Universidade do Minho
Escola de Engenharia

Daniel Fernandes da Cruz

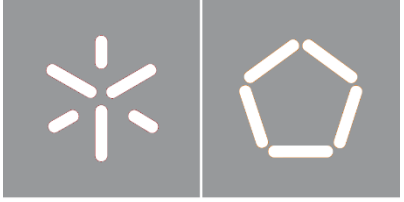
Sistema de geração de conteúdos web para serviços de informação situados

Dezembro de 2021

Sistema de geração de conteúdos web para serviços de informação situados

Daniel Fernandes da Cruz

UMinho | 2021



Universidade do Minho
Escola de Engenharia

Daniel Fernandes da Cruz

Sistema de geração de conteúdos web para serviços de informação situados

Dissertação de Mestrado

Mestrado Integrado em Engenharia e Gestão de Sistemas de Informação

Trabalho realizado sob a orientação do

Professor Doutor Rui João Peixoto José

Dezembro 2021

Agradecimentos

Com o fim deste projeto, conclui-se também mais uma etapa na minha vida, chegando ao fim deste ciclo académico. O desenvolvimento desta dissertação foi possível graças ao apoio do meu orientador, Professor Doutor Rui José, que esteve sempre disponível para tirar dúvidas quando necessário, dando sempre a sua opinião nos vários assuntos que foram abordados.

Quero também agradecer à minha família por todo o apoio pessoal que deram ao longo do projeto, sobretudo aos meus pais que sempre estiveram do meu lado sendo pacientes e compreensivos quanto ao sacrifício tido para elaborar esta dissertação.

Agradeço também aos meus amigos que sempre que possível ajudaram, dando conselhos, mostrando compreensão e acima de tudo proporcionando momentos de prazer que ajudaram a equilibrar o trabalho. Entre essas pessoas agradeço ao Bruno, a Diana, ao José e ao Xavier, pois foram aqueles que estiveram mais próximos de todo este processo.

DECLARAÇÃO DE INTEGRIDADE

Declaro ter atuado com integridade na elaboração do presente trabalho académico e confirmo que não recorri à prática de plágio nem a qualquer forma de utilização indevida ou falsificação de informações ou resultados em nenhuma das etapas conducente à sua elaboração.

Mais declaro que conheço e que respeitei o Código de Conduta Ética da Universidade do Minho.

Resumo

A enorme quantidade de informação que circula na internet hoje em dia, suscita situações em que, apesar de sabermos que a informação se encontra em algum lugar, não existe uma forma simples de chegar até ela. Sendo assim, é necessário desenvolver formas convenientes de colocar essa informação à disposição dos utilizadores nas circunstâncias em que ela é necessária.

O objetivo deste projeto de dissertação é desenvolver posters locativos, que são criados com o propósito de serem utilizados numa situação específica tendo em conta o contexto em que está inserido. Estes posters locativos poderão ser visualizados em qualquer dispositivo com acesso à internet e são capazes de apresentar diferentes conteúdos adaptáveis ao dispositivo que é utilizado.

Numa fase inicial foi feito um estudo sobre os *web components*, tecnologia de grande importância para este projeto, uma vez que os posters terão secções e esta tecnologia permite a reutilização de código de forma fácil. Numa segunda fase foram desenvolvidos posters fazendo uso dos *web components* e testadas diferentes formas de visualizar os conteúdos do poster. Por fim foi desenvolvido um caso de estudo onde se estudou a Especificação Geral sobre Feeds de Transporte Público (GTFS) dos Transportes Urbanos de Braga (TUB), desenvolvendo uma *Application Programming Interface* (API) para fornecimento destes mesmos dados e um gerador de posters que com base na especificação definida cria os posters e os seus conteúdos.

Palavras-Chave: *Locative* media, *web components*, GTFS

Abstract

The enormous amount of information circulating on the internet today, raises situations in which, despite knowing that the information is somewhere, there is no simple way to get to it. Therefore, it is necessary to develop convenient ways to make this information available to users in the circumstances in which it is needed.

The objective of the dissertation project is to develop locative posters, which are created with the purpose of being used in a specific situation, taking into account the context in which it is inserted. These locative posters are displayed on any device with internet access and are capable of presenting different content adaptable to the device being used.

At an early stage a study was made about web components, a technology of great importance for this project since the posters will have sections and this technology allows for easy code reuse. In a second phase, posters were developed using web components and different ways of visualizing the contents of the poster were tested. Finally, a case study was developed where the General Specification on Public Transport Feeds (GTFS) of the Urban Transport of Braga (TUB) was studied, developing an Application Programming Interface (API) to supply these same data and a poster generator that based on the defined specification, creates the posters and their contents.

Keywords: *Locative* media, *web components*, GTFS

Índice

Índice de figuras

Índice de tabelas

Lista de Abreviações, Siglas e Acrónimos

API - Application Programming Interface

CSS - Cascading Style Sheets

CSV – Comma-separated values

DIPP - Digital interactive poster presentation

DOM - Document Object Model

GTFS – General Transit Feed Specification

HTML - Hypertext Markup Language

HTTP - Hyper Transfer Protocol

IRI - Internationalized Resource Identifier

JS - JavaScript

JSON - JavaScript Object Notation

JSON-LD - JavaScript Object Notation for Linked Data

PNA - Public Notice Areas

RDF - Resource Description Framework

TUB - Transportes Urbanos de Braga

URI - Uniform Resource Identifier

1 Introdução

O propósito deste capítulo é fazer uma introdução do tema, apresentando o enquadramento e a motivação para a realização do mesmo. Também é descrita a abordagem escolhida, a estrutura do documento, e a respetiva descrição daquilo que vai ser abordado em cada ponto.

1.1 Enquadramento e Motivação

Existe atualmente uma quantidade enorme de informação a circular pela internet, criando a ideia de que essa informação se encontra à distância de um simples clique. No entanto, isto não significa que consigamos ter acesso à mesma no preciso momento em que necessitamos dela. A relevância da informação é fortemente situada, contextual e espontânea. Imagine-se o seguinte caso: ao passar perto de um restaurante deseja saber que pratos são confecionados no mesmo, no entanto, não tem tempo para ficar no local a analisar o menu. Esta informação certamente estará disponível na *web*, mas não existe uma abordagem genérica e escalável que permita disponibilizar esse conteúdo, e outros do género, explicitamente naquelas situações em que a informação realmente é necessária.

Uma solução para este problema poderá passar por um novo conceito de conteúdo especificamente vocacionado para um momento específico, de fácil acesso quando é necessário e que seja possível gerar de forma automatizada para poder ser criado em larga escala. Neste trabalho explora-se o conceito de poster locativo, como sendo um novo tipo de conteúdo *web* concebido especialmente para esse efeito e que sejam facilmente apresentados em qualquer tipo de dispositivo e visualizado de forma automática. A criação destes posters também deverá ser simples utilizando padrões de desenvolvimento *web*.

A concretização desse conceito levanta diversos desafios, tal como, a definição do conteúdo que irá ser posteriormente visualizado em formato de poster, pois é necessário desenvolver uma especificação que possa ser facilmente reutilizável. Outro grande desafio será a geração de posters em grande escala, uma vez que, terá de ser desenvolvida uma solução que seja capaz de produzir posters que poderão ou não ser distintos entre si. Existe ainda o desafio que passa por perceber como esta solução se poderá enquadrar em diferentes domínios de aplicação, tendo em conta que poderão ser criados posters para diversos contextos.

Para este efeito deverão ser utilizados padrões no que toca ao desenvolvimento *web* de forma que qualquer pessoa possa facilmente criar os seus próprios posters locativos.

É também importante que seja capaz de representar diferentes conteúdos, sendo assim possível criar posters como packs de conteúdo *web*, ou seja, a partir de um conjunto de dados estruturados e respetivos ficheiros de configuração criar um poster que é constituído por várias secções que representam esse conteúdo. Surge assim a necessidade de estruturar toda esta informação de maneira a criar um poster e apresentá-lo de uma forma simples e clara ao utilizador.

1.2 Objetivos e Resultados Esperados

O principal objetivo desta dissertação é a especificação e validação no contexto de instalações em ambiente real de um modelo de poster situado que permite publicar conteúdo *web* estruturado para ser visualizado em qualquer dispositivo com ligação à internet. De forma a atingir este objetivo será necessário fazer a especificação dos conceitos fundamentais deste modelo que são os posters *web*. Os posters *web* são conteúdos visuais, que são criados para responder de forma rápida a uma necessidade específica de informação, mas que integram conteúdo estruturado que pode ser processado automaticamente.

De forma a realizar este projeto, este macro objetivo, será dividido nos seguintes objetivos:

1. Propor uma especificação para os posters locativos que define a sua estrutura capaz de responder aos desafios apresentados;
2. Desenvolver um gerador de posters locativos que possibilite a criação automática de posters a partir de conteúdos estruturados;
3. Desenvolver um caso de estudo de aplicação da especificação no âmbito de um sistema de informação para transportes públicos;
4. Avaliar a aplicação da proposta apresentada no âmbito do caso de estudo, identificando eventuais limitações, desafios e novas possibilidades.

1.3 Abordagem metodológica

Nesta secção será explicado o método utilizado para atingir os objetivos definidos para este projeto de dissertação e para o desenvolvimento do gerador de posters.

Em primeiro lugar foi necessário fazer um estudo sobre os diferentes tópicos associados a este tema de dissertação. Este estudo permitiu realizar um relatório de estado de arte, que se encontra dividido tendo em conta estes tópicos. Neste estudo foi feito um levantamento bibliográfico com o propósito de estudar as áreas relacionadas com o tema, assim como uma pesquisa de diferentes implementações de aplicações semelhantes ao que é esperado obter no final deste projeto.

De seguida foram definidos requisitos para os posters locativos. Estes requisitos têm em conta aspetos importantes para o desenvolvimento dos posters uma vez que servem como uma fundação para a construção dos mesmos.

Na fase seguinte, antes de iniciar o desenvolvimento foi necessário analisar e criar uma especificação para a criação de poster. Aqui foi realizado um estudo sobre tecnologias *web*, mais especificamente os *web components,* e também sobre dados estruturados. Também foram realizadas pesquisas para fontes de dados a serem utilizadas neste projeto de forma a conseguir criar posters para um determinado contexto. Inicialmente os dados encontrados assentavam no tema da reciclagem tendo sido utilizada informação obtida na internet através da Braval. No entanto foi encontrada uma fonte de dados mais sólida para ser aplicada neste projeto que foi o *feed* GTFS dos TUB.

Numa fase seguinte realizaram-se diferentes implementações no sentido de testar a especificação definida. Sendo assim descrito o processo para a criação de componentes que posteriormente serão aplicados nos posters. Também foi realizado um estudo sobre diferentes soluções para apresentar estes posters. Deste estudo foram selecionadas duas soluções para demonstrar duas formas distintas de visualizar e interagir com os posters.

Por fim foi realizado um caso de estudo onde se procurou aplicar o conceito de poster locativo a um determinado domínio, neste caso a área dos transportes públicos tendo sido utilizado os dados dos TUB. Neste caso de estudo foi realizado o processamento dos dados no sentido de criar ficheiros em *JavaScript Object Notation* (JSON) para representar diferentes posters representativos de diferentes cenários de utilização. Também foi feita uma avaliação do caso de estudo tendo em conta o resultado final a garantir que está enquadrado com os requisitos definidos, bem como a descrição dos limites e de desafios encontrados e novas possibilidades para futuras iterações.

1.4 Estrutura do Documento

O documento está estruturado em 7 capítulos:

1. Introdução: onde se faz uma breve introdução do tema que irá ser estudado, o enquadramento, os objetivos e os resultados esperados da realização da dissertação.

2. Estado da Arte: em que se faz o levantamento do estado de arte de diferentes áreas que se encontram relacionadas com o projeto.

3. Pressupostos na criação de componentes e posters: neste capítulo são levantados os pressupostos que darão lugar ao desenvolvimento da solução nos seguintes capítulos

4. Especificação e Implementação dos posters: aqui pretende-se fazer a especificação e o enquadramento dos diferentes conceitos abordados neste projeto de dissertação.

5. Caso de Estudo – Transportes Urbanos de Braga (TUB): onde é realizado um caso de estudo que permite aprofundar o processo de desenvolvimento em larga escala.

6. Conclusão: onde são apresentadas as conclusões da realização deste documento.

7. Referências Bibliográficas: onde estão identificadas as diferentes fontes utilizadas para o levantamento do estado da arte.

2 Estado da Arte

Neste capítulo é realizado um estudo e são apresentados os diversos conceitos inerentes ao tema deste trabalho.

Para a realização do estado de arte deste projeto foram consultadas plataformas como o Google Scholar, ACM Digital Library, Mendeley, Scopus, Z-Library para as pesquisas relacionadas com as diferentes áreas do projeto.

Alguns dos exemplos de termos utilizados para realizar o levantamento do estado de arte foram "*Locative* media", "*Pervasive* media", "GTFS *Data*" entre outros. Devido a sua importância para o projeto a pesquisa realizada incide sobre os seguintes temas:

1. Posters em meio não digital
2. Redes de ecrãs públicos
3. *Locative* Media
4. Microsserviços
5. *Linked Data* para conteúdos estruturados
6. Dados GTFS
7. *Web Components*

2.1 Posters em meio não-digital

Os ecrãs públicos são comuns na vida urbana e são cada vez mais compreendidos como um meio de comunicação difundido. Ainda assim, apesar do seu enorme potencial para lidar com uma necessidade muito relevante, que é a comunicação, as telas públicas ainda não se conseguiram tornar um canal digital convencional (Coutinho & José, 2020).

A utilização de posters tem crescido ao longo dos tempos por parte dos profissionais, sendo utilizados para diferentes casos de uso. Posters desenvolvidos para conferências são usualmente exibidos em departamentos, e tornaram-se parte do dia-a-dia dos profissionais de saúde (Murray et al., 1998).

A forma como nos apresentamos é importante, uma vez que, reflete impressões acerca de nós como indivíduos ou instituições/departamentos por nós representados (Murray et al., 1998). Assim sendo o autor descreve 5 princípios para o design de um poster.

1. Definir metas e objetivos: Ao nível das metas é necessário definir aquilo que se pretende comunicar e como se deverá divulgar a informação. Quanto aos objetivos, deve-se ter em conta aspetos como gerar interesse, informar, efetivamente conseguir "vender" o nosso trabalho e deixar uma impressão duradoura.
2. Antes de começar: Deve-se considerar o nosso público-alvo, isto determina em parte o estilo do poster. Em seguida definir o assunto e o conteúdo para depois reunir material de forma seletiva, escolhendo apenas informação que representa bem o assunto em questão.
3. Planear: É uma boa ideia fazer um esboço, este permite experimentar diferentes layouts de forma a encontrar aquele que melhor se adequa a solução. Também ajuda a dar uma ideia de como os diferentes tamanhos, imagens ou materiais serão exibidos. Este é um aspeto muito importante, pois pode economizar tempo no futuro, esforço por parte do designer e despesas com problemas de prevenção.
4. Características de design: Aqui o autor tem em conta diferentes aspetos como o esquema de cores que devem ser utilizadas para dar uma impressão de união e definir pontos particulares ou material relacionado. O tipo, ou seja, o poster deve ser legível de um a dois metros de distância para tal deve-se escolher um tipo de letra que seja clara e fácil de ler.

   O uso de fotografias pode servir dois propósitos, adicionar interesse visual e ilustrar um ponto, pois podem ajudar o espetador a compreender o material.

   Os diagramas/ilustrações, podem ser desenhados à mão ou produzidos por computador, devem ser simples e a negrito de forma a poderem ser visualizados de longe.

   Deve também ser feito uso do ponto focal, atraindo assim a atenção da audiência para um ponto específico através do uso de, por exemplo, uma fotografia dramática.
5. Transporte de posters: O local onde se quer apresentar o poster é determinante para a forma como o poster deve ser produzido. No caso deste autor ele apenas destaca papel e cartão, ambos tem vantagens e desvantagens. Por exemplo, posters em papel são mais leves podendo ser transportados mais facilmente. O poster em cartão, por outro lado, pode ser montado no local e dividido em secções que se podem dispor de diferentes formas.

As *Public Notice Areas* (PNA´s) tradicionais têm uma grande usabilidade dai terem um grande sucesso. Utilizando caneta e papel qualquer pessoa poderá postar conteúdo, permitindo que a informação seja obtida de forma simples e rápida. Consequentemente devem ser providenciadas técnicas de interação adequadas que realizam uma função semelhante sendo altamente intuitivos e fáceis de usar. O estudo levado a cabo por (Alt, Memarovic, et al., 2011) teve como objetivo explorar como estas soluções poderiam ser interligadas de maneira a criar uma rede de displays públicas, suportando assim novas formas de compartilhar o acesso a displays. Os autores propõem 5 princípios e dão ideias de como os aplicar:

1. Projetar para usos específicos dos quadros de avisos:
   Deve ter-se em conta que os utilizadores podem interagir com mais do que um display ao longo da loja. Devendo o design capitalizar o facto de que os usuários vão ser expostos a uma rede de monitores.
2. Respeitar o foco da vizinhança dos quadros de avisos:
   O conteúdo que é publicado deve estar conectado ao local em alguma forma e não campanhas publicitarias centralizadas.
3. Apoiar o perfil emergente de um PNA pelos proprietários e utilizadores do espaço:
   Dando aos proprietários controlo sobre o quadro.
4. Projetar para a flexibilidade de entrada, baixo custo para postar:
   Aqui recomenda-se que sejam preservadas a flexibilidade dos quadros de avisos digitais através da disponibilização de posters ad-hoc para pessoas poderem criar conteúdo na hora, por exemplo, *templates* pré-definidos. Posters sofisticados para pessoas que preparam conteúdo em avanço, poderem transferir os posters para o local, por exemplo, via scanner ou USB. Posters profissionais para aquelas pessoas que distribuem conteúdo com design profissional, por exemplo, folhetos.
5. Suportar informação que pode ser retirada:
   A possibilidade de retirar informação é um requisito crucial. Existem assim duas opções, fornecer algo que o utilizador possa utilizar, por exemplo, um *Uniform Resource Locator* (URL) ou número de telefone ou então fornecer uma copia do conteúdo, por exemplo, um folheto.

Sendo assim os autores recomendam que o design central deve levar em conta o contexto de um potencial display bem como as pessoas que passarão pelo mesmo, a comunidade em que o display se

encontra situado e as expetativas dos proprietários quanto ao conteúdo. Além da criação de conteúdo flexível, publicação de conteúdos e controle dos conteúdos são importantes para permitir que uma grande variedade de pessoas possa utilizar estes serviços.

Tendo em conta que um dos principais objetivos deste projeto de dissertação é propor uma especificação para o desenvolvimento de posters este estudo permitiu adquirir uma base útil para os futuros desenvolvimentos de posters, uma vez que os cinco princípios apresentados anteriormente apesar de terem sido definidos pelo autor tendo em conta os posters tradicionais é possível ver que podem ser aplicados a outros tipos de conteúdos dado que são boas práticas a ter em conta para qualquer tipo de representação de informação.

2.2 Redes de ecrãs públicos

A metáfora do poster inspirou o desenvolvimento de alguns sistemas digitais de apresentação de media. As funcionalidades desses sistemas são desenhadas de modo a facilitar a forma como os utilizadores interpretam os conteúdos apresentados ou conteúdos que podem criar. Os posters digitais apresentam vantagens comparativamente aos posters tradicionais, como, terem um baixo custo, permitir fazer adições multimédia, terem uma grande capacidade de armazenamento e poupam tempo aos apresentadores (DÁngelo, 2012).

Duas destas formas de apresentar posters digitais podem ser vistas em seguida:

- Digital interactive poster presentation (DIPP): é um poster em formato pdf que pode ser projetado em paredes ou ecrãs. Ao longo de uma DIPP o orador pode aumentar determinada parte do poster dando atenção a um determinado aspeto da sua pesquisa. Este tipo de poster pode ser disponibilizado antes ou depois da conferência, de maneira que os participantes possam pesquisar os posters.

- MediaPoster: poderá ser apresentado através de uma tela ou um smartboard, contendo ligações para informação adicional. A particularidade destes é que os espetadores podem selecionar a área que têm interesse tendo acesso a vários documentos e imagens que serão abertos numa área designada para a sua visualização.

2.2.1 Digifieds

O caso da *Digifieds* (Alt, Kubitza, et al., 2011) procura com a utilização de displays digitais tornar as PNA´s mais atrativas para os donos e provedores de conteúdo. Para tal procuram dinamizar estes displays em 5 aspetos:

1. Através do aumento do conteúdo digital com conteúdo e serviços multimédia, por exemplo, imagens, vídeos ou Google Maps.
2. Os recursos de rede permitem facilmente distribuir conteúdo e colaboração remota.
3. Disponibilização de meios para retirar a informação de forma fácil, por exemplo, permitindo que o conteúdo seja enviado para o email ou descarregado para o telefone.
4. A gestão dos conteúdos pode ser facilitada removendo conteúdo antigo automaticamente.
5. O conteúdo digital pode ser pesquisado e filtrado de maneira a encontrar o conteúdo mais relevante e popular.

No caso do *Digifieds* para realizarem as suas pesquisas desenvolveram um protótipo de uma PNA em formato digital. Esta é composta por quatro elementos. O primeiro é um servidor back-end para fazer a gestão dos dados, o segundo um cliente de visualização baseado na *web* para ver as informações. De seguida uma aplicação mobile como meio alternativo de interação com o display e em quarto um cliente *web* publico.

Com este protótipo procuraram preservar as vantagens dos PNA´s tradicionais melhorando ao mesmo tempo a experiência com funcionalidades novas, como conteúdo digital (imagens e vídeos), conteúdo interativo (mapas), filtragem de postagens através de critérios como data e popularidade e remoção automática de conteúdo obsoleto entre outros.

2.2.2 Instant Places

O Instant Places é uma investigação sobre o papel do Bluetooth e a sua presença e nome como técnicas para interações situadas perto de displays público (José et al., 2008). Através do nome do Bluetooth procuraram utilizar um mecanismo de instruções simples onde o sistema consegue detetar partes do nome do dispositivo de Bluetooth como instruções explicitas que permitem manifestar conteúdo em displays situados.

Neste estudo foi implementado um sistema que gera um ecrã público cujo conteúdo é relevante para a situação e que é direta e indiretamente derivado da presença de Bluetooth. Este sistema é composto por

um ou mais computadores com Bluetooth conectados a um ecrã público ligados a um repositório central. As informações sobre os dispositivos na área são coletadas periodicamente por um scanner de Bluetooth.

Este projeto apresenta algumas funcionalidades para interagir com o sistema, fazendo uso de comandos que são contidos no nome do dispositivo Bluetooth. Isto foi conseguido analisando os nomes dos dispositivos e procurando por determinadas palavras-chave que posteriormente seriam reconhecidas como comandos e desencadeiam ações especificas. No total eram suportados dois tipos de comandos. O primeiro é um comando tag, permite que as pessoas associem várias tags com a sua identidade. O segundo tipo é a indicação de um nome de usuário do Flickr.

Para concluir, com o estudo realizado consegue-se perceber que já existem vários casos onde se faz a utilização de posters digitais, dando uma nova vida ao conceito de poster. Com a utilização de tecnologias como ecrãs e os smartphones é possível aumentar a experiência do utilizador e a sua interação com os posters, permitindo assim trazer uma vasta gama de funcionalidades que não se conseguem reproduzir utilizando os posters tradicionais.

2.3 *Locative* Media

Com a crescente utilização de smartphones por parte das pessoas e consequentemente as aplicações *web* e *mobile*, o conceito de localização é cada vez mais enraizado no dia-a-dia das pessoas. Torna-se assim importante explorar o conceito de *Locative* media, uma vez que, este vai de encontro aos objetivos deste projetos de dissertação e como diz (Wilken, 2012) às questões de localização e *Location-awareness* são cada vez mais importantes para os nossos compromissos com a internet e meios de comunicação *mobile*.

Segundo (Nova, 2004), o conceito *Locative* Media engloba toda a informação sobre o local físico bem como outros aspetos contextuais. Para (Wilken, 2012) o termo *Locative* media é utilizado para capturar as várias tecnologias e práticas de reconhecimento de localização. Este meio digital de comunicação encontra-se, portanto ligada a um local.

2.3.1 *Context-awareness*

Um ponto muito falado nesta área é o contexto, (Schmidt et al., 1999) este é visto como algo que está a nossa volta e dá significado a alguma coisa. (Nova, 2004) descreve duas maneiras de obter contexto

utilizando tecnologias de informação, pedir informação ao utilizador ou fazer a monitorização das atividades do utilizador.

Segundo (Schmidt et al., 1999) o conceito de *context-awareness* sofre pela falta de modelos que permitam fazer a comparação entre diferentes abordagens. Sendo assim, o autor propõe um modelo para definir o contexto.

O modelo proposto diz que o contexto deve descrever a situação e o meio em que um utilizador ou um dispositivo se encontra. O contexto é identificado por um nome único. Para cada contexto é relevante um conjunto de recursos e por fim para cada um destes recursos existe um intervalo de valores que é determinado pelo contexto.

O conceito de contexto foi estruturado como um modelo organizado hierarquicamente por (Schmidt et al., 1999) onde se faz a distinção entre duas categorias, o ambiente físico e os fatores humanos. No nível mais baixo existe o ambiente físico onde são referenciadas as variáveis físicas como localização que tanto pode ser relativa ou absoluta, as condições de luminosidade e temperatura, e as infraestruturas como os recursos que circundam o utilizador. No nível mais alto o fator humano, o contexto relacionado é estruturado em informação sobre o fator humano, o meio social do utilizador e as tarefas do utilizador.

O conceito de *Locative* Media poderá mostrar-se útil uma vez que as informações do local físico em que um utilizador se encontra poderão mostrar-se importantes no sentido de apresentar um determinado tipo de conteúdo específico para a situação em que o utilizador se encontra.

2.4 Microsserviços

Nesta secção é realizado um estudo sobre os microsserviços, um tópico pertinente tendo em conta que um dos principais objetivos deste projeto é criar um gerador de posters. É então importante que se tenha um bom conhecimento sobre este tipo de serviços e o seu funcionamento.

Os microsserviços tem tido um grande impulso pelos diferentes setores da indústria, facilitando a entrega ágil de mecanismos para *service-oriented* architecture e para migrar function-oriented *legacy architectures* para uma orientação de serviço com maior flexibilidade (Larrucea et al., 2018).

Segundo (Thönes, 2015) um microsserviço é uma pequena aplicação que pode ser implementada, escalada e testada de forma independente. Esta aplicação também tem uma única responsabilidade no sentido de ter uma única razão para mudar e/ou uma única razão para ser trocada.

Os microsserviços podem afetar significativamente os atributos de qualidade (Thönes, 2015):

- Segurança: uma vez que os dados fluem entre microsserviços, existe a necessidade de proteger essa comunicação com técnicas de criptografia.
- Performance: usualmente estes serviços tem um desempenho pior do que outras soluções, por isso o desempenho final depende de vários fatores, como, aspetos de rede e virtualização.
- Confiabilidade: normalmente por serem distribuídos são menos confiáveis que outras soluções.
- Disponibilidade: nas arquiteturas de microsserviços, a disponibilidade não depende apenas da disponibilidade do microsserviço em si, mas também onde se encontra integrado. Por outro lado, os microsserviços reduzem o tempo de implementação aumentando assim a disponibilidade.
- Manutenção: é um dos melhores aspetos ao utilizar os microsserviços uma vez que eles são independentes o que torna a sua manutenção mais fácil.
- Testabilidade: nas fases iniciais é mais fácil testar o microsserviço, no entanto, ao integrar pode ser mais difícil de testar dependendo do número de componentes e das suas conexões.

O conceito de microsserviço poderá ser aplicado neste projeto se considerar o poster como um microsserviço, ou seja, um poster poderá ser visto como uma pequena aplicação que pode ser utilizada de forma independente tendo uma única responsabilidade apresentar o conteúdo para o qual foi criado.

2.5 Linked Data para conteúdos estruturados

Esta secção tem como objetivo estudar o método de estruturar dados, mais concretamente o *JavaScript Object Notation for Linked Data* (JSON-LD) de forma a perceber como o mesmo pode ser aplicado nos objetivos do projeto. Uma vez que se pretende criar posters que seguem uma estrutura bem definida e amplamente divulgada, este tópico apresenta uma grande importância.

2.5.1 Linked Data

Os dados ligados são uma forma de criar uma rede de dados interpretáveis por máquinas em diferentes documentos e *websites*. Permitindo assim que uma aplicação se inicia em um pedaço de dados ligados, e

siga links incorporados para outras partes de dados ligados que se encontram hospedados em diferentes sites na *web* (Gregg Kellogg, Pierre-Antoine Champin, 2019).

Existem quatro princípios em quais os dados ligados assentam segundo (Bizer et al., 2009):

1. Usar *Uniform Resource Identifiers* (URI´s) como nomes para as coisas.
2. Usar *Hypertext Transfer Protocol* (HTTP) URI´s de forma que as pessoas encontrem estes nomes.
3. Quando uma pessoa procura um URI, fornece informação útil.
4. Incluir links para outros URI´s, de forma que se consiga descobrir mais coisas.

2.5.2 JSON-LD (*JavaScript Object Notation for Linked Data*)

JSON-LD é uma sintaxe leve para serializar dados vinculados em JSON. Esta permite através de um conjunto de transformações mínimas que se consiga interpretar o JSON como dados ligados. O principal objetivo do JSON-LD é o de usar dados ligados em ambientes de programação na *web*, de forma a contruir serviços *web* interoperáveis e armazenar estes dados em mecanismos de armazenamento baseado em JSON (Gregg Kellogg, Pierre-Antoine Champin, 2019).

Em adição a todos os recursos que o JSON oferece, o JSON-LD apresenta:

- Um mecanismo que permite identificar universalmente o objeto JSON por meio do uso de Internationalized Resource Identifiers (IRI´s).
- Uma forma de desambiguar chaves compartilhadas entre diferentes documentos JSON, fazendo o mapeamento para IRI´s por meio de um contexto.
- Um mecanismo pelo qual um valor em um objeto JSON pode referir para um recurso em um *website* diferente.
- A habilidade de anotar strings com a sua linguagem.

O JSON-LD foi desenvolvido para ser utilizado como JSON, sem o conhecimento de *Resource Description Framework* (RDF) e a sintaxe foi projetada para não perturbar sistemas previamente desenvolvidos e que correm com JSON, proporcionando a possibilidade de atualizar facilmente de JSON para JSON-LD (Gregg Kellogg, Pierre-Antoine Champin, 2019).

Para concluir, o estudo realizado permite concluir que o JSON-LD poderá ser uma ferramenta bastante útil neste projeto, uma vez que, permite estruturar os dados através de um formato que é fácil de entender e escrever pelas pessoas.

2.6 Dados GTFS

O GTFS é um formato padrão de dados públicos referentes a horários de trânsito e informação geográfica (Prommaharaj et al., 2020). Este formato de dados uma vez disponibilizado permite o desenvolvimento de aplicações e retirar valor por parte de analistas e engenheiros (Kunama & Phithakkitnukoon, 2017).

Um *feed* de GTFS é um conjunto de ficheiros podendo variar de entre 6 e 13 ficheiros *Comma-separated valões* (CSV) que representam diferentes dados, como rotas, paragens, viagens, tempos de paragens (Prommaharaj et al., 2020).

Através da utilização do GTFS já foram desenvolvidas aplicações com objetivos variados, como por exemplo planeadores de viagens, ferramentas de planeamento de trânsito, aplicações mobile, entre outras categorias (Kunama & Phithakkitnukoon, 2017).

Como foi dito anteriormente o *feed* de GTFS é composto por vários ficheiros o conteúdo dos mesmos é explicado de seguida:

- agency.txt – que contém informações sobre a agência que disponibiliza o *feed*, e o fuso horário da cidade em que opera.
- stops.txt – contém as paragens, ID de paragem, e localizações geográficas das mesmas.
- trips.txt – representa as direções de cada movimento do veículo. Tendo cada viagem um ID de serviço.
- route.txt – tem informação sobre como as viagens são agrupadas.
- stop_times.txt – contém os tempos de partida e chegada de cada paragem, e a sequência de ID das viagens.
- calendar.txt – apresenta a informação dos dias de serviço referenciados por viagens.
- calendar_dates.txt – contém informação sobre os serviços semanais que estendem o calendar.txt.
- shapes.txt – representa as rotas que o veículo faz através de polilinhas geográficas.

Na Figura 1 é possível ver a relação que existe entre diferentes ficheiros do *feed* GTFS bem como as ligações que existem entre eles e os campos que existem em cada ficheiro.

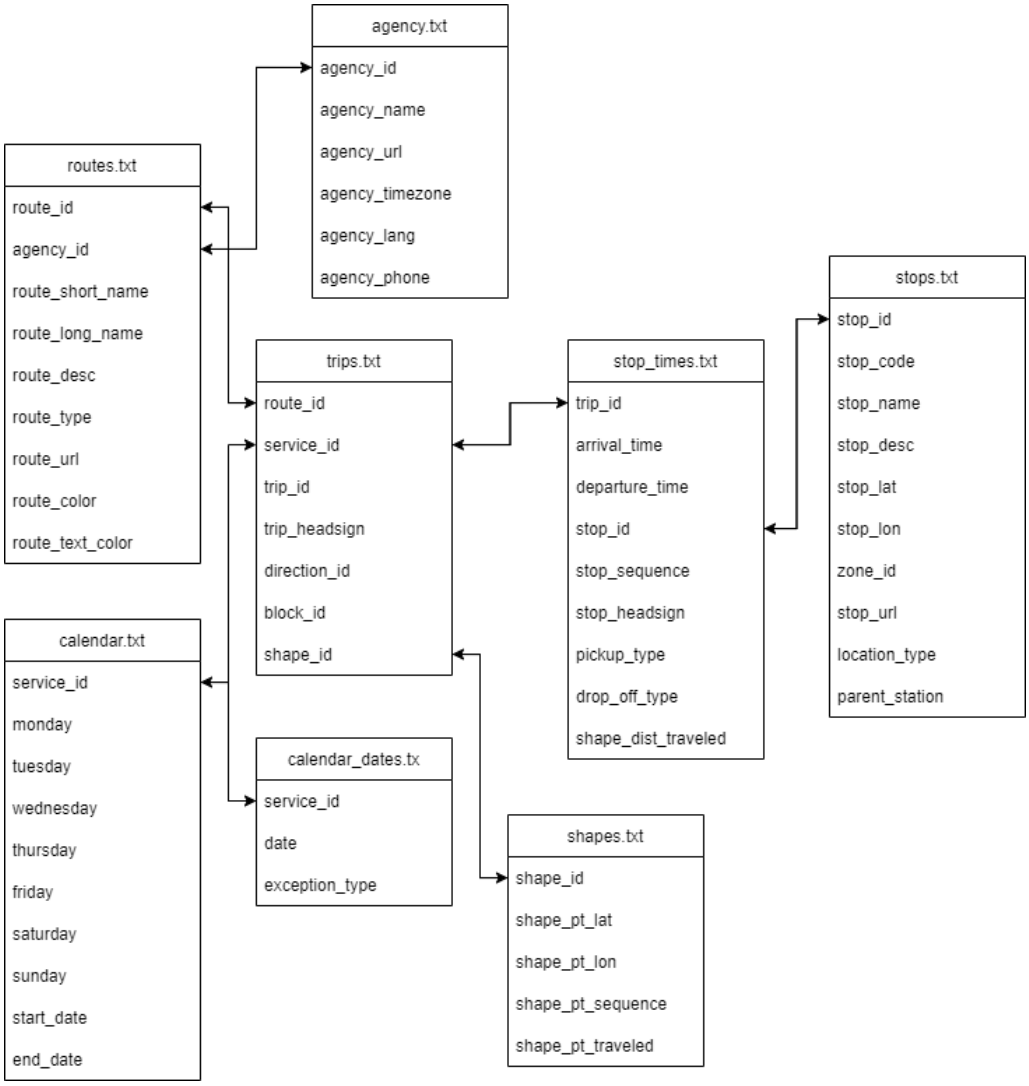


Figura 1 - Relações entre os ficheiros do feed de GTFS – Adaptado de (Kunama & Phithakkitnukoon, 2017)

Nos últimos anos a Google tem vindo a desenvolver o GTFS *Realtime*, que é uma extensão do GTFS *static* falado anteriormente. Este novo *feed* permite entregar atualizações em tempo real para os desenvolvedores. O GTFS *Realtime* deve ser atualizado regularmente e apesar de tudo, utilizar os dados do GTFS *Realtime* apresenta-se mais difícil do que utilizar o tradicional GTFS *static* (Prommaharaj et al., 2020).

Para concluir este tópico, facilmente se percebe que o GTFS é bastante útil para este projeto. A sua utilidade advém do facto de conter dados variados que permitem criar posters em larga escala, uma vez que os dados já se encontram estruturados e com ligações entre si.

2.7 *Web Components*

Nesta secção será realizado um estudo sobre os *web components*, tendo como objetivo perceber o que são e como funcionam de maneira a poderem ser aplicados neste projeto. Este tema revela-se necessário pois um dos principais objetivos deste projeto é desenvolver conteúdo em larga escala sendo para tal essencial criar componentes que possam ser reutilizados em diferentes situações (*Web Components | MDN*, n.d.).

Os *web components* permitem a criação de elementos *Hyper Text Markup Language* (HTML) que são reutilizáveis onde todas as funcionalidades ficam encapsuladas dentro do mesmo de forma a não afetar o resto do código desenvolvido. Isto resolve um dos principais problemas ao trabalhar com estruturas de marcação customizadas, que passa por ter o mesmo código várias vezes repetidas ao longo da página tornando a manutenção da mesma mais difícil.

Os *web components* são formados principalmente por três tecnologias, que podem ou não ser utilizadas em conjunto, dependendo daquilo que se quer fazer. O *Custom Elements* diz respeito à possibilidade de criar elementos personalizados que contém a sua funcionalidade encapsulada na página HTML.

O *Shadow DOM* que permite fazer um dos aspetos mais importante dos *web components* a encapsulação, permitindo anexar um *Document Object Model* (DOM) separado para o elemento em questão. E por fim o uso de *templates* e slots, que se provam uteis quando se esta a utilizar várias vezes a mesma estrutura em uma página, permitindo criar um *template*.

Nos últimos anos têm surgido vários componentes do género de *widgets*, estes têm vindo a ser desenvolvidos e podem ser aplicados a vários elementos padrão do HTML (Krug & Gaedke, 2015).

2.7.1 SmartComposition

Em (Krug & Gaedke, 2015) os autores fazem uma introdução à utilização de *web components* com o objetivo de criar aplicações capazes de serem utilizados em diferentes dispositivos. Esta abordagem passa por melhorar os *web components* com um canal de comunicação baseado em eventos.

Estes *SmartComponents* não requerem plataformas dedicadas ao contrário de outras abordagens, uma vez que são baseados em tecnologias *web* padrão que são suportadas nos browsers modernos. Isto é uma

das principais vantagens de se utilizar os *web components* pois proporciona o desenvolvimento de elementos que podem ser utilizados sem a necessidade de importar bibliotecas de terceiros.

Os autores observaram alguns problemas com outras abordagens como o facto dos outros componentes não poderem ser aplicados diretamente no código HTML, tendo de se usar algum tipo de elemento para marcar a sua posição. Outro aspeto encontrado é que usualmente esses componentes utilizam dependências e mostram-se mais complicados para configurar. Por fim, estes componentes são criados no mesmo DOM o que pode criar conflitos caso existam elementos com ids duplicados ou regras de *Cascading Style Sheets* (CSS) que podem ser aplicadas a outros elementos. Este último problema como foi dito anteriormente pode ser resolvido com a utilização de *web components* graças ao *Shadow DOM*.

Os SmartComponents são definidos em três partes. A primeira diz respeito ao *template*, onde se definem o aspeto e a estrutura HTML do componente. Em segundo lugar existe uma secção onde se definem os estilos a ser aplicados ao componente. A terceira parte passa por cobrir as dependências e o código do programa.

2.7.2 ChemDoodle *web components*

Tal como os SmartComponents citados anteriormente os ChemDoodle *web components* fazem uso desta nova tecnologia para criar componentes para desenvolver um kit de ferramentas para visualizar gráficos químicos, editar estruturas e interfaces (Burger, 2015).

A biblioteca de *web components* da ChemDoodle foi lançada em 2009 e é o primeiro kit de ferramentas de química para a visualização de estruturas e edição a ser contruída utilizando apenas tecnologias *web* padrão.

O autor faz o levantamento de várias vantagens possibilitadas pela utilização de tecnologias *web* padrão. Uma vez que *JavaScript* (JS)é habilitado nos browsers por defeito não é necessário fazer instalações ou atualizações. As aplicações JS são rápidas a ser carregadas e existem menos preocupações com seguranças visto que o código é isolado no browser. Por fim a integração do HTML e JS permite uma experiência sem transtornos ao utilizador, podendo está biblioteca ser carregada e visualizada em qualquer lugar que utilize HTML5.

Com a realização deste estudo foi possível perceber que os *web components* têm sido utilizados em diferentes abordagens ao longo dos últimos anos. Está tecnologia mostra-se como sendo uma mais-valia para o desenvolvimento de conteúdo reutilizável para a *web*, podendo ser uma tecnologia de grande interesse para a realização deste projeto de dissertação.

3 Requisitos na criação de posters locativos

Nesta secção será feito o levantamento dos requisitos e pressupostos necessários para o desenvolvimento das diferentes vertentes deste projeto de dissertação. Também é realizada uma curta introdução a questões como os conteúdos dos posters locativos a sua criação e por fim a sua publicação.

3.1 Poster Locativo

Um poster locativo é um poster que é criado para um determinado contexto e ser apresentado em um determinado local. Existe assim a necessidade de desenvolver estes posters tendo em conta os mais variados casos de uso para a sua visualização pois estes serão consumidos preferencialmente através de smartphones e tablets por parte do seu utilizador, visto que nos dias de hoje este é um dos principais canais de acesso a informação. Para todos os efeitos, esta abordagem não impede a utilização de diferentes meios, como por exemplo um portátil ou desktop, visto que todos os conteúdos são disponibilizados na *web*, podendo ser visualizados em qualquer ecrã.

3.2 Requisitos para os posters locativos

Tendo em conta a especificação do poster locativo, chegou-se a um conjunto de requisitos que devem ser cumpridos.

- Requisito 1: *Web*

  Este requisito surge de a necessidade dos posters locativos serem representados na *web*, sendo assim o conteúdo dos posters e a sua especificação deve ser aberta e flexível para o uso corrente de *web technologies*, ferramentas, competências e standards. Desta forma a especificação deve apenas providenciar um número mínimo de regras.

- Requisito 2: *Sandbox*

  O segundo requisito prende-se com o facto de poderem existir vários posters a serem representados num determinado locar desta forma um poster não deve interferir no conteúdo da *web* de seu contexto de execução ou no conteúdo de outro poster executado no mesmo contexto.

- Requisito 3: *CrossChannel*

  Este terceiro requisito deve-se a necessidade de os posters poderem ser visualizados em qualquer dispositivo, logo os posters devem acomodar mudanças significativas de tamanhos de display de forma a conseguir ser visualizado em diferentes formatos.

- Requisito 4: *Multipart*

  Este último requisito assenta no conteúdo que o poster irá apresentar podendo o mesmo ser composto por várias partes, tendo assim diferentes secções em que cada uma delas apresenta informação com um propósito específico

3.3 Criação de posters locativos

Qualquer pessoa poderá criar os seus próprios posters, desde pessoas com conhecimento ligado à área de desenvolvimento *web*, a pessoas que não tenham um conhecimento tão profundo desta área. Para tal necessita apenas de ter noção dos diferentes componentes que constituem um poster.

Os posters deverão ser locativos, ou seja, de alguma forma devem ser identificáveis e acessíveis em diferentes locais com base no seu contexto. Sendo assim, para que o poster seja locativo o criador deve fazer a publicação do mesmo dando-lhe um id, de preferência aquele que é utilizado no JSON que define o poster.

Quanto ao local onde os posters poderão ser criados, não existe há partida nenhum local específico para o efeito. Uma pessoa que queira desenvolver posters pode fazê-lo em qualquer lugar, seja no seu próprio computador ou no computador de outra pessoa. O criador terá apenas de ter em conta os requisitos que foram definidos e criar os posters locativos.

Para os efeitos deste projeto de dissertação o *template* que define a estrutura do poster que foi proposto não deverá ser alterado. Tal deve-se ao fato que o gerador que foi desenvolvido para criar os posters tem em conta a estrutura previamente definida. No entanto, nada impede que outras pessoas peguem neste *template* base e lhe acrescentem novos atributos de forma a desenvolver novos conteúdos. Desta forma é possível desenvolver novas aplicações destes conteúdos estruturados continuando a poder fazer uso do *template* no contexto deste projeto.

3.4 Publicação de posters locativos

O poster depois de criado deve estar pronto a ser visualizado. Com isto quer se dizer que no final da criação do poster todo o conteúdo que diz respeito ao mesmo, como, *web components* e as suas respetivas partes, ficheiro HTML onde são chamadas as diferentes secções do poster deverão estar dentro da pasta

do poster. Desta forma é garantido o funcionamento do poster no momento em que é necessário visualizar o mesmo.

A estrutura do poster poderá ser alterada tendo em conta que o que define a forma como o mesmo é representado é o ficheiro HTML. Isto é, o criador do poster depois de definir quais os *web components* que quer utilizar ao criar o ficheiro HTML define como as diferentes secções do poster devem ser apresentadas, utilizando por exemplo um carrossel ou um acordeão.

3.5 Ligações entre posters locativos

O poster como definido até ao momento não apresenta a capacidade de se conectar com outros posters e também de navegar de um poster para outro. Todos os elementos que fazem parte do conteúdo de um determinado poster dizem respeito apenas a esse poster. Contudo nada impede que o criador do conteúdo desenvolva por exemplo, componentes cujos dados sejam URL´s para outros posters, podendo assim ter uma secção no poster que faz ligação a outros posters.

4 Especificação e Implementação dos posters

O objetivo desta secção é o de fazer a especificação e o enquadramento dos vários conceitos técnicos que serão abordados neste projeto, bem como o especificamento da solução, que irá ser implementada.

4.1 Definição de *web components*

Optou-se por desenvolver uma solução que utiliza *web components* para criar os posters locativos. Os *web components* serão utilizados como uma forma de criar componentes que podem ser reutilizados e cujas funcionalidades se encontram encapsuladas, não sendo afetadas por outros componentes existentes no poster. Estes componentes terão uma estrutura previamente definida e serão utilizados dentro dos posters como secções, ou seja, diferentes partes que dizem respeito a um determinado tópico.

De maneira a facilitar o desenvolvimento de posters foi criado uma aplicação onde se poderão armazenar os componentes criados e também fazer a visualização dos mesmos. Para fazer o upload de um *web component* deverão seguir-se algumas regras de forma a garantir que o mesmo possa ser utilizado posteriormente para a criação de posters. O utilizador ao desenvolver o *web component* deve ter em atenção uma estrutura de ficheiros como a que se encontra apresentada na Figura 2.

Figura 2 - Estrutura de pastas

Um *web component* será constituído por vários elementos e para poder ser visualizado no *website* deve ter em conta 3 ficheiros, sendo eles os seguintes:

1. Ficheiro JS – Este ficheiro contém toda a estrutura do componente e as regras necessárias para consumir conteúdo a partir de um URL de uma API externa ou um ficheiro JSON que contenha os dados a serem apresentados.
2. Ficheiro CSS – O conteúdo deste ficheiro diz respeito aos estilos que alteram o visual do componente.
3. Ficheiro HTML – Contém o exemplo prático de como o componente deve ser chamado na página.

O utilizador poderá criar o componente tendo em conta que o mesmo receberá dois parâmetros, um que diz respeito ao URL do CSS e outro que diz respeito ao URL da API externa ou JSON no caso da anterior não existir. Na Figura 3 é descrita a definição do componente no ficheiro HTML bem como os atributos "CSS" e "JSON" que serão passados para serem aplicados ao componente.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <titleonly-poster css="./titleOnly.css" json="../dados.json"></ti-
tleonly-poster>
  <script src="./titleOnly.js"></script>
</body>
</html>
```

Figura 3 - Exemplo prático da chamada ao componente

Na eventualidade de não ser associado um URL para uma API externa é possível fazer o upload de um 4º ficheiro que diz respeito a um ficheiro JSON. O conteúdo deste ficheiro será utilizado para popular o componente com os respetivos dados.

4.2 Modelo de dados estruturados (formato JSON do poster)

Um poster locativo é conteúdo visual e público criado com o propósito de atender a uma necessidade de informação concreta. Os posters são compostos por conteúdo *web* estruturado e *templates web* flexíveis que permitem a apresentação de diferentes secções de um poster numa grande variedade de contextos. Na Figura 4 esta representado o poster por meio de um JSON que define a estrutura semântica do poster nele é definido o id do poster, o nome de quem o pública, o título do mesmo e por fim as secções do poster. Estas secções serão apresentadas no poster e para cada uma das mesmas é definido o nome das várias partes utilizadas no componente dessa mesma secção e o URL para a API externa que fornece os dados que o componente irá consumir, ou então me do ficheiro com os dados. É de salientar que este JSON que diz respeito a estrutura do poster deve ser aplicado do ficheiro HTML de forma a ser apresentado aquando da abertura do poster.

```
{
    "ID":"P203",
    "publisher":"andre",
    "PosterTitle":"Poster sobre a empresa BRAVAL",
    "sections":[{
        "js":"information.js",
        "css":"information.cs",
        "jsonld":"../jsons/braval.json"
  }]
}
```

Figura 4 - Exemplo de JSON para especificação do poster

4.3 Estrutura do poster

O objetivo desta secção é mostrar a estrutura escolhida e o funcionamento da criação do poster. No decorrer desta secção será explicado o propósito de cada parte e alguns detalhes técnicos relacionados. Na sua essência um poster pode conter apenas uma secção ou várias. Sendo assim, podem existir um ou mais ficheiros de estilo (CSS), JSON, JS.

Com base nos requisitos definidos na secção 3.2 o poster consiste num conjunto de ficheiros, entre eles um ficheiro HTML onde os diferentes componentes serão definidos. Os ficheiros de estilo (CSS) que são aplicados nas diferentes secções dos posters e os respetivos ficheiros JS onde os componentes se encontram definidos.

Para além dos ficheiros anteriormente descritos deve ser assumido no ficheiro HTML uma forma de fazer a apresentação dos diferentes componentes, ficando esta aplicação ao critério do criador do poster. Também deve existir no ficheiro HTML uma *script tag* com o JSON que diz respeito ao conteúdo do poster uma vez que o poster deve ser pronto a visualizar no momento em que é aberto. Na Figura 5 está representado o diagrama de fluxo para o desenvolvimento dos posters. Os diferentes processos serão explicados nas seguintes secções.

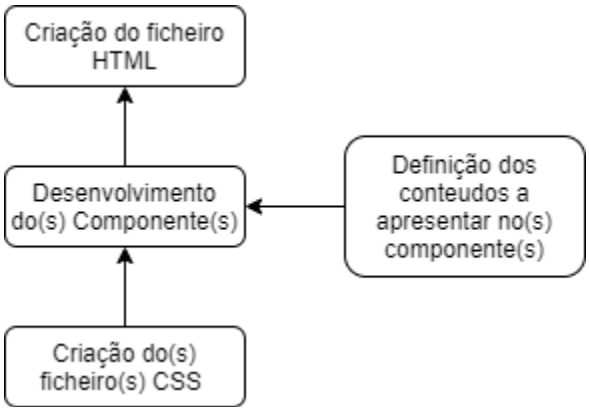


Figura 5 - Diagrama de fluxo para a criação de posters

4.3.1 Definição dos conteúdos a apresentar no(s) componente(s)

Esta componente poderá englobar um ou mais JSON´s, uma vez que, o poster pode ter mais do que uma secção. O conteúdo destes JSON´s diz respeito aos dados que serão apresentados em cada secção do poster. Na Figura 6 pode ser visto um exemplo de um JSON que será utilizado para popular uma secção do poster. Dependo de como o criador implementar o poster este conteúdo poderá ser representado nos respetivos ficheiros JSON ou então consumidos por meio de uma API externa.

```
{
    "@context": "https://schema.org",
    "@type": "Organization",
    "address": {
      "@type": "PostalAddress",
      "addressLocality": "Portugal, Póvoa de Lanhoso",
      "postalCode": "4830-166",
      "streetAddress": "Rua do Aterro"
    },
    "image":"../images/braval.jpg",
    "email": "Email: braval@braval.pt",
    "faxNumber": "Número de fax: 253639229",
    "name": "BRAVAL",
    "telephone": "Número de telefone: 253639220",
    "description":"A BRAVAL ..."
}
```

Figura 6 - Exemplo de documento JSON-LD

4.3.2 Desenvolvimento do(s) componente(s)

Neste processo são desenvolvidos os diferentes *web components* que serão utilizados no poster como secções. Dentro destes ficheiros é especificada a estrutura do componente e definido os atributos que estão

conectados ao ficheiro HTML. Sendo assim, é definida uma variável "*template*" onde se escreve a estrutura do componente em HTML e também se faz a ligação para o ficheiro de CSS externo que contém os diferentes estilos a ser aplicados ao componente como se pode ver na Figura 7. Na criação do *web component* também é importante que se coloque o seu "*shadow mode*" para aberto de forma que o CSS apenas se aplique a este componente e não interfira com outros *templates* Figura 8.

```
const template = document.createElement('template');
template.innerHTML = `
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="company.css">

<div class="card">
<img src="" alt="Avatar">
<div class="container">
  <h4><slot name="name"/></h4>
  <p><slot name="email"/></p>
  <p><slot name="phone"/></p>
  <p><slot name="fax"/></p>
  <p><slot name="description"/></p>
</div>
</div>
`;
```

Figura 7 - Definição da estrutura do componente

```
class Description extends HTMLElement {
  constructor() {
    super();
    this.attachShadow({ mode: 'open' });
    this.shadowRoot.appendChild(template.content.cloneNode(true));
    this.shadowRoot.querySelector('img').src = this.getAttribute('im-
age');

  }

}
window.customElements.define('company-poster', Description);
```

Figura 8 - Attach Shadow mode e criação do componente

4.3.3 Criação do(s) ficheiro(s) CSS

O conteúdo destes ficheiros é composto pelos estilos que serão aplicados ao *template* definido no ficheiro JS, sendo assim, dependendo do número de componentes criados pode existir um ou vários ficheiros deste tipo. Visto que o criador do poster terá conhecimento dos respetivos conteúdos do poster pode eventualmente definir apenas um ficheiro de estilo que possa ser aplicado em todos os componentes do poster. Na Figura 9 é apresentado o conteúdo de um desses ficheiros CSS.

```css
@import url(https://fonts.googleapis.com/css?family=Open+Sans);
*{
    font-family: Arial, "Helvetica Neue", Helvetica, sans-serif;
}
.card {
    box-shadow: 0 4px 8px 0 rgba(0,0,0,0.2);
    transition: 0.3s;
    width: 60%;
    border-radius: 5px;
    display: grid;
    grid-template-columns: 1fr 2fr;
    grid-gap: 10px;
    margin: auto ;
  }
.card:hover {
    box-shadow: 0 8px 16px 0 rgba(0,0,0,0.2);
  }
img {
    display: block;
    margin-left: auto;
    margin-right: auto;
    border-radius: 5px 5px 0 0;
  }
.container {
    padding: 2px 16px;
  }
img {
    max-width: 100%;
    height: auto;
  }
  h4 {
    text-align: center;
  }
```

Figura 9 - Conteúdo do ficheiro CSS

4.3.4 Criação do ficheiro HTML

O ficheiro HTML é necessário para expor o(s) *template*(s) criado(s) nas fases anteriores e por fim fazer a exibição do poster. Como dito em 4.2 deve-se aplicar na cabeça do documento o JSON que define a estrutura do poster. Posteriormente ou por definição à mão ou utilizando de um *script* devem ser chamados os diferentes componentes criados permitindo assim visualizar o poster. Uma vez que, este ficheiro serve como intermediário para passar a informação que se encontra dentro das secções do JSON do poster com os dados estruturados para o *template*. A relação entre um poster e um *web component* surge da necessidade de se criar uma forma de conduzir a informação. Nesse caso, para criar um poster, é necessário que haja pelo menos um ou mais *web components* para escolher, que possam acomodar o tipo de dados em questão. Na Figura 10 é demonstrado a aplicação do JSON com o conteúdo do poster representado na

cabeça do documento e na Figura 11 tem-se um exemplo de como recolher os dados das seções e aplicá-las no documento.

```
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <script id="json" type="application/json">
    {
      "ID": "P203",
      "publisher": "andre",
      "PosterTitle": "Poster sobre a empresa BRAVAL",
      "sections": [{
        "js": "information.js",
        "css": "information.cs",
        "jsonld": "../jsons/braval.json"
      }]
    }
  </script>
</head>
```

Figura 10 - Conteúdo do ficheiro HTML

```
<script>
    // get json data from de script tag about the poster
    var jsonData = JSON.parse(document.getElementById("json").textContent)
    var  conteudo;
    console.log(jsonData)
    //fetch data to populate component
        fetch(jsonData.sections[0].jsonld)
          .then(response => {
            return response.json();
          })
          .then(function (data) {
            conteudo = data
            console.log(conteudo)
            document.body.innerHTML = `<company-poster image=${conteudo.image}>
                                    <div id="1" slot="name">${conteudo.name}</div>
                                    <div id="2" slot="email">${conteudo.email}</div>
                                     <div id="3" slot="phone">${conteudo.phone}</div>
                                     <div id="4" slot="fax">${conteudo.fax}</div>
                                     <div id="5" slot="description">${conteudo.descri
ption}</div>
                                  </company-poster>`
});
</script>
```

Figura 11 - script para adicionar as secções ao HTML

4.4 Abordagem inicial para criação de um componente

Para consolidar a estrutura definida nos pontos anteriores foi desenvolvido um novo componente neste caso com o objetivo de demonstrar informação relativa a uma pessoa. Para o efeito foi utilizado o JSON-LD disponível em "*schema.org*".

4.4.1 JSON-LD do componente *person*

Na Figura 12 estão representados os dados que definem uma determinada pessoa utilizando os dados estruturados. Está informação será aplicada no componente. Uma das principais vantagens da utilização destes dados estruturados previamente é que já são amplamente utilizados em diferentes *websites*, possibilitando desenvolver componentes que consomem este tipo específico de dados.

```
{
    "@context": "https://schema.org",
    "@type": "Person",
    "colleague": [
      "http://www.xyz.edu/students/alicejones.html",
      "http://www.xyz.edu/students/bobsmith.html"
    ],
    "email": "mailto: jane-doe@xyz.edu",
    "image": "../images/janedow.jpg",
    "jobTitle": "Professor",
    "name": "Jane Doe",
    "telephone": "Telephone: (425) 123-4567",
    "url": "http://www.janedoe.com"
}
```

Figura 12 - JSON-LD de person

4.4.2 JS do componente *person*

Numa segunda fase foi necessário desenvolver um componente capaz de utilizar e mostrar as informações que lhe são passadas e que seguem a estrutura definida no ponto anterior. Na Figura 13 é possível ver a definição do *template* que vai ser utilizado para apresentar a informação relativa a pessoa bem como o construtor onde se define toda a funcionalidade que o elemento vai ter quando instanciado e por fim é feito o registo do elemento bem como a classe que o define.

```
const template = document.createElement("template");
template.innerHTML = `
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="person.css">

<div class="card">
  <img src="" alt="avatar" style="width:100%">
  <h1><slot name="name"/></h1>
  <p class="title"><slot name="jobTitle"/></p>
  <p><slot name="email"/></p>
  <p><slot name="phone"/></p>
 <button> <p><slot name="url"/></button></p>
</div>
`;

class Description extends HTMLElement {
  constructor() {
    super();
    this.attachShadow({
      mode: "open",
    });
    this.shadowRoot.appendChild(template.content.cloneNode(true));
    this.shadowRoot.querySelector("img").src = this.getAttribute("im-
age");

  }
}
window.customElements.define("person-poster", Description);
```

Figura 13 - JS de person

4.4.3 HTML do componente *person*

Numa última fase criou-se o ficheiro HTML que liga o JSON-LD ao componente lendo os dados e colocando os mesmos nas áreas definidas do componente como é possível ver na Figura 14.

```
<script>
    var jsonData = JSON.parse(document.getElementById("json").textContent)
    var conteudo;
    fetch("./person1.json")
      .then(response => {
        return response.json();
      })
      .then(function (data) {
        conteudo = data
        document.body.innerHTML = `<person-poster image=${conteudo.image}>
                                    <div id="1" slot="name">${conteudo.name}</div>
                                    <div id="2" slot="jobTitle">${conteudo.jobTitle}</div>
                                    <div id="3" slot="email">${conteudo.email}</div>
                                     <div id="4" slot="phone">${conteudo.telephone}</div>
                                     <div id="5" slot="url">${conteudo.url}</div>
                                  </person-poster>`
      });
  </script>
  <script src="./person.js"></script>
```

Figura 14 - HTML de person

4.4.4 Visualização do componente *person*

No final deste processo tem-se como produto um componente que permite visualizar a informação relativa a uma determinada pessoa seguindo a estrutura de dados descrita no JSON-LD. A apresentação dos dados no *browser* utilizando o componente desenvolvido bem como o CSS escolhido é observável Figura 15.

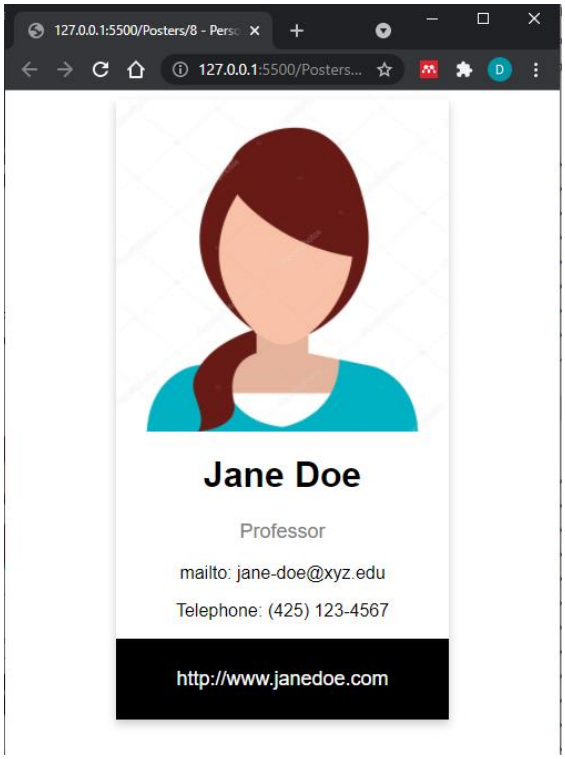


Figura 15 - Resultado final do componente person

Tendo em conta que esta é uma primeira abordagem através do desenvolvimento da mesma consegue-se ver a forma como as diferentes partes se ligam entre si tendo também chegado a conclusões em aspetos que podem ser melhorados, neste exemplo apenas foi utilizado um componente para representar um tipo de dados e o ficheiro HTML já contém bastantes linhas para o efeito. Quando existirem mais partes para serem visualizadas, o ficheiro ainda se tornará mais difícil de ler e conseguir acompanhar as diferentes partes. Outro aspeto é que o ficheiro HTML também tem conhecimento dos dados que serão apresentados, sendo necessário desenvolver um *script* que cria o respetivo componente e coloca os dados nos respetivos *slots*.

4.5 Abordagem final para criação de componentes

Nesta segunda abordagem decidiu-se proceder a uma alteração na forma como os componentes devem ser criados, de maneira a poder colmatar o problema descoberto na primeira abordagem. O problema encontrado na fase inicial foi que, quer o ficheiro HTML, quer o ficheiro JS que diz respeito ao componente em si detinham regras de negócio para a criação do componente. Sendo assim, desenvolveu-se uma nova forma para a criação dos componentes que passa por ter todas as regras necessárias para a criação do mesmo dentro do ficheiro JS.

Para exemplificar esta nova forma de definir os componentes irá utilizar-se o exemplo anterior podendo ver assim as alterações realizadas nos diferentes ficheiros.

4.5.1 Alterações feitas no HTML do componente *person*

No exemplo anterior o ficheiro HTML era encarregue de importar os dados e em seguida teria de colocar esses dados nos devidos *slots* de forma que o componente depois soubesse quais os dados que cada campo deveria utilizar. Nesta nova versão no HTML apenas é definido a *tag* que diz respeito ao componente que vai ser utilizado e é passado por meio de variáveis a localização do ficheiro JSON ou o URL para a API externa com os dados que posteriormente serão utilizados Figura 16.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <person-poster css="./person.css" json="./person1.json"></person-
poster>
  <script src="./person.js"></script>
</body>
</html>
```

Figura 16 - Novo HTML de person

4.5.2 Alterações do JS do componente *person*

Neste ficheiro existiram mudanças grandes do ponto de vista do funcionamento do componente, uma vez que o mesmo agora é encarregue de ir buscar os dados e também fazer a associação dos dados aos campos certos. Sendo assim, deixou de se usar os *slots* e passaram-se a usar ids dentro de cada lugar onde deve existir informação proveniente do JSON-LD Figura 17.

```
const person = document.createElement("template");
person.innerHTML = `
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" id="css" >

<div class="card">
  <img id="imageSpace"src="" alt="avatar" style="width:100%">
  <h1 id="nome"></h1>
  <p class="title" id="profissao"></p>
  <p id="email"></p>
  <p id="telefone"></p>
 <button> <p id="url"></button></p>
</div>
`;
```

Figura 17 - Novo JS de person

Em seguida alterou-se o corpo do construtor do componente criando variáveis que contém a informação passada ao chamar o componente esta informação diz respeito ao ficheiro CSS e ao JSON que será utilizado pelo componente. A função "*fetch*", que anteriormente ficava no ficheiro HTML e é encarregue de buscar os dados, fica agora dentro deste construtor e através dos ids definidos são associados os respetivos dados. Na Figura 18 é possivel ver o conteúdo do ficheiro e as alterações realizadas.

```
class Person extends HTMLElement {
  constructor() {
    super();
    this.attachShadow({
      mode: "open",
    });
    this.shadowRoot.appendChild(person.content.cloneNode(true));

    //variavel que contem o shadowroot
    var sombra = this.shadowRoot;
    //variavel que contem o url para o css desejado
    var estilo = this.getAttribute("css");
    //variavel que contem url para o json desejado
    var linkFetch = this.getAttribute("json");

    //pede os dados a api
    fetch(linkFetch)
      .then((response) => response.json())
      .then((data) => {
        // coloca os dados dentro dos respetivos lugares no html
        sombra.getElementById("imageSpace").setAttribute("src", data.im-
age);
        sombra.getElementById("nome").innerHTML = data.name;
        sombra.getElementById("profissao").innerHTML = data.jobTitle;
        sombra.getElementById("email").innerHTML = data.email;
        sombra.getElementById("telefone").innerHTML = data.phone;
        sombra.getElementById("url").innerHTML = data.url;
        //console.log(data)
        return data;
      })
      .catch((error) => {
        alert(error)
        return error;
      });

      //coloca o ficheiro definido para o css
      sombra.getElementById("css").setAttribute("href",estilo)
  }
}
window.customElements.define("person-poster", Person);
```

Figura 18 - Construtor do componente

Apesar destas alterações o resultado final continua a ser o mesmo encontrado na secção 4.4.4, no entanto torna-se uma mais-valia porque desta forma a "inteligência" do componente por assim dizer encontra-se toda dentro de um ficheiro podendo assim a geração do HTML ser facilmente automatizada.

4.6 Bibliotecas de slides em JS

Esta secção tem como objetivo apresentar diferentes bibliotecas e *scripts* escritos em JS para apresentar conteúdo *web* em formato de slides. Foi então realizada uma pesquisa sobre o assunto tendo sido encontradas as seguintes bibliotecas:

- WebSlides: Consiste em um conjunto de ficheiros que permite desenvolver apresentações em HTML.

- Reveal js: É um framework de apresentações em HTML open source.
- How to – Slideshow by W3Schools: Um *script* bastante simples, que permite criar um carroussel de vários slides.
- Splide.js: É um carroussel leve, poderoso e flexível escrito em JS.
- Remark.js: É uma ferramenta de apresentação de slides simples.
- Impress.js: É também uma framework para apresentação de informação que utiliza CSS3 e é inspirada no prezi.

4.7 Desenvolvimento de poster com utilização de slides

Neste projeto procurou-se criar um poster que utiliza vários *web components* em simultâneo, podendo se percorrer os mesmos de diferentes formas, a importância de realizar este estudo advém do facto da utilização dos slides para criar posters que contenham mais do que um componente. Para o efeito foi preciso definir o que cada componente iria representar.

Com a conclusão da definição da estrutura que o poster deve ter, deu-se início a implementação do trabalho. Primeiramente foram criados 5 *web components*, distintos entre si, um poster que apresenta um título, um que apresenta um título mais subtítulo, título e conteúdo, conteúdo e imagem e um com imagem e legenda.

Após o desenvolvimento dos componentes foi criado um *script* em JS cujo propósito é o de ler o JSON que contém a informação relativa ao poster.

4.7.1 Criação do JSON

Primeiramente foi necessário desenvolver um JSON que contém a informação que irá ser visualizada em cada um dos componentes criados para este exemplo na Figura 19 encontram-se representados estes dados.

```
{
    "title":"TUB - Transportes Urbanos de Braga",
    "subtitle":"A paragem de autocarro mais utilizada de Braga",
    "image":"https://encrypted-tbn0.gstatic.com/ima-
ges?q=tbn:ANd9GcTQ40kmPGITjJ7kLPLBplV9zPr-kwsGrMPg8A&usqp=CAU",
    "content":"Lorem  et feli... ",
    "caption":"Paragem construida no seguimento de ..."
}
```

Figura 19 - JSON para o poster de slides

4.7.2 Componentes desenvolvidos

Com base no JSON definido foram criados 5 componentes distintos seguindo a estrutura final definida na secção 4.5. Os componentes são os seguintes:

- titleOnly - Este componente demonstra apenas um determinado título.

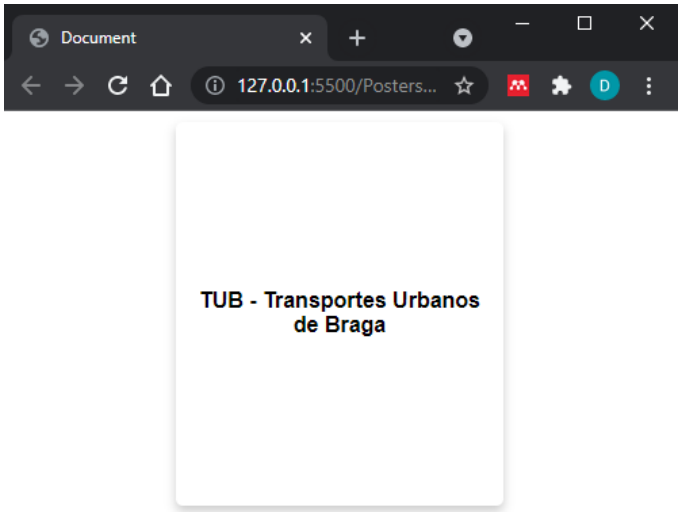


Figura 20 - Poster titleOnly

- title – Ao contrário do primeiro componente este já demonstra um título e um subtítulo.

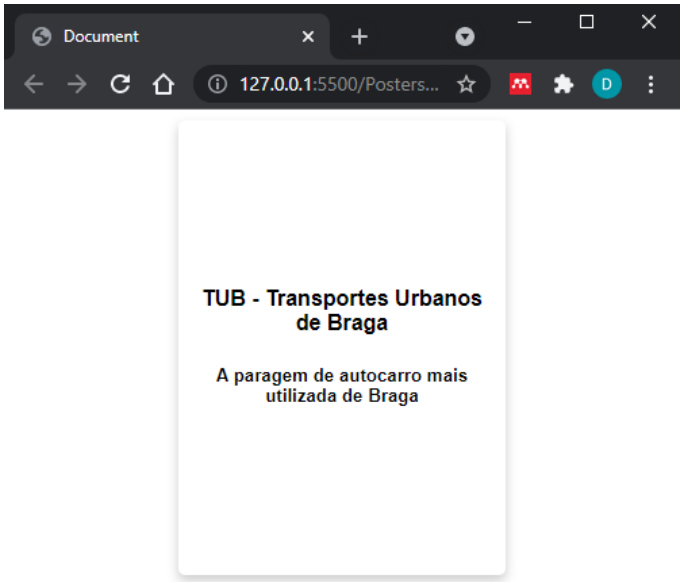


Figura 21 - Poster title

- titleContent – Permite visualizar um determinado título e conteúdo que seja escrito para o mesmo.

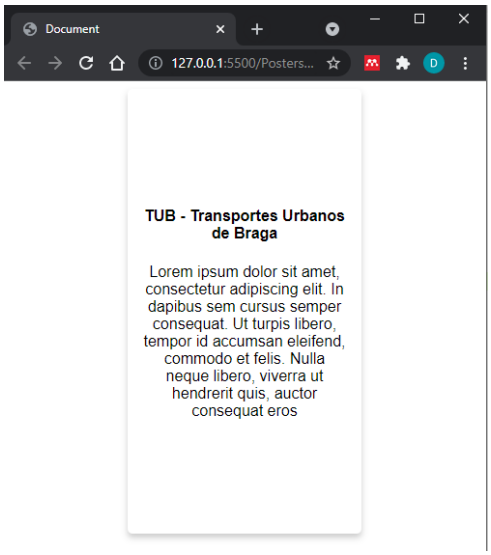


Figura 22 - Poster titleContent

- contentCaption – O componente mostra conteúdo e uma legenda do mesmo.

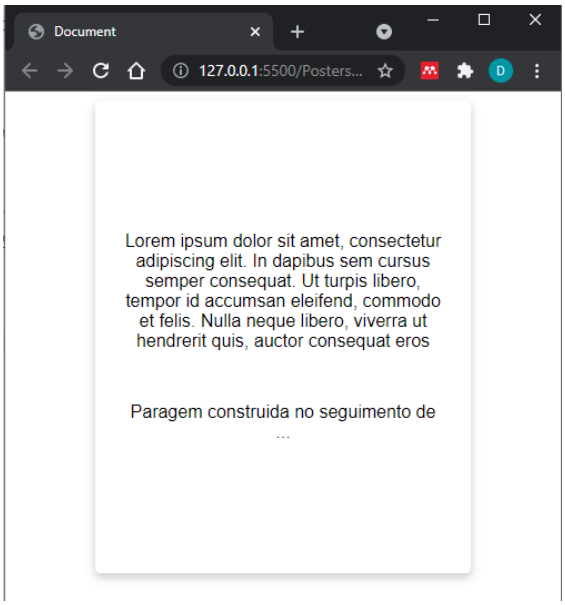


Figura 23 – Poster contentCaption

- imageCaption – Para visualizar imagens com legenda.

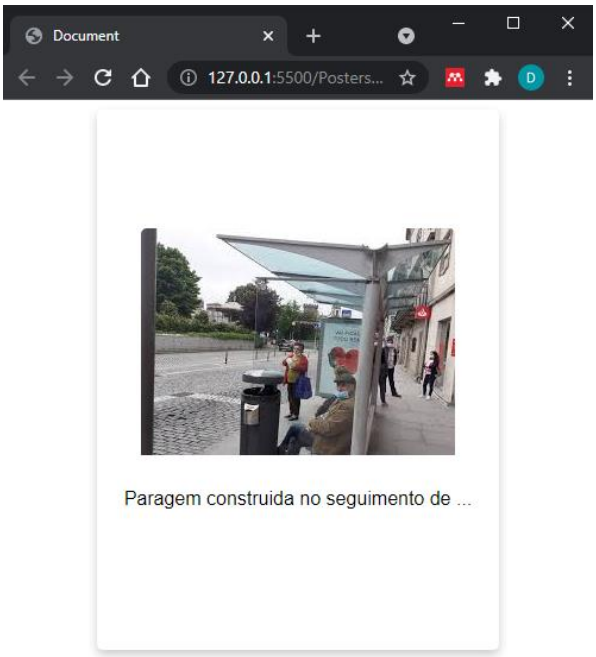


Figura 24 - Poster imageCaption

4.7.3 Definição do HTML

Para poder ver o poster são utilizadas as respetivas *tags* que dizem respeito ao componente, fazendo passar os URL´s para o ficheiro de estilo e ficheiro de JSON com os conteúdos a ser consumidos. Também é necessário definir os *scripts* dos componentes. Na Figura 25 é possível ver a aplicação dos vários componentes e os seus parâmetros e a chamada dos ficheiros JS externos que dizem respeito a cada *web component*.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
</head>
<body>
    <titleonly-poster css="./1 - titleOnly/titleOnly.css" json="./da-
dos.json"></titleonly-poster>
    <script src="./1 - titleOnly/titleOnly.js"></script>
    <title-poster css="./2 - title/title.css" json="./dados.json"></title-
poster>
    <script src="./2 - title/title.js"></script>
    <titlecontent-poster css="./3 - titleContent/titleCon-
tent.css" json="./dados.json"></titlecontent-poster>
    <script src="./3 - titleContent/titleContent.js"></script>
    <contentcaption-poster css="./4 - contentCaption/contentCap-
tion.css" json="./dados.json"></contentcaption-poster>
    <script src="./4 - contentCaption/contentCaption.js"></script>
    <imagecaption-poster css="./5 - imageCaption/imageCap-
tion.css" json="./dados.json"></imagecaption-poster>
    <script src="./5 - imageCaption/imageCaption.js"></script>
</body>
</html>
```

Figura 25 - HTML do poster

4.7.4 Visualização final do poster

O resultado final é um poster constituído por 5 slides onde cada um diz respeito a um componente individual e que é independente dos outros, tal como é possível observar na Figura 26, onde está representado o resultado quando aberto no *browser*.

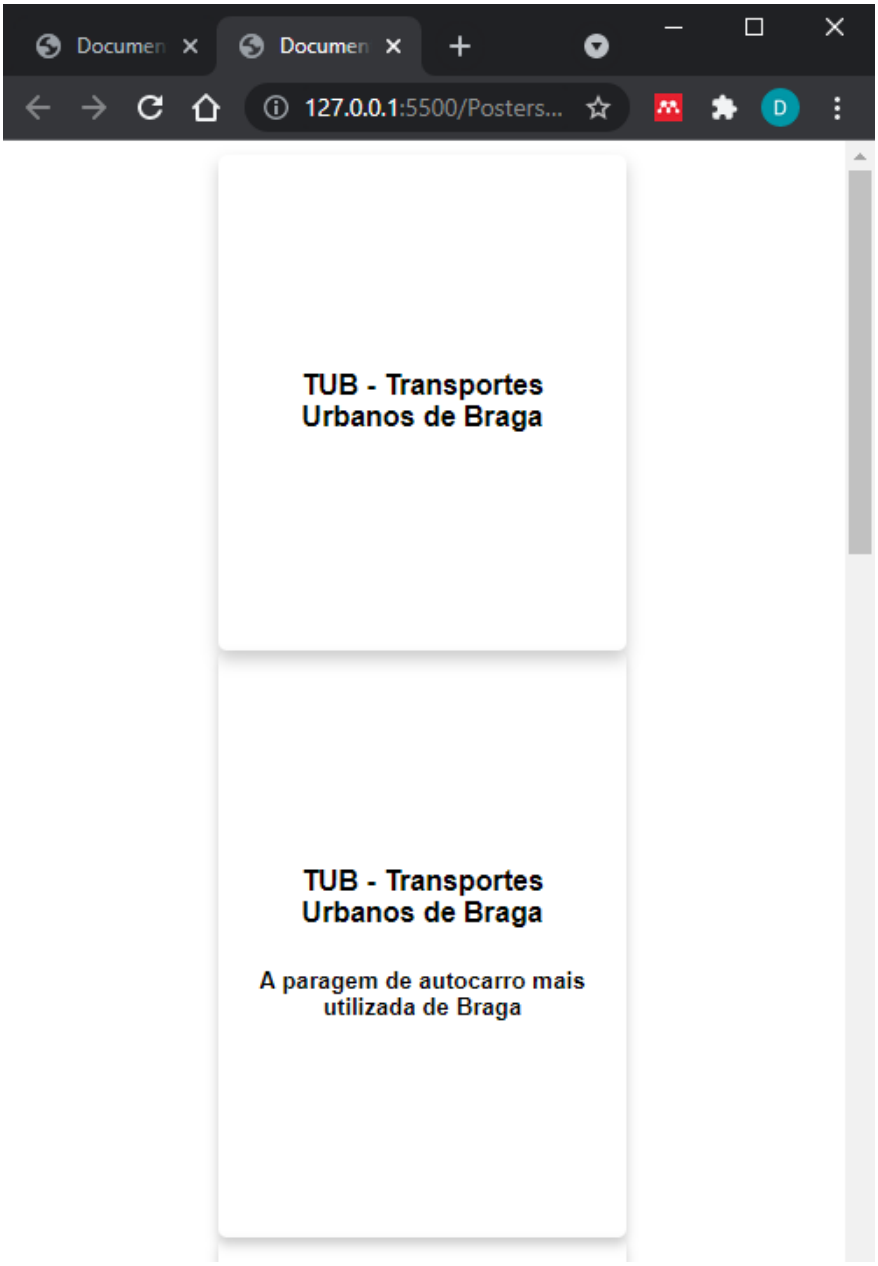


Figura 26 - Poster com 5 componentes

4.7.5 Implementação de visualização com slides

Como foi dito anteriormente o poster pode ser apresentado de diferentes formas nesta secção é feita a descrição da utilização de um *script* encontrado na pesquisa realizada sobre diferentes bibliotecas slides. Este *script* permite a apresentação dos componentes em formato de slides, onde se pode interagir com o poster por meio de botões de maneira a alternar entre os slides.

Numa primeira fase foi necessário criar um *script* que permite chamar os diferentes componentes e alterar a forma como os mesmos são representados na página. Assim, tal como descrito na Figura 27, envolve-se as *tags* dos respetivos componentes dentro de uma "*div*" com uma classe chamada de "mySlides".

```
<div class="mySlides fade">
    <titleonly-poster css="./1 - titleOnly/titleOnly.css" json="./dados.json"></titleOnly-
poster>
</div>
```

Figura 27 - Exemplificação da definição do componente

No seguimento, na Figura 28 é possível ver o *script* que permite navegar as diferentes secções do poster. Neste *script* é realizada uma procura no ficheiro HTML de todos os elementos que utilizam a classe "mySlides" em seguida, é colocado o display do elemento para "*none*" o que esconde todos os componentes, mas em última instância é colocado o display do primeiro componente a ser visualizado para "*block*" o que faz com que o mesmo se torne visível.

```
<script>
        var slideIndex = 1;
        showSlides(slideIndex);

        function plusSlides(n) {
            showSlides((slideIndex += n));
        }

        function currentSlide(n) {
            showSlides((slideIndex = n));
        }
        function showSlides(n) {
            var i;
            var slides = document.getElementsByClassName("mySlides");
            var dots = document.getElementsByClassName("dot");
            if (n > slides.length) {
                slideIndex = 1;
            }
            if (n < 1) {
                slideIndex = slides.length;
            }
            for (i = 0; i < slides.length; i++) {
                slides[i].style.display = "none";
            }
            slides[slideIndex - 1].style.display = "block";
        }
    </script>
```

Figura 28 - Script para slides

O resultado final é um poster que permite ser navegado por meio de botões, esta é uma forma bastante simples de podermos definir a apresentação das diferentes secções do poster. Na Figura 29 é apresentado o exemplo pratico deste poster no *browser* sendo possivel navegar as diferentes secções criadas.

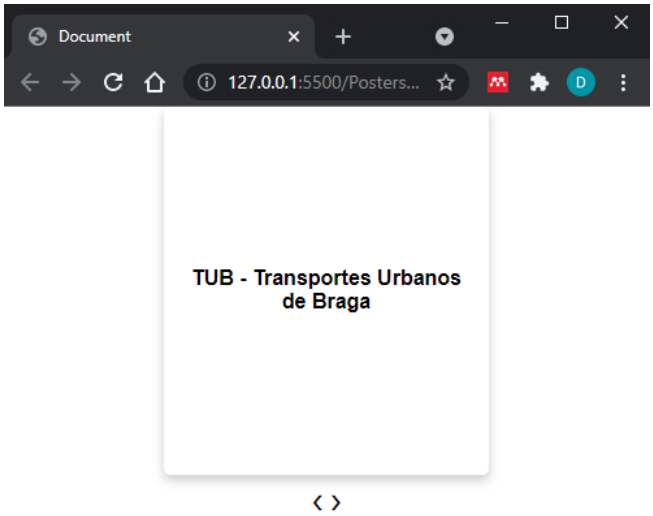


Figura 29 - Poster final com 5 slides

4.7.6 Implementação de visualização utilizando webslides

De forma a consolidar a criação de poster com diferentes formas de apresentação foi utilizada uma biblioteca JS cujo propósito é facilitar a criação de apresentações por meio de HTML. A biblioteca escolhida chama-se webslides e permite fazer apresentações de forma simples e rápida podendo está também ser customizada de diferentes formas.

A implementação para este caso é bastante simples, primeiramente é necessário adicionar os estilos e os *scripts* referentes a biblioteca. Ao contrário do exemplo descrito em 4.7.5 onde se utilizou uma classe para definir um slide, esta biblioteca utiliza a *tag* do HTML "*section*" e assume que estas são um slide único a ser representado de alguma forma, sendo assim, envolve-se as *tags* dos componentes dentro de uma *tag* "*section*" como se pode ver na Figura 30.

```
<section>
      <titleonly-poster css="./1 - titleOnly/titleOnly.css" json="./dados.json"></ti-
tleonly-poster>
</section>
```

Figura 30 - Exemplificação da definição do componente com webslides

Esta implementação dá origem a um poster moderno bastante intuitivo, sendo o mesmo representado na Figura 31. De forma automática é criado um elemento visual onde se podem ver o número de slides que o poster tem e setas que permitem navegar os mesmos. Por fim, também apresenta uma forma diferente

de percorrer os slides que é através de um menu onde se pode escolher qual slide ver não tendo de seguir uma ordem em específico, tal como se pode observar na Figura 32.

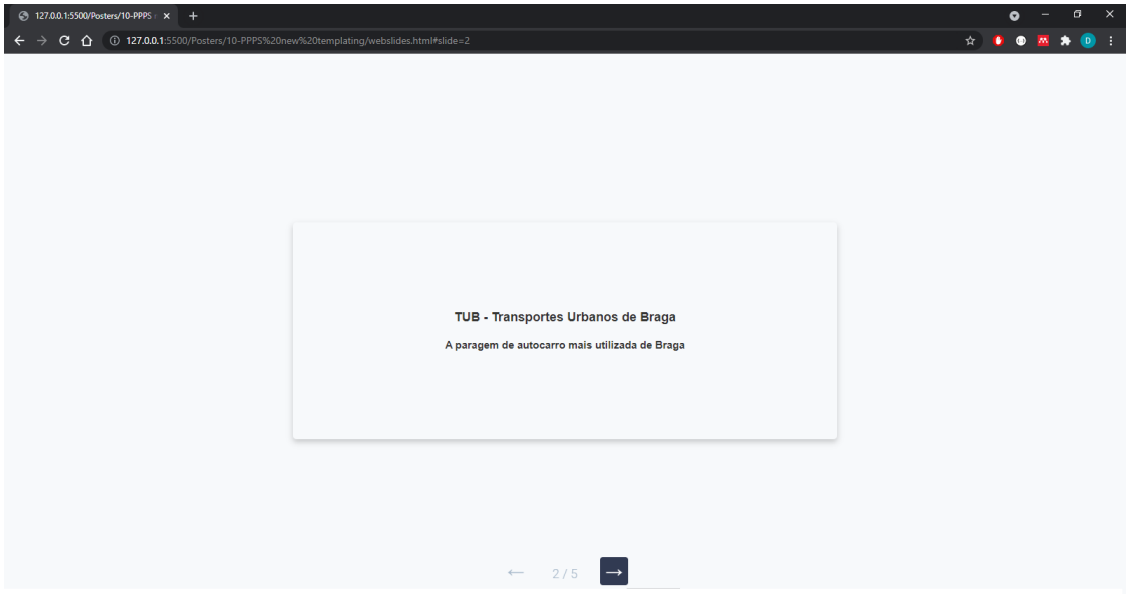


Figura 31 - Visualização com webslides

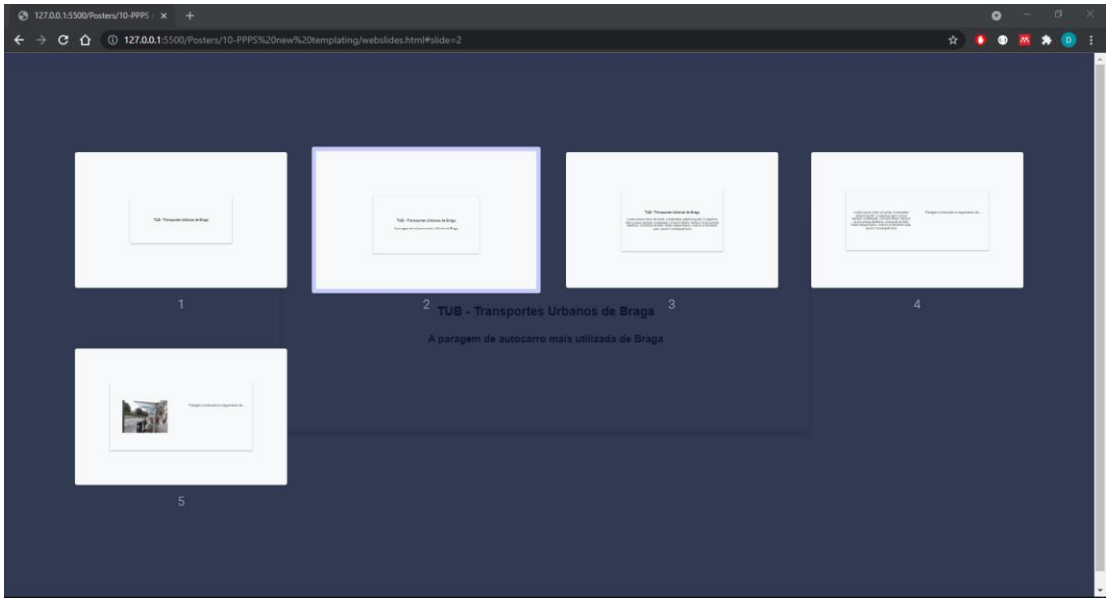


Figura 32 - Escolha de slides

5 Caso de Estudo – Transportes Urbanos de Braga (TUB)

Para realizar o caso de estudo escolheu-se os Transportes Urbanos de Braga (TUB). A razão pela qual foi escolhido foi o facto de possuir dados relativos a vários aspetos do negócio dos transportes urbanos seguindo o padrão proposto pela Google. O GTFS traduzido para português é a Especificação Geral sobre *Feeds* de Transporte Público, este define um formato comum para os horários dos transportes públicos e as informações geográficas relacionadas.

Para o desenvolvimento deste projeto, este caso de estudo tem como objetivo colocar em prática todos os conceitos abordados anteriormente, testando assim a criação de posters e a utilização dos *web components*, fazendo uso dos dados disponibilizados pela TUB para fazer a geração em grande escala de posters. Por exemplo, com base no ficheiro que diz respeito às rotas existentes em Braga produzir um poster para todas elas contendo informação descritiva da rota.

Através desta implementação é possível verificar o processo de criação de posters em grande escala com utilização do GTFS, podendo posteriormente ser aplicado a outras entidades que também fazem uso deste padrão.

5.1 Fase 0: Definição de componentes e posters

Após a disponibilização dos dados por parte da TUB foi realizada uma análise dos dados de maneira a definir quais os Posters que poderão ser criados com base na informação existente. Cada um desses posters, dependendo das secções com que venha a ser definido poderá ter um ou mais *web components*.

Assim, numa fase inicial foi necessário definir quais os posters e quais componentes que iriam ser desenvolvidos para este caso de estudo. Na Tabela 1 estão descritos os componentes que foram desenvolvidos e as funções que os mesmos desempenham.

Tabela 1 - Componentes e as respetivas funções

Componente(s)	Função
Dados de uma Paragem	Este componente permite visualizar informação relativa a uma paragem com os dados existentes em stops.txt
Dados de uma Rota	Este componente permite visualizar informação relativa a uma rota com os dados existentes em routes.txt
Próximas Viagens em determinada Paragem	Com este componente é possivel ver as próximas quatro viagens com base no tempo em que se visualiza o poster com um *countdown* do tempo de chegada
Próximas paragens na viagem	Com a junção de diferentes ficheiros é possível ver a informação da respetiva viagem em que se está no momento bem como a rota a que pertence e o percurso com as respetivas paragens

Com base nos componentes definidos foram criados diferentes posters contendo um ou mais componentes mencionados, sendo que esta relação pode ser devidamente consultada na Tabela 2, onde é possível observar para cada poster quais os respetivos componentes utilizados.

Tabela 2 - Posters e componentes utilizados

Posters	Componente(s)
Sobre paragem e viagens que passam nessa mesma paragem (Anexo 1)	• Dados de uma Paragem • Próximas Viagens em determinada Paragem

Sobre rotas (Anexo 2)	• Dados de uma Rota
Sobre o percurso do autocarro (Anexo 3)	• Próximas paragens na viagem

Nas seguintes fases será descrito o processo desde a leitura e processamento dos dados ao desenvolvimento de uma API para fornecer os dados que posteriormente serão utilizados para gerar JSON´s que por sua vez serão utilizados para criar os posters. Numa fase final são apresentados dois cenários de utilização onde podem ser vistos dois posters distintos que fazem uso dos componentes mencionados anteriormente.

5.2 Fase 1: Pré-processamento de dados

Nesta fase foi feito o processamento dos dados, fazendo assim a leitura dos ficheiros do *feed* GTFS e posteriormente a sua inserção nas respetivas tabelas na base de dados.

Este processamento é feito através de um *script* desenvolvido em *Python* (*3.10.1 Documentation*, n.d.) onde com a utilização da biblioteca *pandas* (*pandas documentation — pandas 1.3.5 documentation*, n.d.) que permite analisar e manipular dados de forma rápida fazendo a leitura dos respetivos ficheiros para um dataframe é possível ler os dados e inserir os mesmos numa base de dados de forma simples. Este dataframe é uma estrutura de dados, podendo os mesmos ser dos mais vários tipos. Tendo sido definido o dataframe para cada ficheiro de modo a inserir os dados na base de dados é necessário criar uma conexão com a mesma, a criação das respetivas tabelas é realizada por meio de uma função disponibilizada pelo próprio dataframe que recebendo como parâmetros o nome da tabela e a conexão, automaticamente cria a tabela. Na Figura 33 é apresentado o exemplo deste processo, onde é feita a leitura do ficheiro que diz respeito as rotas e este é associado ao respetivo dataframe "routes", posteriormente é inserido na base de dados com um parâmetro que caso já exista uma tabela na base de dados realiza a substituição dos dados.

```
# Create dataframe
routes = pd.read_csv("routes.txt", encoding='iso-8859-1')
# Convert dataframe to sql table
routes.to_sql('routes', engine, index=False, if_exists='replace')
```

Figura 33 - Código para processar o ficheiro de rotas

Com os vários ficheiros do *feed* GTFS foi possível criar tabelas para as rotas, as paragens, as viagens, os tempos de paragens.

No entanto com a análise realizada aos dados conseguiu-se perceber que alguns ficheiros têm uma relação entre si podendo assim ser feita a ligação entre os mesmos de maneira a retirar mais valor dos dados. Por meio da função de união existente no *pandas* foi realizada a união dos ficheiros "routes.txt", "stops.txt", "trips.txt" e "stop_times.txt", criando assim um dataframe que contém a informação relativa aos quatro ficheiros, o exemplo de como esta união é feita pode ser vista na Figura 34.

```
#merge dos ficheiros
df_merge1 = pd.merge(routes, trips, on="route_id")
df_merge2 = pd.merge(stops, stop_times, on="stop_id")
df_merge3 = pd.merge(df_merge2, df_merge1, on="trip_id")
```

Figura 34 - União dos diferentes ficheiros

Para facilitar a leitura dos dados e também a posterior utilização dos mesmos foram realizadas transformações na tabela de forma a ordenar os dados. Para este efeito foram criados dois dataframes distintos, o primeiro será utilizado para deter informação relativa a uma determinada paragem e quais os autocarros que lá passarão, e o segundo dataframe contém informação relativa ao autocarro e qual a paragem em que irá parar de seguida. Na Figura 35 é possível observar, como se realizam estas transformações por meio das funções disponibilizadas pelo *pandas*.

```
#ordenacao para saber paragens e ordem em que o autocarro para
proxima_paragem = df_merge3.sort_values(["trip_id", "stop_sequence"], ascend-
ing = (True, True))
proxima_paragem.to_sql('next_stop', engine, index=False, if_exists='replace')


#ordenacao para saber quais o proximo autocarro em x paragem
proximo_autocarro = df_merge3.sort_values(["stop_id", "arrival_time"], as-
cending = (True, True))
proximo_autocarro.to_sql('next_bus', engine, index=False, if_exists='re-
place')
```

Figura 35 - Ordenação dos dados para facilitar a leitura

Concluída a execução do *script* tem-se na base de dados as tabelas com os dados referentes a um determinado aspeto do GTFS. Na Figura 36 são apresentados os atributos da tabela das paragens. Por sua vez os atributos da tabela das rotas podem ser consultados na Figura 37.

#	Nome	Tipo
1	stop_id	bigint(20)
2	stop_code	bigint(20)
3	stop_name	text
4	stop_lat	double
5	stop_lon	double
6	zone_id	bigint(20)

Figura 36 - Tabela para paragens

#	Nome	Tipo
1	route_id	bigint(20)
2	agency_id	bigint(20)
3	route_short_name	bigint(20)
4	route_long_name	text
5	route_type	bigint(20)

Figura 37 - Tabela para rotas

Na Figura 38 e na Figura 39 encontram-se respetivamente representados os atributos das viagens e dos tempos em que se para em cada paragem

#	Nome	Tipo
1	route_id	bigint(20)
2	service_id	bigint(20)
3	trip_id	bigint(20)
4	direction_id	bigint(20)

Figura 38 - Tabela para viagens

#	Nome	Tipo
1	trip_id	bigint(20)
2	arrival_time	text
3	departure_time	text
4	stop_id	bigint(20)
5	stop_sequence	bigint(20)
6	timepoint	bigint(20)

Figura 39 - Tabela para tempos de paragem

Por fim, na Figura 40 são expostos os atributos que existem nas tabelas onde se realizou a junção dos diferentes ficheiros. A variação entre as duas tabelas é a ordem com que os dados aparecem nas respetivas tabelas.

#	Nome	Tipo
1	stop_id	bigint(20)
2	stop_code	bigint(20)
3	stop_name	text
4	stop_lat	double
5	stop_lon	double
6	zone_id	bigint(20)
7	trip_id	bigint(20)
8	arrival_time	text
9	departure_time	text
10	stop_sequence	bigint(20)
11	timepoint	bigint(20)
12	route_id	bigint(20)
13	agency_id	bigint(20)
14	route_short_name	bigint(20)
15	route_long_name	text
16	route_type	bigint(20)
17	service_id	bigint(20)
18	direction_id	bigint(20)

Figura 40 - Junção de tabelas para criação de conteúdos

5.3 Fase 2: Desenvolvimento da API de dados GTFS

Tendo sido realizada a fase de processamento de dados foi desenvolvida uma aplicação *web* utilizando o *Flask* (*Welcome to Flask — Flask Documentation (2.0.x)*, n.d.). O *Flask* é um framework escrito em *Python* que permite desenvolver aplicações *web*. Esta aplicação servirá como API onde fazendo pedidos a URL´s específicos à aplicação é retornado o JSON com os dados que serão utilizados por um componente específico que faz parte de uma das secções do poster.

O *endpoint* "/stops" retorna uma lista com todas as paragens existentes. Por sua vez o *endpoint* "stops/<id>" retorna informação relativa a paragem que for introduzida na pesquisa. De igual forma criou-se os mesmos *endpoints* para as rotas e as viagens, seguindo a estrutura "/routes" e "/trips" para retornar nas respetivas listas informação referente a todas as rotas e viagens respetivamente. Quanto a pedidos de uma única rota e uma única viagem tem-se os *endpoints* "/routes/<id>" e "/trips/<id>". Por fim também

foram criados dois *endpoints* para poder aceder a informação das tabelas onde se realizaram transformações, sendo assim, tem-se o *endpoint* "/next_bus/<id>" que retorna uma lista onde se pode ver a informação relativa ao próximo autocarro que irá parar em determinada paragem, no caso do *endpoint* "/next_stop/<id>" recebe-se uma lista com informação relativa a todas as paragens no percurso do autocarro.

5.4 Fase 3: Criação dos JSON´s com a estrutura do poster

Uma vez criada a base de dados e a API a terceira fase passou por gerar o conteúdo estruturado que diz respeito ao poster. Neste passo utilizando os "*endpoints*" definidos na secção 5.3 foi desenvolvido um *script* para gerar os JSON´s referentes aos posters finais que foram definidos.

Realizando a iteração sobre os dados e adicionando os mesmos a um dicionário *Python* com a estrutura pretendida, sendo possível observar na Figura 41 o exemplo do *script* que permite fazer a criação dos JSON´s para as diferentes rotas existentes e em seguida adicionar as mesmas em um ficheiro que contém uma lista com todas elas.

```
import requests
import json
response = requests.get("http://127.0.0.1:5000/routes")
data = response.json()
guardar = []
for i in range(len(data)):
    dicionario = {
        "ID": data[i]["route_id"],
        "publisher":"daniel",
        "PosterTitle": data[i]["route_long_name"],
        "sections":[{
            "html":"InfoRoute.html",
            "js":"InfoRoute.js",
            "css":"InfoRoute.css",
            "jsonld": f"http://127.0.0.1:5000/routes/{data[i]['route_id']}",
            "componente":"route-poster"
        }]
    }
    guardar.append(dicionario)
# Serializing json
json_object = json.dumps(guardar, indent=4)

# Writing to sample.json
with open("ficheirosEscalaRoutes.json", "w") as outfile:
    outfile.write(json_object)
```

Figura 41 - Exemplo de script para geração de json para posters de rotas

5.5 Fase 4: Criação automática de posters

Agora que se criaram os conteúdos estruturados para os posters e foram devidamente colocados num único ficheiro, desenvolveu-se o *script* que permite gerar os posters. Com este *script* é possível automatizar a geração dos posters seguindo a estrutura previamente definida. Este *script* consome o ficheiro JSON que contém todos os posters e faz o processamento dos mesmos criando para cada objeto uma pasta que diz respeito a um poster específico contendo os ficheiros HTML, JS, CSS e JSON utilizados pelo mesmo.

A geração do ficheiro HTML é feita com recurso ao "*jinja*", um mecanismo que permite conceber *templates* (*Jinja — Jinja Documentation (3.0.x)*, n.d.). O uso deste mecanismo permite colocar os dados estruturados do poster na forma em que é pretendida e que os mesmos sejam escritos dentro do ficheiro HTML, sendo o *script* desenvolvido para gerar os posters representado na Figura 42.

```
def gerador(input_file, folder):
    with open(input_file, encoding="utf8") as f:
        data = ujson.load(f)
    for i in range(len(data)):
        # criar pasta do poster
        if not os.path.exists(f'{folder}/{data[i].get("ID")}'):
            os.makedirs(f'{folder}/{data[i].get("ID")}')
        # iterar sobre todos os componentes que foram definidos para ser utilizados no poster
        sections = data[i].get("sections")
        for n in range(len(sections)):
            # copia ficheiros js
            jsoriginal = f'componentes/{sections[n].get("js")}'
            jsnovo = f'{folder}/{data[i].get("ID")}/{sections[n].get("js")}'
            shutil.copyfile(jsoriginal, jsnovo)
            # copia ficheiros css
            cssoriginal = f'componentes/{sections[n].get("css")}'
            cssnovo = f'{folder}/{data[i].get("ID")}/{sections[n].get("css")}'
            shutil.copyfile(cssoriginal, cssnovo)
        # Serializing json
        dic = data[i]
        json_object = ujson.dumps(dic, indent=4)
        # guarda num ficheiro json a informacao usada para criar o poster
        with open(f'{folder}/{data[i].get("ID")}/{str(data[i].get("ID")) +".json"}', "w") as out
file:
            outfile.write(json_object)
        root = os.path.dirname(os.path.abspath(__file__))
        # diretoria onde se encontra o template do jinja para o html
        env = Environment(loader=FileSystemLoader(root))
        template = env.get_template('templateParaJinja.html')
        # nome do ficheiro html
        filename = f'{folder}/{data[i].get("ID")}/{str(data[i].get("ID")) + ".html"}'
        # remove ficheiros js duplicados para nao existir erros ao mostrar o poster
        list_scripts = {sections[sections.index(i)]["js"] for i in sections}
        s = list(list_scripts)
        # print(s)
        # cria e guarda o ficheiro html
        with open(filename, 'w') as fh:
            fh.write(template.render(
                data=data[i]["sections"],
                script=s,
                poster=json_object
))
```

Figura 42 - Código para gerar os posters com base no json

No momento em que é criado o ficheiro HTML são passados três parâmetros que dizem respeito ao próprio JSON do poster, uma lista com os nomes dos ficheiros JS que deverão ser importados para os respetivos componentes e por fim uma lista que diz respeito às secções do posters que será utilizada pelo

"jinja" para gerar as tags dos respetivos componentes. Na Figura 43 e Figura 44 é possível observar os placeholders para os dados que são passados na criação do ficheiro HTML.

```
<script id="json" type="application/json"> {{poster}} </script>
```

Figura 43 - Adição do json no template

```
<div class="slideshow-container">

    {% for dados in data %}
           <div class="mySlides fade">
                <{{dados.componente}} css='{{dados.css}}' json = '{{da-
dos.jsonld}}' > </{{dados.componente}}>
           </div>
    {% endfor %}

    </div>
    {% if data|length > 1 %}
          <div class="center">
                <a class="prev" onclick="plusSlides(-1)">&#10094;</a>
                <a class="next" onclick="plusSlides(1)">&#10095;</a>
          </div>
    {% endif %}


    {% for scripts in script %}
        <script src='{{scripts}}'></script>
    {% endfor %}
```

Figura 44 - Conteúdo do body do template do jinja

5.6 Fase 5: Estudo de cenários de utilização

Esta secção tem como objetivo demonstrar possíveis aplicações do GTFS em união com os posters locativos. Sendo assim, foram criados alguns cenários de utilização, onde foram desenvolvidos posters específicos para um dado caso de uso em que a fonte de dados é proveniente da API desenvolvida na secção 5.3.

5.6.1 Cenário 1 – Próximas paragens

Este cenário procura responder à situação em que existe a necessidade de saber quais as próximas paragens de uma determinada rota. Para realizar este cenário inicialmente foi necessário desenvolver um poster. Este poster é constituído por um *web component* que faz um pedido a API para o *endpoint* "/next_stop/<id>", onde "<id>" diz respeito ao id que identifica a viagem.

O pedido realizado retorna uma lista com todas as paragens pertencentes à viagem, posteriormente estes dados são colocados numa tabela contendo informação relativa à viagem em questão, a rota que é

percorrida na mesma e a listagem das paragens por ordem crescente nas respetivas horas de chegada. O resultado final deste poster pode ser observado na Figura 45.

Figura 45 - Poster próximas paragens na viagem

5.6.2 Cenário 2 – Próximas viagens a passar na paragem

O segundo cenário procura responder à situação em que um possível utilizador se encontra em determinada paragem, ou então, gostaria de saber quais as próximas viagens que passam em determinada paragem. Para este efeito foi criado um poster que utiliza dois *web components.*

O primeiro *web component* faz um pedido a API para o *endpoint* "/stops/<id>", onde "<id>" diz respeito ao id que identifica a paragem em questão. O pedido retorna informação relativa a paragem que é utilizada para popular o *web component* com o nome da paragem, o respetivo id e as coordenadas geográficas da mesma. A visualização deste componente no poster é apresentada na Figura 46.

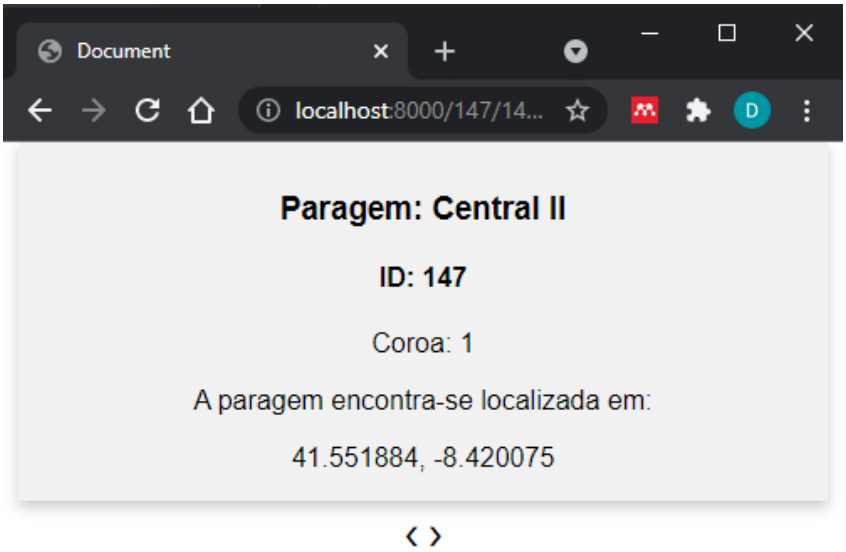


Figura 46 – Poster próximas viagens, componente 1

O segundo *web component* faz um pedido a API para o *endpoint* "/next_bus/<id>", onde "<id>" diz respeito ao id que identifica a paragem em questão. O pedido realizado retorna uma lista com todas as viagens que passam por aquela paragem, em seguida o componente faz uma série de funções, como filtrar as viagens que já ocorreram até ao momento em que é feito o pedido e por fim mostra apenas às cinco viagens mais próximas do tempo no momento de abertura do poster atualizando constantemente o tempo esperado até a chegada do autocarro. O resultado deste componente no poster pode ser visto na Figura 47.

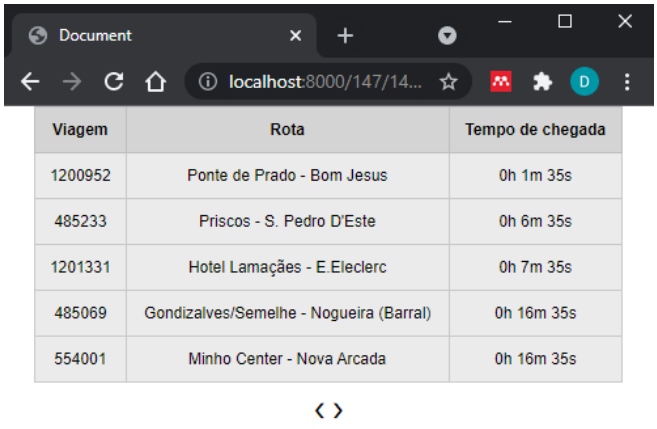


Figura 47 - Poster próximas viagens, componente 2

5.7 Avaliação do caso de estudo

A avaliação deste caso de estudo é de extrema importância para este projeto de dissertação, uma vez que, através dela é possivel medir o sucesso da especificação criada para os posters locativos e se os requisitos dos mesmos foram cumpridos. Para avaliar foram então criados dois tipos de posters diferentes, um poster onde se pode ver o percurso de uma determinada viagem e um segundo poster com informação relativa a uma determinada paragem e quais as próximas viagens que lá passam.

Estes posters foram avaliados tendo em conta os requisitos definidos para os posters locativos na secção 3.2. Sendo assim, são apresentados os vários requisitos e como os mesmos foram ou não cumpridos:

- **Requisito 1: *Web***

  Este requisito foi cumprido uma vez que os posters locativos são efetivamente representados na *web* e utilizam tecnologias padrão como HTML, CSS, JS e JSON. Também foi feito uso dos *web components* que é um conjunto de tecnologias que permite a criação de elementos customizáveis presente em qualquer browser. Quanto a especificação do poster está também apresenta um número reduzido de regras para garantir o correto funcionamento dos posters.

- **Requisito 2: *Sandbox***

  O segundo requisito também foi cumprido cada poster tem os seus conteúdos e ficheiros separados de outros posters. Para além disso ao utilizar *web componentes* para definir as diferentes secções dos posters todas as funcionalidades destas secções ficam separadas não afetando o resto do poster.

- **Requisito 3: *CrossChannel***

  Este terceiro requisito foi cumprido através da utilização de técnicas disponíveis no CSS, com a utilização de media queries é possivel definir diferentes *breakpoints* para tamanhos de ecrãs diferentes sendo o conteúdo do poster automaticamente ajustado com base no dispositivo em que esta a ser visualizado.

- **Requisito 4: *Multipart***

  Este último requisito também foi cumprido como se pode constatar no poster criado na secção 5.6.2 onde o poster tem duas secções apresentando conteúdos distintos.

A implementação deste caso de estudo permitiu concluir que a especificação proposta para os posters locativos consegue responder aos requisitos propostos. A definição de uma estrutura prévia para os conteúdos que o poster locativo deve ter quando aplicada a uma fonte de dados, neste caso o GTFS *static* da TUB, permite gerar conteúdos *web* capazes de serem apresentados em qualquer dispositivo e aplicáveis em diversas situações do dia-a-dia das pessoas.

Com este caso de estudo também foi possível utilizar os dados GTFS *static* e desenvolver todo um processo para o processamento e utilização dos dados. Fazendo numa fase inicial a leitura dos ficheiros que compõem o *feed* GTFS, a criação de tabelas na base de dados tendo em conta o conteúdo dos respetivos ficheiros e numa fase final a criação de uma API que consome os dados e os fornece às pessoas no sentido de utilizarem estes dados no contexto que quiserem.

5.7.1 Limitações e desafios

Uma das limitações existentes no desenvolvimento deste caso de estudo deveu-se ao facto de numa fase inicial se estar a utilizar um *feed* GTFS onde os dados não apresentavam certos atributos. E com base nestes atributos era possível tirar partido dos diferentes ficheiros e criar ligações que permitiam tirar conclusões sobre os horários em que os autocarros paravam em determinada paragem. Este *feed* GTFS inicial também apresentava os tempos de uma forma que não representava a realidade, uma vez que, os tempos entre paragens eram sempre iguais porque eram calculados com base no tempo que se demorava a chegar da primeira paragem até a última. Esta limitação foi levantada quando se obteve acesso ao *feed* GTFS mais recente da TUB que apresenta os tempos próximos daquilo que é a realidade e com os atributos necessários para tirar conclusões melhores sobre os dados.

Um dos principais desafios encontrados foi o facto de apesar de o formato GTFS ser amplamente utilizado, não é tão simples saber lidar com os dados. Sendo este trabalho mais fácil para as pessoas que detém conhecimento nesta área, podendo tirar o melhor proveito dos dados.

5.7.2 Novas possibilidades

Apesar de no momento em que se deu o desenvolvimento deste caso de estudo ainda não estar disponível o *feed* GTFS *Realtime* da TUB. No futuro será possível desenvolver diferentes cenários de utilização, com dados que são atualizados regularmente e permitem entregar aos utilizadores informações de valor no

seu dia-a-dia, como por exemplo, atualizações sobre o posicionamento dos autocarros em determinado momento o que também permite ver se existem atrasos no serviço.

6 Conclusão

O objetivo deste capítulo é o de apresentar uma conclusão sobre o trabalho que foi realizado neste projeto de dissertação e avaliar se os objetivos propostos foram cumpridos. Também é realizada uma reflexão sobre o trabalho a ser realizado no futuro.

6.1 Conclusões sobre o trabalho

Esta dissertação tem como objetivo especificar e validar, no contexto de instalações em ambiente real, um modelo de poster locativo que permite publicar conteúdo *web* estruturado para ser visualizado em qualquer dispositivo com ligação à internet.

Em relação ao objetivo de propor uma especificação para os posters locativos que define a sua estrutura capaz de responder aos desafios apresentados, este foi concluído com sucesso, pois foi criada uma estrutura que define os posters locativos assente nos requisitos definidos para as várias partes que constituem o poster.

Quanto ao objetivo de desenvolver um gerador de posters locativos que possibilite a criação automática de posters a partir de conteúdos estruturados, também foi conseguido com sucesso. Para conseguir realizar este objetivo foi desenvolvido um *script* que utiliza o conteúdo estruturado do poster locativo e em seguida constrói o ficheiro HTML que contém as secções a ser apresentadas bem como elas deverão ser apresentadas.

Relativamente ao objetivo de desenvolver um caso de estudo de aplicação da especificação no âmbito de um sistema de informação para transportes públicos, é possível afirmar que o *feed* GTFS da TUB foi uma mais-valia para o desenvolvimento de componentes e posters. Proporcionando dados de diferentes aspetos dos transportes públicos que proporcionaram o desenvolvimento de posters distintos.

Em relação ao objetivo de avaliar a aplicação da proposta apresentada no âmbito do caso de estudo, identificando eventuais limitações, desafios e novas possibilidades, conclui-se que o *feed* GTFS possibilitou o desenvolvimento de um caso de estudo que permitiu a aplicação dos vários conceitos abordados ao longo deste projeto de dissertação. Tendo em conta que também foram encontrados alguns desafios bem como pensadas novas possibilidades para futuras aplicações do GTFS.

6.2 Trabalho futuro

Neste projeto de dissertação foram desenvolvidos vários *web components* para diferentes situações. Estes componentes foram utilizados para apresentar determinada informação dentro de um poster locativo, podendo este ter uma ou várias secções. Também se criou um *script* que permite com base no conteúdo estruturado do poster criar o poster e as diferentes partes que o compõem. Com o fim deste projeto é importante salientar o trabalho que se pode realizar no futuro.

Um aspeto importante a ter seguimento é o de desenvolver uma aplicação onde é possível fazer a publicação dos posters. Permitindo adicionar diferentes entidades que queiram publicar posters, e posteriormente gerar os posters por meio dessa aplicação e atribuir um URL ao poster para que possa ser acedido nos respetivos locais para o qual ele foi criado.

Também é importante aprofundar o desenvolvimento de posters locativos com ligações a outros posters. Podendo assim interligar por exemplo, posters cujo contexto seja similar e apresentar ao utilizador outras opções.

Referências Bibliográficas

3.10.1 Documentation. (n.d.). Retrieved December 12, 2021, from https://docs.python.org/3/

Alt, F., Kubitza, T., Bial, D., Zaidan, F., Ortel, M., Zurmaar, B., Lewen, T., Shirazi, A. S., & Schmidt, A. (2011). Digifieds: Insights into deploying digital public notice areas in the wild. *Proceedings of the 10th International Conference on Mobile and Ubiquitous Multimedia, MUM'11*. https://doi.org/10.1145/2107596.2107618

Alt, F., Memarovic, N., Elhart, I., Bial, D., Schmidt, A., Langheinrich, M., Harboe, G., Huang, E., & Scipioni, M. P. (2011). Designing shared public display networks - Implications from today's paper-based notice areas. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *6696 LNCS*, 258–275. https://doi.org/10.1007/978-3-642-21726-5_17

Bizer, C., Heath, T., & Berners-Lee, T. (2009). Linked data - The story so far. *International Journal on Semantic Web and Information Systems*. https://doi.org/10.4018/jswis.2009081901

Burger, M. C. (2015). ChemDoodle Web Components: HTML5 toolkit for chemical graphics, interfaces, and informatics. *Journal of Cheminformatics*, *7*(1), 1–7. https://doi.org/10.1186/s13321-015-0085-3

Coutinho, P., & José, R. (2020). Understanding public displays as a medium for place-based communication: implications from current practices with non-digital displays. *Personal and Ubiquitous Computing*. https://doi.org/10.1007/s00779-019-01362-6

DÁngelo, L. (2012). From posters to e - posters : the evolution of a genre. *University of Reading Language Studies Working Papers*.

Gregg Kellogg, Pierre-Antoine Champin, D. L. (2019). JSON-LD 1.1 – A JSON-based Serialization for Linked Data. [Technical]. *[Technical Report] W3C.*

Jinja — Jinja Documentation (3.0.x). (n.d.). Retrieved December 21, 2021, from https://jinja.palletsprojects.com/en/3.0.x/

José, R., Otero, N., Izadi, S., & Harper, R. (2008). Instant places: Using bluetooth for situated interaction in public displays. *IEEE Pervasive Computing*, *7*(4). https://doi.org/10.1109/MPRV.2008.74

Krug, M., & Gaedke, M. (2015). Smartcomposition: Enhanced web components for a better future of web development. *WWW 2015 Companion - Proceedings of the 24th International Conference on World Wide Web*, 207–210. https://doi.org/10.1145/2740908.2742832

Kunama, N., & Phithakkitnukoon, S. (2017). *GTFS-Viz : Tool for Preprocessing and*. 388–396.

Larrucea, X., Santamaria, I., Colomo-Palacios, R., & Ebert, C. (2018). Microservices. *IEEE Software*. https://doi.org/10.1109/MS.2018.2141030

Murray, R., Thow, M., & Strachan, R. (1998). Visual literacy: Designing and presenting a poster. *Physiotherapy*. https://doi.org/10.1016/S0031-9406(05)63456-6

Nova, N. (2004). Locative Media : a literature review. *Computer*, *February*.

pandas documentation — pandas 1.3.5 documentation. (n.d.). Retrieved December 12, 2021, from https://pandas.pydata.org/docs/index.html

Prommaharaj, P., Phithakkitnukoon, S., Demissie, M. G., Kattan, L., & Ratti, C. (2020). Visualizing public transit system operation with GTFS data: A case study of Calgary, Canada. *Heliyon*, *6*(4), e03729. https://doi.org/10.1016/j.heliyon.2020.e03729

Schmidt, A., Beigl, M., & Gellersen, H. W. (1999). There is more to context than location. *Computers & Graphics*, *23*(6), 893–901. https://doi.org/10.1016/S0097-8493(99)00120-X

Thönes, J. (2015). Microservices. In *IEEE Software*. https://doi.org/10.1109/MS.2015.11

Web Components | MDN. (n.d.). Retrieved December 12, 2021, from https://developer.mozilla.org/en-US/docs/Web/Web_Components

Welcome to Flask — Flask Documentation (2.0.x). (n.d.). Retrieved December 12, 2021, from https://flask.palletsprojects.com/en/2.0.x/

Wilken, R. (2012). Locative media: From specialized preoccupation to mainstream fascination. *Convergence*, *18*(3), 243–247. https://doi.org/10.1177/1354856512444375

Anexo 1 – Poster sobre paragem e viagens que passam nessa mesma paragem

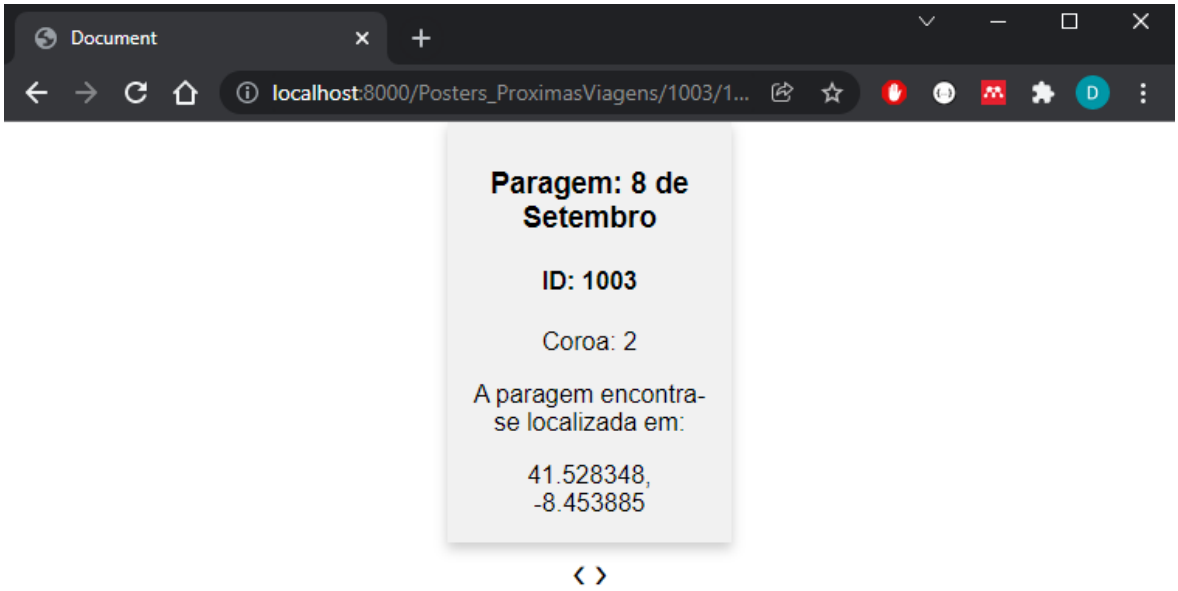


Figura 48 - Primeira secção do poster

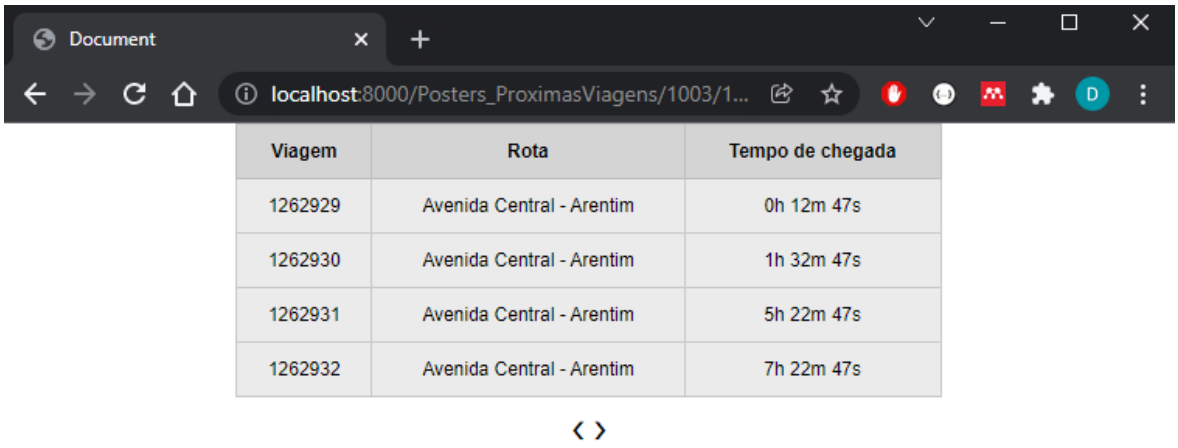


Figura 49 - Segunda secção do poster

Anexo 2 – Poster sobre rotas

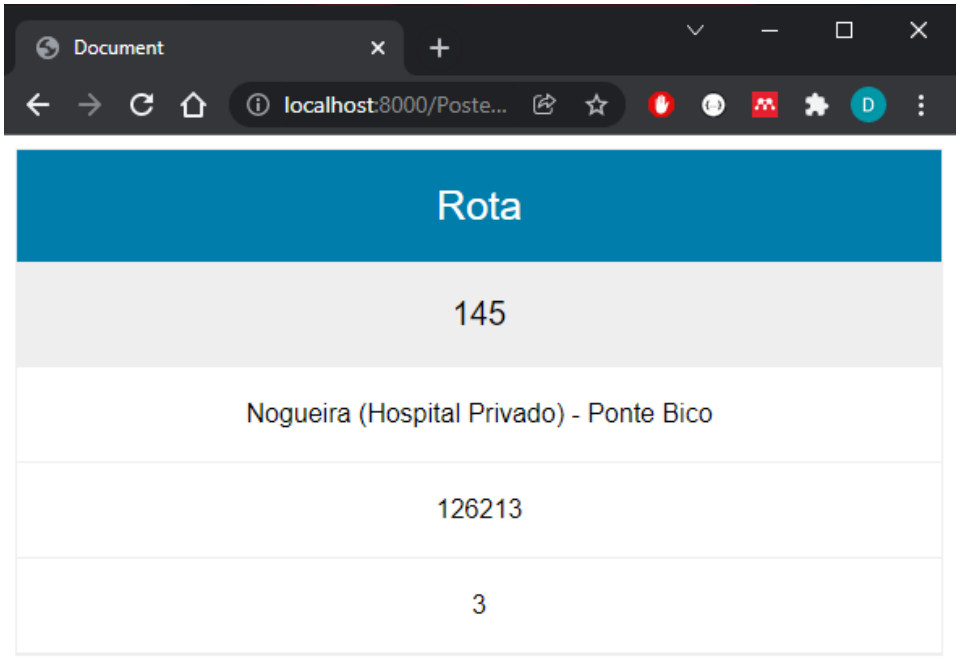
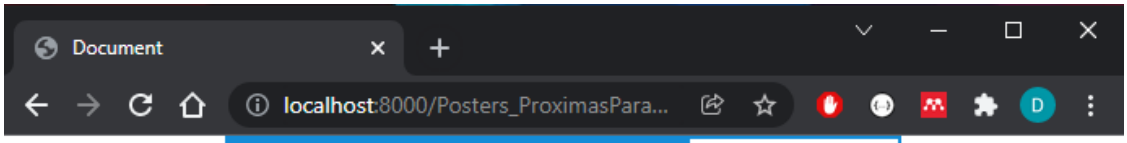


Figura 50 - Poster sobre rotas

Anexo 3 – Poster sobre o percurso do autocarro



Rota	Camélias - Hospital	
Viagem	1108794	
Paragem	**Horas**	**Tempo de Chegada**
2030	Camélias (António Menici Malheiro)	08:00:00
1604	Augusto Veloso	08:01:00
282	Cons Lobato (Monsenhor Airosa)	08:02:00
283	Cons Lobato (Travessa)	08:04:00
2062	Fujacal I	08:04:00
303	Fujacal II	08:05:00
228	Imac Conceição (Fujacal)	08:07:00
261	Liberdade (Igreja S Lázaro)	08:08:00
117	25 de Abril (D Maria II I)	08:10:00
1912	25 de Abril (D Maria II III)	08:11:00
118	Senhora-a-Branca	08:13:00
119	S Victor	08:15:00
120	D Pedro V - I	08:16:00
122	D Pedro V - II	08:18:00
140	Júlio Fragata I	08:20:00
141	Júlio Fragata II	08:21:00
136	Fernando O Guimarães I	08:23:00
1978	Fernando O Guimarães II	08:25:00
135	Quinta Armada (Bairro d Alegria)	08:26:00
1651	Quinta Armada (Escola)	08:28:00
319	Verdosas (B Alegria)	08:28:00
2029	Hospital	08:28:00
188	Hospital (Parque)	08:30:00

Figura 51 - Poster sobre percurso do autocarro