

Desenvolvimento da Aplicação “Student.Home”

(Anúncio e Gestão de uma Plataforma para arrendamento de Imóveis para estudantes)

Nuno Barbosa Leão Beça e Silva

Nº 11589 – Regime Laboral

Orientação

Joaquim José de Almeida Soares Gonçalves

Ano Letivo 2018/2019

Licenciatura em Engenharia de Sistemas Informáticos

Escola Superior de Tecnologia

Instituto Politécnico do Cávado e Ave

Identificação do Aluno

Nuno Barbosa Leão Beça e Silva

Aluno número 11589, regime laboral

Licenciatura em Engenharia de Sistemas Informáticos

Orientação

Joaquim José de Almeida Soares Gonçalves

Professor Convidado

Resumo

Este relatório tem como objetivo documentar a informação do trabalho realizado durante a elaboração do projeto final de curso do seu autor.

Este projeto é realizado no âmbito da conclusão da Licenciatura em Engenharia de Sistemas Informáticos, do Instituto Politécnico do Cávado e Ave.

Este projeto de final de curso, consistiu na elaboração de uma plataforma online, simples e acessível para estudantes que procurem casa, quarto ou apartamento para residir durante os seus anos universitários. Foi desenvolvido também uma outra plataforma de “*back-office*” de forma a que o administrador do website consiga controlar esse website.

Para atingir esse objetivo, defini que se devia desenvolver uma plataforma WEB, as razões para tal decisão passaram pelo facto de este website poder ser acedido através de qualquer *browser*, não sendo necessário instalar a mesma.

No entanto, para aceder a esta aplicação é necessário estar ligado à *internet*, bem como pela mobilidade “*cross-plataform*”, podendo ser esta aplicação ser executada em mais do que uma plataforma, Windows, Linux, etc...

De forma a criar um plano simples do que seria a aplicação, recorri a processos de Engenharia de *Software* para garantir a organização e o bom funcionamento do projeto.

Por isso decidi comprar um curso online de PHP, aprender, e aplicar os conhecimentos aprendidos nesse curso online, desenvolvendo este projeto.

A escolha da linguagem PHP (com arquitetura MVC) foi feita devido á sua simplicidade, e a sua capacidade de gerar conteúdo dinâmico na WEB e também porque foi criada, precisamente, para fazer este tipo de projetos.

Abstract

The aim of this report is to document the information of the work done during the preparation of the final course project of its author.

This project is carried out under the completion of the Degree in Computer Systems Engineering, at Instituto Politécnico do Cávado e Ave.

This end-of-course project consisted in the development of a simple and accessible online platform for students looking for a house, room or apartment to live in during their university years. Another back-office platform was also developed so that the website administrator can control this website.

To achieve this goal, I defined that a WEB platform should be developed, the reasons for that decision were that this website can be accessed through any browser, not being necessary to install it.

However, in order to access this application, it is necessary to be connected to the internet, as well as by the "cross-platform" mobility, and this application can be run on more than one platform, Windows, Linux, etc....

In order to create a simple plan of what the application would be, I resorted to Software Engineering processes to ensure the organization and proper functioning of the project.

So, I decided to buy an online PHP course, learn, and apply the knowledge learned in this online course, developing this project.

The choice of PHP language (with MVC architecture) was made due to its simplicity, and its ability to generate dynamic content on the Web.

Agradecimentos

Este projeto é o culminar de 4 anos, de aprendizagem e trabalho, e nestes 4 anos, existiram algumas pessoas que foram absolutamente determinantes para a conclusão do mesmo, pessoas essas que quero prestar os meus sinceros agradecimentos.

Gostaria de começar por agradecer ao docente e meu orientador, Joaquim José de Almeida Soares Gonçalves, que no momento da proposta para me acompanhar neste projeto, aceitou sem “pensar duas vezes”, agradecer também, pela sua disponibilidade, empenho e conhecimento que foram muito importantes para conseguir concluir este projeto.

A todos os docentes que lecionaram as disciplinas do meu curso, todos esses docentes estão, de uma maneira ou outra, representados neste projeto, pois, foram eles que me deram conhecimentos fulcrais e sem esses conhecimentos era impossível chegar a este ponto.

Aos meus pais e à minha irmã, por serem os meus pilares, por me motivarem e por acreditarem em mim, mesmo quando eu não o acreditava, e achava, erradamente, que se calhar este curso não era para mim.

A todos os amigos, em especial à Matilde, Nádia, José, Gonçalo, Mário, Pedro e o outro Pedro Sanches, pela entreaajuda, companheirismo, apoio durante estes primeiros anos no ensino superior, não esquecendo as tardes e as madrugadas passadas a trabalhar para que tudo saísse bem.

Em especial à Inês, por tudo, pelos altos e baixo, mas agradecer, especialmente por estar lá.

Índice

Desenvolvimento da Aplicação “Student.Home” (Anúncio e Gestão de uma Plataforma para arrendamento de Imóveis para estudantes).....	1
Resumo	6
Abstract.....	8
Agradecimentos.....	10
Índice.....	12
INDÍCE DE IMAGENS	15
Siglas e Acrónimos	17
1. -Introdução	18
1.1. - Objetivos.....	19
1.1. -Estrutura do Documento.....	19
1. - Introdução	19
2. - Materiais e Métodos	20
3. - Análise e Modelação do Sistema	20
4. - Implementação.....	20
5. - Conclusão.....	20
2. - Materiais e Métodos	21
2.1. - Frameworks, Linguagens e Paradigmas.....	22
2.1.1. - Orientação de Objeto	22
2.1.1.1. – Classes	22
2.1.1.2. - Objeto	23
2.1.1.3. – Herança	23
2.1.1.4. – Polimorfismo.....	23
2.1.2. – Linguagem PHP	24
2.1.3. – Bootstrap	24
2.1.4. - JQuery	25
2.1.5. - Composer	25
2.1.6. - PHP Mailer	26
2.1.7. - Rain TPL.....	26
2.1.8. – AdminLTE	26

2.1.9.	- Slim Framework.....	27
2.1.10.	- Arquitetura de Software	27
2.1.10.1.	- MVC.....	28
2.2.	- Aplicações	29
2.2.1.	– Sublime Text 3	29
2.2.2.	– Apache.....	30
2.2.3.	– XAMPP.....	31
2.2.4.	– MySql Workbench 6.3	31
2.2.5.	-Visual Paradigm.....	32
3.1.	– Cascata	33
3.1.1.	– Vantagens do Modelo	34
3.1.2.	– Desvantagens do Modelo.....	34
3.2.	– Identificação dos Requisitos.....	35
3.2.1.	– Análise de Requisitos.....	35
3.2.2.	– Técnicas de Análise de Requisitos.....	36
3.2.2.1.	– Questionários.....	36
3.2.2.2.	– Entrevista	37
3.2.2.3.	– Brainstorming	37
3.2.3.	– Tipo de Requisitos	37
3.2.4.	– Definição de Prioridades -Técnica Usada.....	38
3.2.4.1.	– MUST HAVE (MUST)	38
3.2.4.2.	– SHOULD HAVE (SHOULD).....	39
3.2.4.3.	– COULD HAVE (COULD)	39
3.2.4.4.	– WOULD LIKE (WOULD).....	39
3.2.5.	– Vantagens da Técnica MoSCoW	39
3.2.6.	– Desvantagens da Técnica MoSCoW	40
3.2.7.	– Atores.....	40
3.2.7.1.	- Administradores	40
3.2.7.2.	- Cliente	41
3.2.7.3.	– Visitante	41
3.3.	– Requisitos Funcionais	41
3.3.1.	– Requisitos Funcionais <i>BackOffice</i>	41
3.3.2.	- Requisitos funcionais – Website	45

3.4.	- Requisitos Não Funcionais	47
3.5.	- Casos de Uso	49
3.5.1.	- Controlo de Acesso ao Sistema	49
3.5.2.	- Imóveis	51
3.5.3.	- Categorias	52
3.6.	- Diagrama de Classes	53
3.7.	- Diagrama Entidade-Relação (Base de Dados).....	54
3.8.	- Criação da Base de Dados.....	56
4.	- Implementação	57
4.1.	- Estruturas de Dados.....	57
4.1.1.	- Arrays	57
4.1.2.	- Class	57
4.1.3.	- Variables	58
4.1.4.	- Constants	59
4.2.	- Aplicação	59
4.2.1.	- <i>BackOffice</i>	59
4.2.1.1.	- Acesso á Aplicação	59
	Nota:.....	60
4.2.1.1.1.	- Esquecimento Da Palavra-Passe	61
4.2.1.2.	- <i>Dashboard</i> do Backoffice	62
4.2.1.3.	- Gestão das Contas	63
4.2.1.4.	- Gestão de Categorias	65
4.2.1.5.	- Gestão de Imóveis	67
4.2.2.	- Website	70
4.2.2.1.	- Imóveis	71
4.2.2.2.	- Conta de Utilizador	74
4.2.2.3.	- Formulário de Contacto	75
5.	- Conclusão	76
	Bibliografia	77

ÍNDICE DE IMAGENS

Figura 1- Funcionamento da arquitetura MVC	29
Figura 2-Modelo Cascata.....	35
Figura 3-Casos de Uso de Controlo de Acesso ao Sistema.....	50
Figura 4-Caso de Uso dos Imóveis.....	51
Figura 5-Casos de Uso das Categorias.....	52
Figura 6-Diagrama de Classes.	53
Figura 7-Diagrama-Entidade Relação criado em MySql Workbench.	55
Figura 8-Tabelas da Base de Dados do Projeto.	56
Figura 9-Stored Procedures do Projeto.....	56
Figura 10-Exemplo de Declaração de Arrays.....	57
Figura 11-Exemplo de uma classe em PHP com variáveis e as propriedades da classe.	58
Figura 12- Declaração e uso das variáveis através da função echo.	58
Figura 13- Declaração de uma constante em PHP fora da classe (em cima) e dentro da classe (em baixo).	59
Figura 14- Acesso á Aplicação	60
Figura 15- Ficheiro Hosts	60
Figura 16- Pedido de E-mail para redefinição de palavra-passe.	61
Figura 17- Mail com o link para redefinir Palavra-Passe.	61
Figura 18- Redefinição da Palavra-Passe.....	62
Figura 19- Dashboard	62
Figura 20- Lista dos Utilizadores registados no website.	63
Figura 21- Registo de um novo Utilizador	63
Figura 22- Edição de um utilizador.....	64
Figura 23- Eliminar o Utilizador.....	64
Figura 24- Alterar Palavra-Passe.....	65
Figura 25- Lista de Categorias de Imóveis.	65
Figura 26- Registo de uma nova Categoria de Imóvel.	66
Figura 27- Edição de uma Categoria.	66
Figura 28- Gerir Categorias dos Imóveis.	67
Figura 29- Lista dos Imóveis existentes.	68
Figura 30- Registo de um novo Imóvel.....	68
Figura 31- Edição de Imóveis.....	69
Figura 32- Página Principal do website, parte de cima.	70
Figura 33- Página Principal do website, parte de baixo.	70
Figura 34- Casas registadas na Base de dados.	71
Figura 35- Detalhes de um imóvel através de um visitante.....	72
Figura 36- Imóveis Observados pelo utilizador.	72
Figura 37-Formulário de contacto para avaliar imóvel.....	73
Figura 38- Editar dados da Minha Conta.....	74

Figura 39- Alterar Palavra-Passe.	74
Figura 40– Formulário de Contacto.	75

Siglas e Acrónimos

API – Application Programming Interface

CRUD- Create, Read, Update e Delete

CSS - Cascading Style Sheets

DB- DataBase

FTP – File Transfer Protocol

HTML- HyperText Markup Language

HTTP – HyperText Transfer Protocol

IPCA – Instituto Politécnico do Cávado e do Ave

JSON – JavaScript Object Notation

MVC – Model, View, Controller

PHP - Hypertext Preprocessor

POO- Programação Orientada a Objetos

SMTP- Simple Mail Transfer Protocol

SQL – Structured Query Language

WWW –World Wide Web

XML - eXtensible Markup Language

1. -Introdução

Ano passado tinha chegado a altura de alguns amigos meus entrarem para a Universidade, o processo de decisão de qual o curso que queremos, o local onde vamos tirar esse curso é sem dúvida, muito exaustivo, quer a nível psicológico, quer a nível físico.

Após sabermos os resultados das colocações do ensino superior e caso sejamos colocados em locais longe de casa, segue-se a seguinte etapa: procurar um sítio para ficar a viver.

Esta segunda etapa, achava eu, que era algo fácil, mas estava enganado. O processo de escolher a casa, apartamento, quarto, é ridiculamente difícil.

Primeiro porque não existe um sítio onde os estudantes possam ver, em tempo real, quartos, apartamentos, e por isso, exige aos mesmos, uma ida ao local e uma procura exaustiva pela zona, de casas para alugar.

Depois, porque, em sites online, muitas vezes os anúncios estão desatualizados e esses imóveis já nem estão à espera de serem alugados, ou então temos de pagar um valor para conseguir saber o número do proprietário de certo imóvel.

Sendo assim e após deparar-me com esta situação, decidi, com a supervisão e orientação do Professor Joaquim José de Almeida Soares Gonçalves fazer um website, simples, dedicado somente a estudantes, para encontrarem o apartamento, o quarto, a casa na sua zona, de forma cómoda e de confiança e acima de tudo gratuita e sempre atualizada.

1.1. - Objetivos

Este projeto final de curso, teve como principal objetivo o desenvolvimento de um website denominado “Student.Home” que é capaz de mostrar os imóveis a alugar para os estudantes.

De forma a conseguir gerir este site foi também criado um *back office* capaz de gerir, os utilizadores, os imóveis e as categorias de imóveis da aplicação.

Neste *back office* é também adicionado algumas estatísticas, como os últimos utilizadores adicionados, últimos imóveis etc.

A mais-valia da utilização desta aplicação e o seu propósito é permitir que os estudantes do ensino superior procurem alguns imóveis de forma cómoda, procurar imóveis da sua zona, visualizar os imóveis por categoria (apartamento, casa, quarto).

É também possível ao utilizador normal (sem permissão de administrador) conseguir enviar um email para a administração da aplicação de forma a conseguir adicionar um imóvel que seja proprietário.

Futuramente, poderia ser implementada a funcionalidade de permitir aos utilizadores adicionarem os seus próprios imóveis.

1.1. -Estrutura do Documento

Este documento está estruturado de forma planeada e organizada para que seja possível consolidar, documentar e resumir os conteúdos e o trabalho realizado:

1.- Introdução

É feita a apresentação do Projeto de final de curso, dando uma contextualização do problema, apresentação da ideia que deu azo à aplicação, assim como os objetivos definidos para este projeto.

2.- Materiais e Métodos

É identificado as plataformas, *frameworks*, linguagens e paradigmas assim como as aplicações (software) usado na criação deste projeto.

3.- Análise e Modelação do Sistema

É apresentada a modelação do sistema através da identificação de requisitos, diagramas de casos de usos, diagramas de classes, e diagrama de implementação.

Algumas arquiteturas foram desenvolvidas ao longo do projeto ao nível da Engenharia de *Software*.

4.- Implementação

Detalhes sobre a implementação, particularmente das estruturas de dados, manipulação de dados e interfaces.

5.- Conclusão

É feito um apanhado final deste relatório, assim como do trabalho realizado ao longo do projeto.

2. - Materiais e Métodos

De forma a criar esta aplicação WEB foi necessário definir quais os materiais e os métodos a serem usados, assim como a linguagem de programação, base de dados.

Essas plataformas, *frameworks*, linguagens, paradigmas e *software* usado neste trabalho vão ser documentadas e detalhadas ao longo deste capítulo.

A escolha das mesmas é o reflexo da aprendizagem adquirida ao longo do curso.

No entanto, a linguagem em específico, foi uma escolha pessoal por ser uma linguagem criada para atuar do lado do servidor, assim como uma linguagem apropriada para este tipo de aplicações.

A aprendizagem desta linguagem foi feita, há um ano, através de um curso online certificado pelo site Udem¹.

e pela Hcode², empresa brasileira de cursos³ de alto nível online e presencial.

Foi também na disciplina de “Multimédia e Tecnologia Web” que a aplicação começou a fazer sentido na minha cabeça, passando mais tarde para a aplicação final do relatório e do trabalho realizado ao longo do projeto

¹ “Udem” 2019. [Online]. Disponível em: <https://www.udemy.com>

² “Hcode” 2019. [Online]. Disponível em: <https://www.hcode.com.br/>

³ “Curso de PHP – Hcode” 2019. [Online]. Disponível em: <https://www.udemy.com/curso-php-7-online/>

2.1. - *Frameworks*, Linguagens e Paradigmas

2.1.1. - Orientação de Objeto

A POO, é uma técnica de programação que utiliza conceitos de Objetos e Classes como elementos centrais para conseguir representar e processar os conceitos dados usados em ambientes de *software*.

Resumidamente, a POO foi criada para tentar recriar o mundo real, num ambiente computacional.

2.1.1.1. – Classes

Uma classe é uma estrutura que abstrai um conjunto de objetos que contêm características similares entre si.

O **comportamento desses objetos** é definido pela classe.

Para definir esse comportamento dos objetos, é necessário criar **métodos**, **esses métodos** modificam o estado desses mesmo atributos.

Resumidamente, uma classe usa os objetos como forma de “instanciar-se”, esses objetos irão ter **atributos** e **métodos**.

Esses **atributos** são características da classe e conseqüentemente são atributos de todos os objetos derivados dessa classe.

Por sua vez, os **métodos** operações possíveis de uma classe.

2.1.1.2. - Objeto

Um objeto é uma “instância da classe” e são a base deste paradigma que é a POO.

Os objetos podem ter vários tipos de representação, podem ser representados por entidades do mundo real, como pessoas, veículos etc. assim como outros conceitos como gráficos etc.

O objeto é constituído por características próprias, que nós chamamos de atributos, e executa certas ações que são definidas através dos métodos. Estes atributos e métodos são definidos na classe que originou o objeto.

2.1.1.3. – Herança

Uma herança, em POO, é a capacidade de um objeto de uma certa classe, adquirir atributos e operações de um outro objeto de uma outra classe.

É interessante o uso de heranças devido á possibilidade de reutilização de código que já foi implementado em classes anteriormente, assim é reduzido o esforço no desenvolvimento do trabalho.

Resumidamente uma herança consegue evitar a repetição na definição de classes com características semelhantes que estão relacionadas entre si. É um relacionamento entre classes, que permite que haja uma partilha de atributos e métodos de outra classe.

2.1.1.4. – Polimorfismo

Polimorfismo é a capacidade que uma classe tem de se sobrepor a métodos da classe abstrata, modificando-os. Para isso, é necessário que ela possua a mesma assinatura de método.

(Souza, 2016)

O polimorfismo ocorre quando um objeto tem um comportamento diferente para uma mesma ação.

2.1.2. – Linguagem PHP

PHP- Hypertext Preprocessor é uma linguagem criada em 1994 por Rasmus Lerdof, a primeira versão era um simples conjunto de binários *Common Gateway Interface (CGI)* escritos em linguagem de programação C.

Rasmus Lerdof inicialmente usou o PHP para conseguir obter o número de visitas ao seu currículo online.

Com o passar do tempo mais funcionalidades foram adicionadas, e foram criadas várias versões.

A versão usada neste projeto é a 7, PHP 7, mais especificamente a versão **7.2.12**, no entanto já existe uma versão mais atual lançada em maio de 2019, PHP 7.3.6.

2.1.3. – Bootstrap

Bootstrap é um *open source web framework* criado para atuar no *front-end* de sites e aplicações WEB.

Foi criado por Mark Otto e Jacó Thornton e inicialmente foi desenvolvido para atuar no Twitter, com o nome de Twitter Blueprint, como um instrumento para incentivar a consistência através de ferramentas internas.

Este *framework* conta com uma série de classes em CSS prontas, além de plugins em *JavaScript* (jQuery) para implementar recursos como *dropdowns*, *carrosséis* e *slideshows* de maneira fácil e com pouco esforço ao nível do código.

A versão do Bootstrap usada neste projeto é Bootstrap **v3.2.0**, no entanto já existe a versão 4 deste *framework*⁴.

⁴ “Bootstrap” 2019. [Online]. Disponível em: <https://github.com/twbs/bootstrap>

2.1.4. - JQuery

JQuery é uma biblioteca de funções programadas em *JavaScript* que interage com o HTML, foi criada com o objetivo de simplificar os scripts interpretados do lado do cliente. O JQuery é uma biblioteca de código aberto (*open source*) e foi desenvolvida em 2006 por John Resig,

Com o JQuery é possível fazer diversos efeitos com poucas linhas de código, que se fossem feitos em *JavaScript* puro demorariam tempo e custariam dezenas de linhas para serem feitos.

Estes são alguns dos recursos que o JQuery ⁵oferece:

- Seleção e manipulação de elementos HTML
- Manipulação de CSS
- Efeitos e animações
- Navegação pelo DOM
- Ajax
- Eventos
- Entre outros...

2.1.5. - Composer

O Composer é uma ferramenta de gestão de dependências em PHP do lado do servidor, foi lançado em 2012 e foi criado por Nils Adermann e Jordi Boggiano.

Esta ferramenta permite que sejam declaradas bibliotecas que o projeto necessite, e a partir daí, essas bibliotecas são automaticamente instaladas/atualizadas⁶.

⁵ "JQuery" 2019. [Online]. Disponível em: <https://jquery.com/download/>

⁶ "Composer" 2019. [Online]. Disponível em: <https://getcomposer.org/download/>

2.1.6. - PHP Mailer

PHP MAILER⁷ é uma biblioteca de código aberto (*open source*) desenvolvida para enviar emails com suporte a HTML e a diferentes protocolos e níveis de autenticação.

Como possui uma implementação do protocolo SMTP, é capaz de ser utilizada em plataformas que não possuem um servidor para envio de e-mails nativo, como por exemplo o Windows.

Foi criado em 2001 por Brent R. Matzelle para um projeto da *SourceForge*.

2.1.7. - Rain TPL

Rain TPL⁸ é um *template engine*, isto é, um *script* que carrega os *templates* e desenha a parte gráfica do projeto.

O objeto desta ferramenta é separar a parte gráfica da parte lógica do projeto, para que os desenvolvedores e os designers WEB consigam trabalhar melhor, juntos.

Basicamente é criado um modelo, e cada vez que é necessário um *layout*, chama-se esse modelo e troca-se alguns valores.

2.1.8. – AdminLTE

AdminLTE⁹ é um *template frontend* usado para construir *websites* e *webapps*, é usado como um painel de controlo de uma certa aplicação.

É feito em HTML responsivo e em CSS *framework* Bootstrap 3.

⁷ “PHP MAILER” 2019. [Online]. Disponível em: <https://github.com/PHPMailer/PHPMailer>

⁸ “RainTPL” 2019. [Online]. Disponível em: <https://github.com/feulf/raintpl3>

⁹ “AdminLTE” 2019. [Online]. Disponível em: <https://adminlte.io/>

2.1.9. - Slim Framework

Slim Framework¹⁰ é uma *microframework* concebida para trabalhar com rotas.

Este *framework* é essencialmente um *dispatcher*, que recebe uma solicitação de um HTTP, e invoca uma rota de retorno de chamada apropriada, a resposta a esta solicitação é também feita, através de um HTTP.

É uma ferramenta que requer pouco esforço por parte do desenvolvedor, rápida e simples.

Neste projeto, o Slim Framework, funcionou como *CONTROLLER*, portanto, é este *framework* que tem a função de intermediário entre as VIEWS E O MODEL.

2.1.10. - Arquitetura de Software

A arquitetura de software está preocupada com a compreensão de como um sistema de ser organizado e com a estrutura geral desse sistema.

Num projeto, o projeto de arquitetura deve ser o primeiro passo do desenvolvimento de qualquer projeto de software, pois é o elo crítico entre o projeto e a engenharia de requisitos.

É a arquitetura de software que identifica os componentes estruturais de um sistema e os relacionamentos entre eles.

O resultado do processo de projeto de arquitetura é um modelo de arquitetura que descreve como o sistema está organizado em conjunto de componente de comunicação.

¹⁰ "Slim Framework" 2019. [Online]. Disponível em: <https://github.com/slimphp/Slim>

Atualmente há várias arquiteturas, e por isso, existem várias formas de planejar o nosso projeto, a arquitetura escolhida para este projeto tem o nome de MVC.

2.1.10.1. - MVC

Esta arquitetura (imagem 1)

em 1979, originalmente com o nome MVCU (Model, View, Controller, User), por Trygve Reenskaug. User surgiu como quarta camada pois Reenskaug acreditava que o User tivesse autonomia para definir algumas funcionalidades do sistema. Com o tempo o conceito de User foi desaparecendo, pois, existem muitos casos onde o User não consegue definir, de forma autónoma, as necessidades do sistema, por isso, atualmente fala-se somente de MVC.

MVC é uma arquitetura que permiti dividir o desenvolvimento de uma aplicação em 3 camadas concetuais: Model, View e Controller (Modelo - Visão - Controlador).

- Model- Nesta camada concetual é onde serão definidas as regras de negócio, e onde será feito a modelagem dos dados, vai ser aqui onde serão criadas as classes, as consultas à base de dados e, como já foi referido, as regras de negócio do sistema.
- View- Nesta camada concetual é onde é feito a representação dos dados da aplicação numa *interface* apropriada, é onde os dados são representados aos utilizadores dessa aplicação, normalmente é composta por via do HTML ou XML, CSS (e *JavaScript* caso seja necessário).
- Controller – Nesta camada concetual é onde o sistema é controlado. Nesta camada são feitas as ligações das mais variadas partes do sistema, recebe e valida os dados fornecidos pelo utilizador, faz a ligação desses dados com as regras definidas no Model, assim como as consultas à base de dados, processa esta requisição e retorna um resultado, geralmente numa nova View.

Ao haver esta abstração destas três camadas conceituais, o MVC permite que o desenvolvimento seja mais fácil, visto que organiza as partes do projeto em três pequenos módulos, cada um com uma função específica no sistema.

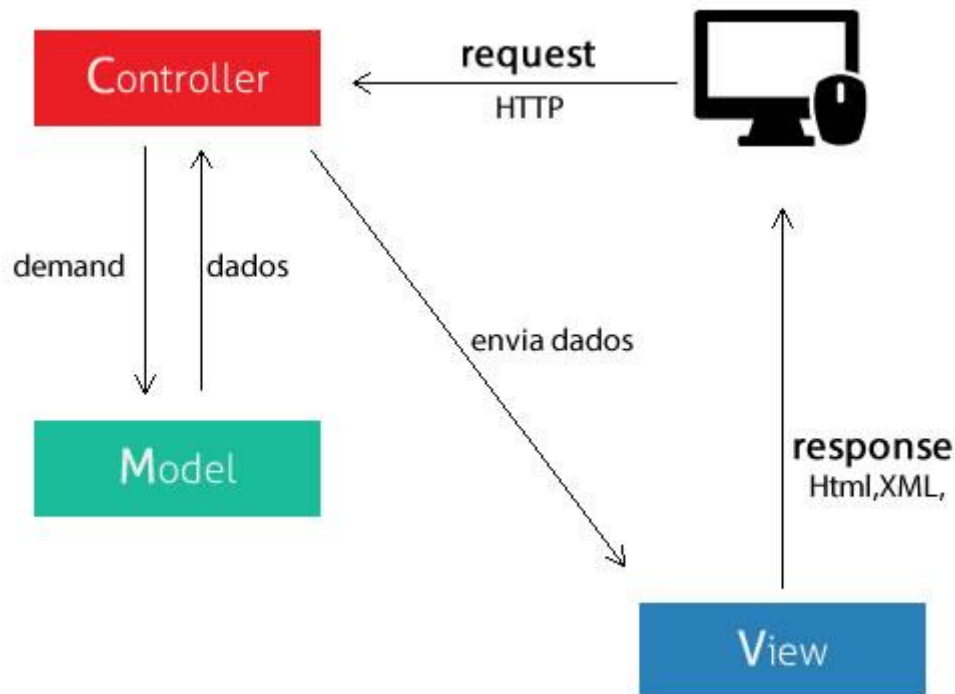


Figura 1- Funcionamento da arquitetura MVC

2.2. - Aplicações

2.2.1. – Sublime Text 3

Sublime Text é um software multiplataforma de edição de texto, no entanto é usado por muitos desenvolvedores de software, como editor de código-fonte.

Escrito em linguagem C++, o Sublime Text, que foi inicialmente pensado para ser uma extensão do VIM (outro editor de texto), oferece recursos extraordinários e um desempenho simplesmente surpreendente.

É um software que contém uma licença paga caso o desenvolvedor pretenda, no entanto é possível usar este programa gratuitamente.

De muitas utilizações, é de destacar, o seu uso como produto de desenvolvimento de aplicações WEB ou websites, utilizando o PHP (linguagem usada no projeto “Student.Home”).

Esta plataforma¹¹ foi escolhida, porque no curso online tirado na *Udemy*, o editor de texto usado foi precisamente este, e por isso, visto que já estava acostumado às suas funcionalidades, decidi continuar com ele.

2.2.2. – Apache

Criado em 1995 por Rob McCool, o servidor Apache ou Servidor HTTP Apache é o mais bem-sucedido servidor web livre que existe.

O Apache é responsável por disponibilizar páginas e recursos que podem ser acessados pelo internauta.

Envio de e-mails, mensagens, compras online e diversas outras funções podem ser executadas graças a servidores como o Apache. O que vale destacar no Apache é que, apesar de tudo, ele é distribuído sob a licença GNU, ou seja, é gratuito e pode ser estudado e modificado através de seu código fonte por qualquer pessoa.

O servidor Apache é compatível com o protocolo HTTP, e as suas funcionalidades são mantidas através de uma estrutura de módulos, isso permite que os desenvolvedores escrevam os seus próprios módulos por meio de uma API do software.

Está disponibilizado para Windows, OS etc.

¹¹ “Sublime Text 3” 2019. [Online]. Disponível em: <https://www.sublimetext.com/>

2.2.3. – XAMPP

Xampp- **X** (para qualquer dos diferentes sistemas operativos), **A**pache, **M**ySQLDB, **P**HP, **P**erl, foi inicialmente lançado em 2002, desenvolvido por um grupo chamado *Apache Friends*.

O Xampp consiste num pacote que contem os principais servidores de código aberto do mercado, como o FTP, base de dados MySql e Apache e suporta linguagens como o PHP e Perl.

Com esta plataforma é possível executar aplicações localmente, o que facilita e agiliza o desenvolvimento das mesmas. Como o conteúdo dessas aplicações está armazenado numa rede local, o acesso ao arquivo é realizado quase instantaneamente.

Atualmente, o XAMPP¹² está disponível para quatro sistemas operativos, Windows, Linux, Mac OS e Solaris e não é necessário haver instalação para ser usado.

2.2.4. – MySql Workbench 6.3

MySql Workbench¹³ é uma ferramenta que permite o desenho e gestão de bases de dados.

Desenvolvida pela empresa Oracle, é um Sistema de Gestão de Base de Dados (SGBD) relacionais que permite editar dados, editar scripts SQL, gerir conexões, novo modelo de dados, modelo de dados, entre muitas outras funcionalidades.

¹² “XAMPP” 2019. [Online]. Disponível em: <https://www.apachefriends.org/download.html>

¹³ “MySql Workbench 6.3” 2019. [Online]. Disponível em: <https://downloads.mysql.com/archives/workbench/>

2.2.5. -Visual Paradigm

Visual Paradigm¹⁴ é uma ferramenta UML CASE que suporta UML 2, SysML, *Business Process Modeling Notation* (BPMN), é uma ferramenta indispensável para o desenvolvimento de aplicações de grande escala, segundo uma abordagem por objetos.

Suporta as versões mais recentes de Java e da notação UML. Pode ser integrada noutros ambientes de desenvolvimento como o Visual Studio, Eclipse, etc.

¹⁴ “Visual Paradigm” 2019. [Online]. Disponível em: www.visual-paradigm.com

3. -Análise e Modelação do Sistema

3.1. – Cascata

É de extrema importância haver uma definição da metodologia a seguir para a realização do projeto, visto que essa metodologia influencia profundamente o resultado final. É na parte de desenvolvimento onde se poderá gastar energias, tempo, recursos em situações que não têm tanta prioridade/interesse.

Para evitar a situação descrita anteriormente, decidimos usar a metodologia Cascata, esta metodologia é a forma mais tradicional de abordar o desenvolvimento do projeto.

O objetivo principal deste modelo é fazer com que as diferentes fases de desenvolvimento sigam uma sequência. Cada tarefa só inicia quando a anterior está finalizada, além disso, os requisitos são definidos no início do projeto e normalmente sofrem pouca ou nenhuma alteração durante a sua execução.

Este modelo, geralmente apresenta as seguintes fases básicas de elaboração do sistema:

- Análise e Definição de requisitos.
- Projeção do Sistema.
- Implementação do Sistema.
- Teste ao Sistema.
- Manutenção do Sistema.

3.1.1. – Vantagens do Modelo

O uso deste modelo torna o processo de desenvolvimento estruturado, dando uma ordem sequencial para cada fase, além disso, todas as atividades identificadas nas fases do modelo são fundamentais e estão na ordem certa de execução.

Se for feito de forma minuciosa, o projeto torna-se mais fácil de ser gerido, já que cada fase está bem definida.

3.1.2. – Desvantagens do Modelo

No entanto, este modelo (imagem 2) também algumas desvantagens, uma delas a impossibilidade de atualizar ou redefinir fases após estarem concluídas, é também, impossível modificar os requisitos iniciais.

Caso haja um atraso, todas as fases seguintes vão ser afetadas.

O *feedback* entre fases não existe, por isso só haverá *feedback* após conclusão de cada etapa.

Como o *project owner* é o desenvolvedor desta aplicação, existe um profundo conhecimento das necessidades e dos requisitos do projeto, e por isso é que se optou por este modelo mais tradicional, por isso não existe os riscos de apresentar feedback entre as fases do modelo, mas sim no final do projeto.

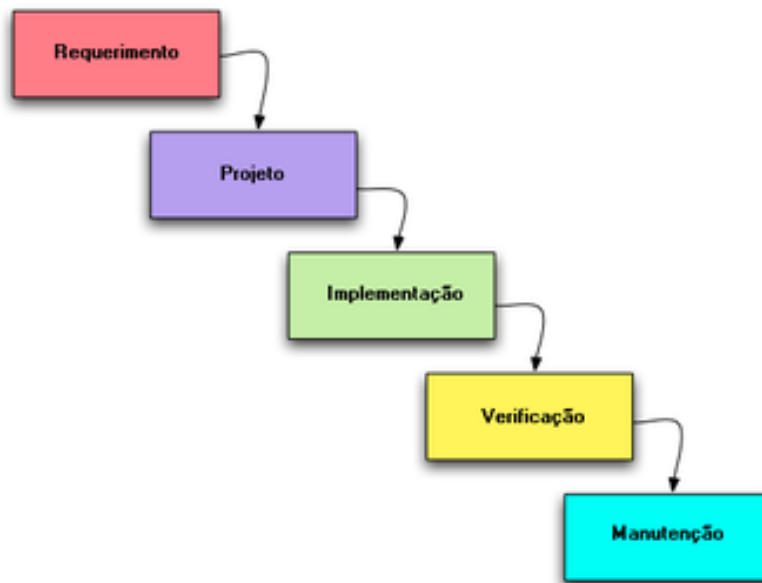


Figura 2-Modelo Cascata.

3.2. – Identificação dos Requisitos

A identificação dos requisitos é uma etapa essencial na gestão de projetos.

Esta etapa consiste na recolha de dados indispensáveis para o cliente/utilizador de forma a tentar resolver um certo problema.

Esta recolha de requisitos deve ser feita de forma detalhada e de, e relevante para o projeto.

Deve ser também, nesta etapa que existe o acordo entre o cliente e o desenvolvedor sobre a forma como o *software* vai funcionar.

É também essencial haver uma linha constante de comunicação e o envolvimento constante com os utilizadores, pois estes, irão influenciar o resultado final do produto.

3.2.1. – Análise de Requisitos

A Análise de Requisitos consiste em:

- **Reconhecer o problema** – É nesta fase onde é encontrada a especificação do sistema, o planeamento, o contato do analista com o cliente com a intenção de entender a visão do cliente em relação ao problema.
- **Avaliar o problema e a síntese da solução** – Nesta fase há uma tentativa de entender o problema, para isso faz-se a identificação das informações que serão necessárias ao utilizador e ao sistema é selecionado também a melhor solução possível dentro das soluções propostas.
- **Modelar (Modelagem)** – É um recurso usado para o suporte da síntese da solução, o modelo vai apresentar ferramentas que facilitarão o entendimento do sistema, como as funcionalidades, informações e comportamento do sistema.
- **Especificar os requisitos** – consolida funções, interfaces, desempenho, o contexto e as restrições do sistema.
- **Revisão** – O cliente e o analista avaliam, em conjunto, o objetivo do projeto com o intuito de eliminar possíveis redundâncias, inconsistências e omissões do sistema, obtendo uma mesma visão.

3.2.2. – Técnicas de Análise de Requisitos

3.2.2.1. – Questionários

Existem vários tipos de questionários que podem ser utilizados. Entre estes podemos listar: escolha múltipla, lista de verificação e questões com espaços em branco. O

questionário deve ser desenvolvido de forma a minimizar o tempo gasto na sua resposta.

O uso de questionário é indicado, por exemplo, quando há diversos grupos de utilizadores que podem estar em diversos locais do país. Neste caso, elaboram-se pesquisas específicas de acompanhamento com utilizadores selecionados, que a contribuição em potencial pareça mais importante, pois não seria prático entrevistar todas as pessoas em todos os locais.

3.2.2.2. – Entrevista

A entrevista é uma das técnicas tradicionais mais simples de utilizar e que produz bons resultados na fase inicial de obtenção de dados. É importante que o entrevistador dê margem e tempo ao entrevistado de forma a que esta consiga expor as suas ideias, opiniões etc. Para tentar evitar uma dispersão do assunto principal, não tornar a entrevista longa, e acima disso tudo, para haver um “fio condutor” da conversa é uma boa prática ter um plano de entrevista para que não haja dispersão do assunto principal e a entrevista fique longa, deixando o entrevistado cansado e não produzindo os resultados esperados.

3.2.2.3. – Brainstorming

Brainstorming é uma técnica para criação de ideias. Consiste em várias reuniões que permitem que as pessoas sugiram e explorem ideias para a evolução de um certo projeto.

As ideias devem ser consideradas e compiladas, sendo uteis na evolução para uma solução final.

3.2.3. – Tipo de Requisitos

Existem vários tipos de requisitos, aos quais vamos enumerar:

- **Requisitos Funcionais** – Estes requisitos descrevem a funcionalidade (funções que o sistema deve realizar) ou os serviços que se espera que o sistema faça.
- **Requisitos Não Funcionais** – São requisitos que não dizem respeito ao sistema, mas expressam propriedades do sistema ou/e restrições sobre os serviços ou funções por ele providas.
- **Requisitos do Produto** – São requisitos que especificam o comportamento dos produtos.

3.2.4. – Definição de Prioridades- Técnica Usada

A técnica de priorização escolhida foi **MOSCOW**.

MoSCoW é um método de priorização usado para decidir quais são os requisitos que devem ser concluídos primeiro, quais devem vir depois e quais devem ser “deixados de lado”.

Este método usa palavras como forma de definir a prioridade, cada palavra significa algo e facilita a discussão sobre o que é, ou não importante.

3.2.4.1. – MUST HAVE (MUST)

Todos os requisitos classificados com **MUST**, são os requisitos críticos para a completção do projeto final, caso um destes requisitos críticos não tiver sido satisfeito, o projeto não pode ser considerado como concluído com sucesso.

3.2.4.2. – SHOULD HAVE (SHOULD)

Todos os requisitos classificados com **SHOULD**, são requisitos importantes, no entanto não são tão críticos numa escala de tempo, e por isso, normalmente há outros meios de se atender á necessidade.

3.2.4.3. – COULD HAVE (COULD)

Todos os requisitos classificados com **COULD**, são requisitos considerados desejáveis, mas não necessários e muito menos críticos.

Normalmente, são requisitos que melhoram a experiência do utilizador ou a satisfação do cliente, e requerem pouco esforço de desenvolvimento.

Caso haja tempo, estes tipos de requisitos são incluídos, caso contrário não.

3.2.4.4. – WOULD LIKE (WOULD)

Todos os requisitos classificados por **WOULD**, são requisitos que têm a concordância dos interessados, no entanto, como são itens menos críticos e com menor retorno sobre o investimento, não são tão críticos como os anteriores.

3.2.5. – Vantagens da Técnica MoSCoW

As vantagens deste método de priorização são as seguintes:

- É uma técnica fácil de implementar e de fácil compreensão.
- Permite a participação de todos os envolvidos no projeto(*stakeholders*).

3.2.6. – Desvantagens da Técnica MoSCoW

As desvantagens deste método de priorização são as seguintes:

- A priorização de cada *Stakeholder* é subjetiva.
- O *Stakeholder* tem de conhecer o negócio.

3.2.7. – Atores

Esta aplicação tem vários intervenientes capazes de gerir, controlar, administrar o website para que os utilizadores/clientes possam usufruir ao máximo com a utilização desta aplicação.

3.2.7.1. - Administradores

De forma a garantir que os imóveis adicionados ao site são exclusivamente para estudantes é necessário ter um administrador na zona que se pretende adicionar o imóvel, onde, após o contacto connosco essa pessoa se desloca ao imóvel para observar o mesmo, e seguidamente o adiciona á nossa base de dados.

Para adicionar necessita de referir o local do imóvel, a área, o preço, o certificado de energia, o número de quartos e casas de banho, assim como modificar o estado de aluguer do imóvel.

3.2.7.2. - Cliente

Para conseguir utilizar esta aplicação com todas as funções (conseguir obter o número do proprietário do imóvel, ver o estado de aluguer do imóvel) é necessário o cliente, fazer um registo no website. Após a conclusão desse registo, o cliente poderá usar todas as funções feitas para este ator, como guardar numa lista os imóveis que já foi visitar, contactar os administradores do website, como também, caso queira, contactar um administrador para ir observar o seu imóvel para ver se é elegível.

3.2.7.3. – Visitante

Caso um utilizador da *internet* aceda ao website é considerado como visitante, o visitante consegue ver os imóveis em destaque, consegue procurar imóveis na sua zona, no entanto não consegue usar as funções de observação de imóveis, não consegue visualizar o número do proprietário.

3.3. – Requisitos Funcionais

Para melhor compreensão, dividimos os requisitos funcionais do *website* e do *BackOffice*.

3.3.1. – Requisitos Funcionais *BackOffice*

ID	Tag	Requisito	Descrição	Prioridade	Tipo R
1.1	Acesso ao Sistema	Ecrã de login para acesso ao painel de controlo	O sistema deve possuir um ecrã de acesso ao painel de controlo do <i>website</i> , para prevenir o acesso de desconhecidos.	MUST	RF01
1.2		Registo de utilizadores com permissão de administrador	O sistema deve ter um formulário de registo de um novo administrador, esse registo deve ser feito por outro administrador e deverá conter os seguintes campos: <ul style="list-style-type: none"> • Nome de login para facilitar acesso; • E-mail; • Palavra-Passe; • Dar permissão de administração; 	MUST	RF02
1.3		Reposição da palavra - passe	O sistema deve permitir a redefinição da palavra-passe caso a mesma seja esquecida.	SHOULD	RF03
1.4		Atualizar dados de conta	O sistema deve permitir alterar dados da conta.	SHOULD	RF04
1.5		Autenticação por via do Facebook	O sistema poderá autenticar o utilizador no sistema por via da plataforma Facebook.	WOULD	RF05
1.6		Autenticação por via do Google	O sistema poderá autenticar o utilizador no sistema por via da plataforma Google.	WOULD	RF06
1.7		Login de Utilizadores	O sistema deve possuir um ecrã de login para o painel de controlo, para que haja um acesso ao mesmo, são necessárias as seguintes credenciais: <ul style="list-style-type: none"> • Nome de Login; • Palavra-Passe; 	MUST	RF07
1.8			O sistema poderá permitir o acesso de um utilizador através do Facebook.	WOULD	RF08
1.9			O sistema poderá permitir o acesso de um utilizador através do Google.	WOULD	RF09

2.1	Imóveis	Registo de Imóveis	<p>O sistema deve permitir a adição de novos imóveis com os seguintes campos:</p> <ul style="list-style-type: none"> • Local do Imóvel; • Preço de Aluguer do Imóvel; • Número de Quartos; • Área do Imóvel; • Número de Casas de Banho; • Certificado de Energia; • URL do Imóvel; • Telefone do Proprietário; 	MUST	RF10
2.2		Visualizar Detalhes de Imóveis	O sistema deve permitir visualizar os detalhes dos imóveis registados na base de dados.	SHOULD	RF11
2.3		Editar Imóveis	O sistema deverá permitir a edição dos Imóveis, permitindo adicionar mais imagens do mesmo.	SHOULD	RF12
2.4		Eliminar Imóveis	O sistema poderá permitir a eliminação de imóveis.	COULD	RF13
3.1	Categorias	Registo de Categorias	<p>O sistema deve permitir o registo de categorias de imóveis com o seguinte campo:</p> <ul style="list-style-type: none"> • Nome do Tipo de Categoria; 	MUST	RF14
3.2		Visualizar Categorias	O sistema poderá permitir a visualização das categorias existentes na base de dados.	SHOULD	RF15
3.3		Editar Categoria	O sistema deverá permitir a edição de uma categoria.	SHOULD	RF16
3.4		Eliminar Categoria	O sistema poderá permitir a eliminação de uma categoria.	COULD	RF17
3.5		Adicionar Imóveis a uma certa Categoria	O sistema deverá permitir a adição de imóveis a uma certa categoria.	MUST	RF18

3.6		Visualizar Imóveis de uma certa categoria	O sistema poderá mostrar os imóveis correspondentes a uma certa categoria.	COULD	RF19
3.7		Eliminar Imóveis de uma certa categoria	O sistema poderá conseguir eliminar os imóveis correspondentes a uma certa categoria.	COULD	RF20
4.1	Utilizadores	Registo de Utilizador	<p>O sistema deverá permitir fazer um registo de um novo utilizador com os seguintes campos:</p> <ul style="list-style-type: none"> • Nome do utilizador; • Nome de login para facilitar acesso; • Telefone; • E-mail; • Palavra-Passe; • Dar permissão de administração; 	MUST	RF21
4.2		Detalhes do Utilizador	O sistema poderá permitir a visualização dos detalhes do utilizador.	SHOULD	RF22
4.3		Editar Utilizador	O sistema deve permitir a edição dos utilizadores.	SHOULD	RF23
4.4		Eliminar Utilizador	O sistema poderá permitir a eliminação de utilizadores.	COULD	RF24

3.3.2. - Requisitos funcionais – Website

ID	Tag	Requisito	Descrição	Prioridade	Tipo R
1.1	Acesso ao Sistema	Ecrã de login para acesso ao painel de controlo	O sistema deve possuir um ecrã de acesso ao painel de controlo do <i>website</i> , para prevenir o acesso de desconhecidos.	MUST	RF01
1.2		Registo de utilizadores	<p>O sistema deve ter um formulário de registo de um novo utilizador, esse registo deverá conter os seguintes campos:</p> <ul style="list-style-type: none"> • Nome da Pessoa; • E-mail; • Telefone; • Palavra passe; 	MUST	RF02
1.3		Reposição da palavra -passe	O sistema deve permitir a redefinição da palavra-passe caso a mesma seja esquecida.	SHOULD	RF03

1.4		Atualizar dados de conta	O sistema deve permitir alterar dados da conta.	SHOULD	RF04
1.5		Autenticação por via do Facebook	O sistema poderá autenticar o utilizador no sistema por via da plataforma Facebook.	WOULD	RF05
1.6		Autenticação por via do Google	O sistema poderá autenticar o utilizador no sistema por via da plataforma Google.	WOULD	RF06
1.7		Login de Utilizadores	O sistema deve possuir um ecrã de login, para que haja um acesso ao mesmo, são necessárias as seguintes credenciais: <ul style="list-style-type: none"> • E-mail; • Palavra-Passe; 		
2.1	Imóveis Observados	Visualização de Imóveis já observados	O sistema deve permitir que o utilizador consiga guardar numa lista os imóveis que observou.	MUST	RF11
2.2		Detalhes dos Imóveis Observados	O sistema deve permitir visualizar os detalhes dos imóveis observados.	SHOULD	RF12
2.3		Eliminar Imóveis Observados	O sistema poderá permitir a eliminação de imóveis eliminados.	COULD	RF13
3.1	Imóveis	Procurar Imóveis	O sistema deverá permitir procurar imóveis pelo seu preço, local, e estado de aluguer.	SHOULD	RF14
3.2		Observar Imóveis	O sistema deverá permitir a visualização mais detalhada dos imóveis.	SHOULD	RF15

3.3		Observar Imóveis por Categoria	O sistema deverá permitir a visualização de imóveis por categoria.	SHOULD	RF16
3.4		Pedir Observação do Imóvel	O sistema poderá ter um formulário de contacto para o utilizador pedir que um administrador vá observar o seu imóvel.	COULD	RF17
4.1	Utilizadores	Alterar Dados Utilizador	O sistema deverá permitir fazer uma alteração nos dados do utilizador.	SHOULD	RF18
4.2		Detalhes do utilizador	O sistema poderá permitir a visualização dos detalhes do utilizador.	SHOULD	RF19
4.4		Eliminar Utilizador	O sistema poderá permitir a eliminação de utilizadores.	COULD	RF20
5.1	Contacto	Contactar Administração	O sistema poderá ter um formulário de contacto para contactar os administradores do site.	COULD	RF21

3.4. - Requisitos Não Funcionais

ID	Requisito	Descrição	Prioridade	Tipo R
1.1	Segurança e Privacidade dos dados	O sistema deverá conseguir proteger os dados de todos os utilizadores.	MUST	RNF01
2.1	Interoperabilidade	O sistema deverá comunicar com MySql.	MUST	RNF02

3.1	Interface Simples	O <i>website</i> tem de possuir uma interface simples e intuitivo e de fácil utilização. Deve ser desenvolvido a pensar em utilizadores com pouca destreza em trabalhar com sistemas informáticos	MUST	RNF03
4.1	Confiabilidade	É muito importante o sistema, executar sem falhas, cerca de 99% do tempo.	SHOULD	RNF04
5.1	Idiomas	A aplicação poderá vir a suportar mais que um idioma.	WOULD	RNF05
6.1	Moeda Corrente	A aplicação poderá vir a suportar mais que uma moeda corrente.	WOULD	RNF06
7.1	Manutenção	O sistema deverá ter um suporte de manutenção para correção de alguns problemas que poderão aparecer, assim como criação de novas funcionalidades.	COULD	RNF07

Legenda:

- **Must** - Tem de ter
- **Should** - Deve ter
- **Could** - Poderá ter
- **Would** - Aplicação deve permitir acrescentar dada funcionalidade

3.5. – Casos de Uso

3.5.1. – Controlo de Acesso ao Sistema

Será aqui (imagem 3) que existirá o controlo de acesso ao sistema.

Começará com o registo do “Utilizador”, onde o mesmo insere as suas credenciais, e cria o seu registo, após isso tem oportunidade de fazer o “Login” e de “Atualizar os Dados” da sua conta.

Em relação aos “Utilizadores com Permissão de Administrador”, esses registos com permissões elevadas, terão de ser feitos pelo “Administrador do Sistema” que os regista e dá a permissão, após esse registo, os “Utilizadores com Permissão de Administrador” podem fazer “Login Administrativo”, “Atualização dos Dados Administrativos”.

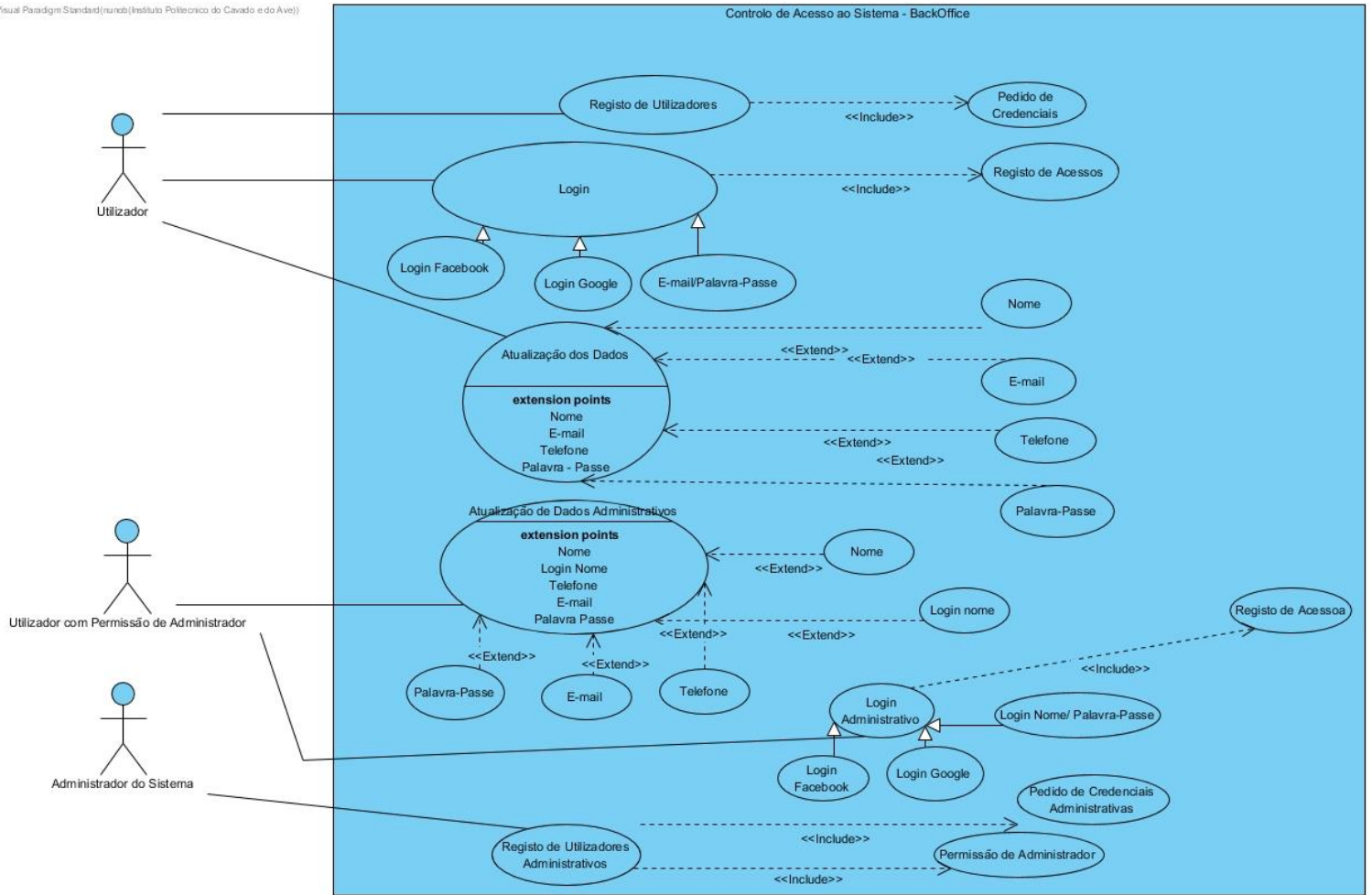


Figura 3-Casos de Uso de Controlo de Acesso ao Sistema.

3.5.2. – Imóveis

Na imagem 4 podemos ver o caso de uso da “Gestão de Imóveis”.

O “Utilizador com permissão de Administrador” é quem “Regista”, “Edita” e “Elimina” Imóveis.

O “Cliente” é um utilizador com registo que pode consultar os imóveis registados no sistema e pedir a avaliação do seu imóvel de forma a este ser adicionado ao sistema.

O “Visitante” é um utilizador que acede ao *website*, mas que não tem registo, e sendo assim só consegue ver os dados do imóvel e não do proprietário.

O “Sistema de E-mail” envia para a administração o pedido de avaliação do “Cliente”.

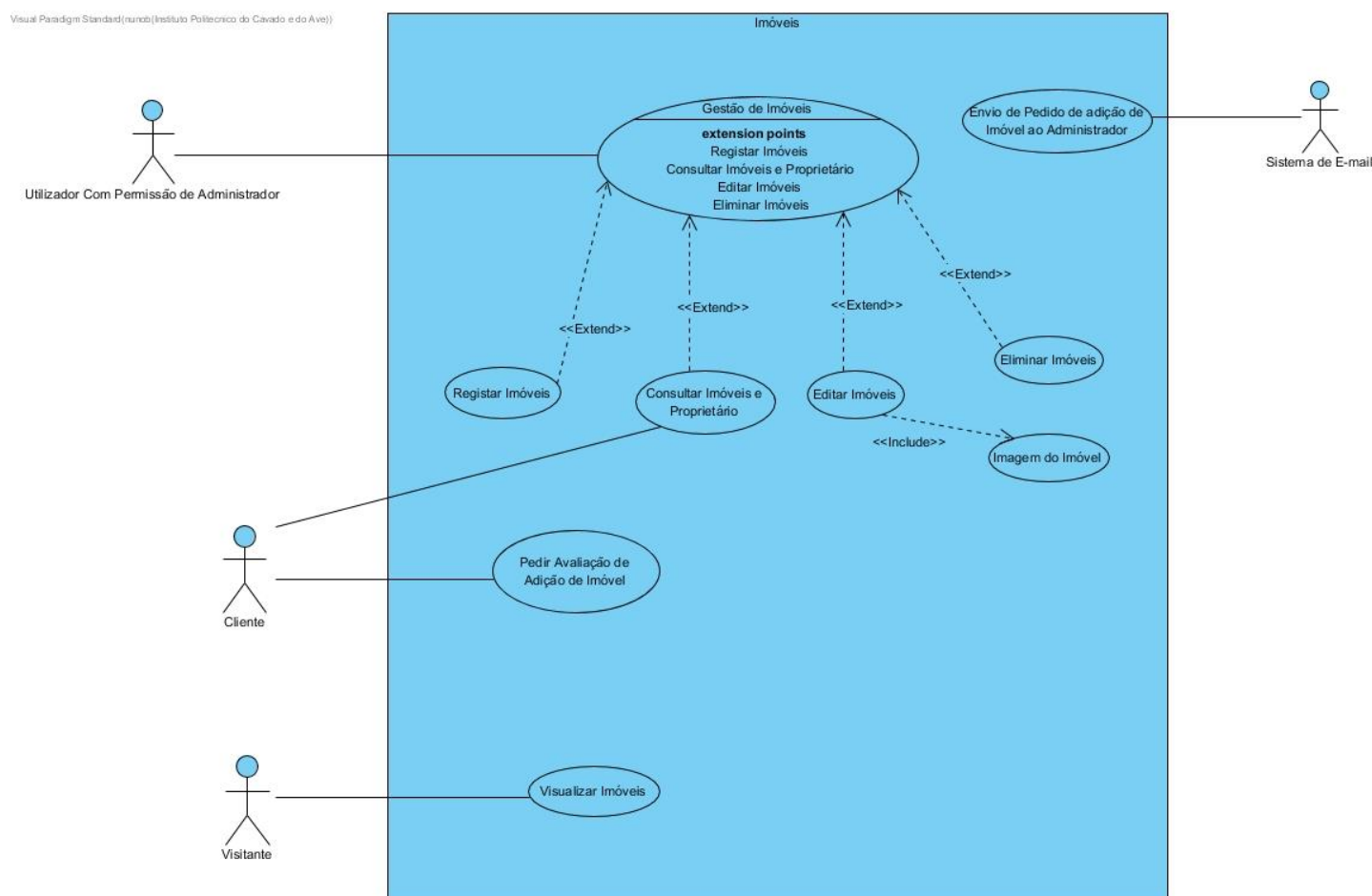


Figura 4-Caso de Uso dos Imóveis

3.5.3. – Categorias

Na imagem 5 podemos ver o caso de uso das “Categorias”.

O “Utilizador com Permissão de Administrador” tem a capacidade de “Registrar”, “Editar”, “Eliminar” Categorias, assim como “Adicionar” e “Eliminar” imóveis de uma certa “Categoria”.

O “Cliente” tem a opção de ver todos os imóveis que pertencem a uma “Categoria” assim como o “Visitante”.

Visual Paradigm Standard (runch@Instituto Politécnico do Cávado e do Ave)

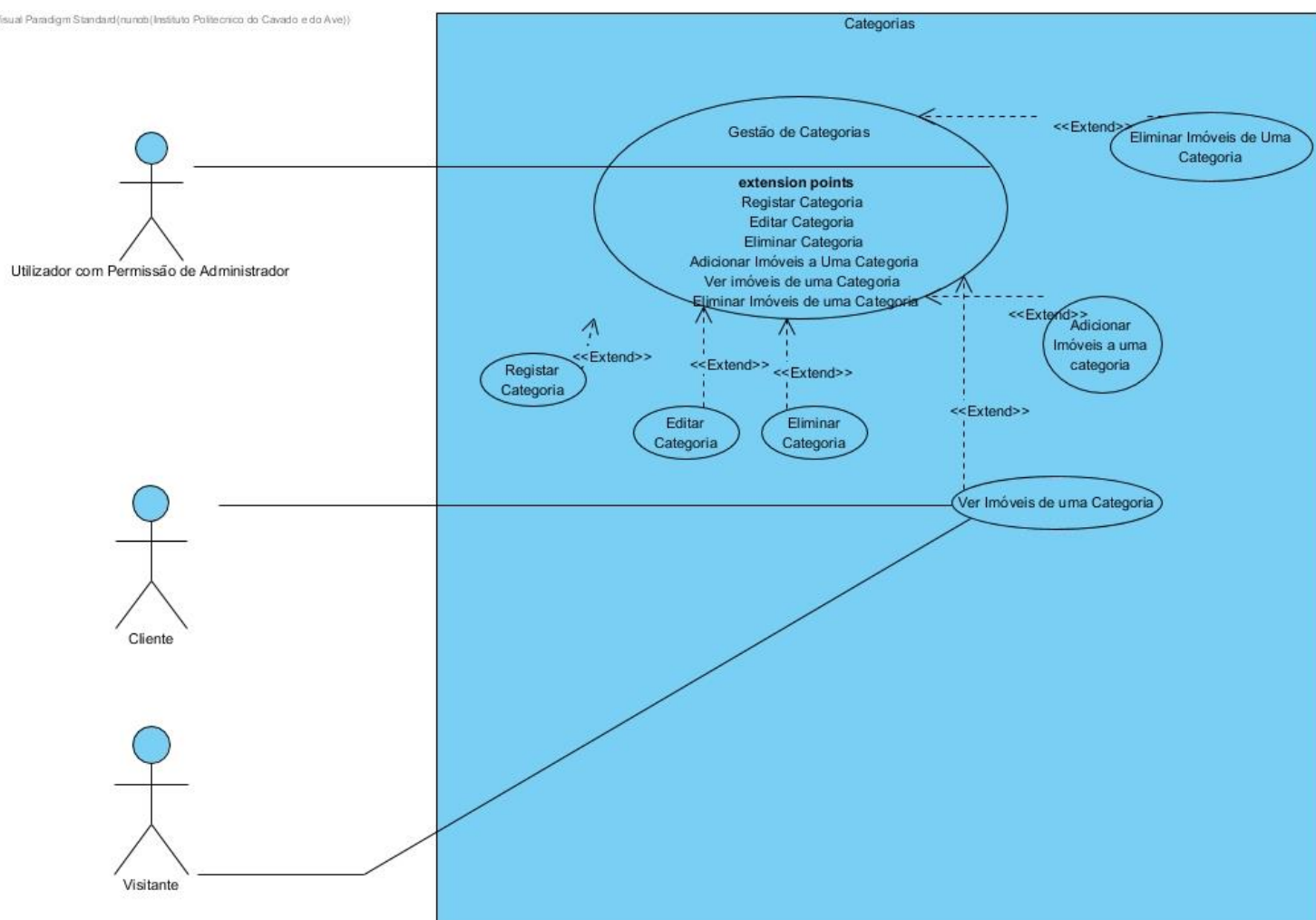


Figura 5-Casos de Uso das Categorias

3.6. – Diagrama de Classes

Neste diagrama (imagem 6) podemos ver a representação da estrutura e relações das classes que servem de modelo para os objetos.

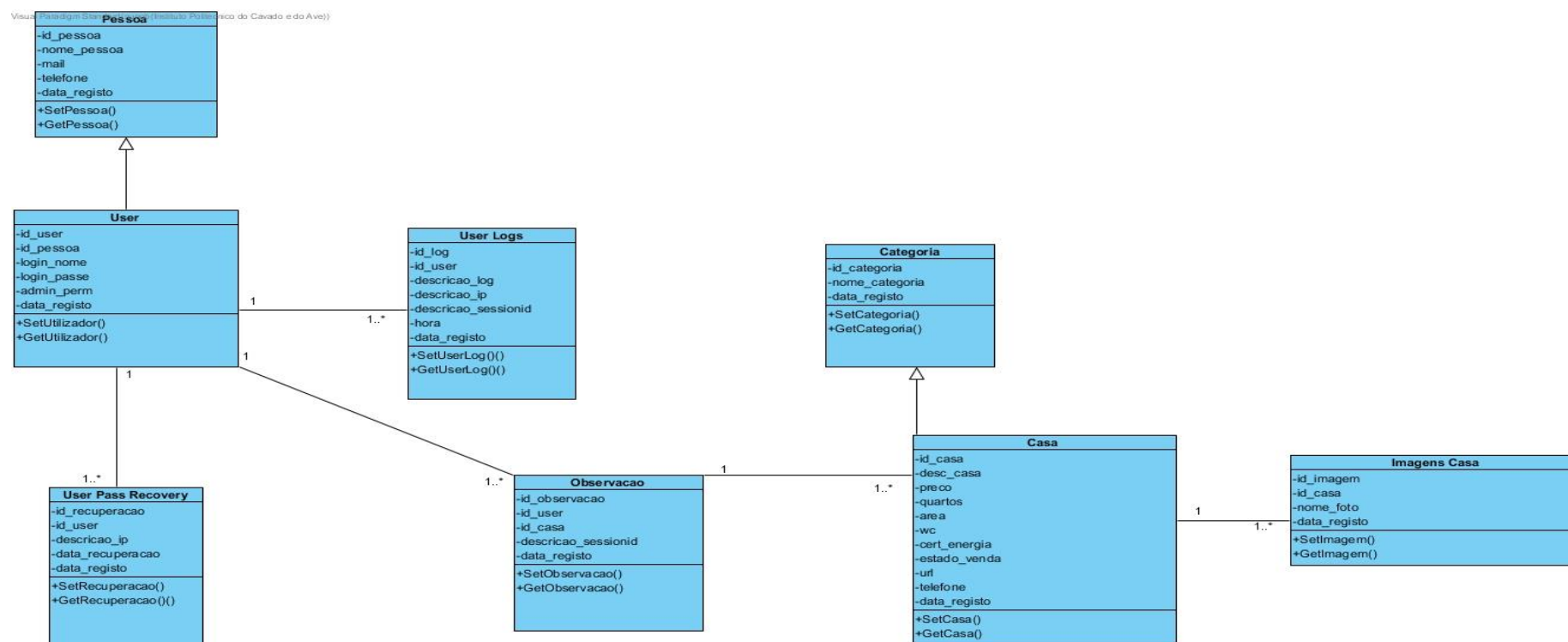


Figura 6-Diagrama de Classes.

3.7. – Diagrama Entidade-Relação (Base de Dados)

Na página seguinte (imagem 6) poderemos ver o diagrama Entidade- Relação da base de dados, feito no *software* MySql Workbench.

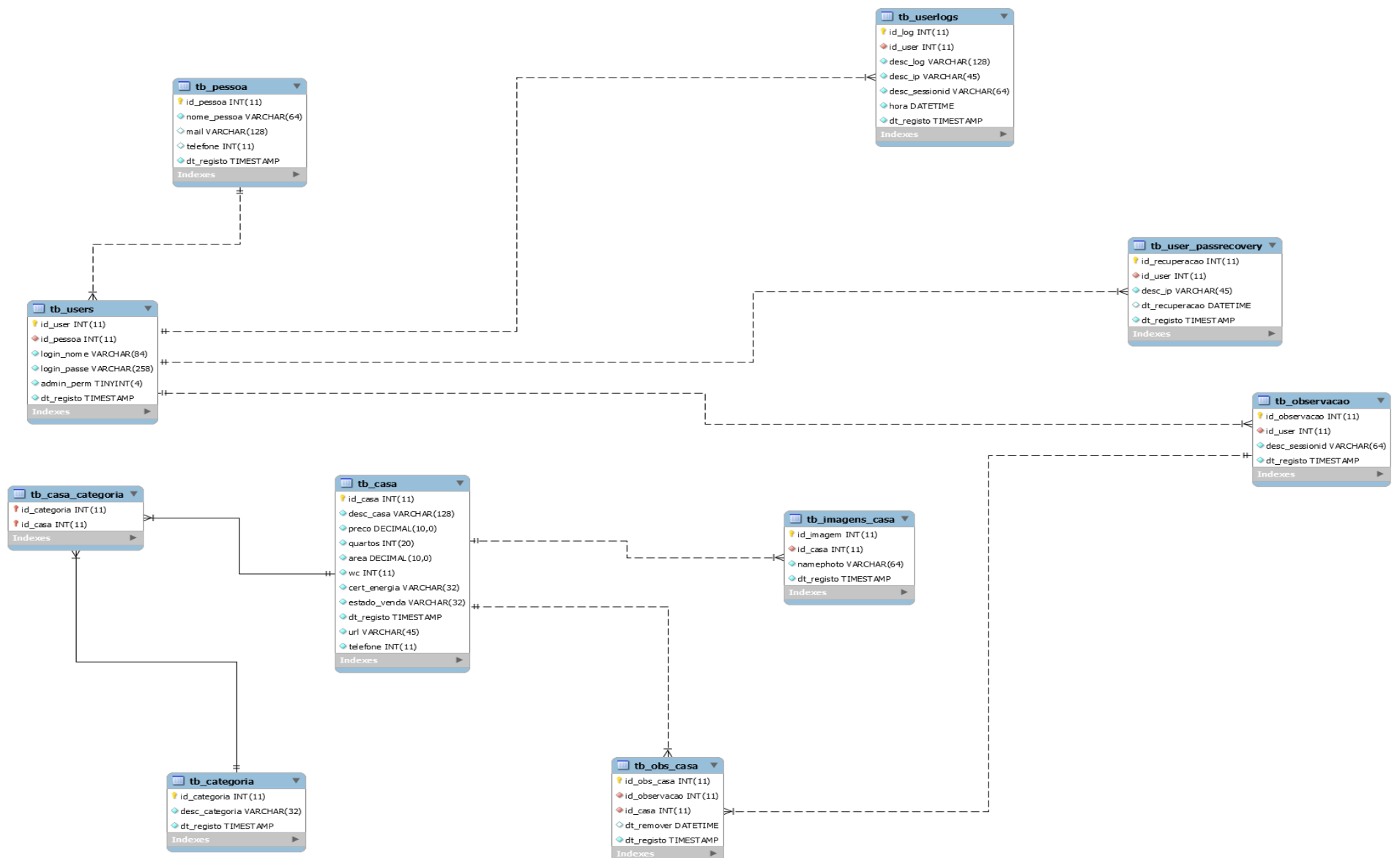


Figura 7-Diagrama-Entidade Relação criado em MySql Workbench.

3.8. – Criação da Base de Dados

A partir da ferramenta MySQL Workbench foi construída a Base de Dados relacional com as suas respetivas tabelas (imagem 7).

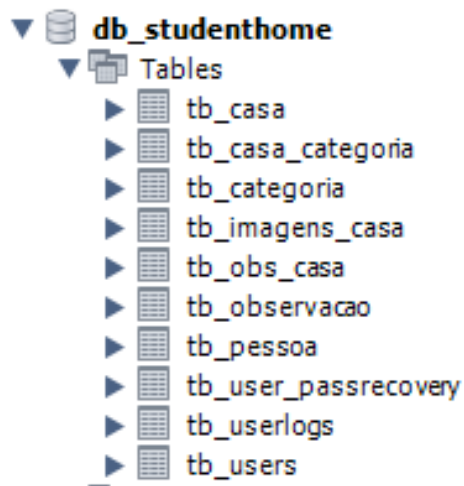


Figura 8-Tabelas da Base de Dados do Projeto.

De forma a ajudar nas funções de inserção, alteração, eliminação assim como de forma a mostrar as informações das tabelas foram criadas *stored procedures* (imagem 8).

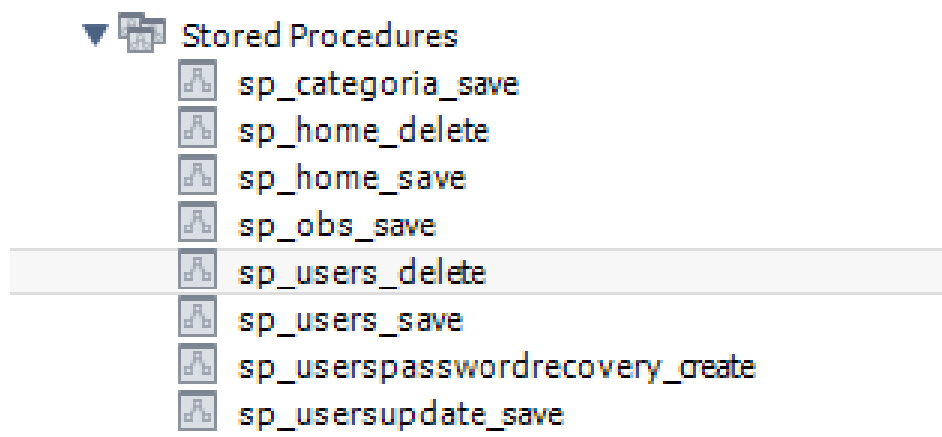


Figura 9-Stored Procedures do Projeto.

4. – Implementação

4.1. – Estruturas de Dados

4.1.1. – Arrays

Um array (imagem 9) é um “mapa ordenado” que relaciona valores a chaves do mesmo tipo de dados, esta estrutura simplifica o código, pois poupa esforço ao desenvolvedor, visto que evita a criação de várias variáveis, juntando tudo numa estrutura de dados.

```
// Definição simples e rápida
$lista = array('Pão', 'Ovos', 'Carne', 'Macarrão');

// Definição mais longa, porém mais fácil de entender
$lista = array();
$lista[0] = 'Pão';
$lista[1] = 'Ovos';
$lista[2] = 'Carne';
$lista[3] = 'Macarrão';
```

Figura 10-Exemplo de Declaração de Arrays.

4.1.2. - Class

Uma classe (imagem 10) destina-se essencialmente para agrupar informação num objeto, com isso tornamos o código muito mais limpo e centralizado. Ao fazer isto conseguimos reutilizar o código por meio instâncias do objeto pertencente a uma classe.

Com esta estrutura de dados, o código torna-se mais fácil de alterar, caso seja necessário e fica mais organizado.

É importante referir que em PHP não é obrigatório usar POO, e não é uma forma nem mais nem menos correta de programar.

```
class Produto {  
  
    private $codigo;  
    private $nome;  
    private $preco;  
  
    public function getCodigo()  
    {  
        return $this->codigo;  
    }  
    public function setCodigo($codigo)  
    {  
        $this->codigo = $codigo;  
    }  
}
```

Figura 11-Exemplo de uma classe em PHP com variáveis e as propriedades da classe.

4.1.3. – Variables

Uma variável (imagem 11) é a maneira mais simples de armazenar a informação, em programação. A informação guardada é alocada na memória como forma de um número ou carater, por exemplo.

A variável é composta por dois elementos básicos:

- O nome que permite o uso futuro da mesma;
- O seu conteúdo;

Em PHP todas as variáveis devem começar por serem declaradas com o carater '\$' antes.

```
$var = 'Bob';  
$Var = 'Joe';  
echo "$var, $Var";
```

Figura 12- Declaração e uso das variáveis através da função echo.

4.1.4. – Constants

Uma constante (imagem 12) é um identificador(nome) para um valor único. Como o nome indica, o valor da constante não pode mudar ao longo da execução de um determinado código.

As constantes são *case-sensitive* (sensível a letras maiúsculas/minúsculas) por padrão. Por convenção, identificadores de constantes são sempre em maiúsculas.

```
const NOME = 'Alex'; //Fora do escopo de classe

class Empresa {
    const NOME_EMPRESA = 'DevMedia'; //Dentro do escopo de classe
}
```

Figura 13– Declaração de uma constante em PHP fora da classe (em cima) e dentro da classe (em baixo).

4.2. – Aplicação

4.2.1. – *BackOffice*

4.2.1.1. – Acesso á Aplicação

De forma a aceder ao *Backoffice* (imagem 13) do *website* o utilizador deve introduzir, em qualquer *browser*, o seguinte endereço <http://www.studenthome.pt/admin/login>.

Aqui o utilizador encontrará a opção de fazer *Login* no sistema. É também possível alterar a palavra passe, caso o utilizador se tenha esquecido da mesma.

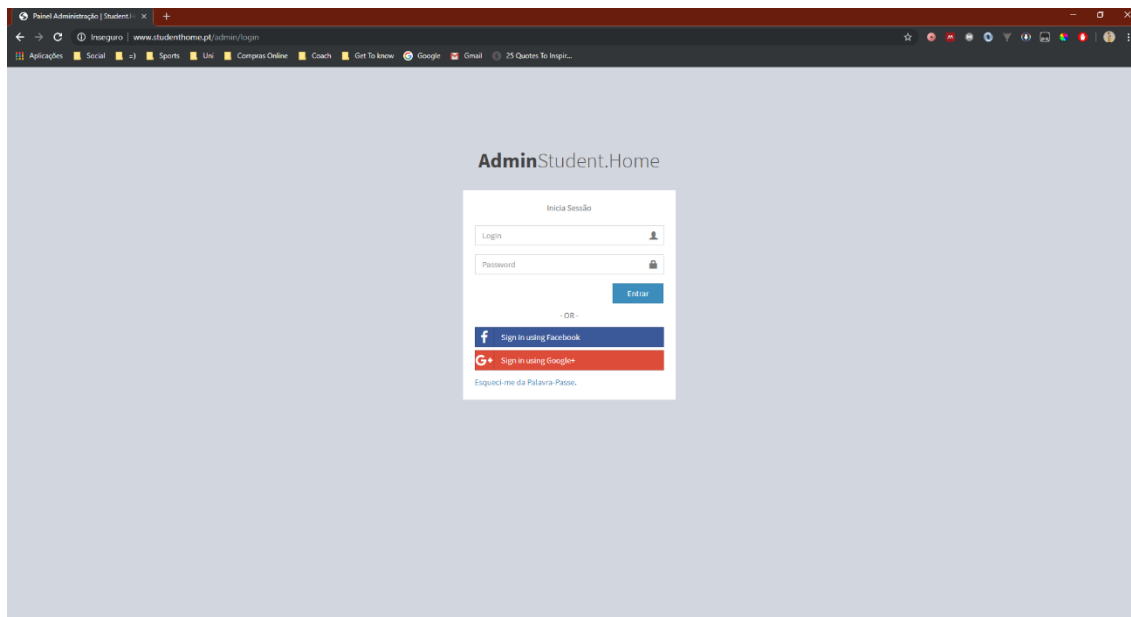


Figura 14- Acesso à Aplicação

Nota:

O url referido em cima não corresponde a um domínio obtido, de forma a tornar o uso desta aplicação o mais próximo da realidade que se pretende, modificou-se o ficheiro *hosts* (contido no seguinte caminho C:\Windows\System32\drivers\etc) e relacionou-se o *localhost* (127.0.0.1) ao url www.studenthome.pt (imagem 14).

```

1 # Copyright (c) 1993-2009 Microsoft Corp.
2 #
3 # This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
4 #
5 # This file contains the mappings of IP addresses to host names. Each
6 # entry should be kept on an individual line. The IP address should
7 # be placed in the first column followed by the corresponding host name.
8 # The IP address and the host name should be separated by at least one
9 # space.
10 #
11 # Additionally, comments (such as these) may be inserted on individual
12 # lines or following the machine name denoted by a '#' symbol.
13 #
14 # For example:
15 #
16 #       102.54.94.97       rhino.acme.com           # source server
17 #       38.25.63.10       x.acme.com              # x client host
18
19 # localhost name resolution is handled within DNS itself.
20 #   127.0.0.1       localhost
21 #   ::1             localhost
22
23 185.104.184.130 de-005.staticnetcontent.com #added by Windscribe, do not modify.
24
25
26 127.0.0.1 www.studenthome.pt
27
28

```

Figura 15– Ficheiro Hosts

4.2.1.1.1. – Esquecimento Da Palavra-Passe

Caso o utilizador não saiba a password, é pedido o seu e-mail para ser enviado um e-mail com um *link* de redefinição de palavra-passe.

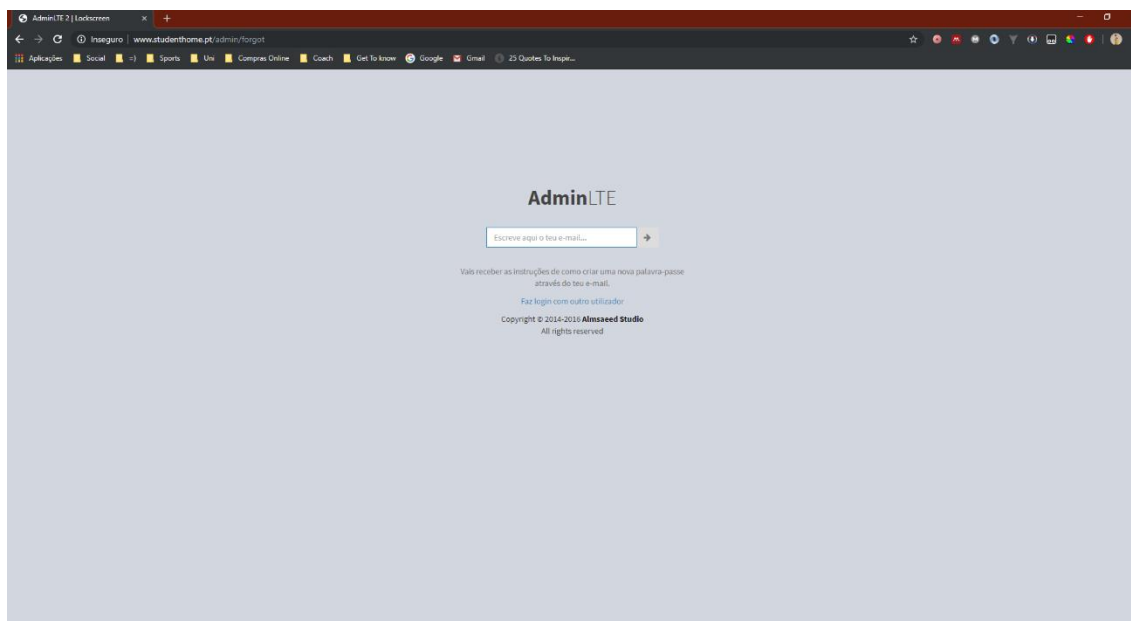


Figura 16– Pedido de E-mail para redefinição de palavra-passe.

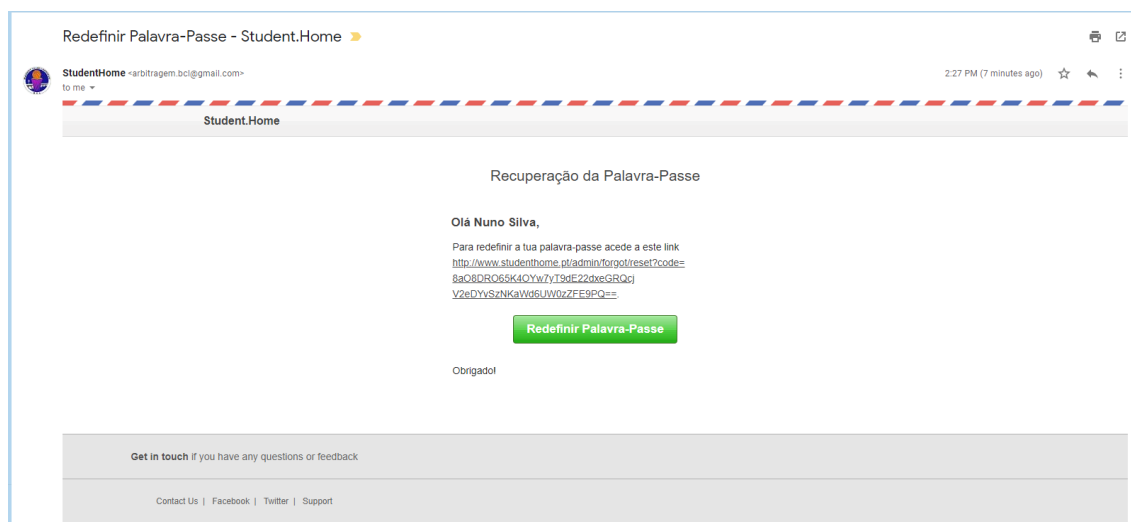


Figura 17– Mail com o link para redefinir Palavra-Passe.

Após clicar nesse link recebido no E-mail, é então possível mudar a palavra-passe.

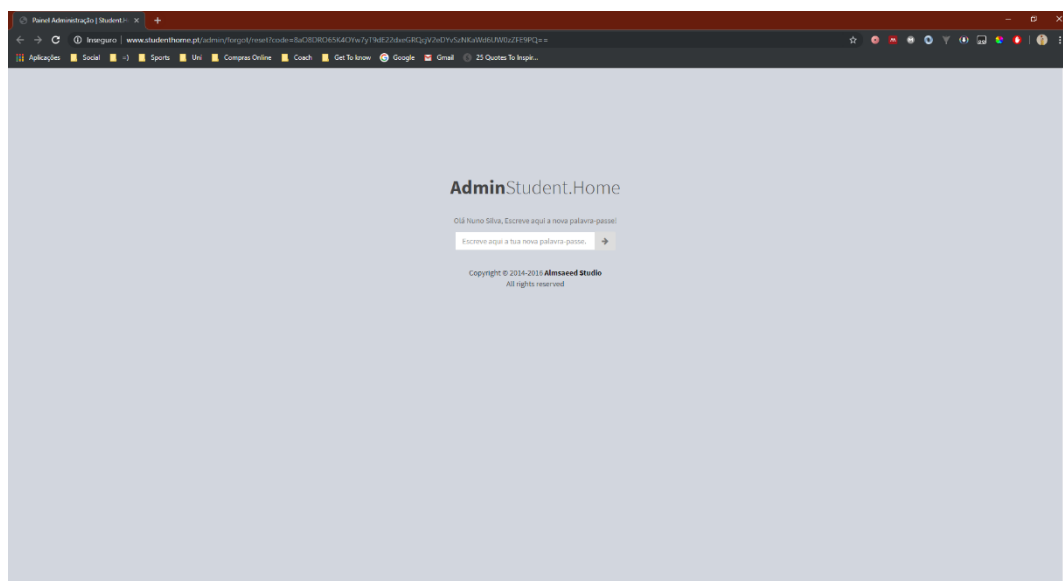


Figura 18– Redefinição da Palavra-Passe.

4.2.1.2. - Dashboard do Backoffice

Após o utilizador inserir as suas credenciais, entra no *dashboard* do *Backoffice*, aqui, poderá ver informações importantes, como os últimos imóveis adicionados ou os últimos visitantes que criaram conta no *website*.

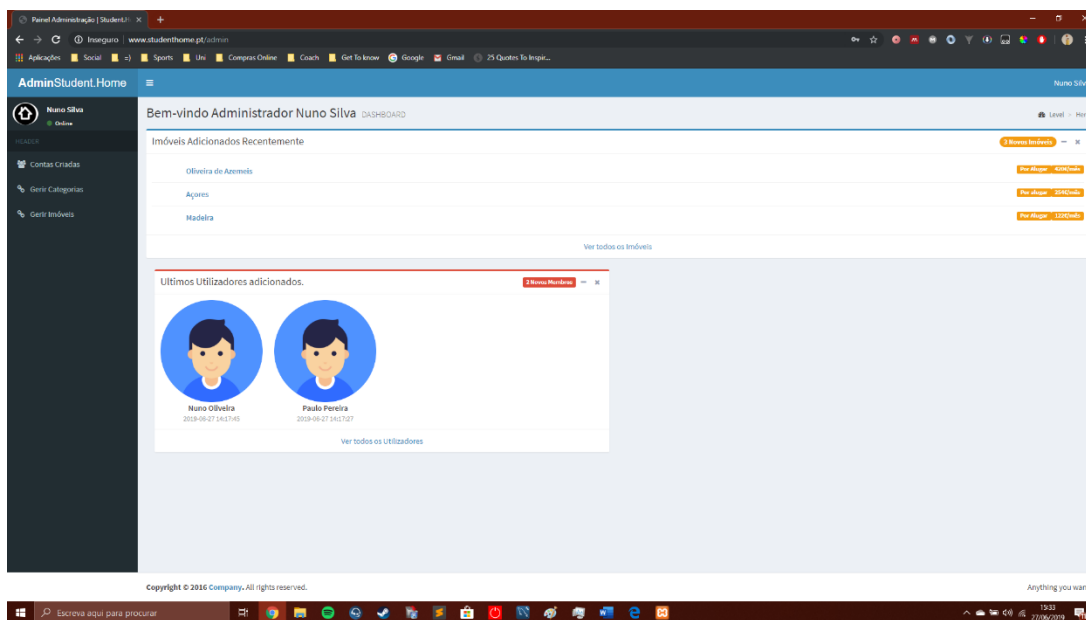


Figura 19– Dashboard

4.2.1.3. – Gestão das Contas

Após o utilizador clicar na aba “Contas Criadas”, um ecrã com todas as contas criadas vai aparecer, sendo possível criar uma nova, editar ou eliminar as existentes, e mudar somente a palavra-passe da sua conta.

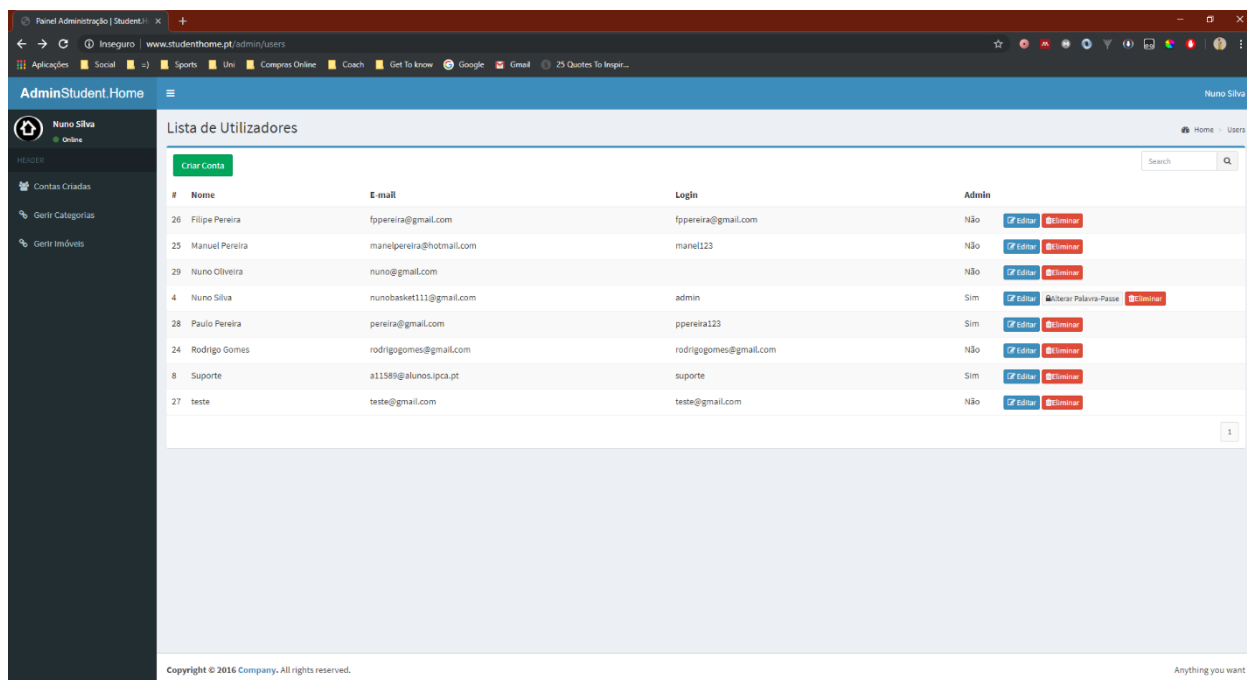


Figura 20- Lista dos Utilizadores registados no website.

Caso o administrador pretenda criar uma conta, deve clicar no botão verde no canto superior esquerdo, o que lhe levará para o seguinte ecrã:

Figura 21- Registo de um novo Utilizador

Caso se pretenda que este novo utilizador tenha acesso a este *backoffice* a caixa que diz Acesso de Administrado deve ser preenchida.

Se se pretender editar o utilizador, também é possível através do botão “Editar” no ecrã da lista de utilizadores.

AdminStudent.Home

Nuno Silva

Lista de Utilizadores

Editar User

Nome: Filipe Pereira

Login: fppereira@gmail.com

Telefone: 0

E-mail: fppereira@gmail.com

☐ Acesso de Administrador

Guardar

Figura 22– Edição de um utilizador.

Caso se pretenda eliminar um utilizador, é possível carregando no botão “Eliminar”, após o clique aparecerá uma janela de confirmação para o utilizador confirmar a ação.

Panel Administração | Student

www.studenthome.pt/admin/users

AdminStudent.Home

Nuno Silva

Lista de Utilizadores

Criar Conta

#	Nome	E-mail	Login	Admin
26	Filipe Pereira	fppereira@gmail.com	fppereira@gmail.com	Não
25	Manuel Pereira	manelpereira@hotmail.com	manel123	Não
29	Nuno Oliveira	nuno@gmail.com		Não
4	Nuno Silva	nunobasket111@gmail.com	admin	Sim
28	Paulo Pereira	pereira@gmail.com	ppereira123	Sim
24	Rodrigo Gomes	rodrigogomes@gmail.com	rodrigogomes@gmail.com	Não
8	Suporte	a11589@alunos.ipca.pt	suporte	Sim
27	teste	teste@gmail.com	teste@gmail.com	Não

Deseja realmente eliminar esta Conta?

OK Cancelar

Figura 23- Eliminar o Utilizador

Finalmente, é também possível alterar a palavra-passe do utilizador que está “logado”, para isso deverá ser carregado o botão “Alterar Palavra-Passe”, e logo a seguir aparecerá o seguinte ecrã:

The screenshot shows a web browser window with the URL `www.studenthome.pt/admin/users/42/password`. The page title is "Editar Palavra-Passe do User". It features a sidebar on the left with the user's name "Nuno Silva" and a menu with options like "Contas Criadas", "Gerir Categorias", and "Gerir Imóveis". The main content area contains a form with two input fields: "Nova Palavra-Passe" and "Confirma a Palavra-Passe", followed by a "Guardar Alterações" button. The footer includes a copyright notice for 2016 and the text "Anything you want".

Figura 24– Alterar Palavra-Passe

4.2.1.4. – Gestão de Categorias

Na gestão de categorias existentes, para ir para o ecrã de gestão das mesmas é necessário que o utilizador clique no botão da aba “Gerir Categorias”, após esse clique irá deparar-se com o seguinte ecrã:

The screenshot shows the "Lista de Categorias de Imóveis" page. It includes a sidebar with the user "Nuno Silva" and a menu with "Contas Criadas", "Gerir Categorias", and "Gerir Imóveis". The main content area has a green button "Adicionar Categoria de Imóvel" and a search bar. Below is a table with three rows of property categories. Each row has buttons for "Imóveis", "Editar", and "Eliminar".

#	Nome da Categoria	Imóveis	Editar	Eliminar
1	Apartamento	Imóveis	Editar	Eliminar
3	Casa	Imóveis	Editar	Eliminar
2	Quarto	Imóveis	Editar	Eliminar

Figura 25– Lista de Categorias de Imóveis.

Caso se pretenda adicionar uma nova categoria de Imóvel é também possível carregando no botão verde no canto superior esquerdo, esta ação irá levar o utilizador para o seguinte ecrã:

AdminStudent.Home

Nuno Silva

Home > Categorias > Adicionar Categoria de Imóvel

Lista de Categorias de Imóveis

Nova Categoria de Imóvel

Nome da categoria de imóvel

Escreve aqui a categoria

Adicionar Categoria

Copyright © 2016 Company. All rights reserved. Anything you want

Figura 26- Registo de uma nova Categoria de Imóvel.

Caso se pretenda editar a categoria, é também possível, carregando no botão para esse efeito no ecrã de gestão de categorias:

AdminStudent.Home

Nuno Silva

Home > Categorias > Adicionar Categoria de Imóvel

Lista de Categorias de Imóveis

Editar Categoria

Nome da categoria de imóvel

Apartamento

Guardar Categoria

Copyright © 2016 Company. All rights reserved. Anything you want

Figura 27- Edição de uma Categoria.

A forma de eliminar uma categoria é a mesma que a de um utilizador.

Para adicionarmos imóveis a uma certa categoria, devemos carregar no botão “Imóveis”, a partir daí aparece um ecrã com duas listas, a lista da esquerda que são todos os imóveis que não pertencem à categoria em questão, podemos adicionar à categoria carregando no botão “Adicionar”. A lista da direita mostra todos os imóveis que já estão dentro dessa categoria, podemos remover um Imóvel da lista em questão, carregando no botão “Remover”.

Os imóveis podem ter várias categorias (apartamento, quarto, casa etc..).

4.2.1.5. – Gestão de Imóveis

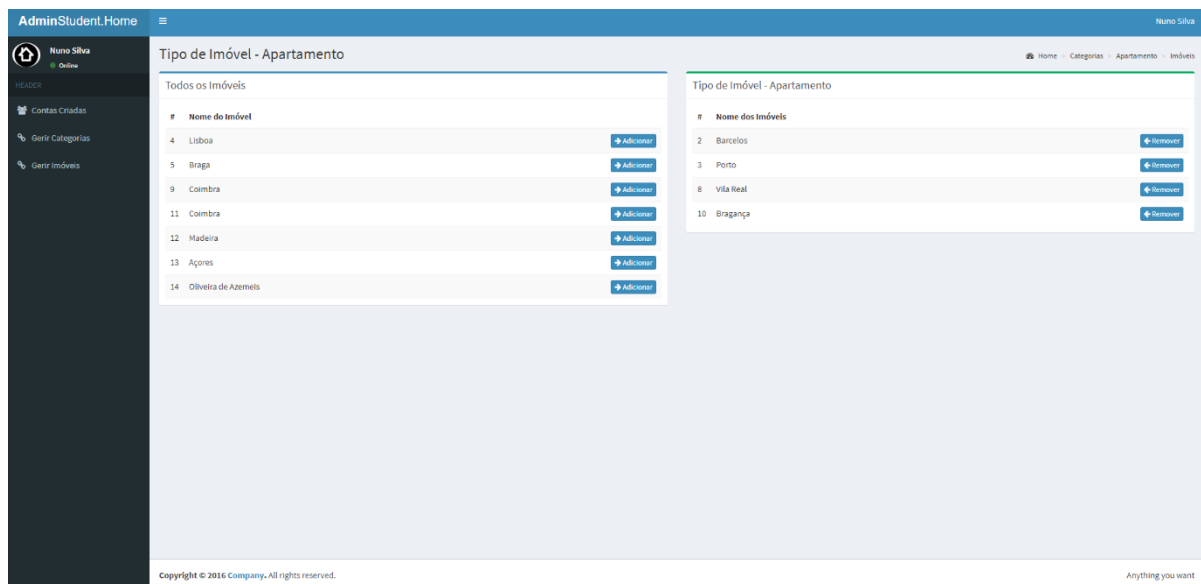


Figura 28- Gerir Categorias dos Imóveis.

A última funcionalidade deste *back office* é a gestão dos imóveis do *website*. Para isso é necessário clicar na opção “Gerir Imóveis” na aba do *dashboard*.

Após o clique, aparecerá o ecrã com a lista de todos os imóveis já registados na base de dados do sistema.

AdminStudent.Home

Nuno Silva

Lista de Imóveis

Adicionar Imóvel

#	Imóvel	Preço	Quartos	Área	Casas de Banho	Certificado de Energia	Estado de venda do Imóvel	
13	Açores	254	3	122	2	B	Por alugar	Editar Eliminar
2	Barcelos	150	1	251	3	C	Por Alugar	Editar Eliminar
5	Braga	150	1	120	1	C	Por Alugar	Editar Eliminar
10	Bragança	123	2	130	2	D	Por Alugar	Editar Eliminar
9	Coimbra	120	3	123	2	D	Por Alugar	Editar Eliminar
11	Coimbra	100	1	100	1	E	Por Alugar	Editar Eliminar
4	Lisboa	350	2	285	2	D	Por Alugar	Editar Eliminar
12	Madeira	122	2	123	2	C	Por Alugar	Editar Eliminar

Copyright © 2016 Company. All rights reserved.

Anything you want

Figura 29– Lista dos Imóveis existentes.

Caso o utilizador queira adicionar um novo imóvel, é possível carregando no botão verde no canto superior esquerdo:

AdminStudent.Home

Nuno Silva

Lista de Imóveis

Adicionar Imóvel

Imóvel

Local do Imóvel

Preço Imóvel

0.00/Mês

Quartos

1,2,3

Área

0.00 m2

Casas de Banho

1,2,3

Certificado de Energia

A,B,C,D,E

Estado de Venda do Imóvel

Alugado/For Alugar

URL

Telefone do Proprietário

Adicionar Imóvel

Figura 30– Registo de um novo Imóvel.

Após o registo o imóvel fica com uma imagem predefinida pelo sistema, para alterar essa imagem, e adicionar mais imagens é necessário editar o imóvel, pode ser feito através do botão “Editar” no ecrã que lista todos os imóveis existentes:

The screenshot shows a web application interface for managing properties. The header bar is blue with 'AdminStudent.Home' on the left and 'Nuno Silva' on the right. A dark sidebar on the left contains a home icon and a menu with 'Contas Criadas', 'Gerir Categorias', and 'Gerir Imóveis'. The main content area is titled 'Lista de Imóveis' and contains a form for editing a property. The form fields are: 'Imóvel' (text input), 'Preço Imóvel' (text input with value '254'), 'Quartos' (text input with value '3'), 'Área' (text input with value '122'), 'Casas de Banho' (text input with value '2'), 'Certificado de Energia' (text input with value 'B'), and 'Estado de Venda do Imóvel' (text input with value 'Por alugar'). Below these is a 'Fotos Imovel' section with an 'Escolher Ficheiros' button and the text 'Nenhum ficheiro selecionado'. At the bottom of the form is a blue 'Guardar Alterações' button.

Figura 31– Edição de Imóveis

A forma de eliminar o imóvel é a mesma que os outros campos, carregamos no botão “Eliminar” no ecrã que lista todos os imóveis e seguidamente vai aparecer uma mensagem que pede para confirmar a ação.

4.2.2. - Website

Ao entrar no url: www.studenthome.pt, o visitante depara-se com o seguinte ecrã:

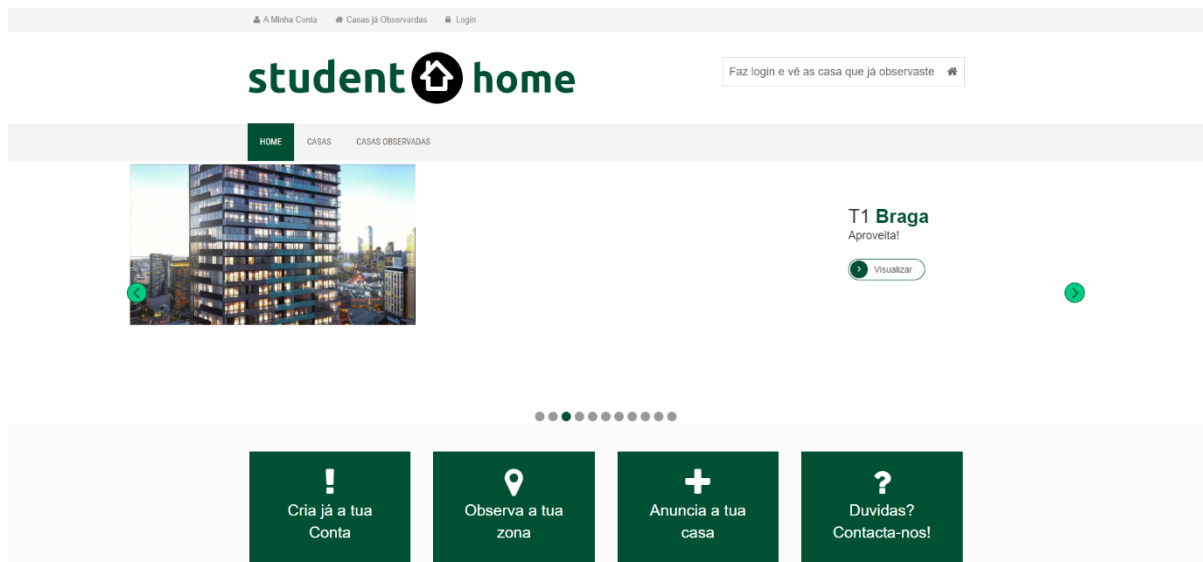


Figura 32- Página Principal do website, parte de cima.

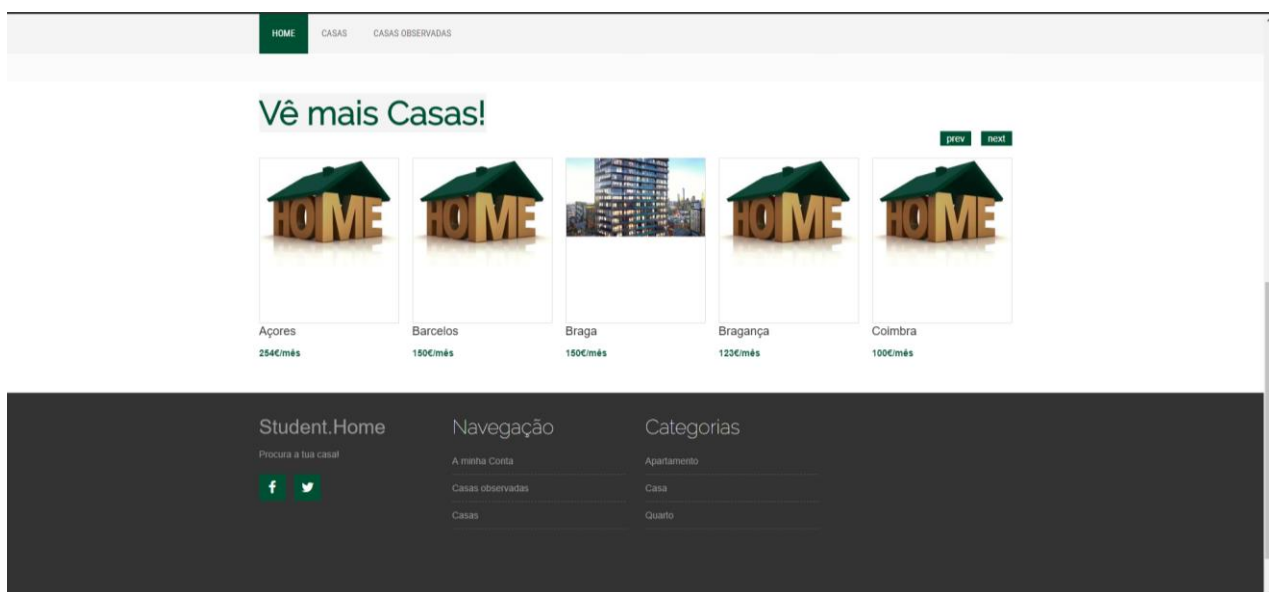


Figura 33- Página Principal do website, parte de baixo.

4.2.2.1. – Imóveis

Quando um visitante/cliente pretende visualizar todos os imóveis registados na base de dados, basta carregar em casas no painel de controlo da página principal, após essa ação, irá deparar-se com o seguinte ecrã:

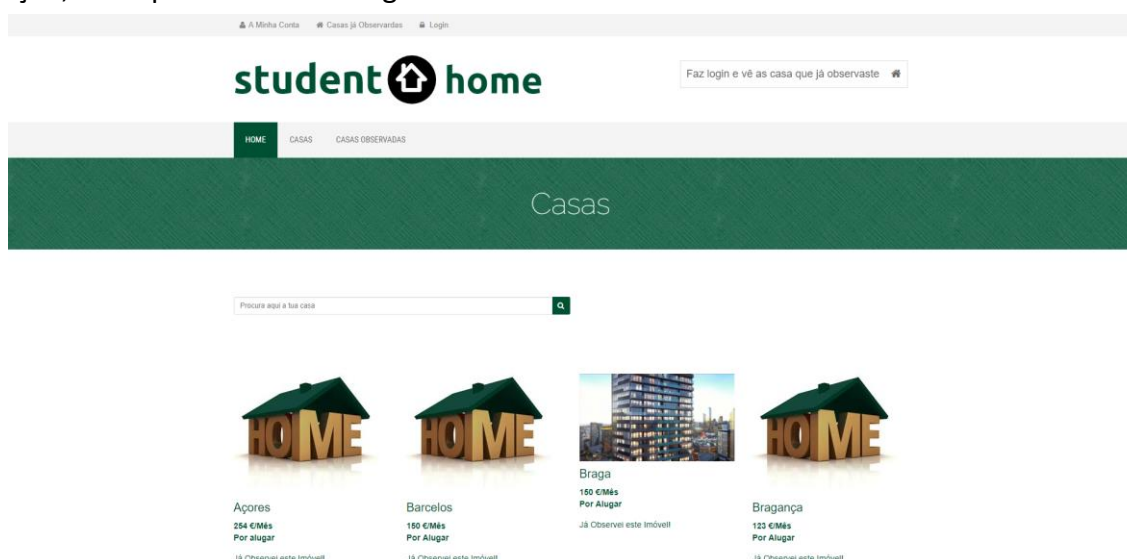


Figura 34- Casas registadas na Base de dados.

É também possível fazer uma pesquisa dos imóveis para alugar na sua zona, usando a barra de pesquisa.

A pesquisa pode ser feita através do local, preço do imóvel ou até o estado do imóvel.

Após o utilizador encontrar o anúncio ideal para si, deve carregar nele para ver com detalhe, lá encontrará todos os detalhes do imóvel, assim como o telefone do proprietário caso tenha conta no website.

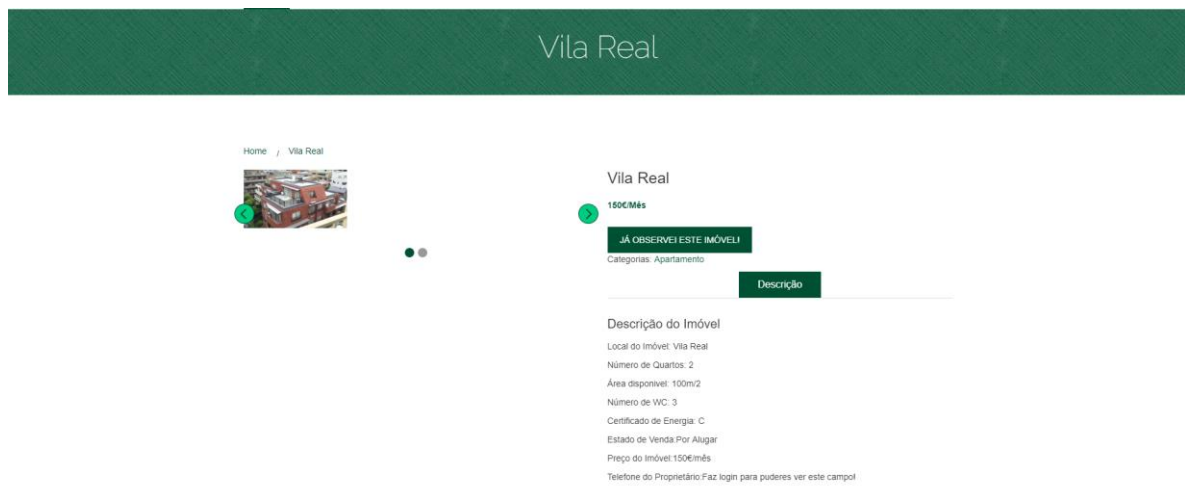


Figura 35- Detalhes de um imóvel através de um visitante.

Para o utilizador conseguir saber que imóveis foi observando, e caso esteja logado, é possível criar uma lista que auxilie este utilizador, para isso basta carregar nos vários botões que dizem “Já Observei este Imóvel!”

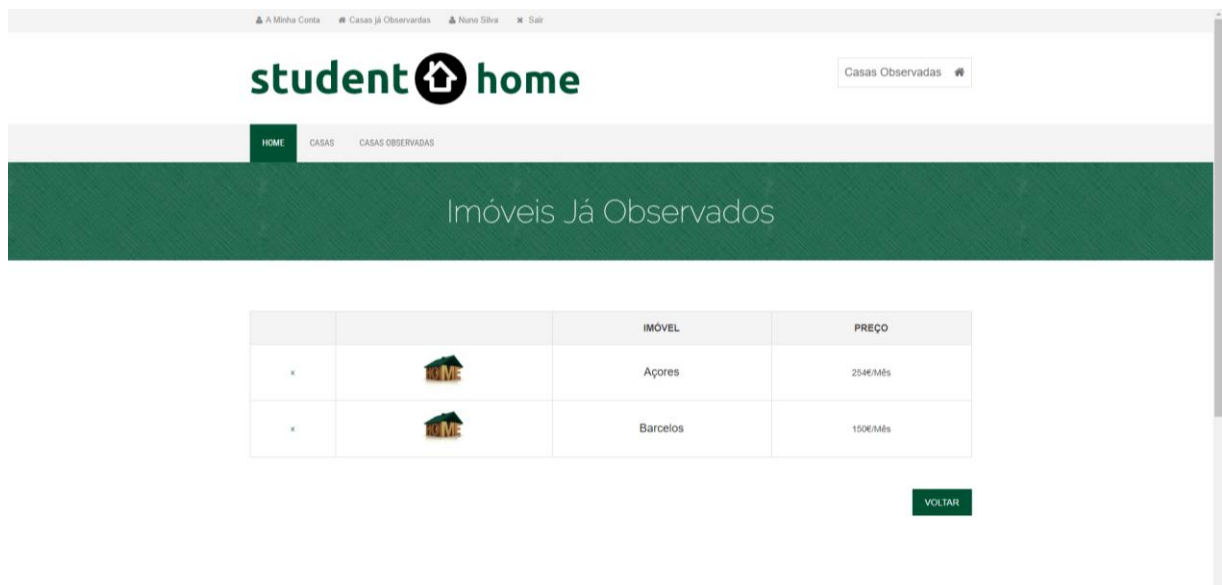
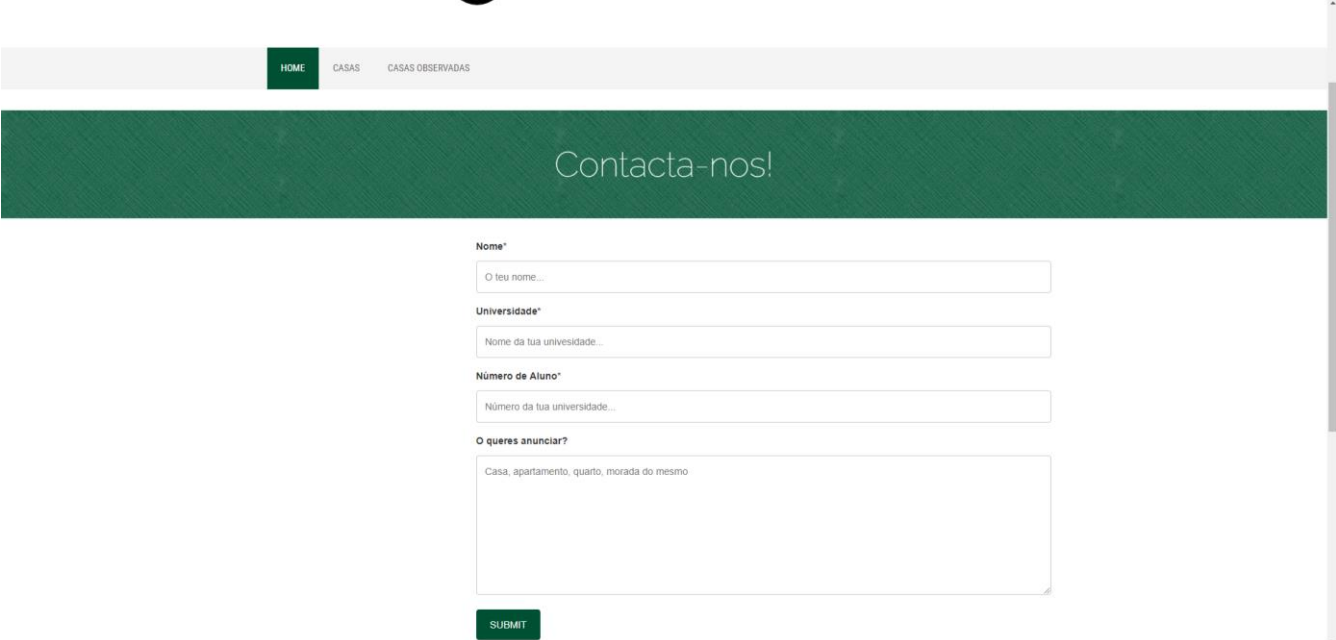


Figura 36– Imóveis Observados pelo utilizador.

Caso o utilizador logado queira contactar os administradores de forma a ver se o seu imóvel é elegível para ser anunciado no *website*, existe um formulário de contacto, onde o utilizador tem de nos dizer o seu nome, a sua universidade, o seu número de Aluno, morada do imóvel e algumas outras informações.



HOME CASAS CASAS OBSERVADAS

Contacta-nos!

Nome*

Universidade*

Número de Aluno*

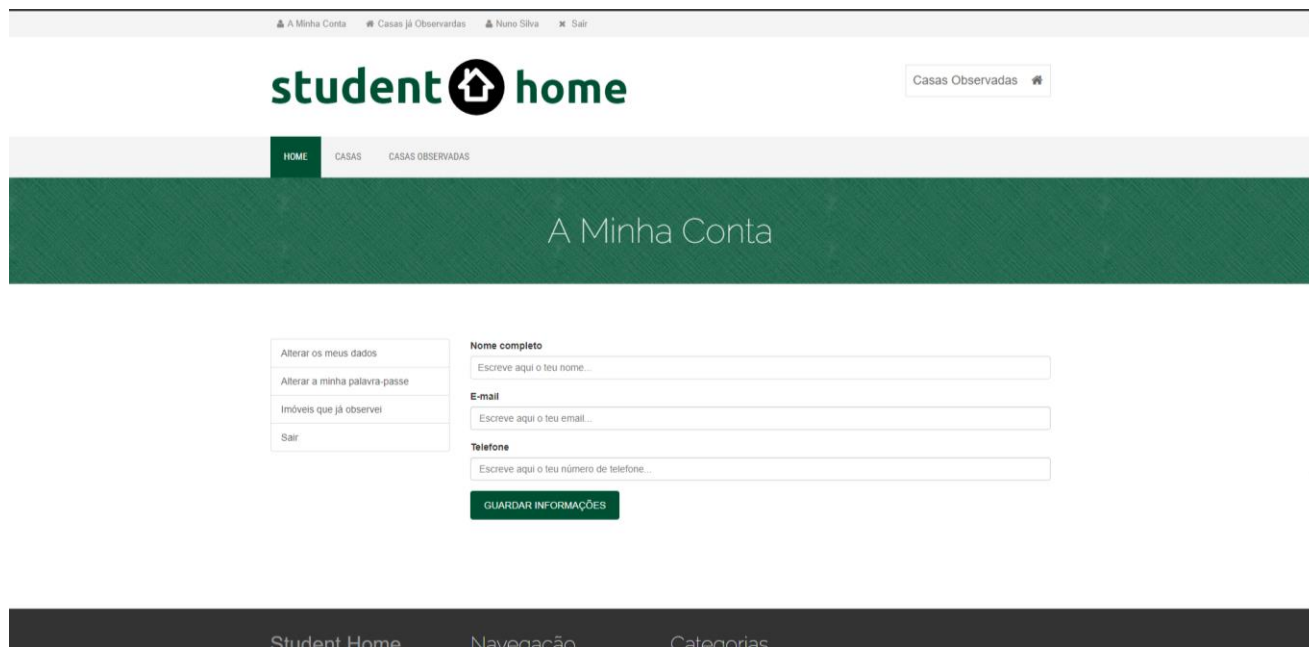
O queeres anunciar?

SUBMIT

Figura 37-Formulário de contacto para avaliar imóvel.

4.2.2.2. – Conta de Utilizador

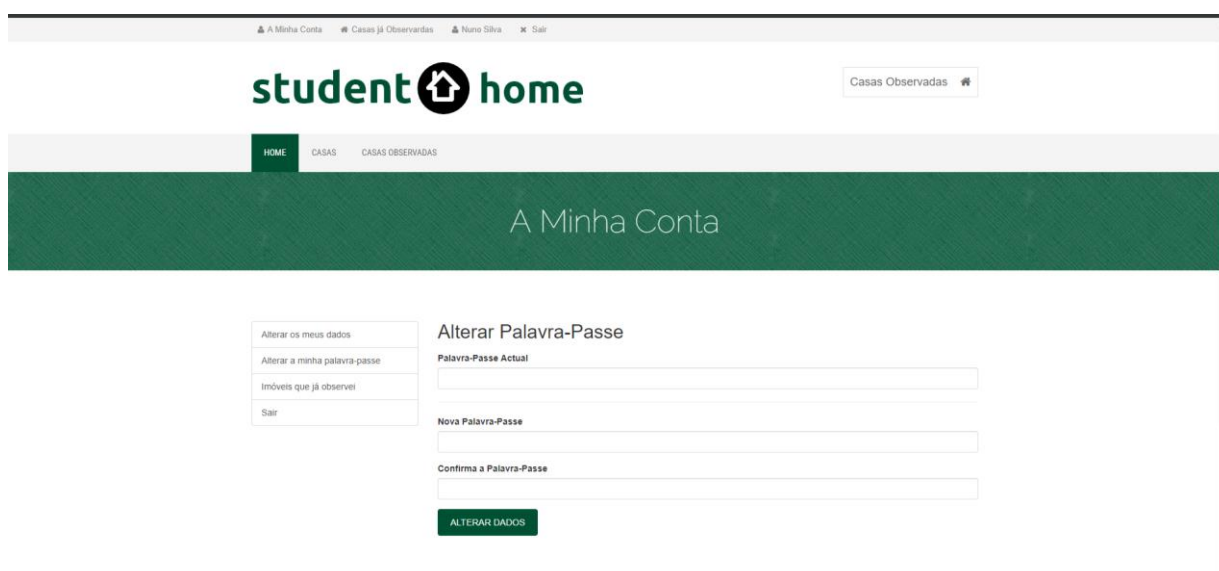
O utilizador tem possibilidade de editar os seus dados de conta, para isso no canto superior tem uma opção chamada “A Minha Conta”, após carregar ai, o utilizador tem duas opções, mudar os seus dados menos importantes, como o Nome, E-mail, telefone.



The screenshot shows the 'A Minha Conta' page of the 'student home' website. The page has a dark green header with the 'student home' logo and a 'Casas Observadas' button. Below the header is a navigation bar with 'HOME', 'CASAS', and 'CASAS OBSERVADAS'. The main content area is titled 'A Minha Conta' and contains a form for editing user data. On the left, there is a sidebar with links: 'Alterar os meus dados', 'Alterar a minha palavra-passe', 'Imóveis que já observei', and 'Sair'. The main form has three sections: 'Nome completo' with a text input field, 'E-mail' with a text input field, and 'Telefone' with a text input field. Below these fields is a green button labeled 'GUARDAR INFORMAÇÕES'.

Figura 38– Editar dados da Minha Conta.

Caso o utilizador deseje mudar somente a palavra-passe:

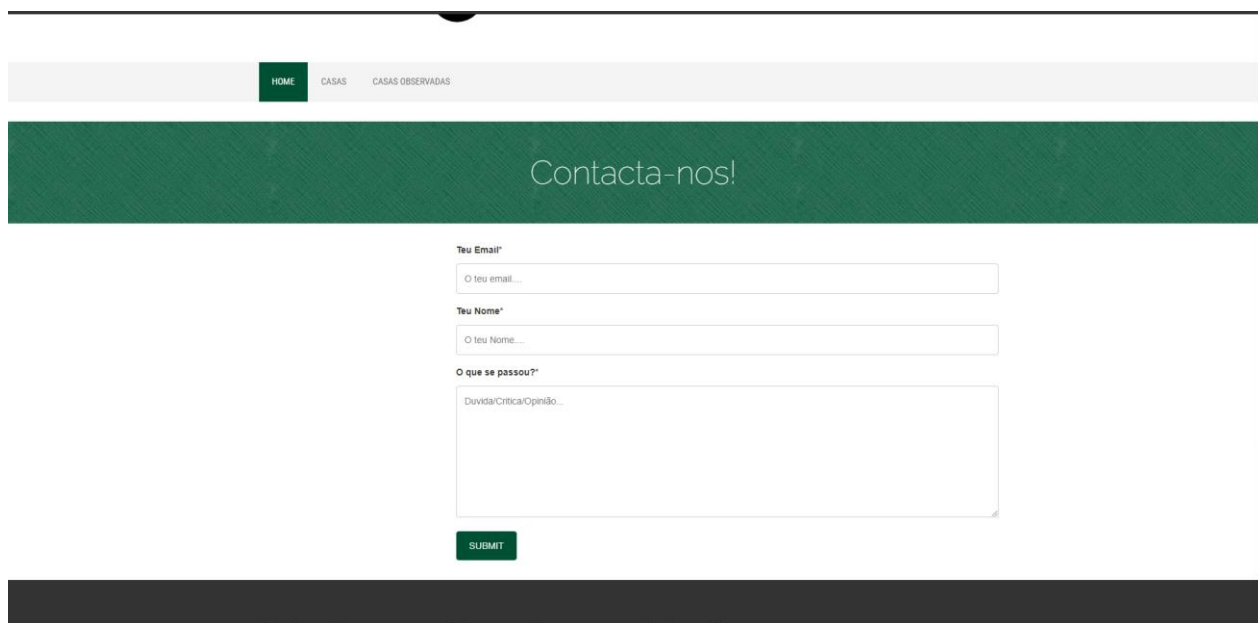


The screenshot shows the 'A Minha Conta' page of the 'student home' website, specifically the 'Alterar Palavra-Passe' form. The page layout is identical to the previous screenshot, but the main form section is titled 'Alterar Palavra-Passe'. It contains three text input fields: 'Palavra-Passe Actual', 'Nova Palavra-Passe', and 'Confirma a Palavra-Passe'. Below these fields is a green button labeled 'ALTERAR DADOS'.

Figura 39- Alterar Palavra-Passe.

4.2.2.3. – Formulário de Contacto

De forma a o utilizador conseguir contactar-nos, para retirar duvidas, dar opiniões/criticas, foi criado um formulário de contacto.



The screenshot shows a web interface with a navigation bar at the top containing links for HOME, CASAS, and CASAS OBSERVADAS. Below this is a green banner with the text "Contacta-nos!". The main content area contains a contact form with the following fields:

- Teu Email***: A text input field with a placeholder "O teu email...".
- Teu Nome***: A text input field with a placeholder "O teu Nome...".
- O que se passou?***: A large text area with a placeholder "Dúvida/Critica/Opinião...".
- SUBMIT**: A green button to submit the form.

The form is set against a light gray background with a subtle pattern.

Figura 40– Formulário de Contacto.

5. – Conclusão

Em forma de desfecho do presente relatório do projeto final de curso, o qual incidiu na explicação do desenvolvimento da aplicação “Student Home”.

Posso dizer que foi sem dúvida uma aprendizagem muito importante para o meu futuro, pois além de me obrigar a ser “auto-didata” e aprender uma nova linguagem, totalmente do zero, deu-me uma noção de como uma aplicação como um *website* deve funcionar.

É, no entanto, importante referir que ao longo deste processo, foram sendo detetados e corrigidos alguns erros (quer estéticos, quer nas funções), no entanto acreditamos que este processo não tenha chegado ao fim.

É uma aplicação acessível e fácil de navegar, não sendo necessário nenhuma formação para usar.

É também fácil de gerir através do *Back Office*, sendo muito intuitivo.

No geral, todos os objetivos propostos no início do projeto, foram concluídos, uma vez que nos foi possível criar um produto capaz de auxiliar os estudantes numa das etapas mais complicadas, etapa essa em que, na nossa ótica, deve ser facilitada através de mecanismos como este.

Bibliografia

- Análise de Requisitos*. (2019). Obtido de InfoEscola:
<https://www.infoescola.com/engenharia-de-software/analise-de-requisitos/>
- CRUZ, B. B. (30 de Outubro de 2017). *Programação Orientada a Objetos no PHP – Parte 1*. Obtido de UniaoGeek: <https://www.uniaogeek.com.br/poo-no-php-parte-1/>
- Haughey, D. (2018). *MOSCOW METHOD*. Obtido de Project Smart:
<https://www.projectsmart.co.uk/moscow-method.php>
- Hcode. (2019). *Hcode*. Obtido de <https://www.hcode.com.br/>.
- Hcode. (s.d.). *Curso de PHP*. Obtido de <https://www.udemy.com/curso-php-7-online/>.
- <https://casadaconsultoria.com.br/modelo-cascata/>. (2019). *Modelo Cascata: O que é e como funciona?* Obtido de Casa da Consultadoria:
<https://casadaconsultoria.com.br/modelo-cascata/>
- Janaína. (2009). *Técnicas para levantamento de Requisitos*. Obtido de devmedia:
<https://www.devmedia.com.br/tecnicas-para-levantamento-de-requisitos/9151>
- Mark Otto, J. T. (2019). *GitHub*. Obtido de BootStrap:
<https://github.com/twbs/bootstrap>
- PHP. (2019). *PHP*. Obtido de Manual do PHP: https://www.php.net/manual/pt_BR/
- PHP. (2019). *PHP MANUAL*. Obtido de PHP:
https://www.php.net/manual/pt_BR/history.php.php
- Soeusei199. (2011). *A arquitetura MVC no desenvolvimento em PHP*. Obtido de Devmedia: <https://www.devmedia.com.br/a-arquitetura-mvc-no-desenvolvimento-em-php/23121>
- Souza, E. G. (30 de Novembro de 2016). *Programação Orientada a Objeto em PHP — Polimorfismo*. Obtido de medium: <https://medium.com/emanuelg-blog/programa%C3%A7%C3%A3o-orientada-a-objeto-em-php-polimorfismo-359ad60b947b>
- Udemy. (s.d.). *Udemy*. Obtido de <https://www.udemy.com>.

Wesley. (2012). *Criando Classe em PHP*. Obtido de DevMedia:
<https://www.devmedia.com.br/criando-classe-em-php/24371>