Relatório Final

de

Laboratório de Projeto em Engenharia Informática

da

Licenciatura em Engenharia Informática

Nome do aluno	Número do aluno
André Filipe Correia Pereira	**74066**
Francisco Caldeira Alves Azevedo	**71506**

Título do trabalho

Gestão de equipamentos autárquicos

Nome do orientador

João Paulo Moura

Nome do(s) coorientador(es)

Nome do coorientador externo à UTAD

Prof. António Ribeiro (executivo da junta de freguesia de Vila Real)

INDICE

1. INTRODUÇÃO

A sociedade atual é amplamente influenciada pelo aspeto visual, onde somos facilmente atraídos por padrões e cores, o cérebro humano tem uma grande facilidade em decifrar a informação que lhe é transmitida nessa forma. A comunicação visual é altamente eficiente, facilitando a interpretação dos dados e a identificação de tendências. Desta forma, estas ferramentas são de grande importância no nosso cotidiano. O melhor gráfico para interpretar informação geográfica é um mapa. O objetivo deste projeto é publicar na Web um conjunto de dados, sob a forma de mapas intuitivos e interativos, referentes aos equipamentos públicos da Freguesia de Vila Real, representando-os geograficamente e de uma forma interativa recorrendo a soluções open-source baseados em GEOJSON.

O trabalho foi estruturado em 3 fases (Figura 1): 1. Recolha dos dados; 2. Processamento e armazenamento da informação resultante; 3. Desenvolvimento de um Protótipo Web.

Figura 1 - Processo de Criação de um Mapa Interativo

01. Recolha dos dados

Para a recolha de informação foi usado o QGIS (Quantum Geographic Information System), um software open-source que é um GIS (Geografic Informatic Software). Este software permite ao utilizador visualizar, editar e analisar dados georreferenciados, e através desses mesmos criar camadas com diferentes formas (pontos, linhas, polígonos). Os dados geográficos foram recolhidos no local e os restantes dados foram fornecidos pela Junta de Freguesia.

02. Processamento e armazenamento da informação resultante

A informação foi processada e armazenada em GEOJSON que é um formato de dados usado para representar dados geográficos e os seus respetivos atributos usando coordenadas geográficas. O formato de dados GEOJSON é baseado no JSON (Java Script Object Notation). Os dados armazenados podem ser pontos, linhas, polígonos ou um misto de todos.

Definimos várias camadas:

- Camada base (mapa OpenStreetMap).
- Camada do limite administrativo de Vila Real.
- Camada para as Escolas.
- Camada para os Campos Polidesportivos.
- Camada para os Parques Infantis.
- Camada para os Edifícios Administrativos da Junta de Freguesia.

03. Desenvolvimento de um Protótipo Web

O protótipo, uma página web baseada num mapa interativo foi desenvolvida usando a biblioteca Leaflet , uma biblioteca JavaScript usada para criar mapas web. É muito prático para a representação de mapas interativos, dado que disponibiliza uma larga variedade de plugins que permitem ao programador tratar os dados da melhor forma possível, personalizar camadas e adicionar marcadores e pop-ups.

2. REVISÃO BIBLIOGRÁFICA

QGIS - software livre com código-fonte aberto, multiplataforma de sistema de informação geográfica (SIG) que permite a visualização, edição e análise de dados georreferenciados. Similar a outros softwares de SIG, o QGIS permite ao utilizador de analisar e editar informações espaciais, além de criar mapas com várias camadas usando diferentes projeções. Os dados podem ser do tipo: Vetorial com pontos, linhas ou polígonos ou Raster sendo uma imagem georreferenciada.

GEOJSON - O GeoJSON é um formato open-source projetado para representar recursos geográficos simples, juntamente com seus atributos não espaciais. É baseado no formato interoperável JSON (JavaScript Object Notation). Os recursos incluem pontos (e.g.: endereços e locais), sequências de linhas (e.g.: ruas, rodovias e limites), polígonos (e.g.: países, províncias, terrenos) e coleções com várias partes desses tipos.

WebSIGs - sistema de informação geográfica para a Web, compreendendo pelo menos um servidor de dados geográficos e um cliente web. Na sua forma mais simples, o WebSIG pode ser definido como qualquer SIG que use a tecnologia da Web para comunicar entre um servidor e um cliente (FU & SUN, 2010).

Leaflet - O Leaflet é uma biblioteca JavaScript de código aberto amplamente usada para criar aplicativos de mapas na Web. Lançado pela primeira vez em 2011, suporta a maioria das plataformas móveis e de desktop, suportando HTML5 e CSS3. O Leaflet permite que programadores sem experiência em SIG consigam publicar mapas na Web, com sobreposições de variados dados, controlados pelo utilizador.

3. MODELO PROPOSTO

O objetivo deste projeto é o desenvolvimento de um mapa interativo em que o utilizador pudesse observar e selecionar as camadas a visualizar, que tivesse a opção de manipular o mapa conforme a sua preferência, controlando o zoom, selecionando os objetos de interesse no mapa de forma que este apresentasse no ecrã apenas a informação com relevância. O mapa deve apresentar vários painéis com opções de controlo do mapa: ativar/desativar camadas, selecionar o tipo de mapa, fazer zoom, obter coordenadas da posição do rato e selecionar objetos.

Para isso, realizamos uma análise de requisitos funcionais e não funcionais. Os Requisitos Funcionais são definidos como as funcionalidades a desenvolver, focando-se nos vários atores do sistema. Já os Requisitos Não Funcionais, definem restrições aplicadas ao sistema. Podem ser relativamente à interface apresentada ao utilizador, software, hardware, restrições de desempenho, velocidade, capacidade de armazenamento, restrições sobre a portabilidade, a facilidade de manutenção, e ainda restrições sobre outras normas existentes e limites de recursos externos.

Neste projeto, são definidos como requisitos funcionais:
• **RF1** – Selecionar camadas a visualizar;
• **RF2** – Efetuar *zoom;*
• **RF3** – Efetuar *pan;*
• **RF4 –** Atualizar mapa (em função do zoom/área/seleção);
• **RF5 –** Consultar ficha técnica;
• **RF6 –** Selecionar objeto no mapa;
• **RF7 –** Apresentar informação detalhada referente ao objeto selecionado;
• **RF8** – Obter coordenadas da posição atual do mapa;
• **RF9 –** Reinicializar mapa;
• **RF10 –** Apresentar *tooltip;*

Já os requisitos não funcionais são definidos como:
• **RNF1 –** Desenvolver o projeto em Leaflet;
• **RNF2 –** Visualizar os resultados numa página web (mapas temáticos);
• **RNF3** – Apresentar os limites administrativos da freguesia de Vila Real;
• **RNF4** – Apresentar obrigatoriamente os parques infantis, campos desportivos, espaços da junta e escolas;
• **RNF5** – Apresentar as respetivas informações de cada ponto da layer quando for premido o rato em cada ícone;
• **RNF6** – Definir regras de ordenação de apresentação de mapas;
• **RNF7** – Mapa de abertura da aplicação centrado em Vila Real.

Realizamos também Diagrama de Casos de Uso como ver na presente Figura 2:

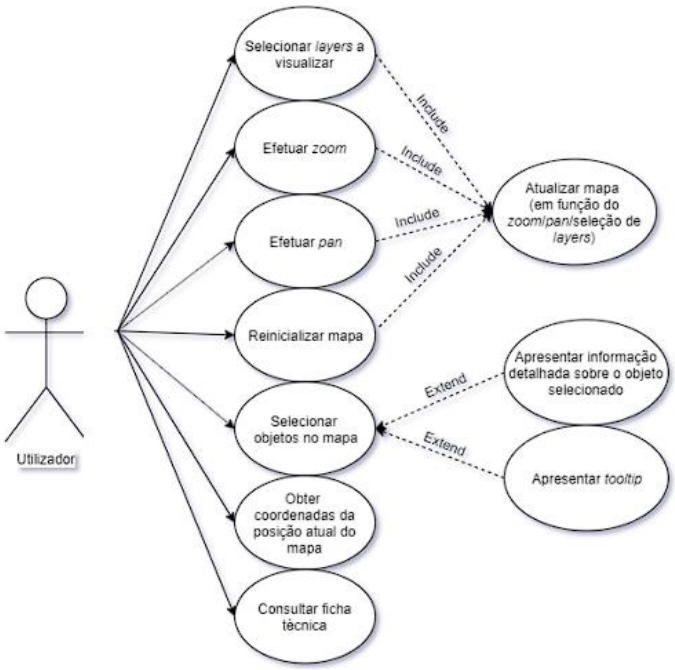


Figura 2 - Diagramas de Casos de Uso

O Diagrama de Atividades refere-se às funcionalidades do sistema, indicando quem é responsável por cada uma, estabelecendo as relações entre os componentes. Sendo assim, a interação entre Utilizadores e Sistema pode ser visualizada através da Figura 3.

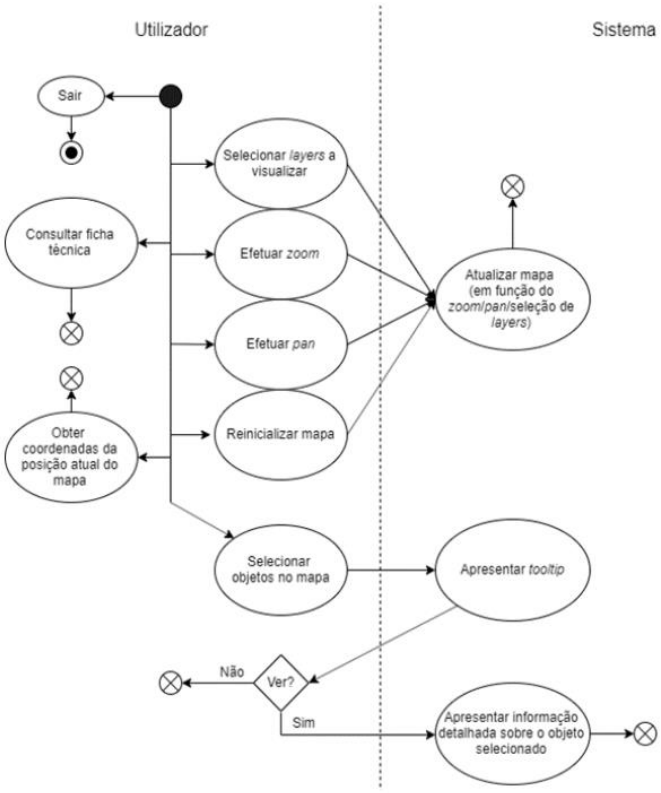


Figura 3 - Diagrama de Atividades

4. IMPLEMENTAÇÃO DO PROTÓTIPO

As camadas temáticas, assim como o seu tipo de representação espacial, podem ser observadas na Tabela 1.

Tabela 1 - Tabela das camadas

CAMADAS	DADOS
Limite administrativo (freguesia de Vila Real)	polígonos
Parques Infantis	pontos
Escolas	pontos
Campos Polidesportivos	pontos
Espaços da Junta	pontos

As estruturas de dados de cada camada estão explícitas nas Tabelas 2 a 5:

Tabela 2 - Estrutura da camada Escolas

Id	Nome	Tipo	Nome do tipo	Comprimento
0	id	Integer (64 bit)	Integer64	10
1	nome	Texto (cadeia de caracteres)	String	80
2	DataManun	Data	Date	10
3	Limpeza	Data	Date	10
4	Inspecoes	Data	Date	10
5	Equipament	Texto (cadeia de caracteres)	String	254

Tabela 2 - Estrutura da camada Espaços da Junta VR

Id	Nome	Tipo	Nome do tipo	Comprimento
0	id	Integer (64 bit)	Integer64	10
1	Nome	Texto (cadeia de caracteres)	String	80
2	Horario	Texto (cadeia de caracteres)	String	20
3	Telefone	Integer (32 bit)	Integer	9
4	email	Texto (cadeia de caracteres)	String	30

Tabela 4 - Estrutura da camada Parques Infantis

Id	Nome	Tipo	Nome do tipo	Comprimento
0	id	Integer (64 bit)	Integer64	10
1	Nome	Texto (cadeia de caracteres)	String	80
2	Instruçõ	Texto (cadeia de caracteres)	String	100
3	Limiteldad	Integer (64 bit)	Integer64	10
4	DataM	Data	Date	10
5	Limpeza	Data	Date	10
6	nApólice	Integer (64 bit)	Integer64	11
7	Inspeçõe	Data	Date	10
8	RelInsp	Texto (cadeia de caracteres)	String	50
9	RepConf	Data	Date	10
10	TelEmerg	Integer (32 bit)	Integer	9
11	Fiscaliza	Data	Date	10
12	MedidasP	Texto (cadeia de caracteres)	String	20
13	Vegetacao	Texto (cadeia de caracteres)	String	10
14	Iluminacao	Integer (64 bit)	Integer64	10
15	ListaEquip	Texto (cadeia de caracteres)	String	254

Tabela 5 - Estrutura da camada Campos Polidesportivos

Id	Nome	Tipo	Nome do tipo	Comprimento
0	id	Integer (64 bit)	Integer64	10
1	Nome	Texto (cadeia de caracteres)	String	80
2	Instrucoes	Texto (cadeia de caracteres)	String	254
3	LimitIdade	Decimal (double)	Real	20
4	DataManunt	Data	Date	10
5	Limpeza	Data	Date	10
6	nApolice	Decimal (double)	Real	20
7	Inspecoes	Data	Date	10
8	RelatorioI	Texto (cadeia de caracteres)	String	20
9	ReposConf	Data	Date	10
10	ContactoEm	Integer (32 bit)	Integer	9
11	Fiscaliza	Data	Date	10
12	MedidasPre	Integer (32 bit)	Integer	1
13	Vegetacao	Integer (32 bit)	Integer	1
14	Iluminacao	Integer (32 bit)	Integer	1
15	ListaEquip	Texto (cadeia de caracteres)	String	254

4.1 Implementação no QGIS:

No nosso projeto no GGIS , começamos por implementar por fases:

Figura 4 - Camada OpenStreetMap

1. Adicionar uma camada com os mapas do Open Street Map, através de una ligação online e de acesso gratuito.

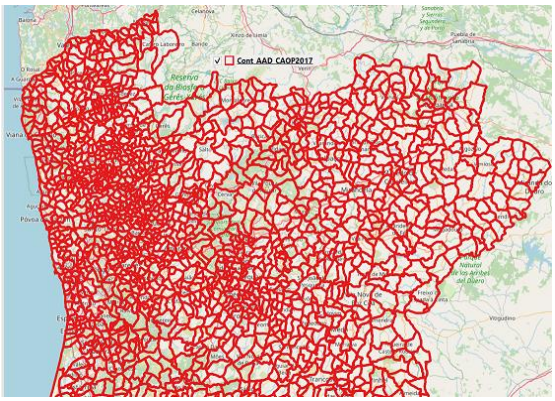


Figura 5 - Camada geral de limites de todas as freguesias

2. Importar os limites administrativos de todas Freguesias, obtidos através do site oficial do governo, dados.gov.pt, que fornece um ficheiro *shapefile* constituído por polígonos que representam as Freguesias.

Figura 6 - Criação dos limites da freguesia de Vila Real

3. Adicionar uma camada com os limites da Freguesia de Vila Real, selecionada a partir do ficheiro fornecido pelo site dado.gov.pt (camada denominada F_VR).

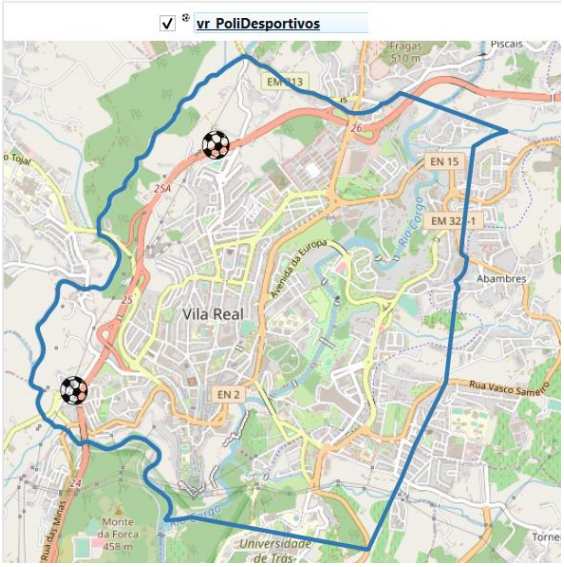


Figura 7 - Criação da camada dos Polidesportivos

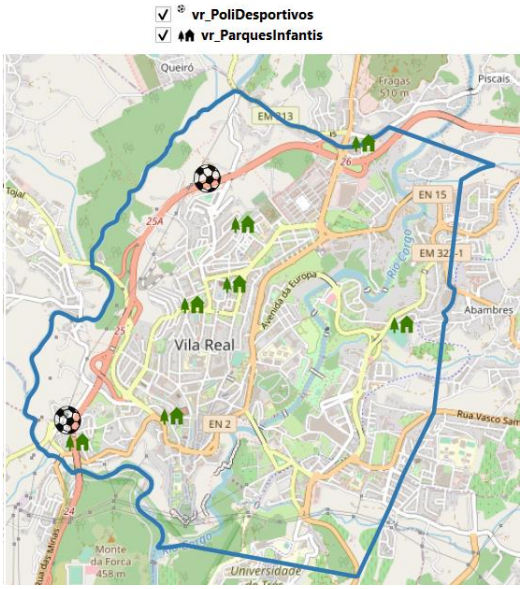


Figura 8 - Criação da camada dos Parques Infantis

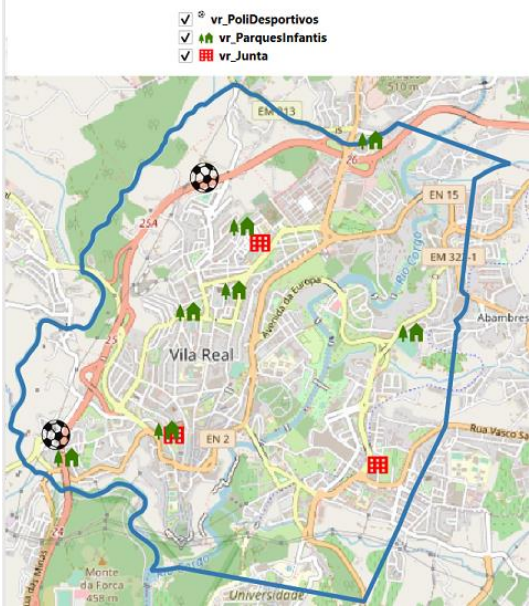
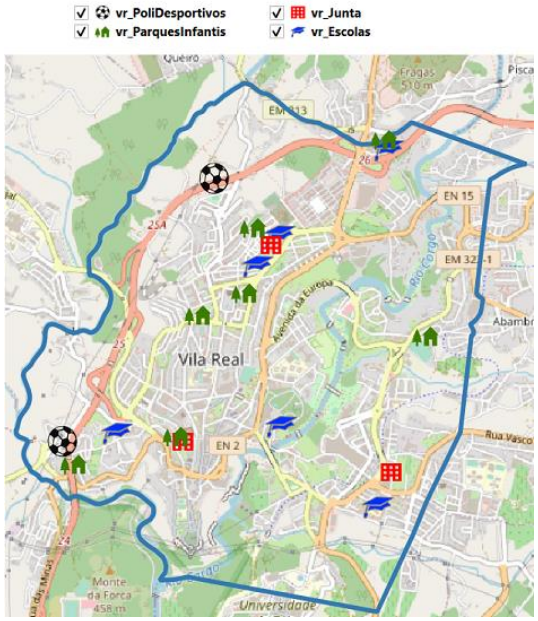


Figura 9 - Criação da camada da Junta

4. Criar a camada para o mapa temático dos Campos Polidesportivos (camada denominada vr_PoliDesportivos) , onde a inserção dos dados é feita com a estrutura definida na Tabela 5 onde , serão inseridos no mapa os pontos referentes aos polidesportivos existentes, e neste caso, para facilitar a diferenciação no QGIS , é definido o ícone gráfico 'bola de futebol'.

5. Criar a camada para o mapa temático dos Parques Infantis (camada denominada vr_ParquesInfantis) , onde a inserção dos dados é feita com a estrutura definida na Tabela 4 onde , serão inseridos no mapa os pontos referentes aos parques infantis existentes, e neste caso, para facilitar a diferenciação no QGIS , é definido o ícone gráfico de uma casa e uma árvore, com a cor verde.

6. Criar a camada para o mapa temático dos espaços de atendimento da Junta de Freguesia de Vila Real (camada denominada vr_Junta) , onde a inserção dos dados é feita com a estrutura definida na Tabela 3 onde , serão inseridos no mapa os pontos referentes aos locais de atendimento existentes e é definido o ícone gráfico de um edifício , com a cor vermelha.



7. Criar a camada para o mapa temático das Escolas (camada denominada vr_Escolas) , onde a inserção dos dados é feita com a estrutura definida na Tabela 2 onde , serão inseridos no mapa os pontos referentes às escolas existentes, e é definido o ícone gráfico de uma cartola, simbolizando a escola, com a cor azul.

Figura 10 - Criação da camada das Escolas

Após conclusão dos passos no software QGIS, é exportada cada camada individualmente para um ficheiro GEOJSON.

4.2 Implementação do protótipo Web:

Para a implementação deste projeto, irá ser usada a biblioteca de JavaScript Leaflet. Esta é dedicada ao desenvolvimento de mapas interativos, onde é possível apresentar todo o tipo de dados de variadas origens.

A página web é codificada em HTML . O JavaScript irá lidar com toda a componente de funcionalidades da página. O CSS lida com a componente visual e também estrutural da página.

Para se poder testar a página em condições adequadas, é necessário visualizar a página num servidor web. Utilizamos o Visual Studio Code que nos permite visualizar a mesma.

Nesta secção, apresenta-se algumas partes do código, HTML, CSS e JavaScript, acompanhadas de uma breve explicação do seu efeito no produto final.

A estrutura de ficheiros do projeto é a seguinte:

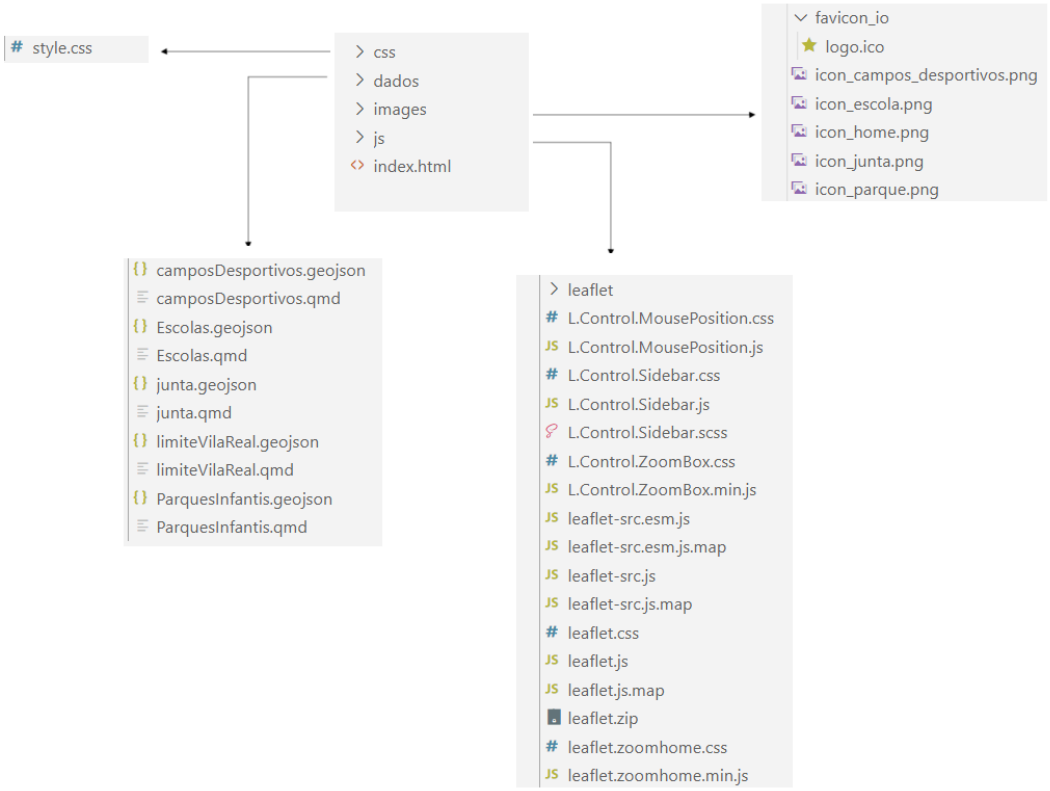


Figura 11 - Estrutura dos ficheiros do projeto

dados – Pasta que possui os dados referentes à informação apresentada no sistema. Neste caso trata-se de 5 ficheiros, em formato GEOJSON e suas extensões do tipo qmd, criadas automaticamente na exportação.

js – Pasta que possui a biblioteca de JavaScript Leaflet, tal como é descarregada do website, e alguns plugins destinados a determinadas funcionalidades, nomeadamente:

Zoomhome – *Plugin* para botões de zoom e para reinicializar o mapa

MousePosition – *Plugin* para captura das coordenadas do rato

ZoomBox – *Plugin* para desenho de uma caixa, onde depois se efetua zoom

index.html – Ficheiro HTML, que será visualizado num browser, e possui o código HTML e JavaScript necessário. Serve como agregador de todo o projeto, pois todos os ficheiros usados no sistema devem estar referenciados no seu cabeçalho. Este ficheiro define a estrutura e o conteúdo da página.

css – Pasta que contém o ficheiro style.css, que possui o código necessário para estilizar o projeto, permitindo definir o aspeto e a formatação do conteúdo armazenado no ficheiro HTML. Apesar de este código poder ser inserido também no ficheiro index.html, é boa prática manter esta parte em separado, pois caso seja necessário alterar algum detalhe visual, ele é aplicado em todos os outros ficheiros, não sendo necessário repetir as alterações uma a uma.

Images – Nesta pasta são incluídas imagens como imagens para cada ícone de diferentes tipos (escolas, parques infantis, etc) bem como o ícone da página web.

Já em termos de projeto, a seguir será apresentado um conjunto de imagens, referentes ao ficheiro HTML, ao ficheiro CSS e às respetivas funções no WebSIG.

Iremos analisar o ficheiro 'Index.html' , que contém praticamente todo o código usado no projeto, com a exceção do ficheiro 'Style.css'. Na Figura 12, pode-se visualizar o cabeçalho do documento, que determina o tipo de ficheiro (HTML), determina os caracteres a usar (UTF-8), e invoca variados ficheiros, necessários para o sistema. Esta parte é muito importante, pois serve como agregador de todos os ficheiros do projeto. Se um ficheiro não for incluído nesta parte, não será lido pelo browser. Já no CSS, apresenta-se o código integral, pois apenas é necessário para uma apresentação adequada do mapa. Todos os restantes aspetos visuais são controlados pelos respetivos *plugins* adicionados.

```
<!DOCTYPE html>
<html>

<head>
    <title>JUNTA DE FREGUESIA VILA REAL</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="css/style.css" />
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/leaflet@1.7.1/dist/leaflet.css" />
    <link rel="shortcut icon" href="images/favicon_io/logo.ico" type="image/x-icon">
    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
    <script src="/dados/Escolas.geojson"></script>
    <script src="/dados/limiteVilaReal.geojson"></script>
    <script src="/dados/ParquesInfantis.geojson"></script>
    <script src="/dados/junta.geojson"></script>
    <script src="/dados/camposDesportivos.geojson"></script>
</head>
```

Figura 12 - Cabeçalho do Ficheiro HTML 'Index.html'

Na Figura 13, o código cria um mapa interativo utilizando a biblioteca Leaflet.js e exibe um mapa base do OpenStreetMap. Além disso, o código também inclui controlos e camadas adicionais no mapa. O código HTML define a estrutura básica da página, com um elemento ** ** com ID "map" para exibir o mapa, elementos **<input>** e **<label>** para os controles de exibição no mapa, as respetivas checkbox para podermos selecionar as camadas que queremos ver. O mapa é exibido no elemento HTML com ID "map" e é inicialmente centrado nas coordenadas geográficas de Vila Real, com um nível de zoom de 14. Em seguida, é definida a URL do provedor de mapas OpenStreetMap, que será usada para carregar o mapa. Uma atribuição é definida para os dados do mapa, incluindo um link para o OpenStreetMap e inserimos aqui a ficha técnica. Um objeto de camada tile layer é criado usando a URL do OpenStreetMap e a atribuição definida anteriormente, adicionando-a ao mapa usando o método **addTo**, fazendo com que o mapa seja exibido com o mapa base do OpenStreetMap.
O código, até este ponto, configura o mapa com o mapa base do OpenStreetMap e define a visualização inicial (Figura 14).

```
<body>
    <div id="map"></div>
    <div id="controls1" class="label2">
        <button id="satellite-btn">Satélite</button>
        <button id="osm-btn">Mapa</button>
    </div>
    <div id="controls">
        <table>
            <tr>
                <td>
                    <input type="checkbox" id="toggle-limit" checked>
                </td>
                <td>
                    <label for="toggle-limit">Limite da freguesia</label>
                </td>
            </tr>
            <tr>
                <td>
                    <input type="checkbox" id="toggle-escolas">
                </td>
                <td>
                    <label for="toggle-escolas">Escolas</label>
                </td>
            </tr>
            <tr>
                <td>
                    <input type="checkbox" id="toggle-parques">
                </td>
                <td>
                    <label for="toggle-parques">Parques infantis</label>
                </td>
            </tr>
        <tr>
            <td>
                <input type="checkbox" id="toggle-campos">
            </td>
            <td>
                <label for="toggle-campos">Campos Desportivos</label>
            </td>
        </tr>
        <tr>
            <td>
                <input type="checkbox" id="toggle-junta" checked>
            </td>
            <td>
                <label for="toggle-junta">Locais de Atendimento</label>
            </td>
        </tr>
    </table>
</div>
<button id="home-btn"></button>
<script src="https://cdn.jsdelivr.net/npm/leaflet@1.7.1/dist/leaflet.js"></script>
<script src="js/leaflet/leaflet.js"></script>
<script>
    // Cria um objeto mapa usando a biblioteca Leaflet.js e
    //define a visualização inicial com coordenadas geográficas de Vila Real e o nível de zoom 14.
    var map = L.map('map').setView([41.2995, -7.7425], 14);

    // Define a URL para o provedor de mapas OpenStreetMap.
    var osmUrl = 'https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png';

     // Define a atribuição dos dados do mapa, incluindo um link para o OpenStreetMap.
     var osmAttrib = 'Ficha Técnica: <br> André Pereira - Desenvolvedor <br> Francisco Azevedo - Desenvolvedor <br> Map data © <a href="

    // Cria um objeto tile layer usando a URL do OpenStreetMap e a atribuição definida acima.

    var osm = new L.TileLayer(osmUrl, { attribution: osmAttrib });
    // Adiciona a camada tile layer ao mapa.
    osm.addTo(map);
```

Figura 13 - Configuração inicial do mapa

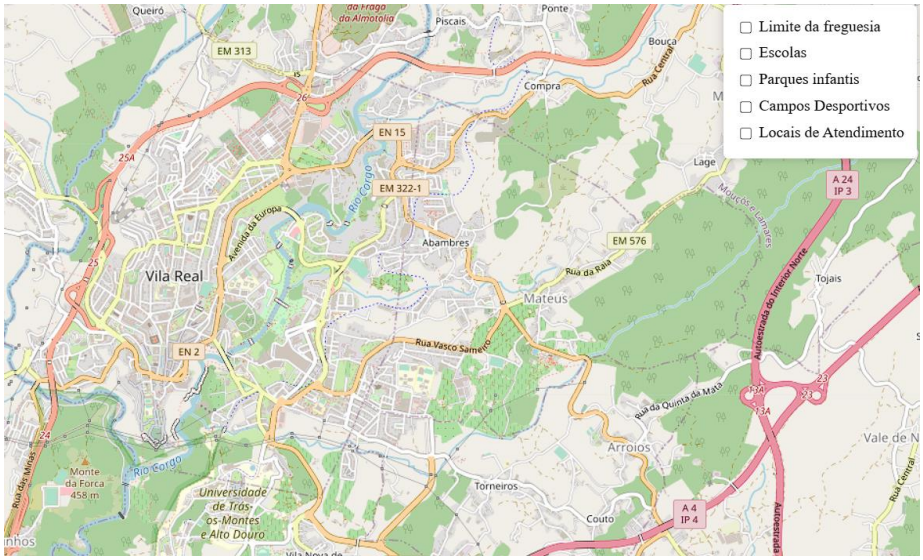


Figura 14 - Visualização Inicial

Posteriormente como se pode observar na Figura 15, definimos quatro objetos de ícone diferentes usando a biblioteca Leaflet.js. Cada objeto de ícone é configurado com propriedades específicas: url de imagem, tamanho e pontos de ancoragem do ícone para serem usados posteriormente na criação de marcadores personalizados no mapa. Na Figura 16, podemos ver os respetivos ícones e estão por esta ordem (da esquerda para a direita): 1. Polidesportivos; 2. Escolas; 3. Espaços da Junta; 4. Parques Infantis;

```
// Criação de quatro ícones personalizado para as escolas , parques, campos desportivos e espaços da junta
var escolaIcon = L.icon({
    iconUrl: 'images/icon_escola.png',
    iconSize: [40, 40],
    iconAnchor: [12, 41],
    popupAnchor: [0, -41]
});

var parqueIcon = L.icon({
    iconUrl: 'images/icon_parque.png',
    iconSize: [40, 40],
    iconAnchor: [12, 41],
    popupAnchor: [0, -41]
});
var camposDesportivosIcon = L.icon({
    iconUrl: 'images/icon_campos_desportivos.png',
    iconSize: [40, 40],
    iconAnchor: [12, 41],
    popupAnchor: [0, -41]
});
var juntaIcon = L.icon({
    iconUrl: 'images/icon_junta.png',
    iconSize: [40, 40],
    iconAnchor: [12, 41],
    popupAnchor: [0, -41]
});
```

Figura 15 - Criação dos ícones personalizados para as 4 camadas

Figura 16 - Ícones respetivos

O código da Figura 17, cria várias camadas no mapa, usando a biblioteca Leaflet.js e dados GeoJSON. Cada camada representa um tipo diferente de informação geográfica e possui marcadores personalizados com ícones específicos. O código segue um padrão para cada camada: É definida uma variável para a camada de dados geográficos, inicialmente vazia (**null**).
A função **pointToLayer** é utilizada para criar um marcador personalizado para cada ponto do GeoJSON, especificando o ícone a ser usado. A função **onEachFeature** é utilizada para criar uma janela de informação (popup) para cada marcador, exibindo informações relevantes, neste caso o nome (Figura 18) . Os dados GeoJSON são carregados a partir de um arquivo usando **$.getJSON** e adicionados à camada usando **addData**. A camada é adicionada ao mapa usando o método **addTo**. Esse processo é repetido para cada camada, resultando em um mapa interativo com várias camadas de dados geográficos, onde cada camada exibe marcadores personalizados com ícones específicos e informações relacionadas. Os dados são carregados a partir de arquivos GeoJSON e exibidos no mapa com base em suas coordenadas geográficas.

```
var escolas = L.geoJSON(null, {
    pointToLayer: function (feature, latlng) {
        return L.marker(latlng, { icon: escolaIcon });
    },
    onEachFeature: function (feature, layer) {
        layer.bindPopup("<strong>Nome: </strong>" + feature.properties.nome);
    }
});

// carrega o arquivo GeoJSON escolas adiciona a camada ao mapa
$.getJSON('dados/Escolas.geojson', function (data) {
    escolas.addData(data);
});
escolas.addTo(map);

//-------------------------------------------------------------------------------
var parquesInfantis = L.geoJSON(null, {
    pointToLayer: function (feature, latlng) {
        return L.marker(latlng, { icon: parqueIcon });
    },
    onEachFeature: function (feature, layer) {
        layer.bindPopup("<strong>Nome: </strong>" + feature.properties.Nome);
    }
});
parquesInfantis.addTo(map);
$.getJSON('dados/ParquesInfantis.geojson', function (data) {
    parquesInfantis.addData(data);
});

//-------------------------------------------------------------------------------
var limiteVilaReal = L.geoJSON(null, {
    style: function (feature) {
        return { color: "#0000ff", weight: 2 };
    }
});
limiteVilaReal.addTo(map);
$.getJSON('dados/limiteVilaReal.geojson', function (data) {
    limiteVilaReal.addData(data);

});
//-------------------------------------------------------------------------------
var camposDesportivos = L.geoJSON(null, {
    pointToLayer: function (feature, latlng) {
        return L.marker(latlng, { icon: camposDesportivosIcon });
    },
    onEachFeature: function (feature, layer) {
        layer.bindPopup("<strong>Nome: </strong>" + feature.properties.Nome);
    }
});
$.getJSON('dados/camposDesportivos.geojson', function (data) {
    camposDesportivos.addData(data);
});
camposDesportivos.addTo(map);
//-------------------------------------------------------------------------------
var junta = L.geoJSON(null, {
    pointToLayer: function (feature, latlng) {
        return L.marker(latlng, { icon: juntaIcon });
    },
    onEachFeature: function (feature, layer) {
        layer.bindPopup("<strong>Nome: </strong>" + feature.properties.Nome);
    }
});
$.getJSON('dados/junta.geojson', function (data) {
    junta.addData(data);
});
junta.addTo(map);
```

Figura 17 - Adição das camadas e pop-us

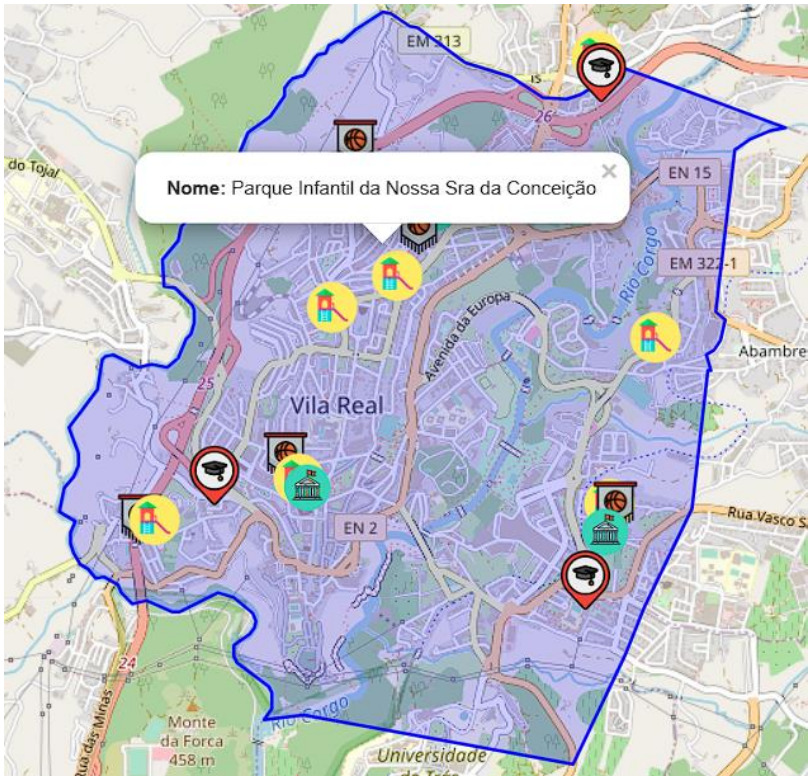


Figura 18 - View já com todas as camadas e pop-up após o clique num dos ícones

Para completar o código, implementamos as funcionalidades finais relacionadas com o controlo de camadas no mapa, interação com botões e exibição de coordenadas (Figura 19).

A primeira parte do código cria um objeto **overlays** que mapeia o nome de cada camada com sua respetiva variável. Esse objeto será usado posteriormente para adicionar um controlo de camadas ao mapa. Em seguida, é definida a função **atualizarVisibilidadeCamadas()**, que verifica o estado das checkboxes e adiciona ou remove as camadas correspondentes ao mapa. Essa função é chamada sempre que houver uma mudança nas checkboxes. As próximas linhas de código associam a função **atualizarVisibilidadeCamadas()** aos eventos **change** das checkboxes. Isso garante que a visibilidade das camadas seja atualizada quando as checkboxes forem marcadas ou desmarcadas. Depois disso, as camadas são definidas como invisíveis no início, removendo-as do mapa, exceto a camada "Junta" e "Limite da freguesia", que são adicionadas ao mapa sempre por defeito.

Em seguida, são definidas duas variáveis para os layers dos mapas de satélite e do OpenStreetMap. Os botões **#satellite-btn** e **#osm-btn** são associados a eventos de clique. Quando o botão de satélite é clicado, a camada do OpenStreetMap é removida e a camada de satélite é adicionada ao mapa (Figura 20), e vice-versa quando o botão do OpenStreetMap é clicado.

O botão **#home-btn** define uma função para redefinir a visualização do mapa para as coordenadas geográficas iniciais e com o zoom inicial. Em seguida, é definido um controle de coordenadas que exibe as coordenadas do rato no mapa. Esse controle é adicionado ao canto inferior esquerdo do mapa. Por fim, a tag **<script>** é encerrada e o código HTML é finalizado.

```
// adiciona controlo de camadas ao mapa
var overlays = {
    "Limite da freguesia": limiteVilaReal,
    "Escolas": escolas,
    "Parques infantis": parquesInfantis,
    "Campos Desportivos": camposDesportivos,
    "Junta": junta
};
L.control.layers(null, overlays).addTo(map);

// Função para definir a visibilidade das camadas de acordo com a seleção da checkbox
function atualizarVisibilidadeCamadas() {
    if ($('#toggle-limit').is(':checked')) {
        limiteVilaReal.addTo(map);
    } else {
        limiteVilaReal.removeFrom(map);
    }

    if ($('#toggle-escolas').is(':checked')) {
        escolas.addTo(map);
    } else {
        escolas.removeFrom(map);
    }

    if ($('#toggle-parques').is(':checked')) {
        parquesInfantis.addTo(map);
    } else {
        parquesInfantis.removeFrom(map);
    }

    if ($('#toggle-campos').is(':checked')) {
        camposDesportivos.addTo(map);
    } else {
        camposDesportivos.removeFrom(map);
    }
    if ($('#toggle-junta').is(':checked')) {
        junta.addTo(map);
    } else {
        junta.removeFrom(map);
    }
}

// Atualiza a visibilidade das camadas quando as checkboxes são alteradas
$('#toggle-limit').on('change', atualizarVisibilidadeCamadas);
$('#toggle-escolas').on('change', atualizarVisibilidadeCamadas);
$('#toggle-parques').on('change', atualizarVisibilidadeCamadas);
$('#toggle-campos').on('change', atualizarVisibilidadeCamadas);
$('#toggle-junta').on('change', atualizarVisibilidadeCamadas);

// Define as camadas como invisíveis no início
escolas.removeFrom(map);
parquesInfantis.removeFrom(map);
camposDesportivos.removeFrom(map);
junta.addTo(map);
limiteVilaReal.addTo(map);

var satelliteLayer =
L.tileLayer('https://server.arcgisonline.com/ArcGIS/rest/services/World_Imagery/MapServer/tile/{z}/{y}/{x}', {
    maxZoom: 18,
    attribution: 'Esri, DigitalGlobe, GeoEye, Earthstar Geographics, CNES/Airbus DS, USDA, USGS, AeroGRID, IGN, an
});

$('#satellite-btn').click(function () {
    map.removeLayer(osm);
    map.addLayer(satelliteLayer);
});
$('#osm-btn').click(function () {
    map.removeLayer(satelliteLayer);
    map.addLayer(osm);
});
$('#home-btn').click(function () {
    map.setView([41.2995, -7.7425], 14);

});

L.Control.Coordinates = L.Control.extend({
    options: {
        position: 'bottomleft',
        // Aqui está o formato padrão, que exibe as coordenadas no formato [latitude, longitude]
        coordinateFormat: function (latLng) {
            return '[' + latLng.lat.toFixed(4) + ', ' + latLng.lng.toFixed(4) + ']';
        },
        coordinateText: 'Coordenadas:',
    },
    onAdd: function (map) {
        this._container = L.DomUtil.create('div', 'leaflet-control-coordinates');
        L.DomEvent.disableClickPropagation(this._container);
        map.on('mousemove', this._update, this);
        this._container.innerHTML = this.options.coordinateText;
        return this._container;
    },
    onRemove: function (map) {
        map.off('mousemove', this._update, this);
    },

    _update: function (e) {
        this._container.innerHTML = this.options.coordinateText + ' ' + this.options.coordinateFormat(e.latlng);
    }
});
L.control.coordinates = function (options) {
    return new L.Control.Coordinates(options);
};
L.control.coordinates({ position: 'bottomleft' }).addTo(map);

    </script>
</body>
</html>
```

Figura 19 - Implementação das funcionalidades finais

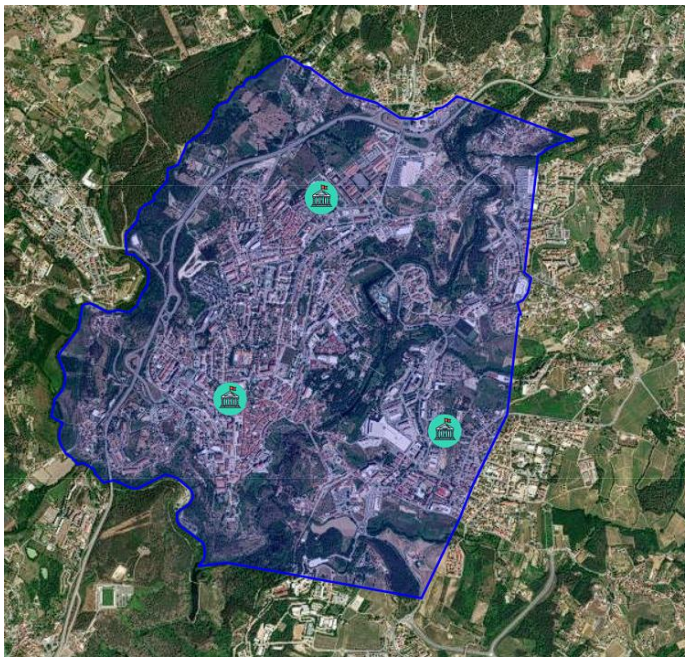


Figura 20 - Vista de Satélite

Por fim, o ficheiro 'Style.css' é simples e define as propriedades de algumas *divs* (Figura 21). Por fim, podemos ver o produto final, nas Figuras 22 e 23.

```css
html,
body,
#map {
  height: 100%;
  margin: 0;
  padding: 0;
}
#controls {
  position: absolute;
  top: 55px;
  right: 10px;
  z-index: 1000;
  background-color: #fff;
  padding: 10px;
  border-radius: 5px;
  box-shadow: 0 0 10px rgba(0, 0, 0, 0.2);
}
#controls label {
  display: block;
  margin-bottom: 5px;
}
#controls1.label2 {
  position: absolute;
  top: 10px;
  right: 10px;
  z-index: 1000;
  background-color: #fff;
  padding: 10px;
  border-radius: 5px;
  box-shadow: 0 0 10px rgba(0, 0, 0, 0.2);
}
#controls1.label2 label {
  display: block;
  margin-bottom: 5px;
}
.leaflet-control-coordinates {
  background-color: #fff;
  padding: 5px;
  border-radius: 5px;
}
#home-btn {
  background-image: url("../images/icon_home.png");
  background-color: #fff;
  background-size: 70%;
  background-position: center;
  background-repeat: no-repeat;
  width: 33px;
  height: 33px;
  position: fixed;
  top: 70px;
  left: 10px;
  z-index: 9999;
}
```

Figura 21 - Código do Style.css

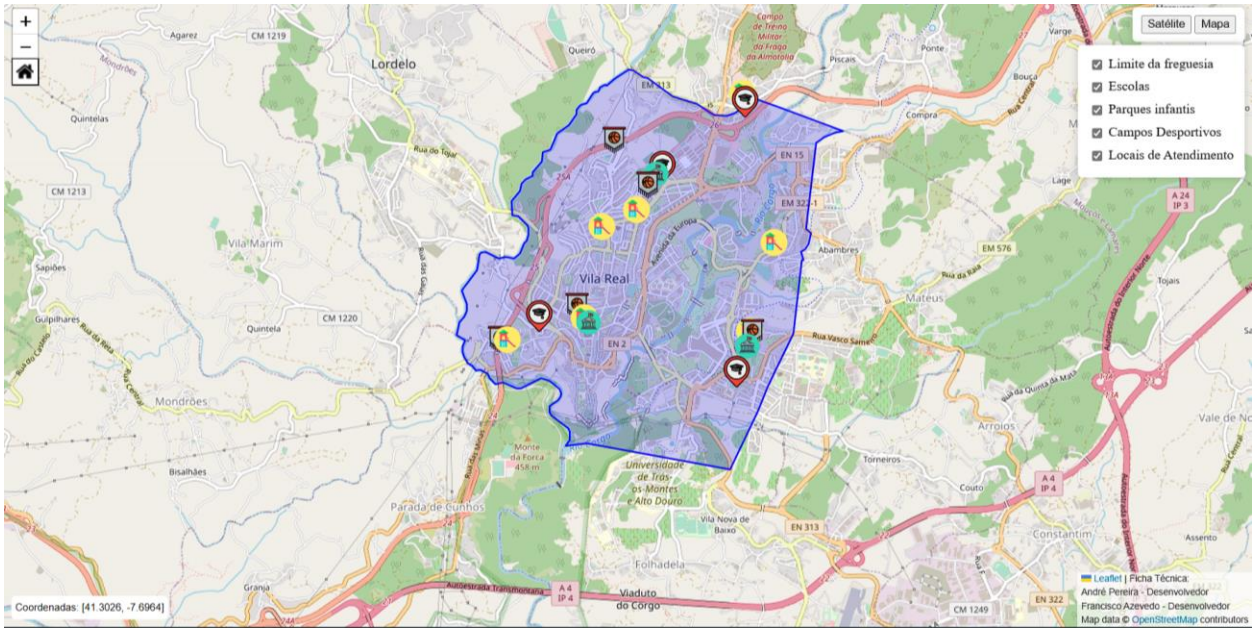


Figura 22 - Produto final (Versão Mapa)

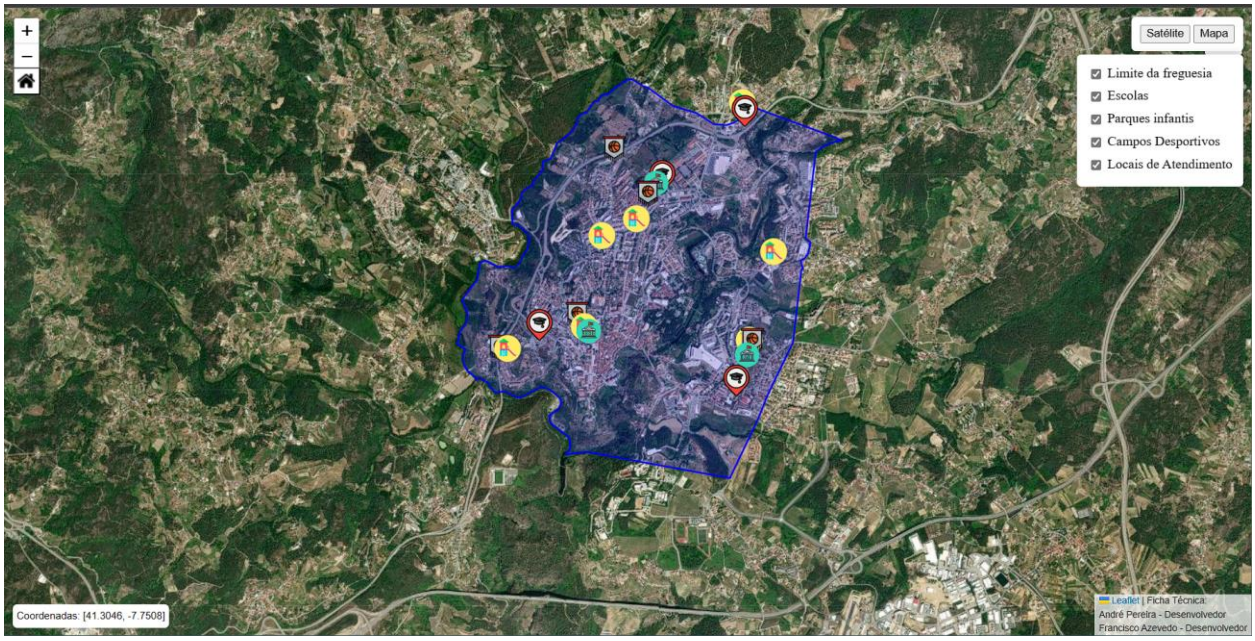


Figura 23 - Produto final (Versão Satélite)

Após a realização do protótipo e realização do presente relatório realizamos também um poster científico de divulgação , que poderá ser observado na figura seguinte (Figura 24).

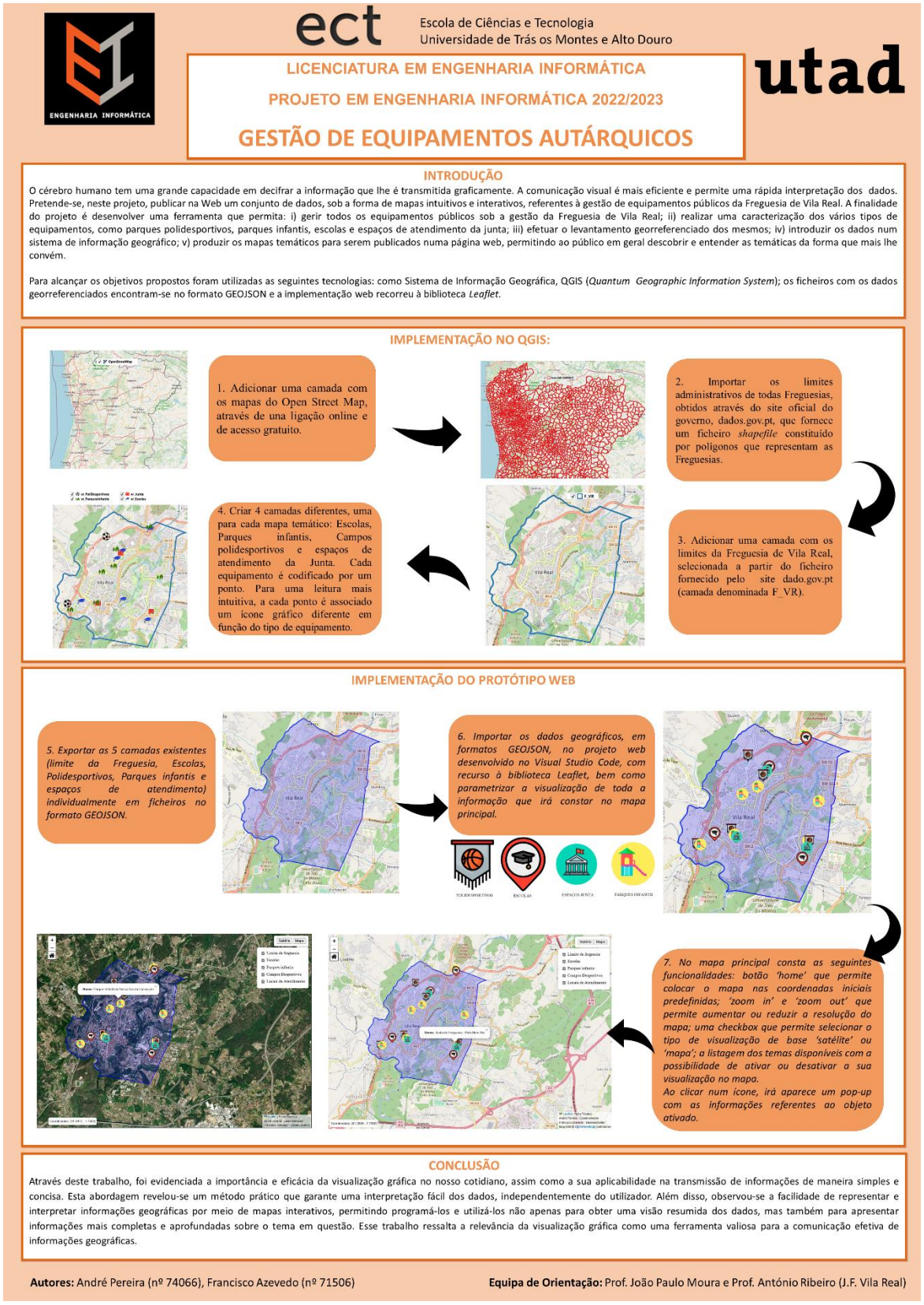


Figura 24 - Poster Científico de Divulgação do Projeto

5. RESULTADOS E DISCUSSÃO

O projeto foi bem-sucedido e a ferramenta criada permite a qualquer utilizador consultar informações sobre os diferentes equipamentos pertencentes à Junta de Freguesia de Vila Real, de forma fácil e intuitiva. Em termos de requisitos funcionais e não funcionais, todos foram implementados com sucesso.
Em suma, pode-se afirmar que este projeto foi bem sucedido, cumprindo-se praticamente todos os objetivos propostos no início da sua conceção. Achamos que fizemos tudo o que se pedia e esperamos ver este nosso protótipo implementado no site da Junta de Freguesia de Vila Real.

6. CONCLUSÃO

Através deste trabalho, foi evidenciada a importância e eficácia da visualização gráfica no nosso cotidiano, assim como a sua aplicabilidade na transmissão de informações de maneira simples e concisa. Esta abordagem revelou-se um método prático que garante uma interpretação fácil dos dados, independentemente do utilizador. Além disso, observou-se a facilidade de representar e interpretar informações geográficas por meio de mapas interativos, permitindo programá-los e utilizá-los não apenas para obter uma visão resumida dos dados, mas também para apresentar informações mais completas e aprofundadas sobre o tema em questão. Esse trabalho ressalta a relevância da visualização gráfica como uma ferramenta valiosa para a comunicação efetiva de informações geográficas.

A. BIBLIOGRAFIA

- M. T. RODRIGUEZ, S. Nunes, and T. Devezas, “Telling stories with data visualization,” Tech. Rep., 2015.
- H. A. D. do Nascimento and C. B. R. Ferreira, “Uma introdução à visualização de informações,” Tech. Rep., December 2011.
- Boulos, M. N. K., & Honda, K. (2006). Web GIS in practice IV: publishing your health maps and connecting to remote WMS sources using the Open Source UMN MapServer and DM Solutions MapLab.
- Alesheikh, A. A., Helali, H., & Behroz, H. A. (2002, July). Web GIS: technologies and its applications. In Symposium on geospatial theory, processing and applications (Vol. 15).