geral do que ela faz:

Recebimento dos Dados:

A função recebe os dados do novo gestor por meio do objeto Request.

Os dados incluem o nome de utilizador, o email e outros detalhes necessários.

Verificação de Existência de Utilizador:

Verifica se já existe um utilizador com o email fornecido no banco de dados.

Se o email já estiver em uso, redireciona de volta à página anterior com uma mensagem de erro.

Geração de Senha Aleatória:

Gera uma senha aleatória para o novo gestor usando a função generateRandomPassword.

Hashing da Senha:

A senha gerada é então hashada usando a função bcrypt antes de ser armazenada no banco de dados.

Inserção no Banco de Dados:

Insere os detalhes do novo gestor, incluindo nome de utilizador, email, senha hashada, estado, tipo e avatar, no banco de dados.

Envio de Email de Boas-Vindas:

Um email de boas-vindas é enviado para o novo gestor contendo o seu nome de utilizador e senha gerada aleatoriamente.

Redirecionamento com Mensagem de Sucesso:

Após adicionar o gestor com sucesso e enviar o email, o utilizador é redirecionado de volta à página anterior com uma mensagem de sucesso.

Essa função garante que novos gestores possam ser adicionados ao sistema de forma segura e eficiente, além de fornecer feedback adequado ao utilizador sobre o resultado da operação.

5.10. Profile

5.10.1. Back-end

```
2 usages  • MuaiadHadad
public function GetPageProfGestor(){
    if(session( key: 'tipo_usuario')=="gestor"){
        $token=session( key: 'ActivationToken');
        $utilizadores = DB::table( table: 'user')->where( column: 'ActivationToken', $token)->get();
        if ($utilizadores) {
            return view( view: 'Page_Gestor\Profile' , ['utilizadores' => $utilizadores]);
        } else {
            abort( code: 403, message: 'Bad request');
        }
    }else{
        abort( code: 403, message: 'Acesso não autorizado.');
    }
}
```

Figure 49-dados do utilizador

```

public function updateProfile(Request $request, $id){
    if($request->passwordnovo!=null){
        $hashedPassword = bcrypt($request->passwordnovo);
        if($request->passwordold!=null ){
            $password=$request->input( key: 'passwordold');
            $utilizadores = DB::table( table: 'user')->where( column: 'id', $id)->first();
            if (!$utilizadores || !password_verify($password, $utilizadores->Password)) {
                return back()->with(['error' => 'Palavra passa está incorreta!']);
            }else{
                DB::table( table: 'user')->where( column: 'id', $id)->update([
                    'Password' => $hashedPassword
                ]);
            }
        }
    }
    if($request->file( key: 'avatar')!=null){
        $avatarPath = $request->file( key: 'avatar')->store( path: 'avatars', options: 'public');
        DB::table( table: 'user')->where( column: 'id', $id)->update([
            'Avatar' => $avatarPath,
        ]);
    }
    DB::table( table: 'user')->where( column: 'id', $id)->update([
        'UserName' => $request->username,
        'Email' => $request->Email,
    ]);
    return redirect()->back()->with('success', 'Perfil do usuário atualizado com sucesso!');
}

```

Figure 50-update dados de utilizador

Este trecho de código HTML representa um formulário de registo de utilizador. Aqui está uma explicação de cada parte do código:

Estrutura do Formulário:

O formulário está contido numa estrutura de divs com as classes container-main, container-login e form-box-login, que podem ser usadas para aplicar estilos de layout específicos.

Imagen:

Uma imagem é exibida acima do formulário. A URL da imagem é definida como <https://comum.rcaap.pt/retrieve/104938>.

Campos do Formulário:

Existem quatro campos no formulário:

Username: Um campo de texto onde os utilizadores podem inserir o seu nome de utilizador.

Email: Um campo de texto onde os utilizadores podem inserir o seu endereço de e-mail.

Password: Um campo de senha onde os utilizadores podem inserir a sua senha.

Repassword. Um campo de senha onde os utilizadores devem confirmar a sua senha, digitando-a novamente.

Botão de Envio:

Um botão "Registrar" é fornecido para enviar o formulário. Quando clicado, o formulário será submetido para a rota /registar usando o método HTTP POST.

Validação dos Campos:

Todos os campos do formulário são marcados com o atributo required, o que significa serem obrigatórios. Isso garante que os utilizadores não possam enviar o formulário sem preencher todos os campos obrigatórios.

Este formulário é usado para coletar informações de registo de utilizadores, como nome de utilizador, e-mail e senha. Uma vez preenchido e enviado, os dados do formulário podem ser processados por um servidor “web” para criar uma conta de utilizador no sistema.

5.10.2. Resultado



Figure 51-profile

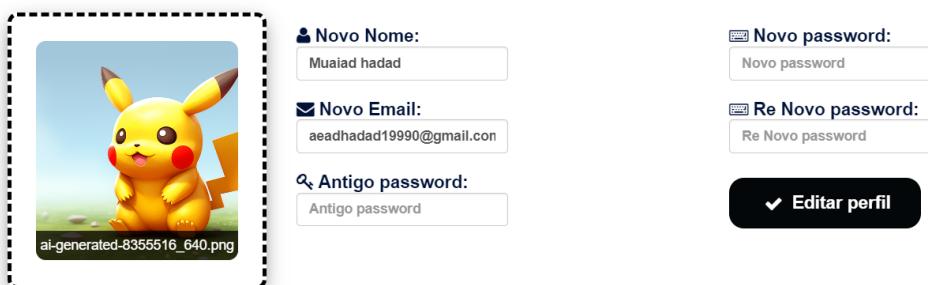
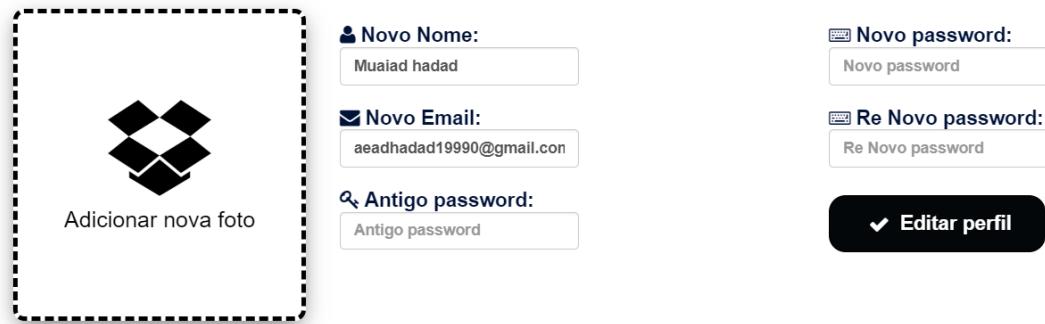


Figure 52-editar porfile

5.11. Back-end remover utilizador

```

1 usage  ↗ MuaiadHadad
public function removerUser($id){
    $user = DB::table('user')->where('id', $id)->first();

    if (!$user) {
        return redirect()->back()->with('error', 'Não foi possível remover o utilizador!');
    }
    DB::table('user')->where('id', $id)->delete();

    return redirect()->back()->with('success', 'utilizador foi removido com sucesso!');
}

1 usage  ↗ MuaiadHadad
public function GetPageAddGestor(){
    if(session('tipo_usuario')=="gestor"){
        $token=session('ActivationToken');
        $user = DB::table('user')->where('ActivationToken', $token)->get();
        return view('Page_Gestor\Adicionar_Gestor',['Data'=>$user]);
    }else{
        abort(403, 'Acesso não autorizado.');
    }
}

```

Figure 53-remover utilizador

Aqui está uma análise das funções removerUser e GetPageAddGestor:

`removerUser($id)`

Recuperação do Utilizador:

A função tenta recuperar o utilizador com o ID fornecido do banco de dados.

Verificação da Existência do Utilizador:

Verifica se o utilizador existe. Se não existir, redireciona de volta com uma mensagem de erro.

Remoção do Utilizador:

Se o utilizador existir, remove-o do banco de dados.

Redirecionamento com Mensagem de Sucesso:

Após a remoção bem-sucedida, redireciona de volta com uma mensagem de sucesso.

`GetPageAddGestor()`

Verificação de Permissões:

Verifica se o tipo de utilizador armazenado na sessão é "gestor". Se não for, gera um erro 403 (Acesso não autorizado).

Recuperação de Dados do Utilizador:

Se o tipo de utilizador for "gestor", recupera os detalhes do utilizador (gestor) atual com base no token de ativação armazenado na sessão.

Renderização da Página de Adicionar Gestor:

Se os detalhes do utilizador (gestor) forem recuperados com sucesso, renderiza a página de adicionar gestor, passando os detalhes do utilizador como dados para a visualização.

Caso contrário, retorna um erro 403 (Solicitação inválida).

Essas funções garantem que apenas gestores possam remover utilizadores e acessar a página para adicionar gestores. Além disso, fornecem feedback apropriado ao utilizador sobre o resultado das operações.

5.12. Back-end consultar anúncio

```
1 usage  ~ MuaiadHadad
public function GetPageDetalheQuarto($id)
{
    if (session('ActivationToken') != null) {
        $token = session('ActivationToken');
        $user = DB::table('user')->where('ActivationToken', $token)->get();
        $QuartosAtive = DB::table('quarto')
            ->join('banho', 'banho.id_quarto', '=', 'quarto.id')
            ->join('contato', 'contato.id_quarto', '=', 'quarto.id')
            ->join('cozinha', 'cozinha.id_quarto', '=', 'quarto.id')
            ->join('endereco', 'endereco.id_quarto', '=', 'quarto.id')
            ->join('quartos_de_casa', 'quartos_de_casa.id_quarto', '=', 'quarto.id')
            ->join('sala', 'sala.id_quarto', '=', 'quarto.id')
            ->join('servicos', 'servicos.id_quarto', '=', 'quarto.id')
            ->join('outros', 'outros.id_quarto', '=', 'quarto.id')
            ->join('user', 'user.id', '=', 'quarto.id_user')
            ->where('quarto.id', $id)
            ->select(['contato.Email as EmailQuarto', 'user.*', 'cozinha.Micro-ondas as Micro', 'quarto.id as id',
                    'quartos_de_casa.*',
                    'sala.*', 'servicos.*', 'outros.*', 'servicos.Wi-Fi as wifi'])->get();
        if (!$QuartosAtive){
            abort(404, 'Bad Request.');
        }
        $PhotoQuarto = DB::table('quarto')
            ->join('midia_de_casa', 'midia_de_casa.id_quarto', '=', 'quarto.id')
            ->where('quarto.id', $id)
            ->select('midia_de_casa.*')->get();

        return view('Page_Gestor\detalhe_quarto_Gestor', ['DadosUser' => $user, 'DadosQuarto' => $QuartosAtive, 'PhotoQuarto' => $PhotoQuarto]);
    } else{
        $QuartosAtive = DB::table('quarto')
            ->join('banho', 'banho.id_quarto', '=', 'quarto.id')
```

```

    return view( view: 'Page_Gestor\detalhe_quarto_Gestor', [ 'DadosUser' => $user, 'DadosQuarto'=>$QuartosAtive,'P...
} else{
    $QuartosAtive =DB::table( table: 'quarto')
        ->join( table: 'banho', first: 'banho.id_quarto', operator: '=', second: 'quarto.id')
        ->join( table: 'contato', first: 'contato.id_quarto', operator: '=', second: 'quarto.id')
        ->join( table: 'cozinha', first: 'cozinha.id_quarto', operator: '=', second: 'quarto.id')
        ->join( table: 'endereco', first: 'endereco.id_quarto', operator: '=', second: 'quarto.id')
        ->join( table: 'quartos_de_casa', first: 'quartos_de_casa.id_quarto', operator: '=', second: 'quarto.id')
        ->join( table: 'sala', first: 'sala.id_quarto', operator: '=', second: 'quarto.id')
        ->join( table: 'servicos', first: 'servicos.id_quarto', operator: '=', second: 'quarto.id')
        ->join( table: 'outros', first: 'outros.id_quarto', operator: '=', second: 'quarto.id')
        ->join( table: 'user', first: 'user.id', operator: '=', second: 'quarto.id_user')
        ->where( column: 'quarto.id', $id)
        ->select( columns: 'contato.Email as EmailQuarto','user.*','cozinha.Micro-ondas as Micro','quarto.id as id
        , 'quartos_de_casa.*'
        , 'sala.*', 'servicos.*', 'outros.*','servicos.Wi-Fi as wifi')->get();
    if(!$QuartosAtive){
        abort( code: 404, message: 'Bad Request.');
    }
    $PhotoQuarto =DB::table( table: 'quarto')
        ->join( table: 'midia_de_casa', first: 'midia_de_casa.id_quarto', operator: '=', second: 'quarto.id')
        ->where( column: 'quarto.id', $id)
        ->select( columns: 'midia_de_casa.*')->get();

    return view( view: 'Page_Gestor\detalhe_quarto_Gestor', [ 'DadosUser' => null,'DadosQuarto'=>$QuartosAtive,'P...
}

```

Figure 54-consultar anúncio

Aqui está uma análise da função GetPageDetalheQuarto:

GetPageDetalheQuarto(\$id)

Verificação da Sessão:

Verifica se há um token de ativação armazenado na sessão.

Recuperação dos Dados do Utilizador:

Se houver um token de ativação na sessão, a função recupera os detalhes do utilizador com base nesse token.

Recuperação dos Dados do Quarto:

A função busca os detalhes do quarto com o ID fornecido no banco de dados.

Os dados incluem informações sobre o quarto, como detalhes de contrato, comodidades, localização, etc.

Se nenhum quarto for encontrado com o ID fornecido, a função aborta com um erro 404 (Solicitação inválida).

Recuperação das Fotos do Quarto:

A função também busca as fotos associadas ao quarto no banco de dados.

Renderização da Página:

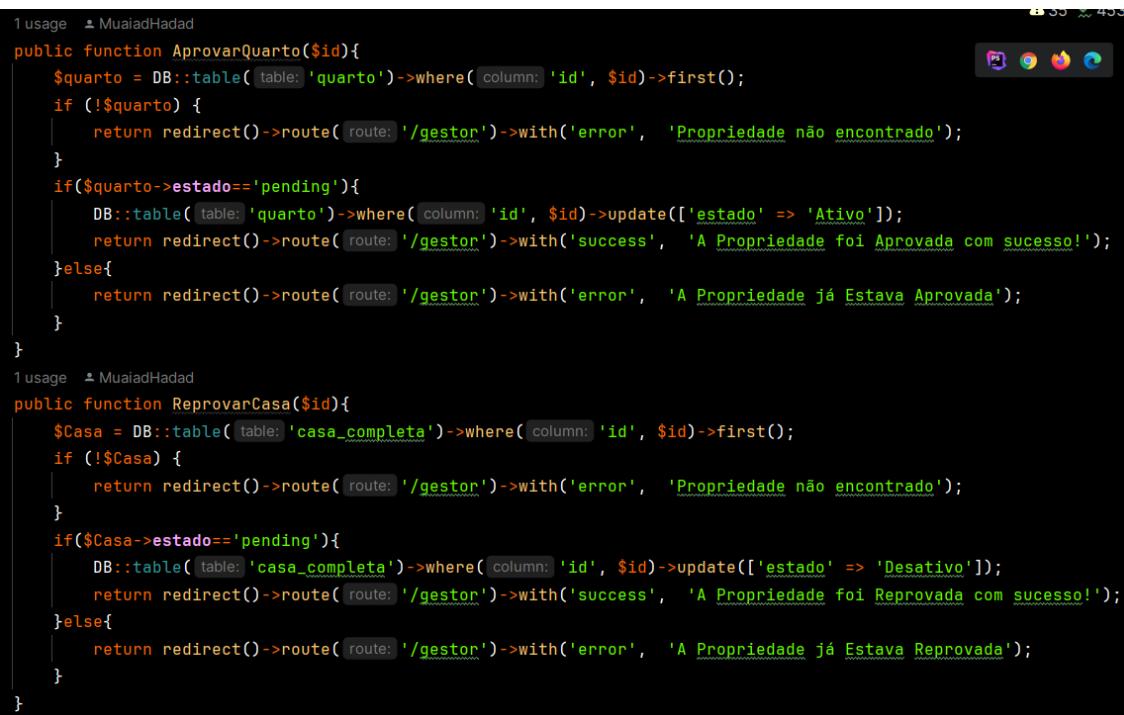
Com base nos dados recuperados, a função renderiza a página de detalhes do quarto.

Se houver um utilizador na sessão, os detalhes do utilizador são passados para a visualização, com os detalhes e fotos do quarto.

Se não houver utilizador na sessão, apenas os detalhes e fotos do quarto são passados para a visualização.

Essa função permite visualizar os detalhes de um quarto específico e as fotos associadas a ele. Dependendo da presença ou ausência de um utilizador na sessão, a página renderizada pode exibir ou não informações adicionais sobre o utilizador.

5.13. Back-end aprovar e reprovar anúncio



```

1 usage  ± MuaiadHadad
public function AprovarQuarto($id){
    $quarto = DB::table('quarto')->where('id', $id)->first();
    if (!$quarto) {
        return redirect()->route('/gestor')->with('error', 'Propriedade não encontrado');
    }
    if($quarto->estado=='pending'){
        DB::table('quarto')->where('id', $id)->update(['estado' => 'Ativo']);
        return redirect()->route('/gestor')->with('success', 'A Propriedade foi Aprovada com sucesso!');
    }else{
        return redirect()->route('/gestor')->with('error', 'A Propriedade já Estava Aprovada');
    }
}

1 usage  ± MuaiadHadad
public function ReprovarCasa($id){
    $Casa = DB::table('casa_completa')->where('id', $id)->first();
    if (!$Casa) {
        return redirect()->route('/gestor')->with('error', 'Propriedade não encontrado');
    }
    if($Casa->estado=='pending'){
        DB::table('casa_completa')->where('id', $id)->update(['estado' => 'Desativo']);
        return redirect()->route('/gestor')->with('success', 'A Propriedade foi Reprovada com sucesso!');
    }else{
        return redirect()->route('/gestor')->with('error', 'A Propriedade já Estava Reprovada');
    }
}

```

Figure 55-Aprovar e reprovar anúncio

Aqui está uma análise das funções AprovarQuarto e ReprovarCasa:

AprovarQuarto(\$id)

Recuperação do Quarto:

A função tenta recuperar os detalhes do quarto com o ID fornecido no banco de dados.

Verificação da Existência do Quarto:

Se o quarto não for encontrado, a função redireciona de volta à página do gestor com uma mensagem de erro.

Verificação do Estado do Quarto:

Verifica se o estado do quarto é 'pending' (pendente).

Se o estado for 'pending', atualiza o estado do quarto para 'Ativo' e redireciona de volta à página do gestor com uma mensagem de sucesso.

Se o estado do quarto não for 'pending', significa que já está aprovado, então a função redireciona de volta à página do gestor com uma mensagem de erro.

ReprovarCasa(\$id)

Recuperação da Casa:

A função tenta recuperar os detalhes da casa completa com o ID fornecido no banco de dados.

Verificação da Existência da Casa:

Se a casa completa não for encontrada, a função redireciona de volta à página do gestor com uma mensagem de erro.

Verificação do Estado da Casa:

Verifica se o estado da casa é 'pending' (pendente).

Se o estado for 'pending', atualiza o estado da casa para 'Desativo' e redireciona de volta à página do gestor com uma mensagem de sucesso.

Se o estado da casa não for 'pending', significa que já está reprovada, então a função redireciona de volta à página do gestor com uma mensagem de erro.

Ambas as funções garantem que a aprovação ou reprovação seja feita apenas se o item estiver no estado adequado e fornecem feedback apropriado ao utilizador após a operação.

5.14. Back-end mudar estado de um anúncio

```
1 usage  ± MuaiadHadad
  public function MudarestadoQuarto($id){
    $quarto = DB::table('quarto')->where('id', $id)->first();

    if (!$quarto) {
      return redirect()->back()->with('error', 'Quarto não encontrado');
    }
    if($quarto->estado=='pending'){
      return redirect()->back()->with('error', 'Ainda não pode mudar o estado deste Propriedade');
    }
    $novoEstado = $quarto->estado == 'Ativo' ? 'Desativo' : ($quarto->estado == 'Desativo' ? 'Ativo' : 'Ativo');

    DB::table('quarto')->where('id', $id)->update(['estado' => $novoEstado]);

    return redirect()->back()->with('success', 'Estado do Quarto alterado com sucesso!');
  }
```

Figure 56-mudar estado de um anúncio

Aqui está uma análise da função **MudarestadoQuarto(\$id)**:

MudarestadoQuarto(\$id)

Recuperação do Quarto:

A função tenta recuperar os detalhes do quarto com o ID fornecido no banco de dados.

Verificação da Existência do Quarto:

Se o quarto não for encontrado, a função redireciona de volta à página anterior com uma mensagem de erro.

Verificação do Estado do Quarto:

Verifica se o estado do quarto é 'pending' (pendente).

Se o estado for 'pending', significa que ainda não pode mudar o estado deste quarto, então a função redireciona de volta à página anterior com uma mensagem de erro.

Atualização do Estado do Quarto:

Determina o novo estado do quarto com base no estado atual.

Se o estado atual for 'Ativo', o novo estado será 'Desativo'.

Se o estado atual for 'Desativo', o novo estado será 'Ativo'.

Atualiza o estado do quarto no banco de dados com o novo estado.

Redirecionamento com Mensagem de Sucesso:

Após a atualização bem-sucedida, a função redireciona de volta à página anterior com uma mensagem de sucesso informando que o estado do quarto foi alterado com sucesso.

Essa função permite alternar entre os estados 'Ativo' e 'Desativo' do quarto e fornece feedback adequado ao utilizador após a operação.

5.15. Senhorio área pessoal

5.15.1. Back-end

```

public function GetPageSenhorio(){
    if(session( key: 'tipo_usuario')=="senhorio"){
        $token=session( key: 'ActivationToken');
        $user = DB::table( table: 'user')->where( column: 'ActivationToken', $token)->get();
        $userqu = DB::table( table: 'user')->where( column: 'ActivationToken', $token)->first();
        $quartos =DB::table( table: 'quarto')
            ->join( table: 'banho', first: 'banho.id_quarto', operator: '=', second: 'quarto.id')
            ->join( table: 'contato', first: 'contato.id_quarto', operator: '=', second: 'quarto.id')
            ->join( table: 'cozinha', first: 'cozinha.id_quarto', operator: '=', second: 'quarto.id')
            ->join( table: 'endereco', first: 'endereco.id_quarto', operator: '=', second: 'quarto.id')
            ->join( table: 'quartos_de_casa', first: 'quartos_de_casa.id_quarto', operator: '=', second: 'quarto.id')
            ->join( table: 'sala', first: 'sala.id_quarto', operator: '=', second: 'quarto.id')
            ->join( table: 'servicos', first: 'servicos.id_quarto', operator: '=', second: 'quarto.id')
            ->join( table: 'outros', first: 'outros.id_quarto', operator: '=', second: 'quarto.id')
            ->where( column: 'id_user', $userqu->id)
            ->select( columns: 'quarto.id as idnow','quarto.*', 'banho.*', 'contato.*', 'cozinha.*', 'endereco.*'
                , 'quartos_de_casa.*'
                , 'sala.*', 'servicos.*', 'outros.*')->get();
        $Casa =DB::table( table: 'casa_completa')
            ->join( table: 'banho', first: 'banho.id_casa', operator: '=', second: 'casa_completa.id')
            ->join( table: 'contato', first: 'contato.id_casa', operator: '=', second: 'casa_completa.id')
            ->join( table: 'cozinha', first: 'cozinha.id_casa', operator: '=', second: 'casa_completa.id')
            ->join( table: 'endereco', first: 'endereco.id_casa', operator: '=', second: 'casa_completa.id')
            ->join( table: 'sala', first: 'sala.id_casa', operator: '=', second: 'casa_completa.id')
            ->join( table: 'servicos', first: 'servicos.id_casa', operator: '=', second: 'casa_completa.id')
            ->join( table: 'outros', first: 'outros.id_casa', operator: '=', second: 'casa_completa.id')
            ->where( column: 'id_user', $userqu->id)
            ->select( columns: 'casa_completa.id as idnow','casa_completa.*', 'banho.*', 'contato.*', 'cozinha.*', 'endereco.*'
                , 'sala.*', 'servicos.*', 'outros.*')->get();
        return view( view: 'Page_Senhorio/Senhorio_Principal_page', ['Data'=>$user, 'Quarto'=>$quartos,'Casa'=>$Casa]);
    }else{
        abort( code: 403, message: 'Acesso não autorizado.');
    }
}

```

Figure 57-controlar área pessoal senhorio

Aqui está uma análise da função GetPageSenhorio():

GetPageSenhorio()

Verificação do Tipo de Usuário:

Verifica se o tipo de usuário na sessão é "senhorio". Se não for, aborta a solicitação com um código de status 403 (Acesso não autorizado).

Recuperação do Usuário Atual:

Obtém o token da sessão.

Recupera os detalhes do utilizador com base no token de ativação da sessão.

Recuperação dos Quartos e Casas do Senhorio:

Recupera os quartos do senhorio do banco de dados, com os seus detalhes de casa de banho, contato, cozinha, endereço, sala, serviços e outros.

Recupera as casas completas do senhorio do banco de dados, com os seus detalhes de casa de banho, contrato, cozinha, endereço, sala, serviços e outros.

Renderização da Página:

Renderiza a página 'Senhorio_Principal_page' e passa os seguintes dados para a visualização:

Detalhes do utilizador.

Quartos do senhorio.

Casas completas do senhorio.

Essa função é responsável por exibir a página principal para os senhorios após verificar se o tipo de utilizador na sessão é um senhorio. Ele fornece uma visão geral dos quartos e casas do senhorio, permitindo que eles gerenciem as suas propriedades.

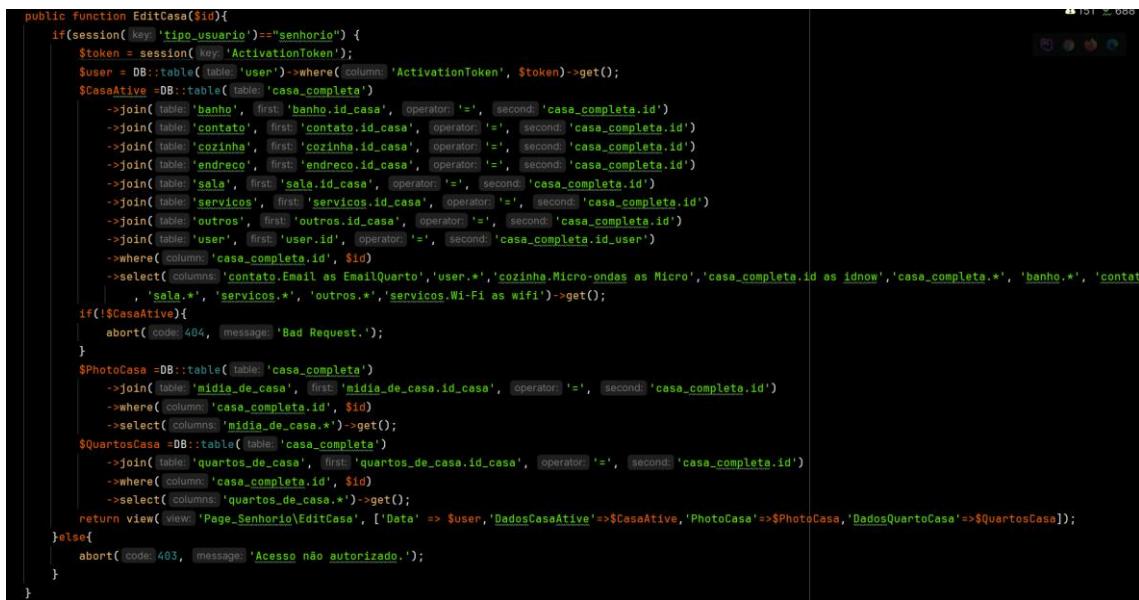
5.15.2. Resultado

Lista dos Propriedades							
ID	Título	Endereço	Código-Postal	Data Expirou	Preço	Estado(Inativar/Reativar)	Pesquisar anúncio...
1	test Q 1	coimbra	3220	2025-04-07	200€	Ativo	
2	test Q 2	rua dr.carlos pinto	3220-301	2025-04-07	300€	Ativo	
3	test Q 3	rua dr.carlos pinto	3220-301	2025-04-07	200€	Ativo	
4	test Q 4	Rua Almeida Garrett , Oliveira do Hospital	3400-080	2025-04-07	300€	Ativo	
5	test Q 5	Rua Almeida Garrett , Oliveira do Hospital	3400-080	2025-04-07	300€	Ativo	
6	Test Q 6	Rua Almeida Garrett , Oliveira do Hospital	3400-080	2025-04-07	1150€	Ativo	
7	Test Q 7	Rua Almeida Garrett , Oliveira do Hospital	3400-080	2025-04-07	155€	Ativo	
8	Test Q 8	rua dr.carlos pinto	3220-301	2025-04-07	160€	Ativo	
9	Test Q 9	Rua Almeida Garrett , Oliveira do Hospital	3400-080	2025-04-07	200€	Ativo	

Figure 58-lista das propriedades

5.16. Editar anúncio

5.16.1. Back-end



```

public function EditCasa($id){
    if(session('key','tipo_usuario')=="senhorio") {
        $token = session('key','ActivationToken');
        $user = DB::table('table: user')->where(column: 'ActivationToken', $token)->get();
        $CaseActive = DB::table('table: casa_completa')
            ->join( table: 'banho', first: 'banho.id_casa', operator: '=', second: 'casa_completa.id')
            ->join( table: 'contrato', first: 'contrato.id_casa', operator: '=', second: 'casa_completa.id')
            ->join( table: 'cozinha', first: 'cozinha.id_casa', operator: '=', second: 'casa_completa.id')
            ->join( table: 'endereco', first: 'endereco.id_casa', operator: '=', second: 'casa_completa.id')
            ->join( table: 'sala', first: 'sala.id_casa', operator: '=', second: 'casa_completa.id')
            ->join( table: 'servicos', first: 'servicos.id_casa', operator: '=', second: 'casa_completa.id')
            ->join( table: 'outros', first: 'outros.id_casa', operator: '=', second: 'casa_completa.id')
            ->join( table: 'user', first: 'user.id', operator: '=', second: 'casa_completa.id_user')
            ->where(column: 'casa_completa.id', $id)
            ->select(columns: 'contato.Email as EmailQuarto','user.*','cozinha.Micro-ondas as Micro','casa_completa.id as idnow','casa_completa.*', 'banho.*', 'contrato.*', 'sala.*', 'servicos.*', 'outros.*','servicos.Wi-Fi as wifi')->get();
        if(!$CaseActive){
            abort(code: 404, message: 'Bad Request.');
        }
        $PhotoCasa =DB::table('table: casa_completa')
            ->join( table: 'midia_de_casa', first: 'midia_de_casa.id_casa', operator: '=', second: 'casa_completa.id')
            ->where(column: 'casa_completa.id', $id)
            ->select(columns: 'midia_de_casa.*')->get();
        $QuartosCasa =DB::table('table: casa_completa')
            ->join( table: 'quartos_de_casa', first: 'quartos_de_casa.id_casa', operator: '=', second: 'casa_completa.id')
            ->where(column: 'casa_completa.id', $id)
            ->select(columns: 'quartos_de_casa.*')->get();
        return view('Page_Senhorio>EditCasa', ['Data' => $user,'DadosCasaActive'=>$CaseActive,'PhotoCasa'=>$PhotoCasa,'DadosQuartoCasa'=>$QuartosCasa]);
    }else{
        abort(code: 403, message: 'Acesso não autorizado.');
    }
}

```

Figure 59-enviar dados de anúncio para página de edit

Aqui está uma análise da função EditCasa(\$id):

EditCasa(\$id)

Verificação do Tipo de Utilizador:

Verifica se o tipo de utilizador na sessão é "senhorio". Se não for, aborta a solicitação com um código de status 403 (Acesso não autorizado).

Recuperação do Utilizador Atual e Detalhes da Casa:

Obtém o token da sessão.

Recupera os detalhes do utilizador com base no 'token' de ativação da sessão.

Recupera os detalhes da casa completa com o ID fornecido, com os seus detalhes de casa de banho, contrato, cozinha, endereço, sala, serviços e outros.

Se não encontrar a casa com o ID fornecido, aborta a solicitação com um código de status 404 (Solicitação inválida).

Recuperação das Fotos da Casa:

Recupera as fotos da casa completa com o ID fornecido.

Recuperação dos Quartos da Casa:

Recupera os quartos da casa completa com o ID fornecido.

Renderização da Página de Edição da Casa:

Renderiza a página 'EditCasa' e passa os seguintes dados para a visualização:

Detalhes do utilizador.

Detalhes da casa completa.

Fotos da casa completa.

Quartos da casa completa.

Essa função permite que os senhorios editem os detalhes de uma casa completa específica, incluindo os seus quartos, e visualizem as fotos associadas a ela.

```
public function SubEditCasa($id, Request $request){
    if(session( key: 'tipo_usuario')=="senhorio") {
        if (empty($request->input( key: 'title')) || empty($request->input( key: 'descricao')) || empty($request->input( key: 'Tipo')) || empty($request->input( key: 'sexo')) )
            return redirect()->back()->with('error', 'Por favor, preencha todos os campos obrigatórios.');
    }

    $token=session( key: 'ActivationToken');
    $user = DB::table( table: 'user')->where( column: 'ActivationToken', $token)->first();
    DB::table( table: 'casa_completa')->where( column: 'id', $id)
        ->update([
            'Titulo' => $request->input( key: 'title'),
            'descricao' => $request->input( key: 'descricao'),
            'Tipo' => $request->input( key: 'Tipo'),
            'Genero' => $request->input( key: 'sexo'),
            'Preco' => $request->input( key: 'preco'),
            'area' => $request->input( key: 'area'),
            'estado' => 'pending',
            'id_user'=>$user->id,
        ]);

    DB::table( table: 'cozinha')->where( column: 'id_casa', $id)
        ->update([
            'Forno' => $request->has( key: 'Forno') ? true : false,
            'Fogao' => $request->has( key: 'Fogao') ? true : false,
            'Calefaria' => $request->has( key: 'Calefaria') ? true : false,
            'Mad_cafe' => $request->has( key: 'Mad_cafe') ? true : false,
            'Placa' => $request->has( key: 'Placa') ? true : false,
            'Micro-ondas' => $request->has( key: 'Micro-ondas') ? true : false,
            'Pratos' => $request->has( key: 'Pratos') ? true : false,
            'Utensilios' => $request->has( key: 'Utensilios') ? true : false,
            'Frigorifico' => $request->has( key: 'Frigorifico') ? true : false,
        ]);

    DB::table( table: 'banho')->where( column: 'id_casa', $id)
        ->update([
            'Chuveiro' => $request->has( key: 'Chuveiro') ? true : false,
            'Toalhas' => $request->has( key: 'Toalhas') ? true : false,
        ]);
    DB::table( table: 'contato')->where( column: 'id_casa', $id)
        ->update([
            'Nome' => $request->input( key: 'name'),
            'Email' => $request->input( key: 'email'),
            'Telefone' => $request->input( key: 'title'),
        ]);
    DB::table( table: 'endereco')->where( column: 'id_casa', $id)
        ->update([
            'Endereco' => $request->input( key: 'Endereco'),
            'N_andar' => $request->input( key: 'N_andar'),
            'Codigo_postal' => $request->input( key: 'Codigo_postal'),
            'Distancia' => $request->input( key: 'Distancia'),
            'let' => $request->input( key: 'letlag'),
        ]);
    DB::table( table: 'midia_de_casa')->where( column: 'id_casa', $id)->delete();
    if (!is_null($request->file( key: 'photos'))) {
        foreach ($request->file( key: 'photos') as $photo) {
            $filename = $photo->store( path: 'upload1', options: 'public');
            DB::table( table: 'midia_de_casa')
                ->insert([
                    'Path' => $filename,
                    'id_casa'=> $id
                ]);
        }
    }
}
```

```

foreach ($request->input('key:existing_photos') as $photo) {
    DB::table('table: 'midia_de_casa')
        ->insert([
            'Path' => $photo,
            'id_casa'=> $id
        ]);
}

DB::table('outros')->where( column: 'id_casa', $id)
    ->update([
        'Maquina_lavar_roupa' => $request->has( key: 'Maquina_lavar_roupa') ? true : false,
        'Maquina_sacar_roupa' => $request->has( key: 'Maquina_sacar_roupa') ? true : false,
        'Aquecimento_central' =>$request->has( key: 'Aquecimento_central') ? true : false,
        'passar_Ferro' => $request->has( key: 'passar_Ferro') ? true : false,
        'Aquecedor_eletrico' => $request->has( key: 'Aquecedor_eletrico') ? true : false,
    ]);
DB::table( table: 'sala')->where( column: 'id_casa', $id)
    ->update([
        'estar_partilhada' => $request->has( key: 'estar_partilhada') ? true : false,
        'Sofas' => $request->has( key: 'Sofas') ? true : false,
        'Televisao' => $request->has( key: 'Televisao') ? true : false,
        'Mesa_jantar' => $request->has( key: 'Mesa_jantar') ? true : false,
    ]);
DB::table('servicos')->where( column: 'id_casa', $id)
    ->update([
        'Wi-Fi' => $request->has( key: 'Wi-Fi') ? true : false,
        'Elevador' => $request->has( key: 'Elevador') ? true : false,
        'Despesas' => $request->has( key: 'Despesas') ? true : false,
        'Recibo' => $request->has( key: 'Recibo') ? true : false,
        'Limpeza' => $request->has( key: 'Limpeza') ? true : false,
    ]);
return redirect()->back()->with('success', 'A sua propriedade foi Editada com sucesso!');

```

Figure 60-submeter editar

Aqui está uma análise da função SubEditCasa(\$id, Request \$request):

SubEditCasa(\$id, Request \$request)

Verificação do Tipo de Utilizador:

Verifica se o tipo de utilizador na sessão é "senhorio". Se não for, aborta a solicitação com um código de status 403 (Acesso não autorizado).

Validação dos Campos Obrigatórios:

Verifica se os campos obrigatórios estão preenchidos. Se algum estiver vazio, redireciona de volta com uma mensagem de erro.

Atualização dos Detalhes da Casa Completa:

Recupera o utilizador atual com base no token de ativação da sessão.

Atualiza os detalhes da casa completa com o ID fornecido usando os dados fornecidos no formulário, incluindo título, descrição, tipo, género, preço, área e estado (definido como "pending").

Atualiza também o ID do utilizador associado à casa.

Atualização dos Detalhes da Cozinha, Casa de banho, Contrato, Endereço, Outros, Sala e Serviços:

Atualiza os detalhes da cozinha, casa de banho, contrato, endereço, outros, sala e serviços associados à casa completa com base nos dados fornecidos no formulário.

Atualização das Fotos da Casa:

Exclui todas as fotos associadas à casa com o ID fornecido.

Se novas fotos forem enviadas, as armazena e as associa à casa.

Também adiciona quaisquer fotos existentes selecionadas para permanecer.

Redireccionamento de Volta com Mensagem de Sucesso:

Redireciona de volta para a página anterior com uma mensagem de sucesso após a edição bem-sucedida da propriedade.

Essa função permite que os senhorios editem os detalhes de uma casa completa, incluindo as suas configurações de cozinha, casa de banho, contrato, endereço, outros, sala, serviços e fotos associadas.

5.16.2. Resultado

The screenshot shows a web-based form for editing a property. The form is divided into several sections:

- Descrição e preço do imóvel** (Description and price of the property):
 - Titulo da propriedade** (Property title): test Q 1
 - Descrição da Propriedade** (Property description): test Q 1
 - Tipo** (Type): Casa
 - Género de residência** (Residence gender): Masculino
- caractéristica** (Characteristics):
 - roupa de cama cama mesa cabeceira Candeeiro de mesa do estudo Mesa do estudo Janelas Varanda
 - Armário Casa de banho privativa
- Preço** (Price): 200
- Area M₂ da Casa** (House M₂ area): 250

Mídia de propriedade


Clique aqui para fazer upload

Localização da propriedade

Endereço	Nº Andar
coimbra	3 dir
Código-postal	
3220	
📍 Distância	Google MAP

Clique no mapa para definir o local:



📍 Distância é 0,9 km

Características da Propriedade

Cozinha

- Forno
- Fogão
- Caldeira de água
- Máquina de café
- Placa
- Micro-ondas
- Pratos e talheres
- Utensílios de cozinha
- Frigorífico

Sala

- Área de estar partilhada
- Sofás
- Televisão
- Mesa de jantar com cadeiras

Casa de banho

- Chuveiro
- Toalhas

Outros

- Máquina de lavar roupa
- Máquina de sacar roupa
- Aquecimento central
- máquina passar Ferro
- Aquecedor elétrico

Serviços

- Wi-Fi
- Elevador
- Despesas incluídas
- Recibo
- limpeza

Informações de contato

Nome	Email
<input type="text" value="muaiad mhd fahd hadad"/>	<input type="text" value="aeadhadad5@gmail.com"/>
Telefone	
<input type="text" value="test Q 1"/>	

Editar

Success A sua propriedade foi Editada com sucesso!

Descrição e preço do imóvel

Título da propriedade	<input type="text" value="test Q 1"/>
Descrição da Propriedade	

ID	Título	Endereço	Código-Postal	Data Expirou	Preço	Estado(Inativar/Reativar)	Consultar	Editar	Remover
1	test Q 1	coimbra	3220	2025-04-07	200€	Pending			
2	test Q 2	rua dr.carolos pinto	3220-301	2025-04-07	300€	Ativo			

Figure 61-editar

5.17. Adicionar anúncio

5.17.1. Front-end

```

<div id="Porquarto" style="...>
    <form action="/Senhorio/Adicionar/AddQuarto" method="POST" enctype="multipart/form-data">
        @csrf
        <div class="submit_1 clearfix">
            <h4 class="mgt col_1">Descrição e preço do Quarto</h4>
            <hr>
            <h5>Título da propriedade</h5>
            <input name="title" class="form-control" placeholder="Property Title" type="text">
            <h5>Descrição da Propriedade</h5>
            <textarea name="descricao" placeholder="Property Description" class="form-control form_o"></textarea>
            <div class="col-sm-6 space_all">
                <div class="submit_111 clearfix">
                    <h5>Tipo</h5>
                    <select class="form-control" name="Tipo" required>
                        <option value="Apartamento">Selecione Tipo</option>
                        <option value="Casa">Casa</option>
                        <option value="Apartamento">Apartamento</option>
                        <option value="Estúdio">Estúdio</option>
                    </select>
                </div>
            </div>
            <div class="col-sm-6 space_right">
                <div class="submit_111 clearfix">
                    <h5>Género de residência</h5>
                    <select class="form-control" name="sexo" required>
                        <option value="M/F">Selecione Sexo</option>
                        <option value="Masculino">Masculino</option>
                        <option value="Feminino">Feminino</option>
                        <option value="M/F">Masculino e Feminino</option>
                    </select>
                </div>
            </div>
        </div>
    </div>
    <div class="submit_11 clearfix">
        <div class="col-sm-6 space_left">
            <div class="submit_111 clearfix">
                <h5>Preço</h5>
                <input name="preco" class="form-control" placeholder="EUR" type="text" required>
            </div>
        </div>
        <div class="col-sm-6 space_right">
            <div class="submit_111 clearfix">
                <h5>Área M<sub>2</sub> do quarto</h5>
                <input name="area" class="form-control" placeholder="m²" type="text" required>
            </div>
        </div>
    </div>
    <br>
    <h4><i class="fa fa-area-chart" aria-hidden="true"></i> características</h4>
    <div class="submit_2i clearfix">
        <h5><input name="roupa_de_cama" type="checkbox"> <span class="span_1">roupa de cama</span></h5>
        <h5><input name="cama" type="checkbox"> <span class="span_1">cama</span></h5>
        <h5><input name="mesa_cabeceira" type="checkbox"> <span class="span_1">mesa cabeceira</span></h5>
        <h5><input name="Candeeiro_de_mesa_do_estudo" type="checkbox"> <span class="span_1">Candeeiro de mesa do estudo</span></h5>
        <h5><input name="Mesa_do_estudo" type="checkbox"> <span class="span_1">Mesa do estudo</span></h5>
        <h5><input name="Janelas" type="checkbox"> <span class="span_1">Janelas</span></h5>
        <h5><input name="Varanda" type="checkbox"> <span class="span_1">Varanda</span></h5>
        <h5><input name="Armario" type="checkbox"> <span class="span_1">Armário</span></h5>
        <h5><input name="Casa_de_banho_privativa" type="checkbox"> <span class="span_1">Casa de banho privativa</span></h5>
    </div>
</div>

```

```

<h4 class="mgt col_1">Midia de propriedade</h4>
<hr>
<div class="submit_lii clearfix" id="uploadArea">
    <span class="span_1"><i class="fa fa-cloud-upload"></i></span>
    <h5>Clique aqui ou solte os arquivos para fazer upload</h5>
</div>
<input type="file" name="photos[]" id="fileInput" multiple="multiple" style="display: none;" required>
<hr>
<div id="fileListPorQuarto"></div>
</div>


Figure 62-Html adicionar anúncio



Este formulário parece ser destinado a adicionar um novo quarto a uma propriedade. Aqui está uma análise das seções do formulário:



77


```

Seção Descrição e Preço do Quarto

Campos para inserir o título e a descrição do quarto.

Seleção do tipo de propriedade (Casa, Apartamento, Estúdio).

Seleção do gênero da residência (Masculino, Feminino, M/F).

Inserção do preço do quarto em EUR.

Inserção da área do quarto em metros quadrados.

Seção Características do Quarto

Checkboxes para selecionar as características do quarto, como roupa de cama, cama, mesa de cabeceira, etc.

Seção Mídia de Propriedade

Área para fazer upload de fotos do quarto.

Botão para selecionar arquivos para upload.

Seção Localização da Propriedade

Inserção do endereço do quarto.

Inserção do número do andar.

Inserção do código postal.

Seleção da distância por meio de um mapa ou manualmente.

Seção Cozinha

Checkboxes para selecionar as características da cozinha, como forno, fogão, micro-ondas, etc.

Seção Sala

Checkboxes para selecionar as características da sala, como área de estar compartilhada, sofás, televisão, etc.

Seção Casa de Banho

Checkboxes para selecionar as características da casa de banho, como chuveiro, toalhas, etc.

Seção Outros

Checkboxes para selecionar outras características, como máquina de lavar roupa, aquecimento central, etc.

Seção Serviços

Checkboxes para selecionar os serviços oferecidos, como Wi-Fi, elevador, limpeza, etc.

Seção Informações de Contato

Inserção do nome do contato.

Inserção do endereço de e-mail do contato.

Inserção do número de telefone do contato.

Botão de Submissão

Botão para enviar o formulário e adicionar a propriedade.

Parece ser um formulário abrangente para adicionar um novo quarto a uma propriedade, permitindo que os utilizadores forneçam detalhes específicos sobre o quarto e as suas características.

5.17.2. Back-end

```
public function AddCasa(Request $request){
    if (empty($request->input('title')) || empty($request->input('descricao')) || empty($request->input('photos'))) {
        return redirect()->back()->with('error', 'Por favor, preencha todos os campos obrigatórios.');
    }
    if (is_null($request->file('photos'))) {
        return redirect()->back()->with('error', 'Por favor, adicione pelo menos uma foto.');
    }
    for ($i = 1; $i <= $request->input('n_quartos_cont'); $i++) {
        if (empty($request->input('area_'.$i))) {
            return redirect()->back()->with('error', 'Por favor, preencha a área de todos os quartos.');
        }
    }
    $token=session('ActivationToken');
    $user = DB::table('user')->where('column: ActivationToken', $token)->first();
    $currentDate = Carbon::now();
    $currentDate->addYear();
    $formattedDate = $currentDate->format('Y-m-d');
    $propertyId = DB::table('casa_completa')->insertGetId([
        'Titulo' => $request->input('title'),
        'description' => $request->input('descricao'),
        'Tipo' => $request->input('Tipo'),
        'Genero' => $request->input('sexo'),
        'Preco' => $request->input('preco'),
        'area' => $request->input('area'),
        'estado' => 'pending',
        'data_fim'=>$formattedDate,
        'id_user'=>$user->id,
        'N_quartos'=>$request->input('n_quartos'),
    ]);
}
```

```

for ($i = 1; $i <= $request->input('n_quartos_cont'); $i++) {
    DB::table('quartos_de_casa')->insert([
        'area_quarto' => $request->input('area_'.$i),
        'roupa_de_cama' => $request->has('roupa_de_cama_.$i') ? true : false,
        'cama' => $request->has('cama_.$i') ? true : false,
        'mesa_cabeceira' => $request->has('mesa_cabeceira_.$i') ? true : false,
        'Candeeiro_de_mesa_do_estudo' => $request->has('Candeeiro_de_mesa_do_estudo_.$i') ? true : false,
        'Mesa_do_estudo' => $request->has('Mesa_do_estudo_.$i') ? true : false,
        'Janelas' => $request->has('Janelas_.$i') ? true : false,
        'Varanda' => $request->has('Varanda_.$i') ? true : false,
        'Armario' => $request->has('Armario_.$i') ? true : false,
        'Casa_de_banho_privativa' => $request->has('Casa_de_banho_privativa_.$i') ? true : false,
        'id_casa' => $propertyId,
    ]);
}

DB::table('cozinha')->insert([
    'Forno' => $request->has('Forno') ? true : false,
    'Fogao' => $request->has('Fogao') ? true : false,
    'Caldeira' => $request->has('Caldeira') ? true : false,
    'Mag_cafe' => $request->has('Mag_cafe') ? true : false,
    'Placa' => $request->has('Placa') ? true : false,
    'Micro-ondas' => $request->has('Micro-ondas') ? true : false,
    'Pratos' => $request->has('Pratos') ? true : false,
    'Utensilios' => $request->has('Utensilios') ? true : false,
    'Frigorifico' => $request->has('Frigorifico') ? true : false,
    'id_casa' => $propertyId,
]);

```

```

DB::table('banho')->insert([
    'Chuveiro' => $request->has('Chuveiro') ? true : false,
    'Toalhas' => $request->has('Toalhas') ? true : false,
    'id_casa' => $propertyId,
]);
DB::table('contato')->insert([
    'Nome' => $request->input('nome'),
    'Email' => $request->input('email'),
    'Telefone' => $request->input('title'),
    'id_casa' => $propertyId,
]);
DB::table('endereco')->insert([
    'Endereco' => $request->input('Endereco'),
    'N_andar' => $request->input('N_andar'),
    'Codigo_postal' => $request->input('Codigo_postal'),
    'Distancia' => $request->input('Distancia'),
    'let' => $request->input('letLag'),
    'id_casa' => $propertyId,
]);
foreach ($request->file('photos') as $photo) {
    $filename = $photo->store('upload', 'public');
    DB::table('midia_de_casa')->insert([
        'Path' => $filename,
        'id_casa' => $propertyId,
    ]);
}

```

```

DB::table( table: 'outros')->insert([
    'Maquina_lavar_roupa' => $request->has( key: 'Maquina_lavar_roupa') ? true : false,
    'Maquina_sacar_roupa' => $request->has( key: 'Maquina_sacar_roupa') ? true : false,
    'Aquecimento_central' =>$request->has( key: 'Aquecimento_central') ? true : false,
    'passar_Ferro' => $request->has( key: 'passar_Ferro') ? true : false,
    'Aquecedor_eletrico' => $request->has( key: 'Aquecedor_eletrico') ? true : false,
    'id_casa' => $propertyId,
]);
DB::table( table: 'sala')->insert([
    'estar_partilhada' => $request->has( key: 'estar_partilhada') ? true : false,
    'Sofas' => $request->has( key: 'Sofas') ? true : false,
    'Televisao' => $request->has( key: 'Televisao') ? true : false,
    'Mesa_jantar' => $request->has( key: 'Mesa_jantar') ? true : false,
    'id_casa' => $propertyId,
]);
DB::table( table: 'servicos')->insert([
    'Wi-Fi' => $request->has( key: 'Wi-Fi') ? true : false,
    'Elevador' => $request->has( key: 'Elevador') ? true : false,
    'Despesas' => $request->has( key: 'Despesas') ? true : false,
    'Recibo' => $request->has( key: 'Recibo') ? true : false,
    'limpeza' => $request->has( key: 'limpeza') ? true : false,
    'id_casa' => $propertyId,
]);
return redirect()->back()->with('success', 'A sua propriedade foi adicionada com sucesso!');

```

Figure 63-controlar adicionar anúncio

Este método AddCasa parece lidar com a adição de uma nova propriedade (casa). Aqui está uma análise passo a passo do que acontece no método:

Validação de Campos Obrigatórios: Verifica se todos os campos obrigatórios do formulário são preenchidos. Se algum estiver vazio, redireciona de volta com uma mensagem de erro.

Verificação de Fotos: Verifica se pelo menos uma foto foi enviada. Se não houver nenhuma foto, redireciona de volta com uma mensagem de erro.

“Loop” para Adicionar Quartos:

Itera sobre o número de quartos fornecidos no formulário.

Insere os detalhes de cada quarto na tabela quartos_de_casa.

Inserção de Detalhes da Casa:

Insere os detalhes principais da casa na tabela casa_completa.

Insere os detalhes da cozinha na tabela cozinha.

Insere os detalhes da casa de banho na tabela banho.

Insere os detalhes de contacto na tabela contacto.

Insere os detalhes de endereço na tabela endereço.

Insere as fotos na tabela midia_de_casa.

Insere os outros detalhes na tabela outros.

Insere os detalhes da sala na tabela sala.

Insere os serviços na tabela servicos.

Redirecionamento: Finalmente, redireciona de volta à página anterior com uma mensagem de sucesso.

Este método parece ser bastante abrangente, lidando com a inserção de múltiplos detalhes de quartos, características da casa, informações de contacto, serviços e muito mais.

5.17.3. Resultado

Descrição e preço do imóvel

Titulo da propriedade

Property Title

Descrição da Propriedade

Property Description

Tipo

Selecionar Tipo

Género de residência

Selecionar Sexo

Quantos quartos

Selecionar Nº Quartos

Preço

EUR

Area M₂ do quarto

m²

Mídia de propriedade

Clique aqui ou solte os arquivos para fazer upload

Localização da propriedade

Endereço

Digite seu endereço

Nº Andar

Digite seu Nº Andar

Código-postal

Digite seu Código-postal

📍 Distância

Google MAP

📍 Distância

Google MAP

Clique no mapa para definir o local:



📍 Distância é

Características da Propriedade

🍴 Cozinha

- Forno
- Fogão
- Caldeira de água
- Máquina de café
- Placa
- Micro-ondas
- Pratos e talheres
- Utensílios de cozinha
- Frigorífico

📺 Sala

- Área de estar partilhada
- Sofás
- Televisão
- Mesa de jantar com cadeiras

🛁 Casa de banho

- Chuveiro
- Toalhas

▢ Outros

- Máquina de lavar roupa
- Máquina de sacar roupa
- Aquecimento central
- máquina passar Ferro
- Aquecedor elétrico

🔧 Serviços

- Wi-Fi
- Elevador
- Despesas incluídas
- Recibo
- limpeza

Informações de contato

Nome

Email

Telefone

Adicionar propriedade

Figure 64-Adicionar anúncio

5.18. Requisitos Implementados

5.18.1. REQUISITOS

Nº	Descrição	Prioridade	Utilizadores	Implementado
A-01	Autenticação do utilizador com e-mail e palavra-passe.	Alta	Todos	Sim
A-02	Opção para o utilizador terminar a sessão.	Alta	Todos	Sim
A-03	Recuperação da palavra-passe com envio de nova palavra-passe por e-mail.	Alta	Todos	Sim
5X	Acesso à aplicação sem autenticação.	Alta	Todos	Sim

Tabela 1- Requisitos de Autenticação e Sessão

Nº	Descrição	Prioridade	Utilizadores	Implementado
G-1	Os Utilizadores poderão filtrar os anúncios das casas por: <ul style="list-style-type: none">• Distância à ESTGOH• Valor da renda• Sexo	Alto	Todos	Sim
G-2	Possibilidade de alterar nome, palavra-passe e avatar por qualquer utilizador.	Alta	Alunos/ Senhorio	Sim
G-3	Acesso à aplicação sem autenticação.	Alta	Todos	Sim

G-4	Enviar um código para email para ativar conta.	Alta	Alunos/ Senhorio	Sim
G-5	Despolarizar uma página para inserir código de ativação.	Alta	Alunos/ Senhorio	Sim
G-6	Recebimento de e-mail sobre aproximação do fim de validade de anúncio.	Alta	Senhorio	Sim
G-7	Recebimento de e-mail de aprovação de registo de conta pelos utilizadores.	Alta	Alunos/ Senhorio	Sim
G-8	Recebimento de e-mail de aprovação ou reprovação de anúncio pelos utilizadores.	Alta	Alunos/ Senhorio	Sim

Tabela 2- Requisitos Geral

Nº	Descrição	Prioridade	Utilizadores	Implementado
GT-01	criar contas de gestores sem necessidade de aprovação.	Alta	Gestor	Sim
GT-02	Edição de dados pessoais.	Alta	Gestor	Sim
GT-03	Listagem de senhorios, com possibilidade de mudar ordenação, ou pesquisa.	Alta	Gestor	Sim
GT-04	Listagem de anúncios, com possibilidade de mudar ordenação, ou pesquisa.	Alta	Gestor	Sim
GT-05	Consulta de todos os anúncios.	Alta	Gestor	Sim
GT-06	aprovAÇÃO/reprovaÇÃO de anúncios, com notificação por e-mail.	Alta	Gestor >> Senhorio	Sim
GT-07	inativAR/reativAR anúncios, com notificação por e-mail ao senhorio.	Alta	Gestor >> Senhorio	Sim
GT-08	Listagem de gestores.	Alta	Gestor	Sim
GT-09	ativAR/inativAR gestores	Alta	Gestor	Sim

Tabela 3- Requisitos Gestor

Nº	Descrição	Prioridade	Utilizadores	Implementado
S-01	Registo de senhorios na plataforma.	Alta	senhorio	Sim
S-02	Consultar o porfie.	Alta	senhorio	Sim

S-03	Edição de dados pessoais.	Alta	senhorio	Sim
S-04	Inserção de anúncios por senhorios.	Alta	senhorio	Sim
S-05	Listagem dos próprios anúncios, com possibilidade de mudar ordenação, ou pesquisa.	Alta	senhorio	Sim
S-06	Consultar dos próprios anúncios.	Alto	senhorio	Sim
S-07	Edição de anúncios próprios.	Alto	senhorio	Sim
S-09	Remoção de anúncios próprios.	Alta	senhorio	Sim
S-10	Listar às mensagens dos alunos, com possibilidade de pesquisa.	Alta	senhorio	Não
S-11	Listar às mensagens dos alunos, com possibilidade de pesquisa.	Alta	senhorio	Não
S-12	Consultar mensagens dos alunos.	Alta	senhorio	Não
S-13	Responder às mensagens dos alunos.	Alta	senhorio	Não

Tabela 4- Requisitos Senhorio

Nº	Descrição	Prioridade	Utilizadores	Implementado
AL-01	Registo de aluno na plataforma.	Alta	Aluno	Não
AL-02	enviar uma mensagem com senhorio.	Alta	Aluno	Não
AL-03	Listar às mensagens dos senhorios, com possibilidade de pesquisa.	Alta	Aluno	Não
AL-04	Consultar mensagens dos senhorios.	Alta	Aluno	Não
AL-05	Responder às mensagens dos senhorios.	Alta	Aluno	Não
AL-06	Adicionar um anúncio para lista de favoritos	media	Aluno	Não
AL-07	Listar os anúncios favoritos, com possibilidade de mudar ordenação, ou pesquisa.	media	Aluno	Não
AI-08	Consultar um anúncio de lista de favoritos	media	Aluno	Não
AL-09	Remoção de anúncios de lista de favoritos.	media	Aluno	Não

Tabela 5-Requisitos Aluno

Nº	Descrição	Prioridade	Implementado
NF-01	Encriptação das palavras-passe.	Alta	Sim
NF-02	Criação automática do Gestor na primeira execução da aplicação.	Alta	Sim
NF-03	Utilização de PHP 7, HTML 5, Laravel, MySQL para o desenvolvimento da aplicação.	Alta	Sim
NF-04	Utilização da framework Laravel para facilidade de desenvolvimento e manutenção.	Alta	Sim
NF-05	Armazenamento dos dados em uma base de dados MySQL para escalabilidade e robustez.	Alta	Sim
NF-06	Indisponibilidade de casa quando o número de quartos é zero.	Alta	Sim
NF-07	Encriptação das palavras-passe.	Alta	Sim

Tabela 6- Requisitos NÃO FUNCIONAIS

5.18.2. Requisitos planeados

Foi plantado a fazer todos requisitos.

5.18.3. Principais motivos para eventuais desvios (Sprint 1)

O principal motivo de não fazer os requisitos que foram planeados que caso de criar o fronte-end e falta em requisitos.

5.18.4. Principais motivos para eventuais desvios (Sprint 2)

O principal motivo de não fazer os requisitos que foram planeados que tive exames de outras cadeiras.

5.18.5. Lista dos diagramas

Nome de diagrama	Descrição	Implementado
Fluxograma Autenticação	processo de login dos utilizadores na plataforma	Sim
Fluxograma Registro de senhorios na plataforma	Mostra como os proprietários se registam na plataforma.	Sim
Fluxograma Gestor na aprovação/reprovação de anúncios	Define como os gestores reveem os anúncios submetidos pelos senhorios.	Sim

Fluxograma Filtragem de anúncios por alunos	Descreve como os alunos podem filtrar anúncios de propriedades.	Sim
Diagrama de Classe UML	Representação visual das classes de objetos, os atributos e relacionamentos num sistema.	Sim
Diagrama de caso de uso	Ilustra as interações entre os utilizadores e o sistema em diferentes cenários.	Sim
ER Gestor De Alojamentos	Diagrama que mostra a estrutura do banco de dados do sistema de gestão de alojamentos.	Sim

Tabela 7-Lista dos diagramas

6. Conclusões

O desenvolvimento do projeto de gestão de alojamentos para a Escola Superior de Tecnologia e Gestão de Oliveira do Hospital (ESTGOH) representou uma jornada significativa, culminando numa solução abrangente e eficiente para atender às necessidades dos alunos em busca de alojamento. Ao longo deste processo, várias conclusões importantes foram alcançadas.

Atendimento às Necessidades dos Utilizadores: A plataforma desenvolvida foi projetada para oferecer uma experiência amigável e intuitiva, facilitando a busca, reserva e comunicação entre alunos e senhorios. Com funcionalidades adaptadas às necessidades específicas dos utilizadores da ESTGOH, a solução proporciona uma resposta eficaz às demandas do público-alvo.

Utilização de Tecnologias Modernas: A escolha de tecnologias modernas, como o “framework” Laravel e o banco de dados MySQL, demonstrou ser fundamental para o desenvolvimento de uma plataforma robusta e escalável. Essas ferramentas possibilitaram a implementação de funcionalidades avançadas e a garantia da segurança e integridade dos dados dos utilizadores.

Metodologia Ágil e Iterativa: A adoção de uma metodologia de desenvolvimento ágil e iterativa permitiu uma abordagem flexível e adaptável ao longo do projeto. Isso possibilitou a entrega de incrementos funcionais em curtos períodos, possibilitando uma rápida validação das funcionalidades e a incorporação contínua do “feedback” dos utilizadores.

Importância da Avaliação Contínua: A realização de testes regulares e avaliações de usabilidade foi fundamental para garantir a qualidade e a eficácia da plataforma. Através dessas avaliações, foi possível identificar áreas de melhoria e implementar ajustes necessários para otimizar a experiência do utilizador.

Compromisso com a Privacidade e Segurança dos Dados: Num ambiente cada vez mais preocupado com questões de privacidade e segurança de dados, foi essencial garantir a conformidade com as regulamentações vigentes e adotar medidas robustas de proteção de informações sensíveis dos utilizadores.

Em suma, o projeto de gestão de alojamentos para a ESTGOH representa não apenas a conclusão bem-sucedida de um desafio académico, mas também uma contribuição tangível para a melhoria da experiência dos alunos em busca de alojamento. Ao oferecer uma plataforma eficiente e intuitiva, o projeto visa facilitar o processo de alojamento, contribuindo para a qualidade de vida e bem-estar dos estudantes.

6.1. Forças

Durante o desenvolvimento do projeto de gestão de alojamentos para a Escola Superior de Tecnologia e Gestão de Oliveira do Hospital (ESTGOH), várias forças foram identificadas, contribuindo positivamente para o sucesso e eficácia da solução desenvolvida:

Usabilidade Aprimorada: A plataforma foi projetada com foco na usabilidade, oferecendo uma “interface” intuitiva e fácil de usar para os utilizadores. Isso aumenta a eficiência e a satisfação do utilizador ao navegar, buscar e reservar alojamentos.

Tecnologias Modernas: A utilização de tecnologias modernas, como Laravel e MySQL, permitiu o desenvolvimento de uma plataforma robusta, escalável e segura. Essas tecnologias oferecem recursos avançados que suportam as necessidades do sistema e garantem uma experiência de utilizador consistente e confiável.

Agilidade no Desenvolvimento: A adoção de uma metodologia ágil e iterativa possibilitou a entrega rápida de incrementos funcionais, permitindo uma rápida validação das funcionalidades e a adaptação contínua às necessidades dos utilizadores.

Adaptação às Necessidades dos Utilizadores: A plataforma foi desenvolvida com base nas necessidades específicas dos alunos da ESTGOH, garantindo que as funcionalidades e recursos atendam adequadamente às demandas do público-alvo.

Segurança e Privacidade dos Dados: Foram implementadas medidas robustas de segurança e privacidade dos dados dos utilizadores, garantindo a proteção e confidencialidade das informações pessoais e sensíveis.

Essas forças destacam os aspectos positivos e vantagens da solução desenvolvida, demonstrando o potencial para atender às necessidades dos utilizadores e proporcionar uma experiência de gestão de alojamentos eficaz e satisfatória para os alunos da ESTGOH.

6.2. Limitações

Durante o desenvolvimento e implementação do projeto de gestão de alojamentos para a Escola Superior de Tecnologia e Gestão de Oliveira do Hospital (ESTGOH), algumas

limitações foram identificadas, as quais são importantes reconhecer e considerar para futuras melhorias e aprimoramentos:

Escopo Limitado: Devido a restrições de tempo e recursos, algumas funcionalidades desejáveis podem não ter sido totalmente implementadas ou exploradas. Isso pode incluir recursos adicionais de comunicação entre alunos e senhorios, integração com sistemas de pagamento “online”, entre outros.

Testes Limitados: Apesar dos esforços para realizar testes rigorosos, incluindo testes de unidade, integração e aceitação, o escopo e a profundidade dos testes podem ter sido limitados. Isso pode resultar em possíveis falhas ou bugs não detetados, afetando a qualidade e estabilidade da plataforma.

Limitações de “Hardware” e Infraestrutura: Dependendo dos recursos disponíveis, como capacidade de armazenamento e poder de processamento do servidor, podem surgir limitações no desempenho e escalabilidade da plataforma, especialmente em períodos de alta demanda.

Adoção e Aceitação dos Utilizadores: A aceitação e adoção da plataforma pelos utilizadores pode ser uma limitação, especialmente se houver resistência à mudança ou falta de concretização sobre os benefícios da nova solução. Isso pode impactar a eficácia e a utilidade da plataforma no longo prazo.

Questões de Privacidade e Segurança: Apesar das medidas de segurança implementadas, podem surgir preocupações adicionais relacionadas à privacidade e segurança dos dados dos utilizadores. A conformidade com regulamentações de proteção de dados, como GDPR, deve ser constantemente monitorada e aprimorada.

Ao reconhecer essas limitações, é possível direcionar esforços para mitigar os seus impactos e buscar oportunidades para melhorias futuras, garantindo uma plataforma de gestão de alojamentos mais eficaz, segura e satisfatória para os utilizadores da ESTGOH.

6.3. Trabalho Futuro

Com base nas experiências adquiridas durante o desenvolvimento do projeto de gestão de alojamentos para a Escola Superior de Tecnologia e Gestão de Oliveira do Hospital (ESTGOH), diversas oportunidades de trabalho futuro foram identificadas, visando aprimorar e expandir a plataforma para atender ainda melhor às necessidades dos utilizadores. Algumas dessas oportunidades incluem:

Implementação de Recursos Adicionais: Desenvolver e integrar novos recursos e funcionalidades à plataforma, como sistemas de pagamento “online”, calendário de disponibilidade de quartos em tempo real, avaliações e “feedback” dos utilizadores, entre outros.

Melhoria da “Interface” do Utilizador: Realizar análises de usabilidade e “design” para identificar áreas de melhoria na “interface” do utilizador e na experiência do utilizador, visando tornar a plataforma mais intuitiva, acessível e esteticamente agradável.

Expansão para Outros Dispositivos e Plataformas: Adaptar a plataforma para dispositivos móveis, como “smartphones” e “tablets”, garantindo uma experiência consistente e otimizada em diferentes dispositivos e sistemas operativos.

Integração com Outros Sistemas: Integrar a plataforma com outros sistemas e serviços utilizados pela ESTGOH, como sistemas de gestão académico e portal do aluno, para oferecer uma experiência integrada e coesa aos utilizadores.

Análise de Dados e Inteligência Artificial: Utilizar técnicas de análise de dados e inteligência artificial para obter intuições e prever tendências relacionadas à demanda por alojamentos, otimizando a alocação de recursos e melhorando a experiência dos utilizadores.

Ampliação da Base de Utilizadores: Expandir o alcance da plataforma para além dos alunos da ESTGOH, tornando-a disponível para outras instituições de ensino superior ou mesmo para o mercado de alojamentos em geral.

Essas são apenas algumas das oportunidades de trabalho futuro que podem ser exploradas para aprimorar e expandir a plataforma de gestão de alojamentos. Ao continuar a investir em inovação e desenvolvimento contínuo, é possível garantir que a plataforma permaneça relevante e eficaz, atendendo às necessidades em constante evolução dos utilizadores da ESTGOH e além.

7. Referências

Neste projeto todo foi utilizado um template com base de html e css e depois foi alterado a dependência que precisava.

E foi utilizado outros sites para alguns códigos ou ícones ou fotos que foi necessário obter.

7.1. Lista de Referência

Para tabelas: <https://colorlib.com/wp/css3-table-templates/>

Para icons: <https://icons8.com/icons>, <https://fontawesome.com/v4/icons/>

Para todas páginas: [250+ Free Website Templates - TemplateOnWeb](#)

Para teste(alertas): <https://alvarotrigo.com/blog/css-alerts>

