universidade de aveiro estga escola superior de tecnologia e gestão de águeda

Licenciatura em Tecnologias da Informação

PROJETO TEMÁTICO EM APLICAÇÕES SIG

Ano Letivo de 2018/2019 | 2º Semestre

EstaArte - Roteiros de Arte Urbana

Autores:

Hugo Neves | 88167 _____

João Santos | 88007 _____

Luís Batista | 78387 _____

Tiago Silva | 87913 _____

Grupo 3

Águeda | 23 de junho de 2019

universidade de aveiro estga escola superior de tecnologia e gestão de águeda

Licenciatura em Tecnologias da Informação

PROJETO TEMÁTICO EM APLICAÇÕES SIG

Ano Letivo de 2018/2019 | 2º Semestre

EstaArte - Roteiros de Arte Urbana

Autores:

Hugo Neves | 88167 _____

João Santos | 88007 _____

Luís Batista | 78387 _____

Tiago Silva | 87913 _____

Grupo 3

Professor Orientador: Luís Jorge Gonçalves

Águeda | 23 de junho de 2019

Agradecimentos

Serve este espaço para agradecer a quem merece, a quem acredita em nós, e a quem luta incondicionalmente pelo nosso sucesso e bem-estar profissional. Deixamos os seguintes agradecimentos:

Em primeiro lugar, ao nosso orientador do projeto temático, o docente Luís Jorge Gonçalves que acompanhou todos os nossos passos ao longo de todo o trabalho, orientando-nos sempre da melhor forma possível.

Por fim agradecemos a todos os nossos colegas, que de uma forma ou de outra, contribuíram para que conseguíssemos levar a bom porto esta tarefa, com conselhos e incentivos variados. A todas estas pessoas e outras que cruzaram o nosso caminho nesta jornada o nosso muito obrigado. Um grande bem-haja para todos vós!

Resumo

O presente relatório foi elaborado como parte integrante do Projeto Temático em Aplicações SIG, associado à unidade curricular Sistemas de Informação Geográfica, orientado no âmbito da Licenciatura em Tecnologias da Informação da Escola Superior de Tecnologia e Gestão de Águeda. O projeto foi encaminhado pelo docente Luís Jorge Gonçalves durante o segundo semestre do segundo ano da licenciatura. Neste trabalho foi proposto a criação de uma plataforma de caracter turístico para orientar os utilizadores na visita de pontos de arte urbana numa cidade, e como exemplo foi focada na cidade de Estarreja. Numa fase inicial descreveu-se o âmbito do problema, baseado no tema escolhido, fez-se o planeamento do projeto, identificou-se e caracterizou-se os requisitos necessários, funcionais e não funcionais. Posteriormente seguiu-se o desenho do sistema, onde se identificou e descreveu os casos de utilização, o modelo de dados persistente e o protótipo de alta fidelidade. Por fim, na fase de implementação, implementou-se a camada de apresentação e programação server side.

O presente trabalho culmina com uma reflexão final acerca deste percurso ao longo do desenvolvimento do sistema para orientar os utilizadores na visita de pontos de arte urbana na cidade de Estarreja, bem como algumas sugestões a colmatar lacunas do mesmo.

Índice Geral

Índice de Tabelas

Índice de Figuras

Lista de Siglas e Abreviaturas

PAU – Ponto de Arte Urbana. Ponto no mapa marcado a azul que pode ter vários tipos e autores.
SIG- Sistemas de Informação Geográfica
AJAX- *Asynchronous Javascript and XML*
JSON - *Javascript Object Notation*
PHP - *Personal Home Page*
PI - Ponto de Interesse

1 Introdução

No âmbito do Projeto Temático em Aplicações SIG, do segundo semestre do segundo ano da Licenciatura em Tecnologias de Informação da Escola Superior de Tecnologia e Gestão de Águeda - Universidade de Aveiro (ESTGA-UA), foi proposto desenvolver uma aplicação para orientar os utilizadores na visita de pontos de arte urbana na cidade de Estarreja. O docente responsável pela orientação do grupo foi o professor Luís Jorge Gonçalves.

O desenvolvimento desta aplicação revelar-se-á importante tanto para as cidades que se pretendem promover e apresentar os pontos turísticos de arte urbana disponibilizados pelas mesmas, como para qualquer pessoa que pretenda procurar a localização desses pontos e visitá-los. Permitirá à cidade gerir os pontos de arte urbana e apresentar os mesmos de uma maneira fácil e intuitiva para qualquer utilizador. Permitirá aos utilizadores, por sua vez, tomar conhecimento da arte urbana da cidade de Estarreja e planear rotas de visita que mais lhe agrada consoante as suas variantes (interesse, tempo, distância). Outro aspeto importante da aplicação é que as rotas só serão possíveis de ser realizadas de carro.

A cidade de Estarreja irá então disponibilizar os pontos de arte urbana da cidade assim como os pontos de interesse que estão próximos dos mesmos (restaurantes, cafés, multibancos), e consequentemente a possibilidade de os filtrar consoante a escolha do utilizador.

É pretendido, no âmbito do projeto temático, definir a arquitetura de uma aplicação Web para resolver o problema em questão, criar páginas estáticas utilizando linguagens de marcação (HTML), desenvolver a camada de apresentação de uma aplicação Web utilizando programação do lado do cliente e desenvolver aplicações Web robustas e extensíveis utilizando programação do lado do servidor utilizando *routing* e funcionalidades de informação geográfica. Para cumprir os objetivos do projeto foi proposta uma solução em que alguns dos objetivos do sistema são os seguintes:

- Apresentar no mapa os pontos de arte urbana;
- Visualizar informação dos pontos de arte urbana;
- Filtrar pontos de arte urbana no mapa;
- Filtrar rotas de visita;
- Apresentar rota que passa por todos os pontos de arte urbana;
- Apresentar pontos de calor que definem a maior concentração de pontos de arte urbana;
- Apresentar pontos de interesse;

Quanto à metodologia de trabalho adotada, o grupo reúne duas vez por mês com o orientador, e a atribuição e controlo das tarefas por realizar, em curso e concluídas, é efetuada através de um software de gestão de projetos - Trello - baseados no cronograma previsto construído com o software Smartsheet. As tarefas são distribuídas pelos membros do grupo antes do término de cada reunião, para melhor gerir o tempo e os recursos ao longo da semana de trabalho que se segue. O grupo utiliza também a plataforma code.ua, uma ferramenta disponibilizada pelos serviços académicos que permite criar um projeto novo ou utilizar um já existente tendo acesso a alguns parâmetros relacionados com o planeamento e gestão de projetos. Permite ao gestor ou ao grupo estabelecer um planeamento conciso e organizado do projeto, utilizando os recursos disponibilizados pelo site.

1.1 Organização interna do relatório

O presente relatório encontra-se dividido em 5 capítulos, correspondendo o primeiro ao tópico atual. A segunda parte diz respeito à planificação e execução do projeto, que engloba as tarefas a serem realizadas e o cronograma, previsto e executado. A terceira parte corresponde ao levantamento do estado da arte, onde se descrevem as soluções semelhantes/inspiradoras ao sistema em desenvolvimento e especificação e prototipagem de alta fidelidade, encontra-se o levantamento de requisitos, tanto funcionais como não funcionais, o diagrama de casos de utilização com os respetivos casos de utilização. Segue-se a base de dados que engloba o modelo de dados persistente. Num quinto momento surge o desenvolvimento da camada de apresentação, correspondente à implementação da funcionalidade tanto por parte do cliente como do servidor e ao desenvolvimento da aplicação, no que toca à programação. Segue-se a análise de resultados, a conclusão, as referências bibliográficas e os anexos, correspondendo, respetivamente, ao oitavo, nono, décimo e décimo primeiro capítulo.

2 Planificação do Projeto

2.1 Tarefas a Serem Realizadas

A fase de planeamento deve contemplar várias etapas onde se definem os objetivos gerais, o tipo de tarefas, bem como o seu âmbito e a sua divisão, estabelecem-se os recursos necessários à concretização dos objetivos e tarefas e procede-se à calendarização. No âmbito da fase de planeamento, a elaboração de um cronograma é uma forma de organizar e gerir as tarefas durante a execução de um projeto. Consiste na determinação da melhor forma de posicionar as tarefas ao longo do tempo de acordo com a duração das mesmas, das relações de precedência entre elas e dos prazos a cumprir. Com base nisto, foi elaborado um cronograma correspondente a este projeto no qual estão presentes as atividades previstas bem como o tempo previsto de execução de cada uma delas.

2.2 Escalonamento das Tarefas – Planeado

Resumidamente, as tarefas definidas para a primeira fase do projeto foram as seguintes:

- Planeamento;
- Levantamento dos requisitos funcionais e não funcionais;
- Levantamento do estado de arte;
- Desenvolvimento da introdução;
- Identificação dos casos de utilização;
- Desenvolvimento de um protótipo de alta fidelidade;
- Desenho e implementação de um modelo de dados;
- Descrição do processo de implementação da aplicação Web;
- Descrição da arquitetura e organização da aplicação Web;
- Descrição da forma como o site está distribuído;
- Análise dos resultados;
- Conclusões;
- Reuniões;
- Relatório

No cronograma é possível observar as tarefas executadas, bem como o tempo previsto de execução de cada uma delas.

Nome da Tarefa	Duração	Início	Conclusão	Predecessoras
1.1 Escolha do tema	1 sem	Sex 01/03/19	Qui 07/03/19	
1.2 Planificação do projeto	1 sem	Sex 01/03/19	Qui 07/03/19	
2. Levantamento de requistos	1 sem	Sex 08/03/19	Qui 14/03/19	4
3. Levantamento do estado de arte	5 dias	Sex 08/03/19	Qui 14/03/19	4
4. Identificação dos casos de utilização	7 dias	Sex 15/03/19	Seg 25/03/19	5
5. Desenho e implementação de um modelo de dados	7 dias	Ter 26/03/19	Qua 03/04/19	7
6. Desenvolvimento de um protótipo de alta fidelidade	2 sems	Qui 04/04/19	Qua 17/04/19	8
7. Densolvimento da aplicação	**40 dias**	**Qui 18/04/19**	**Qua 12/06/1**	
7.1 Descrição do processo de implementação da aplicação	4 sems	Qui 18/04/19	Qua 15/05/19	9
7.2 Descrição da arquitetura e organização da aplicação	3 sems	Qui 16/05/19	Qua 05/06/19	11
7.3 Descrição da forma como o site está distribuído	1 sem	Qui 06/06/19	Qua 12/06/19	12
8. Análise dos resultados	1 sem	Qui 13/06/19	Qua 19/06/1	13
9. Conclusões	2 dias	Qui 20/06/19	Sex 21/06/19	14

Figura 1- Lista de tarefa do mapa de gantt

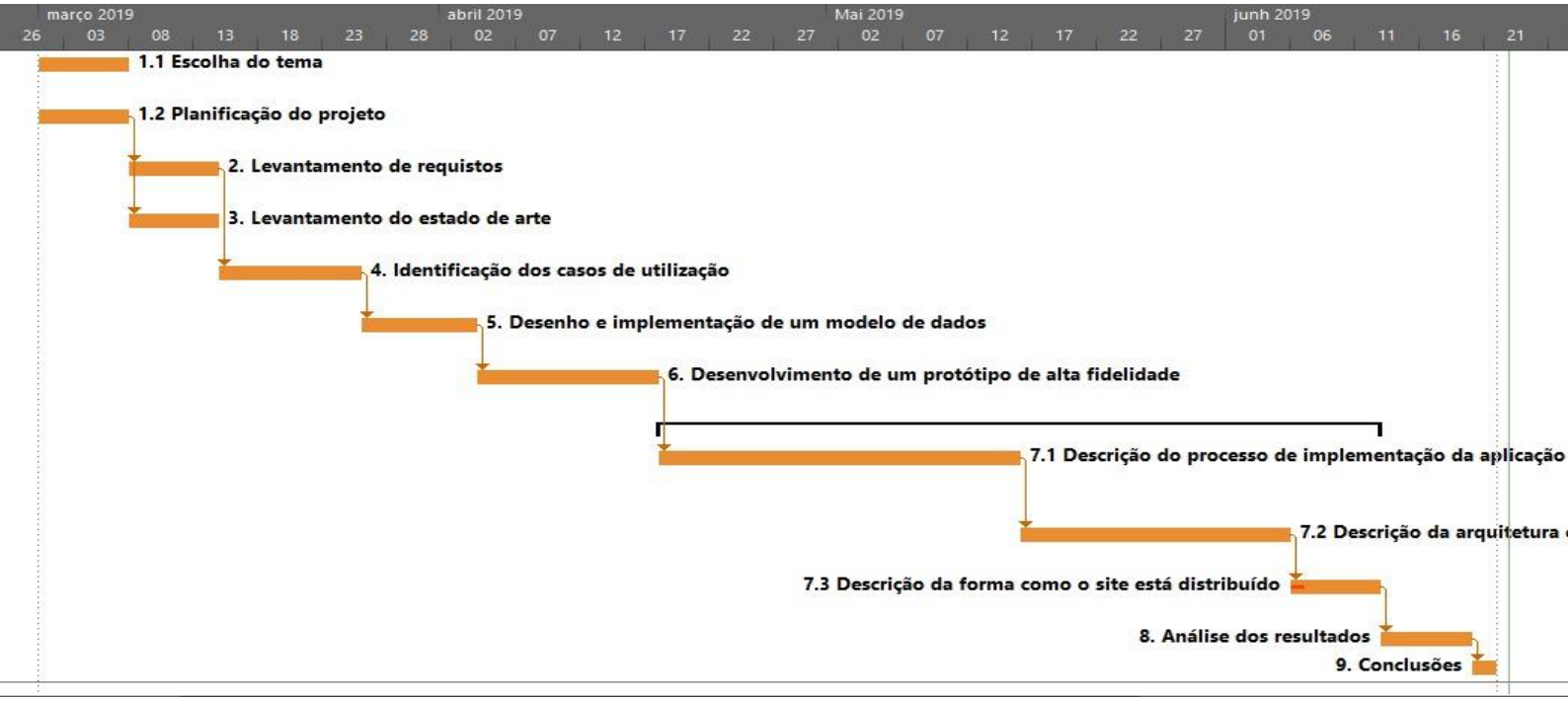


Figura 2 - Cronograma executado

3 Levantamento do estado de arte

Esta fase do planeamento tem como objetivo analisar as soluções já disponíveis no mercado e estudar como é que estas são implementadas. Com isto pretende-se retirar ideias que sejam úteis ao desenvolvimento da aplicação e melhorar funcionalidades que estas disponibilizem no sentido de criar uma aplicação de nível superior.

Na área de aplicações de roteiro de arte urbana foram encontradas duas soluções. Estas soluções têm o intuito de facilitar aos utilizadores encontrarem os pontos de arte urbana e visita-los.

AgitÁgueda

Pelo que foi possível apurar, a aplicação AgitÁgueda fornece várias funcionalidades. Tais como a apresentação de roteiros com os pontos de arte urbana da cidade de Águeda e apresenta as informações. Também existe certos pontos de arte que com a acesso à aplicação podem tornar se animados no telemóvel do utilizador.

Esta solução foi uma das selecionadas por conter certas características que o grupo considerou serem fundamentais para o desenvolvimento do projeto, sendo uma delas a apresentação os pontos de arte num mapa, que é uma funcionalidade fundamental da nossa aplicação. Outra característica que se pretende incluir na solução a ser proposta é o conceito de *pop-up,* que ao clicar num ponto irá aparecer uma caixa com a informação do ponto selecionado.

Figura 3 - AgitÁgueda

Lisbon Street Art

A *Lisbon Street Art* é uma aplicação que fornece informação sobre os vários pontos de arte na cidade de Lisboa. Além de providenciar várias informações, esta aplicação têm a possibilidade de mostrar ao utilizador, através da sua localização, o trajeto de um ponto selecionado através da API do google.

Perante esta solução, foi considerado necessário implementar a apresentação do trajeto para a aplicação. Uma das características que se pretendem ser introduzida é a opção do utilizador escolher mais que um ponto de arte urbana, sendo apresentado depois um trajeto que passa por esses pontos selecionados.

3.1 Levantamento de Requisitos

3.1.1 Requisitos Funcionais

Os requisitos funcionais são definidos como as funcionalidades ou atividades que um sistema/software deve realizar. Representa o que o software faz, em termos de tarefas e serviços, através de cálculos, detalhes técnicos, manipulação de dados e de processamento e outras funcionalidades específicas que definem o que um sistema, idealmente, será capaz de realizar.

3.1.1.1 Requisitos Funcionais

Na seguinte tabela (**Erro! A origem da referência não foi encontrada.**) estão representados os requisitos funcionais, ordenados por prioridade.

Tabela 1- Requisitos funcionais e suas prioridades

Refª	**Requisito Funcional**	**Prioridade**
RF1.01	Apresentação do PAU (Pontos de arte Urbana) no mapa	Alta
RF1.02	Filtragens de pontos por tipo de arte	Alta
RF1.03	Selecionar os pontos que eu quero visitar	Alta

RF1.04	Calcular Rota com pontos pré-definidos	Alta
RF1.05	Rota pré-definida a passar por todos os pontos	Alta
RF1.06	Apagar Rota	Alta
RF1.07	Mostrar pontos de interesse circundantes	Alta
RF1.08	Selecionar pontos de partida e chegada através da localização GPS	Alta
RF1.09	Selecionar pontos de partida e chegada através da localização opcional	Alta
RF1.10	Apresentação da maior concentração de pontos no mapa	Media
RF1.11	Filtragem por tempo disponível	Media
RF1.12	Como se vai realizar o percurso	Baixa
RF1.13	Selecionar rotas pelo tempo disponível do utilizador	Baixa

3.1.2 Requisitos Não Funcionais

Os requisitos não funcionais devem conter elementos específicos, tais como a descrição da tarefa a ser executada pelo software, a origem do requisito e o seu utilizador, a relação da passagem de informação entre o software e o utilizador e, se existirem, algumas restrições lógicas associadas à tarefa. Dentro dos requisitos não funcionais estão incluídos os requisitos de interface e facilidade de uso, de desempenho, de segurança e integridade de dados, de interface com sistemas externos e ambientes de execução, entre outros. Os requisitos não funcionais estão relacionados com os requisitos funcionais e indicam como o sistema/software deve ser feito e como deve funcionar, ou seja, são os critérios que qualificam os requisitos funcionais.

3.1.2.1 Requisitos Não Funcionais

Na seguinte tabela (Tabela 2) estão representados os requisitos não funcionais, ordenados por prioridade.

Tabela 2 - Requisitos não funcionais e suas prioridades

Refª	**Requisito Não Funcional**	**Prioridade**
RNF1.01	Diferenciação de marcadores que representam diferentes entidades implícitas no mapa	Alta
RNF1.02	Responsividade na aplicação	Media

3.2 Modelo de Casos de Utilização

Neste tópico será abordado o ator que irá interagir com o sistema, através da representação de um diagrama de casos de utilização com a descrição do mesmo, servindo assim como complemento ao levantamento dos requisitos funcionais e não funcionais.

No diagrama de casos de utilização será descrito todas as interações por parte do utilizador ao sistema.

3.2.1 Descrição dos atores

. O ator que irá realizar a interação com o sistema, será o utilizador que poderá visualizar pontos de arte urbana, definir rotas e visualizar rotas.

Ator do sistema

Ator	**Descrição**
Utilizador	Funcionário da empresa cliente que gere as máquinas da empresa. Envia pedidos de apoio. Tem acesso a diferentes estatísticas, por exemplo, do estado das máquinas.

3.2.2 Diagramas de Casos de Utilização

O diagrama de caso de uso abaixo (**Erro! A origem da referência não foi encontrada.**) descrevem a funcionalidade proposta para o sistema implementado. Cada caso de uso representa uma interação entre um utilizador (humano ou máquina) e o sistema, interação essa que tem de ser considerada uma ação (unidade de trabalho significativa).

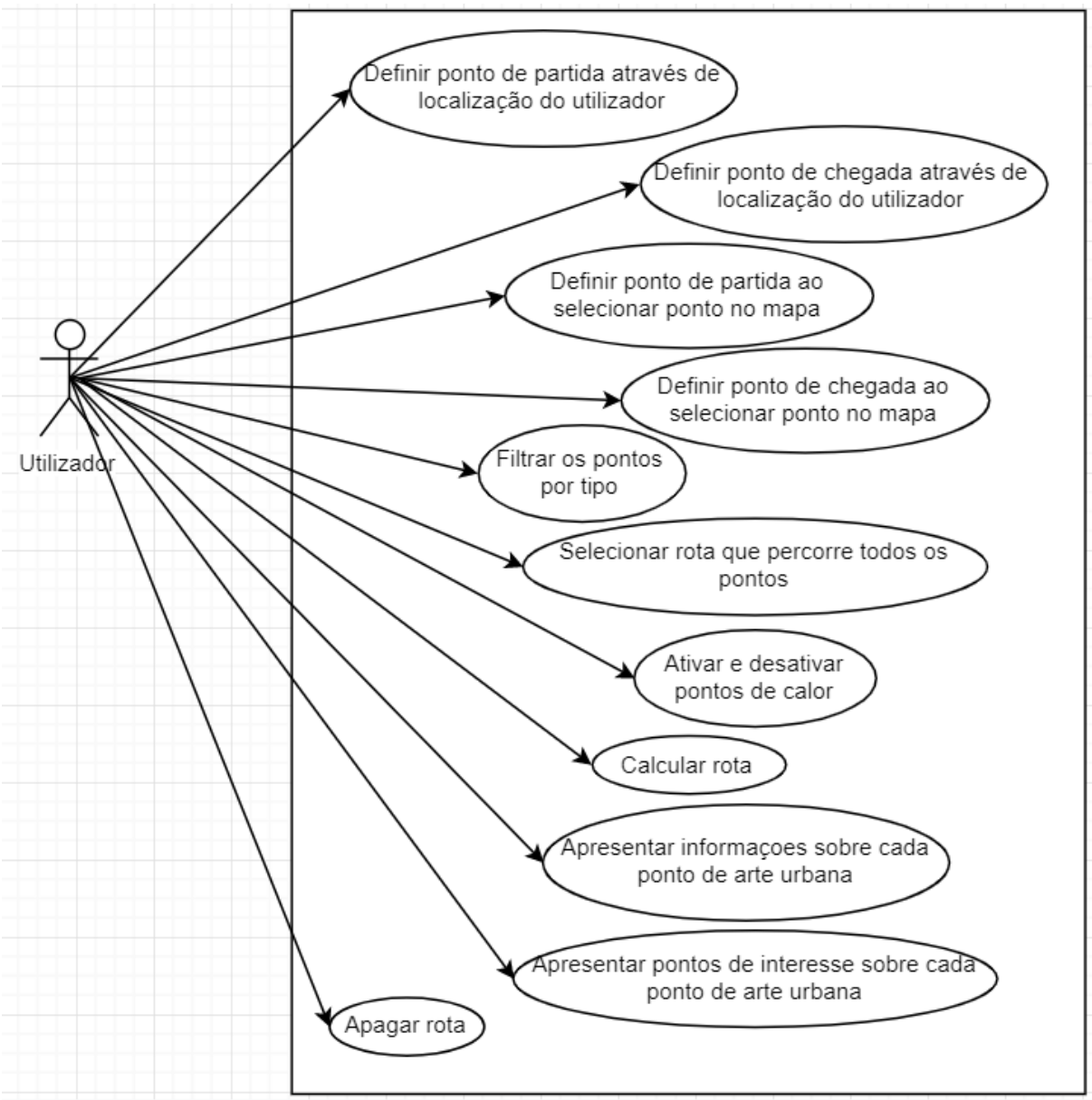


Figura 4- Diagrama de casos de utilização

3.2.3 Descrição dos casos de utilização

Esta secção do relatório destina-se a uma breve descrição dos casos de utilização mais relevantes, sendo estes determinados pela prioridade dada aos requisitos funcionais.

Tabela 3 - Especificação do caso de utilização " Definir ponto de partida através da localização do utilizador "

Nome	Definir ponto de partida através da localização do utilizador
Requisitos funcionais	RF1.08
Descrição	O utilizador pode definir a sua localização como ponto de partida, pressionando o botão..

Tabela 4 - Especificação do caso de utilização " Definir ponto de chegada através da localização do utilizador "

Nome	Definir ponto de chegada através da localização do utilizador
Requisitos Funcionais	RF1.08
Descrição	O utilizador pode definir a sua localização como ponto de chegada, pressionando o botão.

Tabela 5 - Especificação do caso de utilização "Definir ponto de partida ao selecionar ponto no mapa "

Nome	Definir ponto de partida ao selecionar ponto no mapa
Requisitos Funcionais	RF1.09.
Descrição	O utilizador pode definir o ponto de partida ao selecionar um ponto no mapa.

Tabela 6 -Especificação do caso de utilização "Definir ponto de chegada ao selecionar ponto no mapa”

Nome	Definir ponto de chegada ao selecionar ponto no mapa
Requisitos Funcionais	RF1.09.
Descrição	O utilizador pode definir o ponto de chegada ao selecionar um ponto no mapa.

Tabela 7 - Especificação do caso de utilização " Filtrar os pontos por tipo "

Nome	Filtrar os pontos por tipo
Requisitos Funcionais	RF1.2.
Descrição	No menu lateral é possível que o utilizador filtre os vários pontos de arte urbana por tipo de arte.

Tabela 8 - Especificação do caso de utilização " Selecionar rota que percorre todos os pontos "

Nome	Selecionar rota que percorre todos os pontos
Requisitos funcionais	RF1.04.
Descrição	Ao entrar no site o utilizador terá acesso a um menu lateral onde terá várias opções. Para aceder ao mapa e à lista de máquinas terá que selecionar a opção Máquinas. Será redirecionado para uma nova página onde estarão disponíveis o mapa e a lista de máquinas.

Tabela 9- Especificação do caso de utilização "Definir ponto de chegada ao selecionar no mapa "

Nome	Definir ponto de chegada ao selecionar ponto no mapa
Requisitos Funcionais	RF1.09.
Descrição	O utilizador pode definir o ponto de chegada ao selecionar um ponto no mapa.

Tabela 10- Especificação do caso de utilização "Ativar e desativar pontos de calor"

Nome	Ativar e desativar pontos de calor
Requisitos Funcionais	RF1.10
Descrição	O utilizador ao selecionar opção dos pontos de calor na barra superior, pode ativar ou desativar, consoante o estado anterior.

Tabela 11- Especificação do caso de utilização "Calcular rota"

Nome	Calcular rota
Requisitos Funcionais	1.04
Descrição	No menu lateral, o utilizador quando terminar de escolher o seu percurso poderá calcular a rota ao pressionar o botão “Calcular”.

Tabela 12- Especificação do caso de utilização "Apresentar informações sobre cada ponto de arte urbana"

Nome	Apresentar informações sobre cada ponto de arte urbana
Requisitos Funcionais	RF1.01
Descrição	Ao ser pressionado um ponto de arte urbana no mapa, é apresentado ao utilizador as informações do mesmo através de um “popup”.

Tabela 13- Especificação do caso de utilização "Apresentar pontos de interesse sobre cada ponto de arte urbana"

Nome	Apresentar pontos de interesse sobre cada ponto de arte urbana
Requisitos Funcionais	RF1.07
Descrição	Ao ser pressionado um ponto de arte urbana no mapa, dentro do “popup” se o utilizador pressionar o botão “Pontos de interesse” serão destacados no mapa os pontos de interesse que se localizam a um minuto.

Tabela 14- Especificação do caso de utilização "Apagar rota"

Nome	Apagar rota
Requisitos Funcionais	RF1.06
Descrição	Na barra lateral o utilizador ao pressionar o botão “Apagar Rota” apaga a rota calculada.

4 Modelo de Dados Persistente

4.1 Modelo de dados persistente

Uma base de dados é uma ferramenta de recolha e organização de informações. Para a aplicação é crucial a utilização deste tipo de armazenamento de dados pois é nesta que será guardada todo o tipo de informações sobre os pontos de arte urbana e os pontos de interesse à sua volta, assim como a rede viária para que seja possível o cálculo de rotas entre os pontos. Primeiramente começou-se por pensar nas tabelas que são importantes para o bom funcionamento da aplicação em seguida nos campos que preenchem cada uma destas tabelas, com o objetivo de ser possível encaixar o conteúdo de forma organizada.

Das nove tabelas podemos destacar:

•pontos_arte_urbana: Tabela que armazena toda a informação relativa aos pontos de arte urbana da aplicação. De destacar os campos “tipo” do tipo inteiro, o campo “localização” do tipo geometry e o campo “is_active” do tipo boolean. O campo “tipo” define o tipo de arte que o ponto representa, o campo “localização” terá as coordenadas em que se encontra o ponto que posteriormente serão usadas para os representar no mapa e até incluir os mesmos nas rotas criadas e o campo “is_active” serve para dar informação dos pontos que existem e estão em exposição(valor a “true”) e os que existem mas não estão expostos(valor a “false”);

•tipos: Tabela que armazena os diferentes tipos de arte disponíveis;

•artistas: Contem os artistas que pelo menos realizaram uma das obras disponíveis na aplicação;

•pontos_interesse: Guarda todos os pontos de interesse na zona de Estarreja, onde guarda informação do nome horário e localização. No campo localização podemos retirar as coordenadas para depois criar funções de routing;

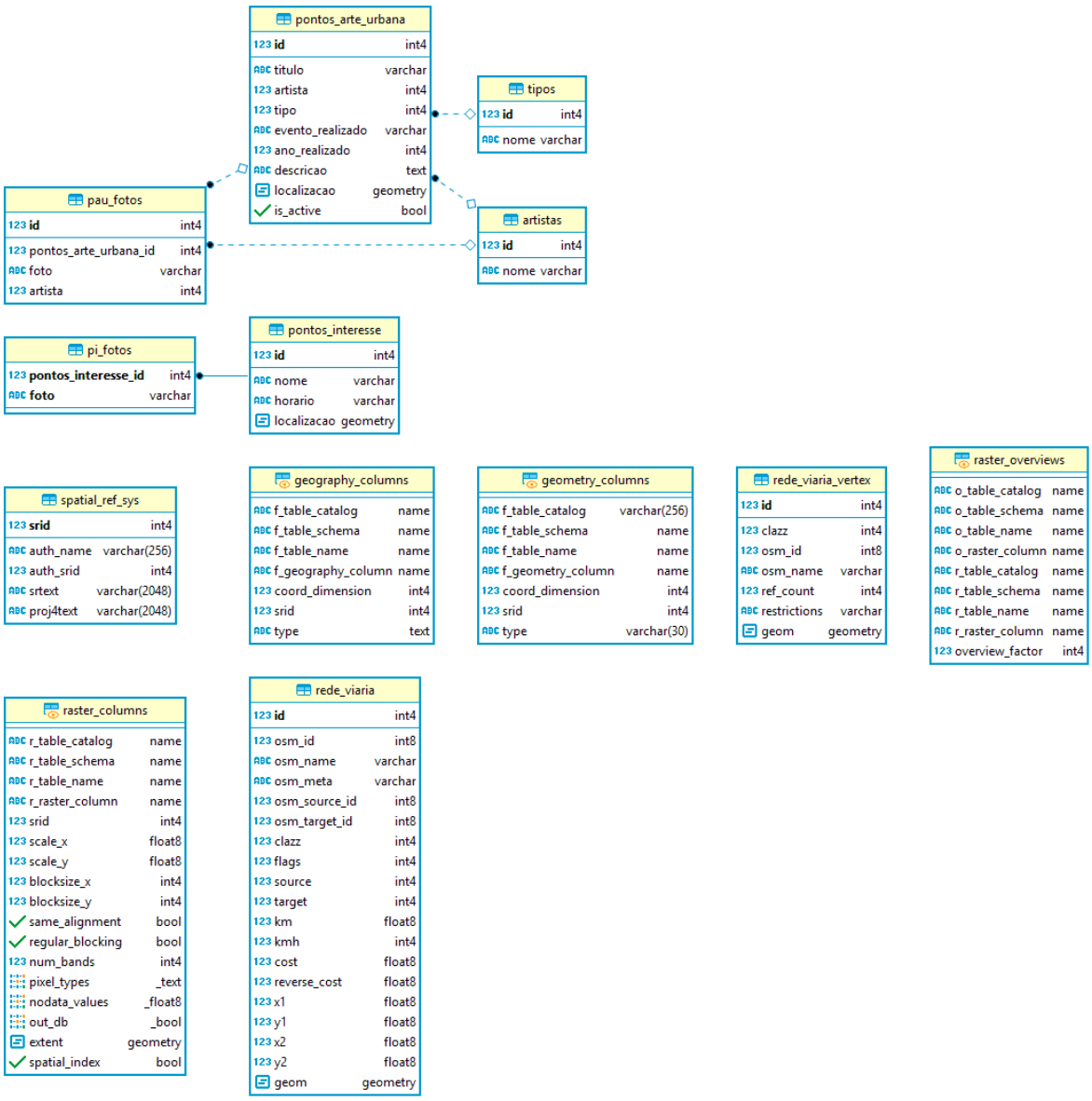


Figura 5- Modelo de base de dados

5 Desenvolvimento da Aplicação

5.1 Implementação

5.1.1 Implementação da Funcionalidade – (Apresentação do PAU)

5.1.1.1 Client Side

Os pontos de arte urbana são apresentados no mapa assim que a aplicação é carregada. Estão definidos com marcadores no mapa a cor azul. Estes marcadores são impressos no mapa através de uma camada que vai buscar os pontos a uma fonte que por sua vez faz uma chamada ajax, onde vai buscar toda a informação relativa a estes em formato GeoJSON à base de dados do projeto.

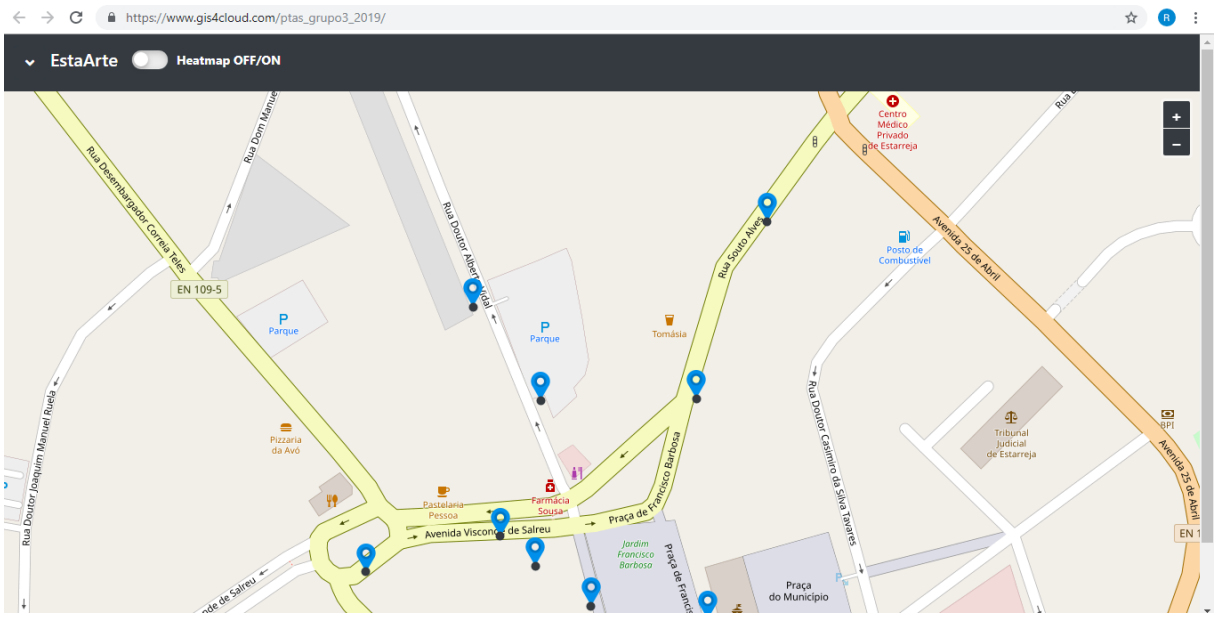


Figura 6- PAU no mapa

```
function load_paus() {

    var pontos_urbanos = new ol.layer.Vector({
        source: pontos_urbanos_source,
        // projection: 'EPSG:4326',
        style: pubStyle // var 'pubStyle' definida no file 'ponteiro_estilo.js'
    });

    $.ajax({
        url: 'php/get_pontos.php',
        async: false,
        success: function (data) {

            var geojsonFormat = new ol.format.GeoJSON();

            pontos_urbanos_source.clear();
            pontos_urbanos_source.addFeatures(geojsonFormat.readFeatures(data));

            map.addLayer(pontos_urbanos);
        },
        error: function (data) {
            return 'Error:' + data;
        }
    });
}
```

Figura 7- Pedido Ajax para o PAU

Para definir o estilo do marcador foi criado um style definido no ficheiro ponteiro_estilo.js e nomeado pubStyle como poderemos ver na figura seguinte:

```
pubStyle = [
    new ol.style.Style({
        image: new ol.style.Icon(({
            anchor: [0.5, 1],
            anchorXUnits: 'fraction',
            anchorYUnits: 'fraction',
            src: 'img/ponto.png',
            scale: 0.5
        }))
    }),
    new ol.style.Style({
        image: new ol.style.Circle({
            radius: 5,
            fill: new ol.style.Fill({
                color: 'rgba(52, 58, 64, 1)'
            })
        })
    })
];
```

Figura 8- Estilo dos marcadores para o PAU

5.1.1.2 Server Side

A query para o pedido retorna em formato GeoJSON todos os pontos de arte urbana. Neste formato é retornado o id, o título, o artista, o tipo, o evento a que pertencem, ano de realização, descrição e por fim localização.

```php
<?php

try{

include "db_connection.php";

$result = pg_query($db_connection, "SELECT jsonb_build_object(
    'type',     'FeatureCollection',
    'features', jsonb_agg(features.feature)
)
FROM (
  SELECT jsonb_build_object(
    'type',       'Feature',
    'geometry',   ST_AsGeoJSON(localizacao)::jsonb,
    'properties', to_jsonb(inputs) - 'localizacao'
  ) AS feature
  FROM (SELECT pontos_arte_urbana.id AS id, pontos_arte_urbana.titulo AS titulo, artistas.nome AS artista, tipos.nome AS tipo,
    pontos_arte_urbana.evento_realizado AS evento_realizado, pontos_arte_urbana.ano_realizado AS ano_realizado, pontos_arte_urbana.
    descricao AS descricao, pau_fotos.foto AS imagem, ST_TRANSFORM(pontos_arte_urbana.localizacao,3857) AS localizacao
FROM pontos_arte_urbana
INNER JOIN artistas ON pontos_arte_urbana.artista = artistas.id
INNER JOIN tipos ON pontos_arte_urbana.tipo = tipos.id
INNER JOIN pau_fotos ON pau_fotos.pontos_arte_urbana_id = pontos_arte_urbana.id
WHERE is_active = true) inputs) features") or die('Query failed: ' . pg_last_error());




echo pg_fetch_assoc($result)["jsonb_build_object"];
pg_close($db_connection);
} catch (Exception $e) {
    echo 'Caught exception: ',  $e->getMessage(), "\n";
}


?>
```

Figura 9- Query para obter o pau (retorna em formato GeoJSON)

5.1.2 Implementação da Funcionalidade – (Filtrar por tipo de arte)

5.1.2.1 Client Side

Na barra de navegação da aplicação existe um botão com uma seta em que ao clicar abre uma janela com diferentes opções como mostra a figura:

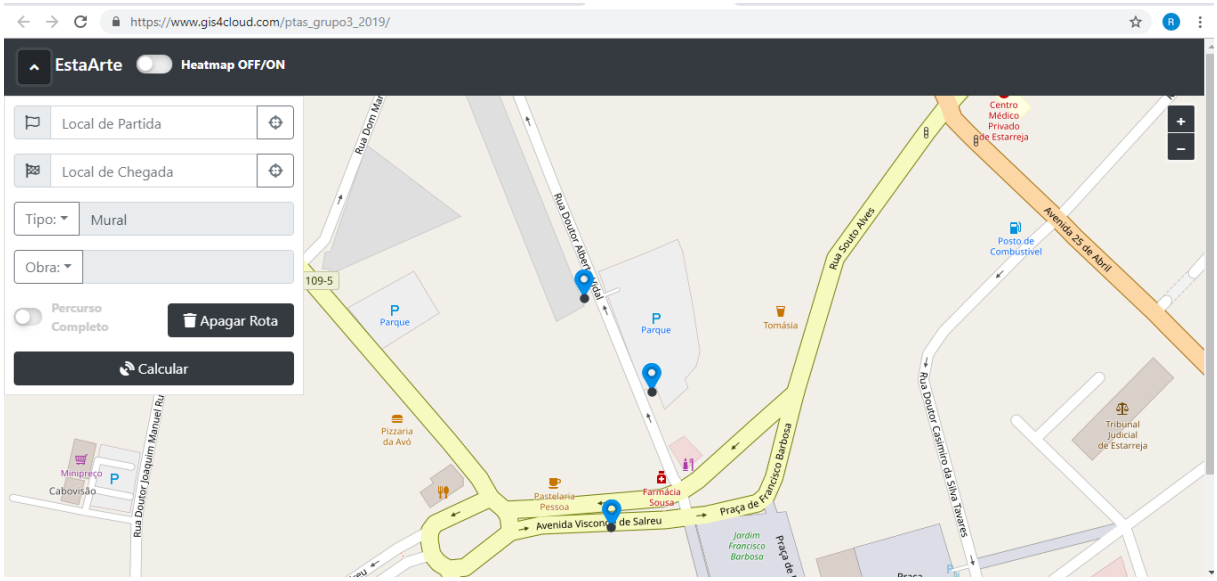
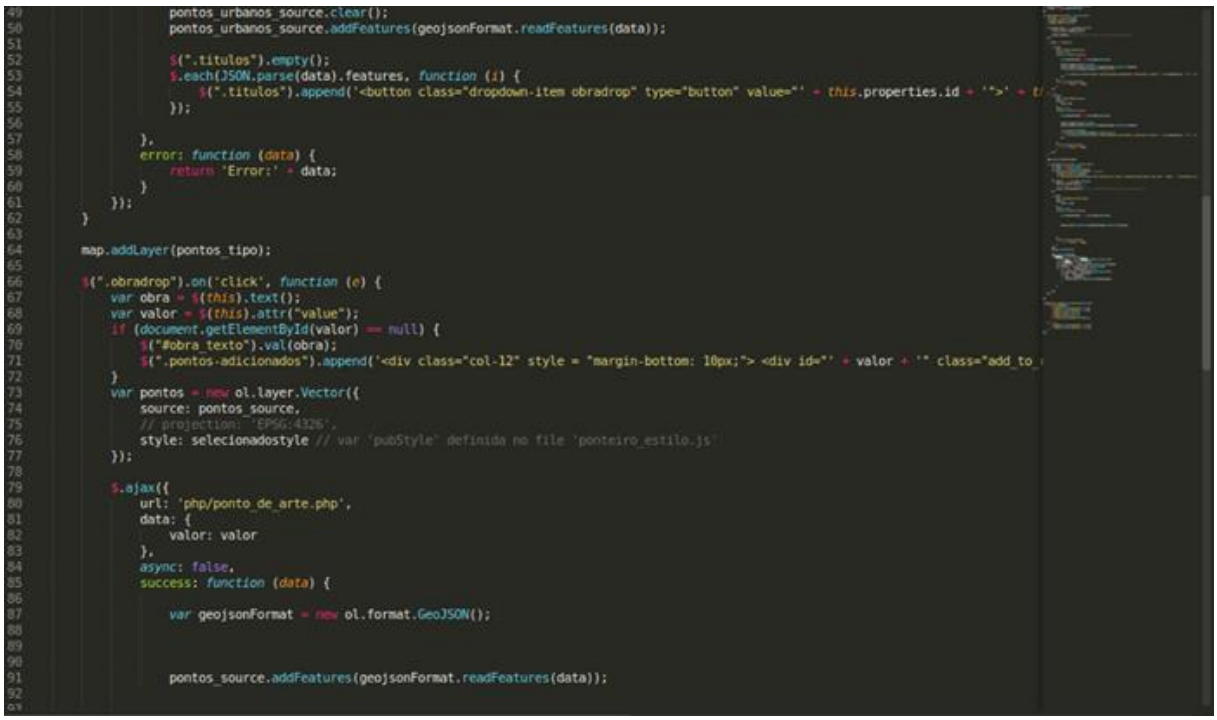


Figura 10- Janela de filtros

Para podermos escolher o tipo de arte que queremos visitar, basta apenas ir à opção “Tipo” e selecionar um de entre os vários tipos existentes. Com isto, no mapa apenas serão apresentados com marcador azul os pontos de arte urbana que são do tipo que o utilizador escolheu. Todo este processo foi feito através da criação de uma layer vai buscar uma fonte que é atualizada através de pedidos ao servidor através de chamadas Ajax consoante o tipo de arte escolhido.

```
$(".tipodrop").on('click', function (e) {
    var tipo = $(this).text();
    $("#tipo_texto").val(tipo);
    $("#obra_texto").val("");

    var pontos_tipo = new ol.layer.Vector({
        source: pontos_urbanos_source,
        // projection: 'EPSG:4326',
        style: pubStyle // var 'pubStyle' definida no file 'ponteiro_estilo.js'
    });

    if (tipo == "Todos") {

        $.ajax({
            url: 'php/get_pontos.php',
            async: false,
            success: function (data) {

                var geojsonFormat = new ol.format.GeoJSON();

                pontos_urbanos_source.clear();
                pontos_urbanos_source.addFeatures(geojsonFormat.readFeatures(data));
                $.each(JSON.parse(data).features, function (i) {

                    $(".titulos").append('<button class="dropdown-item obradrop" type="button" value="' + this.properties.id + '">' + t
                });
            },
            error: function (data) {
                return 'Error:' + data;
            }
        });

    } else {
        $.ajax({
            url: 'php/tipodearte.php',
            data: {
                tipo: tipo
            },
            async: false,
            success: function (data) {

                var geojsonFormat = new ol.format.GeoJSON();
```

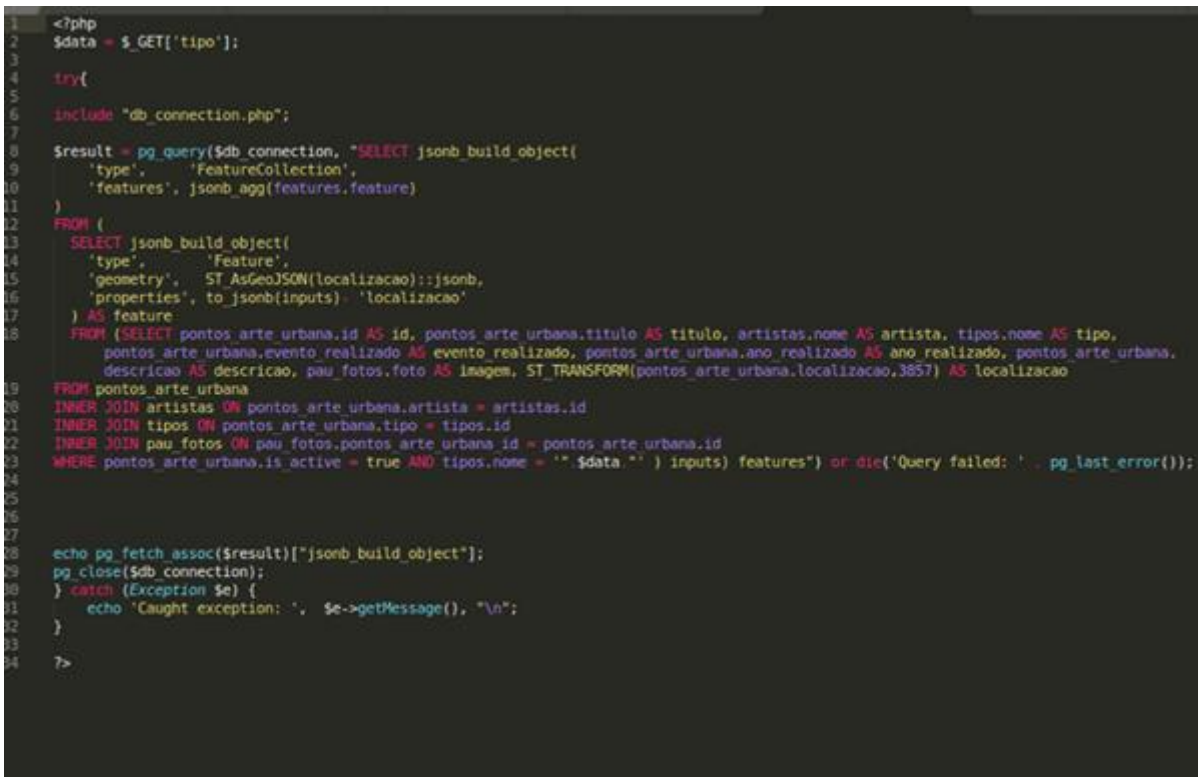


```
                pontos_urbanos_source.clear();
                pontos_urbanos_source.addFeatures(geojsonFormat.readFeatures(data));

                $(".titulos").empty();
                $.each(JSON.parse(data).features, function (i) {
                    $(".titulos").append('<button class="dropdown-item obradrop" type="button" value="' + this.properties.id + '">' + t
                });

            },
            error: function (data) {
                return 'Error:' + data;
            }
        });
    }

    map.addLayer(pontos_tipo);

    $(".obradrop").on('click', function (e) {
        var obra = $(this).text();
        var valor = $(this).attr("value");
        if (document.getElementById(valor) == null) {
            $("#obra_texto").val(obra);
            $(".pontos-adicionados").append('<div class="col-12" style = "margin-bottom: 10px;"> <div id="' + valor + '" class="add_to_
        }
        var pontos = new ol.layer.Vector({
            source: pontos_source,
            // projection: 'EPSG:4326',
            style: selecionadostyle // var 'pubStyle' definida no file 'ponteiro_estilo.js'
        });

        $.ajax({
            url: 'php/ponto_de_arte.php',
            data: {
                valor: valor
            },
            async: false,
            success: function (data) {

                var geojsonFormat = new ol.format.GeoJSON();


                pontos_source.addFeatures(geojsonFormat.readFeatures(data));
```

Figura 11- JavaScript para o tipo de arte

5.1.2.2 Server Side

Com o pedido Ajax é enviado um parametro de nome “tipo” que contém o tipo de arte escolhida na dropdown da janela de filtros. Esta variavel enviada pelo ajax serve para fazer o filtro na query em que retornará todos os pontos de arte urbana que sejam do “tipo” escolhido.



```php
<?php
$data = $_GET['tipo'];

try{

include "db_connection.php";

$result = pg_query($db_connection, "SELECT jsonb_build_object(
    'type',     'FeatureCollection',
    'features', jsonb_agg(features.feature)
)
FROM (
  SELECT jsonb_build_object(
    'type',       'Feature',
    'geometry',   ST_AsGeoJSON(localizacao)::jsonb,
    'properties', to_jsonb(inputs) - 'localizacao'
  ) AS feature
  FROM (SELECT pontos_arte_urbana.id AS id, pontos_arte_urbana.titulo AS titulo, artistas.nome AS artista, tipos.nome AS tipo,
      pontos_arte_urbana.evento_realizado AS evento_realizado, pontos_arte_urbana.ano_realizado AS ano_realizado, pontos_arte_urbana.
      descricao AS descricao, pau_fotos.foto AS imagem, ST_TRANSFORM(pontos_arte_urbana.localizacao,3857) AS localizacao
FROM pontos_arte_urbana
INNER JOIN artistas ON pontos_arte_urbana.artista = artistas.id
INNER JOIN tipos ON pontos_arte_urbana.tipo = tipos.id
INNER JOIN pau_fotos ON pau_fotos.pontos_arte_urbana_id = pontos_arte_urbana.id
WHERE pontos_arte_urbana.is_active = true AND tipos.nome = '" $data "' ) inputs) features") or die('Query failed: ' . pg_last_error());



echo pg_fetch_assoc($result)["jsonb_build_object"];
pg_close($db_connection);
} catch (Exception $e) {
    echo 'Caught exception: ',  $e->getMessage(), "\n";
}

?>
```

Figura 12- Query que retorna pontos de arte do tipo selecionado

5.1.3 Implementação da Funcionalidade – (Abrir informação dos pontos de arte urbana)

5.1.3.1 Client Side

Com os marcadores postos no mapa a marcar os pontos de arte urbana é possível aceder à sua informação mais específica ao clicar nos mesmos. Depois do clique é aberto um popup que de forma organizada mostra as características do ponto e também dois botões (aos quais serão explicados mais para a frente) como podemos verificar na figura 23. A estrutura do popup só é criada assim que algum ponto seja selecionado e precede-se a construção da mesma.

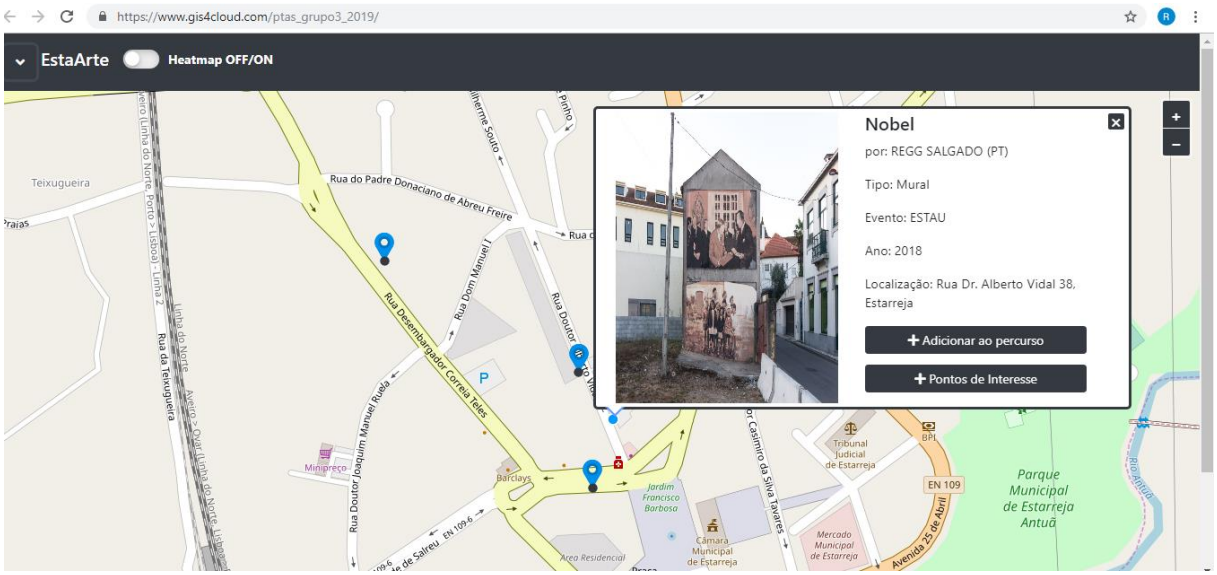


Figura 13- popup com a informação dos PAU

5.1.3.2 Server Side

Neste caso o popup já tem disponivel a informação necessária no GeoJSON criado no ficheiro php representado na figura 19.

5.1.4 Implementação da Funcionalidade - (Selecionar os pontos que eu quero visitar)

A seleção dos pontos vai servir para o cálculo da rota calculada para a visita dos mesmo como iremos ver nos pontos mais à frente. Existem duas maneiras de adicionarmos pontos à rota. Uma delas através da janela dos filtros e outra através do botão “adicionar ao percurso” que se econtra no popup ao clicar no mapa em cada ponto.

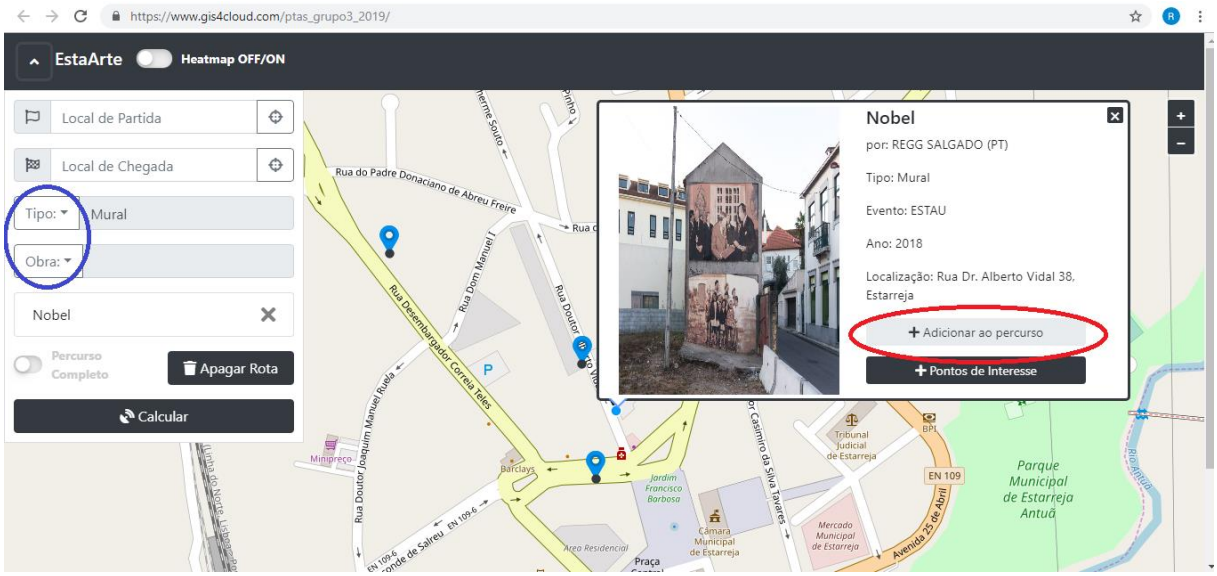


Figura 14- adicionar pontos à rota (vermelho-através do mapa, azul através dos filtros)

Pode verificar-se que à medida que os pontos são adicionados aparece na janela dos filtros uma caixa com o nove do ponto adicionado.

5.1.5 Implementação da Funcionalidade - (Selecionar pontos de partida e chegada)

Na aplicação são disponibilizadas duas formas para definir os pontos de partida e de chegada, sendo através da localização do dispositivo que se está a usar, ou através de um ponto definido pelo utilizador ao clicar no mapa. Se a partida e a chegada forem definidas através da localização do dispositivo basta carregar no botão do lado direito da caixa de texto que define o ponto de partida, podendo repetir-se o processo para o ponto de chegada. Caso o utilizador não queira começar a rota no sítio onde se encontra então tem que selecionar a caixa de texto onde está definida a partida e clicar no mapa no local de onde quer começar e repetindo o mesmo processo para o ponto de chegada. No mapa o ponto de partida está representado a verde e o de chegada a vermelho.

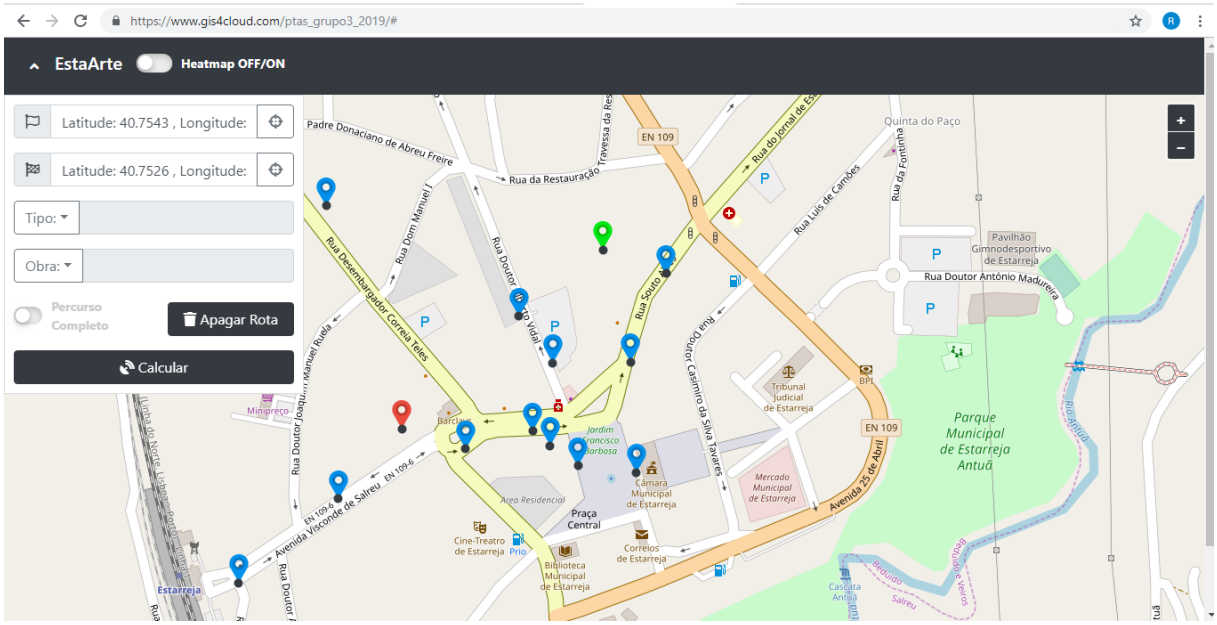


Figura 15- pontos de partida e chegada

5.1.6 Implementação da Funcionalidade – (Rotas)

Esta funcionalidade tem como objetivo ajudar os utilizadores que não conheçam a cidade a percorrer todos os PAU ou apenas os que selecionar de forma mais eficiente. Para esta funcionalidade é preciso ser fornecido um ponto de partida, um ponto de chegada e pelo menos um PAU a visitar.

5.1.6.1 Client-side

Após o click do botão “Calcular” a aplicação verifica se os pontos de partida e chegada se encontram preenchidos e se está pelo menos um PAU selecionado. Se alguma destas for falsa é lançado um alerta e o AJAX não será efetuado.

A aplicação verifica se o switch “Percurso Completo” está ON ou OFF. Se estiver ON executa um script de PHP via AJAX para realizar a rota para alguns pontos, senão executa outro script de PHP via AJAX para realizar a rota para todos os pontos.

O resultado do AJAX será um GeoJSON que será a source para uma layer da biblioteca OpenLayers que terá linhas que serão pretas com uma seta de direção no fim de cada uma.

O botão “Apagar Rota” remove a layer que foi referida no parágrafo anterior.

5.1.6.2 Server-side

Com o click no botão “Calcular” um script de PHP recebe, via JSON, três parâmetros: um array com a longitude e latitude do ponto de partida, um array com o id dos pontos a visitar e um array com a longitude e latitude do ponto de chegada.

Primeiramente para os pontos de partida e chegada o script encontra os nós da rede viária mais próximos dos mesmos.

De seguida o script verifica decide se é necessário executar o algoritmo do caixeiro viajante ou não. Esta decisão é feita com base no número de PAU passados. O algoritmo do caixeiro viajante tenta determinar a menor rota para percorrer uma série de locais (visitando uma única vez cada um deles), retornando ao local de origem.

Se o número de PAU passados for igual ou menor que um não é necessário executar o algoritmo do caixeiro viajante e apenas se executa o algoritmo do dijkstra que devolve o caminho mais curto entre dois pontos. Tem-se, no entanto, que iterar entre todos os pontos para encontrar o caminho mais curto entre cada um e no fim juntar esses mesmos caminhos num GeoJSON. O PostreSQL 9.5+ já fornece uma função para construir JSONs (jsonb_build_object), que foi usada nesta parte do projeto.

Se o número de PAU passados for maior que um então é executado o algoritmo do caixeiro viajante para ser fornecida a sequência de visita aos pontos e só depois é executado algoritmo do dijkstra da mesma maneira mencionada no parágrafo anterior. Como o algoritmo do caixeiro viajante retorna ao ponto de partida, algo não é pretendido então não é executado o algoritmo do dijkstra do penúltimo ponto para o último ponto anulando assim esse problema.

Com o click no botão “Calcular” e com a opção “Percurso Completo” selecionada um script de PHP recebe via JSON dois parâmetros: um array com a longitude e latitude do ponto de partida e um array com a longitude e latitude do ponto de chegada.

Primeiramente para os pontos de partida e chegada o script encontra os nós da rede viária mais próximos dos mesmos.

De seguida são obtidos todos os PAU ativos que estão guardados na base de dados.

Por último é executado o algoritmo do caixeiro viajante para obter a sequência de visita e o algoritmo do dijkstra pra obter o caminho mais curto entre os pontos da sequência de igual maneira supramencionada. O resultado é fornecido também em GeoJSON.

Figura 16 - Botões “Calcular”, “Apagar Rota” e “Percurso Completo” em modo OFF

Figura 17- Botões “Calcular”, “Apagar Rota” e “Percurso Completo” em modo ON

5.1.7 Implementação da Funcionalidade – (Pontos de Interesse)

Esta funcionalidade tem como objetivo dar a conhecer os utilizadores que não conheçam a cidade os PI a menos de um minuto do PAU selecionado.

5.1.7.1 Client-side

Após o click do botão “Pontos de Interesse” é executado um script de PHP via AJAX para a obtenção dos PI que estão a menos de um minuto do PAU. De momento este tempo é fixo, no entanto, numa iteração futura poderá ser pedido ao utilizador para inserir esse mesmo valor.

O resultado do AJAX será um GeoJSON que será a source para uma layer da biblioteca OpenLayers onde os pontos terão um marcador amarelo.

O GeoJSON também contém as informações relativas ao PI, como o nome, o horário e a imagem do mesmo. Essas informações são adicionadas ao popup do PI.

Ao sair do popup a layer dos PI é removida.

5.1.7.2 Server-side

Com o click no botão “Pontos de Interesse” um script de PHP recebe, via JSON, o id do PAU selecionado e o tempo máximo, em minutos, que o PI deve estar do PAU.

Primeiramente para o id do PAU o script encontra o nó da rede viária mais próximo do mesmo.

De seguida é realizada uma *query* que devolve o id e o nó de rede viária de todos os PI. É executado um algoritmo do *dijkstra* para todos os PI onde é devolvido o custo de ir do PAU selecionado até ao PI. Todos os PI em que o custo seja menor que aquele presente na key “tempo” do JSON passado para o script são adicionados a um *array*. Executa-se o algoritmo do dijkstra que devolve o caminho mais curto entre dois pontos. Tem-se, no entanto, que iterar entre todos os PI presentes no array para encontrar o caminho mais curto entre cada um e no fim juntar esses mesmos caminhos num GeoJSON. O PostreSQL 9.5+ já fornece uma função para construir JSONs (jsonb_build_object), que foi usada nesta parte do projeto.

Figura 18 - Pontos de Interesse

5.1.8 Implementação da Funcionalidade – (Heatmap)

Esta funcionalidade tem como objetivo dar a conhecer os utilizadores as zonas de maior concentração de PAU.

5.1.8.1 Client-side

O heatmap é um tipo de layer da biblioteca OpenLayers que precisa de uma source de pontos. Para este projeto essa source é a mesma que contém os PAU.

A aplicação verifica se o switch "Heatmap" está ON ou OFF. Se estiver ON cria a layer do tipo heatmap senão remove a mesma.

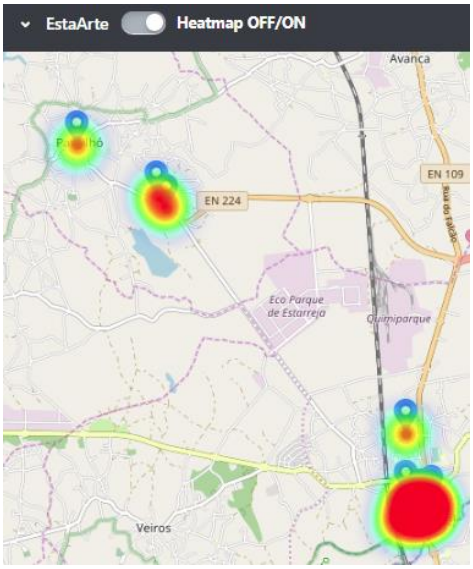


Figura 19 - Heatmap ON

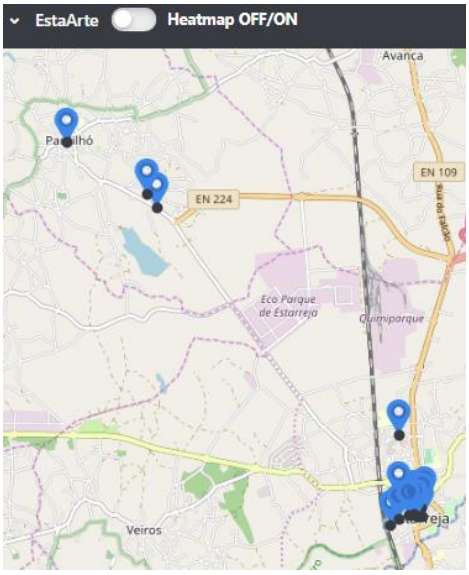


Figura 20 - Heatmap OFF

6 Análise de Resultados

Tendo em conta as tabelas em baixo dos requisitos funcionais, está análise face à sua concretização tendo em conta o projeto final. O símbolo “ ✓ “ corresponde a um requisito que foi atingido e por fim o símbolo “ × “ significa que o requisito não foi atingido.

Refª	**Requisito Funcional**	**Realizado**
RF1.01	Apresentação do PAU (Pontos de arte Urbana) no mapa	✓
RF1.02	Filtragens de pontos por tipo de arte	✓
RF1.03	Selecionar os pontos que eu quero visitar	✓
RF1.04	Rota pré-definida a passar por todos os pontos	✓
RF1.05	Mostrar pontos de interesse circundantes	✓
RF1.06	Selecionar pontos de partida e chegada através da localização GPS	✓
RF1.07	Selecionar pontos de partida e chegada através da localização opcional	✓
RF1.08	Apresentação da maior concentração de pontos no mapa	✓
RF1.09	Filtragem por tempo disponível	×
RF1.10	Como se vai realizar o percurso	×
RF1.11	Selecionar rotas pelo tempo disponível do utilizador	×

7 Conclusão

Com a realização deste projeto, foi possível pôr em prática os conhecimentos adquiridos durante a unidade curricular de Programação de Sistemas de Informação Geográfica, bem como adquirir novos conhecimentos através dos desafios ao qual o grupo se propôs. Em suma os requisitos funcionais definidos pelo grupo foram cumpridos, faltando os requisitos um requisito de média prioridade todos os de baixa prioridade, tendo por motivo em geral a falta de gestão de tempo, sendo estes realizados numa futura versão do projeto. Após a realização do projeto, foi possível ter a noção de que todo o processo de criação de uma aplicação S é um trabalho complexo, uma vez que se tem de ter em conta todos os pormenores e princípios a seguir e nunca esquecer que o fundamental é a forma como os utilizadores olham para a aplicação e ao mesmo tempo torná-la funcional e intuitiva. Em suma, pode-se concluir que foi um projeto interessante de desenvolver.

8 Referências Bibliográficas

OpenLayers. (s.d). Obtido de Openlayers: https://openlayers.org/

Stackoverflow. (s.d.). Obtido de stack'"overflow"': https://stackoverflow.com/

jQuery. (s.d). Obtido de jQuery: https://api.jquery.com/

PHP. (s.d). Obtido de php docs: https://www.php.net/docs.php

Caxeiro-Viajante. (s.d). Obtido de: https://pt.wikipedia.org/wiki/Problema_do_caixeiro-viajante

TSP. (s.d). Obtido de: https://docs.pgrouting.org/2.4/en/pgr_TSP.html

Dijkstra. (s.d). Obtido de: https://docs.pgrouting.org/2.4/en/pgr_dijkstra.html

JSON. (s.d). Obtido de Json: https://geojson.org/

Query geojson. (s.d). Obtido de: https://gis.stackexchange.com/questions/112057/sql-query-to-have-a-complete-ge-ojson-feature-from-postgis