Uma imagem com texto

Descrição gerada automaticamente

**Licenciatura em Engenharia Informática**

**Manual Programador**

**Gestor de Alojamentos**

**Muaiad Mhd Fahd Al Hadad**

Oliveira do Hospital, maio de 2024

Conteúdo

[1.Introdução 5](#_Toc166272609)

[1.2 Objetivo do Projeto 5](#_Toc166272610)

[1.3 Tecnologias Utilizadas 5](#_Toc166272611)

[1.4 Funcionalidades Principais 5](#_Toc166272612)

[1.5 Segurança e Privacidade 5](#_Toc166272613)

[2. install project 6](#_Toc166272614)

[2.1 Fazer clone ao projeto de GitHub 6](#_Toc166272615)

[2.2 Instar Laravel 6](#_Toc166272616)

[2.3 Configurar o Banco de Dados 6](#_Toc166272617)

[2.4 Executar o Servidor de Desenvolvimento 6](#_Toc166272618)

[3. desenvolvimento 6](#_Toc166272619)

[3.1 Página Inicial 6](#_Toc166272620)

[3.1.1 Front end 6](#_Toc166272621)

[7](#_Toc166272622)

[3.1.2 Back end 14](#_Toc166272623)

[3.2 Detalhe 19](#_Toc166272624)

[3.2.1 Front end 19](#_Toc166272625)

[3.2.2 Back-end 23](#_Toc166272626)

[3.3 Login 24](#_Toc166272627)

[3.3.1 Front end 24](#_Toc166272628)

[3.3.2Back-end 25](#_Toc166272629)

[3.4 Registar 28](#_Toc166272630)

[3.4.1 front end 28](#_Toc166272631)

[3.4.2 Back-end 29](#_Toc166272632)

[2.5 Email enviados 31](#_Toc166272633)

[2.5.1 Back-end 31](#_Toc166272634)

[3.6 Validação código 32](#_Toc166272635)

[3.6.1 Back-end 32](#_Toc166272636)

[3.7 Recuperar a Password 35](#_Toc166272637)

[3.7.1 Back-end 35](#_Toc166272638)

[3.8 Gestor área pessoal 36](#_Toc166272639)

[3.8.1 Front-end 36](#_Toc166272640)

[3.8.2 Back-end 37](#_Toc166272641)

[3.9 Adicionar 39](#_Toc166272642)

[3.9.1 Back-end 39](#_Toc166272643)

[3.10 Profile 40](#_Toc166272644)

[3.10.1 Back-end 40](#_Toc166272645)

[3.11 Back-end remover utilizador 42](#_Toc166272646)

[3.12 Back-end consultar anúncio 44](#_Toc166272647)

[3.13 Back-end aprovar e reprovar anúncio 46](#_Toc166272648)

[3.14 Back-end mudar estado de um anúncio 47](#_Toc166272649)

[3.15 Senhorio área pessoal 48](#_Toc166272650)

[3.15.1 Back-end 48](#_Toc166272651)

[3.16 Editar anúncio 49](#_Toc166272652)

[3.16.1 Back-end 49](#_Toc166272653)

[3.17 Adicionar anúncio 53](#_Toc166272654)

[3.17.1 Front-end 53](#_Toc166272655)

[3.17.2 Back-end 57](#_Toc166272656)

[3.18 ControllerChat 60](#_Toc166272657)

[3.18.1 Métodos 60](#_Toc166272658)

[2.18.2 Back end 61](#_Toc166272659)

[3.19 Adicionar para os favoritos 62](#_Toc166272660)

[3.19.1 back end 62](#_Toc166272661)

Conteúdo das figuras

[Figure 1-Base Dados 6](#_Toc166272662)

[Figure 2- cabeçalho da página com login e sem 7](#_Toc166272663)

[Figure 3- Java Script de menu de utilizador 8](#_Toc166272664)

[Figure 4- Lógica de mostra os utilizadores 9](#_Toc166272665)

[Figure 5- filtragem 11](#_Toc166272666)

[Figure 6- As ofertas 13](#_Toc166272667)

[Figure 7- recuperar os dados das propriedades imobiliárias para exibição na página inicial 14](#_Toc166272668)

[Figure 8- exibir a página inicial do site 16](#_Toc166272669)

[Figure 9- Filtragem 18](#_Toc166272670)

[Figure 10- fotos de propriedade 20](#_Toc166272671)

[Figure 11-específicos da propriedade 21](#_Toc166272672)

[Figure 12-Map 22](#_Toc166272673)

[Figure 13-recuperar os detalhes de uma casa específica 23](#_Toc166272674)

[Figure 14-login form 24](#_Toc166272675)

[Figure 15-toast 25](#_Toc166272676)

[Figure 16-controller de login 26](#_Toc166272677)

[Figure 17-form registar 28](#_Toc166272678)

[Figure 18-controler registar 29](#_Toc166272679)

[Figure 19-controlar email 32](#_Toc166272680)

[Figure 20-controlar ativação de utilizador 33](#_Toc166272681)

[Figure 21-controlar reenviar código de ativação 34](#_Toc166272682)

[Figure 22-controlar Forgot Password 35](#_Toc166272683)

[Figure 23-controlar de página principal de gestor 38](#_Toc166272684)

[Figure 24-controlar add gestor 39](#_Toc166272685)

[Figure 25-dados do utilizador 40](#_Toc166272686)

[Figure 26-update dados de utilizador 41](#_Toc166272687)

[Figure 27-remover utilizador 42](#_Toc166272688)

[Figure 28-consultar anúncio 44](#_Toc166272689)

[Figure 29-Aprovar e reprovar anúncio 46](#_Toc166272690)

[Figure 30-mudar estado de um anúncio 47](#_Toc166272691)

[Figure 31-controlar área pessoal senhorio 48](#_Toc166272692)

[Figure 32-enviar dados de anúncio para página de edit 50](#_Toc166272693)

[Figure 33-submeter editar 52](#_Toc166272694)

[Figure 34-Html adicionar anúncio 55](#_Toc166272695)

[Figure 35-controlar adicionar anúncio 59](#_Toc166272696)

[Figure 36-Chat 61](#_Toc166272697)

[Figure 37-enviar chat 62](#_Toc166272698)

[Figure 38- favoritos 62](#_Toc166272699)

# 1.Introdução

Bem-vindo ao manual de desenvolvimento para o projeto de gestão de alojamentos da Escola Superior de Tecnologia e Gestão de Oliveira do Hospital (ESTGOH). Este manual destina-se a fornecer orientações detalhadas sobre a implementação e manutenção do sistema, desenvolvido utilizando PHP Laravel, HTML, JavaScript, MySQL e CSS.

## 1.2 Objetivo do Projeto

O projeto visa criar uma solução abrangente e intuitiva para a gestão de alojamentos, direcionada especificamente aos alunos da ESTGOH. O sistema facilitará a busca, reserva e comunicação entre estudantes e senhorios, proporcionando uma plataforma centralizada e eficaz.

## 1.3 Tecnologias Utilizadas

O projeto utiliza uma combinação de linguagens e tecnologias web, incluindo:

* PHP Laravel: “Framework” PHP moderno e poderoso para o desenvolvimento de aplicativa web.
* HTML: Linguagem de marcação para a criação da estrutura da “interface” do utilizador.
* JavaScript: Linguagem de programação para tornar a “interface” do utilizador interativa e dinâmica.
* MySQL: sistema de gestão de banco de dados relacional para armazenamento e recuperação de informações críticas.
* CSS: Linguagem de estilo para definir a apresentação e o layout da “interface” do utilizador.

## 1.4 Funcionalidades Principais

Entre as funcionalidades principais do sistema estão:

* Busca de quartos disponíveis com critérios específicos (localização, preço, comodidades).
* Comunicação direta entre alunos e senhorios para negociações e esclarecimento de dúvidas.
* Painel de controle para senhorios gerirem as suas propriedades e responderem às consultas dos alunos.

## 1.5 Segurança e Privacidade

A segurança e a privacidade dos dados dos utilizadores são prioridades essenciais. O sistema foi desenvolvido conforme as regulamentações de proteção de dados para garantir a segurança das informações dos utilizadores.

# 2. install project

## 2.1 Fazer clone ao projeto de GitHub

Clonar o Projeto do GitHub: abra o terminal ou prompt de comando e navegue até o diretório onde deseja armazenar o projeto. Em seguida, execute o seguinte comando para clonar o repositório do GitHub:

* git clone <https://github.com/MuaiadHadad/Project_Gestor_Alojamento>

## 2.2 Instar Laravel

Instalar o Laravel: Antes de instalar o Laravel, certifique-se de que possui o Composer instalado. Se ainda não o tiver, pode baixá-lo e instalá-lo em https://getcomposer.org/. Após instalar o Composer, navegue até o diretório do projeto clonado e execute o seguinte comando para instalar as dependências do Laravel:

* composer install

## 2.3 Configurar o Banco de Dados

Renomeie o arquivo .env.example para .env no diretório raiz do projeto.

Abra o arquivo .env e configure as informações do banco de dados, incluindo DB\_CONNECTION, DB\_HOST, DB\_PORT, DB\_DATABASE, DB\_USERNAME e DB\_PASSWORD.

Texto

Descrição gerada automaticamente

Figure 1-Base Dados

Não esquece de instalar as tabelas que estão em anexo.

## 2.4 Executar o Servidor de Desenvolvimento

Após instalar as dependências do Laravel e configurar o banco de dados, pode iniciar o servidor de desenvolvimento executando o seguinte comando:

php artisan serve

# 3. desenvolvimento

## 3.1 Página Inicial

### 3.1.1 Front end

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

### Texto Descrição gerada automaticamente

Figure 2- 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 (<img>) 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.

Texto

Descrição gerada automaticamente

Figure 3- 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 o 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 o 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.

Texto

Descrição gerada automaticamente

Figure 4- 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 (<a href="...">) para diferentes partes do sítio “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.

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.

Texto

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente

Figure 5- 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.

Texto

Descrição gerada automaticamente com confiança baixa

Figure 6- 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 (<p>) 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.

### 3.1.2 Back end

Texto

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente

Figure 7- recuperar 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 sítio 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.

Texto

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente

Figure 8- 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.

Texto

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente

Figure 9- 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 sítio 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 sítio 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.

## 3.2 Detalhe

### 3.2.1 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.

Texto

Descrição gerada automaticamente

Figure 10- 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.

Texto

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente

Figure 11-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.

Texto

Descrição gerada automaticamente

Figure 12-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.

### 3.2.2 Back-end

Texto

Descrição gerada automaticamente

Figure 13-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.

## 3.3 Login

### 3.3.1 Front end

Texto

Descrição gerada automaticamente

Figure 14-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.

Texto

Descrição gerada automaticamente

Figure 15-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.

### 3.3.2Back-end

Texto

Descrição gerada automaticamente

Figure 16-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.

## 3.4 Registar

### 3.4.1 front end

Texto

Descrição gerada automaticamente

Figure 17-form registar

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.

### 3.4.2 Back-end

Texto

Descrição gerada automaticamente

Figure 18-controler registar

Este trecho de código PHP está relacionado ao processo de registo 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.

## 2.5 Email enviados

### 2.5.1 Back-end

### 

Texto

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente

Figure 19-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.

## 3.6 Validação código

### 3.6.1 Back-end

Tela de computador com fundo verde

Descrição gerada automaticamente

Figure 20-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.

Texto

Descrição gerada automaticamente

Figure 21-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.

## 3.7 Recuperar a Password

### 3.7.1 Back-end

Texto

Descrição gerada automaticamente

Figure 22-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.

## 3.8 Gestor área pessoal

### 3.8.1 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.

### 3.8.2 Back-end

Texto

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente

Figure 23-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.

## 3.9 Adicionar

### 3.9.1 Back-end

Texto

Descrição gerada automaticamente

Figure 24-controlar add gestor

Esta função Adicionargestor 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.

## 3.10 Profile

### 3.10.1 Back-end

Texto

Descrição gerada automaticamente

Figure 25-dados do utilizador

Texto

Descrição gerada automaticamente

Figure 26-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.

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 /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.

## 3.11 Back-end remover utilizador

Texto

Descrição gerada automaticamente

Figure 27-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.

## 3.12 Back-end consultar anúncio

Texto

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente

Figure 28-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.

## 3.13 Back-end aprovar e reprovar anúncio

Texto

Descrição gerada automaticamente

Figure 29-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.

## 3.14 Back-end mudar estado de um anúncio

Texto

Descrição gerada automaticamente

Figure 30-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.

## 3.15 Senhorio área pessoal

### 3.15.1 Back-end

Texto

Descrição gerada automaticamente

Figure 31-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.

## 3.16 Editar anúncio

### 3.16.1 Back-end

Texto

Descrição gerada automaticamente

Figure 32-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.

Texto

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente

Figure 33-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.

## 3.17 Adicionar anúncio

### 3.17.1 Front-end

Texto

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente

Figure 34-Html adicionar anúncio

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

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 do 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.

### 3.17.2 Back-end

Texto

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente

Figure 35-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.

## 3.18 ControllerChat

A classe ControllerChat é uma classe controladora que contém métodos para lidar com a lógica de negócios relacionada a chats entre alunos e senhorios.

### 3.18.1 Métodos

CriarChatQuarto($id): Este método é responsável por criar ou obter um chat relacionado a um quarto específico. Ele verifica se o utilizador é um aluno, obtém informações do utilizador atualmente autenticado, verifica se já existe um chat para o quarto especificado e, em seguida, redireciona para a página de conversa com os dados necessários.

CriarChatCasa($id): Similar ao método anterior, mas para criar ou obter um chat relacionado a uma casa completa.

GetChat($id): Este método é usado para obter as mensagens de um chat específico. Ele verifica se o utilizador é um aluno ou senhorio, obtém informações sobre o chat e as mensagens relacionadas a ele e atualiza o estado do chat para indicar que foi visualizado.

sendMessage(Request $request, $id): Este método é responsável por enviar uma mensagem para um chat específico. Ele recebe a mensagem do formulário enviado pelo utilizador, insere a mensagem no banco de dados e redireciona de volta para a página de conversa.

### 2.18.2 Back end

Texto

Descrição gerada automaticamente

Tela de computador com jogo

Descrição gerada automaticamente

Figure 36-Chat

$verfiy : Esta parte do código realiza uma consulta à tabela de chats para verificar se já existe um chat entre o aluno (id\_aluno) e o senhorio (id\_senhorio) específicos. Ele usa o objeto DB do Laravel para interagir com o banco de dados e a função first() para obter apenas o primeiro resultado da consulta.

$conversas:Esta parte da lógica é para recuperar todas as conversas associadas ao aluno logado. Realiza uma junção com a tabela user para obter informações sobre o senhorio, seleciona os campos relevantes e filtra pela identificação do aluno.

Aqui, é feita uma subconsulta para combinar todas as mensagens relacionadas ao chat em uma única coleção de mensagens. São selecionados os campos relevantes e as mensagens são ordenadas por tempo de envio.

$mensagens\_combinadas: aqui, é feita uma subconsulta para combinar todas as mensagens relacionadas ao chat numa única coleção de mensagens. São selecionados os campos relevantes e as mensagens são ordenadas por tempo de envio.

Texto

Descrição gerada automaticamente

Figure 37-enviar chat

Esta parte do código insere a mensagem no banco de dados. A mensagem é associada ao chat específico através do campo id\_chat. O campo Estado é definido como '0' para indicar que a mensagem ainda não foi vista. O campo TimeAlun recebe a data e hora atual.

## 3.19 Adicionar para os favoritos

### 3.19.1 back end

Texto

Descrição gerada automaticamente

Figure 38- favoritos

Este método é responsável por adicionar um quarto aos favoritos de um utilizador. Ele primeiro verifica se o utilizador está autenticado como um aluno, obtém o utilizador correspondente ao token de ativação na sessão e, em seguida, insere um novo registo na tabela de favoritos associando o utilizador ao quarto específico. Se o utilizador não estiver autenticado como aluno, ele será redirecionado para a página de login.