

**ESCOLA SECUNDÁRIA RAUL PROENÇA**  
**Curso Profissional Técnico de Gestão e**  
**Programação de Sistemas Informáticos**

**RELATÓRIO DA PROVA DE**  
**APTIDÃO PROFISSIONAL (PAP)**



Turma: 12º CP-TGPSI

Aluno: Daniel Pankratau Nº6

Orientador: Carla Jesus

**Ano Letivo: 2021/2022**

Cofinanciado por:



## Índice

1. Introdução	4
1.1. Fundamentação da escolha do projeto	4
1.2. Finalidade do projeto (Objetivo)	4
1.3. Enquadramento do projeto	4
2. Desenvolvimento	5
2.1. Tecnologias Utilizadas	5
2.2. Etapas de Desenvolvimento	6
2.2.1. Cronograma	6
2.2.2. Modelação 3D / Maquete (Projetos Hardware)	6
2.2.3. Layout / Mockups (Projetos Software)	6
2.2.4. Modelo EER	6
2.2.5. Esquema geral ligações (Projetos Hardware)	6
2.2.6. Descrição das Fases do projeto	6
2.3. Estratégias adotadas na resolução de problemas	6
3. Relatórios de desenvolvimento (opcionais)	7
4. Conclusão	8
5. Referências	9
6. Anexos	10

## Nota Introdutória

A Prova de Aptidão Profissional (PAP) consiste na apresentação e defesa, perante um júri, de um projeto, consubstanciado num produto, material ou intelectual, numa intervenção ou numa atuação, bem como do respetivo relatório final de realização e apreciação crítica, demonstrativo de saberes e competências profissionais, adquiridos ao longo da formação e estruturante do futuro profissional. A sua realização constitui-se como um dos imperativos legais para a conclusão do curso profissional de Técnico de Gestão e Programação de Sistemas Informáticos, que confere uma dupla certificação: qualificação profissional de nível IV e o 12º ano como certificação escolar de nível secundário. A presente prova foi realizada na Escola Sede do Agrupamento de Escolas Raul Proença, Caldas da Rainha, no ano letivo de 2020/2021 e a sua defesa realizada, perante um júri final, nas datas estabelecidas no calendário escolar.

## Agradecimentos

Agradeço a todos os familiares, amigos e professores que acreditaram em mim e estiveram sempre ao meu lado apoiando-me para tornar este projecto possível.

## Índice de Figuras

Não foi encontrada nenhuma entrada do índice de ilustrações.

## 1. Introdução

O projeto escolhido para a realização da minha Prova de Aptidão Profissional consiste num *website* desenvolvido com a framework Laravel, que servirá como uma ferramenta de pesquisa reunindo imóveis dos maiores websites no ramo imobiliário do mercado português, num só lugar.

### 1.1. Fundamentação da escolha do projeto

Durante o meu primeiro estágio na Janela Digital eu aprendi diversas habilidades, nomeadamente a desenvolver projectos de *web scraping*, e desde então eu desenvolvi um interesse por este tipo de técnicas. Quando dei por mim, já estava a desenvolver frequentemente projectos deste tipo, apenas por diversão, e foi então que reparei que os maiores *websites* de imóveis em Portugal, continham em seus sites vários imóveis que não existiam em outros sites, o que dificulta a pesquisa de um cliente que procura o seu produto, já que existem milhares de casas em vários sites diferentes.

Ou seja, a informação está descentralizada, o que dificulta a vida de uma pessoa que procura um imóvel, já que para tal, terá de fazer várias pesquisas em vários websites diferentes.

Foi então que surgiu a ideia de criar um site onde os clientes poderiam pesquisar e comparar os preços de vários imóveis dos maiores sites num só lugar, poupando tempo e ganhando uma ideia concreta dos preços do mercado, evitando fazer pesquisas vastas.

### 1.2. Finalidade do projeto (Objetivo)

A SherlockHomes quer ajudar as pessoas a encontrarem o seu caminho para casa, dispondo aos seus clientes uma visão clara de todo o mercado imobiliário em Portugal através de uma plataforma simples e moderna.

**Explicar em conformidade com os objetivos do Pré-Projeto**

## 1.3. Enquadramento do projeto

Este projecto é constituído por duas partes, **website** e **Webscraper**.

O website terá um *FrontOffice* simples e de fácil utilização, onde os clientes poderão pesquisar imóveis, ver as suas características e ainda adicioná-los à sua lista de favoritos. Contudo, para poderem usufruir desta última funcionalidade, terão que criar uma conta no nosso site, e caso percam a palavra passe da sua respectiva conta, bastará apenas clicar no botão de recuperação localizado na página de *login* e seguir as instruções.

No *BackOffice* os administradores poderão adicionar e editar imóveis, controlar as entradas mais recentes de imóveis na base de dados, o que é útil para ter uma ideia do comportamento do nosso *webscraper*, e ainda adicionar ou remover cargos de administrador a outras contas.

Já o *webscraper* é um script desenvolvido em python que tem como objectivo automatizar a recolha de dados e imagens de diversos sites na internet para poder popular ou atualizar se necessário, a nossa base de dados. Desta forma, o manuseamento e mapeamento automático de dados em grande escala, poupará tempo e esforço tornando o nosso trabalho prático e eficiente.

## 2. Desenvolvimento

### 2.1. Tecnologias Utilizadas

Durante o desenvolvimento deste projeto, apliquei diversas tecnologias, e como espaço de trabalho eu escolhi o **Visual Studio Code**, que é um IDE(*integrated development environment*) ou seja, um editor de código simples e fácil de usar. O Visual Studio Code permite-me instalar extensões que simplificaram o meu trabalho para poder programar em praticamente todo o tipo de linguagens de programação.



Figura 1: Visual Studio Code

O **Laravel**, é uma framework de *PHP Open Source* gratuita usada no desenvolvimento da aplicação *web* no padrão MVC (Model View Controller), foi deveras essencial na construção do meu website.

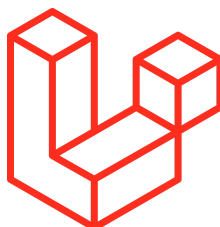


Figura 2: Laravel

Para armazenar todos os dados referentes ao meu website, eu usei a ferramenta “MySQL Workbench”, que me permite gerir um servidor local, podendo guardar todos os dados no meu computador.



Figura 3: MySQL Workbench



Para desenvolver o meu *Web Scraper*, ou seja, o software que faz a recolha de dados nos websites, usei a linguagem de programação “Python”, que mostrou ser eficaz no processo com o auxílio de um pacote chamado “*Beautiful Soup*”, usado para análise de HTML, permitindo ao meu programa, seleccionar as informações desejadas.



Figura 4: Python

O Laragon é uma ferramenta indispensável pois permite utilizar PHP de forma prática, podendo assim guardar os dados do meu *website* na minha base de dados local.



Figura 5: Laragon

Utilizei também o GitHub, onde fui guardando ao longo do tempo, backups do código do meu projeto.



Figura 6: GitHub

Para poder controlar as versões do meu projecto e guardá-las no GitHub, usei o *software* Git Kraken, o que ajudou-me a organizar e a orientar todo o processo de desenvolvimento da minha PAP.



Figura 7: GitKraken

## 2.2. Etapas de Desenvolvimento

### 2.2.1. Cronograma

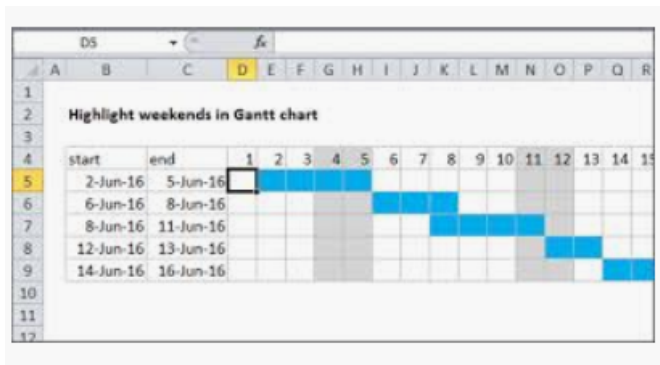


Figura 8: Cronograma

### 2.2.2. Layout / Mockups (Projetos Software)

### 2.2.3. Base de Dados

### 2.2.4. Descrição das Fases do projeto

#### WebSite

Antes de começarmos a programar, devemos preparar o nosso ambiente de trabalho começando por fazer a instalação do Laravel da seguinte forma:

- Instalar a versão mais recente do PHP;

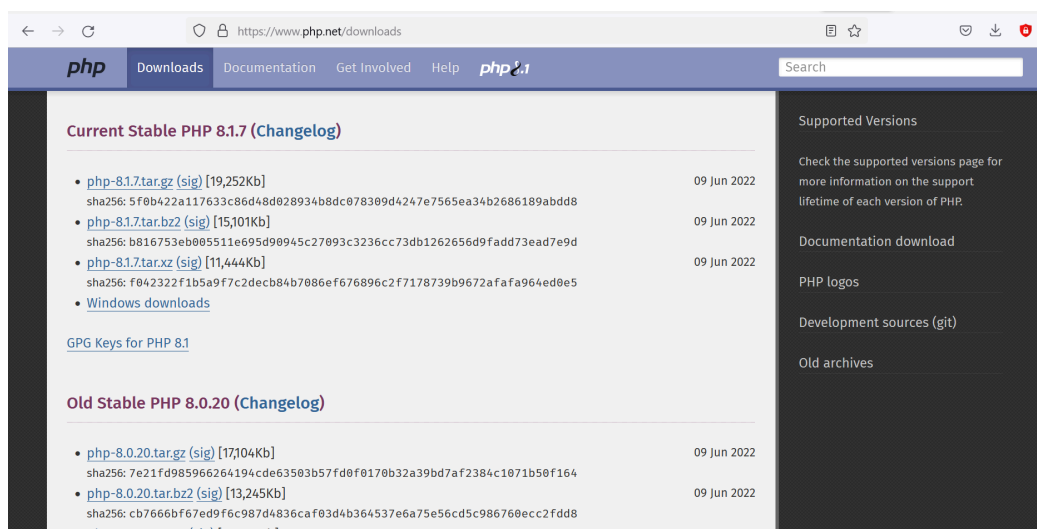
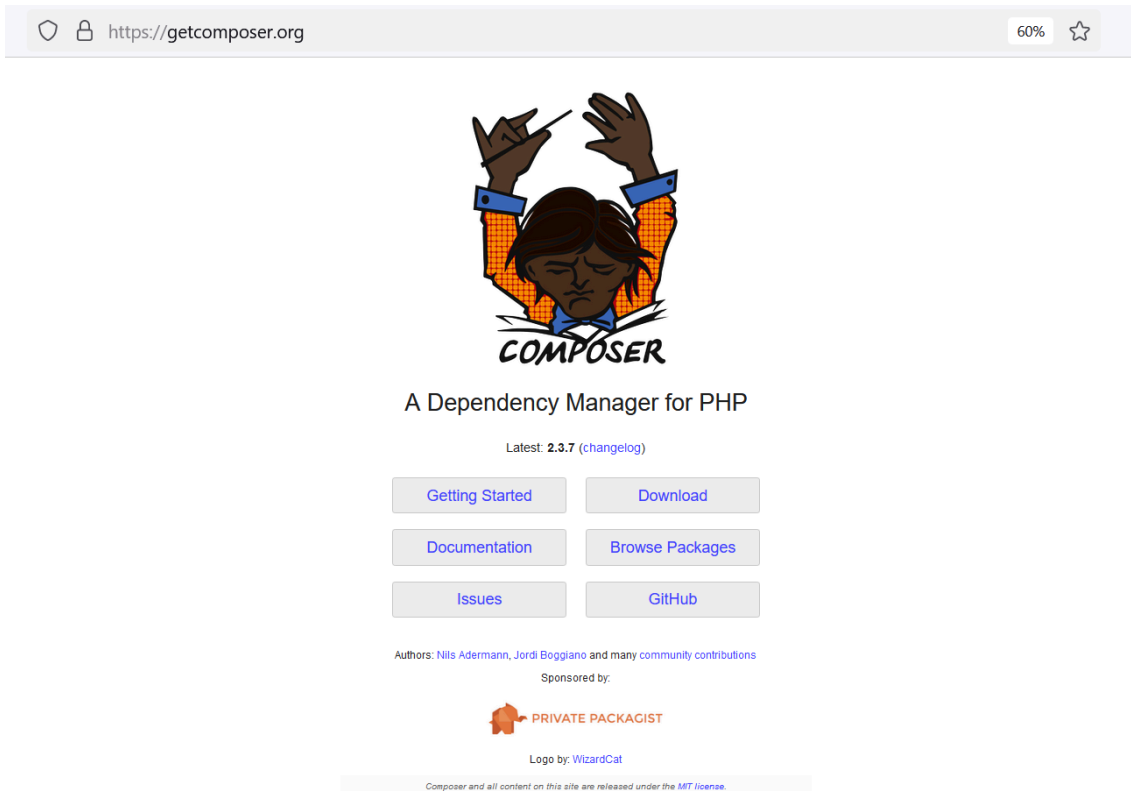
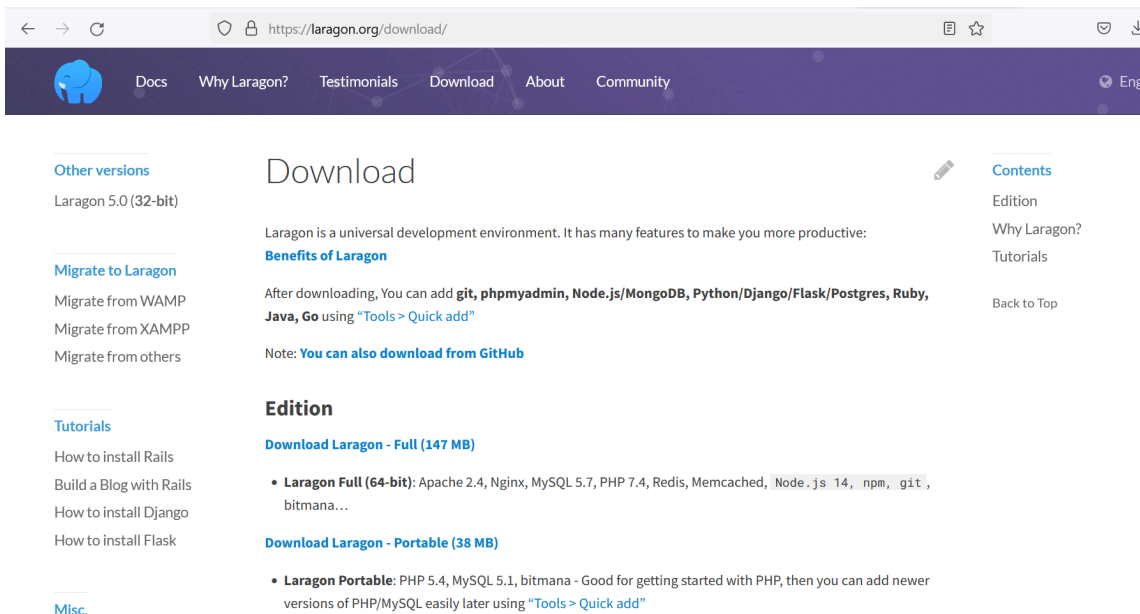


Figura 9: Website PHP

- Instalar o Composer;

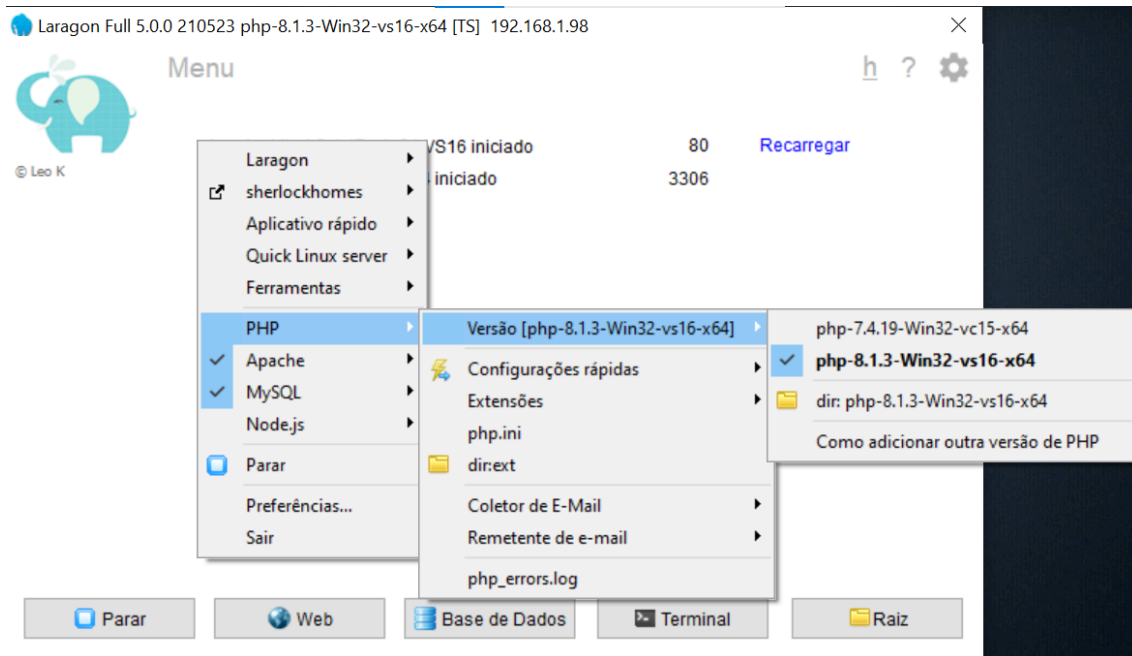


- Instalar o Laragon;



- Depois de instalar o Laragon, devemos iniciar o software e seleccionar a versão PHP instalada, para tal podemos clicar com o botão direito do mouse na janela do software, e seleccionamos pela mesma ordem as seguintes opções:

PHP > Versão > (versão instalada).



Após termos tudo instalado, podemos abrir o Visual Studio Code, e escrevemos a seguinte linha de comando no terminal para criar um novo projecto:

“ **composer create-project laravel/laravel nome\_do\_projecto-app** ”

```
PS C:\Users\aluno\Desktop> composer create-project laravel/laravel example-app
Creating a "laravel/laravel" project at "./example-app"
Installing laravel/laravel (v9.1.10)
- Downloading laravel/laravel (v9.1.10)
- Installing laravel/laravel (v9.1.10): Extracting archive
Created project in C:\Users\aluno\Desktop\example-app
> @php -r "file_exists('.env') || copy('.env.example', '.env');"
Loading composer repositories with package information
Updating dependencies
Lock file operations: 108 installs, 0 updates, 0 removals
- Locking brick/math (0.9.3)
- Locking dflydev/dot-access-data (v3.0.1)
- Locking doctrine/inflector (2.0.4)
- Locking doctrine/instantiator (1.4.1)
```

Com o projeto criado, podemos abrir a pasta do mesmo no Visual Studio Code para começar a desenvolver o código. Então comecei por criar ficheiros *blade* onde eu integrei o HTML do *template* escolhido por mim, isso serão as nossas *views*, ou seja, as páginas do meu website que o utilizador irá ver no seu ecrã.

```

1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5 <title>Sherlock Homes</title>
6 <meta charset="utf-8">
7 <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
8
9 <link href="https://fonts.googleapis.com/css?family=Ubuntu+Sans:200,300,400,600,700,800,900&display=swap"
10 rel="stylesheet">
11
12 <!-- <link rel="stylesheet" href="css/open-iconic-bootstrap.min.css" -->
13 <link rel="stylesheet" href="css/open-iconic-bootstrap.min.css">
14 <link rel="stylesheet" href="css/animate.css">
15
16 <link rel="stylesheet" href="css/owl.carousel.min.css">
17 <link rel="stylesheet" href="css/owl.theme.default.min.css">
18 <link rel="stylesheet" href="css/magnific-popup.css">
19
20 <link rel="stylesheet" href="css/aos.css">
21
22 <link rel="stylesheet" href="css/ionicons.min.css">
23
24 <link rel="stylesheet" href="css/bootstrap-datepicker.css">
25 <link rel="stylesheet" href="css/jquery.timepicker.css">
26
27
28 <link rel="stylesheet" href="css/flaticon.css">
29 <link rel="stylesheet" href="css/icomoon.css">
30 <link rel="stylesheet" href="css/style.css">
31
32 <link rel="shortcut icon" href="images/icon54.png">
33
34 <!-- <link rel="stylesheet" href="css/searchbar-items.css" -->
35 <link rel="stylesheet" href="css/searchbar-items.css">
36 <script src="js/searchbar-items.js"></script>
37 <script src="js/searchbar-items.js"></script>
38 <script src="js/selectImages.js"></script>
39
40 </head>
41
42 <body>
43
44 <nav class="navbar navbar-expand-lg navbar-dark ftco_navbar bg-dark ftco-navbar-light" id="ftco-navbar">
45 <div class="container">
46 <a class="navbar-brand" href="#">Sherlock Homes</a>

```

De seguida, criamos as rotas que são essenciais para o nosso *website*, pois é através delas que o nosso projeto saberá que páginas e informações deverá apresentar.

```

1 <?php
2 use Illuminate\Support\Facades\Route;
3
4 Route::get('/', [App\Http\Controllers\Main::class, 'index']);
5 Auth::routes();
6 Route::get('/about', function () {
7     return view('about');
8 });
9 Route::get('/contact', function () {
10     return view('contact');
11 });
12 Route::get('/favorites', [App\Http\Controllers\PropertyController::class, 'favorites']);
13
14 Route::get('/prop', function () {
15     return view('single-property');
16 });
17 Route::get('/results', [App\Http\Controllers\Main::class, 'search']);
18 Route::group(['middleware' => 'auth'], function () {
19     Route::post('favorite/{prop}/add', [App\Http\Controllers\FavoriteController::class, 'add'])->name('prop.favorite');
20 });
21
22 // Rotas necessárias para a recuperação de password
23 Route::get('forget-password', [App\Http\Controllers\Auth\ForgotPasswordController::class, 'showForgotPasswordForm'])->name('forget.password.get');
24
25 Route::post('forget-password', [App\Http\Controllers\Auth\ForgotPasswordController::class, 'submitForgotPasswordForm'])->name('forget.password.post');
26
27 Route::get('reset-password/{token}', [App\Http\Controllers\Auth\ForgotPasswordController::class, 'showResetPasswordForm'])->name('reset.password.get');
28
29 Route::post('reset-password', [App\Http\Controllers\Auth\ForgotPasswordController::class, 'submitResetPasswordForm'])->name('reset.password.post');
30
31 // Rotas necessárias para a recuperação de password
32 Route::get('/dashboard', [App\Http\Controllers\HomeController::class, 'dashboard'])->name('dashboard');
33 Route::get('/properties', [App\Http\Controllers\PropertyController::class, 'index'])->name('properties');
34 Route::post('/properties', [App\Http\Controllers\PropertyController::class, 'store']);
35
36 Route::get('/users', [App\Http\Controllers\HomeController::class, 'users'])->name('users');
37 Route::put('/user/{user}/addRole', [App\Http\Controllers\HomeController::class, 'addAdmin']);
38 Route::put('/user/{user}/removeRole', [App\Http\Controllers\HomeController::class, 'removeAdmin']);
39 Route::delete('/user/{user}', [App\Http\Controllers\HomeController::class, 'deleteUser']);
40
41 Route::get('/prop/{prop}/create', [App\Http\Controllers\PropertyController::class, 'create'])->name('prop.create');
42
43 Route::get('/prop/{prop}/edit', [App\Http\Controllers\PropertyController::class, 'edit']);
44 Route::put('/prop/{prop}', [App\Http\Controllers\PropertyController::class, 'update']);
45
46 Route::delete('/prop/{prop}', [App\Http\Controllers\PropertyController::class, 'destroy']);

```

Para conseguir passar as informações para as nossas views, tive que criar vários controllers que selecionam os dados pretendidos na nossa base de dados e os enviam para as respectivas páginas.

```

13 use Illuminate\Http\Request;
14 use Illuminate\Support\Facades\Auth;
15 use Illuminate\Support\Facades\DB;
16 use Illuminate\Support\Facades\File;
17
18 class PropertieController extends Controller
19 {
20     /**
21      * Display a listing of the resource.
22      *
23      * @return \Illuminate\Http\Response
24      */
25     public function __construct()
26     {
27         $this->middleware('auth');
28     }
29     public function index()
30     {
31         if (Auth::user()->hasRole('admin') || Auth::user()->hasRole('superadmin')) {
32             $properties = Properties::orderBy('id', 'DESC')->get();
33             $fotos = Foto::all();
34
35             return view('admin.properties', compact('properties', 'fotos'));
36         } else {
37             return redirect(url()->previous());
38         }
39     }
40
41     public function favs()
42     {
43         if (Auth::user() != null) {
44             $favoritePropertiesIds = DB::table('properties_user')->where('user_id', Auth::user()->id)->get();
45             $idsFavs = [];
46             // dd($favoritePropertiesIds);
47             foreach ($favoritePropertiesIds as $f) {
48                 array_push($idsFavs, $f->properties_id);
49             }
50
51             $favProperties = Properties::whereIn('id', $idsFavs)->get();
52
53             return view('favs', compact('favProperties'));
54         }
55     }
56 }

```

É também necessário criar *migrations*, *model's* e *seeder's*. As *migrations* servirão para criar a estrutura da nossa base de dados, os *model's* para criar relações entre as tabelas da nossa base de dados e os *seeder's* para popular a mesma com alguns dados essenciais, como por exemplo, o primeiro administrador, para conseguir aceder ao *BackOffice* e adicionar outros administradores.

```

13 use Illuminate\Database\Migrations\Migration;
14 use Illuminate\Database\Schema\Blueprint;
15
16 class 2023_03_31_131609_create_properties_table.php extends Migration
17 {
18     /**
19      * Run the migrations.
20      */
21     public function up()
22     {
23         Schema::create('properties', function (Blueprint $table) {
24             $table->id();
25             $table->string('name');
26             $table->integer('price');
27             $table->unsignedBigInteger('typeprice_id');
28             $table->unsignedBigInteger('location_id');
29             $table->unsignedBigInteger('typepropertie_id');
30             $table->unsignedBigInteger('typology_id');
31             $table->integer('bathroom');
32             $table->integer('gross_area');
33             $table->integer('usefull_area');
34             $table->unsignedBigInteger('propertywebsite_id');
35             $table->string('url');
36             $table->timestamps();
37
38             $table->foreign('typeprice_id')
39                 ->references('id')
40                 ->on('type_prices')
41                 ->onDelete('cascade');
42
43             $table->foreign('typepropertie_id')
44                 ->references('id')
45                 ->on('type_properties')
46                 ->onDelete('cascade');
47
48             $table->foreign('typology_id')
49                 ->references('id')
50                 ->on('typology_properties')
51                 ->onDelete('cascade');
52
53             $table->foreign('propertywebsite_id')
54                 ->references('id')
55                 ->on('property_websites')
56                 ->onDelete('cascade');
57
58             $table->foreign('location_id')
59                 ->references('id')
60                 ->on('locations')
61                 ->onDelete('cascade');
62         });
63     }
64 }

```

```

SHERLOCKHOMES
├── app
├── bootstrap
├── config
├── database
│   ├── factories
│   ├── migrations
│   └── seeders
│       ├── DatabaseSeeder.php
│       └── LaratrustSeeder.php
├── .gitignore
├── lang
├── public
├── resources
├── routes
├── storage
├── tests
├── vendor
├── .editorconfig
├── .env
├── .env.example
├── .gitattributes
├── .gitignore
├── .styleci.yml
├── artisan
├── Back
├── composer.json
├── composer.lock
├── package-lock.json
├── package.json
├── phpunit.xml
├── README.md
└── webpack.mix.js

database > seeders > LaratrustSeeder.php > PHP Intelephense > LaratrustSeeder > run

1  <?php
2
3  namespace Database\Seeders;
4
5  use Illuminate\Database\Console\Seeds\WithoutModelEvents;
6  use Illuminate\Database\Seeder;
7  use App\Models\Role;
8  use App\Models\User;
9  use Illuminate\Support\Facades\DB;
10 use Illuminate\Support\Facades\Hash;
11 use Illuminate\Support\Facades\Config\Laratrust;
12
13 class LaratrustSeeder extends Seeder
14 {
15     /**
16      * Run the database seeds.
17      *
18      * @return void
19      */
20     public function run()
21     {
22         $admin = Role::create(['name' => 'admin']);
23         $superadmin = Role::create(['name' => 'superadmin']);
24
25         $user = User::create([
26             'email' => 'superadmin@gmail.com',
27             'name' => 'Super Admin',
28             'password' => Hash::make('Daniel+333')
29         ]);
30
31         $user->attachRole('superadmin');
32         $user->attachRole('admin');
33
34     }
35 }
36
37

```

```

EXPLORER
├── SHERLOCKHOMES
│   ├── app
│   │   ├── Console
│   │   ├── Exceptions
│   │   └── Http
│   ├── Models
│   │   ├── Foto.php
│   │   ├── locations.php
│   │   ├── Permission.php
│   │   ├── Propertie.php
│   │   ├── Properties.php
│   │   ├── PropertyWebsite.php
│   │   ├── Role.php
│   │   ├── TypePrice.php
│   │   ├── TypologyProperty.php
│   │   └── User.php
│   ├── Providers
│   ├── bootstrap
│   ├── config
│   ├── database
│   ├── lang
│   ├── public
│   ├── resources
│   ├── routes
│   ├── storage
│   ├── tests
│   ├── vendor
│   ├── .editorconfig
│   ├── .env
│   ├── .env.example
│   ├── .gitattributes
│   ├── .gitignore
│   ├── .styleci.yml
│   ├── artisan
│   ├── Back
│   ├── composer.json
│   ├── composer.lock
│   ├── package-lock.json
│   ├── package.json
│   ├── phpunit.xml
│   ├── README.md
│   └── webpack.mix.js
└── SHERLOCKHOMES

app > Models > Properties.php > PHP Intelephense > Properties > favorite_to_user

1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Model;
7  use Illuminate\Database\Eloquent\Relations\BelongsTo;
8
9  class Properties extends Model
10 {
11     use HasFactory;
12
13     public function typeprice(){
14         return $this->belongsTo(TypePrice::class, 'typeprice_id');
15     }
16
17     public function typeproperty(){
18         return $this->belongsTo(TypeProperty::class, 'typeproperty_id');
19     }
20     // return $this->belongsTo(TypologyProperty::class, 'typology_id');
21
22     public function typologyproperty(){
23         return $this->belongsTo(TypologyProperty::class, 'typeprice_id');
24     }
25
26     public function propertywebsite(){
27         return $this->belongsTo(PropertyWebsite::class, 'propertywebsite_id');
28     }
29
30     public function locations(){
31         return $this->belongsTo(locations::class, 'location_id');
32     }
33
34     public function foto(){
35         return $this->hasMany(Foto::class);
36     }
37
38     public function favorite_to_user(){
39         return $this->belongsToMany(User::class)->withTimestamps();
40     }
41
42 }

```

Para fazer as pesquisas, criei no controller a função "search", para poder criar as *queries* responsáveis por fazer os pedidos à base de dados que por sua vez retornará os resultados dentro dos parâmetros definidos pelo cliente.

```
13 class Main extends Controller
14 {
15     public function search(Request $request){
16
17
18         $PriceType = (request('selectPriceType') == 'Todos') ? '' : request('selectPriceType');
19         $Location = (request('selectDistrict') == 'Todos') ? '' : request('selectDistrict');
20         $PropertyType = (request('selectPropertieType') == 'Todos') ? '' : request('selectPropertieType');
21         $Typology = (request('selectTypology') == 'Todos') ? '' : request('selectTypology');
22         $search_text = request('query');
23
24         $results = Properties::query()->where('name', 'LIKE', '%'.$search_text.'%')
25             ->where('typeprice_id', 'LIKE', '%'.$PriceType.'%')
26             ->where('location_id', 'LIKE', '%'.$Location.'%')
27             ->where('typepropertie_id', 'LIKE', '%'.$PropertyType.'%')
28             ->where('typology_id', 'LIKE', '%'.$Typology.'%')
29             ->inRandomOrder()
30             ->get();
31
32         return view('results', compact('results'));
33     }
34 }
```

A instalação do *Laratrust* foi fundamental para criar um sistema de controlo de acesso ao BackOffice, pois permite-nos criar cargos e atribuí-los a outras contas para que passem a ser administradores do site, e assim, somente quem tenha um cargo de administrador conseguirá aceder ao backoffice.

Para fazer a instalação do *laratrust* é necessário executar os seguintes comandos no terminal:

- Comando para instalar o pacote utilizando o composer:

```
composer require santigarcor/laratrust
```

- Comando para publicar o arquivo de configuração:

```
php artisan vendor:publish --tag="laratrust"
```

- Comando para criar as *migrations* e *models* necessários:

```
php artisan laratrust:setup
```

- Comando para fazer o dump:

```
composer dump-autoload
```



- E para finalizar, o comando para fazer a *migration* que irá atualizar a nossa Base de Dados:

**php artisan migrate**

De seguida, basta criar as *roles* no ficheiro “LaratrusterSeeder.php”, toda a vez que usarem as seeder para popular a base de dados, as *roles* serão guardadas e passaram a ser funcionais.

```

1  <?php
2
3  namespace Database\Seeders;
4
5  use Illuminate\Database\Console\Seeds\WithoutModelEvents;
6  use Illuminate\Database\Seeder;
7  use App\Models\Role;
8  use App\Models\User;
9  use Illuminate\Support\Facades\DB;
10 use Illuminate\Support\Facades\Hash;
11 use Illuminate\Support\Facades\Config\Laratruster;
12
13 class LaratrusterSeeder extends Seeder
14 {
15     /**
16      * Run the database seeds.
17      *
18      * @return void
19      */
20     public function run()
21     {
22         $admin = Role::create(['name' => 'admin']);
23         $superadmin = Role::create(['name' => 'superadmin']);
24
25         $user = User::create([
26             'email' => 'superadmin@gmail.com',
27             'name' => 'Super Admin',
28             'password' => Hash::make('Daniel+333')
29         ]);
30
31         $user->attachRole('superadmin');
32         $user->attachRole('admin');
33     }
34 }
35

```

Figura - Criação de *Roles* no seeder e atribuição das mesmas à conta *master* do nosso website

```

30 public function dashboard(){
31     $numProperties = count(Properties::all());
32     $numUsers = count(User::all());
33     $recentProperties = Properties::orderBy('created_at', 'desc')->take(2)->get();
34
35     if (Auth::user()->hasRole('admin') || Auth::user()->hasRole('superadmin')) {
36         return view('admin.dashboard', compact('numProperties', 'numUsers', 'recentProperties'));
37     }else{
38         return redirect(url()->previous());
39     }
40 }

```

Figura - Exemplo de uso do *Laratruster*

Para criar as listas de favoritos de cada cliente e a possibilidade de remover ou adicionar imóveis a essa lista, eu tive que criar uma tabela na base de dados para registar todos os imóveis que estão na lista de cada um dos clientes.

```

2019_12_14_000001_create_personal_access_tokens_table.php
2022_04_26_183124_create_type_properties_table.php
2022_04_26_183206_create_typology_properties_table.php
2022_04_26_183319_create_property_websites_table.php
2022_04_26_183711_create_type_prices_table.php
2022_04_29_071455_create_locations_table.php
2022_05_12_223543_larastest_setup_tables.php
2023_03_31_131609_create_properties_table.php
2023_05_29_171512_create_prop_user_table.php
2024_04_26_183008_create_fotos_table.php
2024_04_29_233003_update_fotos.php
seeders
DatabaseSeeder.php
LarastestSeeder.php
.gitignore

12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29

return void
*/
public function up()
{
    Schema::create('properties_user', function (Blueprint $table) {
        $table->id();
        $table->string('properties_id');
        $table->bigInteger('user_id');
        $table->timestamps();

        $table->foreign('properties_id')
            ->references('id')
            ->on('properties')
            ->onDelete('cascade');
    });
}

```

Figura - Migration que cria a tabela das listas de favoritos de cada cliente

Foi também necessário criar no model uma relação entre imóveis e utilizadores.

```

40
41
42
43
44

public function favorite_to_user(){
    return $this->belongsToMany(User::class)->withTimestamps();
}

```

De seguida, criou-se a função que adiciona o imóvel à lista de favoritos do cliente. A mesma irá remover da lista caso o imóvel já esteja lá registado.

```

6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25

use Illuminate\Support\Facades\Auth;

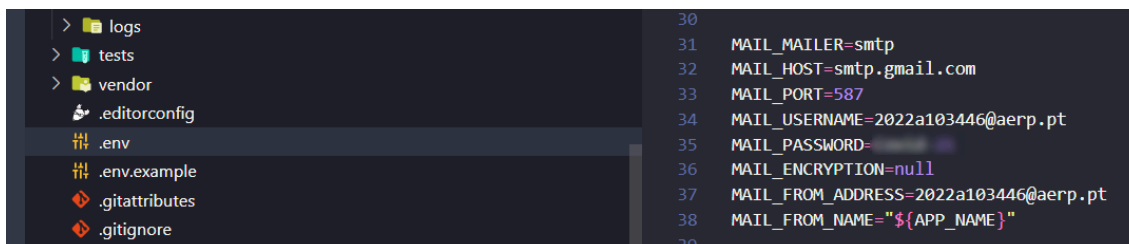
class FavoriteController extends Controller
{
    public function add($prop){
        $user = Auth::user();
        $isFavorite = $user->favorite_props()->where('properties_id', $prop)->count();

        if($isFavorite == 0){
            $user->favorite_props()->attach($prop);
            return redirect(url()->previous());
        }else{
            $user->favorite_props()->detach($prop);
            return redirect(url()->previous());
        }
    }
}

```

Figura - Função que adiciona e remove imóveis da lista de favoritos

Com o objectivo de criar a opção de recuperação da password da conta do cliente, editei o ficheiro `.env` de modo a conseguir enviar emails de recuperação, onde o cliente irá clicar num botão que o levará para uma página específica, gerada apenas para estes tipos de situações.



```
> logs
> tests
> vendor
.editorconfig
.env
.env.example
.gitattributes
.gitignore
30
31 MAIL_MAILER=smtp
32 MAIL_HOST=smtp.gmail.com
33 MAIL_PORT=587
34 MAIL_USERNAME=2022a103446@aerp.pt
35 MAIL_PASSWORD=
36 MAIL_ENCRYPTION=null
37 MAIL_FROM_ADDRESS=2022a103446@aerp.pt
38 MAIL_FROM_NAME="${APP_NAME}"
39
```

## WebScraping

O *webscraper* é a forma mais eficaz de “minerar” informação em páginas web, foi sem dúvida fundamental para manter a minha base de dados atualizada de forma a disponibilizar informações mais precisas para os meus clientes.

Para desenvolver o scraping eu decidi usar a linguagem de programação Python, porque é uma linguagem que eu tenho estado a aprender e que me tem conquistado devido à sua versatilidade.

O primeiro passo é importar todas as bibliotecas necessárias para executar diversas tarefas, como por exemplo a “*BeautifulSoup*” usada para manipular o HTML das páginas web e o “*mysql.connector*” usado para fazer a conexão com a base de dados.

```
C: > Users > aluno > Desktop > SherlockHomes > Scraper_CS.py > ...
1 from bs4 import BeautifulSoup
2 import requests
3
4 import urllib.request
5
6 import time
7 import datetime
8 from datetime import date, datetime
9 from datetime import date
10
11 import colorama
12 from colorama import Fore, Back, Style
13 colorama.init(autoreset=True)
14
15 import os
16 from tomlkit import integer
17
18 import mysql.connector
19
```

Figura - Bibliotecas usadas neste projeto

O trabalho do meu *webscraper* está dividido em 3 etapas:

- Na primeira etapa, o programa verifica a conexão com a base de dados e depois começa a fazer a recolha de todos os links de imóveis nas páginas definidas por mim.
- Na segunda etapa, ele percorre todos os links recolhidos anteriormente e vai de página em página fazendo a seleção de todas as informações necessárias, guardando-as numa lista de objectos.
- Na última etapa, o programa tenta inserir na base de dados cada um dos imóveis guardados na lista de objectos, e no caso de esse imóvel já existir na base de dados, o programa limita-se a atualizar o preço.

```

20
21 mydb = mysql.connector.connect(
22     host="localhost",
23     user="root",
24     password="",
25     database="sherlockhomes"
26 )
27 mycursor = mydb.cursor()
28
29 print(Fore.GREEN+str(mydb))
30

```

Figura - Criação da conexão com a base de dados

```

107
108 #?o seguinte código percorre todas as queries e guarda os links dos imóveis que serão alvos de scraping
109 for query in ListaQueries_CasaSapo:
110     site = requests.get(query, headers=headers)
111     soup = BeautifulSoup(site.content, 'html.parser')
112     Imoveis_ALL_Cards = soup.find_all('div', class_='property')
113     link_for_braker = 0
114     for imoveis in Imoveis_ALL_Cards: #? Este for guarda todos os links das casas da página na lista "Lista_Links"
115         if link_for_braker == 4:
116             break
117         Imovel_Codigo_Link = imoveis.find('a')
118         link = Imovel_Codigo_Link.get('href')
119         if "comprar" in link and "alugar" in link: #? Está a funcionar
120             print(Fore.RED+"IGNORED LINK")
121         else:
122             if link[0] == "/":
123                 newlink = "https://casa.sapo.pt/" + link
124                 Lista_Links.append(newlink)
125                 num_links += 1
126                 # print(Fore.CYAN+"\n" + newlink)
127             else:
128                 Lista_Links.append(link)
129                 num_links += 1
130                 # print(Fore.CYAN+"\n" + link)
131             link_for_braker +=1
132     time.sleep(5)
133     os.system('cls')
134     for link in Lista_Links:
135         print(Fore.BLUE + str(link))
136     print(Fore.YELLOW + "RECOLHA DE LINKS EM PROCESSO")
137     print("Links encontrados: " +Fore.BLUE+str(num_links))
138

```

```

139 for link in Lista_Links:
140     try:
141         if "id=" in link:
142             site = requests.get(link, headers=headers)
143             soup = BeautifulSoup(site.content, 'html.parser')
144
145             splited_link = link.split('id=')
146             id = "CS"+splited_link[len(splited_link)-1]
147             # print("ID: "+ id)
148             website = 1
149
150 >         if "t0" in link: ...
152 >         else: ...
170 >         if "comprar" in link: ...
172 >         else: ...
174 >         if "apartamento" in link: ...
176 >         else: ...
181
182         Titulo_card = soup.find('div', class_='detail-section detail-title')
183         titulo = str(Titulo_card.find('h1').get_text().replace('<h1>', ''))
184         preco = int(soup.find('div', class_='detail-title-price-value').get_text().replace('€', '').replace('.', ''))
185
186         Location_card = soup.find('div', class_='detail-title-location').get_text()
187         splited_location_card = Location_card.split(',')
188         location = splited_location_card[len(splited_location_card)-1]
189         if "Aveiro" in location:
190             location = 1
191 >         else: ...
241
242
243         Caracteristicas_Imoveis = soup.find_all('div', class_='detail-features-item')
244         Dados_Imoveis = soup.find_all('div', class_='detail-main-features-item')
245         Area_Bruta = ""

```

```

243         Caracteristicas_Imoveis = soup.find_all('div', class_='detail-features-item')
244         Dados_Imoveis = soup.find_all('div', class_='detail-main-features-item')
245         Area_Bruta = ""
246         Area_Util = ""
247         N_wc = ""
248
249         for item in Dados_Imoveis:
250             Detail_title = item.find('div', class_="detail-main-features-item-title")
251             # print("Detalhe: "+Detail_title.get_text())
252             if "Área bruta" in Detail_title.get_text():
253                 Area_Bruta = item.find('div', class_='detail-main-features-item-value').get_text().replace('m²', '')
254                 break
255             for item in Dados_Imoveis:
256                 Detail_title = item.find('div', class_="detail-main-features-item-title")
257                 # print("Detalhe: "+Detail_title.get_text())
258                 if "Área útil" in Detail_title.get_text():
259                     Area_Util = item.find('div', class_='detail-main-features-item-value').get_text().replace('m²', '')
260                     break
261             for item in Caracteristicas_Imoveis:
262                 # print(item.get_text())
263                 if "Casa(s) de Banho:" in item.get_text():
264                     N_wc = item.get_text().replace('Casa(s) de Banho: ', '')
265                     break
266
267             if Area_Bruta == "":
268                 Area_Bruta = "0"
269             if Area_Util == "":
270                 Area_Util = "0"
271             if N_wc == "":
272                 N_wc = "0"
273

```

```

273         Descricao = ""
274
275         try:
276             Descricao = soup.find('div', class_='detail-section detail-description').get_text().replace(' ', '')
277             Descricao = Descricao.replace(Descricao[0], '')
278         except:
279             Descricao = ""
280
281         main = soup.find('main')
282         div_fotos = main.find('div', 'detail-media-imgs')
283         Links_Fotos = div_fotos.find_all('img')
284         Foto_Link = Links_Fotos[0].get('src').replace('.jpg.webp', '.jpg')
285

```

```

285
286         Lista_Objects.append( Obj_Imovel(titulo, preco, tipo_preco, location,
287                                         tipo_imovel, tipologia, N_wc, Area_Bruta,
288                                         Area_Util, website, Descricao, link, id, Foto_Link) )
289     else:
290         continue
291 except:
292     continue

```

```

297 for imovel in Lista_Objects:
298
299     TITULO = str(imovel.titulo)
300     PRECO = int(imovel.preco)
301     TIPO_PRECO = int(imovel.tipo_preco)
302     LOCALIZACAO = int(imovel.localizacao)
303     TIPO_IMOVEL = int(imovel.tipo_imovel)
304     TIPOLOGIA = int(imovel.tipologia)
305     N_WC = int(imovel.n_wc)
306     A_BRUTA = int(imovel.area_bruta)
307     A_UTIL = int(imovel.area_util)
308     WEBSITE = str(imovel.website)
309     DESCRICAO = str(imovel.descricao)
310     URL = str(imovel.url)
311     ID = str(imovel.id)
312     FOTO = str(imovel.foto)
313
314     try:
315         mycursor.execute("INSERT INTO properties (id, name, price, typeprice_id, location_id, typepropertie_id, typology_id, bathrooms, "+
316                         "gross_area, usefull_area, propertywebsite_id, descricao, url, created_at)" +
317                         "VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)",
318                         (ID, TITULO, PRECO, TIPO_PRECO, LOCALIZACAO, TIPO_IMOVEL, TIPOLOGIA, N_WC, A_BRUTA, A_UTIL, WEBSITE, DESCRICAO, URL,
319                         datetime.now().strftime("%Y-%m-%d %H:%M:%S")))
320         mydb.commit()
321         print("Imóvel id: "+imovel.id+" adicionado a base de dados!")

```

```

323
324     current_folder = os.path.dirname(os.path.abspath(__file__))
325     try:
326         os.makedirs(current_folder+"/sherlockhomes/public/uploads/"+ID)
327     except FileExistsError:
328         pass
329
330     imgURL = FOTO
331     urllib.request.urlretrieve(imgURL, current_folder+"/sherlockhomes/public/uploads/"+ID+"/"+ID+".jpg")

```

```

347     try:
348         mycursor.execute("UPDATE properties SET price = %s, updated_at = %s WHERE (id = %s)", (PRECO, datetime.now().strftime("%Y-%m-%d %H:%M:%S"), ID))
349         mydb.commit()
350         print("Imovel Atualizado")
351         atualizados_imoves += 1
352     except:
353         print("!!!ERRO AO ATUALIZAR IMOVEL NA BASE DE DADOS!!!\nLink do Imovel: "+imovel.url)
354

```

## 2.3. Estratégias adotadas na resolução de problemas

### 3. Conclusão

### 4. Referências

StackOverflow;  
github  
Laracasts  
ThemeWagon

### 5. Anexos