Relatório do Projeto

Monumentos da Diocese de Beja



Nome: Diogo Alexandre Beco Silva

Numero: 20307

Tema do projeto: Monumentos da Diocese de Beja

Ano Letivo: 2020/21

Relatório do Projeto de Fim de Curso apresentado na Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Beja

Orientadores de Estágio:

Carlos Sancho (professor de Programação para Dispositivos Móveis)

Luís Rosário (professor de Programação para a Web)

Relatório do Projeto de fim de curso apresentado na Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Beja

Índice

[Índice de Imagens 2](#_Toc82974317)

[Introdução 3](#_Toc82974318)

[Abstract 3](#_Toc82974319)

[Faseamento do Projeto 4](#_Toc82974320)

[Desenho da interface da aplicação 5](#_Toc82974321)

[Desenho da Base de Dados 11](#_Toc82974322)

[Projeto Laravel 12](#_Toc82974323)

[Aplicação Android 14](#_Toc82974324)

[Dificuldades Sentidas ao longo do projeto 17](#_Toc82974325)

[Conclusão 18](#_Toc82974326)

[Glossário 19](#_Toc82974327)

[Bibliografia 20](#_Toc82974328)

## 

## Índice de Imagens

[Figura 1 – Organização dos prazos de cada tarefa e lista de funcionalidades a ser desenvolvidas 5](#_Toc83074028)

[Figura 2 – Desenho dos ecrãs relativos à conta de utilizador. 6](#_Toc83074029)

[Figura 3 – desenho do ecrã responsável por exibir a lista de monumentos. 7](#_Toc83074030)

[Figura 4 – desenho do ecrã de detalhes de um monumento específico. 8](#_Toc83074031)

[Figura 5 – desenho do ecrã de detalhes de um ponto de interesse específico. 9](#_Toc83074032)

[Figura 6 – desenho da capa da aplicação web. 10](#_Toc83074033)

[Figura 7 – desenho da apresentação da informação de monumentos em cada linha da lista. 10](#_Toc83074034)

[Figura 8 – desenho da apresentação da informação de cada monumento. 11](#_Toc83074035)

[Figura 9 – desenho da apresentação do slideshow, responsável por fornecer ao utilizador os pontos de interesse do monumento a ser apresentado. 11](#_Toc83074036)

[Figura 10 – Desenho da apresentação de informação do ponto de interesse selecionado 11](#_Toc83074037)

[Figura 11 – Estrutura inicial da base de dados 12](#_Toc83074038)

[Figura 12 – Rotas criadas para a navegação da app Laravel 14](#_Toc83074039)

[Figura 13 – Exemplo de um dos Controllers criados 15](#_Toc83074040)

[Figura 14 – Código do Datasource 19](#_Toc83074041)

[Figura 15 – Código do AppService 19](#_Toc83074042)

## Introdução

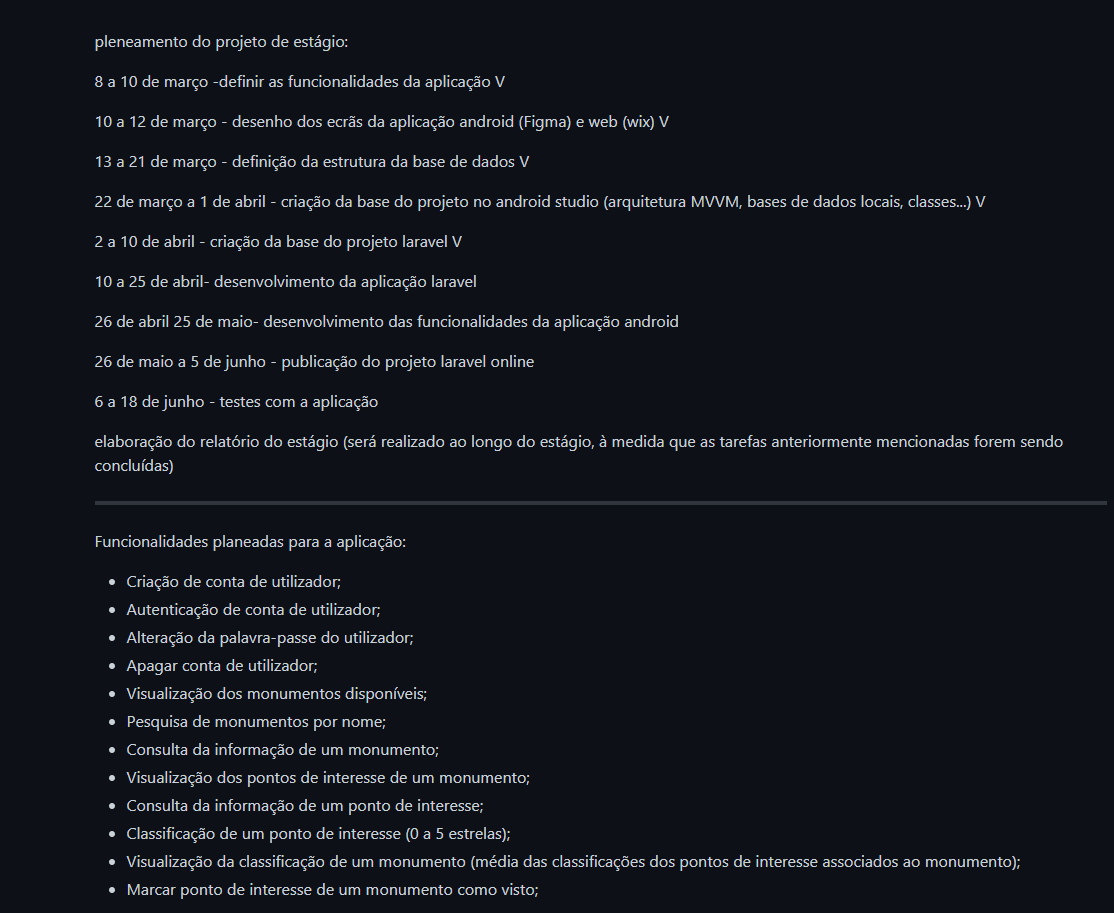
Este relatório está relacionado ao projeto de fim do curso “Tecnologias Web e Dispositivos Móveis”, o qual foi realizado na modalidade de projeto devido à pandemia. Neste projeto de fim de curso, pretende-se que o aluno aprimore os seus conhecimentos com as tecnologias utilizadas no curso, relativamente ao “*Android* Studio” e ao “*Laravel*”. Será entregue ao Aluno o tema “*Monumentos* de Beja”, o qual pretende que seja criada uma *aplicação* *android* que possibilite que o utilizador seja guiado pelos vários *pontos de interesse* dos *monumentos* de Beja e forneça informação sobre os mesmos.

## Abstract

This report is related to the end-of-course “Tecnologias Web e Dispositivos Móveis” project, which was accomplished in the modality of project due to the pandemic. In this end-of-course project, it is intended that the student improves their knowledge with the technologies used in the course, in relation to “Android Studio” and “Laravel”. The Student will be given the theme “Monumentos de Beja”, which intends to create an android application that allows the user to be guided by the various points of interest of the monuments of Beja and provide information about them.

## Faseamento do Projeto

Para a realização do Projeto, foi pedido que se realizasse um faseamento, no qual o aluno deveria organizar as várias tarefas que seriam realizadas ao longo do semestre, assim como uma estimativa do tempo que cada uma dessas tarefas levaria e uma lista de funcionalidades pretendidas para a aplicação. Para a sua organização, acabei por ter como base o projeto realizado no semestre anterior, planeando desenvolver algumas funcionalidades que não tinha conseguido na altura, motivo pelo qual não incluirei uma secção de sistemas semelhantes.

Figura 1 – Organização dos prazos de cada tarefa e lista de funcionalidades a ser desenvolvidas

Para começar, criei uma API fake, através do “my JSON server”, que permitiria fazer um esboço da base de dados, facilitando o desenvolvimento da aplicação Android antes de conectá-la à aplicação Laravel.

## Desenho da interface da aplicação

Para o desenho da interface da aplicação android foi utilizado o figma e para o desenho da aplicação laravel foram utilizados templates disponível no wix.

Começando pelo Figma, criei os vários ecrãs que se pretendiam usar na aplicação android. Foram criados ecrãs para a autenticação do utilizador (que foi decidido que seria uma função exclusiva da aplicação android). Esta função permite que o utilizador registe uma conta de utilizador e faça login. Permite ainda que, quando o utilizador acessa o seu perfil, o mesmo possa alterar a palavra-passe, terminar sessão, ou apagar a conta de utilizador.

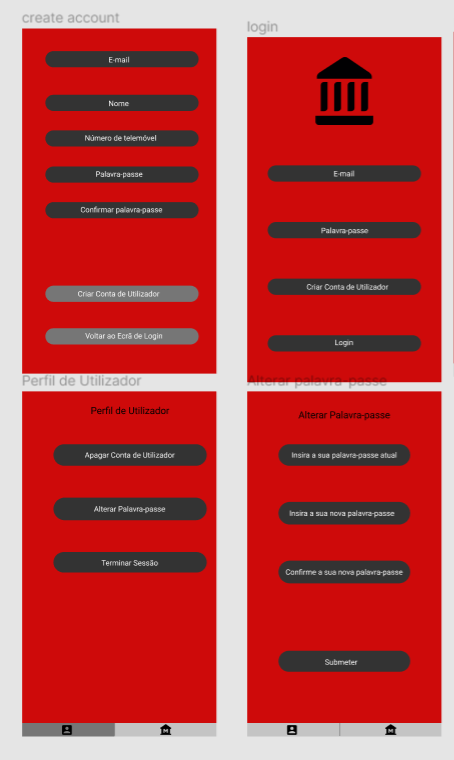


Figura 2 – Desenho dos ecrãs relativos à conta de utilizador.

Em seguida, criei os ecrãs necessários para a consulta de informação dos monumentos.

Primeiramente foi criado um ecrã que possui uma lista de monumentos, constituída por uma imagem do monumento e nome do monumento. Juntamente com esta lista, foi criada uma barra de pesquisa que permite procurar um monumento pelo nome (embora não tenha conseguido de facto implementar no design do Figma).

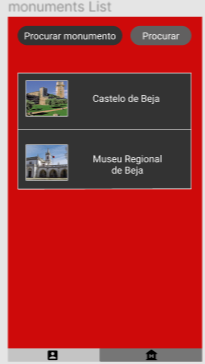


Figura 3 – desenho do ecrã responsável por exibir a lista de monumentos.

A seguir, criei mais um ecrã da aplicação, que representa a página de um monumento. Neste ecrã, são mostradas as várias informações de um monumento, sendo estas informações o nome do monumento, imagem, horário de funcionamento, dia de encerramento, localização do monumento, descrição do monumento, classificação do monumento (a qual será determinada através da média das avaliações dos pontos de interesse correspondentes ao monumento) e uma lista que contém os pontos de interesse de cada monumento, constituída por uma imagem do ponto de interesse, pelo seu nome e por um possível ícone, que representaria se o utilizador já viu aquele ponto de interesse específico ou não.

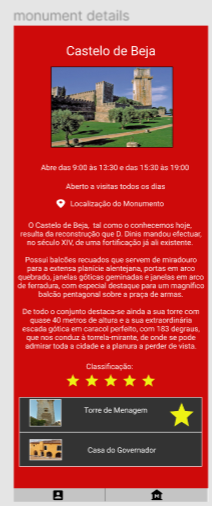


Figura 4 – desenho do ecrã de detalhes de um monumento específico.

Por fim, criei um último ecrã, responsável por apresentar ao utilizador os detalhes do ponto de interesse selecionado no ecrã anterior.

Neste ecrã, é possível ver o nome do ponto de interesse, foto, e descrição.

É possível ainda avaliar o ponto de interesse com entre 1 e 5 estrelas.

Por fim, criei ainda um botão que, ao ser clicado, irá selecionar/desseleccionar o ponto de interesse como visto, que vai adicionar/remover o ícone de estrela da linha do ponto de interesse na listView, no ecrã de detalhes do monumento.

Figura 5 – desenho do ecrã de detalhes de um ponto de interesse específico.

Relativamente ao protótipo do Website, realizado pela Wix, escolhi um template para a sua realização.

Comecei por criar a barra de navegação, que contem o menu home, o qual leva para a página inicial, e um menu de monumentos. Dentro do item de Monumentos existem ainda subitens, correspondentes aos monumentos contidos na base de dados.

Criei então as duas secções da página inicial, que consistem numa capa para o *website*, e uma secção onde são disponibilizados os vários monumentos presentes na base de dados, mostrando a foto do monumento, nome, descrição e uma opção “*learn more*”, que possui uma hiperligação para a página do monumento em questão.

Figura 6 – desenho da capa da aplicação web.

Figura 7 – desenho da apresentação da informação de monumentos em cada linha da lista.

Ao clicar na opção “Learn more” de determinado monumento, o utilizador será redirecionado para outra página, que lhe apresentará os detalhes do monumento selecionado, sendo esses detalhes o nome do monumento, imagem, horário de funcionamento, descrição e localização, tratando-se esta última de uma hiperligação para o link do Google Maps do monumento, o que permite que o utilizador confira onde o monumento se encontra.

Numa outra secção abaixo, criei um slideshow, que apresenta os vários pontos de interesse presentes no monumento a ser visualizado, apresentando em cada slide a foto do ponto de interesse, o nome e uma hiperligação que irá redirecionar o utilizador para a página do ponto de interesse.

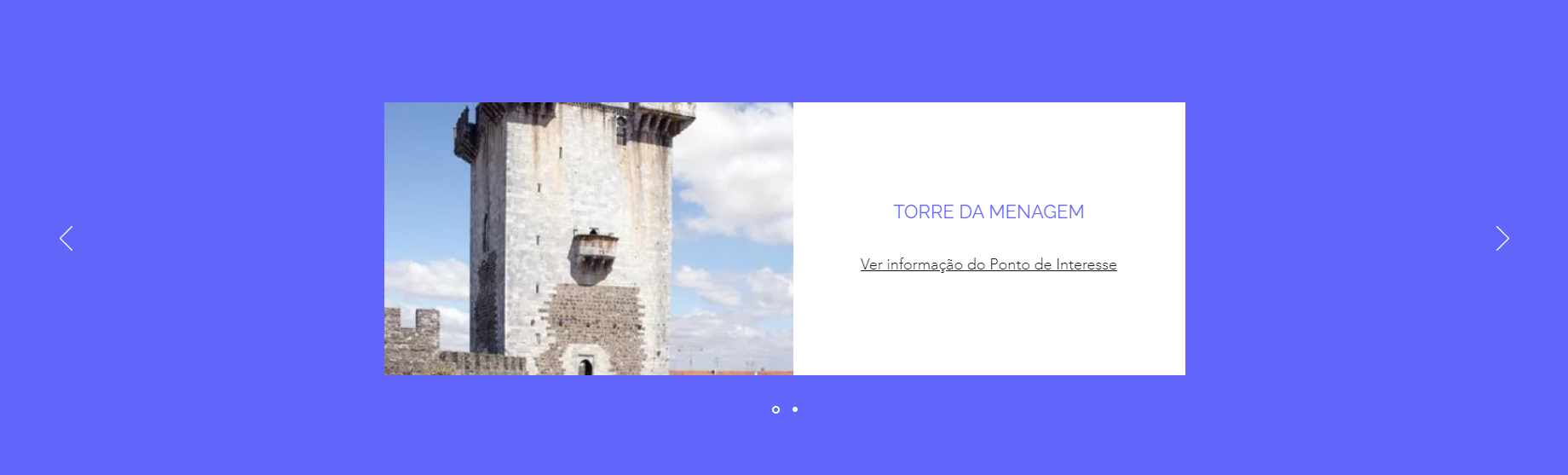
Figura 8 – desenho da apresentação da informação de cada monumento.

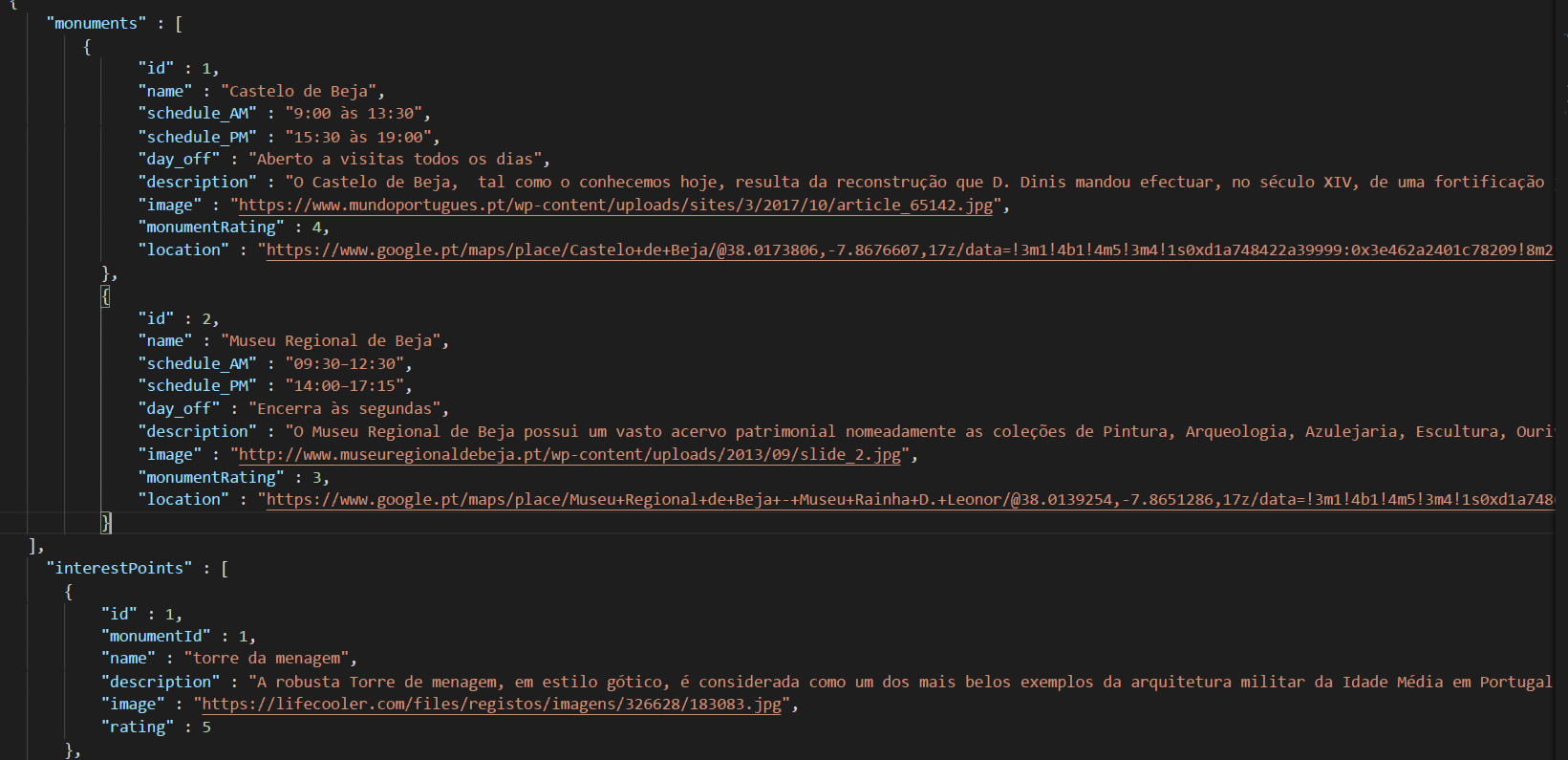
Figura 9 – desenho da apresentação do slideshow, responsável por fornecer ao utilizador os pontos de interesse do monumento a ser apresentado.

Por fim, criei uma página adicional para cada ponto de interesse, que possui a foto do ponto de interesse como fundo da página, com uma caixa de texto centrada. Dentro desta caixa de texto é apresentado o nome do ponto de interesse e a sua descrição, assim como um botão para que o utilizador consiga voltar para trás.

Figura 10 – Desenho da apresentação de informação do ponto de interesse selecionado

## Desenho da Estrutura da Base de Dados

Para que houvesse uma organização relativamente à criação da base de dados, criei inicialmente a estrutura da base de dados, no formato json, para que tivesse uma ideia inicial do que iria precisar para o projeto final.

Figura 11 – Estrutura inicial da base de dados

Depois da criação do github, carreguei o ficheiro para um repositório do github, que, por sua vez, permitiu que fosse possível usá-lo como uma API falsa, através do *my json server,* permitindo que a aplicação android pudesse ir buscar os dados e apresenta-los enquanto não fosse conectada ao projeto laravel.

Quando iniciei o projeto laravel, criei finalmente a base de dados no xampp, para que fosse possível criar as tabelas a partir das *migrations*.

## Projeto Laravel

Para criar o projeto, inseri, na linha de comandos, o comando “laravel new monuments”, que gerou a pasta do projeto denominada “monuments”, assim como os ficheiros necessários para o desenvolvimento da *app web*.

Em seguida, instalei o *jetstream*, que criou o código base necessário para a autenticação referente ao acesso do backend da aplicação, através dos comandos “composer require laravel/jetstream”, seguido dos comandos “php artisan jetstream:install livewire”, “npm install && npm run dev”, que é necessário para que se possibilite a utilização do back end.

Após a criação do projeto, criei a base de dados no xampp. Em seguida, foram criadas as migrações referentes às tabelas da base de dados, através do comando “php artisan make:migration (nome da tabela) ”, que cria os ficheiros php onde os atributos da tabela são definidos, seguido do comando “php artisan migrate”, para que as tabelas fossem criadas no xampp com base na informação presente nos ficheiros das migrations.

A seguir, criei os modelos necessários, que seriam os *models* de Monumentos e de Pontos de Interesse, através do comando “php artisan make:model (nome do *model*)”, onde foram definidos .

Criei ainda os *Controllers* para cada *model*, pelo comando “php artisan make:Controller (nome do *Controller*)”, para criar os *Controllers* necessários.

Em seguida criei as *Routes*, no ficheiro “web.php”. Aqui foram criadas as várias *routes* responsáveis pela navegação do *Website*, tanto para o *backend*, que requereria validação de conta, como para o utilizador. No ficheiro “api.php”, foi ainda criado o acesso à *api*, através dos *Controllers*, que permitiria que o *website* conseguisse ir buscar as informações presentes nas tabelas da base de dados.

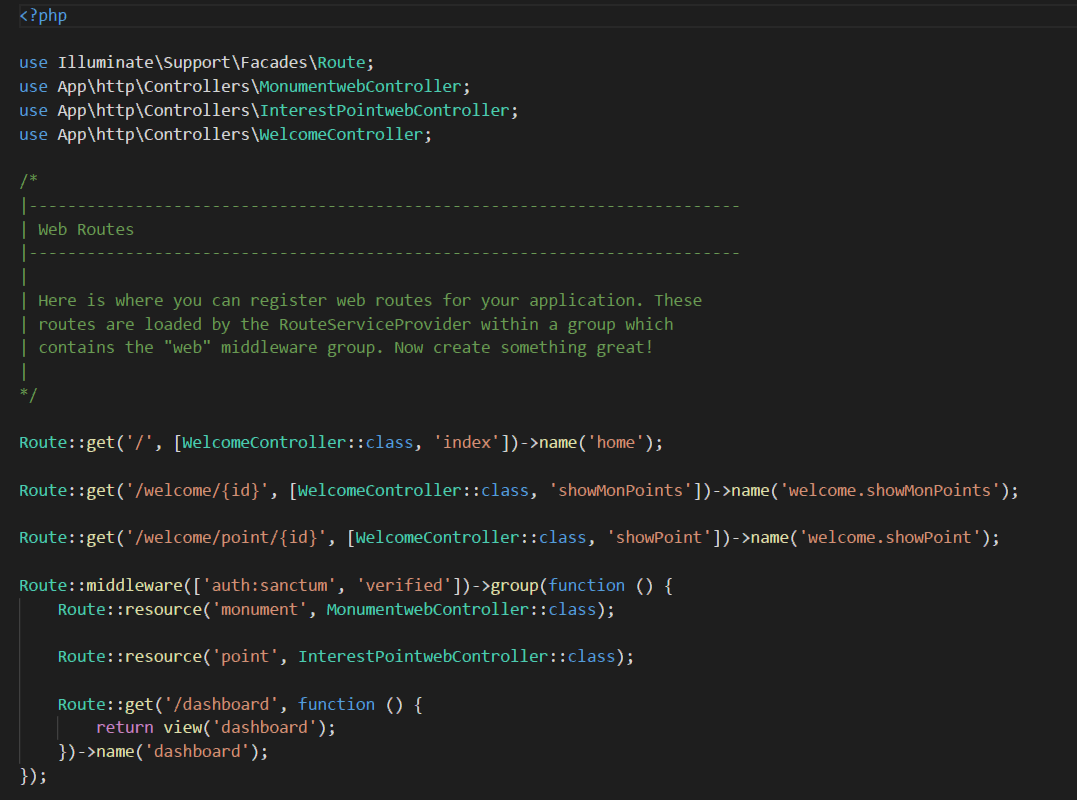


Figura 12 – Rotas criadas para a navegação da app Laravel

Cada rota funciona apenas se for fornecido o Controller necessário. Após fornecer a localização dos *Controllers* necessários, comecei a criar as rotas necessárias para a navegação. O middleware é incluído com a instalação do jetstream, que permite a implementação de verificação de utilizador de uma forma mais fácil, para a implementação do backend, portanto decidi utilizar as resource de monumento e de point criadas anteriormente para o fazer.

Após criadas as rotas do backend, comecei a criar as do frontend, que incluem uma rota de regresso à página inicial, uma rota que navega para a página de detalhes de um monumento e uma página que navega para a página de detalhes de um ponto de interesse.

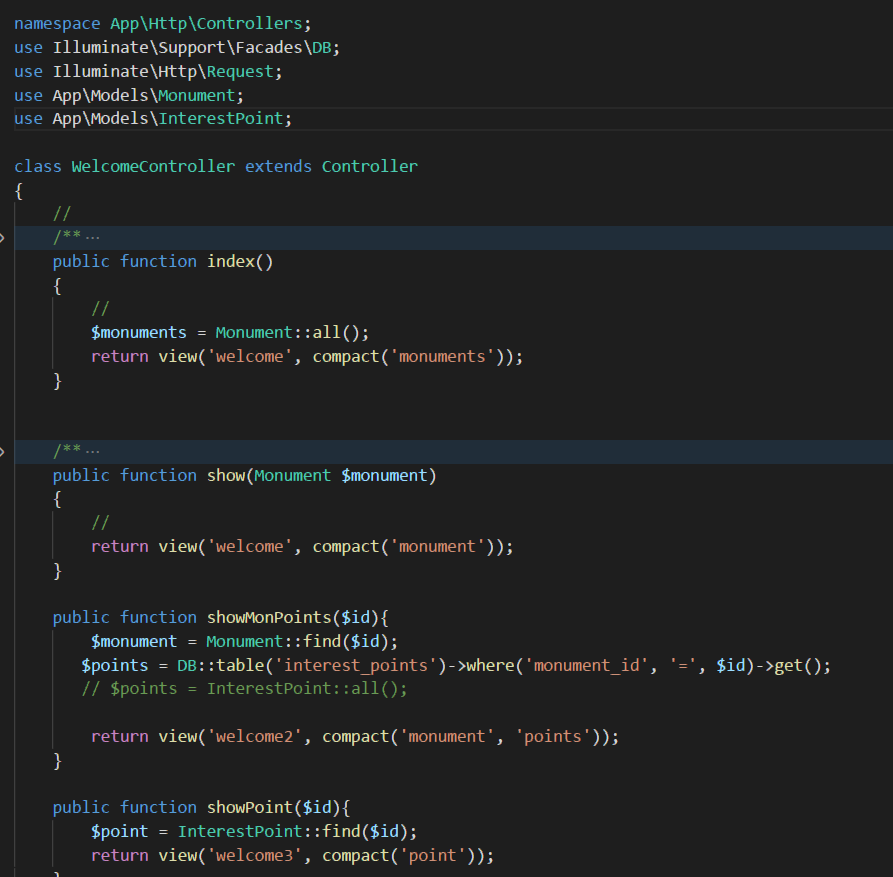


Figura 13 – Exemplo de um dos Controllers criados

Relativamente aos *WelcomeController*, responsável por apresentar a informação no front end, foi criado um método index que recebe os dados de todos os monumentos, o que permite apresentar todos numa lista através do método show. Criei ainda o método “showMonPoints”, que passa o id do monumento selecionado para a página de detalhes do monumento, o que permite ver as informações do mesmo, assim como um slide show que apresenta os vários pontos de interesse relacionados ao monumento selecionado. Por fim, criei um método “showPoint”, que passa o id do ponto de interesse selecionado para a página de detalhes do ponto de interesse, permitindo a apresentação da sua informação.

Por fim, criei os layouts necessários para o backend. Para tal, foram criadas duas pastas manualmente, sendo uma para o backend dos monumentos e outra para o backend dos pontos de interesse. Criei as *blades* necessárias para a lista de monumentos/pontos de interesse disponíveis, visualização de um monumento/ponto de interesse específico, alteração da informação e para apagar um monumento/ponto de interesse específico. Por fim, em cada uma dessas pastas, foi criada mais uma pasta de layouts, onde foi criada mais uma blade, responsável pela uniformidade do layout das páginas do backend.

A seguir, criei uma *blade* que representaria a página principal do *website*. À medida que se foi escrevendo o código *html*, acabou por ser necessário adicionar *Routes* que permitissem que fosse possível obter os dados relativos aos monumentos e pontos de interesse.

Conforme se foi desenvolvendo as páginas do *website*, foram surgindo várias dúvidas. Entre essas dúvidas, a única que não consegui resolver a tempo, mesmo com a ajuda do orientador Luís Rosário, foi a do *carousel*, atributo necessário para a implementação de um *slideshow* que mostrasse ao utilizador os vários pontos de interesse associados ao monumento selecionado, acabando assim por optar por uma lista de pontos de interesse para que não perdesse muito tempo, já que na altura o tempo até a apresentação já estava escasso. No entanto, esta questão de mostrar os pontos de interesse acabou por ficar pendente de qualquer forma, devido a um erro na obtenção dos pontos de interesse ao qual não consegui detetar onde exatamente estava a falha.

## Aplicação Android

Comecei por criar um projeto no Android Studio.

Depois da criação do projeto, criei as várias classes que se viram necessárias para o desenvolvimento do projeto. Estas classes foram, inicialmente, três *activities*, sendo estas a *activity* de registo de conta de utilizador, a *activity* de login (que é também a *activity* principal) e a *activity* de monumentos.

Na *activity* de login, a UI é composta por 2 *edittext*, uma para o e-mail e uma para a password, juntamente com 2 botões, sendo um utilizado para navegar para a *activity* de registo de conta e outro para navegar para a *activity* de monumentos.

Na *activity* de registo de conta, criei 6 *edittext*, sendo estas para o primeiro nome, último nome, número de telemóvel, e-mail, password e confirmação da password. Criei ainda um botão para registar a conta e um botão para regressar ao ecrã de login, caso o utilizador tenha navegado para este ecrã acidentalmente.

A ideia inicial seria criar um *fragment container*, que conteria o conteúdo da lista de monumentos, que iria ser substituído pelo *fragment* seguinte conforme necessário. No entanto, não consegui fazer isso uma vez que surgiram dificuldades relativamente à passagem de argumentos de um *fragment* para outro. Após várias tentativas de passar argumentos entre os *fragments*, acabei por desistir dessa ideia inicial, e comecei a fazer a aplicação apenas com *Activities*, uma vez que o tempo já estava curto e era a forma mais simples para que houvesse o que apresentar.

Sendo assim, criei mais duas *activities*, sendo elas a *activity* de detalhes de um monumento específico e a *activity* de detalhes de um ponto de interesse específico.

Na *activity* de detalhes de um monumento, foram criadas *textview* para o nome, horário de funcionamento, dia de encerramento, localização e descrição do monumento, uma *imageview* que contem a foto do monumento, uma *ratingbar* que apresenta a classificação do monumento, baseado na média das classificações dos pontos de interesse ligados ao monumento em questão. Esta *activity* possui ainda uma lista de pontos de interesse ligados a este monumento, possuindo cada linha desta *listview* a foto do ponto de interesse e o nome, podendo ainda apresentar um ícone de estrela caso o utilizador o marque o ponto de interesse como visto.

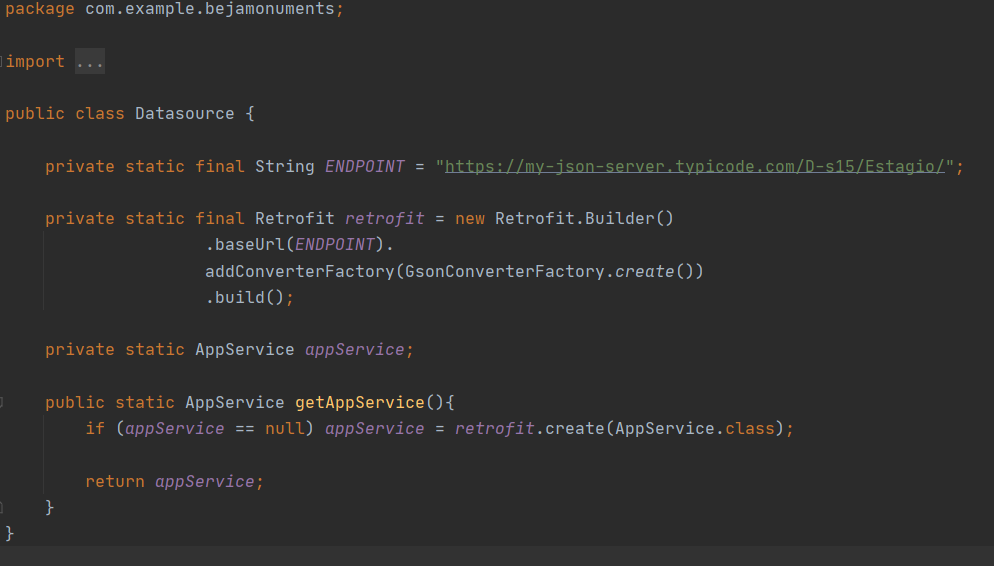
Por fim, para a *Activity* de detalhes de um ponto de interesse, criei *textview* para o nome do ponto de interesse e para a descrição, assim como uma *imageview* para a foto do monumento. Criei ainda um botão que permite alterar se o ponto de interesse já foi visualizado ou não. Também criei uma *ratingbar*, para que o utilizador possa avaliar o ponto de interesse. Por fim, criei mais um botão, que permite submeter a classificação.

Foram criadas duas interfaces, uma para servir de *Dao* para os pontos de interesse, onde seria realizada a *query* necessária que possibilitasse marcar um ponto de interesse como visto e outra para os utilizadores, onde seria realizada a *query* para a autenticação dos utilizadores (embora não tenha havido tempo para implementar as funcionalidades desejadas).

Criei também uma classe abstrata, com o intuito de servir como a base de dados da aplicação. Para tal foi necessário importar a biblioteca *Room*.

Foram criados *adapters* para os monumentos e para os pontos de interesse, para que fosse possível preencher as *listview* com dados programaticamente.

Para além da biblioteca *Room*, houve necessidade de incluir também outras bibliotecas, sendo estas o G*lide*, para a implementação das imagens e o *Retrofit*, para o uso dos *view models* e do *repository.*

Figura 14 – Código do Datasource

A imagem acima apresenta o código presente na classe do *datasource.* Nesta classe, foi criada uma String que representa o endpoint, onde foi fornecido o link base do my json server. Foi então chamado o retrofit que, com a ajuda do conversor *Gson,* transformou o json criado anteriormente em objetos java que pudessem ser lidos pela aplicação android. Depois desta conversão, foi criado o método “getAppService”, responsável pela conexão entre o *datasource* e o *service,* o que permite que a aplicação consiga aceder à informação contida no endpoint e possa então apresenta-la nas activities.



Figura 15 – Código do AppService

A seguir criei a AppService, uma *interface* que faz a conexão entre o datasource e o repositório, responsável por fazer a troca de dados entre o endpoint e o viewModel que, por sua vez, transmite os dados obtidos para as activities para que possam ser apresentados. Dentro do AppService, foram criados 4 “*get*”, responsáveis por obter os dados dos vários monumentos presentes (getMonumentList), de um monumento específico (getMonument), dos pontos de interesse associados a um monumento específico (getInterestPointsByMonumentId), e um ponto de interesse específico (getInterestPoint).

Foram então criados os *viewmodels* para os monumentos e pontos de interesse, ambos ligados a um único *repository*. O *Repository* tem a responsabilidade deaceder o *appservice*, para que consiga os dados necessários e devolvê-los para o *viewmodel* que, por sua vez, tem a responsabilidade de receber estes dados e devolvê-los para as *activities*.

Após escrever o código necessário para que fosse apresentada a informação dos monumentos e dos pontos de interesse nas respetivas activities, foram criadas *Querys* nos *Dao*, responsáveis pela autenticação do utilizador e pela marcação de um ponto de interesse como visto. Havia também intenção de implementar um método para que a *ratingbar* da activity de detalhes dos monumentos utilizasse uma média baseada na classificação dos pontos de interesse e uma opção para procurar monumentos por texto, mas não foi possível concluir a tempo.

## Dificuldades Sentidas ao longo do projeto

Logo de Início, senti alguma insegurança relativamente ao *design* que foi apresentado inicialmente, como esboço do projeto. Sinto que fui muito simplista em relação ao desenho de ecrãs, e que é algo que eu deveria explorar mais.

Relativamente às dificuldades sentidas na prática, a principal parte do projeto que eu senti mais dificuldades foi na aplicação Laravel. Foi a parte que, sem dúvida, eu mais precisei da ajuda dos orientadores de Estágio. Acredito que se deve à falta de prática, acabei por me preparar bem mais para o desenvolvimento de aplicações android e por não praticar tanto assim na parte do desenvolvimento Web, o que me causou algum desânimo em determinado momento do estágio, por sentir que não era capaz de o fazer.

Acredito que o principal erro cometido ao longo do Estágio foi tentar desenvolver tudo por mim mesmo. Acabei por consultar a internet quando precisava, mas em determinado momento acabei por ficar cada vez mais confuso conforme tentava relembrar os conteúdos de html e css. Acabei por pedir ajuda aos orientadores tarde demais, o que resultou em uma correria para tentar terminar o projeto a tempo de entrega, já não tendo tempo suficiente para desenvolver tudo o que pretendia para este projeto.

## Conclusão

Com este estágio, aprimorei os conhecimentos que já tinha sobre desenvolvimento de aplicações, tanto laravel como android. Acabei por ficar com uma ideia melhor sobre os *fragments* (mesmo não tendo conseguido implementar completamente na aplicação), que era algo que eu não tinha entendido bem no primeiro semestre do curso.

Gostaria, no entanto, de ter conseguido implementar uma procura de monumentos por nome na aplicação android, mas acabei por não conseguir implementar essa funcionalidade a tempo. Para além disso, gostaria ainda de melhorar a questão de *design* de ecrãs daqui para a frente.

## Glossário

API - Application Programming Interface.

Json – JavaScript Object Notation.

Html - Hypertext Markup Language.

Css – Cascading Style Sheets.

Blade – um template incluído no laravel que permite a utilização de código php na criação de páginas web.

Activity – Ecrã de uma aplicação android.

Fragment – parte reutilizável da UI da aplicação android.

UI – User Interface

fragment container – Layout responsável pela apresentação de fragments em uma activity.

GSON – biblioteca da google que permite a serialização de objetos java para JSON e vice-versa.

Dao – Data Access Object

Query – pedido de dados ou informação a uma tabela de uma base de dados.

## Bibliografia

<https://github.com/google/gson>

<https://github.com/bumptech/glide>

<https://developer.android.com/topic/libraries/architecture/viewmodel>

<https://developer.android.com/topic/libraries/architecture/livedata>

<https://developer.android.com/guide/fragments/create#groovy>

<https://developer.android.com/jetpack/androidx/releases/room?gclid=CjwKCAjw4qCKBhAVEiwAkTYsPD6smW2K6nWm_A_OyPepTL1_59_SPfzLeh13eXXvD7Do6ZhxWrkLvxoCDegQAvD_BwE&gclsrc=aw.ds>

<https://square.github.io/retrofit/>

<https://laravel.com/docs/4.2/quick>

<https://my-json-server.typicode.com>

[www.figma.com](http://www.figma.com)

[www.wix.com](http://www.wix.com)