Universidade de Aveiro Departamento de Comunicação e Arte
2018

Rui Filipe da Cunha Martins

Web Components - solução de compatibilidade na implementação de componentes *cross framework*

Universidade de Aveiro Departamento de Comunicação e Arte
2018

Rui Filipe da Cunha Martins

Web Components - solução de compatibilidade na implementação de componentes *cross framework*

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Comunicação Multimédia, realizada sob a orientação científica do Doutor Mário Vairinhos, Professor Auxiliar do Departamento de Comunicação e Arte da Universidade de Aveiro e coorientação do Engenheiro Carlos Figueiredo, Consultor UI/UX da Altice Labs.

Projeto de dissertação para obtenção do grau académico no âmbito do Programa de Bolsas de Investigação da Inova-Ria – Programa GENIUS, autorizado pela FCT – Bolsa de Iniciação Científica (BIC) financiada pela Inova-Ria e desenvolvido na Empresa Altice Labs.

Dedico este trabalho à minha namorada pelo incansável apoio.

o júri

presidente

Prof. Doutor Telmo Eduardo Miranda Castelão da Silva
professor auxiliar da Universidade de Aveiro

Prof. Doutor Diogo Nuno Pereira Gomes
professor auxiliar da Universidade de Aveiro

Prof. Doutor Mário Jorge Rodrigues Martins Vairinhos
professor auxiliar da Universidade de Aveiro

agradecimentos

Por todos estes anos ao meu lado, por ser o meu pilar, me motivar incansavelmente e porque sem ela não teria conseguido, quero agradecer à minha namorada, Tânia. Quero agradecer ao meu orientador, Professor Doutor Mário Vairinhos, por me acompanhar e orientar durante todo este projeto. Quero agradecer à equipa de Usabilidade & User Experience da Altice Labs, em especial à Inês Oliveira, Carlos Figueiredo e Nuno Santos, por me darem a oportunidade de desenvolver este projeto e me fazerem sentir em casa. Quero também agradecer à minha família, ao meu pai, mãe e ao meu irmão, por tornarem este meu percurso académico possível, que espero retribuir por agora em diante. Por fim quero agradecer aos meus amigos, por momentos que nunca esquecerei, que tornaram especial o meu percurso académico.

palavras-chave

Web Components, Desenvolvimento Web, Interoperabilidade, *Cross Framework*

resumo

O leque de *frameworks* de desenvolvimento Web para a vertente *front-end* tem vindo a crescer rapidamente nos últimos anos. Atualmente, a gama considerada é bastante extensa, esperando-se que continue em expansão, com ferramentas cada vez mais evoluídas para servir as necessidades dos programadores. Estas ferramentas facilitam e aceleram todo o processo de desenvolvimento dos utilizadores, ao mesmo tempo que permitem a criação de produtos com maior rigor profissional.

No entanto, num contexto *cross framework*, surgem dificuldades de compatibilidade relativas ao desenvolvimento e implementação de componentes visuais. Cada *framework* implementa a sua estrutura específica, dificultando a reutilização de componentes.

Assim, desenvolvido em contexto de projeto de dissertação, este trabalho tem por objetivo a investigação do potencial dos Web Components como solução de interoperabilidade num contexto *cross framework*.

Para materializar o conhecimento adquirido, e considerando a tecnologia em questão, foi desenvolvido um protótipo para servir as necessidades da Altice Labs, uma vez que esta foi a empresa que forneceu o campo para a investigação. Este protótipo foi testado, discutido com a equipa de Coordenação Tecnológica e apresentado à empresa. Revelou ter potencial para, possivelmente, reformular o processo de desenvolvimento das equipas.

keywords Web Components, Web Development, Interoperability, Cross Framework

abstract

The range of frameworks for front-end web development has been growing rapidly for the last years. Nowadays, this range is quite extensive and it is expected to keep growing, with more evolved tools to serve the needs of the developers. Such tools aim to easen and quicken the development process, as well as allowing creating products with professional rigor.

However, within a cross Framework context, it is expected for compatibility issues to arise when talking about the development and implementation processes for visual components. Each framework implements its own specific structure, invalidating the reuse of patterns when developing components.

As such, and considering the project dissertation context, the objective of the work hereafter presented is to investigate the potential of Web Components as an interoperability solution in a cross framework environment.

In order to materialize the findings, and considering the technology in question, a prototype was developed to serve Altice Labs' needs, as this was the company that provided the field for the investigation. This prototype was tested, discussed with the Technological Coordination team, and presented to the company. It has shown to have potential to possibly affect the development process of the teams.

Índice

Índice de Figuras III

Índice de Tabelas IV

Lista de Acrónimos V

1. Introdução 12

2. Enquadramento teórico 13

2.1 Tecnologias *web* 13

2.2 Web Components 16

2.3 *Cross frameworking* 26

2.4 Altice Labs 28

3. Metodologia 31

4. Processo de implementação do Web Component 37

4.1 Requisitos técnicos e funcionais 38

4.2 Análise à tecnologia Web Components 44

4.2.1 HTML Imports 44

4.2.2 HTML Templates 46

4.2.3 Custom Elements 47

4.2.4 Shadow DOM 48

4.3 Análise à biblioteca de desenvolvimento de Web Components 51

5. Processo de avaliação do Web Component 53

5.1 Caracterização dos participantes 55

5.2 Resultados 56

5.2.1 Focus group 56

5.2.2 Questionário 58

6. *Guidelines* para desenvolvimento de Web Components 67

7. Conclusões 69

Referências Bibliográficas 73

Apêndices 77

Índice de Figuras

Figura 1 - Análise de tendências de *frameworks*. Fonte: Ian Allen, (2018). 17
Figura 2 - Shadow DOM. Fonte: Swisher & Mills, (2018). 19
Figura 3 - Suporte nativo dos Web Components. Fonte: “webcomponents.org,” (2018). 20
Figura 4 - Ciclo de vida de frameworks. Fonte: Ian Allen, (2018). 26
Figura 5 - Modelo genérico da metodologia D&R. 32
Figura 6 - Componente Tabs. 38
Figura 7 - Componente Tabs inserido no produto NetQ. 39
Figura 8 - Separadores do componente Tabs. 41
Figura 9 - Botão de adição de separadores. 42
Figura 10 - Ações adicionais dos separadores. 42
Figura 11 - Suporte dos *browsers* ao *standard* HTML Imports. 45
Figura 12 - Estilos contidos numa *tag* "custom-style". 49
Figura 13 - Histograma da pergunta 2. 59
Figura 14 - Histograma da pergunta 3. 59
Figura 15 - Histograma da pergunta 4. 60
Figura 16 - Histograma da pergunta 6. 61
Figura 17- Resultados do modelo de aceitação da tecnologia. 62
Figura 18 - Histograma da pergunta 19. 63
Figura 19 - Histograma da pergunta 21. 64
Figura 20 - Histograma da pergunta 22. 65
Figura 21 - Histograma da pergunta 24. 66

Índice de Tabelas

Tabela 1 - Comparação de bibliotecas de Web Components. 24
Tabela 2 - Participantes da recolha de dados. 33
Tabela 3 - Planeamento da recolha de dados. 35
Tabela 4 - Resumo das versões do protótipo. 38
Tabela 5 - Funcionalidades do componente Tabs original. 40
Tabela 6 - Etapas da sessão de recolha de dados. 54
Tabela 7 - Caracterização dos participantes. 55

Lista de Acrónimos

API - Application Programming Interface

CSS - Cascading Style Sheets

CLI - Command-line Interface

DOM - Document Object Model

FTP - File Transfer Protocol

FUXI – Framework User eXperience and Interface

GWT - Google Web Toolkit

HTML - Hypertext Markup Language

HTTP - HyperText Transfer Protocol

JS - JavaScript

JSP - JavaServer Pages

JSX - JavaScript XML

NPM - Node Package Manager

TAM - Technology Acceptance Model

W3C - World Wide Web Consortium

web - World Wide Web

1. Introdução

A quantidade de *frameworks* de desenvolvimento *web* é vasta e todos os anos surgem novas que prometem ser melhores que as anteriores. No entanto, num meio empresarial, a constante substituição de *frameworks* não é de todo viável uma vez que os ciclos de vida dos produtos são normalmente longos e a tecnologia usada num produto tende a manter-se por pelo menos 5 anos de forma a justificar o investimento inicial. O *overhead*[1] de mudança tecnológica num produto é sempre grande e não é estranho encontrar produtos neste meio ainda com tecnologias ultrapassadas face às mais recentes tendências.

Num contexto em que produtos desenvolvidos por equipas diferentes utilizam tecnologias diversas, mas que, no entanto, devem apresentar ao utilizador final uma interface funcional semelhante, é essencial promover essa partilha de experiência com a utilização de uma *framework* comum. A *framework* criada pela Altice Labs é a FUXI (Framework User eXperience and Interface). Esta permite manter os diversos componentes que servem vários universos tecnológicos, tantos quantos os produtos que a empresa deve dar suporte enquanto equipa de usabilidade transversal. Trata-se de uma tarefa complexa pois obriga à criação e manutenção de um mesmo componente em diferentes tecnologias.

O cenário ideal seria a criação de componentes numa única tecnologia e que cada uma das *frameworks* os pudesse usar, independentemente dos processos internos impostos por cada tecnologia em particular, quer para manutenção, quer para integração dos produtos.

[1] *Overhead* – Custo ou despesa de manutenção de um negócio não atribuíveis a produtos ou itens individuais.

Pergunta de investigação

Tendo em conta o que foi abordado nos tópicos anteriores, o cerne da investigação passa por abordar uma problemática específica, nomeadamente:

"No contexto das tecnologias web aplicadas à implementação de componentes cross framework, quais as potencialidades de uma arquitetura de sistema baseada em Web Components?"

Assim, pretende-se explorar o potencial dos Web Components como solução para ultrapassar os obstáculos gerados num contexto de desenvolvimento *web cross framework* e conhecer quer os desafios que esta abordagem impõe, quer as eventuais dificuldades.

Para tal será criado um protótipo funcional com o objetivo de avaliar se esta tecnologia é capaz de servir as necessidades da empresa.

Finalidades e objetivos

O objetivo desta investigação é conceber e avaliar uma solução de interoperabilidade relativa à criação de componentes visuais que serão implementados num ambiente de desenvolvimento *cross framework*. Para tal foram definidas metas:

- **Investigar o estado da arte da utilização dos Web Components no problema do *cross frameworking***

Os Web Components são um padrão definido pelo W3C (World Wide Web Consortium). Estes têm como finalidade a criação de componentes da interface gráfica reutilizáveis e transversais a qualquer *framework* pois funcionam de forma encapsulada do restante código.
Posto isto é essencial a exploração de casos de estudo relacionados com este tema, de forma a enquadrar o projeto em questão com o estado atual da problemática.

- **Desenvolver um protótipo funcional que servirá de prova de conceito baseado em Web Components**

Concluída a investigação necessária, será desenvolvido um protótipo funcional de alta-fidelidade. Este irá responder a requisitos definidos pelas equipas de desenvolvimento da Altice Labs, que serão essenciais para avaliar a viabilidade de uma solução de desenvolvimento baseada em Web Components.

- **Testar e avaliar as potencialidades de uma solução baseada em Web Components**

A solução irá ser avaliada por forma a responder a várias questões pertinentes, tendo em conta a investigação e o protótipo funcional. Esta será testada em dois cenários tecnologicamente diferentes, em produtos que tenham

sido desenvolvidos com recurso a uma *framework*, por exemplo Angular 5, e em produtos que tenham sido criados de forma nativa, por forma a avaliar custos e obstáculos na integração de Web Components. Este protótipo terá uma abordagem específica à empresa, que irá permitir compreender as vantagens e desvantagens dos mesmos.

2. Enquadramento teórico

2.1 Tecnologias *web*

Para uma melhor compreensão sobre as tecnologias *web* (World Wide Web) é indispensável uma reflexão do que esta é e os seus conceitos base. A *web* é a vertente mais proeminente da Internet que pode ser definida como um sistema tecno-social de interação humana baseada em redes tecnológicas (Aghaei, 2012). Esta já passou por várias fases que devem ser relevadas pois descrevem a sua evolução ao longo do tempo e de que forma alterou o comportamento da comunidade *online*.

A Web 1.0 é considerada a *web* de leitura, ou seja, estática, baseada em informar quem a utilizava e direcionada para os negócios, pelo que era bastante limitada na vertente da interação (Lee, 1998). No entanto, o propósito desta versão foi a criação de um espaço de informação comum às massas, de fácil acesso, onde a partilha de informação era o objetivo. Apesar de estática e unidirecional, esta cumpriu as metas desejadas, servindo de rampa de lançamento para empresas. Na sua génese, destacam-se protocolos base criados para assegurar a comunicação tais como o HTTP (HyperText Transfer Protocol) e FTP (File Transfer Protocol). Dadas as suas limitações e simplicidade, criar conteúdo para a *web* nesta fase apenas implicava conhecimentos de HTML e edição de imagem.

O conceito de Web 2.0 surge numa sessão de brainstorming em 2004 entre Dale Dougherty (O'Reilly Media) e Craig Cline (MediaLive International). Esta nasce como uma resposta às exigências dos utilizadores de estarem mais envolvidos na informação que lhes está disponível. Também designada de *web* social, acima de tudo acrescentou a vertente bidirecional transformando-se então numa *web* de leitura e escrita. O objetivo passou por participar na mesma, tendo o utilizador o papel principal. Assim, evoluiu para uma plataforma de comunicação e partilha (através de blogs, YouTube, Facebook), unindo a comunidade através

das diversas redes disponíveis. Para tal contribuiu a evolução das tecnologias que permitiram a transmissão de dados a velocidades que tornaram possível o carregamento de diferentes média e o acesso à rede móvel que tornou a presença *online* uma constante. Como consequência os serviços publicitados e vendidos através da *web* transitaram para serviços baseados na *web*. Todos estes fatores contribuíram para uma mudança radical do comportamento humano, que se tornou dependente da ligação à rede.

O passo seguinte na evolução da *web*, para Berners-Lee, passou por tornar o conteúdo legível não só para os utilizadores, mas também para os computadores. Tal foi possível através do conceito de serviços *web* que tomaram a forma de API (Application Programming Interface). Uma API caracteriza-se por ser uma interface para um componente de *software* que pode ser acedido à distância através de redes de comunicações usando tecnologias baseadas em *standards* predefinidos (3Scale, 2011). Estas tornaram-se comuns nesta fase da *web*, permitindo a programas comunicar com outros em busca de informação, relacionando-a.

Tal permitiu contextualizar a informação de forma a apresentá-la num formato que melhor se adequa a cada utilizador. Na prática cada utilizador obteve uma versão da *web* personalizada à sua medida, baseada nos seus gostos, hábitos e preferências. Pela capacidade de interpretar o conteúdo e melhor servir as exigências de cada um, esta versão foi designada de Web Semântica. A *web* tornou-se pessoal, atingiu-se um nível de interatividade entre o utilizador e a mesma onde uma parte depende da outra para concluir as tarefas desejadas.

Desde o início da *web* que as necessidades dos utilizadores estão em constante mudança. Desde a presença *online* à vontade de contribuir no conteúdo gerado *online*, até à exigência de uma experiência personalizada, a comunidade cada vez exige mais. Ao longo do tempo novos princípios surgem que moldam os objetivos da versão seguinte, a Web 4.0. Atualmente não há uma definição exata do que será, no entanto há guias que indicam que será desenvolvida com base nos princípios das versões anteriores e terá como foco a inteligência artificial. Esta irá oferecer maior autonomia e oportunidades aos utilizadores, para que

consigam cumprir os seus objetivos através de melhores formas de expressar as suas necessidades (Benhaddi, 2017).

Na génese da *web* foi criada uma linguagem de marcação especial com o objetivo de unificar a forma de criar páginas e, também, para que os *browsers* conseguissem interpretar. Assim surgiu o HTML (Hypertext Markup Language), que é atualmente a linguagem de marcação dominante nas páginas *web*. Através de comandos (HTML *tags*) que permitem a inserção de tabelas, *links*, listas, parágrafos, entre outros, tornava-se possível personalizar a estrutura do conteúdo.

Com a evolução natural da *web* explorou-se cada vez mais as suas potencialidades, que revelaram limitações do HTML na personalização da estrutura e conteúdo. A criação de páginas com um *layout* mais complexo (Ex: revista *online*) era impraticável, pelo que foram desenvolvidos *standards* para tornar possível uma personalização mais detalhada. Esta necessidade levou à criação das CSS (Cascading Style Sheets), uma linguagem de estilos que tem por objetivo descrever como deverão ser processados e apresentados os elementos inseridos numa página *web* (MDN DOCS, 2017).

O HTML é uma linguagem de marcação poderosa, flexível e universal, no entanto carece de ferramentas de programação, pelo que limita a interação do utilizador com a *web*. No entanto, este providencia a opção de adicionar *scripts*, dando assim maior poder ao desenvolvedor no seu processo de construção.

Na vertente cliente, a linguagem que dominou a programação de páginas *web* foi o JS (JavaScript). Este é usado para validar formulários, criação de efeitos visuais, bem como o desenvolvimento de páginas *web* com interfaces dinâmicas. O JS afirmou-se por ser uma linguagem conduzida por eventos, pelo que o código pode ser executado antes ou depois de um evento específico, e também por possibilitar à página *web* ser alterada em tempo real sem ter que enviar ou receber dados do lado do servidor (Akritidis, Katsaros, & Bozanis, 2011).

2.2 Web Components

Atualmente os *standards* do desenvolvimento *web* exigem, entre outros aspetos, que uma aplicação seja capaz de correr da mesma forma em diferentes *browsers*, que se adapte a diferentes tamanhos de ecrã, esteja preparada para telemóveis, relógios e até televisões (Savage, 2015). No entanto as tecnologias de desenvolvimento predefinidas (HTML, CSS, JS) não estão à altura do desafio, pelo que a comunidade tem vindo ao longo do tempo a desenvolver soluções que facilitam este processo. O resultado é a criação de um ecossistema em constante expansão, com diversas *frameworks* de desenvolvimento a serem disponibilizadas. Estas têm como objetivo a construção de aplicações com rigor profissional, com um vasto leque de funcionalidades, ao mesmo tempo que facilitam o processo para o programador.

A diversidade de *frameworks* disponível dá à comunidade a oportunidade de escolher a opção que mais se adequa ao projeto em questão. Desde as mais famosas e completas como a Angular da Google ou React do Facebook, às mais recentes com suporte menor, mas com alta potencialidade, a Riot. No entanto, esta enorme diversidade traz consigo obstáculos à produtividade do programador num contexto *cross framework*.

Devido à natureza das tecnologias *web* em que estas *frameworks* se apoiam (HTML, CSS, JS), cada uma tem o seu ambiente de desenvolvimento proprietário, com um conjunto de ferramentas disponíveis. Contudo esta estratégia leva a um bloqueio no que toca à interoperabilidade num processo de desenvolvimento *cross framework*, pelo que a escolha de uma *framework* é a primeira grande decisão tecnológica quando se inicia um projeto. Isto deve-se ao facto de os custos temporais de substituição de uma *framework* serem elevados pois iria começar-se de raiz.

Outro fator relevante é a elevada volatilidade do ecossistema, que torna a escolha de uma *framework* um investimento de alto risco, pois atualmente o ritmo a que as tecnologias ficam desatualizadas é alarmante. É possível observar a flutuação de popularidade de quatro das mais famosas *frameworks* recorrendo à

ferramenta de análise de tendências do *website* Stack Overflow, nos últimos anos (Figura 1). Esta é uma questão de maior importância nas empresas onde decisões deste caráter implicam investimentos de maior duração, pois os custos associados não permitem alterações frequentes de tecnologias.

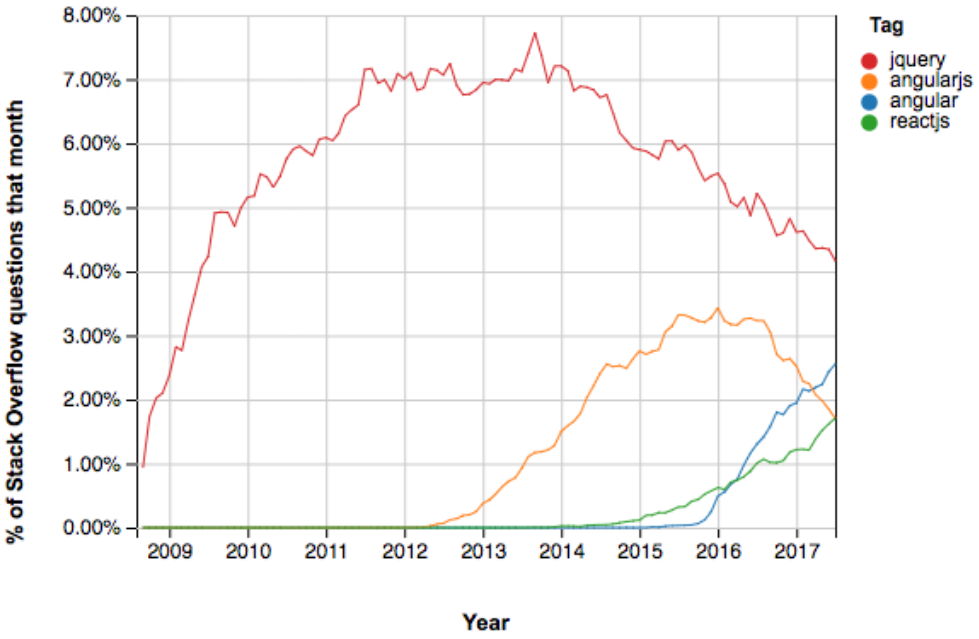


Figura 1 - Análise de tendências de *frameworks*. Fonte: Ian Allen, (2018).

Por fim, o último obstáculo trata-se da escassez de interoperabilidade, neste caso, a incompatibilidade entre *frameworks*. Como referido acima, cada *framework* requer exclusividade por parte do programador, ou seja, o processo de desenvolvimento é único. Isso também se verifica noutros aspetos importantes como a reutilização de elementos visuais.

No atual contexto de desenvolvimento *web* espera-se ser possível criar um componente versátil, que seja capaz de ser reutilizado e manipulado conforme as necessidades do projeto. Tal é possível quando se trata apenas de uma *framework* pois esta terá uma arquitetura preparada para reciclar os seus componentes, mas não quando se trata de um processo *cross framework*. Um componente criado em Angular não terá qualquer funcionalidade em React. Portanto, além da exclusividade requerida por estas, criar componentes reutilizáveis universalmente através destas tecnologias é praticamente impossível

(Savage, 2015). No entanto em paralelo a este ecossistema, o W3C, uma comunidade internacional responsável por desenvolver *standards* que visam garantir o crescimento da *web* a longo prazo, tem desde 2012 publicado normas que introduzem os Web Components.

Web Components são então um conjunto de tecnologias que permitem a criação de componentes reutilizáveis, encapsulados do restante código onde estão inseridos (Martínez-Ortiz, Lizcano, Ortega, Ruiz, & López, 2016). Seguem o mesmo conceito modular dos componentes proprietários das *frameworks* como Angular e React, mas ao contrário destes, são capazes de ser integrados nas diversas *frameworks* de desenvolvimento devido à sua interoperabilidade.

As vantagens relativas a estes componentes são: interoperabilidade, fácil manutenção, encapsulamento e fiabilidade (INKONIQ, 2017), pelo que serão abordadas de seguida.

Interoperabilidade

O maior *selling point*[2] desta tecnologia é o facto de ser interoperável entre *frameworks*, podendo assim ser utilizada em diversos contextos.

Num contexto empresarial onde várias equipas trabalham em diferentes projetos recorrendo a diversas tecnologias em simultâneo, é da maior importância criar componentes que sejam reutilizáveis, minimizando custos através da redução do tempo de desenvolvimento.

Fácil manutenção

O objetivo destes componentes passa por serem simples, focados apenas numa tarefa e otimizados para a mesma. Além disso, os Web Components são

[2] *Selling point* - qualidade ou característica desejável que algo tem que torna provável que as pessoas queiram comprá-lo.

baseados em tecnologias puras da *web* (HTML, CSS e JS). Agregando os dois pontos obtêm-se componentes com pouca necessidade de manutenção.

Encapsulamento

Uma das tecnologias associadas aos Web Components é o Shadow DOM. Esta tecnologia torna possível o encapsulamento do componente quando integrado noutro código. Desta forma é possível manter a estrutura do componente, bem como os estilos e todo o comportamento associado, evitando conflitos.

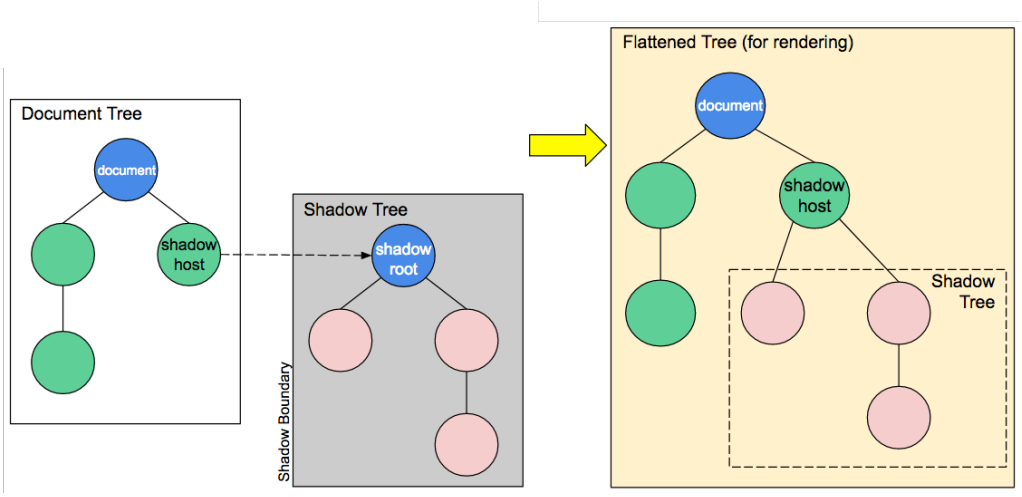


Figura 2 - Shadow DOM. Fonte: Swisher & Mills, (2018).

Fiabilidade

Sendo apenas necessário criar os componentes uma vez, estes serão otimizados e responderão às necessidades do utilizador. Reutilizando os mesmos componentes previamente testados em diferentes ambientes, o nível de confiança nos mesmos é elevado, ao contrário de um método de desenvolvimento que implique a sua constante criação.

Devido aos *standards* dos Web Components terem sido introduzidos em 2012, o suporte nativo por parte dos *browsers* está ainda em desenvolvimento. Entretanto é possível obter todas as funcionalidades dos Web Components

através de *polyfills*. *Polyfills* são excertos de código que providenciam a tecnologia necessária para simular funcionalidades nativas do *browser*, mas que não estão implementadas, solucionando assim qualquer falta de suporte nativo (webcomponents.org, n.d.). A Figura 3 representa o estado atual de suporte nos principais *browsers*, onde é possível verificar uma adesão positiva a esta tecnologia.

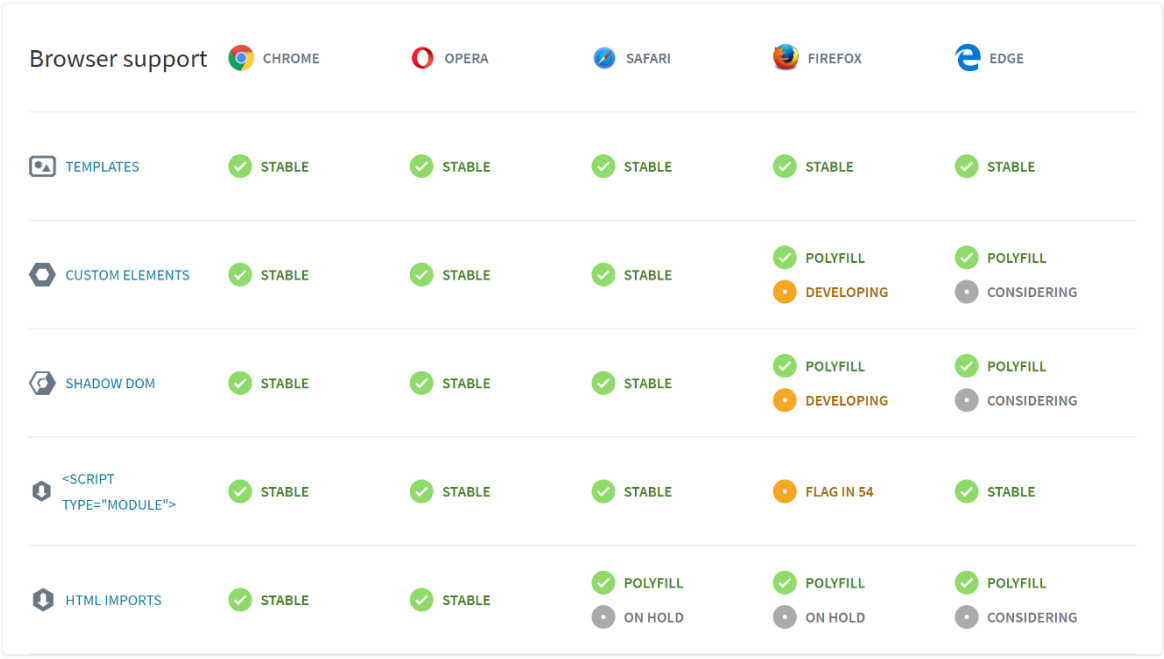


Figura 3 - Suporte nativo dos Web Components. Fonte: “webcomponents.org,” (2018).

Bibliotecas

Os Web Components requerem bastante código padrão (*boilerplate*), necessário para a estruturação dos mesmos. Além disso, a criação de raiz de um Web Component pode tornar-se desafiante pois deve respeitar normas que o desenvolvedor não familiarizado com a tecnologia terá dificuldade em seguir.

Para que a transição seja simples e rápida, a comunidade criou diversas bibliotecas que representam alternativas ao desenvolvimento predefinido, com o objetivo de melhorar a experiência de desenvolvimento com Web Components. Estas encarregam-se de todo o processo padrão, libertando o utilizador para as tarefas de implementação de funcionalidades. Como referido anteriormente quanto às *frameworks*, cada biblioteca aborda de forma diferente a sua solução, cabendo ao utilizador decidir qual se adequa melhor ao trabalho em questão.

Polymer

O Polymer é uma criação do Google, lançado em 2014. Este divide-se em duas vertentes, uma biblioteca para criação de Web Components e uma framework direcionada para a criação de aplicações *web* baseada em Web Components. Este encontra-se na versão 3.0, lançada em Maio de 2018.

A versão atual é largamente apoiada na versão 2.0, sendo quase 100% compatível com a anterior. Posto isto, o Polymer destaca-se por suportar novos *standards* implementados nos *browsers* mais recentes, como Custom Elements v1, Shadow DOM v1 e ligação de dados bidirecional. Além disso, conta com melhorias na interoperabilidade, na depuração de dados entre componentes e na standardização do código devido à adoção da versão ES6 do JavaScript. Além dos pilares da versão 2.0 em que se assenta, a versão 3.0 traz novidades que visam atrair desenvolvedores para a comunidade dos Web Components diminuindo a quantidade de aprendizagem de novos conceitos necessários para implementar os mesmos. Seguindo esta linha de pensamento, a versão 3.0 adapta-se à comunidade, contando com o suporte a TypeScript, migração para o

repositório *online* NPM (Node Package Manager) e alteração na forma de importar os componentes, de importação por HTML para JS (Justin Fagnani, 2017).

Se o facto de pertencer ao Google confere ao Polymer um constante desenvolvimento e suporte, é importante referir que as atualizações trazem problemas de continuidade de componentes mais antigos. No entanto a equipa envolvida priorizou o trabalho em volta de soluções que têm como objetivo a compatibilidade entre versões. Assim, na conferência Google I/O 2018, em conjunto com a apresentação da versão 3.0 também apresentaram uma ferramenta de linha de comandos, Polymer Modulizer, que tem a capacidade de atualizar os componentes desenvolvidos na versão anterior e assim atualiza-los de acordo com os padrões atuais de forma automática e sem custos acrescidos para o utilizador.

Stencil

Apresentado em agosto de 2017 na conferência Polymer da Google, o Stencil aposta numa abordagem focada na criação de componentes personalizados. Por ter esta abordagem, Stencil não é uma *framework* mas sim um compilador de Web Components. Assim, o objetivo passa por oferecer um ambiente de desenvolvimento rápido, com ferramentas como a renderização assíncrona, DOM virtual, JSX (JavaScript XML), *lazy-loading* e suporte a TypeScript, incluídos num pacote de apenas 6KB (“Stencil,” n.d.).

Por ter sido lançado recentemente, este compilador está numa fase de desenvolvimento inicial, pelo que é frequente o aparecimento de bugs. No entanto, devido ao facto de pertencer à Ionic Framework, uma empresa americana fundada em 2012 que fornece soluções de desenvolvimento mobile multiplataforma, o Stencil tem garantias de suporte e desenvolvimento futuro.

Skate

O Skate é uma biblioteca caracterizada por ser ultraleve, contando apenas com 4KB. Quanto à presença *online* e comunidade, esta é atualmente a segunda melhor posicionada, apenas superada pelo Polymer. O foco do Skate passa por ser uma fina camada sobre as especificações de Web Components focada num sistema de renderização otimizado.

As funcionalidades destacadas são a inclusão de *mixins* e *callbacks* personalizados de ciclo de vida dos componentes (Shugart, n.d.).

X-Tag

Esta alternativa é suportada pela Microsoft, foca-se em ser simples e leve, conseguindo na versão mais básica ocupar 5KB. No entanto nesta versão, esta apenas oferece a API Custom Elements do pacote de *standards* dos Web Components, como o Stencil.

A abordagem do X-Tag passa por simplificar o desenvolvimento de Web Components, o que o torna uma boa opção para a iniciantes, mas não acrescenta funcionalidades ao desenvolvimento nativo (Pankaj Parashar, 2015).

Slim

Esta biblioteca destaca-se por ser leve com apenas 5KB e conter ferramentas que a tornam uma alternativa relevante, como *data binding*. Este é um conceito que passa por conectar informação de um elemento a uma propriedade ou atributo de um outro elemento no mesmo DOM (Document Object Model), o que significa que é possível alterar o comportamento da interface dinamicamente (“Data binding - Polymer Project,” n.d.).

Além disso, é compatível com a versão ES6 do JavaScript, não necessita de compilação e baseia-se completamente em API’s nativas dos *browsers* (“Slim.js,” n.d.). No entanto, por não ter suporte de uma grande empresa como as alternativas anteriores, o investimento nesta biblioteca pode representar um risco futuro.

Tabela 1 - Comparação de bibliotecas de Web Components.

	Polymer	Stencil	Skate	X-Tag	Slim
Footprint	~10KB - ~36KB	~6KB	~4KB	~5KB - ~20KB	~5KB
Criador/Patrocinador	Google	Ionic Framework	Trey Shugart (Atlassian)	Microsoft	Avichay Eval
Lançamento	2014	2017	2014	2013	2016
Repositório	https://github.com/Polymer/polymer	https://github.com/ionic-team/stencil	https://github.com/skatejs/skatejs	https://github.com/x-tag/core	https://github.com/eavichay/slim.js/
Versão	v2.6.0	v0.6.8	v4.6.7	v1.5.11	v3.2.3
Licença/Preço	Tipo BSD	MIT	MIT	MIT	MIT
Documentação	Melhor Extensa e com exemplos	Acima da média Extensa e com exemplos	Normal Confusa	Normal Explícita	Normal
Roadmap	Detalhado no github	Não especificado	Não especificado	Não especificado	Não especificado
Github	5902 commits 48 branches 134 releases 131 contributors 19,134 Stars 1,911 Forks Issues 250 Open/3,415 Closed	1793 commits 9 branches 118 releases 30 contributors 2,248 Stars 98 Forks Issues 52 Open/421 Closed	4,125 commits 12 branches 172 releases 53 contributors 2,482 Stars 124 Forks Issues 46 Open/645 Closed	403 commits 16 branches 99 releases 15 contributors 1,168 Stars 150 Forks Issues 21 Open/129 Closed	431 commits 2 branches 105 releases 10 contributors 415 Stars 31 Forks Issues 1 Open/13 Closed
Stack Overflow (perguntas)	7,566	17	1	28	0
NPM (Downloads em janeiro)	21,416	16,769	44,314	1,417	775
Funcionalidades	ES6 Ligação de dados bidirecional Suporte a TypeScript na versão 3.0	ES6 TypeScript JSX Virtual DOM *Rendering* assíncrono *Lazy-loading* automático	ES6 Inclui *Mixins* e *Lifecycle callbacks* personalizados	ES5 2 versões: com ou sem *polyfills* *Polyfills* partilhados com o Polymer Apenas Custom Elements	ES6 Sem *transpiling*/*compiling* Único ficheiro Baseado em APIs nativas Web Components

	Polymer	Stencil	Skate	X-Tag	Slim
Chrome	Sim	Sim	Sim	Sim	Sim
Firefox	Sim	Sim	Sim	Sim	Sim
Opera	Sim	Sim	Sim	11+	Sim
Safari	Sim	Sim	Sim	Sim	Sim
Edge	Sim	Sim	Sim	Sim	Sim
Internet Explorer (IE)	IE11	IE11	IE11	IE9+	Sem informação
Android	Sim	Sem informação	Sem informação	4+	Sem informação
iOS	9+	Sem informação	Sem informação	5+	Sem informação
Pontos Fortes	Pacote completo Web Components Atualizações Comunidade Ligação de dados bidirecional *Plugins* disponíveis para VS Code	Rico em funcionalidades Uso eficiente dos *polyfills* Rápido crescimento *Plugins* disponíveis para VS Code	Importação por JS Funcionalidades Adoção por parte da comunidade Suportado pela Atlassian	Simples/Leve Boa primeira abordagem a Web Components	Simples/Leve Único ficheiro
Pontos Fracos	Maior *footprint* Mudança de *workflow* radical entre versões	Muito recente Futuro imprevisível Virtual DOM em vez do nativo Shadow DOM	*Polyfills* não especificados *Renderers* personalizados	Não adiciona funcionalidades como outras bibliotecas Não tem updates desde Junho 2016 Comunidade praticamente inexistente	Comunidade praticamente inexistente Não apoiado por uma empresa

2.3 *Cross frameworking*

A quantidade de *frameworks* de desenvolvimento *web* é vasta e todos os anos surgem novas que prometem ser melhores que as anteriores. Dados do maior *website* de perguntas e respostas de programação, o Stack Overflow, indicam que estas *frameworks* têm um ciclo de vida de cerca de 2 anos, depois dos quais iniciam uma lenta fase de declínio até deixarem de ser relevantes (Ian Allen, 2018). Um exemplo deste caso são o Knockout, Ember e Backbone, que tiveram uma rápida ascensão em 2011, e desde 2013 estão em queda (Figura 4).

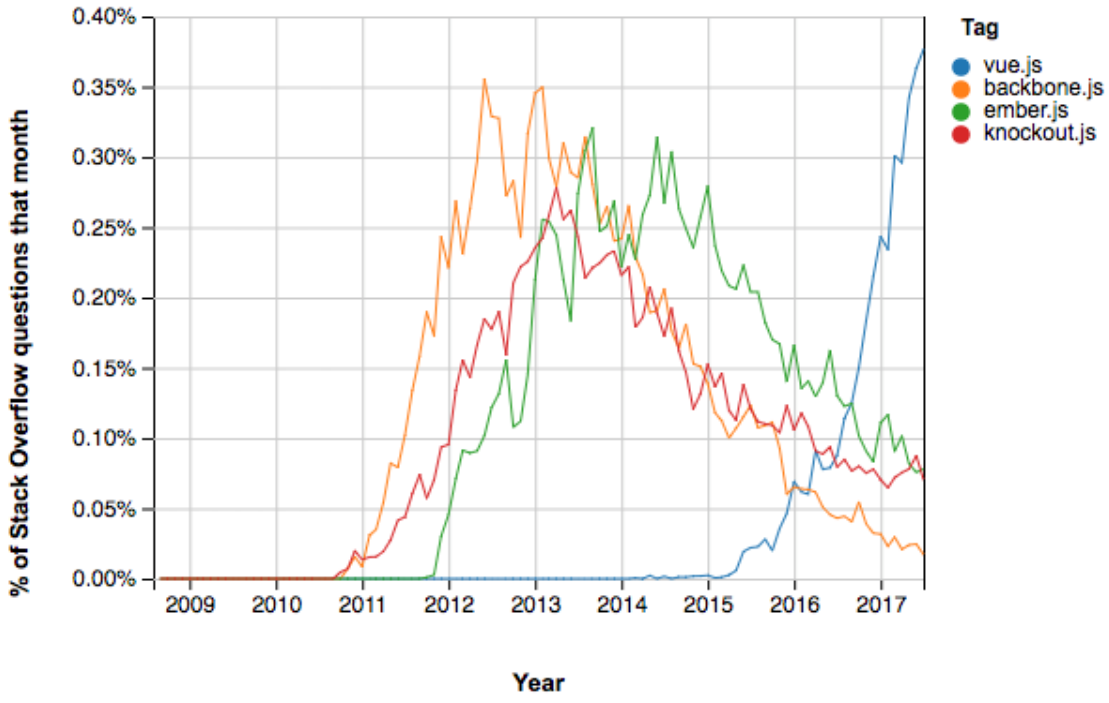


Figura 4 - Ciclo de vida de frameworks. Fonte: Ian Allen, (2018).

No entanto, num meio empresarial, a constante substituição de *frameworks* não é de todo viável uma vez que os ciclos de vida dos produtos são normalmente longos. O *overhead* de mudança tecnológica num produto é sempre grande e não é estranho encontrar produtos neste meio ainda com tecnologias ultrapassadas face às mais recentes tendências.

Neste contexto empresarial é comum existirem várias equipas a utilizar diferentes tecnologias em projetos paralelos, mas que, no entanto, devem apresentar ao utilizador final uma experiência de utilização semelhante. Para tal cada componente visual é criado mais que uma vez para servir cada tecnologia onde está inserido. Deste modo o custo de manutenção dos componentes torna-se elevado e aumenta consoante os universos tecnológicos que a empresa enquanto equipa de usabilidade transversal serve.

O cenário ideal seria a criação de componentes numa única tecnologia e que cada uma das *frameworks* os pudesse usar, independentemente dos processos internos impostos por cada tecnologia em particular quer para manutenção, quer para integração dos produtos. Os Web Components representam uma solução na medida em que os componentes visuais criados através destes são compatíveis com as *frameworks* de desenvolvimento como Angular ou React, e têm cada vez maior suporte por parte dos *browsers*.

2.4 Altice Labs

Desde 2 de junho de 2015, a PT Portugal é uma subsidiária integral da Altice Group, uma multinacional líder no fornecimento de serviços de telecomunicações com presença em França, Israel, Bélgica e Luxemburgo, Portugal, Antilhas Francesas/Área do Oceano Índico e República Dominicana e Suíça. A Altice Labs oferece aos seus clientes e parceiros os produtos, tecnologia e "*state of the art innovation*". Todos os recursos humanos do grupo Altice são responsáveis por pensar, criar e desenvolver novos produtos e serviços da empresa.

Assim, a Altice Labs, empresa do grupo Portugal Telecom, recentemente adquirido pelo grupo Altice e conhecida como o centro de inovação tecnológica sediado em Aveiro, é uma empresa de telecomunicações com mais de 65 anos. Esta tem como valores a inovação tecnológica e criatividade, com o objetivo de oferecer soluções avançadas e diferenciadoras no setor onde se insere. Assim como as telecomunicações, esta evoluiu e está presente nos vários ramos do setor, sendo líder de inovação, o que lhe permite satisfazer as crescentes necessidades do mercado. A infraestrutura de Aveiro conta com três edifícios, por onde se distribuem os recursos humanos que trabalham em áreas tão diversas quanto *Internet of Things*, desenvolvimento de rede 5G ou até projetos de realidade virtual. Para além disso, esta beneficia da estreita colaboração com a Universidade de Aveiro (Rocha, 2016).

A Altice Labs conta com cinco grandes áreas de negócio, sendo elas: conectividade, sistemas de suporte às operações, controlo de rede e plataformas de serviços, aplicações e serviços profissionais ("Altice Labs Whitepaper," 2014). O estudo destas áreas de investigação e desenvolvimento permite transformar o conhecimento em inovação tecnológica para criar diferenciação e valor no mercado. Portanto, a Altice Labs tem como objetivo fornecer aos clientes soluções de qualidade, *on time* e *on budget*, promovendo uma cultura baseada na qualidade e na melhoria contínua dos seus processos e produtos, numa busca permanente por aumentos de eficiência, eficácia e produtividade, suportada em parcerias estratégicas com líderes de mercado que complementam e garantem a

qualidade da oferta, tendo em vista a compreensão, satisfação e superação das necessidades e expectativas dos clientes e procurando constantemente o reconhecimento externo através de certificações organizacionais reconhecidas e prestigiantes (Altice, 2016).

Integrada no departamento DIT - Digital, Internet e Televisão, a equipa de Usabilidade e User Experience é uma equipa transversal, presente em todos os projetos desenvolvidos. Esta foca-se em garantir a satisfação dos clientes através de uma experiência de utilização orientada para a eficiência e facilidade de uso.

O Programa Genius Inova-Ria preconiza Bolsas de Investigação Científica regulamentadas pela Fundação para a Ciência e Tecnologia (FCT) atribuídas para atividades de investigação, desenvolvimento tecnológico ou formação conexa com essas áreas. Aplica-se nos termos da Lei nº 40/2004, de 18 de agosto e Decreto-Lei nº 202/2012 de 27 de agosto, alterada e republicada pelo Decreto-Lei nº 202/2012, de 27 de agosto, e alterada pela Lei nº 12/2013, de 29 de janeiro e pelo Decreto-Lei nº 89/2013, de 9 de julho.

Surge assim, neste âmbito, a temática de Web Components como uma solução de compatibilidade na implementação de componentes *cross framework*, como um projeto de inovação.

3. Metodologia

O projeto em questão centra-se numa solução a ser implementada no processo de desenvolvimento da Altice Labs, com vista à resolução de problemas de interoperabilidade dos componentes *web* utilizados.

A metodologia aplicada é mista, nomeadamente *Design Based Research*, pois baseia-se em resolver problemas do mundo real através de soluções inovadoras. Por este motivo esta metodologia é cada vez mais comum na atualidade tecnológica (De Villiers & Harpur, 2013). Designadamente ao nível do projeto, a sua abordagem específica é a *User Centered Design*, em linha com *Design Based Research*.

Existem quatro tipos de *outputs* relativos às soluções, Idealização, Modelos, Métodos e Instanciação. A Idealização é a formação de conceitos e vocabulário básico em torno de um domínio. Modelos são o resultado da combinação de idealizações com relações entre elas. Estes têm também representações de tarefas, situações ou artefactos. Métodos são formas de realizar tarefas orientadas ao objetivo, por exemplo um algoritmo. A Instanciação distingue-se por ser o único tipo que envolve a implementação do produto desenvolvido. Considerando que o projeto tem como objetivos o desenvolvimento, teste e avaliação de um protótipo em contexto real, este tipo de *output* é o que melhor se enquadra.

Assim, este é um processo iterativo no qual o ponto de entrada pode variar dependendo da fase em que o projeto se encontra. Dado que a Altice Labs tem identificado o problema a ser solucionado, assim como a solução desejada para o mesmo, a continuidade do projeto inicia-se no *design* e desenvolvimento do protótipo (Figura 5).

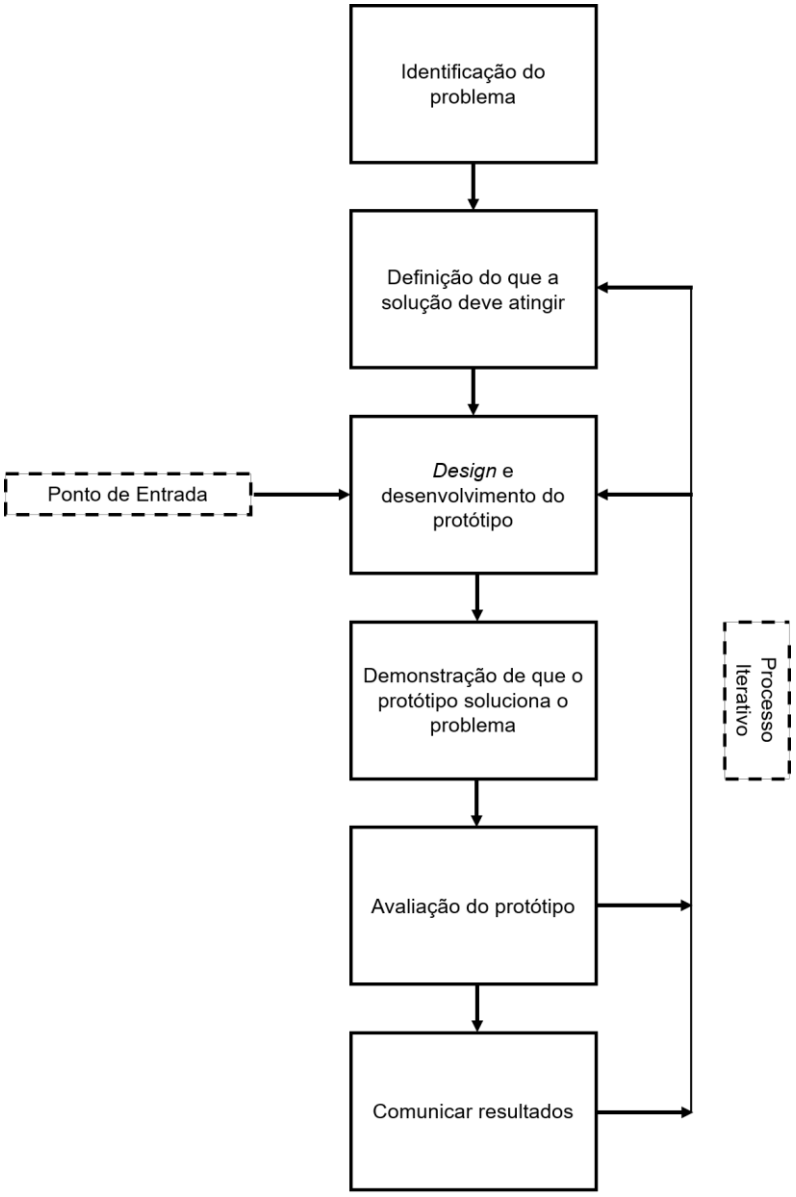


Figura 5 - Modelo genérico da metodologia D&R.

3.1 Participantes

Os participantes fazem parte de dois grupos distintos. Do primeiro grupo fazem parte todos os profissionais que integram as equipas da Altice Labs que terão o seu processo de desenvolvimento alterado pelo protótipo produzido. Neste incluem-se os desenvolvedores que recorrem a tecnologias *web* aplicadas em diversos contextos, como ferramentas *online* de suporte, desenvolvimento mobile e TV. No segundo grupo enquadram-se os utilizadores finais dos produtos desenvolvidos pela empresa. Esta diferenciação deve-se ao facto de a tecnologia potencialmente afetar os dois grupos de forma distinta dado o posicionamento dos grupos ser oposto. Enquanto que o primeiro grupo é diretamente afetado pois o processo de desenvolvimento se altera quanto à tecnologia utilizada, o segundo grupo pode ser afetado indiretamente quanto à experiência de utilização do produto.

A seleção destes será através de uma amostragem pelo método não probabilístico de conveniência, sendo que todos os indivíduos referentes ao projeto e que tenham disponibilidade para participar serão incluídos. Por ser uma amostra por conveniência não será possível generalizar os dados, no entanto será possível retirar informações relevantes sobre o projeto.

Tabela 2 - Participantes da recolha de dados.

Participantes	**Caracterização**
Grupo 1	Desenvolvedores Altice Labs
Grupo 2	Utilizadores finais dos produtos desenvolvidos

3.2 Instrumentos de recolha de dados

A recolha de dados centrar-se-á nas potencialidades de uma arquitetura de sistema baseada em Web Components e a aplicabilidade da mesma no processo de desenvolvimento da Altice Labs. Este processo terá duas fases diferenciadas que terão lugar antes e depois do desenvolvimento do protótipo.

Numa primeira fase o objetivo é a familiarização com as técnicas e cultura de cada *framework* usada na Altice Labs e compreender o *workflow* associado a cada *framework*. Para tal serão feitas entrevistas exploratórias complementadas por uma observação direta. Depois do desenvolvimento de um protótipo funcional, a segunda fase passa por testar o mesmo com vista a avaliar a sua robustez e aplicabilidade.

Será realizada uma sessão de recolha de dados com os participantes do primeiro grupo, composta por um *workshop* e um questionário. O *workshop* terá como objetivo a introdução da tecnologia aos participantes de forma dinâmica e interativa através de um primeiro contacto que será realizado com o protótipo desenvolvido. Esta técnica tem como objetivo a discussão de ideias entre os desenvolvedores, tirando partido da diversidade tecnológica existente. Desta forma será possível a formação de opiniões melhor formadas quanto à tecnologia.

Posteriormente o questionário terá como finalidades a recolha de dados analíticos do ponto de vista de cada participante, sobre a viabilidade e aplicabilidade dos Web Components no contexto de desenvolvimento atual das equipas.

Por forma a obter *feedback* dos participantes do segundo grupo, serão realizadas entrevistas semiestruturadas com o objetivo de recolher dados quanto à experiência de utilização dos produtos afetados pela tecnologia.

Tabela 3 - Planeamento da recolha de dados.

Fase	Participantes	Instrumento	Objetivo
Fase 1	Desenvolvedores Altice Labs	Entrevista semiestruturada	Familiarização com as técnicas e cultura de cada *framework*
		Observação direta	
Fase 2	Desenvolvedores Altice Labs	*Workshop*	Recolha de dados qualitativos sobre a opinião dos participantes
		Questionário	Recolha de dados analíticos sobre a viabilidade e aplicabilidade dos Web Components
	Utilizadores finais dos produtos desenvolvidos	Entrevista semiestruturada	Recolher dados quanto à experiência de utilização dos produtos

4. Processo de implementação do Web Component

Dado o grande leque de possibilidades de implementação desta tecnologia, foi necessária a realização de uma análise profunda sobre a mesma, com vista a expandir o conhecimento sobre as diferentes abordagens. Desta forma foi possível testar as que mostraram ser mais relevantes para o caso de estudo.

O estudo dos Web Components foi pertinente na medida em que se baseiam em quatro *standards* independentes, sendo que a utilização de todos não é obrigatória, além de que os mesmos continuam em desenvolvimento. Desta forma, o contexto de aplicação dos Web Components na Altice Labs, bem como as limitações apresentadas por estes, iriam determinar a sua implementação.

Pretendeu-se explorar as vantagens e desvantagens do uso de cada *standard* bem como as suas limitações, como o suporte por parte dos *browsers* e a conjugação dos mesmos. Assim, foi conduzida uma análise detalhada de cada *standard* a fim de recolher informação relevante para o desenvolvimento do protótipo.

Posto isto, e tendo em conta o âmbito do projeto e o propósito dos Web Components, desenvolveu-se um protótipo com o objetivo de testar a capacidade de esta tecnologia servir as necessidades da empresa. Visto que não há uma forma padronizada de desenvolver Web Components e dado o âmbito exploratório do projeto, o protótipo foi composto por três versões. Como os *standards* são independentes, é pertinente a experimentação de diferentes conjugações dos mesmos. Por outro lado, a falta de consolidação destes implica o teste de uma versão mais limitada mas estável. A primeira recorreu à tecnologia nativa sem aplicação do *standard* Shadow DOM, a segunda recorreu também à tecnologia nativa mas com aplicação do Shadow DOM. Por fim, com o objetivo de analisar o impacto de uma biblioteca, a terceira versão contou com a biblioteca Polymer e a implementação de Shadow DOM.

Tabela 4 - Resumo das versões do protótipo.

Standard	Versão 1	Versão 2	Versão 3
HTML Templates	✓	✓	✓
HTML Imports	×	×	✓
Custom Elements	✓	✓	✓
Shadow DOM	×	✓	✓

4.1 Requisitos técnicos e funcionais

Com vista a testar os Web Components, foi pensado o desenvolvimento de um protótipo que explorasse a tecnologia. Assim, propôs-se replicar um componente da FUXI que abordasse vários aspetos relevantes, sendo esse componente chamado Tabs, que pode ser acedido *online* através das instruções presentes no apêndice D (AP D - Instruções para aceder ao protótipo online). Tipicamente, um sistema de separadores é composto por uma área de navegação e por uma área de conteúdo. A finalidade deste é fornecer uma organização otimizada ao distribuir o conteúdo por diferentes janelas, sendo apenas uma visível de cada vez. O utilizador consegue assim obter uma navegação pela informação disponível melhorada e conta com o título de cada separador para se situar.

Este é composto por diversos elementos como um bloco agregador de conteúdo, separadores para transição entre painéis e botões de adição e navegação entre os separadores. Além dos elementos visuais, este componente tem integrado lógica que o torna dinâmico quanto ao conteúdo e comportamento, sendo capaz de apresentar informação contextualizada.

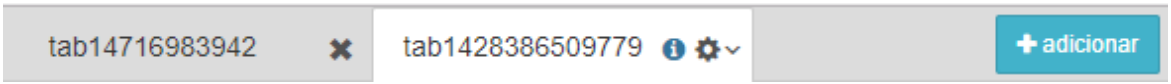


Figura 6 - Componente Tabs.

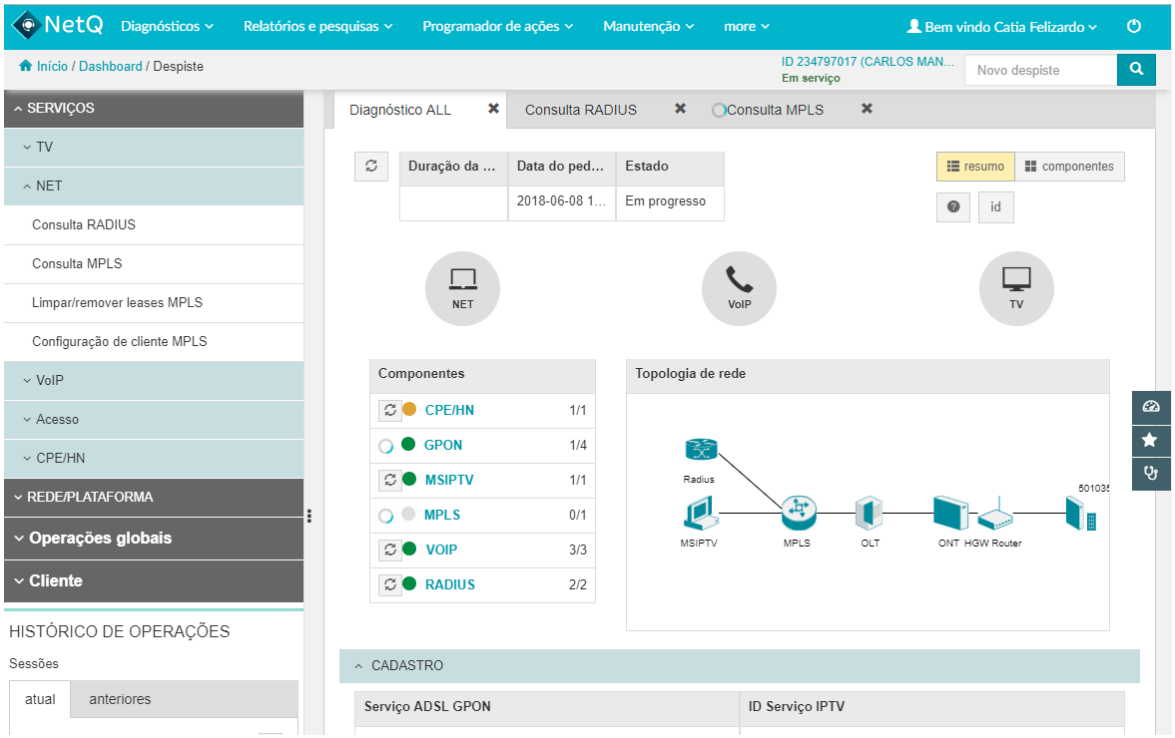
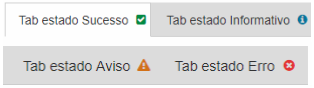
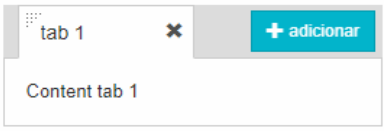
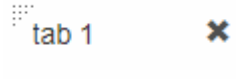
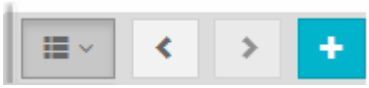
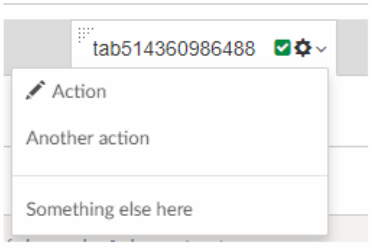
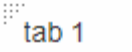


Figura 7 - Componente Tabs inserido no produto NetQ.

Requisitos Funcionais

O componente Tabs é caracterizado por ser bastante flexível quanto à sua aparência e comportamento devido à necessidade de diferentes produtos da Altice Labs implementarem este componente. Para que este consiga servir os requisitos dos produtos onde está inserido, este tem incorporado diversas funcionalidades como o reposicionamento dos separadores, navegação horizontal e a associação de um estado aos separadores (tabela 5).

Tabela 5 - Funcionalidades do componente Tabs original.

	Descrição	*Mockup*
Design		
1	Deve ser possível organizar conteúdo usando uma estrutura tabular.	
2	A estrutura tabular deve poder ser vertical.	
3	Deve ser possível encapsular um sistema de separadores dentro de outro.	
4	Os separadores devem permitir associação de um estado (sucesso, informação, aviso ou erro) através de iconografia e cor.	
5	Os separadores devem indicar através de um *loading* sempre que o seu conteúdo demorar a carregar.	
6	Deve ser possível adicionar separadores de forma programática.	
7	Deve ser possível remover separadores de forma programática ou através do botão disponível na interface	
8	Quando não existir espaço suficiente na interface para acomodar todas os separadores, estas devem empilhar para uma *dropdown* e permitir aí a sua seleção. Essa *dropdown* deve permitir pesquisa. Deve ser possível navegar horizontalmente por todas os separadores através de botões de navegação.	
9	Cada separador deve permitir ter um menu de operações.	
10	Deve ser possível reposicionar os separadores.	

11	Deve ser possível ordenar os separadores.	
Métodos		
12	Deve ser possível ativar programaticamente um separador.	
Eventos		
13	Quando for mostrada um novo separador, devem ser disparados os seguintes eventos pela ordem apresentada: • hide.tab (no separador atualmente ativa) • show.tab (no separador que vai ser apresentada) • hidden.tab (no separador que estava ativa) • shown.tab (no separador que passou a ativa e acabou de ser apresentada) Se não existir um separador previamente ativa, os eventos **hide.tab** e **hidden.tab** não devem ser disparados.	

Posto isto, foi desenvolvida uma lista de requisitos técnicos e funcionais antes do desenvolvimento do protótipo com vista a expor os pontos pertinentes a abordar, para assim se obter conclusões do protótipo que conseguissem responder às necessidades da empresa, como a viabilidade dos Web Components e os casos de estudo onde melhor se aplicam.

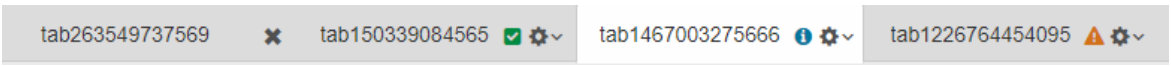


Figura 8 - Separadores do componente Tabs.

Dada a complexidade do componente, filtraram-se as funcionalidades para que o foco estivesse na exploração de potencialidades e limitações da tecnologia em vez de criar uma réplica exata do componente original. Listou-se um requisito funcional implicando que o Web Component contenha funcionalidades essenciais do componente original, ou seja, a adição, seleção e eliminação de separadores.

Além disso também se considerou relevante a inserção de botões de ação nos separadores para a realização de operações adicionais (Figura 10) e a implementação de aleatoriedade dos atributos dos separadores, para mimetizar o comportamento do componente em produção.

Figura 9 - Botão de adição de separadores.

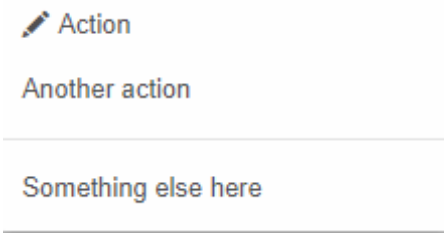


Figura 10 - Ações adicionais dos separadores.

Requisitos Técnicos

O primeiro requisito técnico passou por desenvolver o Web Component sem qualquer dependência externa. Dado que o Web Component deve estar preparado para ser inserido em variados contextos, deve ser independente e abstrair-se de qualquer dependência para cumprir as funcionalidades exigidas, garantindo assim o bom funcionamento do mesmo.

O segundo requisito técnico explorou a compatibilidade do Web Component nos *browsers* aos quais a Altice Labs se compromete a dar suporte. O suporte aos mesmos é uma questão de elevada importância, visto não ser possível controlar a forma como o utilizador final acede aos produtos da empresa. Assim para ser viável, a tecnologia deve ser compatível no maior número de *browsers*. Os *browsers* para os quais o protótipo respondeu foram o Google Chrome 65, Safari 11.1, Firefox 59, Edge 16 e Internet Explorer 11.

A Altice Labs desenvolve vários produtos em paralelo, tendo equipas dedicadas no desenvolvimento recorrendo a diferentes tecnologias. O terceiro requisito técnico teve como objetivo testar a interoperabilidade do Web Component em ambientes de desenvolvimento díspares, garantindo assim que depois de desenvolvido, o componente fosse capaz de se inserir em qualquer produto que o requeresse.

O quarto requisito técnico abordou a customização do Web Component. Este deve estar preparado para ser personalizado graficamente, pois é uma necessidade da Altice Labs que um componente seja usado em diferentes produtos que contam com aspetos gráficos diversos. Assim é essencial que quando inserido num produto, o Web Component permita a alteração de propriedades para se adaptar ao mesmo.

O quinto requisito técnico teve por base um dos *standards* dos Web Components, o Shadow DOM. As circunstâncias de utilização do componente são incertas. Dada a diversidade de tecnologias a que a empresa recorre, este deve estar encapsulado, abstraindo-se assim de propriedades ou comportamentos declarados fora do mesmo, garantindo o bom funcionamento.

Por fim, foi adicionado um último requisito técnico com vista a garantir que o aspeto gráfico do Web Component seja idêntico ao componente original, assim impondo rigor no desenvolvimento do mesmo. Visto que este não terá todas as funcionalidades do original não será possível criar uma réplica visual, no entanto os elementos gráficos partilhados deverão ser iguais.

4.2 Análise à tecnologia Web Components

A tecnologia Web Components, apesar de ser pensada como um todo, foi desenhada para ser flexível. Tal é conseguida através da independência dos *standards* em que se baseia, que têm a capacidade de cumprir as suas finalidades sem intervenção dos outros *standards*. Assim possibilitam a conjugação dos mesmos da forma que for mais pertinente. Logo, a análise à tecnologia apenas faz sentido se se tratar os *standards* de forma independente.

4.2.1 HTML Imports

HTML Imports permite a importação e uso de ficheiros HTML dentro de outros ficheiros do mesmo tipo, possibilitando assim o desenvolvimento de Web Components com recurso a HTML.

Aquando da especificação dos Web Components, o HTML Imports pretendia padronizar a importação de todos os recursos.

Suporte

Dos quatro *standards* desenvolvidos este é o que tem menos adoção por parte dos *browsers*, pois é uma abordagem que não vai ao encontro das tendências de desenvolvimento *web* atuais, que atualmente se centram em remeter para o JS ações como a importação de módulos ou a declaração de HTML.

Em alternativa, é possível utilizar HTML Imports importando um *polyfill* que o torna compatível nos *browsers* modernos[3]. Devido a este obstáculo, este está a cair em desuso, optando assim os desenvolvedores por ciar Web Components em JS, desta forma alterando o método de importação, conseguindo assim contornar este constrangimento.

[3] *Browsers* modernos - *browsers* que se mantêm atualizados automaticamente, mantendo o utilizador seguro e com acesso a novas funcionalidades.

Vantagens

Este *standard* tem como vantagem principal o controlo de importações, conseguindo resolver a duplicação de recursos iguais mesmo que estes sejam importados diversas vezes ou de diferentes localizações. No entanto, atualmente, os *browsers* modernos conseguem também gerir as importações e otimizá-las de um modo semelhante e de forma automática.

Outra vantagem relevante é a legibilidade do código em ficheiros HTML. A estrutura de um Web Component desenvolvido com base em HTML torna a legibilidade do código melhorada. Ao contrário de quando o mesmo é desenvolvido com recurso a JS, a sintaxe HTML num ficheiro JS deve estar contida numa *string* que torna o realce das *tag*s dos elementos indistinguível, afetando a legibilidade.

Desvantagens

Como referido anteriormente, a desvantagem é baseada na fraca adesão quanto ao suporte por parte dos *browsers* (Figura 11), que torna o uso deste *standard* apenas possível com recurso a um *polyfill* menos no Chrome.

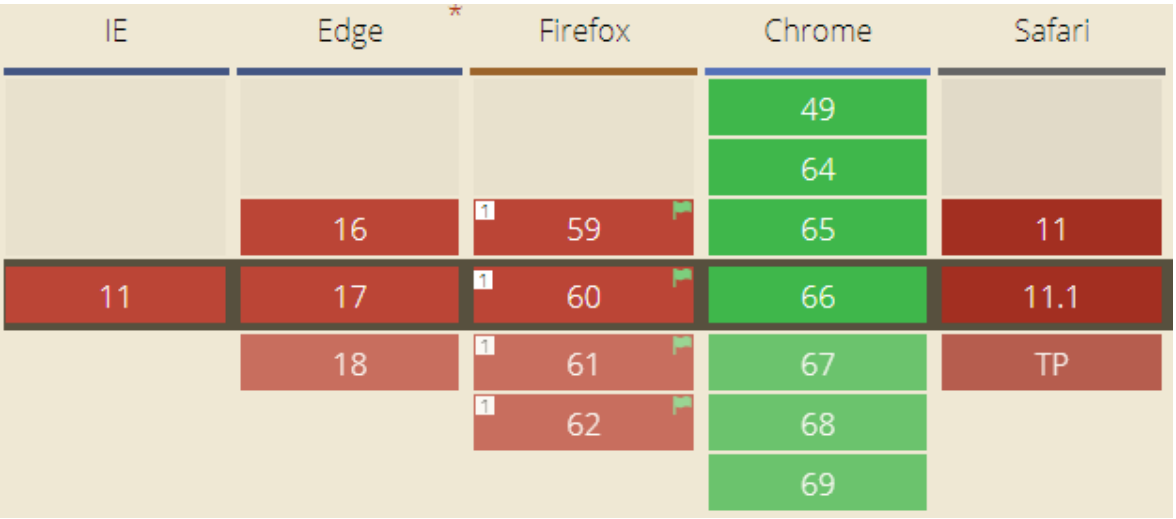


Figura 11 - Suporte dos *browsers* ao *standard* HTML Imports[4].

[4] https://caniuse.com/#search=html%20imports

4.2.2 HTML Templates

HTML Templates é um *standard* que permite a declaração de conteúdo que deve ser *renderizado* apenas quando for clonado ou instanciado via JS, ao invés de ser *renderizado* de forma "natural", quando é carregado.

Suporte

Devido à importância deste *standard*, no âmbito dos Web Components, o HTML Templates tem um elevado suporte por parte dos *browsers*. Este resultado deve-se também ao consenso e estabilidade do mesmo. Apesar disso, o Internet Explorer 11 destaca-se por ser o único *browser* moderno a não dar suporte a este *standard*[5]. De referir que o Internet Explorer 11 não suporta nenhum dos quatro pilares dos Web Components, limitando bastante o desenvolvimento de componentes baseados nos mesmos para este *browser*.

Vantagens

Este *standard* desempenha um papel importante pois permite a declaração de porções de código para futura utilização. Incentiva assim ao desenvolvimento de componentes com propósitos claros e simples, para que sejam passíveis de ser reutilizados. Desta forma vai ao encontro das novas práticas de modularização da *web*.

[5] Apesar de o Internet Explorer 11 não ser um *browser* moderno segundo os parâmetros anteriormente falados, enquadra-se nesta nomenclatura devido à elevada utilização em contexto corporativo, pelo que tem relevância a nível de suporte.

4.2.3 Custom Elements

Este é a base de um Web Component pois permite a definição do mesmo no DOM. Assim torna-se possível usar elementos criados de acordo com a necessidade atual e a consequente utilização no DOM como se tratasse de uma *tag* HTML nativa. Ao contrário dos *standards* referidos anteriormente, este é mandatório para a utilização de um Web Component.

Suporte

Atualmente é suportado pelo Chrome, Opera e Safari. Não é suportado pelo IE11, está sob consideração no Edge e em desenvolvimento no Firefox. No entanto existem *polyfills* que tornam este *standard* compatível em todos os *browsers* (Deveria & Schoors, 2018).

Vantagens

Este *standard* permite o uso de um componente em diversos contextos e tecnologias (Angular, React, etc) pois é interpretado como uma *tag* HTML. Assim, consegue-se atingir um elevado nível de interoperabilidade que favorece a produtividade e a otimização de recursos.

Desvantagens

Apesar de estarem em desenvolvimento desde 2013, estes *standards* ainda não alcançaram consenso por parte da comunidade, pelo que a adoção dos mesmos por parte dos mais importantes *browsers* ainda está em desenvolvimento.

4.2.4 Shadow DOM

O Shadow DOM é o último de quatro *standards* que fazem parte dos Web Components. Este *standard* permite o encapsulamento do componente, por forma a não verter CSS ou JS da página principal para dentro do componente e vice-versa. Assim é possível garantir que o comportamento do componente e a aparência visual do mesmo é igual em diferentes ambientes onde este for incorporado.

Suporte

Como no caso do Shadow DOM, atualmente este é suportado pelo Chrome, Opera e Safari. Não é suportado pelo Internet Explorer 11, está sob consideração no Edge e em desenvolvimento no Firefox (Deveria & Schoors, 2018).
Para corrigir esta limitação de suporte de *browsers*, existem *polyfills*/*shims* (fazem parte da suite de *polyfills* chamada "webcomponentsjs") que de alguma forma mimetizam o seu comportamento, nomeadamente o Shady DOM.

Shady DOM e Shady CSS

O Shady DOM encarrega-se de simular o encapsulamento do componente, tornando o código desenvolvido com Shadow DOM compatível com *browsers* que não o suportam. No entanto não passa de uma simulação, não ativando assim as funcionalidades nativas do Shadow DOM, apenas tornando o código compatível.

Por forma a obter um encapsulamento da estilização dos componentes é necessário usar uma biblioteca opcional, não sendo esta ativada automaticamente pela declaração da *suite* de *polyfills* na aplicação, chamada Shady CSS. Esta é composta por três vertentes, usadas para diferentes aspetos da estilização, que interpretam os vários componentes e, no caso particular do protótipo, os encapsula, prevenindo assim que os estilos aplicados ao componente vertam para os restantes elementos no contexto onde este se encontra.

Contudo, esta solução exige que sejam aplicadas alterações à estrutura do componente, nomeadamente na secção dos estilos aplicados ao mesmo, tendo

estes que ser envolvidos numa *tag* 'pai' chamada "custom-style" como ilustrado na Figura 12.

```
<custom-style>
    <style>
        html,
        body {
            margin: 0;
            font-family: "Arial", Helvetica, Helvetica Neue, sans-serif;
            font-size: 13px;
            line-height: 1.42857143;
            color: #333333;
            background-color: #ececec;
            transition: opacity ease-in 0.2s;
        }

        * {
            box-sizing: border-box;
        }
    </style>
</custom-style>
```

Figura 12 - Estilos contidos numa *tag* "custom-style".

Além disso, e com maior relevância para o processo de desenvolvimento, a mesma adaptação deve ser feita no documento base onde os componentes forem integrados (tipicamente index.html), para que os estilos definidos no mesmo não afetem o Web Component. Num ambiente de desenvolvimento como o da Altice Labs, onde a equipa que desenvolve os componentes não é a mesma que os integra nas diversas aplicações, esta exigência implica uma passagem de conhecimento extra entre equipas para que a integração de Web Components seja bem-sucedida.

Contudo o encapsulamento dos estilos continua a não estar assegurado, pois este *polyfill* não é capaz de encapsular estilos que não sejam declarados de forma *inline*. Ou seja, estilos importados de ficheiros de CSS externos ou criados dinamicamente por JS afetarão todos os componentes[6].

[6] https://github.com/webcomponents/shadycss

Vantagens

Este *standard* traz várias vantagens ao nível do desenvolvimento e produção de Web Components. A vantagem de maior relevância para o projeto em questão é a redução massiva do risco de colisão de estilos (CSS) entre o componente e o ambiente onde está inserido. Desta forma não é necessário recorrer a técnicas de nomenclatura elaboradas nem aplicar estilos com um rigor extra para não se afetar elementos externos, pois tal não é possível através deste *standard*. Além disso, tratando o componente de forma isolada, garante-se a integridade do mesmo.

Outra funcionalidade que eleva o valor deste *standard* é a possibilidade de o desenvolvedor controlar o que pode, ou não, ser estilizado de forma externa no componente. Este *standard* impõe que seja explicitado quais as propriedades do componente que devem ser passíveis de estilização externa, através de variáveis de CSS. Desta forma, num ambiente empresarial onde é da maior importância o controlo sobre a manipulação dos recursos, esta funcionalidade é valorizada. Além disso, numa empresa como a Altice Labs, manter o mesmo *look & feel* ao longo dos seus produtos é extremamente importante para que os utilizadores se sintam confortáveis e confiantes ao interagir com os mesmos, pelo que os componentes que os compõem devem garantir que a sua identidade é mantida.

Por fim, a organização da estrutura do componente é também melhorada pois o Shadow DOM incentiva à composição do mesmo em componentes mais simples e com manutenção menor. Esse incentivo provém de uma *tag* apenas usada neste *standard*, a “slot”. Esta *tag* permite hierarquizar componentes no interior de outros componentes de forma externa e organizada.

Desvantagens

O estado atual do Shadow DOM não está consolidado, pelo que o nível de suporte dos mais importantes *browsers* da atualidade é baixo. Alguns dos aspetos do Shadow DOM que estão por resolver são a falta de suporte a algumas @*rules* (ex: @*font-face*) e a falta de consenso quanto à forma de estilização de componentes, sendo que atualmente a única forma nativa é por CSS *variables*.

Além disso, os *polyfills/shims* disponíveis estão atualmente em desenvolvimento, sem suporte para funcionalidades chave como o encapsulamento de documentos de estilos inseridos externamente, ou estilos inseridos dinamicamente com recurso a JS, pelo que ainda não são uma alternativa viável.

4.3 Análise à biblioteca de desenvolvimento de Web Components

Como parte do protótipo produziu-se uma versão que recorreu a uma biblioteca de desenvolvimento de Web Components, a fim de aferir de que modo esta é capaz de elevar a experiência de desenvolvimento. Das bibliotecas referidas anteriormente, foi escolhida a Polymer.

À data de desenvolvimento do protótipo, a versão atual era a 2.6.0, pelo que foi essa a utilizada. Esta biblioteca é desenvolvida pelo Google e recentemente foi lançada a versão 3.0. Conta com uma comunidade vasta que a apoia de forma direta ou indiretamente. Além disso, pertencendo ao Google, dá garantias de suporte e evolução da mesma. A adoção do Polymer traz algumas vantagens no desenvolvimento de Web Components que devem ser consideradas. A documentação é extensa e organizada, pelo que facilita a introdução à biblioteca. Além disso, esta conta com a maior comunidade referente a bibliotecas de Web Components, presente no GitHub, StackOverflow e *blogs*, sendo uma mais-valia para o desenvolvimento da mesma e para o *debug* de Web Components no processo de desenvolvimento.

O Polymer disponibiliza uma CLI (*command-line interface*) que facilita o desenvolvimento *cross-browser* de Web Components, bem como a preparação dos mesmos para produção. Além disso este ativa por predefinição o Shadow DOM nos componentes, otimizando assim o desenvolvimento de componentes.
Por fim, destaca-se a maior facilidade em contornar obstáculos referentes ao suporte dos *browsers* em relação ao desenvolvimento nativo. Como todas as bibliotecas, o uso da mesma implica um custo em Kilobytes associado ao resultado final do produto, sendo esse custo cerca de 30 a 40KB com o HTML,

CSS e JS comprimidos. Além deste, existe um custo subjetivo de aprendizagem da biblioteca, que neste caso não se revelou muito significativo dadas as similaridades com o desenvolvimento nativo. No entanto o histórico da biblioteca mostra que houve mudanças relevantes no processo de desenvolvimento da versão 1.0 para a 2.0, e o mesmo aconteceu da versão 2.0 para a 3.0.

Apesar desta constante alteração do processo de desenvolvimento, as alterações que a versão 3.0 traz vão ao encontro das tendências da *web*, com a adoção dos ES Modules como solução de importação dos componentes em detrimento do HTML Imports, o desenvolvimento de componentes em JS em vez de HTML e a alteração do gestor de pacotes passando a usar o NPM em vez do Bower que se encontra descontinuado. Assim, é esperado que o Polymer estabilize e consiga dar mais garantias futuras (“Polymer library - Polymer Project,” 2017).

Embora a especificação dos Web Components existir desde 2012, estes ainda não se afirmaram na comunidade *web* como um processo de desenvolvimento de componentes *cross framework*. Tal deve-se à falta de consenso na comunidade quanto aos quatro *standards*. O *standard* mais controverso é o Shadow DOM pois tem sofrido várias alterações ao longo do tempo, sendo que o estado atual oferece de forma limitada opções de estilização, pelo que é frequente se optar pela não utilização do mesmo.

5. Processo de avaliação do Web Component

O cenário ideal para avaliar o componente seria acompanhar a introdução da tecnologia nas equipas distribuindo temporalmente várias técnicas de recolha de dados. Desta forma seria possível analisar com maior profundidade o processo de adoção dos Web Components, bem como a viabilidade dos mesmos na Altice Labs. Contudo, o fluxo de trabalho elevado da empresa, que é exigido pela prestigiada posição global da mesma, requer a adaptação dos métodos de recolha de dados de modo a produzir resultados fidedignos.

Posto isto, foi encontrada uma solução com vista a promover o diálogo e discussão de ideias entre os elementos presentes, conseguindo assim um *feedback* com maior valor como um todo.

Com base nos constrangimentos abordados, a solução encontrada para avaliar o protótipo passa por uma sessão de recolha de dados dividida em três fases distintas. A primeira é a introdução ao tema dos Web Components, para uma contextualização da tecnologia e demonstração das capacidades, bem como as vantagens e desvantagens (AP A - Apresentação da sessão de recolha de dados). Esta primeira fase é de elevada importância devido à diversidade tecnológica dos participantes, que têm ideais e *workflows* distintos.

A segunda fase passa pela realização da recolha de dados qualitativos através de um *focus group*. Este método de investigação foi escolhido porque promove a discussão de ideias entre os participantes. Dada a diversidade cultural e tecnológica dos mesmos, é importante que estes manifestem as suas expectativas, opiniões e o posicionamento quanto ao tema em questão. O *focus group* é bastante útil para avaliar o primeiro contacto dos participantes com o tema pois o seu processo baseia-se na exploração de ideias e pensamentos dos utilizadores.

Realizado o *focus group*, a terceira e última fase é um inquérito por questionário (AP B - Questionário da sessão de recolha de dados) tendo como objetivo a obtenção de dados quantitativos sobre a opinião dos participantes quanto aos Web Components. Este questionário por sua vez é composto por quatro secções. A primeira secção recolhe informação do perfil do participante,

como o departamento onde o mesmo está inserido na Altice Labs, a *framework* de desenvolvido que este usa e o produto onde está a trabalhar. A segunda secção aborda a satisfação atual do participante quanto ao seu processo de desenvolvimento e implementação de componentes no produto onde trabalha. A terceira secção é composta pelo TAM (Technology Acceptance Model). Este modelo foca-se em dois aspetos fundamentais, a utilidade percebida e a facilidade de uso percebida como fatores que determinam a aceitação de uma tecnologia por parte do utilizador. Segundo Davis, as pessoas tendem a adotar uma tecnologia quando, no entender delas, a mesma vai melhorar a performance do seu trabalho. Além da avaliação de utilidade, é realizada também uma avaliação de facilidade de uso que irá balançar a primeira, ou seja, uma tecnologia útil pode não ser viável devido a custos relativos ao esforço necessário para a adoção da mesma (Davis, 1989).

Tabela 6 - Etapas da sessão de recolha de dados.

Etapa	Participantes	Objetivo	Dados
Introdução à tecnologia Web Components	Desenvolvedores Altice Labs	Contextualizar os participantes sobre o tema e realizar uma demonstração do protótipo	**-----**
Focus group		Promover a discussão entre os participantes para ocorrer troca de opiniões entre equipas diferentes	Dados qualitativos sobre a opinião dos participantes
Questionário		Registo de dados quantitativos e anónimos para análise detalhada	Dados sobre a viabilidade e aplicabilidade dos Web Components

5.1 Caracterização dos participantes

Na sessão participaram dezasseis colaboradores da Altice Labs, que representaram todos os departamentos da empresa à exceção do departamento de Desenvolvimento de Negócio por não se enquadrar no âmbito do projeto.

Tabela 7 - Caracterização dos participantes.

Grupo	Nº de participantes	Departamento	Tecnologia
1	8	SSO - Sistemas de Suporte às Operações	AngularJS Angular 2 Angular 5 Angular 6 Google Web Toolkit 2.6 JavaServer Pages Play Framework
2	4	DIT - Digital, Internet e Televisão	Nativa
3	2	CTI - Coordenação Tecnológica e Inovação	Nativa
4	1	SRP - Serviços de Rede e Plataforma	Angular 5
5	1	DSR - Desenvolvimento e Implementação de Sistemas de Rede	Angular JS

5.2 Resultados

Nesta secção iremos descrever como decorreram as várias etapas descritas na tabela 6, bem como analisar os dados obtidos dos participantes.

5.2.1 Focus group

Durante o *focus group* foi referido o receio quanto à lenta evolução dos Web Components, pois desde a primeira especificação em 2012 era esperada a consolidação dos *standards* e o suporte dos *browsers*, que não se verifica. Dada a falta deste suporte, o estado atual dos *polyfills,* cuja tarefa é aumentar a compatibilidade da tecnologia com os *browsers*, é preocupante no que diz respeito à estilização. Estes não são capazes de replicar funcionalidades nativas dos *browsers* quando o Web Component tem o Shadow DOM ativo.

Por estes motivos, os participantes consideraram que a utilização do Shadow DOM não é uma opção, apesar das vantagens quanto ao encapsulamento e garantias de normalização do aspeto gráfico e comportamento ao longo dos vários produtos da empresa, a falta de suporte dos *browsers* não permite a sua aplicação. No entanto a interoperabilidade que a tecnologia fornece continua a ser de elevado interesse para a organização, sendo referido que “um aspeto fundamental é a potencialidade de um Web Component com Light DOM, porque resolve a maior parte dos problemas que a empresa tem”.

Posto isto, um participante disse que é necessário “mudar a forma como nós pensamos, idealizamos os componentes” e “mudar a forma como modularizamos um componente”, pois nem todos os componentes desenvolvidos na empresa servem para o caso de estudo dos Web Components.

Pergunta 1 - "É viável a adoção dos Web Components como solução de compatibilidade na implementação de componentes *cross framework*?"

Todos os participantes não acham viável o uso do Shadow DOM pois não há um nível de suporte dos *browsers* significativo e este *standard* não está consolidado quanto às opções de estilização. Por isso a adoção dos Web Components como um todo não é viável devido às limitações atuais existentes que são consequência da falta de suporte dos *browsers*. Assim, a não ser que seja garantido o controlo sobre a utilização dos mesmos pelos utilizadores finais, a melhor opção seria o uso de Web Components com recurso a Light DOM. Esta opção removeria a mais-valia que é o encapsulamento dos componentes, continuando a existir a possibilidade de colisão de estilos. No entanto foi também demonstrada segurança quanto a essa questão pois esta está presente no processo de desenvolvimento há bastante tempo e as equipas desenvolveram boas práticas para evitar que tal seja um problema.

Pergunta 2 – "Que obstáculos os Web Components criam no processo de desenvolvimento e implementação de componentes?"

Como em qualquer tecnologia, um dos obstáculos mencionados por um participante, que pertence ao departamento de Sistemas de Suporte às Operações, é o período de aprendizagem da mesma. Tal é um fator relevante a ter em conta e que ganha mais relevância numa equipa que atualmente gere várias tecnologias. No entanto, sendo esta tecnologia baseada apenas em *standards* nativos da *web* e recorrendo a tecnologias nativas (HTML, CSS e JS), tal obstáculo pode ser considerado pouco significativo.

Outro obstáculo foi o já referido anteriormente que passa pelo suporte dos *browsers*. A implementação de funcionalidades referentes aos Web Components, de forma global, é ainda fraca, pelo que é necessário recorrer a *polyfills* gerando assim obstáculos ao desenvolvimento.

Pergunta 3 – "Que vantagens trazem os Web Components ao processo de desenvolvimento e implementação de componentes?"

Para os participantes as vantagens são claras. Os participantes do Grupo 1 referiram a padronização dos componentes e consequente sistematização do processo de desenvolvimento. O Grupo 3, responsável pela gestão de tecnologias adotadas pela empresa, mencionou que tal iria trazer um aumento substancial de produtividade ao mesmo tempo que reduz os custos de desenvolvimento para a organização. Por fim, o Grupo 2, constituído por uma equipa transversal de usabilidade presente na sessão, referiu que uma mais-valia seria a garantia de que os componentes desenvolvidos comportar-se-iam da mesma forma em produtos distintos.

5.2.2 Questionário

A primeira secção composta pela informação do participante teve como objetivo garantir que estavam presentes elementos dos vários departamentos de desenvolvimento da Altice Labs, para se recolher dados que representem a empresa como um todo. Tal foi confirmado com, pelo menos, cada departamento a ser representado por um colaborador.

Pergunta 2 - "Que *framework(s)* de desenvolvimento utiliza?"

Como previsto, na questão seguinte referente à *framework* de desenvolvimento usada por cada participante, observou-se também uma diversidade elevada, representada na Figura 13, com *frameworks* distintas desde o mais recente Angular 6 até ao GWT (Google Web Toolkit) ou JSP (JavaServer Pages).

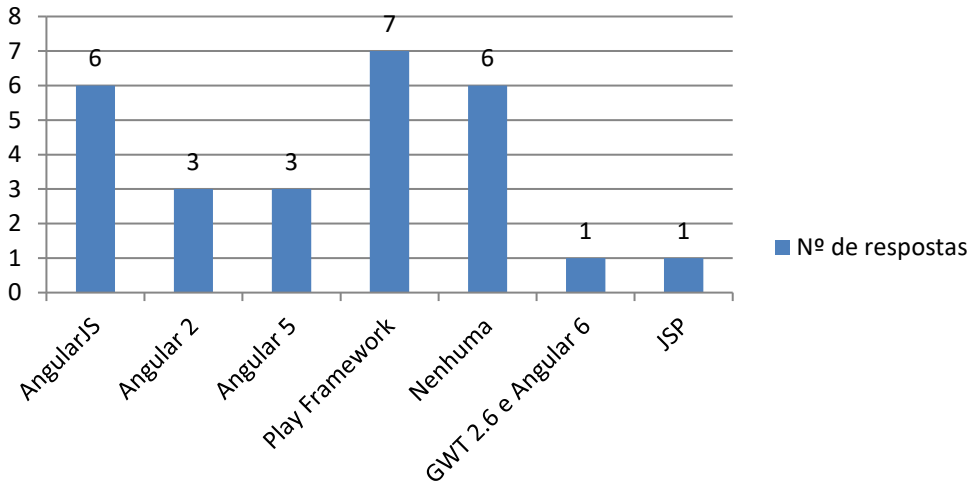


Figura 13 - Histograma da pergunta 2.

Pergunta 3 - "Em que projetos/produtos trabalha atualmente?"

Da mesma forma, foi possível confirmar que os participantes encontram-se em desenvolvimento nos vários produtos da organização, estando os dezasseis participantes distribuídos por treze produtos.

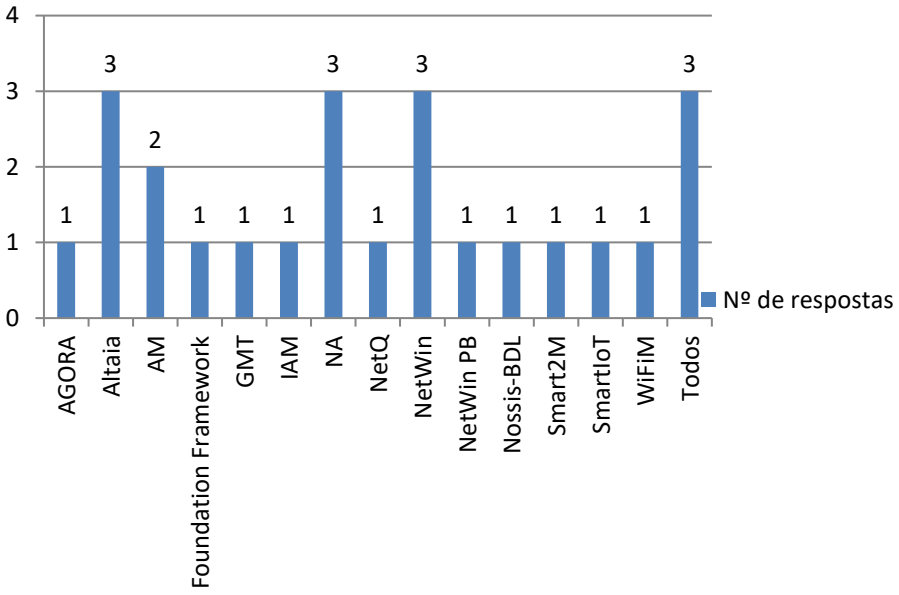


Figura 14 - Histograma da pergunta 3.

Pergunta 4 - "Como classifica o seu processo atual de implementação de componentes?"

A secção seguinte abordou a satisfação quanto ao processo de desenvolvimento atual de implementação de componentes. Desta forma é possível concluir se os participantes sentem a necessidade de alterar o seu processo de desenvolvimento. Foi utilizada uma escala de Likert com seis valores possíveis em que o menor remetia para "Insatisfação" e o maior para "Satisfação". A média obtida foi de 2,56 e a mediana 3, indicando que os participantes estão insatisfeitos.

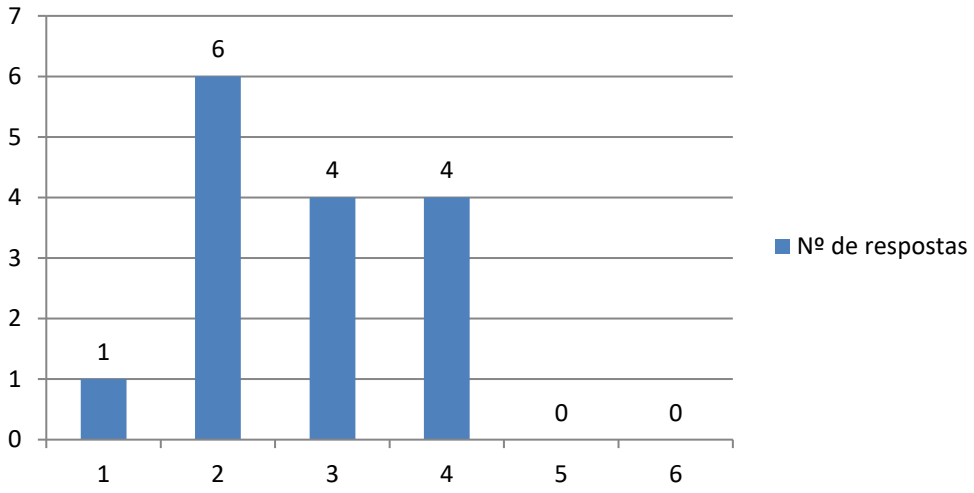


Figura 15 - Histograma da pergunta 4.

Pergunta 5 - "Porquê?"

Na questão seguinte foi pedido que justificassem a resposta anterior. Estas centraram-se no descontentamento por não ser possível reutilizar os componentes, sendo necessário desenvolver o mesmo componente em várias frameworks. Além disso, foi referido o risco de colisão de dependências dos componentes. Estas questões resultam num esforço extra necessário para manter o processo de desenvolvimento atual.

Pergunta 6 - "Como classifica o *workflow* do mesmo?"

Por fim a última questão desta secção centrou-se em avaliar o fluxo de trabalho. Foi usada a mesma escala da pergunta inicial sendo que os indicadores passaram a ser "Complexo" no menor valor e "Simples" no mais elevado. A média obtida foi de 3,07 e a mediana 3, revelando alguma complexidade do *workflow*.

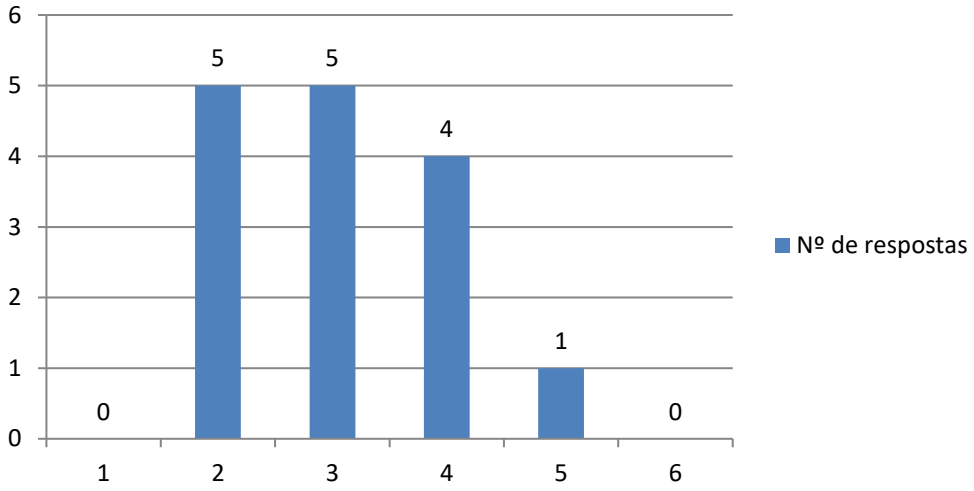


Figura 16 - Histograma da pergunta 6.

A terceira secção, composta pelo TAM, avaliou a utilidade percebida dos Web Components bem como a sua facilidade de uso percebida. Este modelo é composto por doze perguntas baseadas numa escala de Likert, de um a sete, em que o menor valor remete para "Provável" e o maior para "Improvável". A média das respostas obtidas foi de 3,15, a mediana foi 3,13 e o desvio-padrão registado foi 0,38. No geral há uma boa aceitação da tecnologia. No entanto, como é possível verificar no gráfico da Figura 17, a pergunta oito e dez destacaram-se pela negativa.

A pergunta oito questiona se o participante "Acharia fácil conseguir que os Web Components fizessem o que eu quero que eles façam." e a pergunta dez questiona se "Acharia os Web Components flexíveis de trabalhar.". Estas tiveram uma média de 3,69 e 4, respetivamente. Tal deve-se às limitações apresentadas anteriormente quanto às opções de implementação de Web Components.

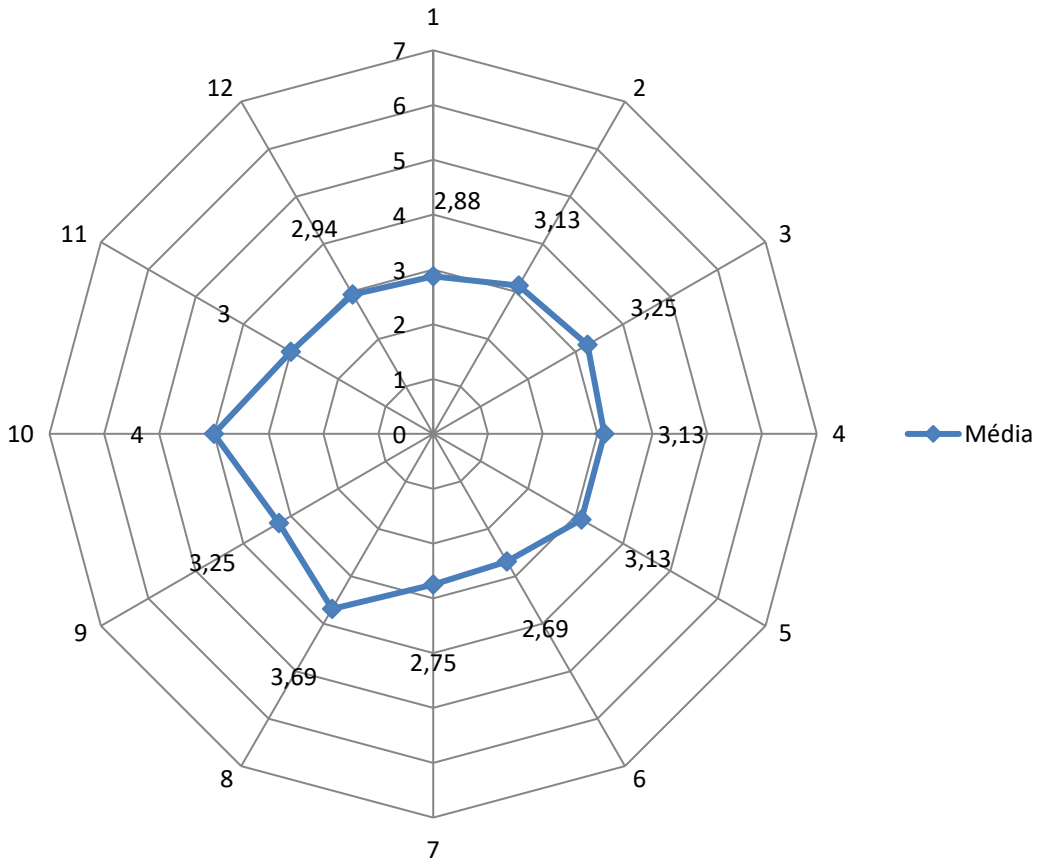


Figura 17- Resultados do modelo de aceitação da tecnologia.

Pergunta 19 - "Como classifica o workflow na implementação de Web Components?"

Por fim a última secção abordou os Web Components como solução de compatibilidade na implementação de componentes *cross framework*. A primeira questão passou por analisar o *workflow* percebido dos Web Components por parte dos participantes. Foi usada a mesma escala e indicadores da pergunta relacionada com o *workflow* atual dos participantes. Os dados obtidos revelaram uma média de 3,88, a mediana foi 4 e o desvio-padrão 1,20. Em comparação com o processo atual, os participantes acharam o fluxo de trabalho dos Web Components mais simples que o atual.

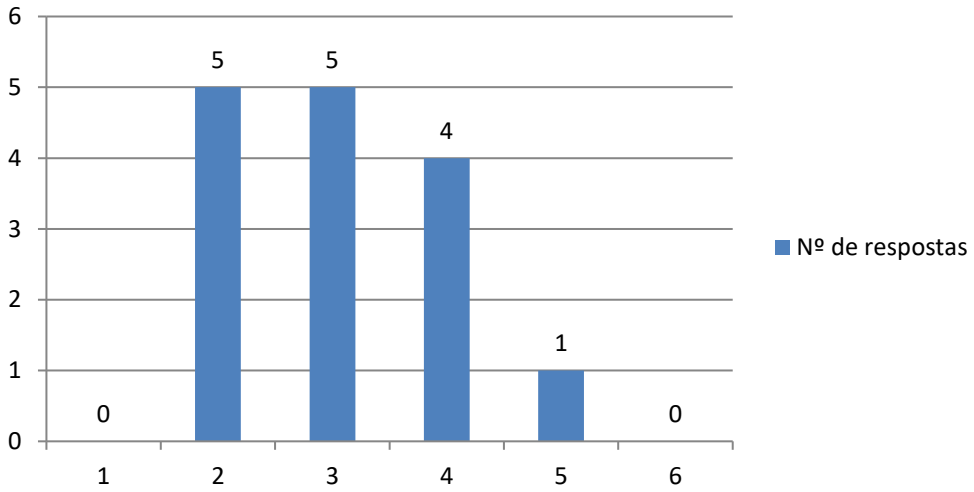


Figura 18 - Histograma da pergunta 19.

Pergunta 20 - "Que diferenças existem entre a implementação de um componente baseado na tecnologia da sua framework de trabalho e um Web Component?"

Quando questionados pelas diferenças entre o processo atual de implementação e o processo de implementação através de Web Components, os participantes mencionaram a potencialidade de reutilização de componentes, a uniformização e centralização do código, bem como o facto de os Web Components se basearem em JS nativo.

Pergunta 21 - "Do seu ponto de vista, qual/quais a(s) maior(es) dificuldade(s) de implementação de um Web Component?"

Quando questionados sobre quais as maiores dificuldades de implementação de um Web Component, a opção mais vezes respondida foi "Suporte dos *browsers*" com 15 respostas, seguido de "Falta de solidificação dos standards e contínua alteração dos mesmos" com 13 respostas e por fim "Falta de opções de personalização de estilos" com 10 respostas. É portanto flagrante o suporte dos *browsers* como o maior obstáculo ao desenvolvimento de Web Components. De referir que a opção "Adaptação à tecnologia" apenas foi

selecionada uma vez, o que permite cimentar a facilidade de uso dos Web Components.

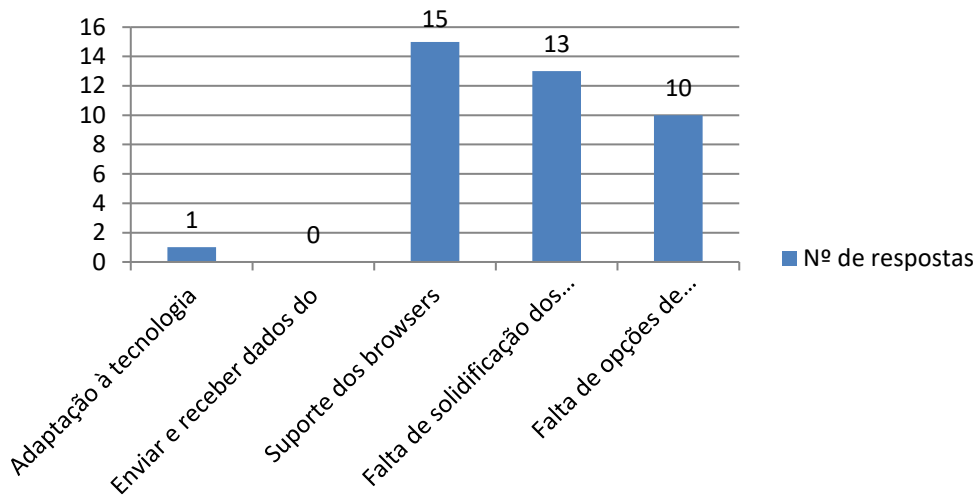


Figura 19 - Histograma da pergunta 21.

Pergunta 22 - "Do seu ponto de vista, quais os maiores benefícios do uso de Web Components?"

Por outro lado, os participantes foram também questionados sobre quais os maiores benefícios do uso de Web Components. Com 11 respostas "Interoperabilidade entre *frameworks*" destacou-se como a maior vantagem percebida pelos participantes. Pelo contrário, a opção "Encapsulamento de CSS" não obteve respostas, devendo-se ao estado do Shadow DOM.

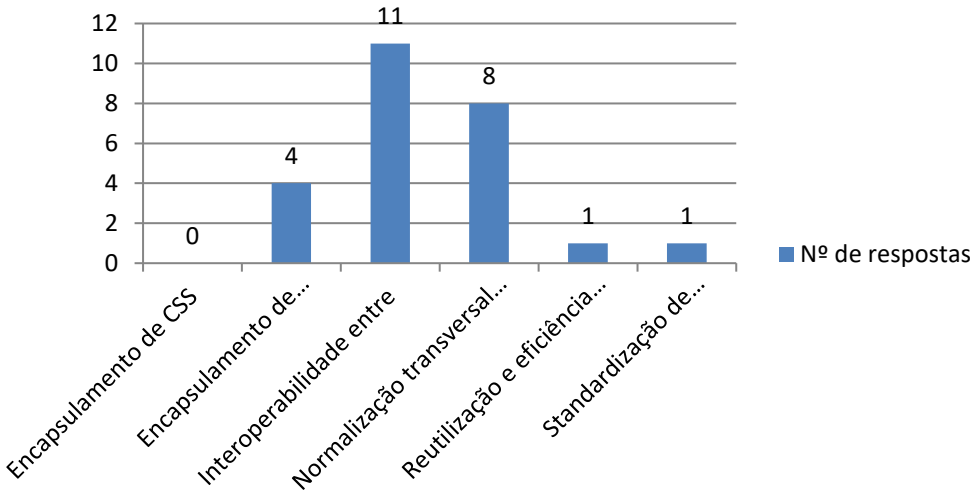


Figura 20 - Histograma da pergunta 22.

Pergunta 23 - "Do seu ponto de vista, como classifica a viabilidade dos Web Components como solução de compatibilidade na implementação de componentes na framework que utiliza?"

Por fim, foi questionado aos participantes como classificam a viabilidade dos Web Components como solução de compatibilidade na implementação de componentes na *framework* que utilizam. Foi usada a mesma escala de Likert anterior, sendo os indicadores "Nada viável" para o valor 1 e "Muito viável" para o valor 6. A média obtida foi 4,14, a mediana foi 4,5 e o desvio-padrão 1,03. Estes valores permitem concluir que apesar do estado atual dos Web Components não ser o mais desejado, os participantes revêem-se a adotar a tecnologia.

Pergunta 24 - "Porquê?"

Quando lhes foi pedido que justificassem, estes mencionaram que "o fator da compatibilidade teria bons ganhos de tempo", "potenciona a reutilização" e que "a transversalidade e compatibilidade com qualquer outra *framework*" seriam úteis.

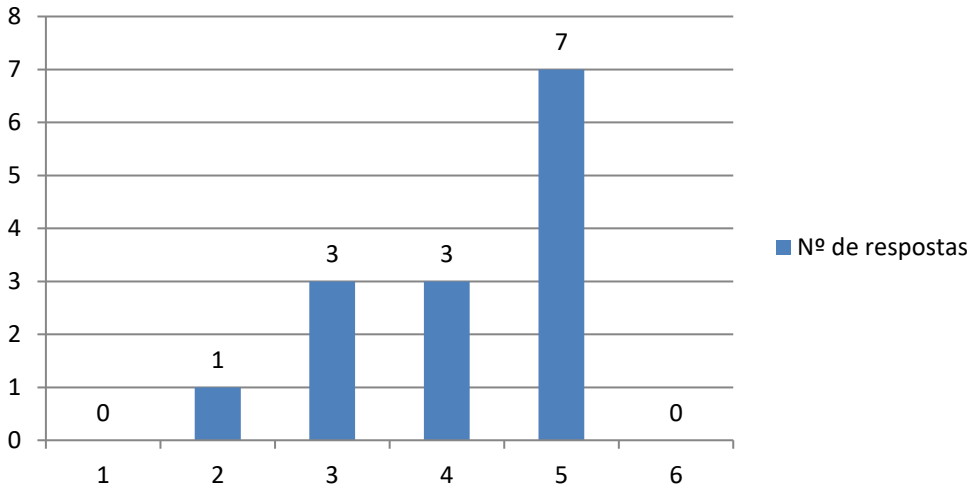


Figura 21 - Histograma da pergunta 24.

Esta sessão de recolha de dados permitiu confirmar a grande diversidade tecnológica existente na Altice Labs, necessária para dar manutenção e evoluir a suíte de produtos da mesma. Por outro lado, foi explícito o descontentamento dos colaboradores face ao atual processo de desenvolvimento devido a não haver uma uniformização do mesmo. Desta forma, estes mostraram-se recetivos à tecnologia Web Components pois esta propõe-se a uniformizar todo o processo. No entanto, a falta de consolidação dos *standards* causa alguma inquietação pois a adoção de uma tecnologia numa organização desta magnitude envolve custos elevados e a partir do momento que se toma uma decisão sobre uma tecnologia esta vai fazer parte do processo de desenvolvimento a médio/longo prazo.

Desses *standards*, o Shadow DOM, responsável por fornecer o encapsulamento do componente e um dos pontos fortes dos Web Components, foi descartado pelos participantes. Tal deve-se à forma de implementação que no seu estado atual é demasiado complexa e limitada, sentindo os participantes que, apesar de uma vantagem, não é uma prioridade encapsular os componentes pois já têm implementado nas equipas boas práticas que minimizam o risco de colisões.

Posto isto, os participantes acham viável o uso de parte dos *standards* dos Web Components como solução para resolver os problemas do processo atual de desenvolvimento e implementação de componentes.

6. *Guidelines* para desenvolvimento de Web Components

Sendo a tecnologia Web Components um conjunto de *standards* independentes, cada um com um nível de desenvolvimento e adoção por parte da comunidade diferente, torna-se relevante explicitar as melhores práticas de desenvolvimento.

São claras as vantagens do uso dos Custom Elements como solução de interoperabilidade entre *frameworks* de desenvolvimento. O suporte por parte dos *browsers* não é o melhor, no entanto os *polyfills* cobrem essa falha sem custos relevantes associados, permitindo assim um componente ser criado uma vez e reutilizado em diversos contextos.

A utilização do Shadow DOM como opção para encapsular o JS e o CSS apenas é viável se for possível controlar os *browsers* que os utilizadores irão usar, pois o suporte dos mesmos é baixo e, neste caso, os *polyfills* atualmente exigem uma restruturação tanto do componente como do documento onde o mesmo seria inserido, nos *browsers* que não o suportam nativamente. Além disso os *polyfills* não fornecem funcionalidades chave do Shadow DOM referidas anteriormente.
Assim, apenas é recomendado o uso deste *standard* como técnica de composição para melhor organização do componente e usar a *tag* “slot” por forma a abstrair o conteúdo do mesmo, pois sem Shadow DOM não é possível usar. De referir que os *polyfills* cobrem esta funcionalidade em *browsers* que não a suportam nativamente.

Quanto à comunicação dos componentes com a *framework* onde estão inseridos, deve seguir-se um conjunto de boas práticas que evitam a ocorrência de dessincronização dos mesmos. Para atribuir estados a um Web Component a melhor prática passa pelo uso de propriedades em vez de atributos pois ao contrário das propriedades, os atributos apenas refletem o seu estado inicial mesmo que sejam alterados, podendo induzir em erro o ambiente externo. O uso de atributos é conveniente quando se quer reter o estado inicial de um componente. Quanto à forma de transmitir informação do Web Component para o

exterior, tal deve ser realizado através da propagação de eventos. Desta forma é possível preparar o ambiente para executar algo quando escutar o evento pretendido (Dodson, 2016).

Por fim, a biblioteca Polymer tem como objetivo minimizar o tempo despendido na criação do *boilerplate* do componente, bem como na preparação do mesmo para produção. Incorpora nativamente o Shadow DOM no componente, torna a estrutura do mesmo mais organizada/fácil de interpretar e a ferramenta de linha de comandos torna-se bastante útil na fase de desenvolvimento. Por outro lado, a carga que exerce a mais no *browser* e as mudanças significativas entre versões são aspetos importantes a ter em conta. Posto isto, a utilização da biblioteca depende da preferência pessoal.

7. Conclusões

A premissa desta investigação teve por base os desafios impostos pela diversidade tecnológica presente na Altice Labs no processo de desenvolvimento e implementação de componentes. Num contexto empresarial de elevada magnitude é imperativa a manutenção dos produtos a longo prazo, resultando ao longo do tempo na criação de ambientes de desenvolvimento paralelos, distintos e incompatíveis. Apesar de estes partilharem do mesmo espectro visual da empresa e grande parte dos elementos que compõem as suas interfaces serem transversais, na prática tal não se verifica devido à incompatibilidade tecnológica inerente. Assim, propôs-se a investigação de Web Components como solução de compatibilidade de componentes *cross framework*. Esta investigação teve como objetivo determinar as potencialidades da tecnologia quando aplicada no contexto empresarial da Altice Labs. Sendo a tecnologia relativamente recente, observou-se uma escassez de literatura relacionada com o tema, sendo notória a falta de casos de estudo, o que indica o estado pouco consolidado da tecnologia e uma adesão reduzida da mesma.

Dado o caráter prático da investigação, o desenvolvimento do protótipo revelou-se fundamental pois permitiu ter uma perspetiva mais detalhada das vantagens e desvantagens, bem como as limitações atuais dos Web Components. São claras as vantagens de uso de Web Components num ambiente de desenvolvimento marcado pela diversidade tecnológica, pois estes têm a capacidade de otimizar os recursos investidos ao uniformizar todo o processo. Dito isto, o desenvolvimento do protótipo permitiu também enaltecer as dificuldades dos Web Componentes quanto à adoção por parte dos *browsers*, sendo atualmente a maior limitação da tecnologia e que põe em causa o uso da mesma.

Analisado em detalhe o estado atual dos Web Components, partilhou-se os resultados com as várias equipas de desenvolvimento numa sessão de recolha de dados, com o objetivo de gerar discussão sobre o tema e recolher *feedback* quanto à recetividade e adoção do mesmo. Os dados obtidos confirmaram a

necessidade de a Altice Labs reformular o processo de desenvolvimento de componentes *cross framework*. Para além disso, os colaboradores da empresa demonstraram interesse na tecnologia apresentada, apesar das limitações atuais da mesma, pois é composta por *standards* independentes, o que a torna flexível e adaptável a diferentes casos de estudo.

Limitações do estudo

Esta investigação representou um desafio, pois tratou-se de uma investigação exploratória, com escassa literatura onde se basear, num contexto empresarial caracterizado por um elevado ritmo de trabalho. No entanto, estes obstáculos foram superados através da adaptação, quando necessária, do planeamento. Referente à sessão de recolha de dados, esta não foi executada de acordo com uma metodologia "ideal" pois num âmbito empresarial nem sempre é possível aplicar as metodologias escolhidas devido à complexidade estrutural e de agenda das equipas.

Perspetivas de trabalho futuro

De modo a aferir-se com rigor a interoperabilidade entre diferentes ambientes de desenvolvimento, o protótipo deve ser testado em ambiente controlado nas diferentes frameworks adotadas pela Altice Labs, nomeadamente as várias versões do Angular presentes nos produtos da empresa. Tal deve explorar possíveis limitações não especificadas, bem como obstáculos que comprometam a viabilidade de adoção da tecnologia.

Não obstante, a avaliação do protótipo executada pelos colaboradores da Altice Labs deve incluir uma etapa dedicada à integração do mesmo em projetos que os participantes estejam a desenvolver, de forma a expor a tecnologia aos hábitos de desenvolvimento de cada equipa. Simultaneamente, esta testará a integração num meio real onde as potencialidades dos Web Components serão postas à prova.

Por fim, com vista a obter um maior nível de fidelidade dos resultados, os requisitos técnicos e funcionais devem ser cumpridos integralmente, garantindo que não existe uma omissão de limitações da tecnologia.

Em conclusão, atualmente os Web Components estão a ganhar maior tração e a captar a atenção da comunidade através da consolidação dos seus *standards* e o acordo dos mais importantes *browsers* em implementar os mesmos. Apesar das suas limitações atuais, a versatilidade é uma característica que torna viável a sua implementação pois a tecnologia é capaz de se adaptar a cada contexto específico.

Portanto, esta requer um acompanhamento do seu desenvolvimento a fim de se avaliar as evoluções incrementais futuras, para finalmente se afirmar como uma solução de compatibilidade na implementação de componentes *cross framework*.

Referências Bibliográficas

3Scale, N. S. L. (2011). What is an API ? Your guide to the Internet Business (R)evolution . *3 Scale Infrastructure for the Programmable Web*, 1–9.

Aghaei, S. (2012). Evolution of the World Wide Web : From Web 1.0 to Web 4.0. *International Journal of Web & Semantic Technology*. https://doi.org/10.5121/ijwest.2012.3101

Akritidis, L., Katsaros, D., & Bozanis, P. (2011). Modern web technologies. *Studies in Computational Intelligence*, *331*, 83–107. https://doi.org/10.1007/978-3-642-17551-0_4

Altice. (2016). Altice Labs | Sobre Nós. Retrieved February 6, 2018, from http://www.alticelabs.com/pt/sobre.html

Altice Labs Whitepaper. (2014). Retrieved February 5, 2018, from http://www.alticelabs.com

Benhaddi, M. (2017). Web of goals: A proposal for a new highly smart web BT - 19th International Conference on Enterprise Information Systems, ICEIS 2017, April 26, 2017 - April 29, 2017, *2*(Iceis), 687–694. https://doi.org/10.5220/0006250306870694

Data binding - Polymer Project. (n.d.). Retrieved February 5, 2018, from https://www.polymer-project.org/2.0/docs/devguide/data-binding

Davis, F. D. (1989). Perceived Usefulness , Perceived Ease of Use , and User Acceptance of Information Technology. *MIS Quarterly*, *13*(3), 319–340. https://doi.org/10.2307/249008

De Villiers, M. R., & Harpur, P. . (2013). Design-based research – the educational technology variant of design research: Illustrated by the design of an m-learning environment. *Proceedings of the South African Institute for Computer Scientists and Information Technologists Conference. ACM*, 252–261. https://doi.org/10.1145/2513456.2513471

Deveria, A., & Schoors, L. (2018). Can I use. Retrieved May 17, 2018, from https://caniuse.com/#search=Web components

Dodson, R. (2016). Custom Elements That Work Anywhere. Retrieved May 17, 2018, from http://robdodson.me/interoperable-custom-elements/

Ian Allen. (2018). The Brutal Lifecycle of JavaScript Frameworks. Retrieved February 5, 2018, from https://stackoverflow.blog/2018/01/11/brutal-lifecycle-javascript-frameworks/

INKONIQ. (2017). Uncomplicate the WEB Using Web Components. Retrieved February 4, 2018, from https://medium.com/inkoniq-blog/uncomplicate-the-web-using-web-components-4d5f7edaac05

Justin Fagnani. (2017). Polymer 3.0 preview. Retrieved February 5, 2018, from https://www.polymer-project.org/blog/2017-08-22-npm-modules

Lee, T. B. (1998). The World Wide Web: A very short personal history. Retrieved from http://www.w3.org/People/Berners-Lee/ShortHistory. html

Martínez-Ortiz, A. L. ., Lizcano, D. ., Ortega, M. ., Ruiz, L. ., & López, G. . (2016). A quality model for web components. *ACM International Conference Proceeding Series*, *Part F1263*, 430–432. https://doi.org/10.1145/3011141.3011203

MDN DOCS. (2017). Css | Mdn. Retrieved February 4, 2018, from https://developer.mozilla.org/pt-BR/docs/Web/CSS

Pankaj Parashar. (2015). Building Custom Web Components with X-Tag. Retrieved February 5, 2018, from https://www.sitepoint.com/building-custom-web-components-with-x-tag/

Polymer library - Polymer Project. (2017). Retrieved May 17, 2018, from https://www.polymer-project.org/2.0/docs/devguide/feature-overview

Rocha, C. (2016). Altice Labs: o ponto central da inovação da Altice fica em Aveiro - PCGuia.

Savage, T. (2015). Componetizing the Web. *Communications of the ACM*, 1–20.

Shugart, T. (n.d.). SkateJS. Retrieved May 17, 2018, from

http://skatejs.netlify.com/

Slim.js. (n.d.). Retrieved February 5, 2018, from http://slimjs.com/#/getting-started

Stencil. (n.d.). Retrieved February 5, 2018, from https://stenciljs.com/

Swisher, J., & Mills, C. (2018). Using shadow DOM - Web Components | MDN. Retrieved May 17, 2018, from https://developer.mozilla.org/en-US/docs/Web/Web_Components/Using_shadow_DOM

webcomponents.org. (n.d.). Polyfills — WebComponents.org. Retrieved February 4, 2018, from http://webcomponents.org/polyfills/

webcomponents.org. (2018). Retrieved May 17, 2018, from https://www.webcomponents.org/

Apêndices

Apêndice A – Apresentação da sessão de recolha de dados

Apêndice B – Questionário da sessão de recolha de dados

Apêndice C – Consentimento informado da sessão de recolha de dados

Apêndice D – Instruções para aceder ao protótipo *online*

Apêndice A – Apresentação da sessão de recolha de dados

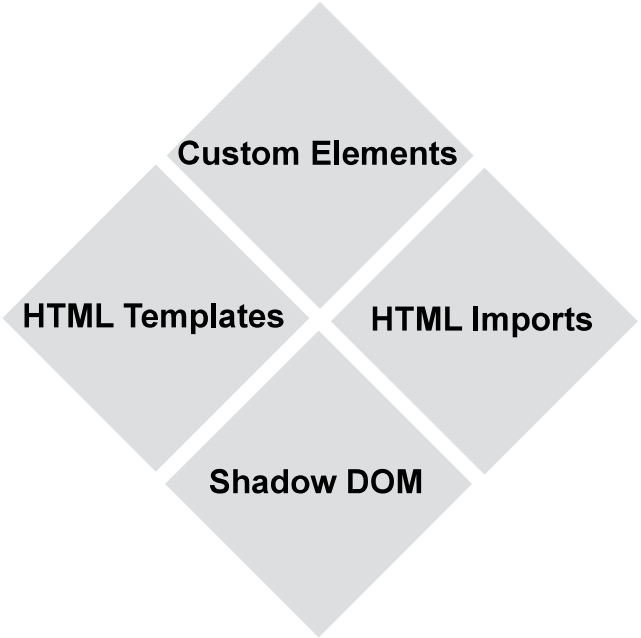
Web Components - Solução de compatibilidade na implementação de componentes *cross framework*

15 de Maio de 2018

altice labs

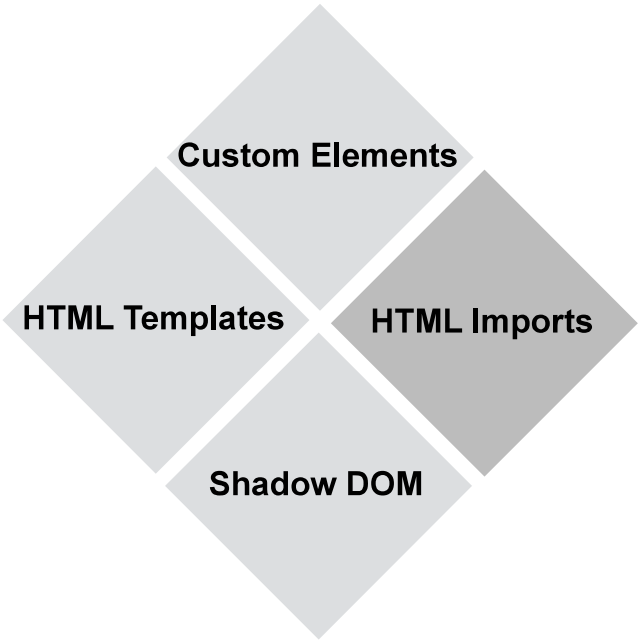
O que é?

- **W3C (Standards World Wide Web Consortium);**
- **Conjunto de tecnologias que permitem a criação de componentes reutilizáveis.**



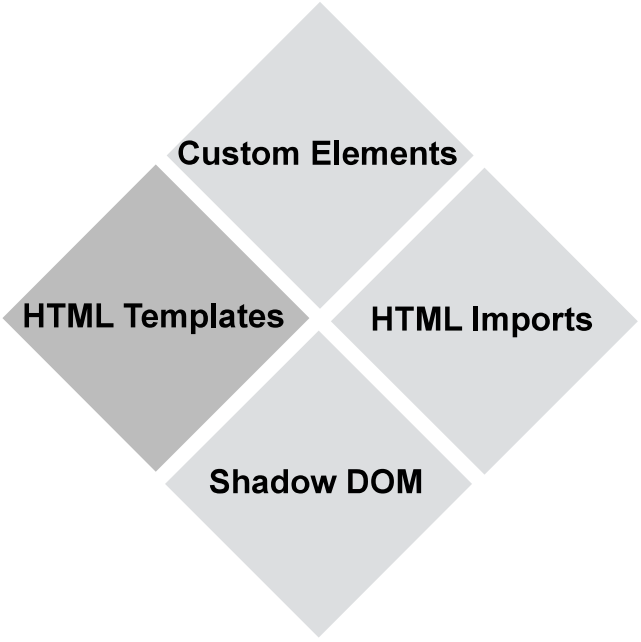
HTML Imports

- Possibilita o desenvolvimento de Web Components com recurso a ficheiros HTML;
- HTML Imports pretende standardizar a importação dos recursos e resolver a duplicação dos mesmos;
- Uma fraca adesão no suporte por parte dos browsers e da comunidade fez este standard cair em desuso.



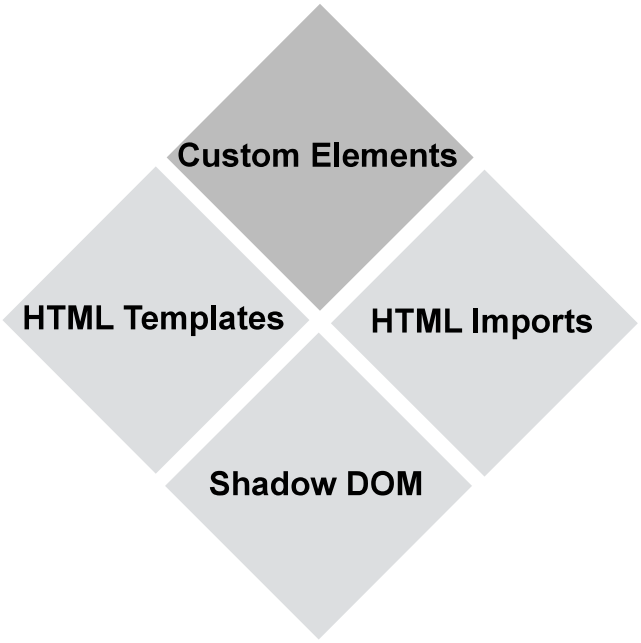
HTML Templates

- Permite a declaração de porções de código para futura utilização;
- Incentiva ao desenvolvimento de componentes com propósitos claros e simples.



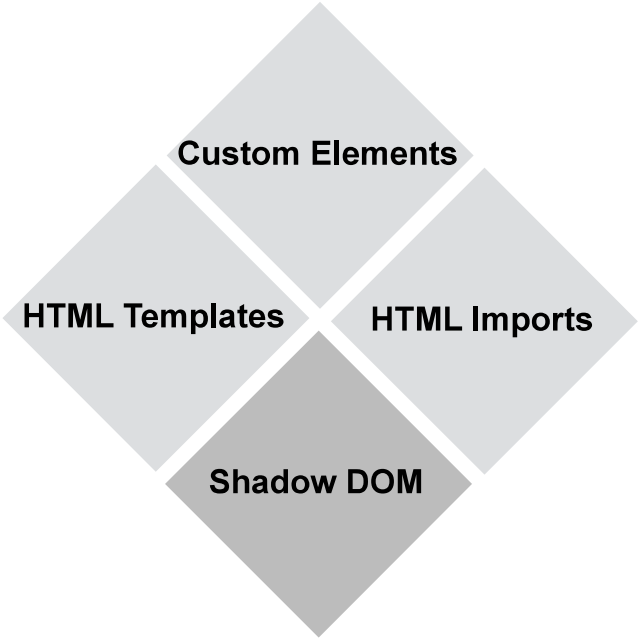
Custom Elements

- Base de um Web Component.
- Torna possível criar elementos e a consequente utilização no DOM como se tratasse de uma *tag* HTML nativa.
- Permite o uso de um componente em diversos contextos e tecnologias (Angular, React, etc) pois é interpretado como uma tag HTML.



Shadow DOM

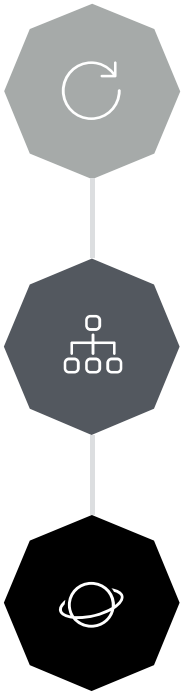
- *Standard* que permite o encapsulamento do componente
- Útil para assegurar o bom funcionamento do componente



Suporte dos browsers

Browser support	CHROME	OPERA	SAFARI	FIREFOX	EDGE
TEMPLATES	STABLE	STABLE	STABLE	STABLE	STABLE
CUSTOM ELEMENTS	STABLE	STABLE	STABLE	POLYFILL DEVELOPING	POLYFILL CONSIDERING
SHADOW DOM	STABLE	STABLE	STABLE	POLYFILL DEVELOPING	POLYFILL CONSIDERING
<SCRIPT TYPE="MODULE">	STABLE	STABLE	STABLE	DEVELOPING	STABLE
HTML IMPORTS	STABLE	STABLE	POLYFILL ON HOLD	POLYFILL ON HOLD	POLYFILL CONSIDERING

Para que serve?



Componentes reutilizáveis

Através de Web Components conseguimos criar um componente e reutiliza-lo da mesma forma que se criam componentes no Angular ou React.

Componentes encapsulados

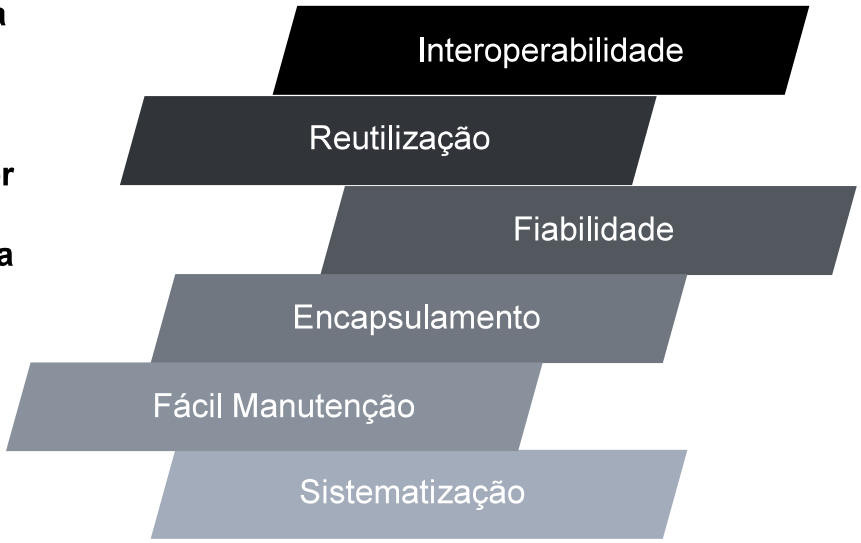
Torna possível o encapsulamento do componente quando integrado noutro código. Desta forma é possível manter a estrutura do componente, bem como os estilos e todo o comportamento associado, evitando conflitos.

Componentes interoperáveis

Num contexto empresarial onde várias equipas trabalham em diferentes projetos recorrendo a diversas tecnologias em simultâneo, é importante criar componentes que sejam capazes de ser implementados em todo o lado.

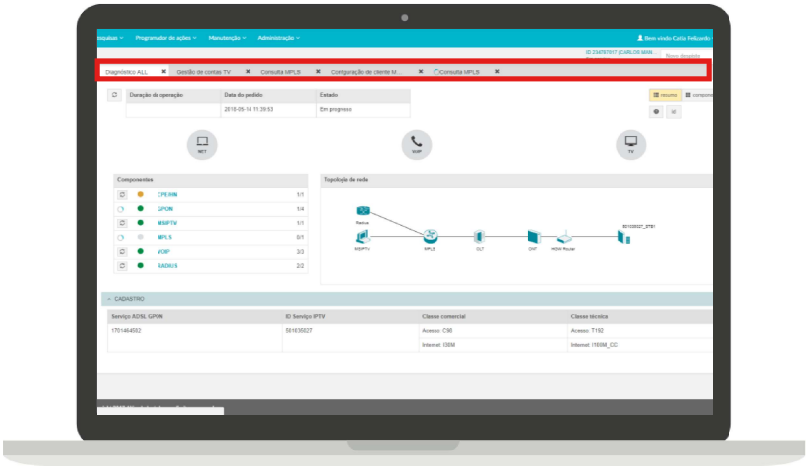
O que resolve?

A criação de componentes numa única tecnologia, que cada uma das *frameworks* possa usar, independentemente dos processos internos impostos por cada tecnologia em particular quer para manutenção, quer para integração dos produtos.



Component Tabs

Com vista a testar os Web Components, foi pensado o desenvolvimento de um protótipo que explorasse a tecnologia. O objetivo passou por replicar um componente da FUXI que abordasse aspetos relevantes.



One more thing…

Polymer

- Suporte a TypeScript
- Polymer CLI
- Organização
- Boilerplate

Protótipo

Rua Eng. José Ferreira Pinto Basto,
3810 - 106 Aveiro Portugal
T: +351 234 403 200
F: +351 234 424 723
www.alticelabs.com

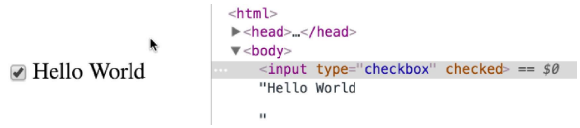
Conclusões

Rua Eng. José Ferreira Pinto Basto,
3810 - 106 Aveiro Portugal
T: +351 234 403 200
F: +351 234 424 723
www.alticelabs.com

The present document has informational purposes only and does not constitute a formal binding offer. The information conveyed does not constitute an undertaking to sell the identified products and services.
The present document is subject to change without notice and Altice Labs cannot be held liable for any possible error or outdated information or for losses or damages of whatever nature resulting from the use of the information.

Como interagir com Web Components?

- Usar propriedades para refletir o estado do componente e atributos apenas para definir configurações iniciais



- Propagar informação através de eventos

```
onClick(e) {
  // This should trigger a setter which updates the DOM
  this.checked = !this.checked;
  this.dispatchEvent(new CustomEvent('checked-changed', {
    detail: { checked: this.checked }, bubbles: false
  }));
}
```

- Usar Shadow DOM para combater problemas de sincronização

Vantagens e Desvantagens

Vantagens

+ Interoperabilidade
+ Manutenção
+ Sistematização
+ Mainstream

Desvantagens

- Suporte dos browsers
- Necessidade de polyfills
- Shadow DOM
- Solidificação dos standards

Focus group

É viável a adoção dos Web Components como solução de compatibilidade na implementação de componentes cross framework?

Focus group

Que obstáculos os Web Components criam no processo de desenvolvimento e implementação de componentes?

Focus group

Que vantagens trazem os Web Components ao processo de desenvolvimento e implementação de componentes?

Questionário

Rua Eng. José Ferreira Pinto Basto,
3810 - 106 Aveiro Portugal
T: +351 234 403 200
F: +351 234 424 723
www.alticelabs.com

Apêndice B – Questionário da sessão de recolha de dados

Questionário

Serve o presente questionário para recolher dados de forma quantitativa, para análise e avaliação dos Web Components como solução de compatibilidade na implementação de componentes Cross Framework.

*Obrigatório

Informação do participante

1. **A que departamento pertence?** *
   Marcar apenas uma oval.

   - ◯ CTI - Coordenação Tecnológica e Inovação
   - ◯ DIT - Digital, Internet e Televisão
   - ◯ DSR - Desenvolvimento e Implementação de Sistemas de Rede
   - ◯ SSO - Sistemas de Suporte às Operações
   - ◯ SRP - Serviços de Rede e Plataforma
   - ◯ Outra: _____

2. **Que framework(s) de desenvolvimento utiliza?** *
   Marcar tudo o que for aplicável.

   - ☐ AngularJS
   - ☐ Angular 2
   - ☐ Angular 5
   - ☐ Play Framework
   - ☐ Nenhuma
   - ☐ Outra: _____

3. **Em que projetos/produtos trabalha atualmente?** *
   Marcar tudo o que for aplicável.

   - ☐ ABC
   - ☐ ACM
   - ☐ AGORA
   - ☐ Altaia
   - ☐ AM
   - ☐ Fiber Gateway
   - ☐ IAM
   - ☐ NA
   - ☐ NetQ
   - ☐ NetWin
   - ☐ Smart2M
   - ☐ Nenhum
   - ☐ Outra: _____

Estado atual de implementação de components

4. **Como classifica o seu processo atual de implementação de componentes?** *
   Marcar apenas uma oval.

	1	2	3	4	5	6	
Insatisfatório	◯	◯	◯	◯	◯	◯	Satisfatório

5. **Porquê?** *

_____

_____

_____

_____

_____

6. **Como classifica o workflow do mesmo?** *
   Marcar apenas uma oval.

	1	2	3	4	5	6	
Complexo	◯	◯	◯	◯	◯	◯	Simples

Technology Acceptance Model

7. **Usar Web Components no meu trabalho permitiria-me completar tarefas mais rapidamente.** *
   Marcar apenas uma oval.

	1	2	3	4	5	6	7	
Provável	◯	◯	◯	◯	◯	◯	◯	Improvável

8. **Usar Web Components melhoraria o meu desempenho no trabalho.** *
   Marcar apenas uma oval.

	1	2	3	4	5	6	7	
Provável	◯	◯	◯	◯	◯	◯	◯	Improvável

9. **Usar Web Components no meu trabalho aumentaria a minha produtividade.** *
   Marcar apenas uma oval.

	1	2	3	4	5	6	7	
Provável	◯	◯	◯	◯	◯	◯	◯	Improvável

10. **Usar Web Components melhoraria a minha eficácia no trabalho.** *
Marcar apenas uma oval.

	1	2	3	4	5	6	7	
Provável	◯	◯	◯	◯	◯	◯	◯	Improvável

11. **Usar Web Components tornaria o meu trabalho mais fácil.** *
Marcar apenas uma oval.

	1	2	3	4	5	6	7	
Provável	◯	◯	◯	◯	◯	◯	◯	Improvável

12. **Acharia os Web Components úteis no meu trabalho.** *
Marcar apenas uma oval.

	1	2	3	4	5	6	7	
Provável	◯	◯	◯	◯	◯	◯	◯	Improvável

13. **Aprender a trabalhar com Web Components seria fácil para mim.** *
Marcar apenas uma oval.

	1	2	3	4	5	6	7	
Provável	◯	◯	◯	◯	◯	◯	◯	Improvável

14. **Acharia fácil conseguir que os Web Components fizessem o que eu quero que eles façam.** *
Marcar apenas uma oval.

	1	2	3	4	5	6	7	
Provável	◯	◯	◯	◯	◯	◯	◯	Improvável

15. **A minha interação com os Web Components seria clara e compreensível.** *
Marcar apenas uma oval.

	1	2	3	4	5	6	7	
Provável	◯	◯	◯	◯	◯	◯	◯	Improvável

16. **Acharia os Web Components flexíveis de trabalhar.** *
Marcar apenas uma oval.

	1	2	3	4	5	6	7	
Provável	◯	◯	◯	◯	◯	◯	◯	Improvável

17. **Seria fácil para mim tornar-me hábil a usar Web Components.** *
Marcar apenas uma oval.

	1	2	3	4	5	6	7	
Provável	◯	◯	◯	◯	◯	◯	◯	Improvável

18. **Acharia os Web Components fáceis de usar.** *
Marcar apenas uma oval.

	1	2	3	4	5	6	7	
Provável	◯	◯	◯	◯	◯	◯	◯	Improvável

Web Components - solução de compatibilidade na implementação de componentes Cross Framework

19. **Como classifica o workflow na implementação de Web Components?** *
Marcar apenas uma oval.

	1	2	3	4	5	6	
Complexo	◯	◯	◯	◯	◯	◯	Simples

20. **Que diferenças existem entre a implementação de um componente baseado na tecnologia da sua framework de trabalho e um Web Component?** *

_____

_____

_____

_____

_____

21. **Do seu ponto de vista, qual/quais a(s) maior(es) dificuldade(s) de implementação de um Web Component?** *
Marcar tudo o que for aplicável.

- [ ] Adaptação à tecnologia
- [ ] Enviar e receber dados do componente
- [ ] Suporte dos browsers
- [ ] Falta de solidificação dos standards e contínua alteração dos mesmos
- [ ] Falta de opções de personalização de estilos
- [ ] Outra: _____

22. **Do seu ponto de vista, quais os maiores benefícios do uso de Web Components?** *
Marcar tudo o que for aplicável.

- [ ] Encapsulamento de CSS
- [ ] Encapsulamento de JavaScript
- [ ] Interoperabilidade entre frameworks
- [ ] Normalização transversal do processo de implementação de componentes
- [ ] Outra: _____

23. **Do seu ponto de vista, como classifica a viabilidade dos Web Components como solução de compatibilidade na implementação de componentes na framework que utiliza?** *
Marcar apenas uma oval.

	1	2	3	4	5	6	
Nada viável	◯	◯	◯	◯	◯	◯	Muito viável

24. **Porquê?** *

_____

_____

_____

_____

_____

Fim

Obrigado pelo preenchimento deste questionário.

Apêndice C – Consentimento informado da sessão de recolha de dados

Projeto de Dissertação, Universidade de Aveiro

Dados recolhidos no âmbito do estudo dos Web Components como solução de compatibilidade na implementação de componentes *cross framework*

Como aluno de mestrado em Comunicação Multimédia e no âmbito do meu projeto de dissertação "Web Components - solução de compatibilidade na implementação de componentes *cross framework*" convidamo-lo a participar na sessão de recolha de dados relativos ao tema em questão. Este projeto tem como objetivo testar e avaliar potencialidades de uma solução baseada em Web Components que sirva as necessidades da Altice Labs, que passam por reutilizar componentes visuais ao longo dos vários produtos da mesma.
Com o seu contributo iremos testar o protótipo e avaliar a viabilidade da solução apresentada.

Para compreender dados de cariz quantitativo e qualitativo iremos gravar áudio e vídeo da atividade e no final será distribuído um breve inquérito. Desta forma esperamos comparar as expectativas que cada participante tem para com a interface prototipada e as suas funcionalidades.

Lembramos que este é um teste à tecnologia e não ao seu utilizador e que todas as informações de caracter pessoal são estritamente anónimas, o direito à privacidade é totalmente garantido. A sua opinião e contributo empírico é extremamente relevante para obtermos conclusões que melhor representam a realidade.

Obrigado pela sua valiosa participação.

==========

CONSENTIMENTO INFORMADO

De acordo com as recomendações da Declaração de Helsínquia, compreendi a explicação que me foi dada sobre a investigação que está a ser realizada e que as informações recolhidas são anónimas.
Eu entendo que os resultados do estudo podem ser publicados em revistas científicas, apresentados em reuniões / eventos técnico-científicos e usados em outras atividades de investigação, sem qualquer violação de confidencialidade/anonimato.
Ao participar nesta atividade, autorizo o uso de dados anónimos para a finalidade da investigação que lhe está associada e mencionada acima.

Nome:
_____

Assinatura:
_____

_____ de _____, 2018

Apêndice D – Instruções para aceder ao protótipo *online*

Instruções para aceder ao protótipo

Para aceder ao protótipo é necessário uma conta registada no GitHub, Bitbucket ou Cloud9.

Versão baseada em tecnologia nativa com Shadow DOM:

1. *Link*: https://ide.c9.io/ruimartins/nativo-shadowdom

Versão baseada em tecnologia nativa com Light DOM:

1. *Link*: https://ide.c9.io/ruimartins/nativo-lightdom

Versão baseada em Polymer com Shadow DOM:

1. *Link*: https://ide.c9.io/ruimartins/polymer-shadowdom

2. Aceder ao *link* da versão pretendida
3. Clicar no ficheiro “index.html” presente na raiz da árvore do projeto (figura 1).

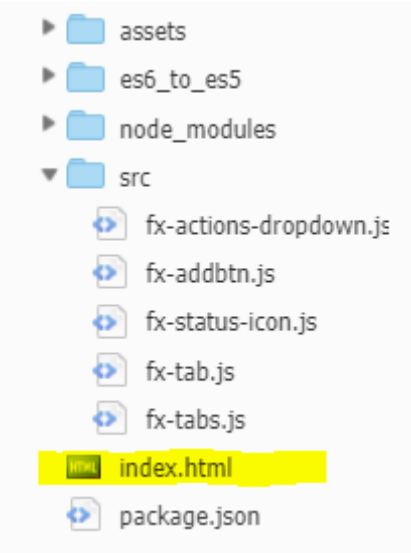


Figura 1 - Árvore do projeto (exemplo)

4. Com o ficheiro “index.html” selecionado, clicar em “Run” na barra de tarefas superior (figura 2) ou na consola situada na parte inferior da janela (figura 3).

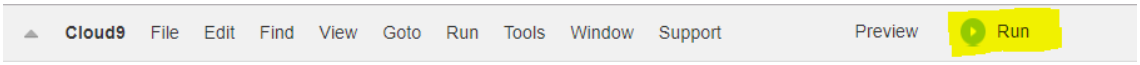


Figura 2 - Barra de tarefas superior

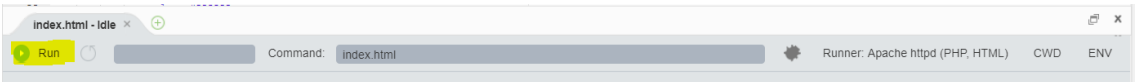


Figura 3 – Consola

5. Clicar no link gerado (figura 4) e dpois em na opção “Open” na janela que irá aparecer (figura 5)

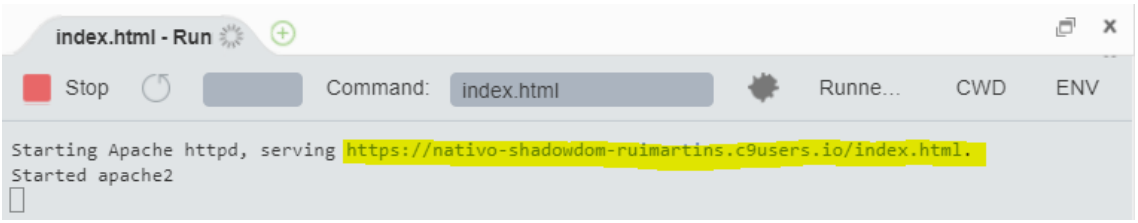


Figura 4 - *Link* gerado

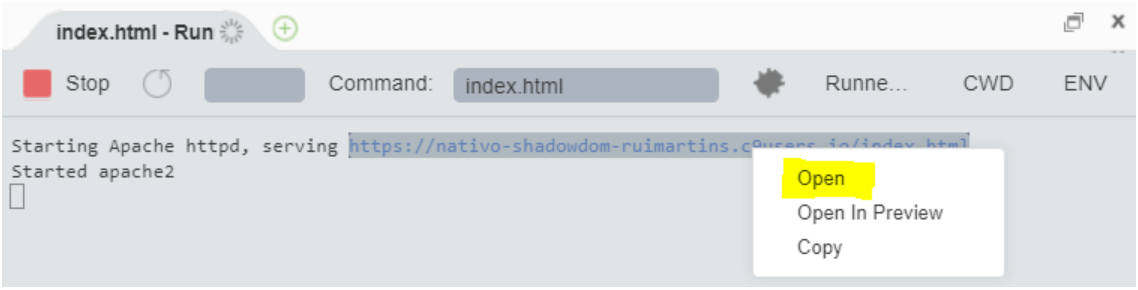


Figura 5 – Janela de opções

6. Navegar pelo componente

Observações: Sendo este link partilhado, qualquer alteração poderá afetar a experiência de utilização futura. Por favor não faça alterações ao código.