INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Monitorização e análise de desempenho em aplicações web e-commerce

DUARTE NUNO FERREIRA AMORIM DO VALE
Outubro de 2019

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Monitorização e análise de desempenho em aplicações *web e-commerce*

Duarte Nuno Ferreira Amorim do Vale

Dissertação para obtenção do Grau de Mestre em Engenharia Informática, Área de Especialização em Engenharia de Software

Orientador: Nuno Silva

Supervisor: Hugo Roda

Porto, Outubro de 2019

Dedicatória

“À minha mãe, por tudo o que sou.
Ao meu avô, por tudo o que me ensinou.
Aos meus amigos e colegas de trabalho, por todos os momentos de alegrias e motivação que me dão.”

Resumo

Nos últimos anos verificou-se um crescimento no mercado *e-commerce*, associado a uma maior disseminação da tecnologia e dos meios de acesso a tal, o que permitiu um aumento do número de compras efetuadas *online*. Considerando este crescimento, e visando a sua evolução, os negócios assentes em *e-commerce* procuram formas de obter mais lucro e de atrair mais consumidores às suas aplicações *web*.

No entanto, visto que população possui cada vez mais acesso a meios tecnológicos muito diversificados, a experiência de utilização e desempenho de uma aplicação *web* pode ser muito diversa – seja pelo dispositivo utilizado, local e condições de acesso a uma rede de *Internet*. Dado que esta temática é, por vezes, menosprezada durante o desenvolvimento, as soluções *web* criadas não vão de encontro às expectativas de desempenho de um *website* que os utilizadores mais jovens possuem nos dias de hoje, bem como não permite um acesso e utilização equitativa a grande parte da população mundial. Ao endereçar esta temática, um negócio *e-commerce* garante que um maior número de utilizadores tenha a possibilidade de aceder a um *website* rápido e eficiente, o que possibilita uma maior taxa de conversão e lucro para o negócio.

De modo a cumprir o objetivo de promover a melhoria do desempenho das aplicações *web* desenvolvidas, irá ser desenvolvida uma ferramenta protótipo que visa analisar e monitorizar os *websites* durante o seu desenvolvimento de modo a prevenir problemas de desempenho.

Para desenvolver este projeto é necessário pesquisar sobre a temática e implementar conceitos inerentes à análise e monitorização de métricas de desempenho no protótipo, aplicando o mesmo em projetos internos, com a finalidade de verificar se o objetivo traçado foi ou não cumprido, pela avaliação da satisfação das partes interessadas com o projeto através da utilização de inquéritos finais.

Palavras-chave: Desempenho, *E-Commerce*, Métricas, Monitorização.

Abstract

Over the last years, the e-commerce market has been growing, due to a higher technology usage, as well as the access to it. This allowed the rise of users making their shopping online. Considering this growth, the e-commerce businesses are looking for ways to increase their profits and attract more customers to their web applications.

However, the types of technology used by the world population is quite miscellaneous and, because of that, the user experience and the performance of a web application can differ – either because of the used device, location or network access conditions. Since the performance topic is often underrated, the web solutions created no longer meet the expectations of younger users of a fast website, neither allows a fair and equal access to people worldwide. By tackling this subject on a business e-commerce, it is possible to ensure access to a fast and efficient website, which translates on a higher conversion rate and business profits.

For accomplish the goal of promote the performance improvement of the developed web applications, it will be created a prototype which aims to analyze and monitoring websites during the development stage.

In order to develop this project, it is required a broad search in the area and the implementation of all concepts with performance analysis, monitoring and metrics for the prototype. For verifying if the defined goal was accomplished, the prototype will be applied on internal projects. It is also important to evaluate the satisfaction rate of the interested parties with the project via inquiries to the interested parties.

Keywords: *E-commerce*, Metrics, Monitoring, Performance.

Agradecimentos

Primeiramente, quero agradecer à *Farfetch* por me dar a oportunidade para a criação e desenvolvimento deste projeto. Pretendo também agradecer às pessoas que fazem parte da unidade de negócio onde estou inserido – *Farfetch Platform Solutions* – por permitir a realização deste projeto em contexto de trabalho, proporcionando a minha evolução académica, técnica e profissional.

Ao Instituto Superior de Engenharia do Porto e, em particular, ao Departamento de Engenharia Informática e a todos os docentes que contribuíram para o meu desenvolvimento e crescimento académico durante a minha formação no âmbito da realização do mestrado.

Ao professor Doutor Nuno Silva, pela orientação e acompanhamento dado no âmbito da realização desta tese, procurando sempre fornecer críticas construtivas visando o meu sucesso na concretização deste projeto.

Ao meu supervisor da organização, Hugo Roda, e ao André Valente que, juntamente comigo, possibilitaram o nascimento deste projeto dentro da organização e pelo apoio manifestado para a realização do mesmo.

À equipa de plataforma de *e-commerce* de *Farfetch Platform Solutions* – *Quiquipa* – pelo apoio, preocupação, disponibilidade e cuidado manifestado para comigo relacionado com o desenvolvimento deste projeto. Pretendo deixar também um agradecimento especial à minha amiga e colega de trabalho Catarina Couto, por toda a motivação, apoio incansável e conselhos partilhados, que não só foram vitais para a realização deste projeto, como permitiram o meu crescimento profissional e académico, desde do momento da minha entrada na *Farfetch*.

Deixo também um especial agradecimento à minha família, e em particular, à minha mãe Margarida Amorim, pelo seu apoio incondicional e por me sempre incentivar a ir mais além.

A todos eles, e às pessoas que fazem parte da minha vida, o meu mais sincero obrigado.

Índice

Lista de Figuras

Lista de Tabelas

Acrónimos

AHP	Processo de análise hierárquica, do inglês *analytic hierarchy process*
API	Interface de programação de aplicações, do inglês *Application Programming Interface*
B2C	*Business to Consumer*
BSD	*Berkeley Software Distribution*
CDN	*Content Delivery Network*
CI/CD	*Continuous Integration/Continuous Delivery*
CLI	Linha de comandos, do inglês *Command Line Interface*
CPU	Unidade central de processamento, do inglês *central processing unit*
CSS	*Cascading Style Sheets*
DOM	*Document Object Model*, interface de programação para documentos HTML
FPS	*Farfetch Platform Solutions*, unidade de negócio da *Farfetch*
GPRS	*General Packet Radio Services*
HTML	*Hypertext Markup Language*
HTTP	*Hypertext Transfer Protocol*
HTTPS	*Hypertext Transfer Protocol Secure*
ISC	*Internet Systems Consortium*
ISEP	Instituto Superior de Engenharia do Porto
JSON	*JavaScript Object Notation*

KPI	*Key Performance Indicator*, valores mensuráveis que permitem demonstrar a eficiência para atingir objetivos específicos de negócio
LTE	*Long-Term Evolution,* um padrão para comunicações móveis em tecnologia 4G
LTS	*Long Term Support*
MB	*Megabytes*
MIT	*Massachusetts Institute of Technology*
NCD	Novo processo de desenvolvimento*,* do inglês *New Concept Development*
npm	*Node.js Package Manager*
PDP	Página de detalhe de um produto, do inglês *Product Detail Page*
PLP	Página de listagem de produtos, do inglês *Product Listing Page*
PWA	*Progressive Web App*, um *website* que se comporta como uma aplicação móvel e é executada num *browser*
QA	*Quality Assurance*
REST	Transferência representacional de estado, do inglês, *Representational State Transfer*
RUM	*Real User Monitoring*
SEO	*Search Engine Optimization*
SSH	*Secure Shell*
TCP	*Transmission Control Protocol*, padrão que define como estabelecer uma comunicação em rede em que as aplicações possam partilhar informação
UML	*Unified Modeling Language*
URL	*Uniform Resource Locator*, a representação de um endereço *web*
UX	Experiência do Utilizador, do inglês, *user experience*
VM	Máquina virtual, do inglês, *Virtual Machine*

1 Introdução

Este capítulo visa apresentar o contexto, problema e objetivos associados à elaboração deste projeto, assim como representar o planeamento efetuado e tarefas realizadas.

Contém também uma breve enumeração de metodologias, ferramentas e tecnologias que irão auxiliar o desenvolvimento do projeto, bem como descreve a organização do documento de forma a facilitar a compreensão da estrutura e conteúdos.

1.1 Contextualização

O presente subcapítulo pretende introduzir aspetos importantes que estão inerentes ao desenvolvimento do projeto, como é o caso do conceito de desempenho, como este se associa à *web* e a sua importância para a organização. Não obstante, é descrito de forma sumária o modelo de negócio da empresa proponente – a *Farfetch*, em particular a unidade de negócio onde o desenvolvimento deste projeto está inserido – *Farfetch Platform Solutions*.

1.1.1 Empresa proponente

A *Farfetch* é uma plataforma de comércio *online,* fundada em 2008 pelo empreendedor de nacionalidade portuguesa José Neves, que disponibiliza acesso a um variado leque de artigos de moda de luxo (Farfetch UK Limited, 2019) e cuja missão "é ser a plataforma tecnológica mundial destinada à moda de luxo, permitindo unir criadores, consumidores" e outras pessoas ligadas ao mundo da moda (Farfetch UK Limited, 2018ª).

O seu *marketplace* – *website e-commerce* disponibilizado e traduzido em 15 idiomas – inclui mais de 3200 marcas distintas (à data de Setembro de 2018), desde marcas de renome até *designers* que ingressaram no mundo da moda recentemente. Os clientes do *website* da

Farfetch podem adquirir artigos de várias categorias desde roupa feminina, masculina e de criança até relojoaria e joalharia, sendo possível que estes produtos sejam enviados para mais de 190 países (Farfetch UK Limited, 2018ª).

"A *Farfetch* é responsável pela construção de uma plataforma tecnológica modular, construída para suportar a ligação ao ecossistema do mundo da moda de luxo, visando construir um sistema integrado que permite endereçar os requisitos complexos para o perfil de consumidor de luxo". Esta plataforma está definida com recurso a tecnologia desenvolvida pela própria, que fornece as fundações para três componentes fundamentais: as aplicações, os serviços e os dados (Farfetch UK Limited, 2018ª).

A 21 de setembro de 2018, a empresa *Farfetch Limited* fez a sua oferta pública inicial (IPO) na bolsa de valores de Nova Iorque (NYSE), estando cotada em mais de 8.2 mil milhões de dólares americanos (Barinka, 2018).

1.1.2 **Unidade de Negócio**

Farfetch Platform Solutions (FPS), anteriormente designada por *Black & White Solutions*, é uma unidade de negócio independente, integrada no grupo *Farfetch Limited*, responsável pela construção e fornecimento de soluções *white-label* – soluções genéricas que podem ser facilmente adaptadas e formatadas como produtos finais para terceiros – destinadas às marcas e retalhistas de moda de luxo que podem beneficiar de todas as vantagens de estarem integradas na plataforma *Farfetch*, nomeadamente o acesso direto a novas funcionalidades e melhorias constantes nos serviços (Farfetch UK Limited, 2018b), baseadas na composição e orquestração dos mesmos serviços que permite operar o *marketplace* da *Farfetch*. Ao terem esta integração, as marcas e retalhistas, passam a usufruir de vantagens como (Kansara, 2015):

- o acesso a mercados globais como a China, Japão e Rússia (considerando as particularidades associadas à operacionalização de cada um destes países);
- ter disponível vários métodos de pagamento verificados, com suporte a pagamentos em tecnologias locais (como é o caso de pagamentos na China, Alemanha e Holanda); suporte em vários idiomas;
- serviço de apoio ao cliente;
- produção e edição do catálogo digital de produtos e fotografias.

As soluções fornecidas estão distribuídas pelas áreas de design, desenvolvimento e manutenção do website, apoio ao cliente e marketing digital (Kansara, 2015).

Ao realizar uma parceria com FPS, uma marca ou retalhista tem acesso a uma infraestrutura de suporte e apoio no processo de criação e operação de um negócio *e-commerce*, dispondo de

todo o conhecimento desenvolvido dentro da plataforma da *Farfetch*, onde lhes é possível (Farfetch UK Limited, 2018b):

- Efetuar o *design* e construir o seu *website* (com FPS ou de forma individual), ter acesso a funcionalidades *omnichannel*[1] e capacidades internacionais, potenciando o envio de artigos para mais de 190 países em 27 idiomas, incluindo para a China;

- Integração com o *marketplace* global da *Farfetch*;

- Cumprir requisitos de acessibilidade no *website* para pessoas com incapacidades – *ADA Compliance*[2], necessários para poderem operar no mercado dos Estados Unidos da América (Interactive Accessibility, Inc, 2019);

- Dispor de uma aplicação para dispositivos *iOS*, integrada com o seu *website* e com suporte para uma operação a nível internacional;

- Operacionalizar uma conta *WeChat*[3], fomentando a presença no mercado Chinês.

O seguinte diagrama ilustra como as soluções *web* para um cliente – um *website* ou uma aplicação nativa com acesso a dados na *web* – podem ser realizadas em conjunto com FPS, bem como a proveniência de lógica de negócio *e-commerce*:

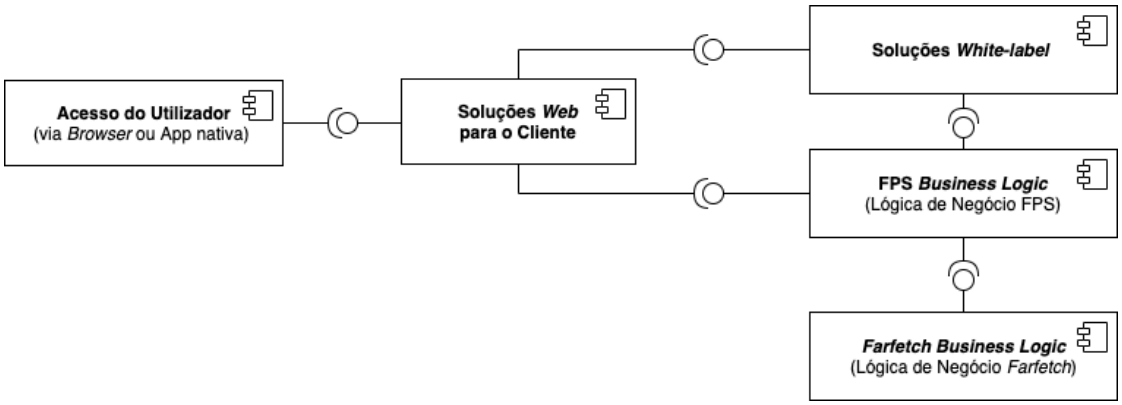


Figura 1. Diagrama de componentes de negócio

Um utilizador final, ao aceder às soluções *web* desenvolvidas pela unidade de negócio *Farfetch Platform Solutions* para um cliente, tem acesso a todo um leque de funcionalidades. Estas

[1] Abordagem multicanal para vendas que visa oferecer a melhor experiência de compras possível, seja *online* (via computador ou dispositivo móvel), por telefone ou por uma loja física, promovendo a integração dos canais de distribuição, promoção e de comunicação.

[2] *Americans with Disabilites Act*, padrões para um *design* acessível de modo a que seja possível que pessoas com incapacidades possam utilizar um serviço eletrónico ou digital.

[3] Aplicação móvel destinada ao mercado chinês com múltiplos propósitos - comunicações voz e escrita, rede social, serviço de pagamentos móvel e integração de subaplicações de terceiros (como por exemplo, de *e-commerce*) dentro do ecossistema da aplicação principal.

podem ser mais genéricas, mas customizadas de acordo com as suas preferências – as soluções *white-label* – assentes na lógica de negócio (*FPS Business Logic*), ou funcionalidades específicas do cliente que consomem recursos diretamente provenientes da mesma lógica. Esta lógica de negócio disponibiliza as funcionalidades específicas de um fornecedor de serviços *e-commerce* para terceiros, apoiando-se para tal na lógica de negócio da *Farfetch* (*Farfetch Business Logic*), em que esta é constituída por um conjunto de serviços interligados que constituem funcionalidades relevantes para a manipulação e interpretação dos dados.

1.1.3 **Desempenho de aplicações *web***

O desempenho, ou performance, define-se como o "quão bem uma pessoa, máquina ou outro objeto realiza um determinado trabalho ou atividade" (Cambridge University Press, 2019).

Atualmente, uma das principais razões para um *website* ser lento é a não consideração do desempenho durante o seu desenvolvimento (Kadlec, 2015). No entanto, a *Internet* continua demasiado lenta (Kadlec, 2015), verificando-se esta afirmação pela média da métrica *SpeedIndex* (cf. secção 2.4) para os mil *websites* mais visitados a nível mundial, testados em dispositivos móveis. Para um *website* considerado rápido, esta métrica linear deve assumir o valor aproximado de 1000 milissegundos, pois quando mais baixo este valor for melhor. No entanto, considerando a média desta métrica para os mil *websites* mais visitados a nível mundial, esta métrica assume o valor de 8220 milissegundos, justificando assim o facto de grande parte dos *websites* não apresentarem um desempenho adequado (Kadlec, 2014ª).

Dada a evolução desta temática e considerando o aumento dos mecanismos disponibilizados, novas ferramentas (disponíveis nas novas versões dos *browsers* mais utilizados), novos padrões (*standards*) e documentação, verifica-se que existe diversas técnicas para colmatar a falta de desempenho nos *websites*. No entanto, as soluções *web* existentes ainda não usufruem destas técnicas, nem mostram as melhorias de desempenho expectáveis.

Este facto pode-se justificar por não existir uma cultura de desempenho organizacional e das equipas de desenvolvimento, que não consideram cuidados desta temática durante o desenvolvimento de um *website*, bem como tendem a optar por outras opções que estão mais em voga (Dalal, 2015), em vez de endereçar a raiz do problema.

O simples facto de não encarar os problemas de desempenho de um *website* e optar por soluções parciais é grave pois limita o acesso de utilizadores a conteúdos e serviços, onde este é obrigado a utilizar um dispositivo específico ou a descarregar uma determinada aplicação (Kadlec, 2015).

Deste modo, é importante atacar estes problemas, com o objetivo de tornar os conteúdos *web* mais acessíveis a nível mundial, conseguindo tornar o conteúdo digital mais abrangente e

tangível a um maior número de utilizadores, tal como a definição da *Internet* (WWW[4]) contempla.

Com vista a responder a estas necessidades, é necessário mudar o *mindset*[5] quando se disponibiliza algum conteúdo na *Internet* para o público, em que o desempenho do mesmo deve ser aplicado logo ao início. Para isso, é necessário aplicar diversas práticas que permitam atingir o objetivo de produzir e disponibilizar conteúdos *web* mais rápidos e que promovam uma experiência de utilização mais agradável junto dos utilizadores finais, evitando a degradação dos mesmos em termos de desempenho no futuro.

Não só é importante aplicar práticas que visam melhorar o desempenho das aplicações *web*, como também é importante realizar um controlo das mesmas, para que seja possível verificar se o conteúdo que está a ser disponibilizado está a ser consumido da melhor forma pelos utilizadores finais, ou se existem incidências que possam afetar a experiência de utilização. Para isso, deve-se proceder à monitorização do desempenho das aplicações *web*, de modo a recolher dados sobre a sua utilização, que permitam conhecer como estas se estão a comportar junto dos utilizadores finais.

Graças aos processos de monitorização e análise, é possível utilizar os dados recolhidos para observar o impacto de um desempenho adequado no negócio, com base na correlação de métricas recolhidas associadas a indicadores de negócio. Por exemplo, com estes dados torna-se possível compreender se os tempos mais reduzidos incentivam o utilizador à compra de mais artigos ou se o facto de os utilizadores experimentarem tarefas mais demoradas num processo de compra aumenta a taxa de desistência de compras. Ao efetuar estas correlações, conseguimos compreender a importância do desempenho junto das aplicações e a necessidade desta disciplina ter destaque durante o desenvolvimento das mesmas (Walton, 2019).

1.1.4 **Causas para a falta de desempenho**

Existem diversos aspetos e motivos que os quais são responsáveis por causar problemas relacionados com a falta de desempenho das soluções *web* desenvolvidas atualmente, podendo ser estas mais do foco da equipa de desenvolvimento, ou da infraestrutura utilizada para disponibilizar o acesso a um determinado *website*.

No entanto, é possível centrar os principais problemas para a falta de desempenho de soluções *web* nos seguintes pontos:

- Os conteúdos devolvidos por um *website* (como imagens e *scripts*) não estarem devidamente otimizados para a *web*. A sua otimização através de utilização de

[4] *World Wide Web*. O espaço onde se encontram documentos e outros recursos *web* acessíveis via *Internet*, o que permite o acesso democratizado à informação por milhares de pessoas a nível mundial (The Editors of Encyclopaedia Britannica, 2019)

[5] Conjunto de atitudes ou práticas tomadas por alguém.

mecanismos de compressão para as imagens e de eliminação de código não-utilizado para os *scripts* permite reduzir o seu tamanho assim como reduzir os tempos de resposta quando estes são solicitados durante o carregamento de um *website* (cf. secção 2.2) (Arsenault, 2017);

- Não utilizar uma CDN, que permite obter tempos de carregamento mais reduzidos assim como permite aumentar o desempenho de um *website* (Arsenault, 2017);

- Os *websites* não estarem adaptadas para serem utilizados por dispositivos móveis, quer em termos de tempos de resposta como de experiência de utilização (UX). Um *website* devidamente preparado para ser utilizado por dispositivos móveis encontra-se desde já com cuidados especiais para ter um desempenho adequado em qualquer dispositivo e condição de ligação à *Internet*, considerando as capacidades mais reduzidas ou possíveis condições de acesso à Internet observada neste tipo de dispositivos quando comparados com as capacidades de processamento e condições de ligação típicas de um computador de secretária/portátil (Arsenault, 2017);

- Não tirar proveito de protocolos e de funcionalidades mais atuais da *web*, como é o caso de HTTPS nos *browsers* mais recentes. A utilização deste protocolo, para além de permitir que a comunicação segura através da encriptação, possibilita a utilização do protocolo HTTP/2, responsável por introduzir o processo de *multiplexing*, que possibilita o assincronismo de pedidos numa só ligação TCP, permitindo reduzir o tempo de carregamento de um *website*, bem como contribuir para um melhor posicionamento deste nos algoritmos de ordenação dos motores de pesquisa (Hunt, 2016).

Ao endereçar estes problemas com algumas das práticas sugeridas pelos pontos anteriores, é possível garantir que as soluções *web* desenvolvidas não denotem graves problemas de desempenho para o público em geral, nomeadamente o seu uso por parte de utilizadores com dispositivos móveis, público-alvo este que pode ter piores experiências de utilização e de tempos de resposta em *websites* com estes tipos de sintomas.

1.2 Problema

Na unidade de negócio FPS, assiste-se a um aumento considerável do número de projetos de maior dimensão e dos agentes intervenientes neste processo – equipas de desenvolvimento e manutenção, gestores de projeto – pelo que nasce a necessidade de desenvolver ferramentas especializadas que permitam garantir a fácil manutenção da plataforma e a monitorização da mesma, de forma a poder prevenir os problemas adjacentes a um ambiente de desenvolvimento distribuído.

Para conseguir realizar a monitorização dos sistemas constituintes da plataforma de e-*commerce* de FPS de uma forma adequada, torna-se necessário obter informação relativa a métricas de forma a que seja possível:

- medir e avaliar o impacto de alterações realizadas durante todo o processo de desenvolvimento;
- mitigar possíveis problemas de desempenho durante o processo de distribuição e ambiente de produção.

No cenário atual, esta análise e monitorização de desempenho não ocorre da melhor forma devido a:

- não existir nenhum processo interno definido e adotado de uma forma generalizada;
- não existir um processo de análise e interpretação automática dos dados no âmbito de aplicações *web*;
- os dados recolhidos pelas ferramentas existentes e já implementadas na unidade de negócio são utilizados de uma forma reativa e não preventiva.

1.3 Objetivos do projeto

O principal objetivo deste projeto é promover a melhoria do desempenho das aplicações *web* desenvolvidas por FPS, para fornecer soluções mais resilientes e otimizadas ao cliente final, permitindo aumentar a sua satisfação e, consequentemente, assistir a um aumento de indicadores de desempenho do negócio, como é o caso do número de vendas, visitas e interações com os *websites*.

Para determinar como ser atingirá este objetivo, adotou-se uma abordagem analítica de análise de valor de alternativas, como descrito no Anexo B. Desse processo analítico, concluiu-se que a alternativa mais vantajosa passa pelo desenvolvimento de um protótipo que permita fornecer informação sobre o estado de desempenho de aplicações e conteúdos *web* de FPS, de forma a analisar o seu desempenho durante a fase de desenvolvimento.

Deste modo, e para a concretização do objetivo, são definidos os seguintes sub-objetivos:

- Reunir indicadores mais adequados para a análise de desempenho em aplicações web vocacionadas para *e-commerce*;
- Definição de um processo para a medição e monitorização do desempenho de aplicações web e-commerce, com ênfase na fase de desenvolvimento dos projetos;
- Desenvolver o protótipo mencionado;

- Disseminar e dar mais visibilidade à importância do desempenho e das ferramentas atualmente existentes pelas equipas de desenvolvimento de FPS, de modo a que estas possam tirar proveito da informação disponível, bem como estarem sensibilizadas para as boas práticas a adotar de modo a privilegiar esta temática

Portanto, através destas iniciativas e *software* resultante, desenvolver-se-á uma cultura e pensamento orientado ao desempenho dentro da organização e das equipas de desenvolvimento.

1.4 Abordagem

O projeto, conforme descrito nas secções anteriores, denota a existência de três partes interessadas (*stakeholders*):

- As equipas de desenvolvimento de FPS, que pretendem obter indicadores e detalhes sobre o estado das soluções *web* desenvolvidas, de modo a ser possível mitigar problemas de desempenho o mais cedo possível, bem como efetuar análises comparativas entre versões, de forma a poder entregar desenvolvimentos mais resilientes e com valor significativo de desempenho. Com esta informação, a equipa não só contribui como beneficia da criação e evolução de soluções tecnológicas que auferem mais cuidados com o seu desempenho. Além do mais, as equipas pretendem estar conscientes e dotadas do conhecimento necessário para utilizar as ferramentas atuais implementadas que fornecem dados sobre o desempenho das aplicações *web* sobre diversos contextos, de modo a poder aplicar essa informação nos seus projetos;

- A empresa/cliente da plataforma tecnológica desenvolvida pelas equipas, que pretende conhecer o impacto que o fornecimento de um *website* com um desempenho adequado tem junto da sua base de clientes, bem como verificar os benefícios que um *website* deste tipo lhes pode fornecer. Graças a este projeto, a organização, no trabalho com os seus clientes, pode considerar o desempenho do *website* como um requisito essencial/obrigatório às soluções *e-commerce* a oferecer ao utilizador final, o que permite o crescimento do negócio dos seus parceiros, visando o aumento das possíveis receitas de ambos;

- O cliente final, que vai usufruir de uma melhor experiência de utilização e satisfação graças a soluções *web e-commerce*, cujo desenvolvimento dotou de cuidados orientados a um adequado desempenho.

Assim, e de forma a ir de encontro às expectativas dos *stakeholders* identificados, serão realizadas as seguintes tarefas:

- Interpretar o problema da falta de desempenho das aplicações *web* existentes, considerando as necessidades da organização;

- Proceder a uma pesquisa ampla de conceitos inerentes à disciplina de desempenho de aplicações *web*, de forma a conhecer métricas, tecnologias de monitorização e análise de desempenho, bem como padrões e boas práticas a adotar numa implementação que privilegie este tema;

- Explorar as tecnologias de monitorização e análise de desempenho de aplicações *web*, que permitam a obtenção de métricas e acompanhamento da evolução desses respetivos valores;

- Definir um conjunto de métricas a monitorizar para as aplicações *web* desenvolvidas em FPS, com a definição de valores padrão associado ao contexto de cada métrica;

- Implementar uma estratégia de monitorização e análise de desempenho, capaz de ser integrada no processo de desenvolvimento atual das aplicações *web* e de fornecer os respetivos resultados;

- Desenvolvimento de um projeto que possibilite a integração de bibliotecas e recursos destinados à monitorização de *websites* e cujos resultados estejam de acordo com as necessidades da organização;

- Criar documentação sobre o desempenho *web* pela identificação de boas práticas a adotar no desenvolvimento de aplicações, bem como dar visibilidade às ferramentas já existentes que recolhem informação sobre o desempenho dos *websites*, de modo a que as equipas possam tirar proveito das mesmas.

De modo a ilustrar a aplicação do projeto dentro fluxo de trabalho efetuado na unidade de negócio FPS, apresenta-se a seguinte figura, que representa as várias fases do desenvolvimento de um projeto para um cliente:

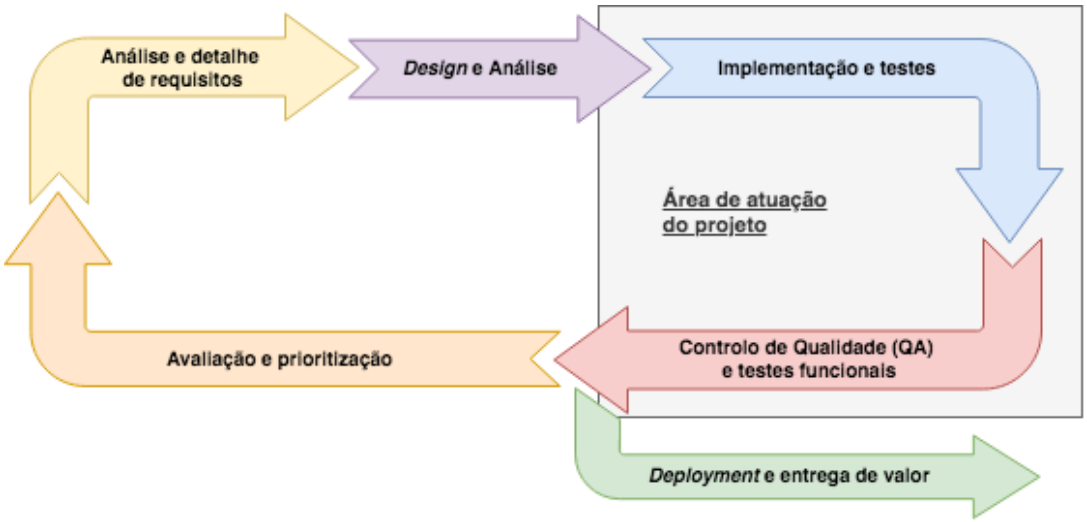


Figura 2. Área de atuação do projeto no ciclo de desenvolvimento de projetos

Com base no que se pode observar na figura anterior, um projeto desenvolvido por FPS contempla a fase de "Análise e detalhe de requisitos", seguida de "*Design* e análise" dos conteúdos idealizados, por parte da equipa de desenvolvimento e por parte do cliente. Segue-se a fase de "Implementação e testes" e consequentemente o "Controlo de Qualidade (QA) e testes funcionais". São estas duas últimas fases descritas que caracterizam a área de atuação do projeto e dedicam-se à criação, desenvolvimento, melhoria e validação de um valor adicional para uma entrega ao cliente, normalmente associado a uma evolução contínua de um produto, podendo representar um ciclo por si só, pois a entrega só deve ocorrer quando o produto se encontra completamente funcional e com um determinado grau de qualidade. Após esta fase, surgem duas fases que decorrem em simultâneo: a implantação e entrega de valor – responsável por disponibilizar os conteúdos desenvolvidos ao cliente (num ambiente de produção/acesso ao público ou privado para efeitos de validação) – e a "avaliação e priorização" – responsável por validar junto do cliente a entrega efetuada e verificar a prioridade de funcionalidades novas ou previamente identificadas. Após a "avaliação e priorização", repete-se novamente o ciclo com a informação obtida na fase anterior.

O projeto pretende atuar nas fases de "Implementação e testes" e "Controlo de Qualidade (QA) e testes funcionais", dado que as fases identificadas:

- Estão diretamente relacionadas com a equipa onde a ideia originou, e o desenvolvimento será feito, o que trará rapidamente mais valor para a unidade de negócio;

- São responsáveis por produzir os resultados desejáveis para um cliente e onde existe a necessidade de garantir que os produtos entregues detêm um desempenho adequado, de acordo com um padrão desejável para aplicações *web e-commerce* destinados ao mercado de moda de luxo.

Com este projeto e os detalhes de desempenho resultantes da análise efetuada por este, será possível às equipas de desenvolvimento terem a informação necessária sobre as partes da solução *web* que necessita de melhorias, de modo a procederem ao processo manual de otimização dos *websites* em desenvolvimento, conforme o seguinte diagrama pretende ilustrar.

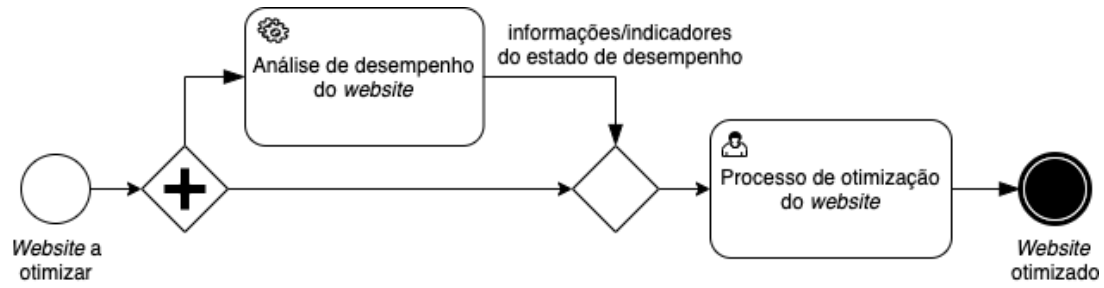


Figura 3. Diagrama do processo de otimização de um *website* em FPS

Desta forma, com base na investigação a realizar em prol de criar estratégias de monitorização e análise de desempenho, será possível não só sensibilizar a equipa de desenvolvimento para a

temática do desempenho, como também elucidar a mesma de algumas temáticas inerentes a esta área que possam ser aplicadas durante novos desenvolvimentos, especialmente em projetos de áreas transversais e partilhadas. Assim, o desempenho passa a ser considerado como um critério de aceitação para o desenvolvimento de qualquer nova funcionalidade.

1.5 Planeamento do projeto

Para o planeamento deste projeto, procedeu-se a diversas reuniões de acompanhamento com os *stakeholders* e o supervisor da organização, ora para delinear os objetivos iniciais ora para proceder à verificação dos desenvolvimentos efetuados e acompanhamento da concretização das tarefas determinadas.

Assim sendo, foi definido um processo de desenvolvimento iterativo e incremental, com base em métodos ágeis, neste caso a *Framework* de trabalho *Scrum*, em que, ao concretizar cada tarefa/fase definida no planeamento, seguia-se um processo de controlo e revisão da mesma, assentes nos quatro pilares fundamentais que a área de gestão e controlo define – Planeamento, Execução, Controlo e Ajustes. Todas as tarefas a implementar foram espelhadas no *software* de apoio à organização e monitorização de tarefas para as equipas – o *Jira*.

Ainda sobre o processo de controlo, o supervisor e os *stakeholders* forneceram o seu *feedback*, promovendo uma melhoria contínua, com base nos pontos assinalados nesse processo, seguido de definição de novas tarefas a realizar, contemplando os seus requisitos e respetiva prioridade.

Deste modo, a concretização deste projeto segregou-se em três fases:

- **1ª fase – Planeamento e pesquisa:** esta fase visou a absorção e exposição do conhecimento inerente e existente ao tema de desempenho de aplicações *web*. Procedeu-se à realização da análise dos pontos positivos e menos positivos do projeto, assim como o contexto onde este está inserido. Em simultâneo, as tecnologias e ferramentas existentes foram alvo de análise de forma a verificar em que medida podem apoiar o desenvolvimento do projeto. Não obstante, as métricas associadas à avaliação de desempenho foram também alvo de estudo, de modo a poder selecionar quais as mais adequadas para serem adotadas para o desenvolvimento do projeto.

- **2ª fase – Análise, implementação, testes e controlo:** nesta fase, procedeu-se a uma análise dos requisitos e prioridades que foram definidas na fase anterior, visando a criação de diagramas UML da arquitetura, componentes e sistema, de modo a documentar e fornecer uma visão do que se pretende com o projeto a implementar.

- **3ª fase – *Outputs* e resultados:** nesta fase procedeu-se à elaboração de documentos de suporte ao projeto, como é o caso de documentação no código-fonte e de suporte, à melhoria de diagramas associados à transmissão da visão e estrutura do projeto e à elaboração de todo o suporte associado à avaliação do projeto. Esta fase decorreu em

simultâneo com a fase anterior, tendo em conta o registo contínuo dos objetivos alcançados a cada tarefa.

1.6 Estrutura do documento

No primeiro capítulo é descrito todo o âmbito onde o presente projeto se encontra enquadrado, nomeadamente a organização e a unidade de negócio para a aplicação do mesmo – *Farfetch* e FPS, respetivamente. É também descrito o problema inicial, objetivos do projeto e a forma como o mesmo foi abordado.

No segundo capítulo procede-se a um enquadramento teórico e tecnológico das áreas adjacentes ao projeto, de forma a permitir uma melhor compreensão dos conceitos inerentes ao mesmo. Procede-se também a uma análise de ferramentas existentes que estão enquadrados com o âmbito deste projeto, focados para análise e monitorização de desempenho.

O terceiro capítulo destina-se à definição do *design* arquitetural do *software* do protótipo a desenvolver no âmbito do projeto, com recurso notação UML para fornecer mais detalhe sobre a organização e funcionamento geral do mesmo.

O quarto capítulo pretende apresentar o protótipo desenvolvido no âmbito do projeto, respetivas funcionalidades e modo de funcionamento, bem como a recolha de métricas efetuada de modo a permitir a análise e monitorização.

No quinto capítulo procede-se à definição do processo e métodos de avaliação a aplicar sobre o projeto a desenvolver futuramente, identificando o público-alvo, bem como as abordagens a adotar para cada nicho.

No sexto capítulo é realizada uma síntese do projeto realizado, refletindo sobre objetivos atingidos, limitações e desenvolvimentos a preconizar futuramente no projeto, além de uma apreciação pessoal relativo ao desenvolvimento do projeto.

No final é possível encontrar dois anexos referentes:

- Uma apresentação de conteúdos disponíveis na organização que foram necessários utilizar para a realização deste projeto;

- A descrição da proposta de valor do presente projeto, através da identificação da oportunidade e a respetiva análise e descrição das ideias geradas (cf. secção B.4) para a realização do projeto. Recorre-se também ao método de avaliação hierárquico para a determinação da ideia mais adequada a implementar, complementado com o modelo CANVAS, de forma a permitir uma melhor compreensão da área de negócio.

2 Estado da Arte

Este capítulo contextualiza o âmbito do desenvolvimento deste projeto de forma a ser enquadrado num contexto mais global, analisando fatores associados ao desempenho de aplicações *web* e a sua importância na atualidade. Em continuidade, são também abordados termos técnicos mais específicos e inerentes à área do desempenho das aplicações *web*.

2.1 Importância do desempenho

Considerando à constante evolução da *web*, existe um denominador comum que afeta este desenvolvimento, de acordo com a Google – o seu desempenho. Muitos *websites* enfrentam dificuldades para conseguirem ter um elevado desempenho no meio de diversas condições de rede e dispositivos existentes (Wagner, 2018).

Os problemas de desempenho podem variar, desde criar pequenos intervalos sem resposta, o que pode causar desconforto ao utilizador final, até deixar um *website* completamente inacessível e/ou sem resposta a interações do utilizador. Numa aplicação *web*, é esperado que a interação com o conteúdo desenvolvido seja significativa para o utilizador, pois o seu desempenho assume um papel importante para o sucesso de qualquer negócio com presença *online* (Wagner, 2018).

Não obstante, a retenção de utilizadores é crucial para o aumento das receitas e KPI de negócio, no caso de se operar um negócio *online* (Wagner, 2018). Ao operar um *website* que apresente melhorias de desempenho, assiste-se a um consequente crescimento de métricas associadas ao desempenho financeiro de um negócio (Enright, 2010).

A experiência do utilizador assume também um papel importante associado ao desempenho. Existem diversas condições que afetam a experiência que um utilizador vivencia durante a visita

a um *website*, como é o caso da sua ligação de rede ou o dispositivo utilizado para visualizar o mesmo (Wagner, 2018).

Quando um *website* disponibiliza um elevado número de conteúdos e/ou recursos, aumenta o tráfego de comunicação de dados efetuado pelos *browsers*. Para o utilizador final, isto é sinónimo de um maior consumo do seu plano de dados para apresentar o conteúdo solicitado. Esta situação é agravada caso se trate de dispositivos móveis (cujo poder computacional e memória são mais reduzidos) que apresentam um desempenho inferior ao lidar com *websites* que não tenha adotado práticas de melhoria de desempenho (Wagner, 2018).

Considerando o problema descrito anteriormente, o fraco desempenho de um *website* conduz à falta de resposta por parte do mesmo e, tendo em conta o comportamento humano, os utilizadores normalmente só toleram aplicações com baixo desempenho num curto espaço temporal, pelo que abandonam a sua utilização caso os sintomas persistam (Wagner, 2018).

O desempenho também é sobre as pessoas, pois as aplicações *web* que apresentam um fraco desempenho acarretam custos reais para os seus utilizadores. Os utilizadores de dispositivos móveis contribuem em grande parte para o número de utilizadores que acedem à *Internet* a nível mundial e, estes acessos ocorrem dentro de redes móveis como LTE, 4G, 3G e até redes 2G[6]. É importante frisar que em países em desenvolvimento o acesso a redes móveis de alta velocidade ainda é reduzido mas, graças à contínua evolução da infraestrutura a nível mundial, o custo de planos de dados móveis tem vindo a reduzir (Schwarz, 2017), permitindo mais acessos à *Internet*. Os dispositivos móveis e o acesso à *Internet* está a deixar de ser um luxo e passou a ser uma ferramenta comum e necessária para gerir o dia-a-dia e viver numa sociedade cada vez mais conectada (Wagner, 2018).

Tendo em conta o aumento substancial do tamanho dos conteúdos web (HTTP Archive, 2019), e com previsões de este número aumentar, os utilizadores necessitam de investir e de aumentar o limite de dados móveis disponíveis (*plafond*), o que acarreta mais custos.

Em suma, é fundamental o *design* arquitetural de soluções que promovam um acesso de forma eficaz e eficiente à informação importante para os utilizadores.

2.2 Desempenho como uma medida para a sustentabilidade

Segundo um estudo por parte da *Mozilla*[7], as diversas centrais de servidores (*data centers*) destinados ao processamento de dados da *Internet* possuem a mesma pegada de CO_2 que todas as viagens aéreas a nível mundial (Mozilla Foundation, 2018).

[6] Abreviaturas de diversas gerações associadas a serviços de rede móvel – 2G para a segunda geração, 3G para terceira geração, 4G para quarta geração. A cada geração, são introduzidas melhorias da velocidade de ligação à *Internet* e a capacidade de ser utilizada por mais dispositivos.

[7] Fundação sem fins lucrativos para apoiar o crescimento o projeto do *software* livre com o mesmo nome, responsável pelo desenvolvimento do *browser Mozilla Firefox*.

Contudo, as empresas de maior dimensão da área da *Internet* estão conscientes deste impacto ambiental e, por esse motivo, têm optado por selecionar fontes de energia renováveis, seja para as suas operações ou para o fornecimento de energia elétrica para os seus equipamentos informáticos.

Considerando a evolução exponencial da *Internet* e a sua disponibilização em novos territórios, a sustentabilidade deve ser um tópico que deve assumir uma maior prioridade para a prevenção de mudanças climáticas e, deste modo, a área da Informática pode contribuir no âmbito da sua atividade através de:

- *Design* de *websites* mais sustentáveis;
- Da adoção de alojamento *web* que estejam orientados para práticas mais sustentáveis, como é o caso do fornecimento de energia elétrica para os seus servidores e *data centres* provenientes de fontes renováveis;
- Do impulsionamento e motivação do sector tecnológico para a aposta em inovação nas energias limpas, à semelhança do que ocorre noutras indústrias, como é o caso da indústria automóvel;
- Da introdução de políticas associadas ao acesso à *Internet*, principalmente nos locais onde a eletricidade seja mais escassa, através do estabelecimento de legislação que dê prioridade à utilização de energias renováveis e alternativas.

Refletindo nas possíveis mudanças na área da Informática que foram descritas anteriormente, existe uma que pode ser aplicada por qualquer entidade individual ou coletiva que se dedique ao desenvolvimento de aplicações *web*, que é o *design* de *websites* mais sustentáveis. A utilização deste tipo de *design* combina a sustentabilidade a outros dois fatores relevantes: a experiência de utilizador (UX) e o seu desempenho (Lenox, 2019).

Ao optar pela adoção de uma filosofia de design sustentável para um *website*, opta-se em simultâneo por aplicar práticas que visam otimizar o desempenho de um *website*. Estas otimizações podem ser de um âmbito mais tecnológico ou de produto.

Deste modo, é possível combinar o interesse do *design* e da conceção de aplicações *web* mais sustentáveis, que adotam práticas relacionadas diretamente com o objetivo das práticas que visam melhorar o desempenho, sendo elas (Lenox, 2019):

- A redução de pedidos de rede, algo que é concretizável ao:
    - Evitar a utilização de bibliotecas externas completas em prol de pequenas adaptações a uma escala reduzida (privilegiar a utilização de funções nativas de *JavaScript* em vez de recorrer a bibliotecas como *lodash* ou *jQuery*, por exemplo);

- Aplicar práticas de eliminação de código não-utilizado em momento de compilação, um processo denominado de *tree shaking* (Bachuk, 2017);

- Utilizar alternativas aos *carousels*[8] para apresentar várias imagens. Algumas pesquisas mostram que os utilizadores raramente interagem ou não apreciam a interação com elementos deste tipo (Creative Bloq Staff, 2013);

- Utilizar formatos de imagem apropriados para a *web*, como é o caso do *WebP* (Google, 2019ª);

- Optar pela utilização de tipografia predefinidas nos dispositivos, em alternativa às tipografias disponibilizadas na *web* (*web fonts*), de modo a evitar pedidos adicionais para o seu carregamento e utilização. Se tal não for possível, estas devem ser de tamanho reduzido e deverão ser carregadas o mais cedo possível no código-fonte (Bevacqua, 2015ª);

- Disponibilizar uma hiperligação para um vídeo em vez de ser incluídos numa página. Por norma, os utilizadores de uma página não chegam a visualizar o vídeo e este acrescenta cerca de 1 MB ao tamanho da página a carregar (Lenox, 2019).

- Examinar cuidadosamente todos os aspetos da aplicação, de forma a avaliar se os elementos presentes nas várias páginas se traduzem em valor para o utilizador final, assim como avaliar se os dados recolhidos dos utilizadores através de plataformas como o *Google Analytics*[9] são relevantes e se se encontram a ser utilizados, pois estes adicionam um peso extra às páginas, além de levantarem questões éticas de como estes dados podem estar a ser tratados (Lenox, 2019);

- Utilizar técnicas de comuns de melhoria de desempenho de um website:

    - Aplicar de técnicas de redução e otimização do código HTML, CSS e *JavaScript* para produção (Google, 2018ª);

    - Nos estilos CSS, tirar mais partido das funcionalidades e sintaxe da linguagem (Pickering, 2016), recorrendo a técnicas de redução de código na compilação para execução em ambiente de produção (Dayan, 2018);

[8] Carrossel. Elemento de *web design* que permite a visualização de diversas imagens num único local, com a possibilidade de trocar a imagem a ser visualizada.

[9] Serviço fornecido pela Google que visa recolher estatísticas relativas à utilização de um *website* e KPI relevantes relacionados com a operação *online*.

    - Comprimir todas as imagens, utilizando os formatos e as definições mais adequadas (Gosha e Farley, 2017) e *Progressive Rendering*[10] (Atwood, 2005).

- Melhorar o desempenho no lado do servidor, através da ativação da compressão *gzip*[11], da utilização de HTTPS (Hunt, 2016), da utilização de servidores *web* como *Nginx* e aplicação de mecanismos de *caching*[12] de conteúdos (Lenox, 2019).

De forma a verificar os resultados da aplicação das práticas de um *design* de aplicações sustentáveis, surge uma métrica a monitorizar para este efeito – *energy usage* (utilização de energia) – que visa medir a sustentabilidade de um produto digital, a qual considera os gastos energéticos pelo trabalho realizado pelo servidor, pelo cliente e por todas as comunicações de dados intermédios. No entanto, obter este valor pode ser algo complexo de ser medido individualmente (i.e. por pedido). Para tal, e de modo a obter um valor discreto relativo à utilização da energia por parte de um produto digital, recorre-se a outros valores resultantes da monitorização de uma aplicação, como:

- a utilização de recursos (*resource usage*);
- utilização do processamento (*CPU usage*);
- utilização de memória (*memory usage*);
- transferência de dados (*data transfer*).

Os valores mencionados anteriormente podem ser interpretados por ferramentas disponíveis, como *WebsiteCarbon* ou o *Ecograder*, capazes de traduzir estas métricas numa pontuação (Lenox, 2019).

Em suma, ao aplicar as ações enumeradas anteriormente no desenvolvimento de uma aplicação *web*, apesar de estar a melhorar diretamente uma ou mais dimensões – o desempenho de um *website* ou do servidor, a experiência de utilização final – é possível contribuir para o desenvolvimento de aplicações mais sustentáveis, bem como contribuir para uma maior visibilidade destas junto dos motores de pesquisa.

[10] Modo de *renderização* onde um programa atualiza gradualmente pequenas partes de uma imagem, refinando-a de uma qualidade inferior até obter o resultado final de melhor qualidade, ao invés de focar no carregamento de uma pequena parte ao longo do tempo.

[11] Método utilizado para reduzir o tamanho dos conteúdos de uma página antes de serem transmitidos para um *browser*, de forma a aumentar a velocidade de um *website* e reduzir a largura de banda ocupada.

[12] Processo de armazenamento de dados temporários num local destinado a esse efeito – *cache*.

2.3 *Performance Budget*

Segundo Jason Grigsby, a *Internet* nos dias de hoje está feita "à nossa imagem... obesa" (Grigsby, 2011), em que se verifica o aumento exponencial quer do tamanho médio das páginas como do número de ligações às mesmas (Kadlec, 2013).

Por este motivo, a preocupação com o desempenho das aplicações deve estar presente na documentação do projeto desde a fase mais embrionária, de forma a poder enfatizar a sua importância.

Assim, nasce o conceito de *performance budget*, ou seja, um orçamento associado ao desempenho de uma aplicação. Este orçamento permite definir limites padrão, associados a diversas características, que não devem ser ultrapassados numa página *web* (Osmani, 2018). Tal como um orçamento financeiro, não devem ser encarados apenas como um valor limite, mas sim ser utilizados com consciência. Relativamente às características que este tipo de orçamento pode controlar são:

- O tamanho máximo do *bundle*[13] de *JavaScript*;
- O tamanho total que uma imagem pode ter;
- Um tempo de carregamento de uma página específico, associado a um determinado tipo de ligação de rede (2G, 3G ou 4G) ou categoria de dispositivo (*desktop* ou *mobile*);
- Um *threshold*[14] aplicado a qualquer tipo de métrica que faça sentido ser controlada.

Uma aplicação deste conceito foi feita pela BBC[15], em que estabeleceu que o carregamento de uma página do seu *website* fosse utilizável em dez segundos ao utilizar uma ligação GPRS[16] (Maslen, 2012).

Com este conceito, o desenvolvimento de novas funcionalidades ou conteúdos para uma aplicação exige que esteja dentro do "orçamento". No caso de estar fora, é possível recorrer a várias hipóteses (Osmani, 2018):

- Melhorar funcionalidades previamente existentes;
- Remover funcionalidades já existentes;

[13] Técnica que permite agrupar ficheiros distintos, de modo a reduzir os pedidos HTTP necessários ao carregamento de uma página *web*

[14] Valor limite.

[15] *British Broadcasting Corporation*, emissora pública televisiva e radiofónica do Reino Unido.

[16] Tecnologia rádio utilizada para a transmissão de pequenos pacotes de dados, utilizada especialmente entre telemóveis e a *Internet*.

- Optar por não adicionar a nova funcionalidade;

- Optar por adicionar outra funcionalidade que fique abaixo do "orçamento", ficando a funcionalidade inicial fora do caminho crítico da aplicação (Osmani, 2018), podendo esta ser carregada *à posteriori*, assim que o utilizador necessite, através da utilização de *lazy loading*[17] (Hlushko et al., 2018).

O conjunto de decisões mencionadas anteriormente devem ser tomadas perante cada página *web* integrante da aplicação em desenvolvimento.

Um orçamento de desempenho a nível organizacional irá permitir que este seja detido por todos os intervenientes da mesma, ao invés de ser simplesmente definido por um grupo, deixando de ser uma preocupação exclusiva das equipas de engenharia (Osmani, 2018), o que irá permitir a melhoria do processo de tomada de decisão, bem como gerar o respetivo conhecimento para permitir o crescimento sustentável de um produto.

Tipicamente, os *stakeholders* fora da área da engenharia (como *Project Managers* e *Designers*) raramente têm em consideração as consequências no desempenho de um *website* ao idealizar novas funcionalidades ou *designs*. Assim sendo, é fulcral que estes conheçam o impacto que as suas decisões possam ter sobre a UX numa aplicação *web* (Osmani, 2018), sendo necessário que as equipas de desenvolvimento transmitam esse conhecimento.

Assim, de modo a operacionalizar um orçamento de desempenho dentro de uma organização, é necessário (Lever, 2016):

- Incluir detalhes sobre o desempenho das aplicações na documentação dos projetos;

- Integrar o mais cedo possível *designs* no *browser*, de modo a detetar implicações dos *designs* mais atempadamente;

- Existir uma colaboração ativa entre as equipas de desenvolvimento e os *designers*, de modo a haver uma sintonia entre as várias disciplinas durante um projeto;

- Fornecer o conhecimento necessário para os *designers* terem em consideração durante o seu trabalho o desempenho como requisito, evitando assim a utilização de múltiplas fontes tipográficas e imagens de alta resolução (Kearney et al., 2019).

A existência deste tipo de orçamentos de desempenho é relevante, principalmente numa fase inicial do projeto pois promove discussões dos componentes a adicionar ou não a uma página, bem como a melhor forma de o apresentar ao invés de como apresentar o conteúdo. É contraproducente aplicar estas técnicas numa fase de maturação do projeto, pois os *designs* e conteúdos já se encontram definidos, sendo mais difícil gerir a remoção e alteração de

[17] Separação lógica do código-fonte, que possibilita o carregamento parcial do mesmo, à medida em que certas partes são necessárias para permitir o funcionamento de um *website*.

determinados aspetos de uma aplicação, pois numa fase inicial, oferece justificações para os *stakeholders* do porquê de uma certa funcionalidade ter sido omitida ou trocada (Kadlec, 2013).

Em suma, os orçamentos de desempenho conduzem para uma cultura de maior responsabilidade associada ao desempenho de soluções *web*, permitindo que os *stakeholders* considerem o impacto das métricas centradas no utilizador a cada alteração de uma página *web*.

2.4 Métricas de desempenho

A métrica, de acordo com a sua definição, define um sistema ou um padrão de medida (Oxford University Press, 2019) em que, no caso da temática do desempenho na *web*, está ligada à definição de métricas mensuráveis e possíveis de serem monitorizadas, que permitam obter informação relativamente ao estado de um *website* a partir do seu carregamento como também do seu do seu funcionamento.

As métricas podem ser agrupadas em diversas categorias (Bevacqua, 2015b):

- *Milestones Timings*, métricas relacionadas com tempo, associadas à experiência de utilização durante o carregamento de páginas (Kadlec, 2014b), tais como:
    - *Load Time*, representa o tempo de carregamento total de uma página;
    - *DOMContentLoaded*, representa o tempo de carregamento da DOM[18];
    - *Time to Interactive (TTI),* responsável por marcar o ponto em que a página se encontra totalmente *renderizada* e pronta a responder a qualquer interação do utilizador, através da identificação do momento em que o *script* inicial de *JavaScript* da página está carregado e se a *thread*[19] principal do *browser* se encontra livre (estado *idle*). É possível verificar esse ponto na Figura 4, assinalado como *Interactive*, que surge no momento em que não existe atividade de *scripts*, identificados como *Short JavaScript Tasks* para o código *JavaScript* a ser executado em pouco tempo e *Long JavaScript Tasks* para o código *JavaScript* mais demorado, que ocupa a *thread* principal e adia a interação do utilizador com o *browser*;
    - *Time to First Byte* (TTFB), responsável por representar o tempo decorrido até que o *browser* receba o primeiro *byte* de resposta, parte da informação solicitada a um servidor *web*, nomeadamente quando se tenta aceder a um URL de um *website*. Este período de tempo representa a latência de um pedido

[18] *Document Object Model*, representação de dados e objetos que representam a estrutura e conteúdo de um documento *web {Citation}*.

[19] Unidade mais reduzida de processamento que pode ser executada por parte do sistema operativo de um computador.

ao servidor, contemplando também o tempo que o servidor demorou a fornecer a resposta (Escardo, 2018).

- *Quantity-based*, baseadas em quantidade, métricas que lidam com pesos ou tamanhos de processos, operações realizadas e recursos da *web*, como é caso (Bevacqua, 2015b) :
    - Da contagem do número de pedidos (*request count*);
    - Do tamanho de uma página (*page weight*);
    - Do tamanho de uma imagem (*image weight*);
    - Outros dados que sejam facilmente monitorizáveis;
- *Rule-based*, baseadas em regras, métricas que disponibilizam pontuações (*scores*), normalmente gerados por ferramentas utilitárias que, através de um valor ou uma escala, permitem classificar o estado de um *website* de acordo com determinados parâmetros que uma ferramenta visa avaliar.

No entanto, e de acordo com a organização Google, existem um conjunto de métricas que devem ser consideradas na monitorização, pois estas focam-se na experiência da utilização na ótica do utilizador – as métricas *user-centric* – que surgem como resposta a questões associadas ao *feedback* visual que uma página *web* deve transmitir, de modo a garantir que tudo está a funcionar como expectável (Walton, 2019).

As métricas *user-centric* utilizadas são:

- **"Está a acontecer?":** avalia se um *website* fornece o conteúdo expectável de modo a permitir a respetiva interação, como é o caso da disponibilização da navegação ou da obtenção de respostas por parte do servidor. De modo a monitorizar esta experiência, utilizam-se as métricas:
    - *First Paint* (FP), responsável por marcar o ponto em que o primeiro conteúdo da página *web* é *renderizado* no *browser*, imediatamente após a *renderização* da navegação (ver Figura 5).
    - *First Contentful Paint* (FCP), responsável por marcar o ponto em que o *browser renderiza* o primeiro conteúdo da DOM – podendo ser texto, uma imagem ou um elemento HTML (ver Figura 5).
- **"É útil":** avalia se um website disponibiliza conteúdo útil suficiente de modo a permitir a interação dos utilizadores com o mesmo. Ao privilegiar o carregamento rápido das partes mais importantes de uma página, o utilizador final dificilmente irá perceber que o resto do conteúdo da página demorou mais tempo. De modo a monitorizar esta experiência, utilizam-se as métricas

- *First Meaningful Paint* (FMP), responsável por marcar o ponto em que o conteúdo primário de uma página surge no ecrã (por exemplo, resultados de pesquisa num motor de pesquisa ou o cabeçalho e a informação textual de um produto numa PDP) (ver Figura 5). É utilizada como métrica para avaliar a experiência percecionada pelo utilizador face ao tempo de carregamento de um *website (Sevilleja, 2018)*.

- *Hero Element Timing*, baseado na métrica anterior, mas que visa monitorizar o ponto em que o conteúdo mais importante (de destaque) de uma página é apresentado (por exemplo, ao abrir uma página de um vídeo no *website YouTube*[20], o elemento cujo tempo de carregamento será monitorizado irá ser o vídeo).

- **"É utilizável?":** avalia se é possível os utilizadores interagirem com uma página, ou se tal não é permitido por esta ainda estar a ser carregada. De modo a monitorizar esta experiência, são utilizadas as métricas:

  - *First CPU Idle*, também conhecida como *First Interactive,* é a métrica responsável por indicar o primeiro momento em que um utilizador pode interagir com uma página. Este momento ocorre quando grande parte dos elementos visuais são apresentados no ecrã e a página consegue responder às ações dos utilizadores (Google, 2019b).

  - *Visually Complete*, responsável por indicar o que um utilizador perceciona como o ponto temporal em que um website aparenta estar completamente carregado e funcional, conforme é possível verificar no momento assinalado na linha temporal da Figura 4.

    Este valor é calculado pela gravação de um vídeo de um carregamento de uma página, sendo este depois alvo de um processo de análise das várias capturas de ecrã (*frames*) realizadas pelo vídeo. Este processo, que recorre a algoritmos de comparação de imagens, procede à análise das mesmas e define o ponto visual do término de carregamento – denominado *visually complete* – na *frame* em que apresenta menos diferenças para a *frame* apresentada no fim do carregamento da página. Considerando o dinamismo que alguns *websites* podem apresentar (por exemplo *carousels* de imagens (cf. secção 2.2), vídeos com reprodução automática, ícones com pouca relevância e carregamento mais tardio) podem aumentar o grau de dificuldade para proceder ao cálculo preciso deste valor. Deste modo, surge a métrica *Visually Complete 85%* e outras variantes, concebida para determinar um *website* como carregado ligeiramente mais cedo do que a métrica *Visually Complete*, de modo a contemplar estes fatores (Calibre, 2019ª)

[20] Serviço de partilha de vídeos *online* entre utilizadores e organizações.

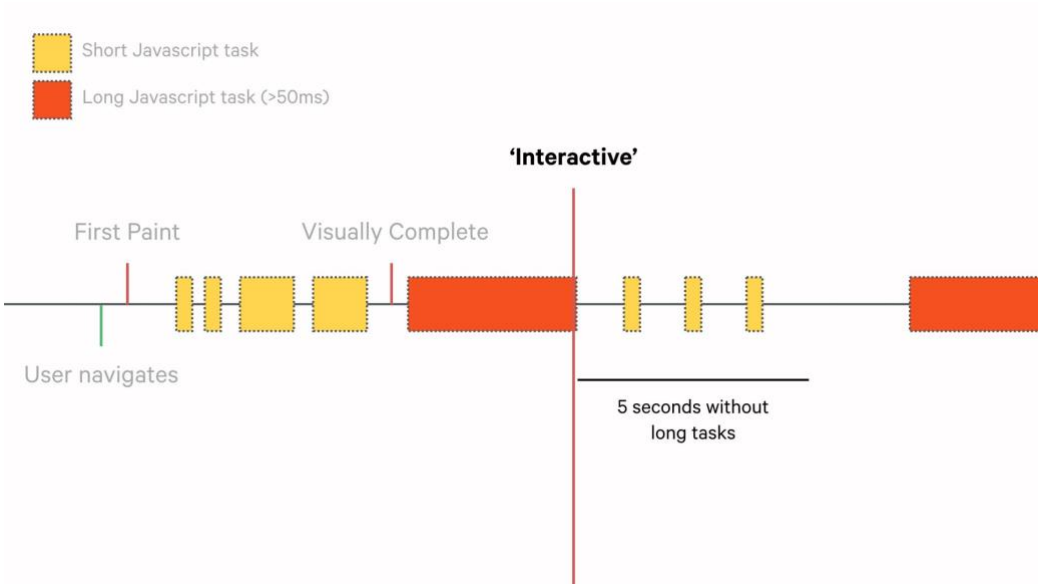


Figura 4. Linha temporal de atividade *JavaScript* durante o carregamento de uma página (Calibre, 2019b)

- **"É agradável?":** avalia se as interações dos utilizadores com o *website* são orgânicas, aprazíveis e sem nenhuma manifestação de entraves ou bloqueios. De modo a monitorizar esta experiência, é utilizada a métrica *Long Tasks*, responsável por verificar a ausência de tarefas de longa duração (ou seja, que possam tomar mais que 50 milissegundos a serem executadas) e possa causar o bloqueio da *thread* principal do *browser* e, consequentemente, o *website* não é capaz de responder ao input/ações do utilizador. A existência deste tipo de tarefas é a principal causa de se verificarem más experiências em diversos *websites*.

Na Figura 5, é possível observar um exemplo que demonstra as diferenças entre as várias métricas centradas no utilizador e na experiência, associado aos diferentes momentos de carregamento de uma página:

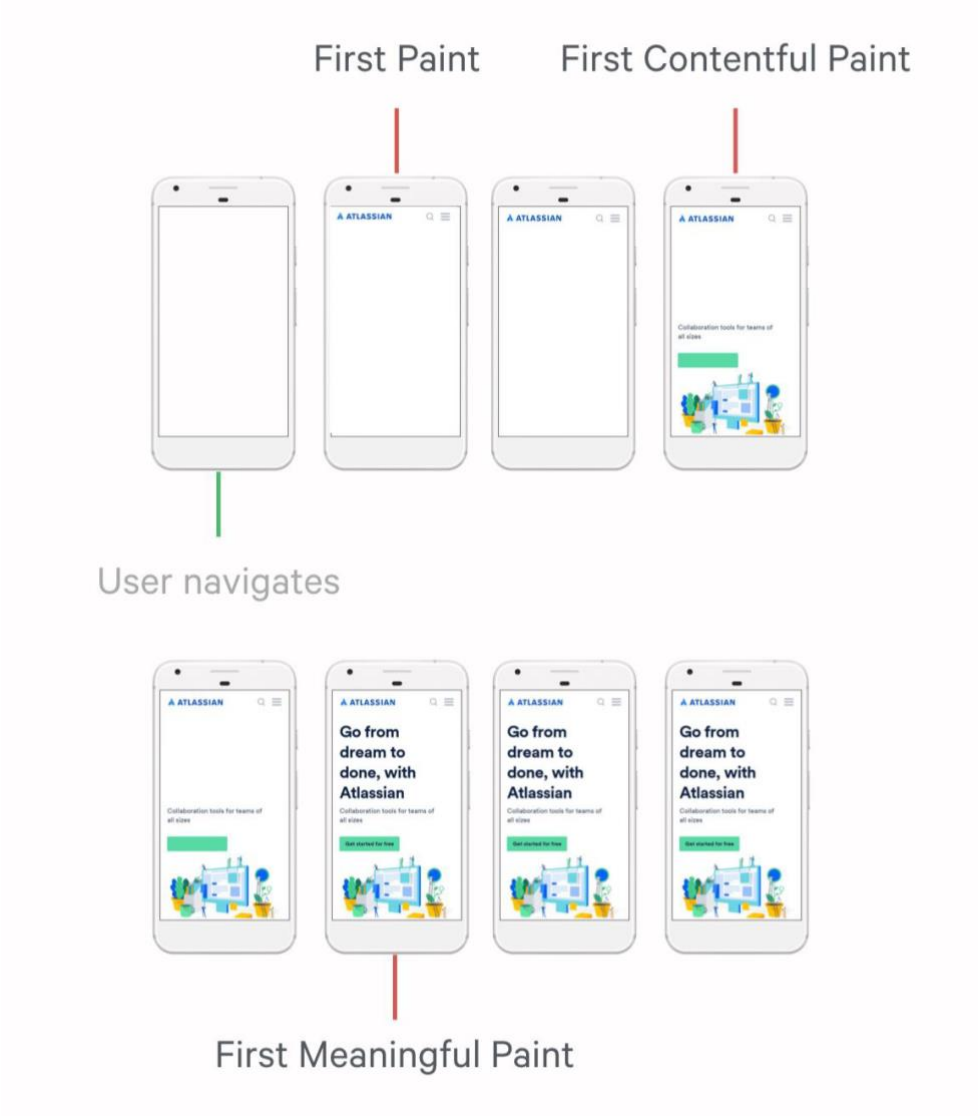


Figura 5. Ciclo de *renderização* de uma página *web*, indicando a métrica associada a cada momento (Calibre, 2019ª)

Existem também métricas proprietárias, que foram criadas por algumas ferramentas utilizadas para analisar e monitorizar o desempenho de *websites*, como é o caso do *Speed Index*, fornecido pela ferramenta *WebPageTest* (cf. secção 2.8.1), que consiste num "método de calcular a pontuação que indica o tempo decorrido até que uma página ficou no estado *Visually Complete*" (Calibre, 2019ª). Esta métrica é responsável por fornecer o tempo médio em que determinadas partes de uma página *web* são apresentadas, em que considera o progresso visual do processo de carregamento da página e, com base nisso, calcula uma pontuação média para a rapidez do *painting*, ou seja, a apresentação de conteúdo de uma página (Bevacqua, 2015ª).

2.5 Monitorização de desempenho

A monitorização de desempenho de aplicações *web* é definida como o processo de testes e verificação se os utilizadores finais conseguem interagir com o conteúdo *web* do modo que é expectável (Gasparyan, 2016) e de um modo eficiente (Uptrends B.V., 2019). Este processo é utilizado pelas organizações com o objetivo de assegurar o *uptime* de *websites*, o seu desempenho e as funcionalidades expectáveis.

Os dados recolhidos neste processo permitem um controlo da velocidade dos serviços oferecidos, bem como aumentar a satisfação do utilizador, promovendo uma maior retenção de clientes e, consequentemente, a melhoria de KPI relacionados com o negócio (Uptrends B.V., 2019).

Existem dois tipos de dados que as ferramentas de apoio à monitorização de desempenho permitem recolher:

- Dados sintéticos/laboratoriais;
- Dados reais (RUM).

Os dados sintéticos/laboratoriais [21] são dados de desempenho recolhidos dentro de um ambiente controlado com um dispositivo e condições de rede predefinidas. Este tipo de monitorização permite obter resultados reproduzíveis e potencia as capacidades de *debugging* [22], de modo a identificar, isolar e resolver possíveis problemas associados ao desempenho, além de permitir uma maior visibilidade na UX. No entanto, este tipo de dados pode não capturar possíveis gargalos (*bottlenecks* [23]) da utilização real, bem como não possibilita a correlação com os KPI (Google, 2019c).

Os dados reais fornecem informação mais precisa sobre experiências reais de utilização de um *website*, podendo recolher informação de condições distintas de utilização – ora por dispositivo, tipo de ligação à rede ou local de acesso – bem como permite a correlação com os KPI de negócio. No entanto, as capacidades de *debugging* são mais diminutas e o conjunto de métricas recolhidas é mais restrito (Google, 2019c).

Durante este processo de monitorização, existem diversos sinais a acompanhar, que são indicadores de possíveis incidências, especialmente no lado do servidor (RisingStack, 2017):

- Taxa de erros (*error rate*), que normalmente são apresentados aos utilizadores e afetam os clientes.

[21] Processo também reconhecido como *Synthetic Monitoring*.

[22] Processo de verificação e remoção de erros no código-fonte de um *software*.

[23] Ponto de bloqueio ou saturação.

- Tempo de resposta (*response time*), dado que a latência afeta diretamente os clientes e consequentemente o negócio.

- Taxa de transferência (*throughput*), o tráfego permite compreender o contexto para o aumento da taxa de erros e da latência.

- Saturação (*Saturation*), que indica a capacidade operacional do serviço, tipicamente associado à utilização de CPU, de modo a saber se é possível o sistema suportar mais tráfego.

2.6 Benchmarking

Benchmarking é o processo utilizado para proceder à comparação do desempenho de um sistema, seja de *hardware* ou de *software*, denominado o "sistema sobre teste" – *system under test* (SUT). Para qualquer processo de *benchmarking*, especialmente de uma aplicação *web*, é relevante ter em consideração os seguintes componentes (Menascé, 2002):

- Especificação da carga (*workload specification*), onde é determinado o tipo de pedidos e a frequência com que estes pedidos são submetidos ao SUT, podendo decorrer num ambiente controlado ou mais realista, onde neste tipo de ambiente utilizadores reais realizam pedidos à SUT.

- Especificação de métricas, onde é determinado o que será medido.

- Especificação do procedimento de medida, onde é determinado o modo como as métricas serão obtidas.

De um modo geral, o processo de *benchmarking* é uma atividade de importância ao nível do negócio, pois deste modo torna-se possível conhecer como uma aplicação *web* está posicionada face à sua concorrência (Menascé, 2002) e, com isto determinar possíveis melhorias para contribuir para a evolução da solução a desenvolver.

2.7 Profiling

O *profiling* de desempenho de uma aplicação consiste no processo em que analisado um dado *software* em execução a fim de avaliar a informação recolhida sobre essa via (Metz e Lencevicius, 2003), de modo a poder analisar essa informação para avaliar se uma aplicação está a ter um desempenho adequado.

Este processo faz uso de diversas ferramentas que procedem à medição de quanto tempo uma rotina pode demorar a ser executada, a frequência e proveniência das chamadas bem como o tempo total despendido a executar em determinados locais (Smartbear, 2019).

De acordo com os métodos utilizados, os resultados deste processo podem variar, bem como podem afetar a possibilidade de otimizar os projetos.

Os métodos de *profiling* dividem-se em duas categorias:

- instrumentação (*instrumenting*);
- amostragem (*sampling*).

Os *profilers* dedicados à instrumentação dedicam-se a medir o tempo real despendido por cada rotina a cada chamada, recorrendo à inserção de instruções especiais no início e fim de cada rotina. Além de poder informar que sub-rotinas podem ser chamadas com origem numa dada rotina, pode também fornecer informação temporal com mais granularidade. Atualmente no mercado existem dois tipos de *profilers* de instrumentação (Smartbear, 2019):

- *Profilers* com modificação de código-fonte (*source-code modifying profilers*), que podem causar algumas dificuldades, dado a possíveis conflitos com o código original. Para utilizar este tipo de *profilers*, recorre-se à duplicação de código-fonte dos projetos de modo a evitar possíveis corrupções de informação.

- *Profilers* hierárquicos (ou binários), que trabalham apenas em tempo de execução (*runtime*), através da inserção de instrumentação diretamente no código executável da aplicação enquanto ainda está em memória, evitando a corrupção do código-fonte por não estar a ser acedido diretamente. Este tipo de *profilers* permite a descoberta de código mais lento em execuções iniciais.

Os *profilers* dedicados à amostragem (*sampling profilers*) permitem que as aplicações sejam executadas sem quaisquer modificações em tempo de execução, onde todo o processo de *profiling* é realizado à parte. É adequado para isolar rotinas pequenas e que são frequentemente chamadas, que podem causar possíveis *bottlenecks* durante a execução de uma aplicação. No entanto, este tipo de *profiling* não permite conhecer a origem da rotina chamada nem o tempo despendido a executar o seu código, pois só fornece informação do estado atual de execução (Smartbear, 2019).

Em suma, é possível verificar que os *profilers* podem diferir entre si e que estes possuem as suas vantagens e desvantagens, devendo cada um ser aplicado de modo adequado a aspetos específicos de testes a uma aplicação em desenvolvimento. Assim, de modo a isolar *bottlenecks* de uma forma precisa e frutífera, é necessário utilizar uma combinação de *profilers* para este processo.

2.8 Soluções Existentes

A presente secção apresenta uma análise e descrição de algumas soluções existentes no mercado, no que diz respeito a ferramentas de análise e monitorização do desempenho de aplicações *web*.

Com esta análise, foi possível verificar quais as ferramentas que melhor se podem enquadrar com os objetivos do projeto, bem como o tipo de resultados e funcionalidades que disponibilizam, de modo a poder considerar a utilização das mesmas ajustadas ao contexto deste projeto.

2.8.1 *WebPageTest*

O *WebPageTest* é uma ferramenta *open-source* utilizada para a medição e análise do desempenho de páginas *web* (Meenan, 2019), originalmente desenvolvida pela AOL[24] para utilização interna e que posteriormente foi disponibilizada como *open-source*[25] em 2008 («WebPageTest - About», 2019).

Para além desta ferramenta estar publicamente disponível em https://www.webpagetest.org, também é possível a sua utilização através de:

- Instâncias privadas, ora com alojamento próprio ou com recurso a imagens *Docker*:
    - Com alojamento próprio é possível implantar um servidor e agentes destinadas à execução do *WebPageTest*. O servidor é responsável por delega os pedidos de execução de testes de desempenho ao agente selecionado, bem como processar, armazenar e disponibilizar os resultados obtidos para consulta. Os agentes são responsáveis por simular dispositivos e/ou *browsers* distintos responsáveis por navegar para o *website* em análise e recolher métricas, comunicando com o servidor para tal (Meenan, 2018);
    - O *Docker* permite correr aplicações num ambiente isolado e virtual – um *container* – utilizando para tal apenas o *hardware* e poder computacional de um computador físico (o *host*), não havendo a necessidade de configurações e instalação de recursos adicionais. As imagens *Docker* do *WebPageTest* – uma para o servidor e outra para os agentes do *WebPageTest* – contém toda as configurações e dependências necessárias para disponibilizar um servidor e agentes num ambiente isolado. Estando na posse dessas imagens, apenas é necessário indicar ao *software Docker* para sendo apenas necessário fornecer as instruções necessárias para iniciar os dois *containers* e respetivas instruções

[24] *America Online*, portal *web* e fornecedor de serviços *online* nova-iorquino.

[25] *Software* cujo código-fonte original é tornado livre e pode ser redistribuído ou modificado.

de orquestração, descritas num ficheiro denominado *docker-compose* (John, 2018).

- Bibliotecas *open-source* criados pela comunidade em *Node.js*[26] - como é o caso do *WebPageTest API Wrapper* (Duran, 2019) que possibilita a sua integração:

    - Em *pipelines*[27] de *Continuous Integration*/*Continuous Delivery*, um ambiente de desenvolvimento que visa aproximar os processos técnicos de desenvolvimento com os operacionais, ao automatizar tarefas de compilação, teste e *deployment* (Duran, 2017; Tuli, 2018);

    - No desenvolvimento de aplicações que necessitem de comunicar com instâncias públicas ou privadas do *WebPageTest*, ao disponibilizar uma biblioteca possível de ser importada e utilizada de modo a poder executar testes e obter resultados programaticamente (Duran, 2019).

O teste de desempenho por parte do *WebPageTest* permite selecionar a localização do teste, o tipo de dispositivo e o *browser* a utilizar. Além destas configurações básicas, é possível fazer outros ajustes mais particulares, como:

- o tipo de conexão de rede a utilizar;
- definir o número de testes para correr;
- definir se pretende a captura de vídeo dos testes;
- a configuração de definições específicas do *browser* Google Chrome;
- a opção de capturar os *logs*[28] da atividade de rede;
- a opção de emitir um relatório do Google *Lighthouse* (cf. secção 2.8.3);
- a definição de métricas customizáveis a capturar;
- a configuração de opções associadas ao modo de funcionamento do *browser* de teste, como é o caso da desativação do *JavaScript*.

Ao realizar um teste de desempenho a uma página, o resultado deste teste é apresentado sob a forma de um relatório (Figura 6) que possui diversas classificações face a um conjunto de pontos-chave alvo de análise, bem como valores associados a algumas métricas medidas por

[26] Motor multi-plataforma para execução de código-fonte *JavaScript*.

[27] Conjunto de elementos processadores de dados ligados em série, em que o resultado do valor produzido por um processador, é o valor de entrada do processador seguinte.

[28] Registos de eventos ligados à execução de um *software*.

esta ferramenta. No caso de se proceder à utilização do *WebPageTest* através de integrações como o *WebPageTest API*, este tipo de relatório é retornado sob a forma de uma resposta em formato JSON, permitindo a sua integração e processamento da informação retornada por outras ferramentas (Viscomi et al., 2015).

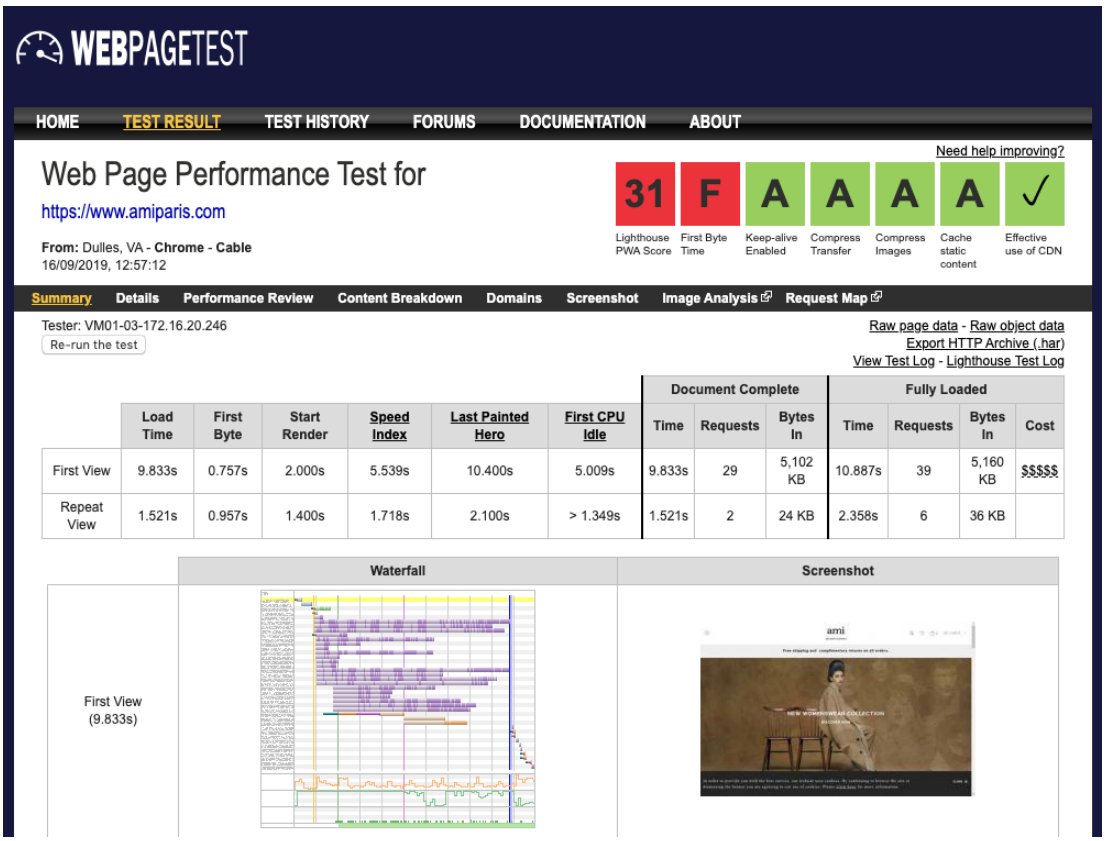


Figura 6. Relatório de um teste de desempenho a um *website* gerado pelo *WebPageTest*

Como se pode observar o relatório apresentado na Figura 6, este fornece diversa informação, que se agrupam nas seguintes categorias (Meenan, 2019):

- Classificação de otimizações, disponibilizadas na área superior do relatório com um conjunto de classificações (numa escala de A a F) para as otimizações de desempenho mais críticas que são aplicáveis a qualquer tipo de *website*. Qualquer rúbrica que não possua uma classificação entre A ou B exige um processo de análise e investigação cuidada. As rúbricas analisadas são:
    - *Keep-alive enabled,* uma configuração do servidor que permite a redução do tempo de carregamento de uma página entre 40 a 50%, pela reutilização de uma ligação HTTP já efetuada para realização de pedidos de conteúdo para uma página;
    - *Compress Text* (compressão de texto), que se trata de uma configuração do lado do servidor que permite a melhoria do desempenho e reduzir custos da

disponibilização de conteúdos, através da redução do tamanho dos dados transmitidos;

- *Compress Images* (compressão de imagens), em que verifica se as fotos disponibilizadas não se encontram na qualidade máxima e em formato otimizado para a *web* (ou seja, sem metadados associados à imagem);

- *Cache Static Content* (persistência de dados estáticos), em que verifica se o conteúdo de um *website* não modificado com frequência é colocado em *cache* por parte do *browser* de modo a reduzir pedidos;

- *Combine JS/CSS Files* (combinação de ficheiros *JavaScript* e CSS), onde é verificado a união de código *JavaScript* num único ficheiro (*bundle*) e o mesmo para o código CSS, de modo a permitir a redução de pedidos e o tempo de *renderização*;

- *Use of CDN* (utilização de *CDN*), que permitem a redução dos pedidos ao servidor da aplicação, através da disponibilização de conteúdo estático de um *website* com servidores distribuídos que estão mais próximos dos utilizadores finais;

- Métricas de alto nível, que fornecem informação sobre a página que foi carregada, como:

  - *First View*, onde fornece os tempos associados a testes realizados com um *browser* com a sua *cache* e *cookies*[29] limpas, representando a experiência de uma visita de um novo utilizador ao *website*;

  - *Repeat View*, onde fornece os tempos associados a testes realizados imediatamente a seguir aos testes *First View*, sem proceder a limpeza de dados previamente obtidos, representando assim a experiência de um utilizador que visita novamente um *website* ao qual já tinha ido anteriormente;

  - *Document complete*, um agrupamento de métricas recolhidas (tempo, número de pedidos e tamanho dos dados recebidos) até que o *browser* considere que a página está carregada, o que normalmente ocorre assim que as imagens estejam carregadas, mas que não contém conteúdo adicionado pela execução de *scripts JavaScript*;

  - *Fully Loaded*, um agrupamento de métricas recolhidas (tempo, número de pedidos e tamanho dos dados recebidos) até que ocorra 2 segundos sem atividade de rede após o *Document Complete*, onde inclui atividade

[29] Pequeno conjunto de dados enviados por um *website* que é armazenado por parte do *browser* de um utilizador durante a sua vista ao respetivo *website*.

normalmente lançada por *scripts JavaScript* após o carregamento inicial da página;

- *Load Time*, em que representa o tempo desde que o utilizador iniciou a navegação da página até que todo o conteúdo da página esteja carregado (evento *Document Complete*);
- *First Byte*, representa o tempo despendido pelo servidor ao construir a página para o utilizador, com base no início da navegação na página até ao momento em que a resposta do servidor chega;
- *Start Render*, que representa o ponto no tempo em que algo foi apresentado no ecrã, sendo o primeiro *feedback* para o utilizador de que algo está a acontecer;
- *DOM Elements*, que representa a contagem total dos elementos DOM medida no fim do teste;
- *Requests*, que representa o número de pedidos realizados pelo *browser* por conteúdo necessário para a página como imagens, *scripts JavaScript* e CSS;
- *Bytes In*, que representa a quantidade de dados que o *browser* teve que descarregar de modo a carregar a página, normalmente denominado de *Page Size* (tamanho da página).

2.8.2 **Calibre**

O *Calibre* é uma suite de ferramentas automáticas que visam monitorizar o desempenho de um site, fornecendo o conhecimento necessário para melhorar a experiência de utilização.

Esta ferramenta disponibiliza diversas funcionalidades como (Calibre, n.d.):

- A integração com *Google Lighthouse* (cf. secção 2.8.3);
- Monitorização automática e distribuída, onde é possível executar testes a partir de várias localizações geográficas;
- Emular dispositivos reais e ligações reais, com a possibilidade de criar perfis de teste que permitem simular as condições dos utilizadores;
- Monitorização e obtenção de métricas centradas no utilizador, como o *Visually Complete* e *Time to Interactive* (cf. secção 2.4), que permitem determinar o quão rápido os utilizadores finais conseguem interagir com a interface do *website*;
- Análise detalhada de pedidos e de conteúdos disponibilizados;

- Definir orçamentos de desempenho (*performance budget*) para o conjunto de métricas suportadas pela ferramenta (cf. secções 2.3 e 4.1.2), com a possibilidade de definir alertas quando um *website* ultrapassa o valor definido;

- A configuração de alertas em tempo real através de *email* ou integrações com ferramentas de comunicação como o Slack[30] .

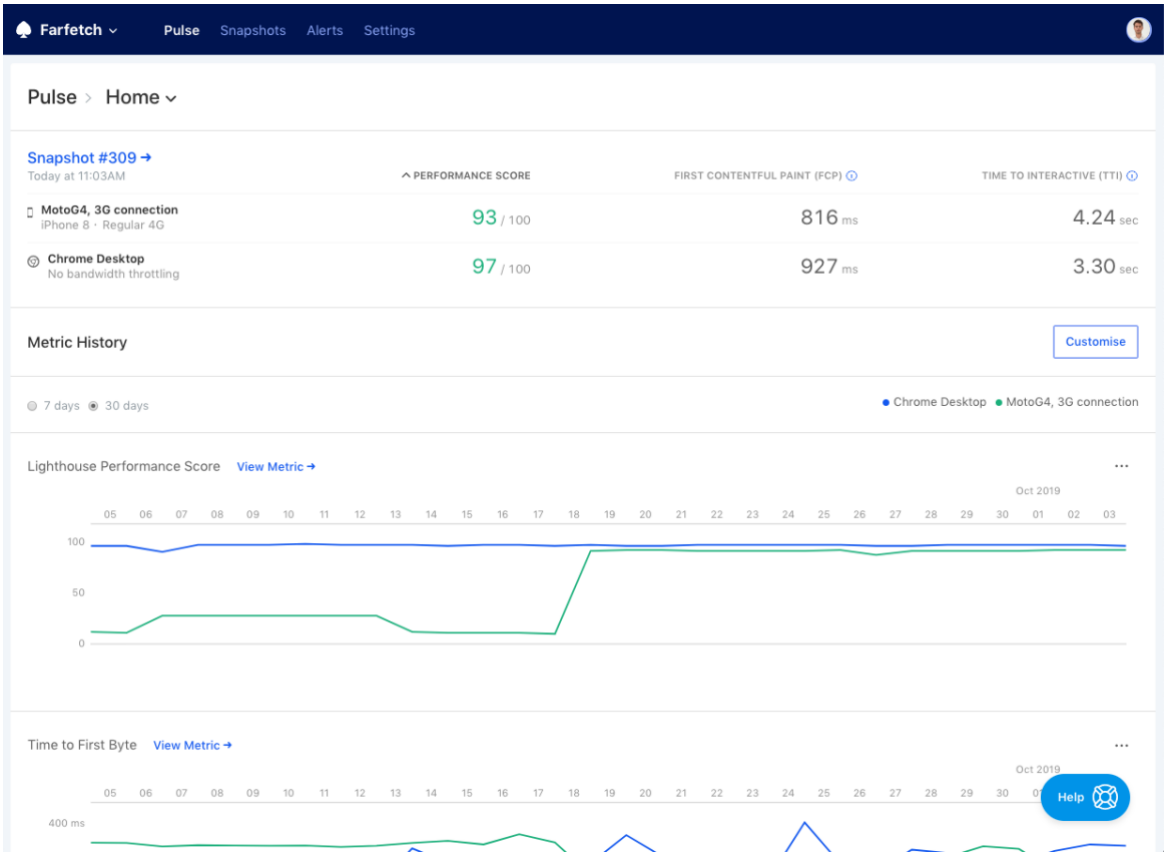


Figura 7. Página principal do *Calibre*, responsável por fornecer informação e gráficos gerais sobre as métricas principais monitorizadas de um *website*

Esta ferramenta, para além de possibilitar a monitorização de um *website* sem recurso a instalação de ficheiros adicionais, disponibiliza também uma interface CLI através de um módulo *Node.js*, o que abre portas a todo um novo conjunto de funcionalidades para a integração com a *pipeline* de desenvolvimento como a criação de testes individuais de desempenho a uma página específica; a integração com sistemas de CI/CD, que possibilita regressões de desempenho antes de determinadas versões de *software* serem colocadas em produção

[30] Conjunto de ferramentas de colaboração e serviços para equipas, utilizado principalmente para comunicação interna dentro de organizações.

2.8.3 **Google Lighthouse**

O *Google Lighthouse* é uma ferramenta *open-source* automatizada que visa melhorar a qualidade das páginas *web*, através de auditorias de desempenho, acessibilidade e *Progressive Web Apps*[31]. Esta ferramenta encontra-se disponível nas ferramentas para desenvolvedores disponibilizadas *browser Google Chrome* (Chrome *DevTools*), bem como é possível a sua utilização via linha de comandos ou como um módulo de *Node.js* (Google, 2018b). Existem também ferramentas que permitem a integração do *Lighthouse* em *Pull Requests*[32] em repositórios *GitHub*[33] (Bidelman, 2019).

Ao fornecer um URL ao *Lighthouse*, este procede à execução de uma série de auditorias relativamente ao *website* fornecido, em que, uma vez terminado esse processo, gera um relatório com os detalhes que refletem o estado da página, bem como a disponibilização de indicadores sobre como melhorar a página de modo a obter uma melhor classificação (Google, 2018b).

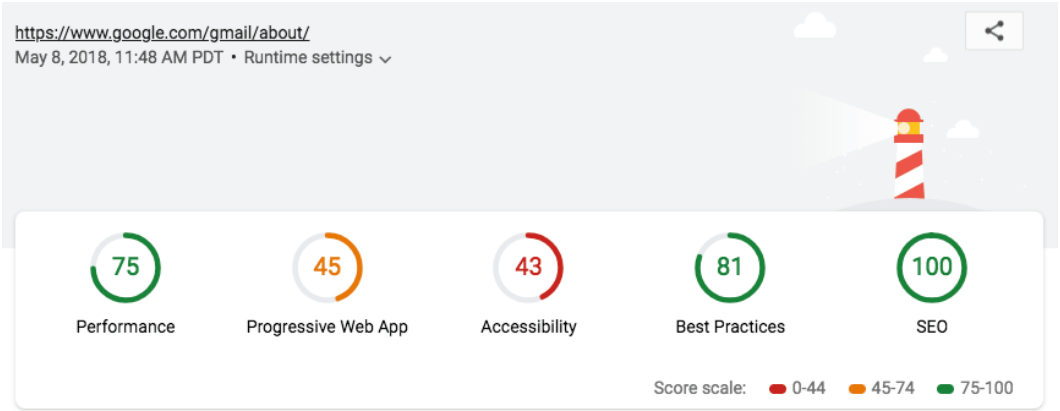


Figura 8. Exemplo de relatório produzido por uma auditoria do *Google Lighthouse*, com as pontuações obtidas para cada categoria auditorada

As auditorias realizadas por esta aplicação dividem-se nas seguintes categorias, tal como é possível observar na figura anterior:

- *Performance*, auditoria dedicada à verificação do valor associado a dadas métricas e à adoção de boas práticas essenciais ao fornecimento de um *website* com um bom desempenho, conforme demonstrado na Figura 9. As verificações realizadas são:

[31] Aplicações *web* que pretendem combinar a flexibilidade da *web* com a experiência de aplicações móveis nativas, ao disponibilizar recursos adicionais como funcionamento em modo *offline*, o envio de notificações e o acesso ao *hardware*.

[32] Processo associado ao sistema de controlo de versões distribuído *Git*, que consiste na comunicação de alterações efetuadas a um projeto partilhado, onde é permitido a revisão e a verificação das modificações efetuadas antes de serem integradas no projeto.

[33] Alojamento *web* para projetos de *software* que utilizam o sistema de controlo de versões *Git*.

- Verificação da corrente de pedidos críticos (*Critical Request Chains*);
- Deferimento de CSS não utilizado;
- Utilização de compressão de texto;
- Latência para introdução de dados (*Estimated Input Latency*);
- Análise da métrica *First Contentful Paint* (cf. secção 2.4);
- Análise da métrica *First CPU Idle* (cf. secção 2.4);
- Análise da métrica *First Meaningful Paint* (cf. secção 2.4);
- Verificação de *payloads*[34] de rede com tamanho elevado;
- Verificação da existência de múltiplos redireccionamentos (*redirects*);
- Análise do tempo necessário para os *scripts JavaScript* serem executados;
- Análise dos tempos de resposta baixos por parte do servidor;
- Minificação de código CSS;
- Verificação de imagens relativamente à sua visibilidade, tamanho adequado e respetiva otimização (formato de imagem utilizado);
- Análise da métrica *SpeedIndex* (cf. secção 2.4);
- Análise do pré-carregamento de pedidos-chave (*preload key requests*);
- Análise da existência de scripts ou folhas de estilo que bloqueiem a *renderização* (*render-blocking scripts* e *render-blocking stylesheets*);
- Análise da métrica *Time to Interactive* (cf. secção 2.4);
- Utilização de uma árvore DOM com um tamanho adequado;
- Utilização de uma política eficiente para colocar em *cache* (*cache policy*) conteúdos estáticos (*static assets*).

[34] Expressão computacional utilizada no âmbito de comunicações de rede, destinada a identificar a informação/mensagem desejada no conjunto de dados enviados.

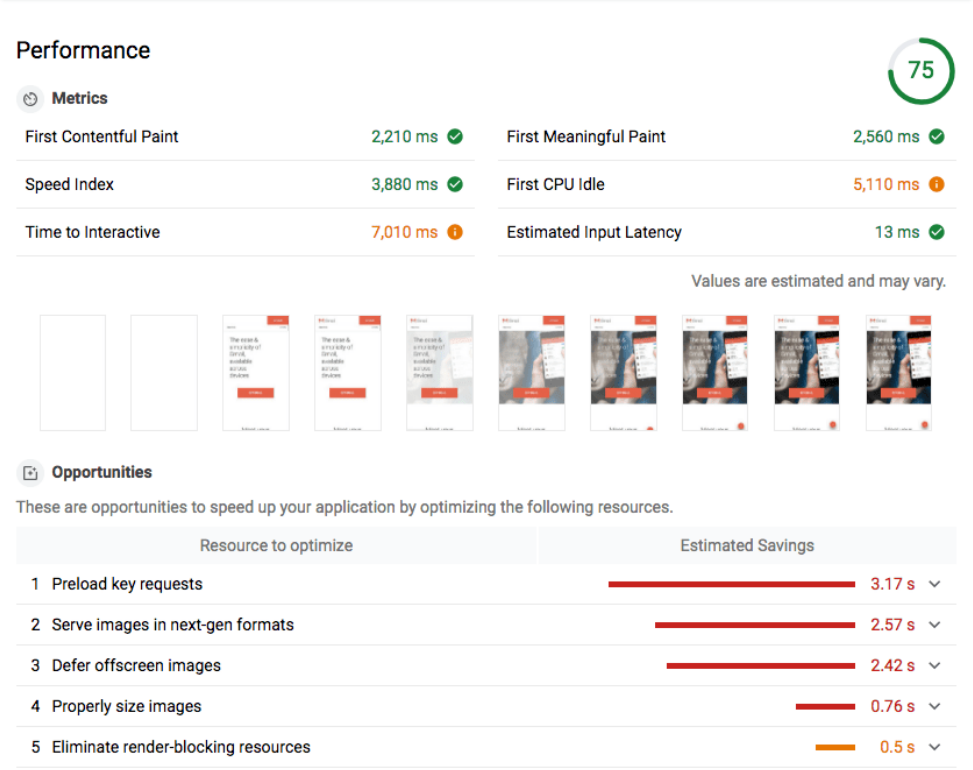


Figura 9. Detalhe da auditoria da categoria *Performance* pelo *Google Lighthouse*

- *Progressive Web App*, em que verifica os pontos essenciais para o fornecimento de uma página *web* preparada para ser utilizada como uma PWA;
- Acessibilidade (*Accessibility*), onde são verificados se os elementos da página possuem os atributos necessários para serem reconhecidos pelas tecnologias assistivas;
- Boas Práticas (*Best Practices*), onde são verificados diversos pontos mais transversais que estão relacionados com o panorama atual do desenvolvimento de *websites*, como é o caso da utilização de HTTPS, a (não) utilização de bibliotecas com vulnerabilidades de segurança, a possibilidade de um utilizador não colar conteúdo num campo de *password*, entre outras;
- *Search Engine Optimization* (SEO), onde são verificados pontos que contribuem para a maior visibilidade do *website* junto dos motores de pesquisa e que garantem um melhor posicionamento nos algoritmos de *ranking*.

2.8.4 Comparação das Ferramentas

Na seguinte tabela, é apresentada uma comparação entre a disponibilidade de funcionalidades entre ferramentas principais apresentadas nesta secção – o *WebPageTest* e o *Calibre*. Esta comparação é relevante, de modo a compreender os pontos fortes e as lacunas entre cada

ferramenta, de modo a poder saber como tirar mais partido de cada funcionalidade disponibilizada.

Tabela 1. Comparação entre o *WebPageTest* e o *Calibre*

Funcionalidade	***WebPageTest***	***Calibre***
Agendamento de monitorização	Não	Sim
Auditoria *Google Lighthouse*	Sim	Sim
Classificações específicas	Sim	Não
Configuração de *performance budget* na monitorização	Não	Sim
Detalhes de pedidos efetuados e recursos solicitados	Sim	Sim
Emulação de dispositivos móveis	Sim	Sim
Integração de alertas com aplicações externas	Não	Sim
Fornece módulo CLI	Sim, da comunidade	Sim, pelo criador
Fornece módulo *Node.js*	Sim, da comunidade	Sim, pelo criador
Monitorização de *websites* com vários dispositivos	Não	Sim
Permite flexibilidade na configuração dos agentes para teste	Sim	Não
Preço	Gratuito	Pago
Relatórios globais de métricas de *websites*	Não	Sim
Relatórios para cada execução de um teste	Sim (quando executado pelo *website*, CLI e módulo *Node.js*)	Sim (apenas via CLI ou módulo *Node.js*)
Sugestões de melhorias	Sim	Não
Suporte a diferentes tipos de *browsers*	Sim	Não
Testes a URL privados (*Intranet*)	Sim, através de uma instância privada	Não

2.9 Sumário

Com base nos conhecimentos adquiridos e desenvolvidos através do estudo de cada subsecção descrita neste capítulo, verifica-se que todos eles serão tidos em consideração durante o desenvolvimento do projeto proposto.

Dos temas abordados neste capítulo, sobressai a importância que o desempenho das aplicações web tem no dia de hoje e os benefícios que pode trazer para todos os intervenientes no desenvolvimento e utilização de uma aplicação *web*. Estes benefícios traduzem na:

- Obtenção de melhores resultados orientados para o sucesso do negócio;
- Verificação do potencial para o crescimento dos negócios das organizações que operam *online*;
- Produção de *software* mais simples, otimizado e com melhor qualidade por parte das equipas de desenvolvimento, o que facilita o processo de manutenção e evolução do mesmo e, consequentemente, fornece uma melhor experiência de utilização a quem visita o *website*.

Ao melhorar o desempenho de uma aplicação *web*, permite que um leque mais vasto de utilizadores a nível mundial tenha interesse e possa desfrutar da experiência de visita a um *website*. Com isto, é permitindo que um negócio *online* consiga explorar novos segmentos de mercado e, consequentemente, potenciar as receitas auferidas – por garantir que todos os dispositivos (*smartphones*, *tablets*, computadores fixos e portáteis) com as mais diversas características técnicas de processamento e armazenamento de informação, consigam utilizar uma aplicação *web* com tempos de resposta e processamento aceitáveis. É de notar que o processo de transmitir uma melhor experiência tem em consideração as condições diversificadas de acesso à *Internet*, que podem ser restringidas por questões sociais, políticas ou de infraestrutura que alguns grupos de utilizadores vivenciam, permitindo assim resolver questões de inclusão social (cf. secção 1.1.3).

Ao se ter consciência da importância do desempenho das aplicações durante o desenvolvimento de um projeto de *software*, compreende-se que esta deve ser adotada logo de início, que deve ser criado um *performance budget* (cf. secção 2.3) permitindo às equipas que assegurem um padrão de desempenho mínimo adequado, que permita reduzir os custos operacionais ao reduzir os recursos de rede e processamento utilizados pelos servidores e, por conseguinte, reduzir o impacto ambiental através de uma menor utilização de energia (cf. secção 2.2).

Em suma, o desempenho de um *website* requer que sejam adotadas boas práticas e alguns cuidados, mas é benéfico para todos os intervenientes do projeto (direta ou indiretamente) garantir a igualdade no acesso e utilização de informação e serviços, bem como ter um impacto positivo nas organizações (e a concretização dos seus objetivos), nas pessoas e no ambiente.

3 Design

O presente capítulo tem como objetivo a sistematização dos requisitos funcionais e não funcionais mencionados nos capítulos anteriores, bem como a apresentação e reflexão do *design* arquitetural adotado no âmbito do *software* protótipo a implementar.

O design arquitetural deve ser tido em consideração como uma das primeiras atividades a ser realizada na conceção de um *software*, pois este permite projetar um sistema antes da sua construção, podendo assim prevenir possíveis falhas, bem como delinear um plano a seguir durante o desenvolvimento, especialmente útil no caso da aplicação em projetos com um determinado grau de complexidade (Staveley, 2011).

Para a descrição de modelos arquiteturais de *softwares*, é utilizado o modelo 4+1, idealizado por Philippe Kruchten, modelo este que visa descrever sistemas de *software* complexos, com base na utilização de vistas múltiplas e concorrentes sendo elas a vista lógica, a vista de processo, a vista física, a vista de processo e a vista de cenários (Kruchten, 1995, p. 1).

De forma a representar as vistas sugeridas pelo modelo 4+1, é possível aplicar o modelo C4, idealizado por Simon Brown. Trata-se de uma abordagem por abstração quanto à representação da arquitetura de um *software* sob a forma de diagramas, de modo a representar o sistema da forma mais abstrata e evoluir, de forma a dar cada vez mais detalhe com uma granularidade mais fina (Brown, 2019). Este modelo defende a adoção dos seguintes níveis:

- Nível 1: de Sistema (*System Context*), representa o ponto inicial, demonstrando como o *software* encaixa no ambiente em que será integrado;
- Nível 2: de Contentores (*Containers*), ponto ampliado, em quem é demonstrado em alto-nível os blocos técnicos relativos ao *software* a ser desenvolvido;
- Nível 3: de Componentes (*Components*), amplia num contentor específico e demonstra os componentes que compõe o contentor em questão;

- Nível 4: de Código (*Code*), representa a ampliação de um componente individual, demonstrando o modo como esse componente é implementado sob a forma de diagramas UML.

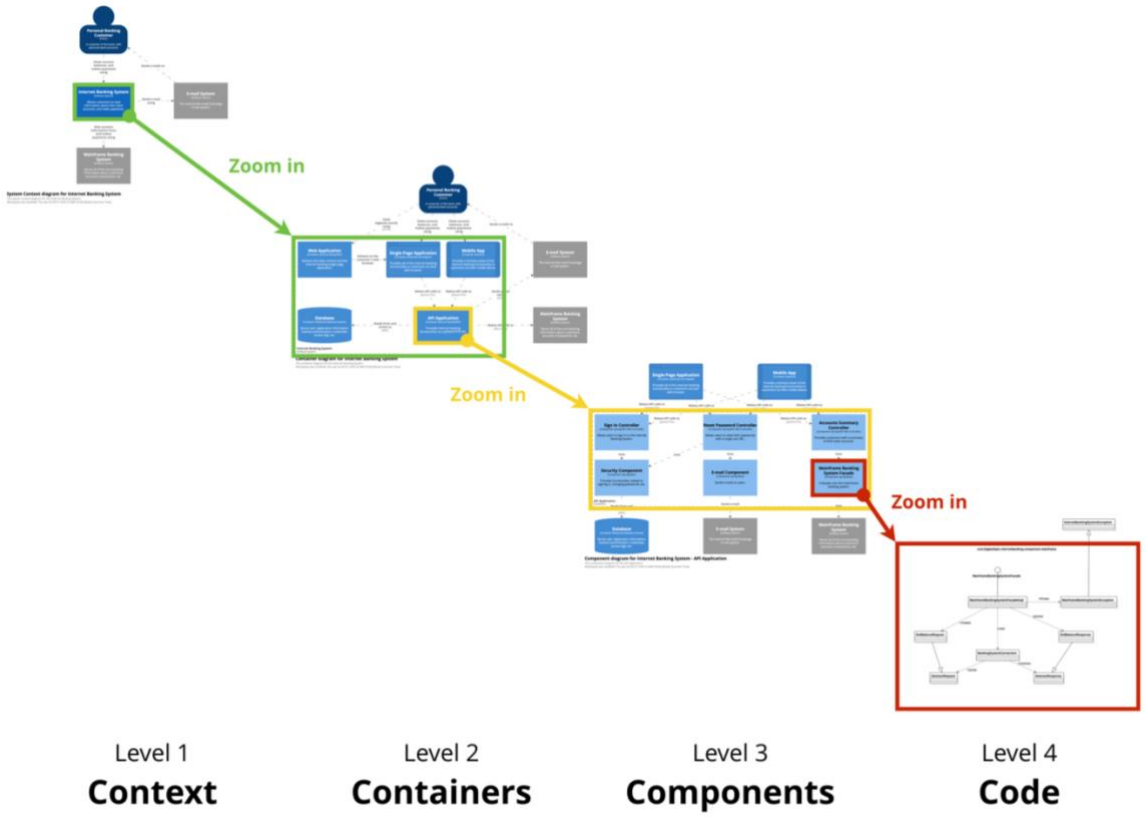


Figura 10. Representação visual dos níveis do modelo C4 (Brown, 2019)

Deste modo, a descrição do *design* arquitetural a adotar na solução idealizada será realizada com base nas vistas lógicas, de processo, física e de implantação provenientes do modelo 4+1, utilizado para tal três níveis de abstração, como sugeridos pelo modelo C4.

3.1 Nível 1 (de Sistema)

Nesta secção pretende-se representar o *software* num nível de abstração de alto-nível, de modo a demonstrar como este intervém no ambiente que o rodeia, sob a forma de diagramas.

Não obstante, serão descritos todos os requisitos funcionais e não funcionais inerentes ao projeto, assim como identificadas as tecnologias e *frameworks* que irão apoiar na implementação do mesmo.

3.1.1 **Vista lógica**

Com base no diagrama apresentado na Figura 11, pode-se verificar o *software* do protótipo a desenvolver em questão, denominado *FPS Performance System* e o modo como se liga com o meio envolvente.

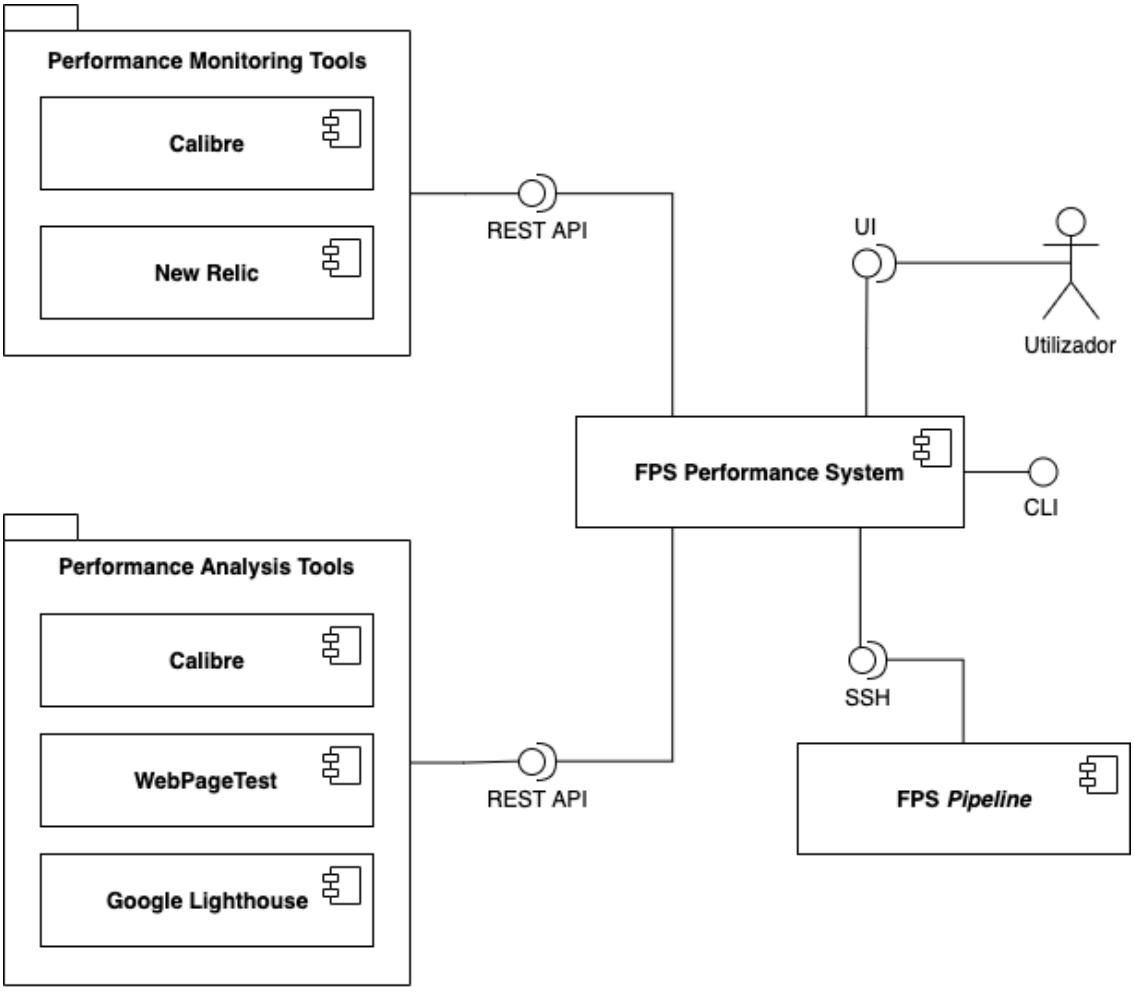


Figura 11. Diagrama de componentes relativo ao nível de sistema do modelo C4

Após análise do anterior diagrama, pode-se verificar:

- Um componente denominado *Performance Monitoring Tools*, onde o sistema se conecta de forma a obter informação relativa à monitorização efetuada nos *websites* de FPS;

- O componente *Performance Analysis Tool*, responsável por representar a conexão do sistema às ferramentas externas a utilizar de modo a realizar avaliações de desempenho dos *websites*;

- A existência de um ator, responsável por interagir com o sistema, de modo a poder definir o *performance budget* (cf. secção 2.3), de acordo com os dados obtidos do componente *Performance Monitoring Tools*;

- As interfaces CLI e SSH, responsáveis por representar a interface para ligação de componente externo ao sistema, como é o caso das *pipelines* de FPS (cf. secção A.2).

3.1.2 **Requisitos Funcionais**

Atendendo ao propósito do projeto em causa e o diagrama anterior, inferem-se os seguintes requisitos funcionais, representados na Figura 12:

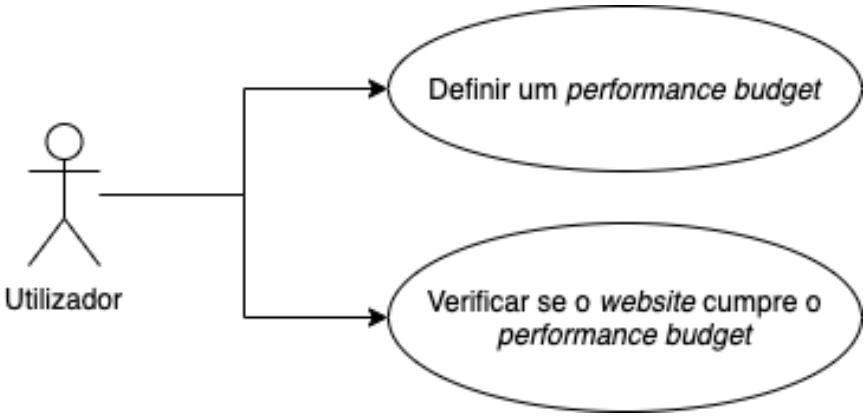


Figura 12. Diagrama de casos de uso (requisitos funcionais) da aplicação

Assim, com base nos casos de uso descritos na figura anterior, a aplicação pode ser utilizada com o seguinte fluxo:

1. Definir um *performance budget* a ser aplicado aquando da avaliação a efetuar a um determinado *website* produzido por FPS;

2. Executar um teste de desempenho a um *website* alvo de teste, com recurso a uma das ferramentas de teste integradas no protótipo, de modo a avaliar se cumpre ou não o *performance budget* definido anteriormente.

O seguinte diagrama, apresentado na Figura 13, pretende sistematizar o funcionamento destes requisitos funcionais, demonstrando como as partes integrantes do sistema identificadas na Figura 11 comunicam entre si.

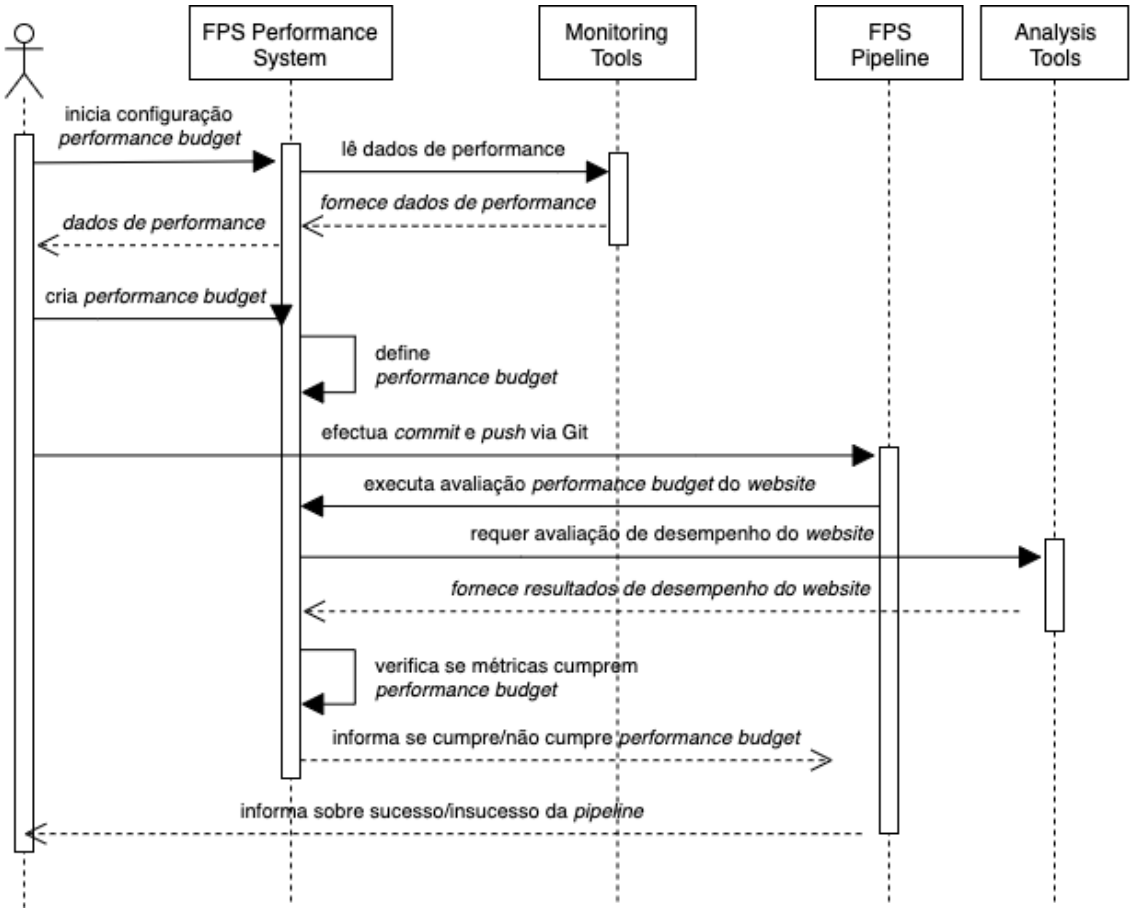


Figura 13. Diagrama de sequência relativo às interações efetuadas ao nível do sistema do modelo C4

3.1.3 **Requisitos Não Funcionais**

Os requisitos não-funcionais da aplicação a desenvolver, que já têm vindo a ser descritos no decorrer do documento, estão classificados segundo o modelo FURPS+.

O FURPS+ é um modelo de representação de atributos de caracterização da qualidade de um *software* sob a forma de requisitos funcionais e não funcionais, agrupados nas seguintes categorias:

- Funcionalidade (F);
- Usabilidade (U);
- Fiabilidade/Confiabilidade (R de *Reliability*);
- Desempenho ou *Performance* (P);

- Suportabilidade (S);
- Outras restrições (+) como: restrições de *design*, requisitos de implementação, de integração/interface ou requisitos físicos.

Os requisitos não-funcionais descritos sob o modelo FURPS+ associadas à aplicação desenvolvida listam-se na seguinte tabela:

Tabela 2. Requisitos não-funcionais da aplicação, segundo modelo FURPS+

Categoria do modelo FURPS+	**Requisito não funcional**
Funcionalidade (F)	Devem ser utilizados os dados de monitorização disponíveis nas ferramentas da organização sobre o estado de desempenho dos *websites* atuais.
	Deve permitir especificar um *performance budget* (cf. secção 2.3).
	Deve poder monitorizar as métricas de desempenho mais indicadas (cf. secção 2.4)
	Deve ser possível de integrar num contexto de *pipelines* destinadas a CI/CD (cf. secção A.2)
Usabilidade (U)	Eficácia e eficiência, na medida em que é possível integrar o protótipo numa *pipeline* de *CI/CD*, garantindo que fornece os detalhes necessários para informar do (não-)cumprimento de um *performance budget*, garantindo que a realização da avaliação continue a permitir o correto funcionamento do sistema de CI/CD da organização.
	Existência de documentação permita a compreensão de: • Como operar o *software* e integrar com um sistema de CI/CD pretendido; • Quais as métricas possíveis de serem monitorizadas; • Como poder definir um *performance budget* para operar o *software*.
Performance (P)	Bom desempenho, permitindo que os testes e a avaliação sejam executados em média num limite razoável de tempo – entre os 3 e 5 minutos

Categoria do modelo FURPS+	**Requisito não funcional**
Suportabilidade (S)	Modularidade, promovendo alta coesão e baixo acoplamento.
	Baixa complexidade algorítmica, permitindo a fácil manutenibilidade da aplicação e respetiva extensão pela utilização de uma arquitetura modular.
	Possibilidade de ser executada nos computadores das equipas de desenvolvimento.
Integração (+)	Aplicação de uma arquitetura flexível, que permita a fácil integração ferramentas de avaliação de *desempenho*, definida pela interface consumida para avaliar o *performance budget*.
Implementação (+)	Tecnologias a adotar: - *JavaScript* e *Node.js* - *WebPageTest* - *Calibre* - *Google Lighthouse* - *Docker* Bibliotecas *JavaScript* a adotar: - *Yargs* - *webpagetest-api* - *caliber-cli* - *lighthouse*

3.1.4 **Requisitos de Implementação (Tecnologias)**

Esta secção tem como dar a conhecer as tecnologias e bibliotecas identificadas como requisito não-funcional a adotar no âmbito do desenvolvimento do *software* protótipo:

As tecnologias a adotar são:

- *JavaScript*: linguagem de programação, compilada em tempo de execução, utilizada maioritariamente para o desenvolvimento de conteúdos Web e suportada por múltiplos browsers e outros ambientes não baseados em browsers. É caracterizada por ser uma linguagem *prototype-based* (que permite a criação de objetos sem a necessidade de definir inicialmente a sua classe), multi-paradigma, dinâmica, com suporte aos estilos de programação orientada a objetos, imperativa e declarativa (e.g. programação funcional) (Mozilla Corporation, 2019).

- *Node.js*: ambiente de execução em *runtime* de código *JavaScript open-source* e gratuito, suportado por múltiplas plataformas. Este ambiente corre o motor V8 de *JavaScript*, proveniente do cerne do *browser Google Chrome*, e com este, é possível correr código *JavaScript* no lado do servidor, bem como permite construir um conjunto de *scripts* com base nesta linguagem de programação (Borins et al., 2019).

- *Yargs*: biblioteca baseada em *Node.js* que possibilita o desenvolvimento de "comandos interativos, que suportam a passagem de argumentos", assim como permite criar uma "interface de utilização intuitiva" para CLI[35], graças à capacidade de gerar menus de ajuda com base nos argumentos fornecidos, validar os argumentos fornecidos com base nos valores autorizados/suportados pela aplicação («yargs/yargs», 2019).

- *Docker*: ferramenta destinada à criação, implantação e execução de aplicação pela utilização de *containers*, uma espécie de conglomerado de uma determinada aplicação e todas as suas dependências como bibliotecas, que é possível fornecer sob a forma de *package*. Graças à criação de um *container*, a aplicação pode correr de forma isolada em qualquer máquina com o mesmo sistema operativo, com qualquer tipo de configuração adicionais que difere das configurações utilizadas para o desenvolvimento e testes da aplicação.

  O *Docker* possui um funcionamento similar ao de uma máquina virtual, mas que ao invés de criar um sistema operativo virtual, este permite que as aplicações utilizem o *kernel* do sistema operativo (parte central do sistema operativo de um computador, responsável por interagir com *hardware*), sendo apenas necessário fornecer os conteúdos que podem não estar disponíveis no computador que irá servir como *host*, ou seja, a porta de acesso ao *hardware* tornando o processo de virtualização mais rápido e reduzido.

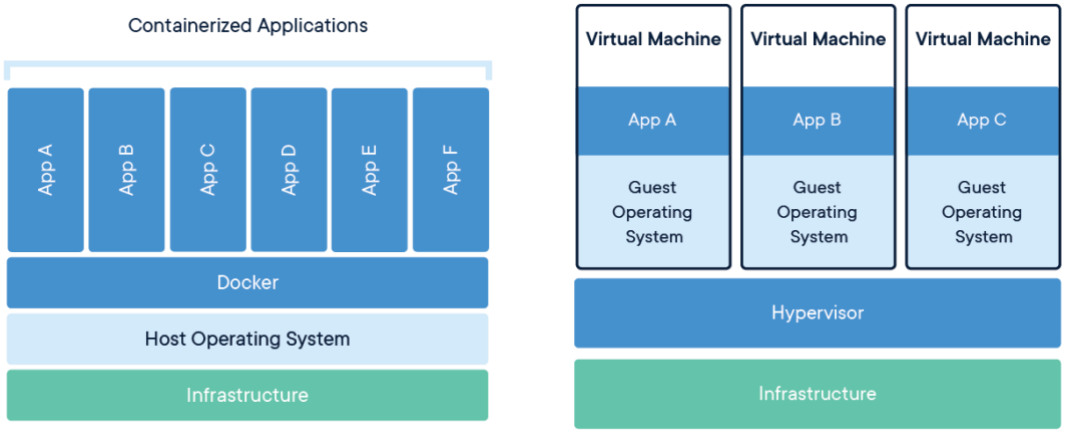


Figura 14. Comparação das camadas utilizadas entre aplicações a correr em *Docker* (à esquerda) e aplicações a correr em VM (à direita)

35

Para a análise e a obtenção de dados relativos ao desempenho de aplicações alvos de testes, foram utilizadas as seguintes ferramentas:

- *webpagetest-api* (Duran, 2019), um módulo *Node.js* utilizado para interagir com a instância pública e instâncias privadas do *WebPageTest* (cf. secção 2.8.1);
- *calibre-cli* (Calibre, 2019c), módulo *Node.js* utilizado para interagir com o serviço *Calibre* (cf. secção 2.8.2) utilizado na organização;
- *lighthouse* (Google, 2019d), módulo *Node.js*, que possibilita a utilização programática do *Google Lighthouse* (cf. secção 2.8.3), através da instanciação de uma janela do *Google Chrome* destinada a abrir o *website* a analisar, de modo a que o módulo possa comunicar com o *browser* e efetuar as devidas auditorias.

3.2 Nível 2 (de Contentores)

Nesta secção pretende-se representar o sistema como ponto ampliado, de modo a demonstrar, em formato alto-nível, os respetivos blocos técnicos inerentes ao *software* a ser desenvolvido.

3.2.1 **Vista Lógica**

O diagrama de componentes de alto-nível apresentado na Figura 13 representa os componentes de alto nível que intervêm no sistema a desenvolver e respetivas interfaces disponibilizadas, de modo a que os componentes exteriores possam comunicar com este componente.

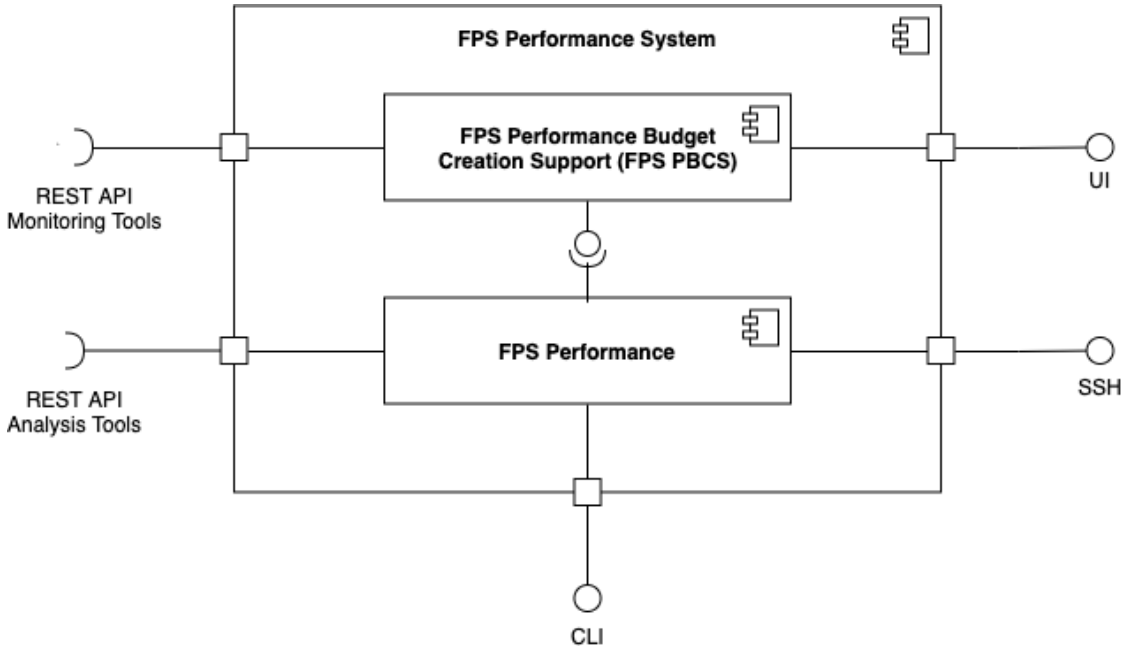


Figura 15. Diagrama de componentes referente ao nível de contentores do modelo C4

Como se pode observar, este sistema é composto por dois contentores:

- *FPS Performance Budget Creation Support*, contentor responsável garantir a definição de um *performance budget*, com base nos dados obtidos pela comunicação com as API REST destinadas à obtenção de informação dos dados de monitorização do desempenho de aplicações *web* de FPS já disponíveis *online*, disponibilizando para tal uma UI de modo a que um utilizador possa interagir com esta parte interveniente do sistema.

- *FPS Performance*, contentor responsável por avaliar o desempenho dos *websites* com base no *performance budget* definido pelo componente *FPS Performance Budget Creation Support*. Este é responsável por interagir com as ferramentas externas disponibilizadas para o efeito de análise e avaliação de desempenho comunicando, portanto, com as API REST disponibilizadas por essas ferramentas para o efeito, e disponibilizando uma interface CLI, podendo ser acedida por um utilizador ou uma *pipeline* de *CI/CD* via SSH de modo a poder inicializar a avaliação de desempenho de um *website*.

No âmbito do desenvolvimento deste projeto, só se irá focar no componente *FPS Performance*, sendo este o componente que consegue trazer mais valor à organização num curto espaço de tempo, de modo a poder já fornecer em tempo útil uma solução mínima viável (MVP), ficando o *FPS Performance Budget Creation Support* como trabalho futuro para este *software*.

As tarefas relativas ao componente *FPS Performance Budget Creation* serão realizadas como um processo humano, em que este interage com as ferramentas de monitorização

pelas respetivas interfaces *web* disponibilizadas ao público em geral, de modo a poder efetuar a recolha dos dados necessários para a definição de um *performance budget*, disponibilizando o resultado para ser usado no componente FPS Performance.

3.2.2 **Vista de Processo**

A vista de processo tem como objetivo apresentar as interações entre as respetivas partes do *software* durante a execução de operações que sejam arquitecturalmente significativas.

De seguida, na Figura 14, é apresentado um diagrama de sequência associado ao processo de interação de um ator com o sistema, de modo a poder visualizar graficamente como os componentes integrantes do *FPS Performance System* comunicam entre si e com as respetivas interfaces para o exterior.

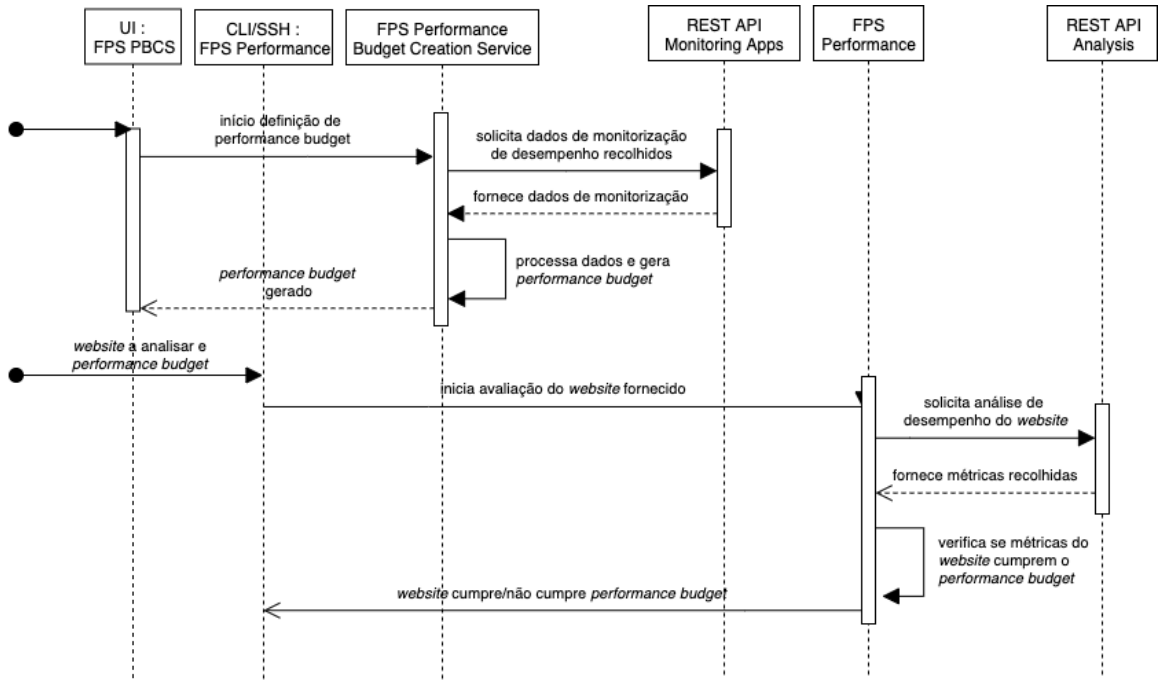


Figura 16. Diagrama de sequência relativo às interações efetuadas ao nível de contentores do modelo C4

3.2.3 **Vista física (de implantação)**

A vista física (ou de implantação) tem como objetivo representar o sistema sob o ponto de vista da disciplina de engenharia de sistemas, estando focado como os componentes integrantes do sistema se conectam fisicamente, em que é representado sob a forma de um diagrama de implantação (*deployment*).

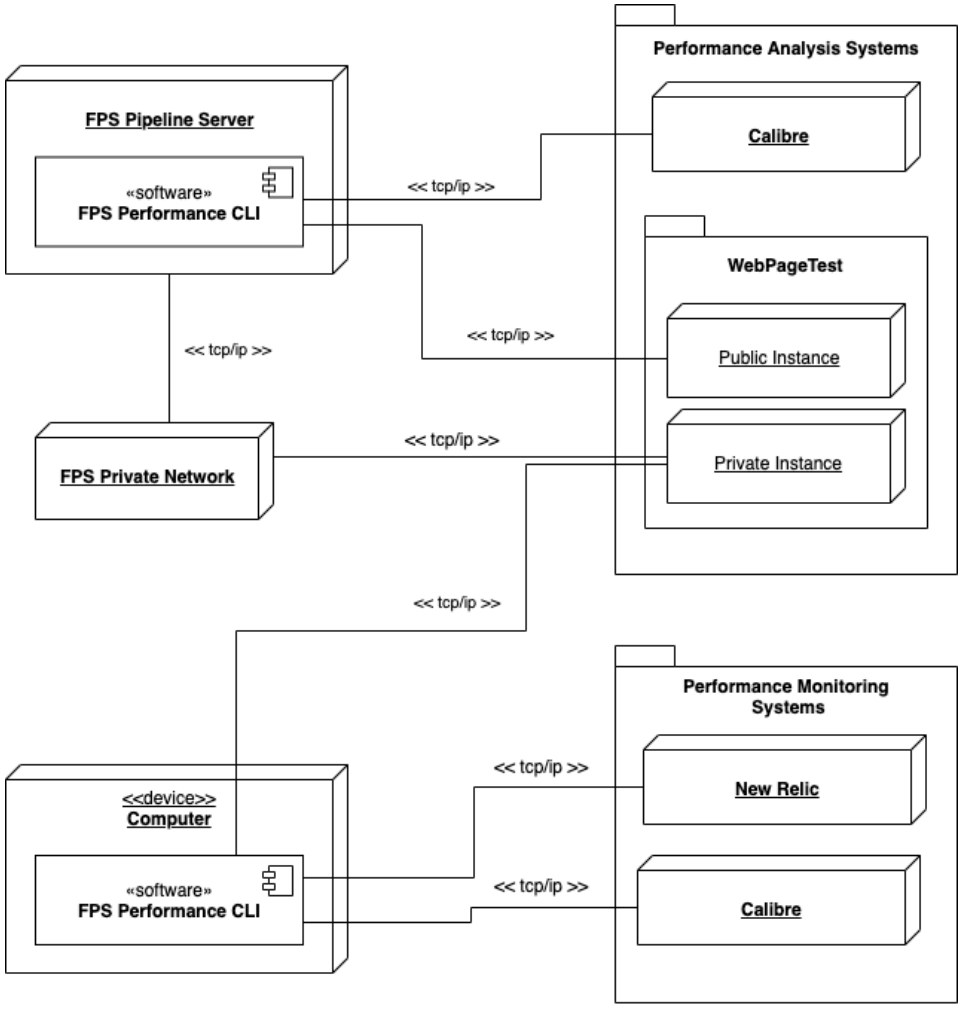


Figura 17. Diagrama de implantação associado ao nível de contentores segundo o modelo C4

De acordo com a Figura 17, é possível verificar o seguinte:

- Existem componentes do sistema que se encontram disponíveis apenas na rede interna da organização, identificada como *FPS Private Network*, sendo eles o servidor destinado à *pipeline* de CI/CD, em que esta contém uma instância do *software* em questão, com a denominação *FPS Performace CLI*.

- O componente *Private Instance* da ferramenta de análise de desempenho *WebPageTest* apenas está disponível também na rede privada da organização, ao contrário da *Public Instance*, que se encontra disponível para qualquer utilizador (inclusive a organização)

- As ferramentas de monitorização *Calibre* e *New Relic* não se encontram alojadas dentro da organização, pelo que cada uma tem o seu servidor público associado e disponível para mais utilizadores para além da organização (sendo FPS apenas um consumidor destes serviços);

- A ferramenta *Calibre* apesar de estar duplicada, representa o mesmo servidor, no entanto possui duplas responsabilidades: uma relacionada com o fornecimento de dados de monitorização e outra com o fornecimento do serviço para efetuar análises de desempenho de um *website*;

- O componente *FPS Performace CLI* também pode ser executado num computador de um colaborador (nomeadamente de um membro da equipa de desenvolvimento), de forma a poder este também executar os seus testes contra as ferramentas desejadas (podendo tirar total proveito, desde que esteja conectado à rede da organização).

3.3 Nível 3 (de Componentes) de FPS Performance

O nível de componentes do modelo C4, deste caso do componente *FPS Performance*, pretende ampliar o contentor identificado, de modo a poder representar os componentes associados ao contentor em questão.

3.3.1 **Vista lógica**

Na Figura 18 é possível observar o diagrama de componentes responsável por representar os vários componentes do contentor *FPS Performance*. Sendo este contentor o responsável por fornecer uma aplicação com uma interface CLI, de modo a que os utilizadores externos possam comunicar com esta, além de novamente a interface para comunicar com as ferramentas externas, identificada como *REST API Analysis Tool*.

Destaca-se os seguintes componentes e organização:

- Componente *Presentation Layer*, responsável por representar a interação do *software* com o utilizador, de modo a poder solicitar dados;

- Componente *Commands*, responsável por agregar os vários comandos que o *software* suporta;

- *Adapters*, responsável por fornecer os adaptadores necessários para que os comandos possam comunicar e rececionar a informação proveniente das ferramentas externas de análise utilizadas;

- *Performance Analysis*, *package* que agrega todos os componentes cuja responsabilidade recai sobre a análise dos dados antes de disponibilizar ao utilizador final. Possui as responsabilidades segregadas pelos seguintes componentes:

    - *Evaluation,* responsável por proceder ao início da avaliação do *performance budget* com base nos dados fornecidos pela ferramenta, delegando a tarefa de reconhecer e interpretar os dados ao componente *Interpreter*;

- *Interpreters*, responsável por reconhecer e processar as métricas recebidas pela ferramenta para um formato legível pela ferramenta, de modo a que o componente *Evaluation* reconheça as métricas e forneça a informação correta relativa a cada avaliação de métrica efectuada;

- *Dictionaries*, responsável por traduzir e fornecer dados fundamentais métricas recebidas para o formato reconhecido*, tais como* o nome e a chave da métrica, bem como a respetiva unidade de medida associada a cada métrica.

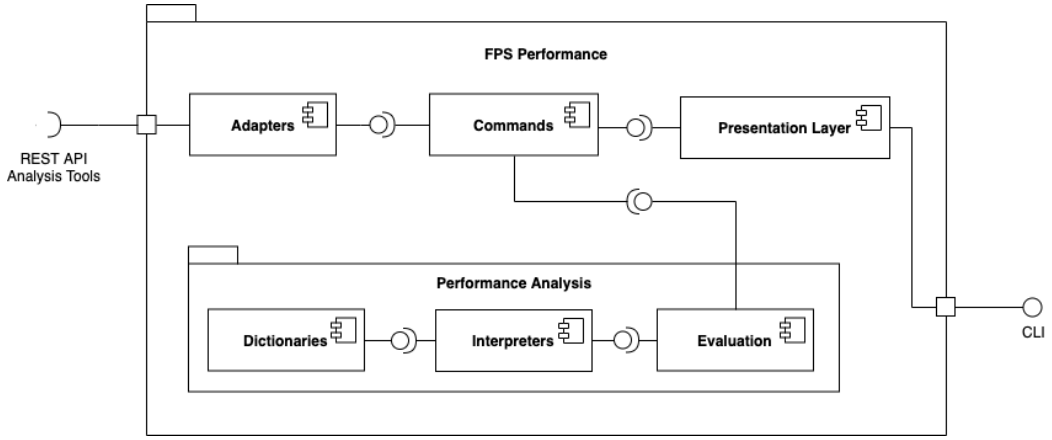


Figura 18. Diagrama de componentes referente ao contentor *FPS Performance*, nível de componentes do modelo C4

3.3.2 **Vista de implementação**

A vista de implementação tem como objetivo representar sob a forma de diagrama de *packages* o detalhe associado ao *software* a desenvolver.

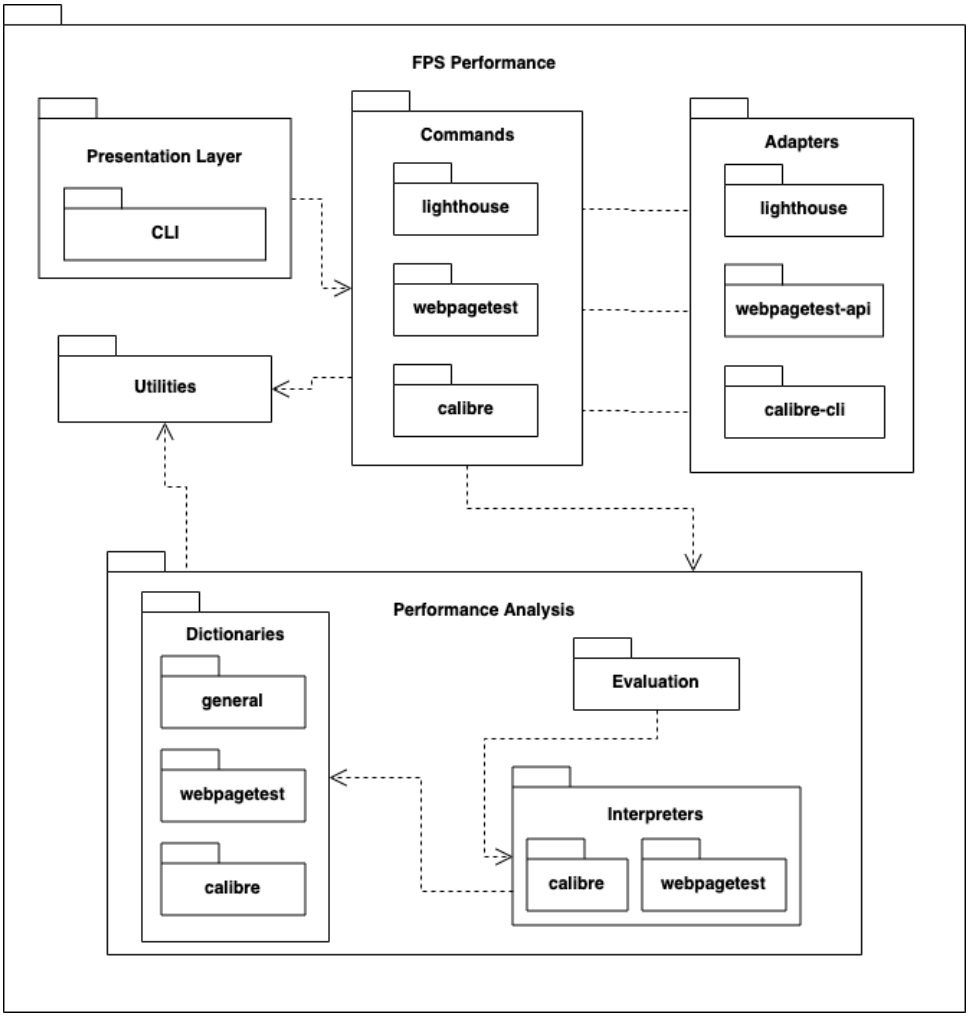


Figura 19. Diagrama de *packages* de *FPS Performance*, do nível de contentores do modelo C4

Quando comparado o diagrama Figura 19, que pretende dar a conhecer as várias partes de um *software* na perspetiva de um desenvolvedor, com o diagrama apresentado na Figura 18, verifica-se o maior detalhe relativo:

- Aos comandos, *adapters*, dicionários e interpretadores disponibilizados, tendo cada ferramenta a sua implementação associada, dado o grau de detalhe de implementação específico que cada ferramenta tem e disponibiliza para ser utilizada (de notar que o *Google Lighthouse* não tem nenhum *interpreter* considerando o facto de quer o *Calibre* como o *WebPageTest* já disponibilizar dados de avaliação sobre o *lighthouse* em cada uma das ferramentas;

- A *Presentation Layer*, ao qual contém o detalhe associado sobre a camada relacionada com a apresentação e interpretação de dados por parte dos utilizadores antes de serem transferidas as informações para o respetivo comando solicitado;

- Uma secção de *Utilities*, que tal como o nome indica, representa recursos úteis partilhados entre as várias partes constituintes da aplicação.

3.3.3 **Vista de processo**

Esta vista de processo, apresentada na Figura 20 tem como objetivo apresentar as interações entre os vários componentes/*packages* associados ao contentor *FPS Performance*, de modo a poder visualizar a sequência de ações possíveis de serem executadas no momento da execução de avaliação de desempenho de um *website.*

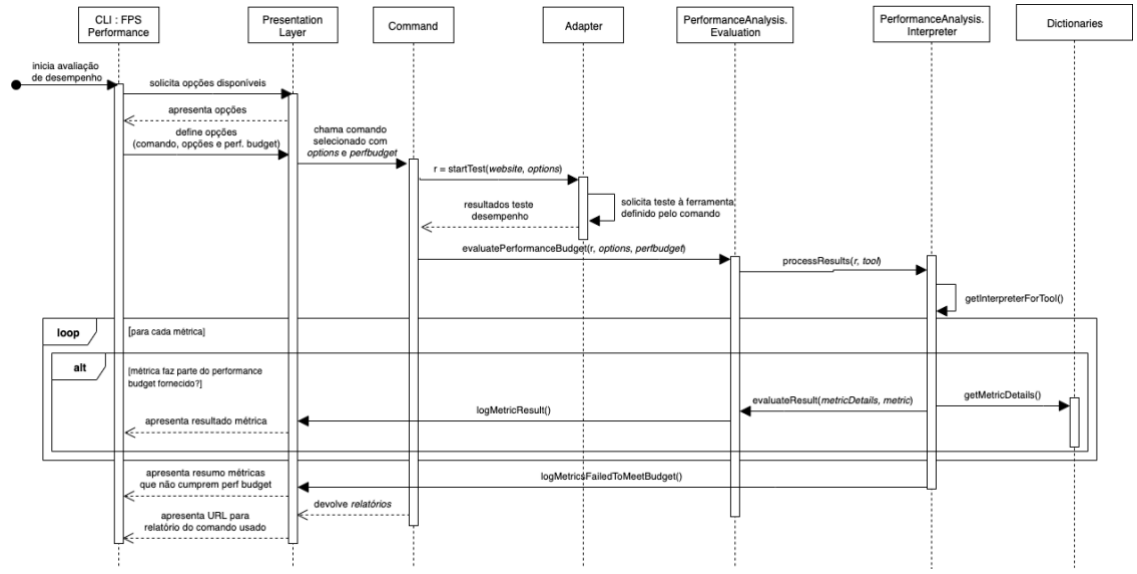


Figura 20. Diagrama de sequência das interações do contentor *FPS Performance*, nível de componentes do modelo C4

Considerando o início desta aplicação para executar a avaliação do desempenho de um *website*, de acordo com o diagrama, o fluxo executado é o seguinte:

1. O sistema apresenta as opções disponíveis e solicita o fornecimento de dados necessários para executar a avaliação de desempenho (comando, opções, *website* e o *performance budget*);

2. De acordo com o tipo de comando inserido, a aplicação é reencaminhada para o comando respetivo, de modo a poder inicializar o teste do *website* definido com as opções especificadas;

3. O comando chama o *Adapter* respetivo relativo à ferramenta selecionada para o teste, de modo a poder comunicar com a respetiva API e aguarda pela resposta contendo o resultado com os respetivos valores das métricas medidas;

4. O comando, com os resultados, inicia o processo de análise de *performance*, ao qual envia os resultados, o *performance budget* associado para o componente responsável por executar a avaliação.

5. Tendo em conta a especificidade dos objetos recebidos, os dados passam por um Interpretador associado à ferramenta escolhida, em que este é responsável por percorrer todas as métricas recebidas na resposta e:

    5.1. Para cada métrica a ser analisada, são solicitados os seus dados ao dicionário, de forma a ser reconhecida e interpretada corretamente pelo sistema;

    5.2. Após receber os dados do dicionário, verifica se a métrica consta na lista de métricas que definem o *performance budget*

        5.2.1. Em caso afirmativo, o valor do *budget* e o valor da métrica, mais os respetivos dados associados à métrica (nome e unidade) são enviados para o componente de avaliação, responsável por verificar e informar se a métrica cumpre ou não o *performance budget*, de acordo com as especificações fornecidas para o tipo de avaliação a executar.

6. Uma vez analisadas as métricas, a aplicação, no caso de existir métricas que não tenham cumprido o *performance budget*, é apresentada uma listagem no final com as respetivas métricas, informando do sucesso ou insucesso da avaliação realizada.

7. É disponibilizado uma hiperligação de acesso para o relatório do teste gerado pela ferramenta utilizada, de modo a ser possível verificar informação mais detalhada relativamente ao estado real do *website* enquanto se encontrava sobre avaliação.

3.4 Sumário

Com base no detalhe apresentado nesta secção, é possível visualizar com diverso tipo de granularidade o impacto da solução desde do sistema em que está integrada, até ao seu funcionamento isolado, graças ao poder de sistematização e abstração que o modelo C4 proporciona, tendo assim uma clara visão das respetivas interações e cuidados necessários a adotar para o desenvolvimento do *software* com o respetivo design expresso sob a forma de diagramas UML.

4 Implementação

O presente capítulo visa sintetizar todos os aspetos inerentes à construção e desenvolvimento do projeto, bem como apresenta recomendações a adotar para se desenvolver um *website* que privilegie um bom desempenho – a começar no seu desenvolvimento até à sua implantação para o público-geral.

Deste modo, os objetivos deste capítulo são:

- Apresentar o protótipo desenvolvido para medir e monitorizar o desempenho e respetivas funcionalidades;
- Descrever o modo como o *performance budget* (cf. secção 2.3) foi definido para a unidade de negócio.
- Propor como este protótipo pode ser integrado nos processos existentes na organização.

4.1 Protótipo

De acordo com a descrição arquitetural apresentada na secção 3, o protótipo será construído/implementado em *Node.js* (cf. secção 3.1.4), o que integra as ferramentas de medição de desempenho já descritas (cf. secção 2.8) - *WebPageTest*, *Calibre* e *Google Lighthouse*.

A aplicação e respetiva integração destas ferramentas permite:

- Utilizar ferramentas disponíveis para proceder à execução de testes de desempenho com determinadas especificações:
    - Definir uma localização geográfica específica para executar o teste;

    - Definir o tipo de dispositivo e *browser* responsável por aceder ao *website* para efetuar a avaliação de desempenho (e.g. *smartphone*, *tablet*);

    - Definir o número de acessos a cada *website*, de forma a obter dados mais precisos e ignorar *outliers*[36].

- Avaliar e expor os resultados obtidos, verificando sempre se cumpre os requisitos mínimos definido pelo *performance budget* definido (cf. secção 2.3)*;*

- Dar visibilidade aos relatórios específicos gerados pelas ferramentas utilizadas.

Uma interface CLI é caracterizada pela interação ser feita com recurso a texto, de modo a executar operações nos sistemas operativos e aplicações. Uma aplicação baseada neste tipo de interface fornece vários comandos, ou seja, instruções possíveis de serem realizadas numa dada aplicação. Esses comandos podem aceitar parâmetros adicionais, parâmetros esses que permitem modificar o modo de operação de um dado comando, de acordo com as especificações fornecidas. Esta aplicação, baseada numa interface CLI, pode ser utilizada isoladamente, ou pode ser integrada em mecanismos de *Continuous Integration/Continuous Delivery* disponíveis na organização (cf. secção A.2).

Para criar esta ferramenta foi utilizada a biblioteca *Yargs* (cf. secção 3.1.4) que possibilitou o desenvolvimento do tipo de interação descrita anteriormente, além de gerar menus de ajuda com base nos parâmetros suportados, possibilitando assim a validação as opções fornecidas com base nos valores autorizados/suportados pela aplicação, assim como respetivas indicações para uma correta utilização dos vários comandos.

4.1.1 Comandos disponíveis

A aplicação disponibiliza três comandos que permite o acesso para proceder à avaliação de desempenho em determinados contextos, sendo eles:

- `webpagetest`**:** permite executar testes de desempenho a um *website* recorrendo à ferramenta *WebPageTest* (cf. secção 2.8.1) e verificar se os resultados obtidos cumprem o *performance budget* configurado e fornecido;

- `calibre:` permite executar testes de desempenho a um website recorrendo à ferramenta *Calibre* (cf. secção 2.8.2) e verificar se os resultados obtidos cumprem o *performance budget* configurado e fornecido;

- `lighthouse:` permite executar uma avaliação automatizada, segundo os parâmetros da ferramenta *Google Lighthouse* (cf. secção 2.8.3), fornecendo na sua conclusão, um ficheiro com os resultados obtidos no formato JSON.

[36] Representação do valor externo/aberrante, termo proveniente da área da estatística.

Nas seguintes secções serão apresentados cada comando em maior detalhe, os parâmetros suportados, valores por defeito e respetiva sintaxe. Note-se no seguinte formato, utilizado na apresentação de um comando:

```
fps-performance <nome comando> <url> [parâmetros]
```

- `fps-performance` refere-se ao nome da aplicação, sendo obrigatório fornecer de forma ao sistema operativo reconhecer qual a aplicação iniciar no sistema;
- `< ... >`, refere-se a uma entrada que deve ser fornecida obrigatoriamente num comando. No exemplo, deve ser fornecido o nome do comando (`webpagetest`, `calibre` ou `lighthouse`) e o URL do *website* a testar;
- `[ ... ]` refere-se ao local onde podem ser inseridos os parâmetros, sendo de entrada opcional.

4.1.1.1 *Comando webpagetest*

O comando `webpagetest` permite executar testes de desempenho a um dado URL, utilizando os serviços disponibilizados pela ferramenta *WebPageTest* (cf. secção 2.8.1). Uma vez obtidos os resultados do teste, é verificado se os mesmos cumprem o *performance budget* definido (cf. secção 4.2).

Tendo em consideração a flexibilidade do *WebPageTest*, este disponibiliza instâncias de acesso público e privado, sendo que as últimas permitem executar testes de desempenho em URLs privados - normalmente acessíveis a partir de uma rede Intranet[37], cenário que se observa em ambientes de desenvolvimento de aplicações a nível organizacional - além dos URL públicos, ao contrário do *Calibre* (cf. secção 2.8.2).

No final de cada avaliação de desempenho cujos resultados tenham sido obtidos com sucesso, este disponibiliza um URL para que se possa consultar o relatório gerado pela ferramenta (cf. secção 4.1.4.1).

Este comando segue o seguinte formato:

```
fps-performance webpagetest <url> [parâmetros]
```

De forma a este comando poder responder a diversas necessidades de utilização, este disponibiliza um conjunto de parâmetros a serem fornecidos, sendo eles:

[37] Rede de privada utilizada para acesso e partilha de informação e recursos computacionais por utilizadores autorizados. Tipicamente utilizada em ambientes empresariais.

- `--user:` nome de utilizador necessário para aceder a um URL que se encontra protegido por autenticação HTTP (normalmente necessário para aceder a websites em ambientes de pré-produção).

- `--password:` palavra-passe necessária para aceder a um URL que se encontra protegido por autenticação HTTP. A ser utilizada em conjunto com o parâmetro --user, conforme o exemplo demonstra:

```
fps-performance webpagetest <url> --user <utilizador> --password <password>
```

- `--usePrivateInstance`: esta opção indica que os testes de desempenho serão executados numa instância privada do *WebPageTest*, iniciada via imagem *Docker* (cf. secção 2.8.1). Esta opção é obrigatória para poder executar testes de desempenho em *websites* privados dentro de uma rede. Caso esta opção não seja especificada, será utilizada a instância pública disponível do *WebPageTest*.

```
fps-performance webpagetest <url> --usePrivateInstance
```

- `--numberOfRuns:` permite especificar qual o número de *test runs*[38] que devem ser executados ao URL especificado, de modo a permitir resultados mais precisos. Por defeito, o número de testes a executar está definido para três, e é possível executar até nove testes numa única avaliação de desempenho de um *website*.

```
fps-performance webpagetest <url> --numberOfRuns <number>
```

- `--runTypes:` permite especificar o tipo de testes a executar na avaliação de desempenho de um *website*, podendo ser do tipo:

    - `firstView:` um teste que é executado com um *browser* com as *cookies* e a *cache* limpas, que representa o que um novo utilizador experiencia quando visita um dado *website* pela primeira vez:

    ```
    fps-performance webpagetest <url> --runTypes firstView
    ```

    - `repeatView:` um teste executado imediatamente a seguir ao `firstView`, sem recorrer à limpeza de *cookies*, cache ou outros dados de navegação. A janela do *browser* é fechada após o teste do tipo *First View* e é aberto novamente o browser para executar o teste do tipo *Repeat View*:

    ```
    fps-performance webpagetest <url> --runTypes repeatView
    ```

[38] Acesso individual ao *website* em que são recolhidas métricas.

- `--location:` permite especificar a localização geográfica onde é executado o teste de desempenho, de acordo com um conjunto de localizações suportadas pela instância pública disponível do *WebPageTest*. A localização por defeito é `Dulles` (Dulles, Virginia, Estados Unidos da América), sendo esta a localização que o *WebPageTest* apresenta inicialmente e que mais dispositivos fornece para teste. A localização no caso do *WebPageTest* surge associada a um dispositivo, pelo que a mesma localização pode ter vários dispositivos, como é o caso de *Dulles* que suporta vários dispositivos como é o caso dos *smartphones Motorola Moto G4* ou *Moto G*, opção especificada como `Dulles_MotoG4` ou `Dulles_MotoG`. Esta opção não está disponível quando o argumento `--usePrivateInstance` é fornecido, dado a localização desta instância privada ser obrigatoriamente a localização atual. Uma listagem das localizações suportadas é apresentada na Tabela 3.

```
fps-performance webpagetest <url> --location <localização>
```

Tabela 3. Localizações suportadas pela instância pública do *WebPageTest*

Chave Localização	**Localização e dispositivo associado**
`Dulles`	Dulles, Virginia, USA
`London_EC2`	Londres, Reino Unido
`ec2-us-east-1`	Virginia, Estados Unidos
`ec2-us-west-1`	California, Estados Unidos
`ec2-sa-east-1`	São Paulo, Brasil
`ec2-eu-west-1`	Irlanda
`ec2-eu-west-3`	Paris, França
`ec2-eu-central-1`	Frankfurt, Alemanha
`ap-south-1`	Mumbai, Índia
`ec2-ap-southeast-1`	Singapura
`ec2-ap-northeast-2`	Seoul, Coreia do Sul
`ec2-ap-northeast-1`	Tóquio, Japão
`ec2-ap-southeast-2`	Sydney, Austrália
`Dulles_MotoG4`	Dulles, Virginia, Estados Unidos (dispositivo Motorola Moto G4)
`Dulles_MotoG`	Dulles, Virginia, Estados Unidos (dispositivo Motorola Moto G)
`Dulles_Edge`	Dulles, Virginia, Estados Unidos (dispositivo com *browser Microsoft Edge*)
`Dulles_IE11`	Dulles, Virginia, Estados Unidos (dispositivo com *browser Internet Exolorer*)

Chave Localização	Localização e dispositivo associado
`Dulles_Thinkpad`	Dulles, Virginia, Estados Unidos (dispositivo *Lenovo Thinkpad T430*)
`Dulles_Pi3`	Dulles, Virginia, Estados Unidos (dispositivo *Raspberry Pi 3*)
`Dulles_Pi4`	Dulles, Virginia, USA (dispositivo *Raspberry Pi 4*)

- `--device:` permite especificar o *browser* onde executar o teste de desempenho, a correr num determinado dispositivo, de acordo com um conjunto de *browsers* suportados pelo *WebPageTest* para uma dada localização selecionada. O *browser* por defeito é o primeiro da lista dos disponíveis na localização selecionada, sendo normalmente o *browser Google Chrome* o selecionado. De frisar que para se obter resultados da avaliação do *Google Lighthouse* (cf. secção 2.8.3) por intermédio da ferramenta *WebPageTest*, é necessário que o browser seja o *Google Chrome*, ou outro browser baseado na tecnologia do mesmo (e.g. *Google Chrome Beta*, *Google Chrome Canary*, *Chromium*).

```
fps-performance webpagetest <url> --device <dispositivo>
```

- `--perfbudget:` permite indicar o caminho para um ficheiro no formato JSON responsável por conter a configuração necessária para se proceder à avaliação do *performance budget* para os resultados. O ficheiro de configuração lido por defeito é o `perfbudget-config.json` que se encontra na localização em que a aplicação foi iniciada.

```
fps-performance webpagetest <url> --perfbudget <ficheiro-de-configuração>
```

4.1.1.2 *Comando calibre*

O comando `calibre` permite executar um teste de desempenho a um determinado URL, utilizando os serviços disponibilizados pela ferramenta *Calibre* (cf. secção 2.8.2), em que uma vez obtidos os resultados do teste, verifica se os mesmos cumprem o *performance budget* (cf. secção 2.3) definido.

Tendo em conta que o *Calibre* é um serviço externo fechado, esta ferramenta apenas permite executar testes e obter resultados que estão disponíveis publicamente na *Internet*.

No final de cada avaliação de desempenho cujos resultados tenham sido obtidos com sucesso, este disponibiliza um URL para que se possa consultar o relatório gerado pela ferramenta (cf. secção 4.1.4.2).

Este comando segue o seguinte formato:

```
fps-performance calibre <url> [parâmetros]
```

De forma a este comando poder responder a diversas necessidades de utilização, este disponibiliza um conjunto de parâmetros a serem fornecidos, sendo eles:

- ***--user*:** nome de utilizador necessário para aceder a um URL que se encontra protegido por autenticação HTTP (normalmente necessário para aceder a *websites* em ambientes de pré-produção.

- ***--password*:** palavra-passe necessária para aceder a um URL que se encontra protegido por autenticação HTTP. A ser utilizada em conjunto com o parâmetro *–user*, conforme o exemplo demonstra:

  ```
  fps-performance calibre <url> --user <utilizador> --password <password>
  ```

- ***--device*:** permite especificar o dispositivo onde se pretende executar o teste de desempenho, de acordo com um conjunto de dispositivos suportados pelo *Calibre*. Neste momento, o *Calibre* só suporta a especificação de dispositivos móveis, e caso não seja fornecido nenhum, o dispositivo por defeito utilizado é o *browser Google Chrome* a correr num computador *desktop*[39].

  ```
  fps-performance calibre <url> --device <dispositivo>
  ```

- ***--location*:** permite especificar a localização geográfica onde é executado o teste de desempenho, de acordo com um conjunto de localizações suportadas pelo *Calibre*. A localização por defeito é *Frankfurt* (Frankfurt, Alemanha), por ser a localização já utilizada neste serviço para monitorizar a maioria dos *websites* já desenvolvidos por FPS.

  ```
  fps-performance calibre <url> --location <localização>
  ```

- ***--perfbudget*:** permite indicar o caminho para um ficheiro no formato JSON responsável por conter a configuração necessária para se proceder à avaliação do *performance budget* para os resultados. O ficheiro de configuração lido por defeito é o `perfbudget-config.json` que se encontra na localização em que a aplicação foi iniciada.

[39] Computador de secretária.

```
fps-performance calibre <url> --perfbudget <ficheiro-de-configuração>
```

4.1.1.3 *Comando lighthouse*

O comando *lighthouse* permite efetuar uma avaliação individual de um *website*, recorrendo às potencialidades fornecidas pela ferramenta *Google Lighthouse* (cf. secção 2.8.3), por via de uma auditoria executada de forma automática, recorrendo a uma instância própria do *browser* Google *Chrome*, necessário para a análise.

A sintaxe deste comando segue o seguinte formato:

```
fps-performance lighthouse <url>
```

Este comando só necessita da informação do URL a analisar, podendo ser ele público ou privado, dependendo do tipo de ligação à rede e acessos disponíveis no dispositivo que iniciou a aplicação.

O relatório produzido após a execução deste comando fica disponível no ficheiro `lighthouse-report.json`, de forma a poder ser lido e reutilizado por terceiros, mas que é possível visualizar o relatório associado nesta página - https://googlechrome.github.io/lighthouse/viewer/ - bastando para tal importar o ficheiro gerado.

Este comando foi criado por motivos experimentais, de forma a ser possível comparar o tipo de informação fornecida pela biblioteca *Node.js* do módulo *Lighthouse* fornecido pela Google (Google, 2019d) face aos resultados do *Lighthouse* pelos resultados já devolvidos do *WebPageTest* e *Calibre*.

4.1.2 **Variáveis de ambiente**

Para permitir uma configuração genérica, permitindo assim o desacoplamento de detalhes relacionados com os serviços que o protótipo deve utilizar, houve a necessidade de definir variáveis de ambiente.

Uma variável de ambiente no contexto de uma aplicação *Node.js* trata-se de um valor, associado a uma variável, caracterizada pelo seu nome ser definido em letras maiúsculas, cujo valor é injetado na variável global `process.global` no arranque de uma aplicação e representa o estado do sistema quando a aplicação é iniciada (Goh, 2018).

As variáveis de ambiente definidas para o protótipo são:

- `CALIBRE_API_TOKEN`: variável responsável por conter a chave de autenticação (*token*) para o módulo de *Node.js* poder comunicar com a API do *Calibre* com a conta criada para a organização.

- `WEBPAGETEST_DEFAULT_SERVER`: variável que contém o URL para a instância pública do *WebPageTest*, definida pelo endereço https://www.webpagetest.org. Pode conter uma outra instância do *WebPageTest*, sendo para tal necessário definir o respetivo *token* de autorização para comunicação com a API da instância pretendida.

- `WEBPAGETEST_API_TOKEN`: variável responsável por conter a chave de autenticação para comunicar com a API da instância pública (ou outra, de acordo com o URL definido na variável `WEBPAGETEST_DEFAULT_SERVER`). É possível solicitar um *token* para a instância pública do *WebPageTest* pelo endereço https://www.webpagetest.org/getkey.php;

- `WEBPAGETEST_DOCKER`: variável responsável por fornecer o endereço e a porta para a instância local do *WebPageTest* que será inicializada via *Docker*. O endereço do servidor (conforme especificado no ficheiro *docker-compose.yml*) é o https://127.0.0.1:4000, onde o endereço de IP 127.0.0.1 representa a própria máquina, e o valor 4000 a porta utilizada.

4.1.3 Ficheiro de configuração

Nos comandos `calibre` e `webpagetest` existe um parâmetro comum mas necessário a ambos – o `--perfbudget`.

Este parâmetro é essencial porque permite especificar um ficheiro de configuração responsável por conter toda a informação necessária para o protótipo proceder à avaliação se o *performance budget* (cf. secção 2.3) é cumprido ou não.

Assim, este ficheiro de configuração é responsável por:

- Especificar os limites (máximos ou mínimos, dependendo da escala) para as várias métricas medidas (cf. secção 2.4) pelas aplicações integradas:
    - De forma genérica, ou seja, métricas transversais à ferramenta *Calibre* e *WebPageTest*;
    - Métricas específicas à ferramenta *Calibre*;
    - Métricas específicas à ferramenta *WebPageTest*;
- Permitir configurar o modo como a avaliação é feita:
    - Ao definir um valor de tolerância percentual, em que o valor da métrica medida pode desviar do valor do *budget*;
    - Ao definir aspetos específicos relativos aos resultados a avaliar que foram produzidos por um teste realizado com o *WebPageTest*, como é o caso.

Este ficheiro tem de ser criado com o formato JSON e segue a seguinte sintaxe:

```
{
      "metrics": {
       "general": {
            "firstCPUIdle": 1000,
            "firstContentfulPaint": 1300,
            "requests": 50,
            "speedindex": 3000,
            "lighthouse": {
                 "accessibility": 80,
                 "best-practices": 80,
                 "seo": 80,
                 "performance": 80,
                 "pwa": 80
             }
       },

      "calibre": {
            "first-meaningful-paint": 3000,
            "first-contentful-paint": 3000,
            "consistently-interactive": 3000,
      },

      "webpagetest": {
            "firstContentfulPaint": 1300,
            "SpeedIndex": 3000
      },

      "settings": {
            "general": {
                  "tolerance": 5
             },

             "webpagetest": {
                  "viewToEvaluate": "repeatView",
                  "resultsToEvaluate": "median",
                  "medianMetric": "loadTime"
             }
      }
}
```

Na configuração geral, acessível na chave `metrics.general` do exemplo JSON anterior, é possível descrever as métricas que são recolhidas por ambas as ferramentas, e deste modo, permite garantir que o *budget* entre as ferramentas é comum e partilhado. Estas métricas determinam os valores que um *website* alvo de análise pode obter, sendo eles em grande parte expressos ora em milissegundos ora em *bytes*.

Na secção `metrics.general` do ficheiro de configuração, são suportadas as seguintes métricas:

Tabela 4. Mapeamento das métricas gerais suportadas pelo ficheiro de configuração

Chave da métrica	**Métrica**
`firstContentfulPaint`	First Contenful Paint (em milissegundos)

`firstCPUIdle`	*First CPU Idle*, também denominada como *First Interactive* (em milissegundos)
`firstMeaningfulPaint`	First Meaningful Paint (em milissegundos)
`firstPaint`	*First Paint* (em milissegundos)
`loadTime`	Tempo de carregamento (em milissegundos)
`speedindex`	*SpeedIndex* (em milissegundos)
`timeToFirstByte`	Time to First Byte (em milissegundos)
`timeToInteractive`	*Time to Interactive* (em milissegundos)
`requests`	Número de pedidos
`visuallyComplete85`	85% Visually Complete (em milissegundos)
`visuallyComplete`	*Visually Complete* (em milissegundos)

A configuração de valores para o *Google Lighthouse* segue um diferente formato, dado este produzir após a sua execução uma pontuação para cada categoria: Acessibilidade (*Accessibility*), Boas Práticas *(Best Practices*), otimização para motores de pesquisa (SEO) e PWA.

Estes valores são descritos sob a forma de valor percentual e determinam a pontuação mínima a obter em cada categoria, sendo expressos sob a forma de uma nova chave dentro da secção `metrics.general` com o nome `lighthouse` e que, associado a esta nova chave, possui um objeto que permite mapear a cada categoria a percentagem mínima possível de se obter, conforme demonstra o seguinte exemplo:

```
{
      "metrics": {
        "general": {
              "lighthouse": {
                    "accessibility": 80,
                    "best-practices": 80,
                    "seo": 80,
                    "performance": 80,
                    "pwa": 80
              }
        }
      }
}
```

Tal como foi mencionado, para além de poder especificar métricas gerais, é possível especificar métricas relativas a cada uma das ferramentas utilizadas, dando assim a possibilidade de:

- Complementar a avaliação com métricas específicas que só esteja disponível numa das ferramentas utilizadas;

- Estabelecer diferentes *budgets* dependendo da localização e dispositivo que cada ferramenta disponibilizada;

- Reescrever valores atribuídos a métricas gerais, sendo o valor especificado na configuração específica da ferramenta o utilizado (podendo assim criar exceções ou restrições para uma dada localização e/ou dispositivo disponível numa ferramenta para um conjunto de avaliações).

As métricas específicas para a ferramenta *Calibre* são descritas na secção `metrics.calibre` e é possível especificar todas chaves relativas às métricas devolvidas por um relatório produzido por esta ferramenta. As chaves suportadas neste momento são:

Tabela 5. Métricas devolvidas pela ferramenta *Calibre*

Chave métrica *Calibre*	**Descrição Métrica**
`asset_count`	Número de pedidos
`consistently-interactive`	*Time to Interactive* (em milissegundos)
`css_body_size_in_bytes`	Tamanho total de conteúdo CSS no *website* (em *bytes*)
`css_size_in_bytes`	Tamanho total de conteúdo CSS transferidos (em *bytes*)
`first-contentful-paint`	*First Contentful Paint* (em milissegundos)
`first-interactive`	*First CPU Idle* (em milissegundos)
`first-meaningful-paint`	First Meaningful Paint (em milissegundos)
`firstRender`	*First Paint* (em millissegundos)
`font_body_size_in_bytes`	Tamanho total de fontes *Web* no *website* (em *bytes*)
`font_size_in_bytes`	Tamanho total de fontes *Web* transferidas (em *bytes*)
`html_body_size_in_bytes`	Tamanho total de conteúdo HTML no *website* (em *bytes*)
`html_size_in_bytes`	Tamanho total de conteúdo HTML transferido (em *bytes*)
`image_body_size_in_bytes`	Tamanho total de imagens no *website* (em *bytes*)
`image_size_in_bytes`	Tamanho total de imagens transferidas (em *bytes*)
`js_body_size_in_bytes`	Tamanho total de conteúdo *JavaScript* no *website* (em *bytes*)

Chave métrica *Calibre*	**Descrição Métrica**
`js_size_in_bytes`	Tamanho total de conteúdo *JavaScript* transferido (em *bytes*)
`js-parse-compile`	Tempo despendido em compilação e interpretação de conteúdo *JavaScript* (em milissegundos)
`json_body_size_in_bytes`	Tamanho total de conteúdo JSON no *website* (em *bytes*)
`json_size_in_bytes`	Tamanho total de conteúdo JSON transferido (em *bytes*)
`lighthouse-accessibility-score`	Pontuação do *Google Lighthouse* – categoria de Acessibilidade (em percentagem)
`lighthouse-best-practices-score`	Pontuação do *Google Lighthouse* – categoria de Boas Práticas (em percentagem)
`lighthouse-performance-score`	Pontuação do *Google Lighthouse* – categoria de *Performance* (em percentagem)
`lighthouse-pwa-score`	Pontuação do *Google Lighthouse* – categoria de *Progressive Web App* (em percentagem)
`lighthouse-seo-score`	Pontuação do *Google Lighthouse* – categoria de *SEO* (em percentagem)
`oncontentload`	Tempo até ao documento completo da página estar carregado - evento DOM *ready* do *browser* (em milissegundos)
`onload`	Tempo para página estar totalmente carregada - conteúdo e recursos descarregados (em milissegundos)
`page_body_size_in_bytes`	Tamanho da página do *website* carregada (em *bytes*)
`page_size_in_bytes`	Tamanho da página transferida (em *bytes*)
`page_wait_timing`	Tempo de resposta da página (em milissegundos)
`speed_index`	*SpeedIndex* (em milissegundos)
`time-to-first-byte`	*Time to First Byte* (em milissegundos)
`visually_complete`	*Visually Complete* (em milissegundos)
`visually_complete_85`	85% Visually Complete (em milissegundos)

As métricas específicas para a ferramenta *WebPageTest* são descritas na secção `metrics.webpagetest` e é possível especificar todas chaves relativas às métricas devolvidas por um relatório produzido por esta ferramenta. As chaves mais relevantes são:

Tabela 6. Métricas mais relevantes devolvidas pela ferramenta *WebPageTest*

Chave métrica *WebPageTest*	**Descrição**
`bytesIn`	Tamanho dos dados transferidos até à página estar no estado *fully loaded*[40] (em *bytes*)
`bytesInDoc`	Tamanho dos dados transferidos até à página estar no estado *Document Complete*[41] (em *bytes*)
`docTime`	Tempo até à página estar no estado *Document Complete* (em milissegundos)
`firstContentfulPaint`	First Contentful Paint (em milissegundos)
`FirstCPUIdle`	*First CPU Idle* (em milissegundos)
`firstMeaningfulPaint`	First Meaningful Paint (em milissegundos)
`firstPaint`	*First Paint* (em milissegundos)
`fullyLoaded`	Tempo até à página estar num estado *fully loaded* (em milissegundos)
`loadTime`	Tempo entre o início da navegação até ao início do evento de carregamento da página pelo *browser* (em milissegundos)
`render`	Tempo entre o início da navegação até surgir o primeiro conteúdo no ecrã (em milissegundos)
`requestsDoc`	Número de pedidos até ocorrer o evento *DOMContentLoaded* do *browser*
`requestsFull`	Número de pedidos total
`SpeedIndex`	*Speed Index* (em milissegundos)
`TimeToInteractive`	*Time to Interactive* (em milissegundos)
`TTFB`	*Time to First Byte* (em milissegundos)
`visualComplete`	*Visually Complete* (em milissegundos)

[40] Estado *fullyLoaded*: estado interpretado pelo *WebPageTest* responsável por monitorizar o que acontece entre o início da navegação até ao momento em que não ocorre nenhuma atividade de rede até uma página estar completamente carregada – *Document Complete* (conteúdo HTML e árvore DOM carregado, mas conteúdo média como imagens pode não estar totalmente disponível).

[41] Momento em que o evento *onLoad* do *browser* dispara, representando o momento em que todo o conteúdo estático da página está carregado.

Chave métrica *WebPageTest*	Descrição
`visualComplete85`	85% Visually Complete (em milissegundos)
`visualComplete90`	90% Visually Complete (em milissegundos)
`visualComplete95`	95% Visually Complete (em milissegundos)
`visualComplete99`	99% Visually Complete (em milissegundos)

De notar que, devido à estrutura da resposta devolvida pelo *WebPageTest*, o resultado do *Google Lighthouse* produzido por esta ferramenta não é interpretado como uma métrica, ao contrário do *Calibre*, mas sim como um resultado opcional devido à estrutura como esta ferramenta foi criada (semelhante ao motivo pelo qual os resultados do *Google Lighthouse* só são disponibilizados quando o dispositivo/*browser* selecionado é baseado em *Google Chrome*). Por este motivo, não é possível reescrever as pontuações mínimas que as categorias do *Google Lighthouse* podem obter durante a avaliação de um *website* ao recorrer ao *WebPageTest*, sendo para tal obrigatório recorrer à configuração `metrics.general.lighthouse` para esse efeito.

O ficheiro de configuração adiciona uma secção criada pela chave `settings` no ficheiro JSON, responsável por introduzir configurações específicas relativas ao modo como os resultados são interpretados.

A configuração geral suportada para ambas as ferramentas neste momento é a `tolerance`. Este valor representa uma percentagem em que representa o valor máximo com que uma métrica medida durante a avaliação pode desviar do valor definido na configuração. Este desvio é identificável no resultado final sob a forma de aviso, em que indica que apesar de exceder o *budget* definido, o valor encontra-se na tolerância do *budget* dessa métrica. Esta opção serve para ajudar a identificar futuros problemas de desempenho num *website*, mas permitindo que haja um pequeno intervalo em não force um erro e falha de comprimento do *performance budget*. Caso esta opção não seja fornecida, não existe qualquer tolerância aplicada, ou seja, 0% de tolerância.

No caso de métricas cuja unidade é definida como sendo em milissegundos ou em *bytes*, a tolerância permite que o resultado medido seja superior ao *budget* definido. Tome-se por exemplo uma tolerância de 5% para a métrica *SpeedIndex*, cujo *budget* são 2500 ms. O valor máximo que se pode obter desta métrica, para que se encontre dentro da tolerância é de 2625 ms (2500 ms do *budget* da métrica *SpeedIndex* mais o valor da tolerância para o *budget* dessa métrica – 2500 x 5% = 625ms, que perfaz o valor de 2625 ms). Qualquer resultado que seja superior a 2500 ms e igual ou inferior a 2625 ms é identificado como um aviso, informando que excede o *budget*, mas que se encontra dentro da tolerância definida.

Para as métricas cuja unidade é definida como sendo percentual, como o caso das pontuações das categorias do *Lighthouse*, a tolerância permite que o resultado medido seja inferior ao

budget definido. Tome-se por exemplo uma tolerância de 5% para a categoria de Acessibilidade do *Google Lighthouse*, cujo *budget* é 80%. O valor máximo que se pode obter desta métrica, para que se encontre dentro da tolerância é de 76% (80% do *budget* da métrica Acessibilidade do *Google Lighthouse* menos o valor da tolerância para o *budget* dessa métrica – 80 x 5% = 4%, que perfaz o valor de 76%). Qualquer resultado que seja igual ou superior a 76% e inferior a 80% é identificado como um aviso, informando que excede o *budget*, mas que se encontra dentro da tolerância definida.

Esta opção é definida do seguinte modo:

```
{
      "settings": {
              "general": {
                    "tolerance": 5
              },
      },
}
```

Na Figura 21, apresenta uma captura de ecrã com o extrato da avaliação de resultados obtidos na execução de um teste de desempenho com o protótipo. Em cada métrica avaliada, além de apresentar o resultado obtido, apresenta sempre uma descrição textual da métrica e respectiva unidade. Assim, é possível verificar-se como os vários resultados possíveis de se obter são apresentados, sendo eles:

- *Pass*, ou seja, uma métrica cujo valor obtido é inferior ao valor do *budget* definido, representado com o texto com a cor verde;
- *Warning*, uma métrica cujo valor obtido é superior ao valor do *budget* definido, mas que o valor obtido se encontra na tolerância aceite pelo *budget*, representado pela cor amarela, e com a informação da percentagem de tolerância definida;
- *Fail*, uma métrica cujo valor obtido excede quer o valor definido pelo *budget* como também a tolerância aceite pelo *budget*, representado com a cor vermelha.

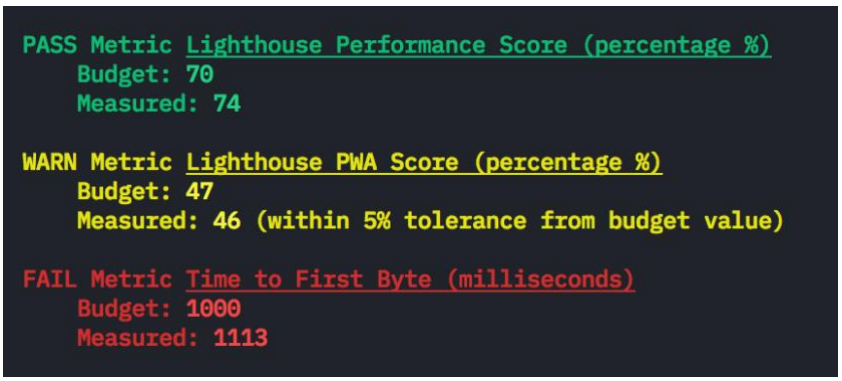


Figura 21. Exemplo de apresentação de uma avaliação de *performance budget*

Relativamente a configurações específicas, a ferramenta que necessita/suporta configurações adicionais é o *WebPageTest*, devido à natureza dos múltiplos resultados obtidos, em que estas opções permitem selecionar que resultados dos testes realizados fazem mais sentido serem alvos de verificações de cumprimento do *performance budget* do *website*.

As opções disponíveis são:

- `viewToEvaluate:` os resultados de vista devem ser utilizados. Os valores aceites são:
    - `firstView:` resultados medidos na primeira visita a um *website*.
    - `repeatView:` resultados medidos num retorno a um *website* previamente visitado (contém *browser cookies* e *cache* do acesso anterior). Esta é a vista utilizada por defeito para os resultados.
    - `average:` média dos resultados obtidos no modo `firstView` e `repeatView`.
- `resultsToEvaluate:` que tipo de resultados medidos devem ser utilizados, sendo suportados os valores:
    - `median:` resultados do teste que possui o valor mediano para a métrica selecionada como a métrica mediana a ser avaliada – `medianMetric`. Estes são os resultados utilizados por defeito.
    - `average:` resultados médios das várias execuções de testes realizadas.
- `medianMetric:` representa qual a métrica a ser utilizada para identificar o valor de teste mediano, caso tenham sido executados vários testes num *website*. Esta opção apenas é necessária quando a opção definida para `resultsToEvaluate` é o valor `median`. O valor por defeito é a métrica *loadTime*, e neste campo é suportada qualquer nome de métrica reconhecida pelo *WebPageTest*.

A sintaxe para configurações específicas para a interpretação de resultados do *WebPageTest* segue o seguinte formato:

```
{
      "settings": {
              "webpagetest": {
                    "viewToEvaluate": "repeatView",
                    "resultsToEvaluate": "median",
                    "medianMetric": "loadTime",
              },
      },
}
```

4.1.4 **Relatórios**

As ferramentas utilizadas para o desenvolvimento deste protótipo, com os dados recolhidos em cada teste são capazes de gerar relatórios detalhados que informam não só os valores das métricas obtidas durante o teste, mas também permitem ter uma clara compreensão de que modo ocorreu uma navegação para o *website* sobre teste, assim como verificar grande parte dos recursos descarregados.

Com esta informação é possível verificar comportamentos anormais durante o ciclo de *renderização* de uma página, bem como descobrir elementos intervenientes que possam aumentar o tempo de carregamento de uma página.

Dada a natureza de cada uma das ferramentas utilizadas, os relatórios produzidos pelo *WebPageTest* e pelo *Calibre* fornecem diferentes dados, assim como outras funcionalidades para interpretar os resultados obtidos.

4.1.4.1 *WebPageTest*

O relatório de um teste de desempenho executado pelo *WebPageTest* contém diversa informação disponível, mas rica em detalhe. A informação obtida agrupa-se nas secções:

- *Summary*, responsável por apresentar um resumo dos valores obtidos no teste de desempenho realizado (cf. secção 4.1.4.1.1);
- *Details*, responsável por fornecer (cf. secção 4.1.4.1.2)
- *Performance Review*, responsável por … (cf. secção 4.1.4.1.3)
- *Content Breakdown*, responsável por …(cf. secção 4.1.4.1.4)
- *Domains*, responsável por representar a distribuição (cf. secção 4.1.4.1.5)
- *Screenshot,* destinada à apresentação de uma captura de ecrã no estado *Fully Loaded* obtida no dispositivo e *browser* selecionados para executar o teste.

O cabeçalho (Figura 22), apresentado em todas as secções de um relatório do *WebPageTest*, contêm informações gerais relativas ao teste realizado,

- URL alvo de análise pelo *WebPageTest* (1);
- A localização do teste e o *browser* utilizado (2);
- Data e hora de execução do teste (3);
- Pontuação da categoria *PWA* do *Google Lighthouse* (cf. secção 2.8.3), caso o dispositivo utilizado para teste suporte este tipo de análise (4);

- Classificações de otimização (cf. secção 2.8.1) (5).

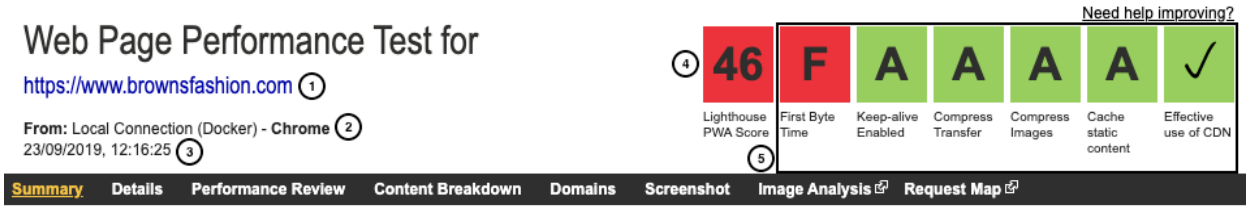


Figura 22. Cabeçalho e menu de um relatório *WebPageTest*

Relativamente às classificações de otimização e pontuação da categoria PWA do *Google Lighthouse*, ao clicar em cada uma das pontuações apresentadas, é possível aceder a mais detalhe sobre a mesma. No caso das classificações de otimização, quando selecionada uma das classificações, é-se reencaminhado para uma página que contém sugestões de otimização e indica aspetos possíveis de ser melhorados, de acordo com a categoria selecionada (cf. secção 4.1.4.1.3).

Ao clicar na classificação de PWA do *Google Lighthouse* no cabeçalho, é possível aceder ao relatório gerado, como o apresentado na Figura 23, onde contém todas as classificações das categorias avaliadas – *Performance*, Acessibilidade, Boas Práticas, SEO, *Progressive Web App* - bem como os resultados da auditoria efetuada e sugestões de melhoria.

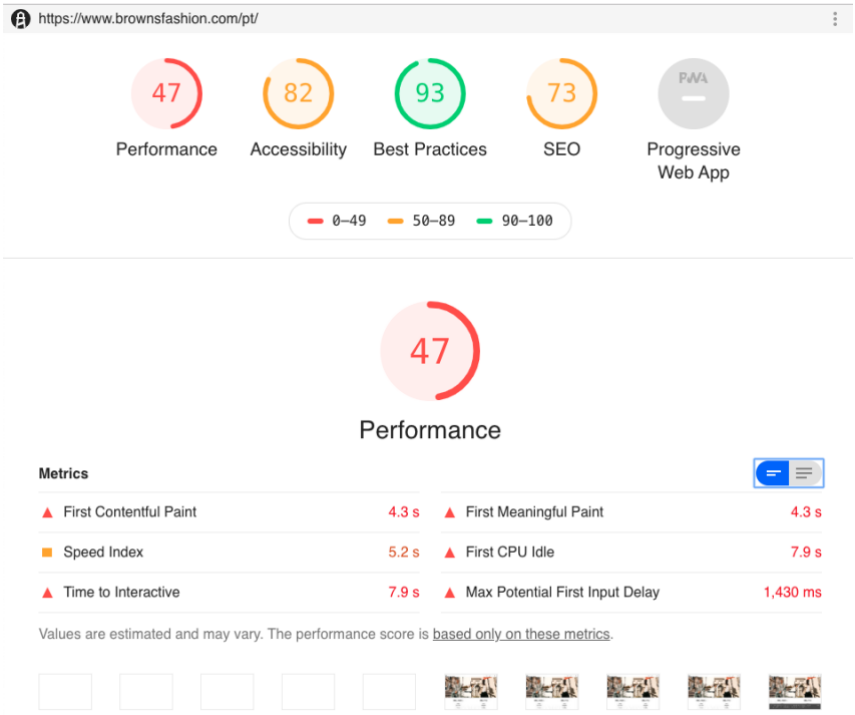


Figura 23. Relatório gerado pelo *Google Lighthouse* ao correr um teste via *WebPageTest*

Uma hiperligação para este relatório é disponibilizada no final de qualquer teste iniciado com sucesso pelo *WebPageTest*, ficando o resultado do teste persistido no servidor público da ferramenta.

No caso de o teste ter sido efetuado recorrendo a uma instância privada, os relatórios são perdidos sempre que o *Docker Container* responsável por iniciar o servidor é reiniciado. Tendo em conta que este servidor não possui persistência na máquina responsável por iniciar o servidor e o agente do *WebPageTest* via *Docker*, é utilizada uma funcionalidade das instâncias privadas do *WebPageTest*, que permite publicar os resultados de uma instância privada no servidor público do *WebPageTest*, gerando assim uma nova hiperligação para o teste realizado, conforme apresentado na Figura 24.

```
Test report available on http://127.0.0.1:4000/results.php?test=191001_27_1
Public Test report available on https://www.webpagetest.org/results.php?test=191001_E9_a0f80d81bb85d243f2f6dee7d7e79152
```

Figura 24. Hiperligações para os relatórios gerados pelo *WebPageTest* no protótipo, durante a execução de um teste de desempenho numa instância privada iniciada via *Docker*

Assim, é possível consultar estes relatórios sem depender da instância privada do *WebPageTest*, prevenindo assim a utilização de um servidor dedicado a este efeito e o consumo de recursos pelo mesmo.

4.1.4.1.1 *Summary*

A secção *Summary* é responsável por apresentar um resumo de toda a informação obtida relativa a um teste realizado:

- O número de *test runs* realizados;
- Os resultados das métricas medidas para as vistas disponíveis;
- Diagrama *Waterfall*, com captura de ecrã no estado *Fully Loaded*;
- Gráficos circulares representando o *Content Breakdown*.

Os resultados das métricas medidas para as vistas disponíveis - *First View* ou *Repeat View*, caso esta tenha sido incluída – são apresentados na tabela identificada como *Performance Results (Median Run*), apresentado na Figura 25, correspondem aos valores obtidos na *test run* cuja métrica definida como mediana obteve tal valor (cf. secção 4.1.2);

Tester: docker-desktop-172.21.0.1
Test runs: **3**
Re-run the test

Raw page data - Raw object data
Export HTTP Archive (.har)
View Test Log - Lighthouse Test Log
Publish to www.webpagetest.org

Performance Results (Median Run)

							Document Complete			Fully Loaded		
	Load Time	First Byte	Start Render	First Contentful Paint	Speed Index	First CPU Idle	Time	Requests	Bytes In	Time	Requests	Bytes In
First View (Run 3)	6.390s	1.244s	4.600s	2.948s	5.645s	4.389s	6.390s	49	2,115 KB	14.345s	106	2,856 KB
Repeat View (Run 2)	4.794s	1.113s	3.800s	2.377s	3.800s	> 2.377s	4.794s	6	54 KB	5.838s	10	75 KB

Figura 25. Resumo das métricas medidas apresentados pelo *WebPageTest*

O diagrama *Waterfall* é responsável por representar todos os recursos de um *website* num formato de lista em cascata, que representa a ordem em que esses recursos foram carregados, assim como o tempo que demorou a carregar cada recurso. É complementado ainda com respetiva captura de ecrã (*screenshot*) no estado *Fully Loaded*.

Na Figura 26 é possível verificar um exemplo do tipo de resultado apresentado, neste caso para a *test run* número 2, com os resultados obtidos quer na *First View* como na *Repeat View*.

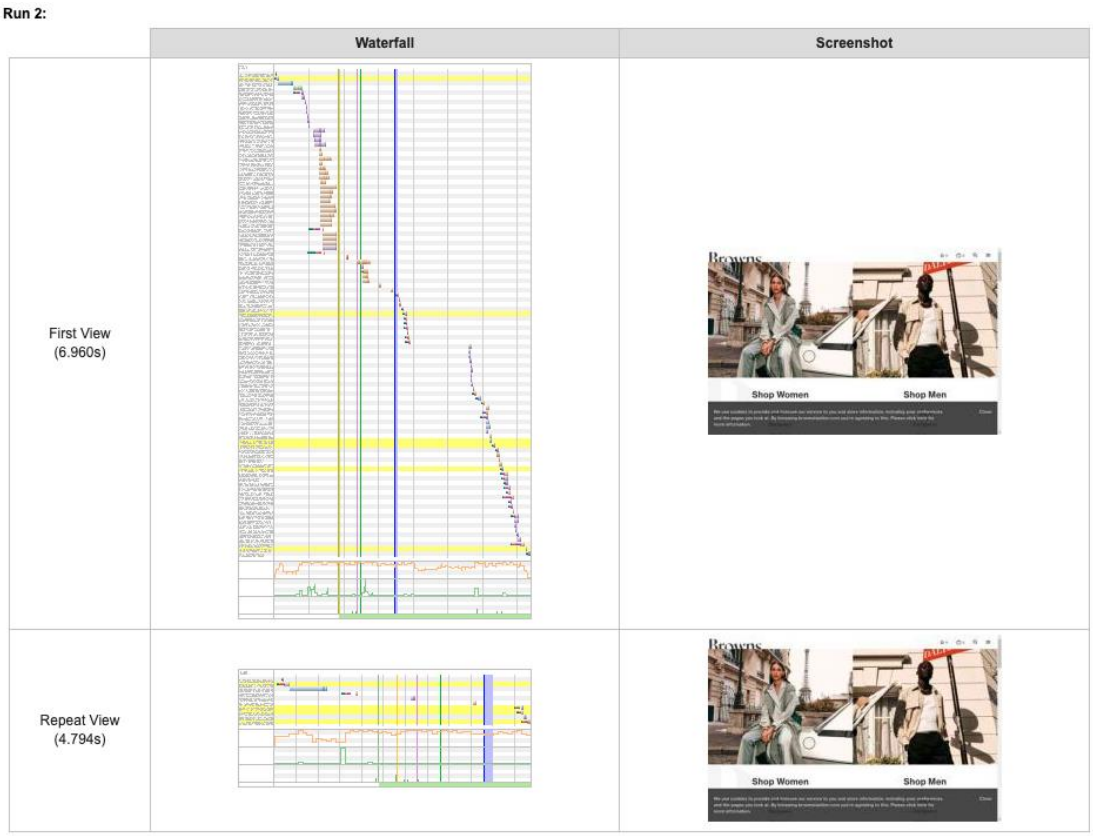


Figura 26. Diagrama *Waterfall* e capturas de ecrã apresentados pelo *WebPageTest*

O conteúdo *Content Breakdown*, representado na Figura 27, contém dois gráficos circulares responsáveis por representar a distribuição por tipo (HTML, CSS, JavaScript, Imagens, Fontes Web):

- Da quantidade de recursos solicitados (1);
- Do seu tamanho, em *bytes* (2).

Os dados utilizados para esta representação são provenientes dos resultados obtidos *First View* associados à *test run* determinada como sendo a *test run* mediana.

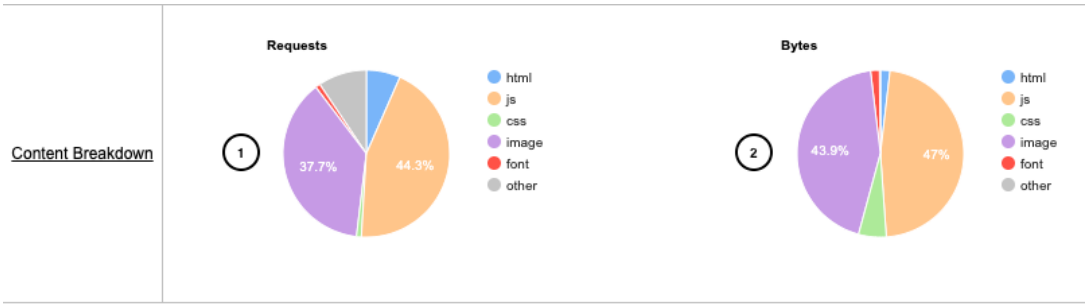


Figura 27. Gráficos *Content Breakdown* resumidos apresentados pelo *WebPageTest*

4.1.4.1.2 *Details*

Esta secção, tal como o nome indica, é responsável por fornecer informação mais técnica e detalhada relativa a um teste de desempenho realizado.

Por defeito, a informação apresentada são os resultados obtidos na primeira *test run* relativos à *First View*. No entanto, caso se clique em qualquer um dos diagramas *Waterfall* na secção *Summary*, esta página passa a apresentar a informação detalhada relativa à *test run* e vista selecionada. Esta secção apresenta os seguintes conteúdos:

- Métricas medidas, semelhante à tabela da secção *Summary* na Figura 25, onde acresce o valor obtido para a métrica *Visually Complete* e informação relativa ao tempo de carregamento da página, conforme representado pela Figura 28:

									Document Complete			Fully Loaded		
	Load Time	First Byte	Start Render	First Contentful Paint	Visually Complete	Speed Index	First CPU Idle	Result (error code)	Time	Requests	Bytes In	Time	Requests	Bytes In
First View (Run 1)	3.563s	1.126s	3.000s	2.067s	5.900s	3.551s	2.652s	0	3.563s	47	2,115 KB	8.143s	102	2,856 KB

domInteractive	domContentLoaded	loadEvent
2.111s	2.652s - 2.652s (0.000s)	3.563s - 3.615s (0.052s)

Figura 28. Métricas apresentadas na secção *Details* do *WebPageTest*

- Vista *Waterfall* interativa, apresentada na Figura 29, em que para além de permitir visualizar os pedidos e recursos solicitados durante o carregamento de um *website*, é possível interagir com cada pedido isoladamente (semelhante à análise de pedidos possíveis de se fazer nas ferramentas para desenvolvedores disponibilizadas na maioria dos *browsers*) (ThemeFusion, 2019);

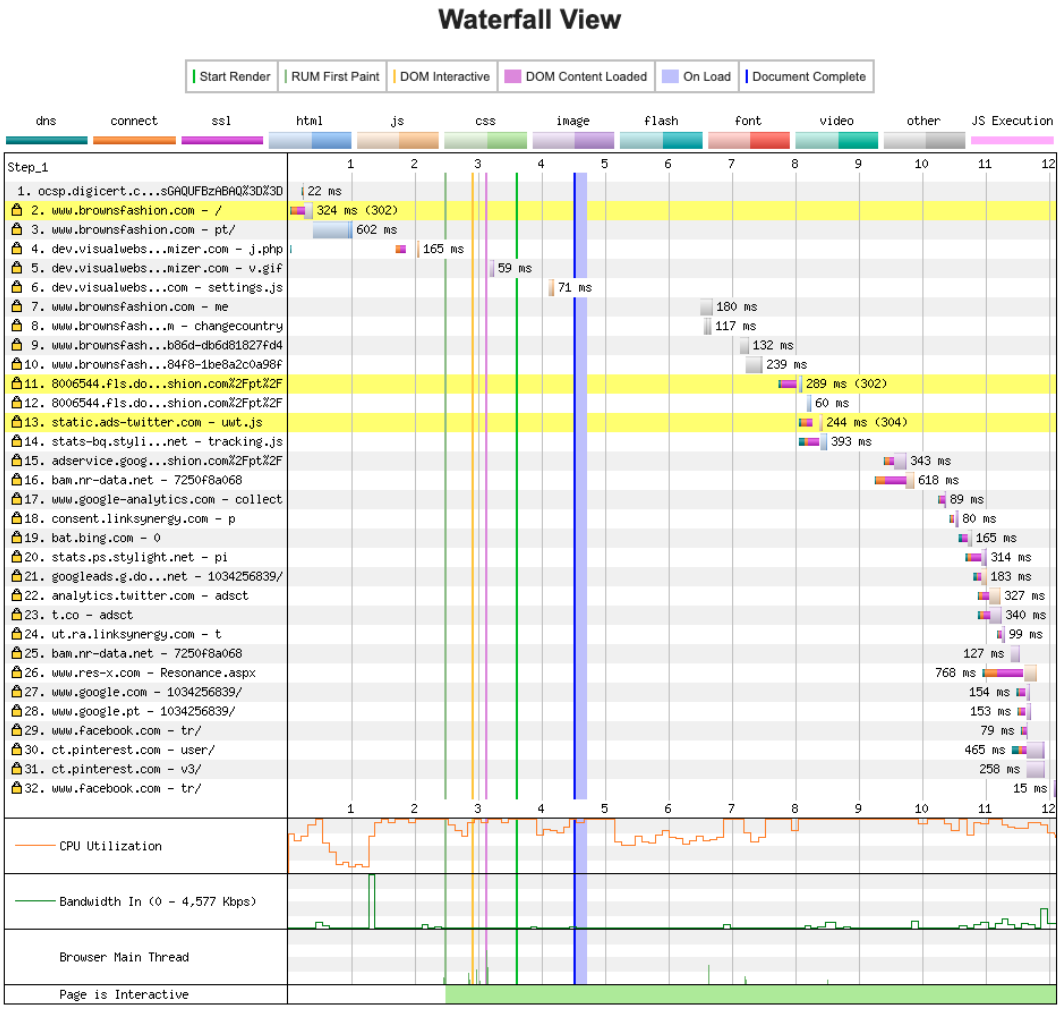


Figura 29. Vista *Waterfall* interativa apresentada pelo *WebPageTest*

- Vista de Conexões (*Connection View*), representada na Figura 30, que se foca em representar a forma como o conteúdo é lido a partir de todas as fontes/domínios utilizados, através da análise dos pedidos efetuados por cada *socket* TCP[42].

[42] Ponto de uma ligação bidirecional feita entre dois aplicações ou dispositivos ligados entre si. Um *socket* está associado a um porto, de modo a que a camada TCP consiga identificar a aplicação destinatária dos dados a transmitir (Oracle, 2019)

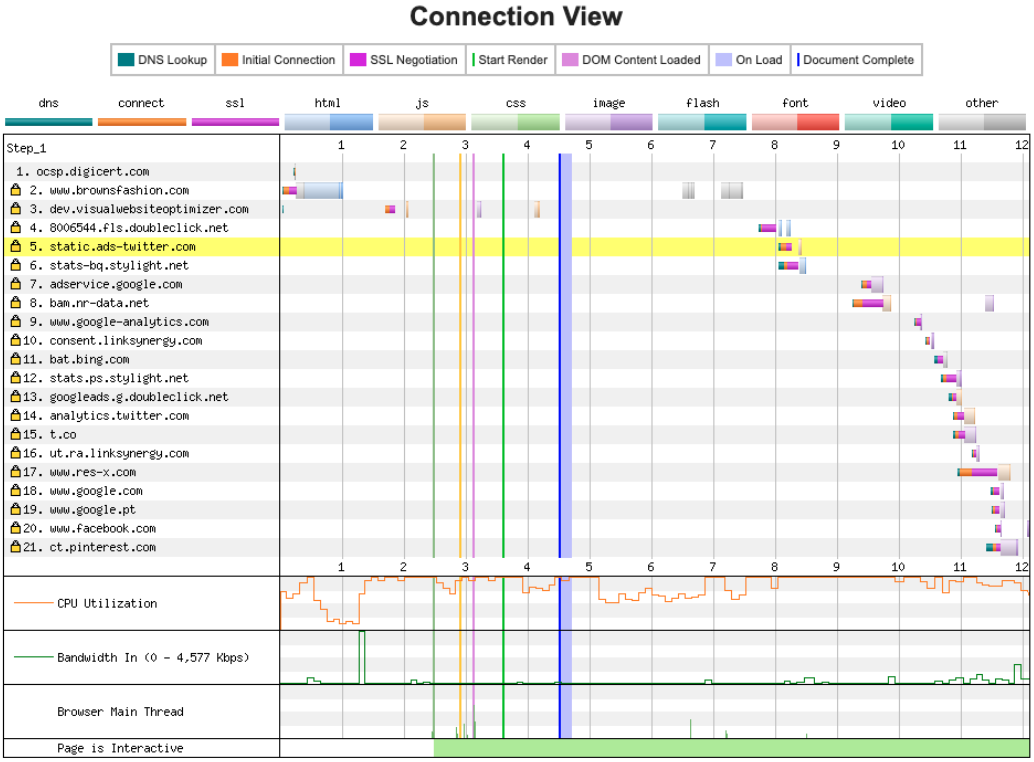
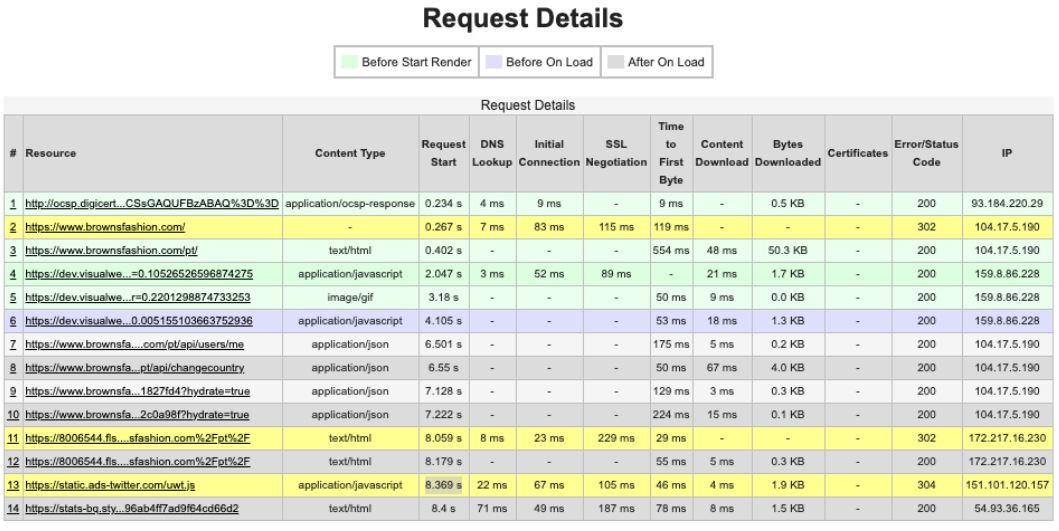


Figura 30. Detalhes da *Connection View* apresentada pelo *WebPageTest*

- Detalhes dos pedidos (*Request Details*), apresentada na Figura 31, que lista todos os recursos solicitados durante o carregamento de um website, fornecendo para cada informação relativa ao tipo, tempo em que foi iniciado o pedido e em que foi descarregado e o respetivo tamanho associado. Ao selecionar um pedido, este relatório disponibiliza os respetivos *request headers* (Figura 32), que contém informação adicional relativa ao pedido a efetuar e sobre quem o solicita.



#	Resource	Content Type	Request Start	DNS Lookup	Initial Connection	SSL Negotiation	Time to First Byte	Content Download	Bytes Downloaded	Certificates	Error/Status Code	IP
1	http://ocsp.digicert...CSsGAQUFBzABAQ%3D%3D	application/ocsp-response	0.234 s	4 ms	9 ms	-	9 ms	-	0.5 KB	-	200	93.184.220.29
2	https://www.brownsfashion.com/	-	0.267 s	7 ms	83 ms	115 ms	119 ms	-	-	-	302	104.17.5.190
3	https://www.brownsfashion.com/pt/	text/html	0.402 s	-	-	-	554 ms	48 ms	50.3 KB	-	200	104.17.5.190
4	https://dev.visualwe...=0.10526526596874275	application/javascript	2.047 s	3 ms	52 ms	89 ms	-	21 ms	1.7 KB	-	200	159.8.86.228
5	https://dev.visualwe...r=0.2201298874733253	image/gif	3.18 s	-	-	-	50 ms	9 ms	0.0 KB	-	200	159.8.86.228
6	https://dev.visualwe...0.005155103663752936	application/javascript	4.105 s	-	-	-	53 ms	18 ms	1.3 KB	-	200	159.8.86.228
7	https://www.brownsfa....com/pt/api/users/me	application/json	6.501 s	-	-	-	175 ms	5 ms	0.2 KB	-	200	104.17.5.190
8	https://www.brownsfa...pt/api/changecountry	application/json	6.55 s	-	-	-	50 ms	67 ms	4.0 KB	-	200	104.17.5.190
9	https://www.brownsfa...1827fd4?hydrate=true	application/json	7.128 s	-	-	-	129 ms	3 ms	0.3 KB	-	200	104.17.5.190
10	https://www.brownsfa...2c0a98f?hydrate=true	application/json	7.222 s	-	-	-	224 ms	15 ms	0.1 KB	-	200	104.17.5.190
11	https://8006544.fls....sfashion.com%2Fpt%2F	text/html	8.059 s	8 ms	23 ms	229 ms	29 ms	-	-	-	302	172.217.16.230
12	https://8006544.fls....sfashion.com%2Fpt%2F	text/html	8.179 s	-	-	-	55 ms	5 ms	0.3 KB	-	200	172.217.16.230
13	https://static.ads-twitter.com/uwt.js	application/javascript	8.369 s	22 ms	67 ms	105 ms	46 ms	4 ms	1.9 KB	-	304	151.101.120.157
14	https://stats-bq.sty...96ab4ff7ad9f64cd66d2	text/html	8.4 s	71 ms	49 ms	187 ms	78 ms	8 ms	1.5 KB	-	200	54.93.36.165

Figura 31. Tabela *Requests Details* apresentada pelo *WebPageTest*

- Request 7: https://www.brownsfashion.com/pt/api/users/me

URL: https://www.brownsfashion.com/pt/api/users/me
Host: www.brownsfashion.com
IP: 104.17.5.190
Error/Status Code: 200
Priority: MEDIUM
Protocol: HTTP/2
HTTP/2 Stream: 5, weight 220, depends on 0, EXCLUSIVE
Initiated By: https://ajax.cloudflare.com/cdn-cgi/scripts/95c75768/cloudflare-static/rocket-loader.min.js
Client Port: 33146
Request Start: 6.501 s
Time to First Byte: 175 ms
Content Download: 5 ms
Bytes In (downloaded): 0.2 KB
Uncompressed Size: 0.3 KB
Bytes Out (uploaded): 0.3 KB

Request Headers:

X-NewRelic-ID: VQUCV1ZUGwIFVlBRDgcA
Accept: application/json
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/76.0.3809.132 Safari/537.36 PTST/19.04
Referer: https://www.brownsfashion.com/
Sec-Fetch-Mode: cors
X-Requested-With: XMLHttpRequest
:method: GET
:authority: www.brownsfashion.com
:scheme: https
:path: /pt/api/users/me
accept: application/json
x-newrelic-id: VQUCV1ZUGwIFVlBRDgcA
x-requested-with: XMLHttpRequest
user-agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/76.0.3809.132 Safari/537.36 PTST/19.04
sec-fetch-mode: cors
sec-fetch-site: same-origin
referer: https://www.brownsfashion.com/
accept-encoding: gzip, deflate, br
accept-language: en-US,en;q=0.9
cookie: [1194 bytes were stripped]

Response Headers:

status: 200
expect-ct: max-age=604800, report-uri="https://report-uri.cloudflare.com/cdn-cgi/beacon/expect-ct"
x-xss-protection: 1; mode=block
x-content-type-options: nosniff
content-encoding: br
strict-transport-security: max-age=31536000; includeSubDomains
vary: Accept-Encoding
expires: Tue, 01 Oct 2019 08:46:33 GMT
server: cloudflare
x-newrelic-app-data:
PxQGUlNVCAQTVVRQBgIOUFMTGhE1AwE2QgNWEVlbQFtcCxY2VANyMi0ZYhIDEUscdwEVl0RIWQVGHQYdUkpTTABQD1APCAEeHlQVQwVZcidRB)
x-server: WEB03
pragma: no-cache
cache-control: no-store, must-revalidate, no-cache
date: Tue, 01 Oct 2019 08:46:33 GMT
x-frame-options: DENY
referrer-policy: strict-origin
content-type: application/json; charset=utf-8
cf-ray: 51ed2ab56ec3da7e-LIS
:status: 200
set-cookie: [99 bytes were stripped]

Figura 32. Exemplo do detalhe dos *Requests Header* apresentado pelo *WebPageTest* a um pedido de rede

4.1.4.1.3 *Performance Review*

Esta secção que apresenta em detalhe as áreas avaliadas pelo *WebPageTest* no âmbito da obtenção das classificações de otimização (cf. secção 2.8.1).

Inicialmente, apresenta uma tabela que representa o modo como cada pedido se demonstra relativo a cada critério avaliado – *Full Optimization Checklist* (Figura 33).

Full Optimization Checklist

Step_1	Keep-Alive 100%	GZip 100%	Compress Img 100%	Progressive N/A	Cache Static 0%	CDN Detected 80%
1: ocsp.digicert.com...sGAQUFBzABAQ%3D%3D						
2: www.brownsfashion.com - /						
3: www.brownsfashion.com - pt/	✓	✓				✓
4: dev.visualwebsiteoptimizer.com - j.php	✓	✓			✗	✗
5: dev.visualwebsiteoptimizer.com - v.gif	✓		✓			
6: dev.visualwebsit...r.com - settings.js	✓	✓			✗	✗
7: www.brownsfashion.com - me	✓	✓				✓
8: www.brownsfashion.com - changecountry	✓	✓				✓
9: www.brownsfashio...4-b86d-db6d81827fd4	✓	✓				✓
10: www.brownsfashi...2-84f8-1be8a2c0a98f	✓	✓				✓
11: 8006544.fls.dou...Fashion.com%2Fpt%2F						
12: 8006544.fls.dou...Fashion.com%2Fpt%2F	✓	✓				✓
13: static.ads-twitter.com - uwt.js						
14: stats-bq.stylight.net - tracking.js	✓					
15: adservice.googl...Fashion.com%2Fpt%2F						
16: bam.nr-data.net - 7250f8a068	✓					
17: www.google-analytics.com - collect	✓		✓			✓
18: consent.linksynergy.com - p	✓		✓		✗	✓
19: bat.bing.com - 0						
20: stats.ps.stylight.net - pi	✓		✓			
21: googleads.g.dou...k.net - 1034256839/	✓	✓				✓
22: analytics.twitter.com - adsct	✓	✓				
23: t.co - adsct	✓	✓	✓			
24: ut.ra.linksynergy.com - t	✓		✓		✗	✓
25: bam.nr-data.net - 7250f8a068	✓		✓		✗	✗
26: www.res-x.com - Resonance.aspx	✓	✓			✗	✗
27: www.google.com - 1034256839/	✓		✓			✓
28: www.google.pt - 1034256839/	✓		✓			✓
29: www.facebook.com - tr/	✓		✓			✓
30: ct.pinterest.com - user/	✓		✓			✓
31: ct.pinterest.com - v3/	✓		✓			✓
32: www.facebook.com - tr/	✓		✓			✓
	Keep-Alive	GZip	Compress Img	Progressive	Cache Static	CDN Detected

Figura 33. Tabela *Full Optimization Checklist* do *WebPageTest*

Contém ainda uma secção de detalhe, em que contém informação e sugestões de melhoria para os problemas detetados e assinalados na tabela da Figura 33.

As informação e sugestões de melhoria apresentadas podem consistir em valores que devem ser atingidos numa dada métrica, de modo a aumentar a sua classificação, ou a informação de possíveis conteúdos solicitados que possam usufruir de otimizações – como a utilização de *cache* ou que o seu tamanho possa/deva ser reduzido.

Na Figura 34 é possível observar algumas sugestões de melhoria, como é o caso do *First Byte Time*, cujo valor neste teste foi de 911 ms. Para obter a classificação máxima, deve ser reduzida para 362 ms. Na secção "*Leverage browser caching of static assets"* indica possíveis recursos estáticos em que pode ser aplicada uma estratégia de *caching*, de modo a reduzir o seu tempo de carregamento e, consequentemente, melhorar a pontuação desta categoria.

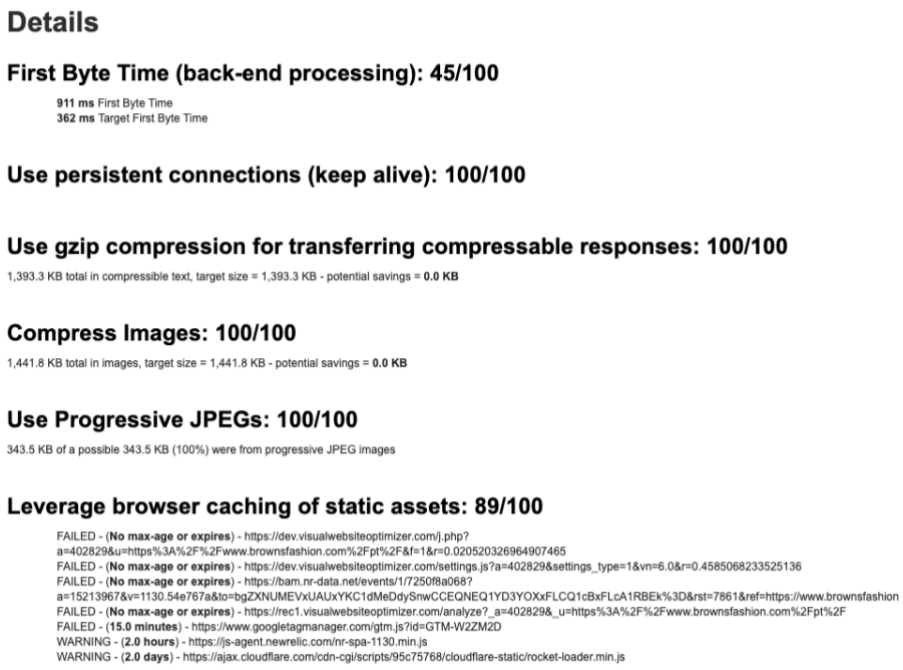
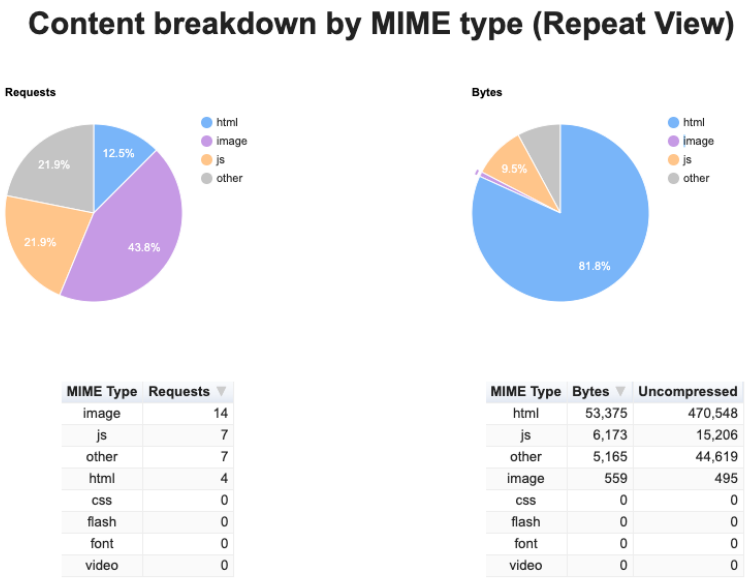


Figura 34. Sugestões de otimização indicadas pelo *WebPageTest*

4.1.4.1.4 *Content Breakdown*

A secção *Conntent Breakdown* é responsável por fornecer mais detalhe face aos gráficos circulares disponíveis na secção *Summary* (cf. secção 4.1.4.1.1), onde apresenta informação sobre a distribuição dos tipos de dados solicitados - em número e tamanho.

Esta secção disponibiliza gráficos com mais detalhe, nomeadamente para os resultados obtidos na *First View* e *Repeat View*, conforme apresentado na Figura 35.



MIME Type	Requests
image	14
js	7
other	7
html	4
css	0
flash	0
font	0
video	0

MIME Type	Bytes	Uncompressed
html	53,375	470,548
js	6,173	15,206
other	5,165	44,619
image	559	495
css	0	0
flash	0	0
font	0	0
video	0	0

Figura 35. Gráficos circulares da secção *Content Breakdown* do *WebPageTest*

Para além dos gráficos circulares, disponibiliza ainda um diagrama em cascata – *Connection View* - em que é possível verificar em que momento cada tipo de recursos estão a ser solicitados, conforme apresenta a Figura 36.

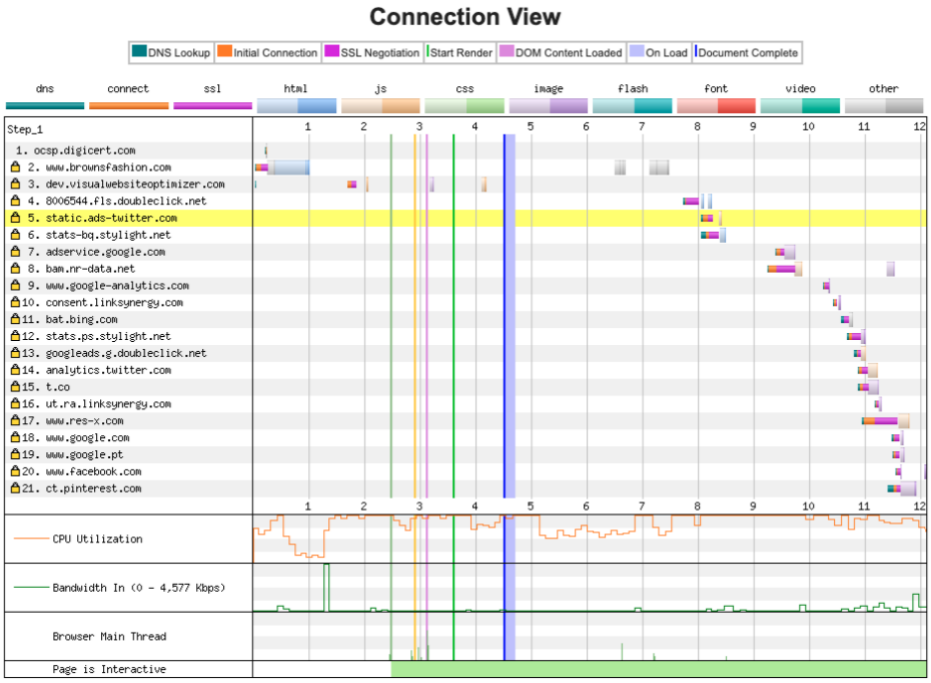
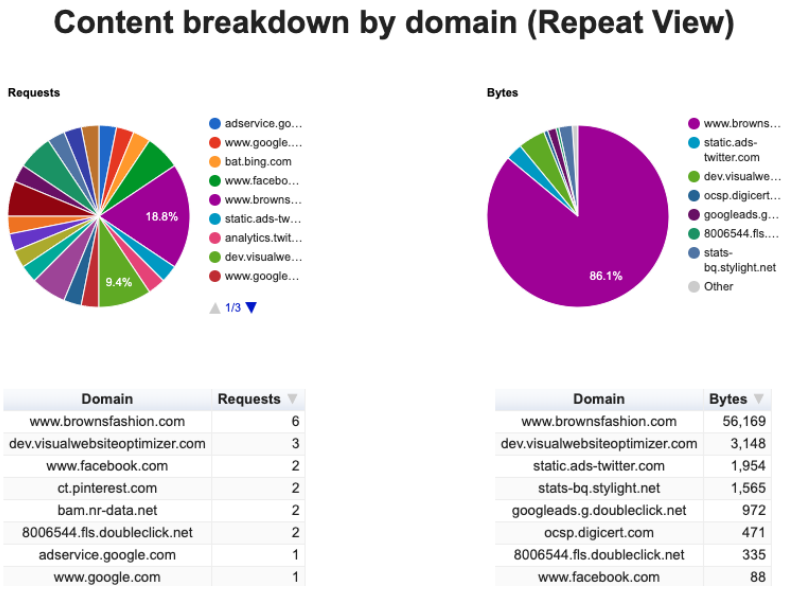


Figura 36. Diagrama *Connection View* da secção *Content Breakdown* do *WebPageTest*

4.1.4.1.5 *Domain*

A secção *Domain* é responsável por apresentar, com recurso aos gráficos circulares, os pedidos efetuados (*requests*) e a quantidade de dados solicitados (*bytes*), distribuídos pelos vários domínios onde tais recursos solicitados provêm, conforme se pode verificar na Figura 37.



Domain	Requests
www.brownsfashion.com	6
dev.visualwebsiteoptimizer.com	3
www.facebook.com	2
ct.pinterest.com	2
bam.nr-data.net	2
8006544.fls.doubleclick.net	2
adservice.google.com	1
www.google.com	1

Domain	Bytes
www.brownsfashion.com	56,169
dev.visualwebsiteoptimizer.com	3,148
static.ads-twitter.com	1,954
stats-bq.stylight.net	1,565
googleads.g.doubleclick.net	972
ocsp.digicert.com	471
8006544.fls.doubleclick.net	335
www.facebook.com	88

Figura 37. Gráficos do *Content Breakdown* por domínio apresentado pelo *WebPageTest*

4.1.4.2 *Calibre*

No relatório produzido com os resultados obtidos após um teste de desempenho executado pelo *Calibre*, é possível consultar:

- Os valores obtidos para as métricas medidas de forma sumarizada, sendo elas o *Time to First Byte*, *Response Time*, *First Contentful Paint, First Meaningful Paint*, *Visually Complete*, *Time to Interactive*, *JS Parse & Compile* (tempo utilizado para interpretar e compilar código *JavsScript*) e *onLoad* (tempo de carregamento de uma página) (cf. secção 2.4);

- A linha temporal de *renderização* do *website*, acompanhada por imagens;

- O tamanho e tempo de execução de *scripts* externos;

- A lista de pedidos efetuados pelo *website* durante o seu carregamento, tendo em conta a sua prioridade, tamanho e tempo;

- Um vídeo que demonstra o que o utilizador final observa até que o *website* se encontra totalmente carregado;

- As avaliações e respetivos parâmetros avaliados pelo *Google Lighthouse* (cf. secção 2.8.3), estando as respetivas categorias distribuídas sob a forma de separadores no relatório produzido;

- Um separador denominado *Third Party* que detalha o tamanho e processamento efectuado de conteúdos externos ao *website*, tipicamente associados a *scripts* externos para recolha de estatísticas de um *website* (e.g. *Google Analytics*).

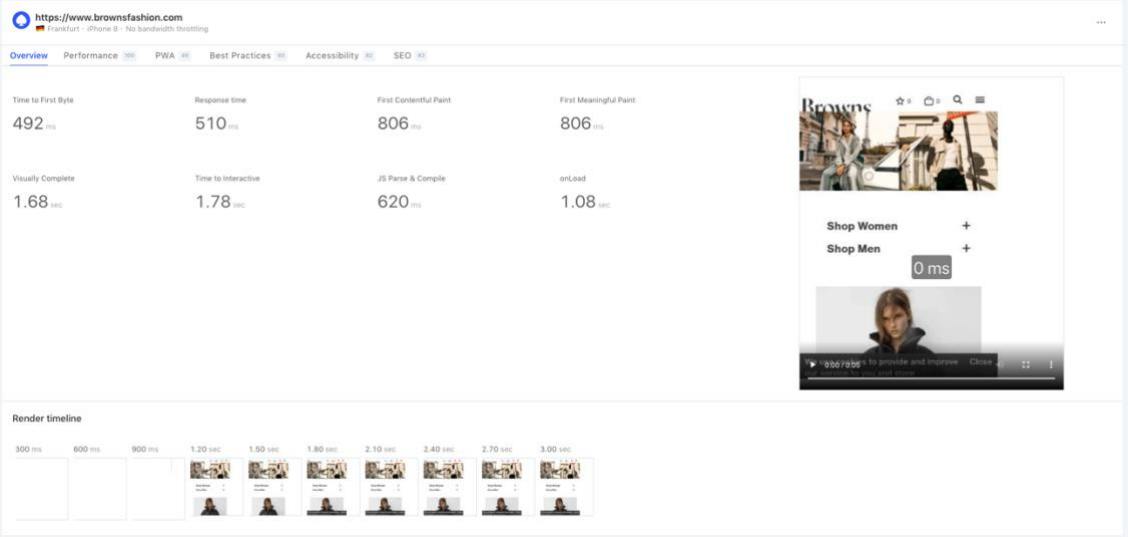


Figura 38. Relatório de um teste de desempenho produzido pelo *Calibre*

Uma hiperligação para este relatório é sempre disponibilizada no fim de qualquer teste iniciado com sucesso com o *Calibre* (cf. secção 2.8.2), ficando o resultado do teste persistido no servidor público da ferramenta.

4.2 *Performance Budget* organizacional

Um *performance budget* destina-se a representar um orçamento associado ao desempenho de uma aplicação (cf. secção 2.3), onde neste são definidos limites padrão, associados a diversas características, que não devem ser ultrapassados numa página *web*, mais concretamente, métricas de desempenho (cf. secção 2.4).

Dada a importância de garantir um nível de desempenho adequado nas aplicações e *websites* desenvolvidos por FPS, foi necessário selecionar as métricas adequadas a analisar, seja nos *websites* já disponibilizados ao público, de modo a fornecer informação de possíveis otimizações e melhorias a aplicar, assim como a *websites* em produção, de modo a poderem cumprir um nível definido organizacionalmente antes de serem implantados para os consumidores.

Foram selecionadas as métricas apresentadas na Tabela 7, visto estas serem as mais relevantes para avaliarem a experiência do utilizador no acesso ao *website* (cf. secção 2.4), por serem possíveis de serem analisadas e comparadas pelo *Calibre* e *WebPageTest* (cf. secção 4.1.3), além de práticas adotadas pela indústria, mais especificamente no caso da ferramenta *Google Lighthouse* (cf. secção 2.8.3), que recorre a algumas destas métricas de modo a poder atribuir a sua classificação à categoria de *performance* (Google, 2019e) (assinaladas com o símbolo * na listagem presente na Tabela 7).

Tabela 7. Listagem das métricas a analisar para definir o *performance budget*

Métrica Selecionadas
*First Contentful Paint**
*First Meaningful Paint**
*First CPU Idle**
*Time to Interactive**
*SpeedIndex**
Time to First Byte
Número de Pedidos
First Paint
Visually Complete
Visually Complete 85%
Tempo de carregamento de página (*Load Time*)
Pontuação categoria de acessibilidade do Google Lighthouse (*Lighthouse Accessibility*)
Pontuação categoria de boas práticas do Google Lighthouse (Lighthouse *Best Practices*)
Pontuação categoria de desempenho do Google Lighthouse (*Lighthouse Performance*)
Pontuação categoria de PWA do Google Lighthouse (*Lighthouse PWA*)
Pontuação categoria de SEO do Google Lighthouse (*Lighthouse SEO*)

De modo a definir os valores mais adequados para o *budget* de cada métrica do conjunto identificado anteriormente, foram recolhidos os seguintes tipos de dados:

- Dados sintéticos, proveniente dos testes programados na ferramenta *Calibre*, executados a cada 16 horas às páginas mais importantes de um *website* acessíveis sem

início de sessão[43] – a Página Inicial (*Homepage*), a página de listagem de produtos (PLP) e a página de detalhe de um produto (PDP). A informação aqui obtida representam dados laboratoriais, por terem sido efetuadas em condições específicas e controladas (cf. secção 2.5).

- Dados de utilização real (RUM) obtidos e registados na ferramenta *New Relic*, acessíveis sob a forma de *dashboards*. São disponibilizadas métricas como o tempo de carregamento da página (*Load Time*), *Time to First Byte*, tempo médio de *renderização* de uma página, tempo de processamento DOM (cf. secções 2.5).

Pela combinação média dos valores obtidos com base nos tipos de dados enumerados, foi possível definir um *performance budget* inicial, a aplicar nos vários projetos de FPS.

No entanto, apesar de estes valores terem sido configurados a partir de uma base de informação recolhida, é de todo relevante que durante os novos desenvolvimentos de um *website* os valores definidos no *performance budget* possam vir a ser redefinidos, colocando limites inferiores, de modo a contribuir para uma constante preocupação em oferecer cada vez melhor desempenho e tempos reduzidos de resposta e de *feedback* da página durante a sua utilização.

A aplicação deste *performance budget* deve ser aplicada da seguinte forma (Mihajlija, 2018; Mishunov, 2015):

- Aos valores das métricas medidas inicialmente para os *websites* desenvolvidos por FPS, apontar como *budget* um valor inferior a 20% em cada métrica. Estes 20% surgem de uma simplificação da regra de Weber-Fechner, onde alguns estudos afirmam que os utilizadores reconhecem diferenças caso os tempos de resposta sejam iguais ou superiores a 20%. (Mishunov, 2015).

- Numa fase posterior, após otimização com base no *performance budget* definido com base nos valores das métricas internas, garantir que cada métrica monitorizada seja no mínimo 20% mais rápida que a mesma métrica recolhida de *websites* concorrentes.

No entanto, tendo em conta a existência de algumas otimizações já aplicadas por parte de FPS na implantação dos *websites*, como a aplicação de políticas de *caching* na CDN, e na integração deste protótipo no ciclo de desenvolvimento das várias aplicações criadas por FPS, o *performance budget* inicial será feito apenas com base nos dados laboratoriais e RUM disponíveis.

Os dados laboratoriais gravados na ferramenta *Calibre*, tal como já foi descrito, dizem respeito aos testes executados diariamente cujos resultados são registados para estatística e posterior consulta. A Tabela 8 apresenta uma listagem de métricas selecionadas para definir o

[43] Tal se deve ao facto da ferramenta não conseguir autenticar previamente numa página que apenas seja acessível com sessão iniciada.

performance budget organizacional, cujos valores dizem respeito a resultados provenientes de dezassete dos *websites* desenvolvidos por FPS que se encontram a ser monitorizados no *Calibre*, realizados entre 7 de setembro a 7 de outubro de 2019.

Tabela 8. Médias das métricas monitorizadas pelo *Calibre* nos *websites* FPS, por área e dispositivo

	Médias					
Área	**Página Inicial**		**PLP**		**PDP**	
Métrica / Dispositivo	**Desktop**	**Mobile**	**Desktop**	**Mobile**	**Desktop**	**Mobile**
Time to First Byte (s)	0,659	0,673	0,712	0,596	0,682	0,650
SpeedIndex (s)	3,062	6,685	3,020	4,858	2,286	5,882
First Meaningful Paint (s)	1,974	3,798	2,036	3,400	2,386	3,366
Load Time (s)	2,329	11,720	2,374	5,082	2,491	5,178
First Contentful Paint (s)	1,254	3,112	1,215	2,138	1,654	2,813
First CPU Idle (s)	3,357	8,921	4,015	6,287	3,470	7,602
First Paint (s)	1,211	3,076	1,215	2,138	1,196	2,612
Time to Interactive (s)	3,819	12,222	4,354	7,399	5,917	3,761
Visually Complete (s)	6,846	14,472	5,092	8,101	4,953	9,638
Visually Complete 85% (s)	4,124	10,786	3,601	5,663	2,765	6,588
Lighthouse Accessibility	83,24%	87,12%	83,29%	85,82%	84,94%	85,59%
Lighthouse Best Practices	88,41%	88,35%	88,06%	88,88%	87,64%	87,53%
Lighthouse Performance	87,35%	60,29%	86,35%	66,29%	91,29%	65,76%
Lighthouse PWA	42,47%	41,75%	47,73%	41,75%	44,20%	41,75%
Lighthouse SEO	93,27%	93,19%	93,27%	93,06%	93,87%	93,75%
Número de pedidos *(un)*	73	76	87	79	75	76

A Tabela 8 apresenta a média das métricas medidas entre o conjunto de dezassete *websites* monitorizados. Estes *websites* configurados no Calibre possuem, pelo menos, dois dispositivos, um com o perfil *Desktop* e outro *Mobile*. Na Tabela 9, é listada os vários dispositivos, e o número de vezes que esse dispositivo é utilizado para efetuar testes por *websites* monitorizados.

Tabela 9. Dispositivos utilizados pelo *Calibre* para os testes de desempenho

Perfil	**Dispositivo/*Browser***	**Tipo de Ligação**	**Utilizações**
Desktop	*Google Chrome*	*Ethernet*	17
Mobile	*Apple iPhone 7*	3G	1

Perfil	**Dispositivo/*Browser***	**Tipo de Ligação**	**Utilizações**
Mobile	*Apple iPhone 7*	4G	1
Mobile	*Apple iPhone 8*	4G	11
Mobile	*Google Nexus 5X*	4G	1
Mobile	*Motorola Moto G4*	3G	3

Com base nos resultados medidos pela ferramenta, onde foram utilizados para tal os dispositivos mencionados na tabela anterior, o *Calibre* registou o valor da métrica individual obtida, fornecendo depois uma página destinada a verificar as médias dessa métrica entre o período de uma semana até os seis meses, comparado sempre com o anterior período.

Cada *website* teve três páginas monitorizadas: a Página Inicial (*Homepage*), a PLP e a PDP. Na Tabela 11 apresenta-se a média desse valor por página, que representa a média do valor obtido nos perfis utilizados para o teste, bem como o valor global sendo a média dos valores obtidos para as três páginas.

Tabela 10. Médias gerais das métricas monitorizadas pelo *Calibre*

	Média			
Métricas	**Homepage**	**PLP**	**PDP**	**Global**
Time to First Byte (s)	0,626	0,713	0,722	0,687
SpeedIndex (s)	6,068	4,198	5,097	5,121
First Meaningful Paint (s)	3,468	2,700	3,691	3,286
Load Time (s)	7,620	4,376	4,741	5,579
First Contentful Paint (s)	2,446	1,788	2,697	2,310
First CPU Idle (s)	7,426	4,642	6,719	6,262
First Paint (s)	2,427	1,788	2,199	2,138
Time to Interactive (s)	8,197	5,917	7,389	7,168
Visually Complete (s)	11,054	6,589	8,913	8,852
Visually Complete 85% (s)	6,825	4,521	5,562	5,636
Lighthouse Accessibility	86,16%	84,05%	85,53%	85,25%
Lighthouse Best Practices	88,16%	87,47%	87,00%	87,54%
Lighthouse Performance	70,21%	75,84%	71,26%	72,44%
Lighthouse PWA	41,50%	43,19%	41,50%	42,06%
Lighthouse SEO	92,74%	92,63%	93,68%	93,02%
Número de pedidos *(un)*	85	100	86	90

Os dados do tipo RUM gravados na ferramenta *New Relic*, tal como já foi descrito, consistem nos dados recolhidos pela monitorização da solução implantada e que representam valores que os utilizadores experienciam. As métricas medidas pelo *New Relic* não são totalmente compatíveis com as que o *Calibre* registou, pelo que se recolheu apenas informação a partir de um pequeno conjunto de métricas que assumem uma maior importância. A Tabela 11 apresenta uma listagem dessas métricas selecionadas utilizadas para a definição do *performance budget* organizacional, cujos valores dizem respeito a resultados provenientes da monitorização de dezassete dos *websites* desenvolvidos por FPS que estão a comunicar com o *New Relic* e cujos valores obtidos dizem respeito ao período entre 30 de setembro a 7 de outubro de 2019, o valor máximo possível para se colocar na ferramenta *New Relic*.

Tabela 11. Médias das métricas mais relevantes recolhidas do *New Relic*

Métricas		**Média (s)**
Tempo de carregamento de Página (*Load Time*)	Página Inicial	5,96
	PLP	7,77
	PDP	8,39
	Todas as páginas	5,89
First Paint		3,25
First Contentful Paint		3,28
Time to First Byte		1,73
Duração *DOM Processing*		1,50
Duração *Page Rendering*		2,44

Com base nos valores recolhidos e expostos nas tabelas anteriores, procedeu-se ao cálculo dos valores médios entre os dados laboratoriais e os disponíveis de utilização real, para as métricas definidas como as mais relevantes para se definir o *performance budget* organizacional, conforme demonstrado na Tabela 7.

Assim, perante os cálculos efetuados, é proposto que a organização utilize o seguinte *performance budget*:

Tabela 12. Proposta do *performance budget* organizacional

Métricas	**Média**
Time to First Byte (ms)	1207
SpeedIndex (ms)	5121
First Meaningful Paint (ms)	3286
Load Time (ms)	5734
First Contentful Paint (ms)	2795
First CPU Idle (ms)	6262
First Paint (ms)	2694
Time to Interactive (ms)	7168
Visually Complete (ms)	8852
Visually Complete 85% (ms)	5636
Lighthouse Accessibility (%)	85,25
Lighthouse Best Practices (%)	87,54
Lighthouse Performance (%)	72,44
Lighthouse PWA (%)	42,06
Lighthouse SEO (%)	93,02
Número de pedidos (un)	90

Com base nestes dados, é possível definir-se um ficheiro de configuração JSON para ser utilizado em conjunto com o protótipo desenvolvido (cf. secção 4.1.3).

Não obstante, é possível também:

- Utilizar estes valores para poder definir novos tipos de alertas e limites para a monitorização efetuada pelo *Calibre* (cf. secção 2.8.2);
- Permitir configurar o *New Relic* (cf. secção A.3), de modo a:
    - Defiinir os limites de aviso e alerta nos *dashboards* apresentados para as métricas definidas no *performance budget*;
    - Permitir configurar alertas enviados para ferramentas de conversação (*Slack*) e ou através de *emails* para listas de distribuição de mensagens dentro da organização.

4.3 Proposta de Utilização

O protótipo desenvolvido pretende responder às necessidades básicas inicialmente detetadas como motivação para a realização deste projeto. Deste modo este protótipo pode ser utilizado das seguintes formas:

- Por um desenvolvedor, para efetuar testes de desempenho iniciados manualmente no seu computador;
- Integrado numa *pipeline* de CI/CD (cf. secção A.2), permitindo controlar o resultado final da mesma.

Para este protótipo ser utilizado por um desenvolvedor, este necessita de instalar o módulo do protótipo no seu computador, disponível no repositório de *software* da organização – o *ProGet*, através do sistema de instalação de módulos do *Node.js* – o *npm*, pela execução da seguinte instrução no terminal do computador:

```
npm i -g @bw/fps-performance
```

Ao executar a instrução anterior, o protótipo fica instalado globalmente no seu computador, tendo apenas que assegurar que possui os seguintes pré-requisitos:

- *Docker Desktop* (cf. secção 3.1.4);
- *Node.js* (versão LTS ou superior) (cf. secção 3.1.4).

No entanto, não é estritamente necessário instalar na máquina do desenvolvedor, sendo possível que este módulo esteja instalado em qualquer projeto. Deste modo, sempre que se instalar todas as dependências requisitadas por um projeto, este módulo também é instalado e é possível executar todos os comandos suportados (cf. secção 4.1.1), ficando o protótipo apenas disponível para executar junto do projeto em questão.

No molde exposto anteriormente, também torna possível a sua integração com *pipelines* pois, uma vez que se pode encontrar instalado num projeto, sempre que uma *pipeline* for executada, todas as dependências são instaladas. Por este motivo, é possível criar um passo numa *pipeline* responsável por executar um conjunto de testes de desempenho através dos comandos disponibilizados e verificar se esses cumprem o *performance budget* definido no âmbito do projeto, disponível num ficheiro JSON (cf. secção 4.1.2). Dependendo da aplicação desejada, é possível a *pipeline*:

- ser responsável por apresentar todos os resultados medidos e devolvidos pelo protótipo, sem executar qualquer tipo de ação adicional;

- além de apresentar os resultados, ser responsável por determinar que uma *pipeline* vai falhar, caso o protótipo indique que hajam métricas que não cumprem o *performance budget* definido (especialmente útil para o caso de se querer garantir que determinadas versões de *websites* tenham obrigatoriamente que cumprir um dado *budget*, antes de ser implantada para o público-geral).

Este protótipo foi desenvolvido com recurso a tecnologias *open-source*, cujas dependências possuem licenças do tipo *Apache 2.0*, *BSD-2-Clause Simplified*, ISC ou MIT. Estes tipos de licenças são conhecidos por autorizarem a modificação do seu código-fonte, o uso privado e distribuição, desde que uma cópia da licença e os direitos de autor sejam devidamente incluídos no novo *software*, no entanto não fornecem qualquer tipo de garantia nem assumem qualquer responsabilidade oriunda da utilização dessas dependências.

Tendo isto em consideração, esta aplicação encontra-se preparada para ser publicada num repositório de módulos *Node.js* pública, ficando assim disponível para ser utilizada por qualquer pessoa que proceda a instalação deste módulo.

Para tal ser feito, existe só a necessidade de obter autorização interna por parte da organização para a respetiva publicação, assim como a remoção de duas dependências internas - `@mondrian/core-scripts` e `@mondrian/commitlint-config`. Estas dependências, quando utilizadas em conjunto, são necessárias para:

- Validar os *commits* realizados no projeto para um repositório *Git* segundo a norma *Conventional Commits* (cf. secção A.1);

- Permitir a criação e publicação de novas versões do protótipo no repositório de módulos internos da *Farfetch* – o *ProGet*, e criação da respetiva versão no projeto *Git* (cf. secção A.1).

4.4 Sumário

Com base no conteúdo presente nesta secção, foi possível verificar diversos aspetos tidos em consideração relativos à implementação do protótipo, mas também associados ao estudo e análise efetuada para a definição de um *performance budget* organizacional.

A definição deste *performance budget* organizacional é importante para o protótipo em questão, visto ser um dos requisitos de modo a poder avaliar e classificar o desempenho, mas também permitiu analisar quer dados sintéticos como de utilização real, pela combinação de diversas fontes de informação existentes na organização, podendo assim tirar partido dos dados disponíveis e assim poder utiliza-los de uma forma preventiva e não reativa, como normalmente tem sido a prática no dia-a-dia das equipas de desenvolvimento.

A adoção de boas práticas de *design* e a seleção das bibliotecas mais adequadas permitiram criar este protótipo de tal forma que pode ser disponibilizado publicamente, conforme mencionado na secção 4.3, considerando que as dependências utilizadas são todas *open-source* e assim permite que este projeto possa servir futuramente como um contributo para a comunidade.

5 Avaliação

O presente capítulo descreve o processo de avaliação do *software* a desenvolver, que terá como objetivo examinar a eficácia e o nível de satisfação dos utilizadores face ao protótipo desenvolvido, no seu estado final. A avaliação terá em conta os interesses das diversas partes interessadas no projeto – *stakeholders*:

- As equipas de desenvolvimento da unidade de negócio FPS - colaboradores da *Farfetch* que pretendem obter informações que lhes permitam saber se as soluções desenvolvidas cumprem um padrão mínimo de desempenho;

- A unidade de negócio e os clientes da plataforma tecnológica desenvolvida pelas equipas que pretendem ter soluções que possuam um desempenho adequado junto dos seus utilizadores, de forma a contribuir para a melhoria da UX;

- O cliente/utilizador final, que pretende usufruir de uma melhor experiência de utilização e satisfação cujo desenvolvimento dotou de cuidados orientados a um adequado desempenho.

5.1 Abordagem

Nesta secção, será descrita a abordagem específica a adotar na avaliação dos interesses de cada uma das partes interessadas.

5.1.1 **Equipas de desenvolvimento**

O interesse deste público-alvo passa por compreender se a monitorização e análise do desempenho das soluções *web* desenvolvidas foi conseguida, ou seja, se fornece informação relevante sobre o estado da mesma e se pode conduzir a otimizações manuais (cf. secção 1.4)

e mais consciência na necessidade de otimização do desempenho de *websites* devido aos dados obtidos.

De forma a aferir se o projeto vai de encontro ao interesse das equipas de desenvolvimento, foi realizado um inquérito de satisfação a membros da equipa de desenvolvimento, com especial atenção aos que acompanharam o desenvolvimento do projeto, bem como os que irão usufruir diretamente com os resultados do projeto, utilizando indicadores capazes de verificar em que medida é que o projeto foi satisfatório, e se promove o desenvolvimento de uma cultura que privilegie o desempenho das soluções desenvolvidas.

5.1.2 *Product, Delivery e Service Managers*

De modo a avaliar se projeto vai de encontro às necessidades e interesses da unidade de negócio, ou seja, os colaboradores de FPS que lidam diretamente com os clientes (marcas e boutiques que operam *websites* de *e-commerce*), nomeadamente os *Product Managers*, *Delivery Managers* e *Service Managers*, pretende-se verificar se:

- A informação fornecida pelo protótipo contribui para acompanharem mais de perto a evolução do desempenho de um *website* durante as fases de desenvolvimento e de manutenção junto das marcas e boutiques;

- Se os dados fornecidos pelo projeto contribuem para o aconselhamento dos clientes a investir em otimizações de desempenho nos *websites*, de modo a melhorar a experiência de utilização, pela demonstração da correlação de valores das métricas de desempenho ótimas com o aumento de KPI relacionados com o aumento do volume de negócio;

- Conseguir apresentar dados reais que comprovem junto de um cliente que uma nova versão de um *website* tem um desempenho melhorado.

Apesar de esta ser uma parte interessada, considerando a existência de limitações para a implantação imediata do projeto, não foram realizadas quaisquer avaliações para este conjunto de *stakeholders*, por não se reunir todas as condições necessárias para se aferir o cumprimento das necessidades desta parte interessada.

5.1.3 **Consumidor**

De forma a poder verificar de o projeto realizado vai de encontro aos interesses dos consumidores, ou seja, os utilizadores que acedem às soluções *web* criadas pela unidade de negócio FPS serão avaliadas com recurso às ferramentas monitorização real de desempenho de *websites* - através da ferramenta *New Relic* (cf. secção A.3).

Deste modo, deseja-se aferir e visualizar graficamente as tendências associadas às métricas de desempenho medidas por estas ferramentas, e concluir se uma solução *web* que foi alvo de

análise de desempenho pelo protótipo desenvolvido e usufruiu de novos desenvolvimentos para privilegiar a melhoria do desempenho, permitiu melhorias na experiência de utilização no lado do consumidor.

Os consumidores não foram alvo da avaliação indicada, pelo facto de nenhum *website* ter sido alvo de um processo de melhoria de desempenho durante a fase de desenvolvimento, por influência da ferramenta.

No entanto, através da realização de testes de desempenho internos dentro da organização, e através da adaptação do *performance budget* com base nos dados recolhidos do *New Relic* indirectamente estão a ser avaliados os interesses do consumidor – a garantia de boa experiência de utilização de um *website* ao fornecer um desempenho adequado.

5.2 Preparação de experiências

Conforme mencionado anteriormente, foram preparadas experiências para avaliar o projeto, destinada a membros das equipas de desenvolvimento. Para tal, foram criados:

- Criação de um guião para efetuar o teste do protótipo desenvolvido;
- Elaboração de Inquéritos de satisfação para os intervenientes no processo de avaliação da solução efetuada, de modo a compreender se estão satisfeitos com o projeto desenvolvido em prol de FPS, e se lhes fornece informação útil.

5.2.1 Guião

Um guião foi integrado no inquérito de satisfação, de modo a fornecer indicações sobre o que deveria ser avaliado e o modo como poderem utilizar a aplicação, de forma a poderem dar a sua opinião.

Este guião foi disponibilizado aquando do término do projeto a alguns membros da equipa de desenvolvimento que se disponibilizaram para experimentar o protótipo desenvolvido em diversos contextos para possíveis utilizações/aplicações. Este guião foi apresentado numa sessão, em que foi dado o contexto da realização do projeto e apresentados alguns conceitos importantes a conhecer antes de utilizar o protótipo, de modo a simplificar e clarificar o processo de testes e compreensão do propósito do projeto, e avaliar de perto como os intervenientes utilizaram a ferramenta.

O guião criado fornece indicações e guia os intervenientes pelos seguintes passos:

- Instalação do protótipo na sua máquina de trabalho;

- Utilização do comando `calibre` do protótipo para avaliar um *website* que se encontra *online*;

- Utilização do comando `calibre` do protótipo para avaliar um *website online*, mas que que se encontra protegido por autenticação (representando um *website* no estado imediatamente antes de ser disponibilizado aos consumidores – estado *pre-live*);

- Utilização do comando `webpagetest` do protótipo para avaliar um *website* apenas disponível na Intranet, com uma instância privada do *WebPageTest*;

- Utilização do comando `webpagetest` do protótipo para avaliar um *website* que se encontra *online*, a utilizar a instância pública do *WebPageTest*;

Figura 39. Parte introdutória do guião distribuído pelos *Stakeholders* alvo de avaliação

5.2.2 **Inquéritos**

Com a realização de questionários é possível apurar respostas que permitam avaliar aspetos do trabalho desenvolvido. Optou-se por adotar um tipo de inquérito simples que permita ouvir e seguir as sugestões dos inquiridos. Estes inquéritos permitem obter a informação necessária dos clientes e utilizadores do produto alvo de avaliação (Mateus, 2017).

Deste modo, e com vista a obter um inquérito que produza resultados fiáveis, este deve adotar as seguintes características:

- Ser curto, simples e objetivo, que leve o inquirido a responder apenas ao essencial;
- Ter poucas questões por página, de modo a evitar que o inquirido perca o seu foco;
- Ter uma estimativa de tempo de resposta entre 10 a 20 minutos, tendo em conta a necessidade da execução de testes de desempenho, o que pode levar algum tempo;
- Ser realizados com recurso a uma ferramenta *online* – como o *Google Forms* (disponível internamente na organização), de modo a poder partilhar facilmente os resultados com o público-alvo e ser possível observar a evolução dos mesmos;
- Ser um inquérito misto - que permita questões de resposta aberta e de resposta fechada, onde o inquirido tem a liberdade de estruturar a sua resposta ou onde está limitado às opções de resposta patentes no questionário, respetivamente (Mateus, 2017).

5.2.3 **Público-alvo**

Ao conceber um questionário e avaliação de uma aplicação através de testes, um dos aspetos a considerar é o público-alvo. A correta definição deste, leva a que as suas perguntas e processos para teste sejam idealizados com maior grau de detalhe e com a linguagem e conceitos mais ajustados (Mateus, 2017).

Esta secção pretende apresentar as características do público-alvo do *stakeholder* que participou na avaliação deste projeto.. Por grande parte do público-alvo identificado estar familiarizado com termos e expressões informáticas, os métodos de avaliação neste estudo podem ser mais objetivos e rigorosos, pois permitiram obter respostas mais críticas, o que se revela vantajoso por permitir conhecer pontos de melhoria da aplicação, assim como retirar conclusões sobre o trabalho realizado.

As equipas de desenvolvimento referidas são constituídas por colaboradores da Farfetch, alocados à unidade de negócio FPS, onde o projeto se insere. O principal interesse desta parte interveniente, de modo a poder concluir se o projeto realizado é-lhes útil centra-se:

- Em dispor de um acesso facilitado a um conjunto de ferramentas (organizacionais ou não), que lhes permita avaliar e conhecer qual o desempenho das soluções *web* desenvolvidas;
- Permitir mecanismos inteligentes que possam ser integrados com as suas ferramentas utilizadas do dia-a-dia, que permita garantir o desempenho dos conteúdos desenvolvidos;

- Estar ciente da importância do desempenho e conhecer o estado de desempenho dos *websites* FPS de uma forma geral.

Os intervenientes neste questionário detêm as seguintes características:

- Idades compreendidas entre os 22 e os 35 anos de idade;
- Colaboradores da *Farfetch*, enquadrados na unidade de negócio FPS, nas quais assumem a função de *Frontend Engineers* (categorizados ora como juniores, intermédios ou seniores);
- Com formação na área da Engenharia Informática ou similar;
- Manifestam interesse em tecnologia e tem especial interesse pela área de desempenho;
- Origem Portuguesa;
- Idioma Português.

5.3 Resultados

Na sessão realizada para a avaliação do protótipo e resposta ao inquérito preparado participaram 5 pessoas. O questionário tinha 45 questões em que:

- 12 questões destinavam-se a identificar o tipo de experiências realizadas pelos inquiridos;
- 4 questões tinham a opção sim/não, de modo a avaliar se o teste de desempenho que inquirido realizou cumpriu o *performance budget* definido;
- 3 questões eram de resposta aberta, de modo a recolher a opinião, sugestões de melhoria e observações dos inquiridos;
- As restantes questões pretendiam avaliar o grau de satisfação do inquirido relativamente a diversos aspetos da experiência de utilização, com um valor de escala de 1 a 5. De acordo com a pergunta efetuada a escala representou:
    - Nível 0 “Nada Satisfeito” e nível 5 “Muito Satisfeito”;
    - Nível 0 “Nada Útil” e nível 5 “Muito Útil”;
    - Nível 0 “Discordo Totalmente” e nível 5 “Concordo Totalmente”.

Com recurso à ferramenta *Google Forms* para a realização deste questionário, foram construídos gráficos de modo a possibilitar a compreensão e sistematização dos resultados obtidos, de como chegar a possíveis conclusões de limitações e melhorias futuras.

5.3.1 Experiência nº 1: Utilização do *Calibre* em website *online*

Para as questões efetuadas relativamente à experiência nº 1, dedicada à utilização do comando `calibre` aplicado a um *website online*, foram obtidos os seguintes resultados:

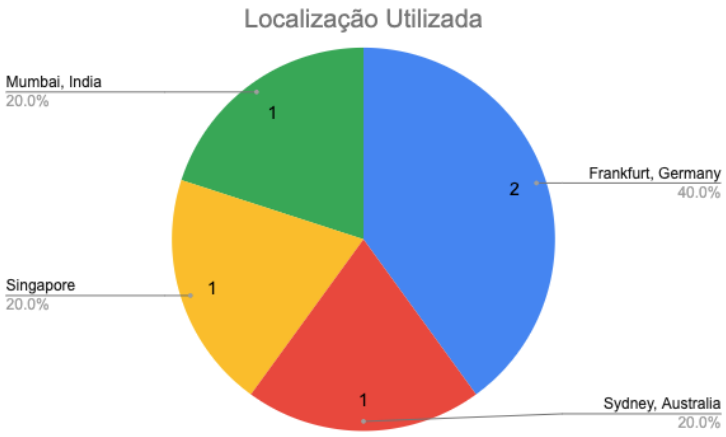


Figura 40. Localizações utilizados na experiência nº1

Figura 41. Dispositivos utilizados na experiência nº1

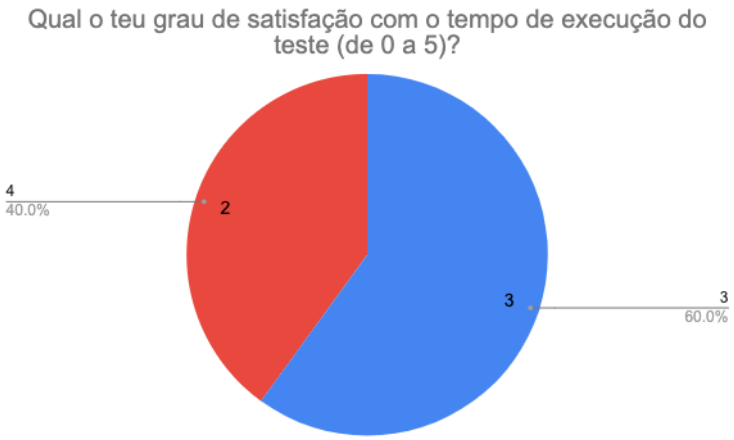


Figura 42. Grau de satisfação com o tempo de execução do teste na experiência nº1

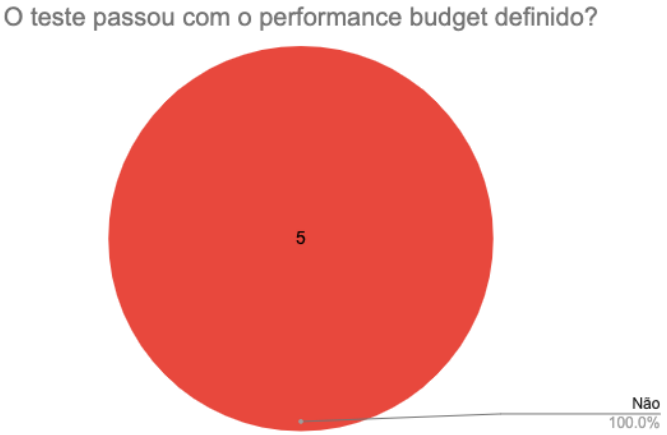


Figura 43. Número de testes que cumpriram o *performance budget* na experiência nº1

5.3.2 **Experiência nº 2: Utilização do *Calibre* com *website* em *prelive***

Para as questões efetuadas relativas à experiência nº 2, dedicada à utilização do comando `calibre` aplicado a um *website* em *prelive* (ou seja *online,* mas protegido com autenticação e não-acessível ao público), foram obtidos os seguintes resultados:

Figura 44.Dispositivos utilizados na experiência nº2

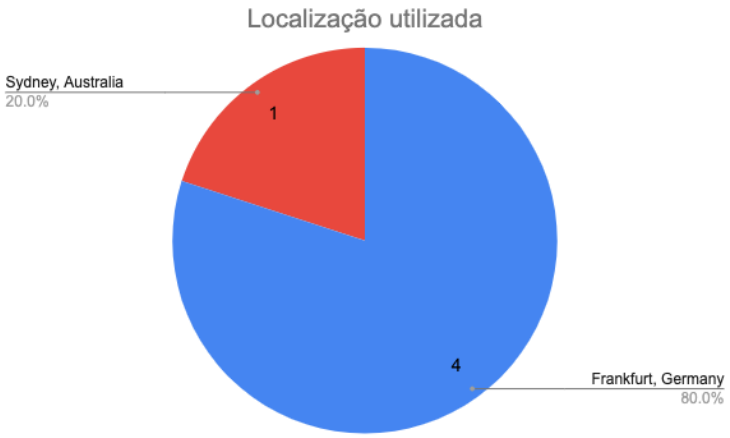


Figura 45. Localizações utilizadas na experiência nº2

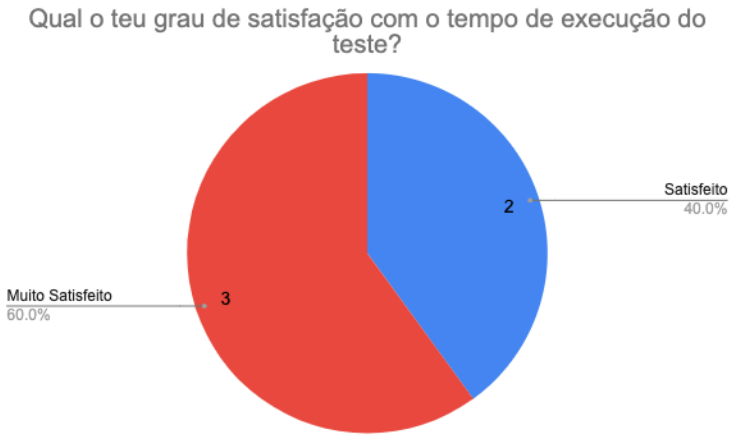


Figura 46. Grau de satisfação com o tempo de execução do teste na experiência nº2

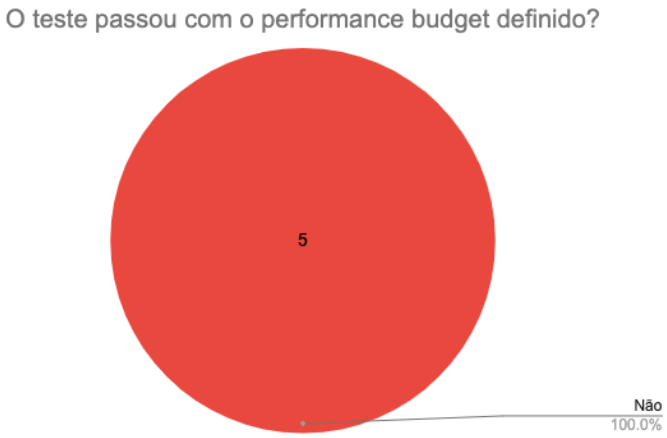


Figura 47. Número de testes que cumpriram o *performance budget* na experiência nº2

5.3.3 ***Feedback* geral *Calibre***

Relativamente às questões efetuadas associadas às experiências nº 1 e nº 2, dedicadas à utilização do comando `calibre` nos contextos idealizados, foram obtidos os seguintes resultados:

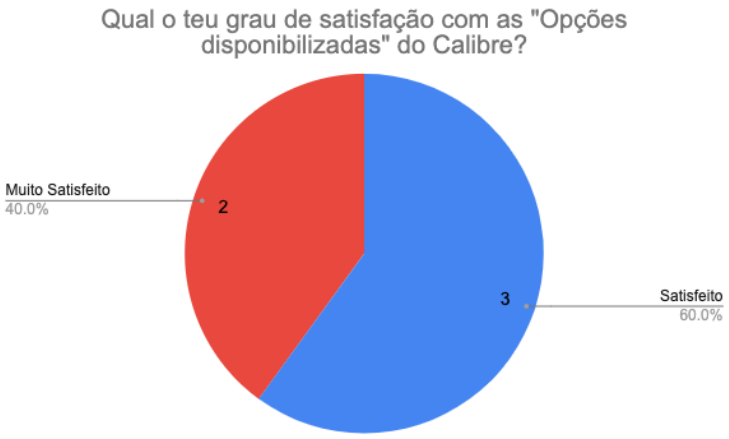


Figura 48. Grau de satisfação com as opções disponibilizadas pelo comando *calibre*

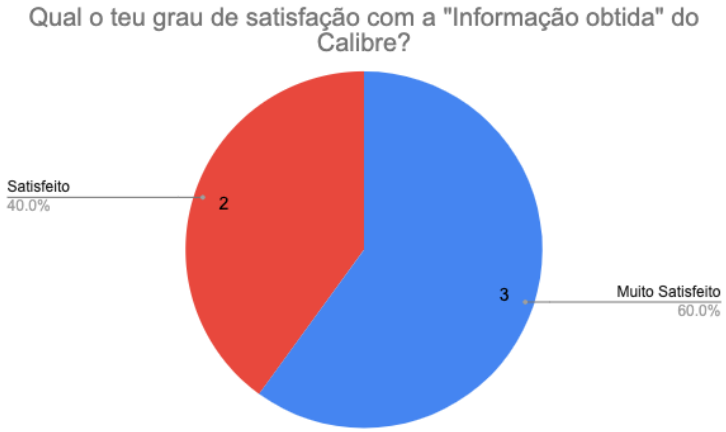


Figura 49. Grau de satisfação com a informação fornecida pelo comando *calibre*

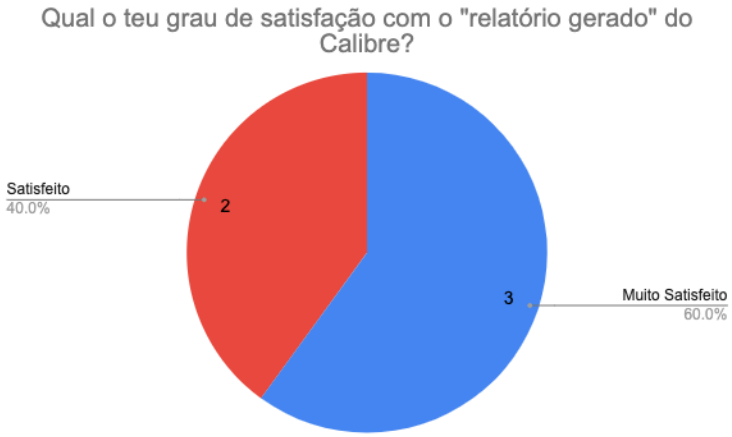


Figura 50. Grau de satisfação com o relatório gerado pela ferramenta *Calibre*

Figura 51. Grau de satisfação com a compreensão do relatório gerado pelo *Calibre*

Figura 52. Grau de satisfação com o tempo de execução dos testes no *Calibre*

Figura 53. Grau de satisfação quanto à utilidade do comando *Calibre*

5.3.4 Experiência nº 3: Utilização do *WebPageTest* num *website* interno com a instância privada

Para as questões efetuadas relacionadas com a experiência nº 3, dedicada à utilização do comando `webpagetest` aplicado a um website privado, apenas disponível na *Intranet* da organização, foram obtidos os seguintes resultados:

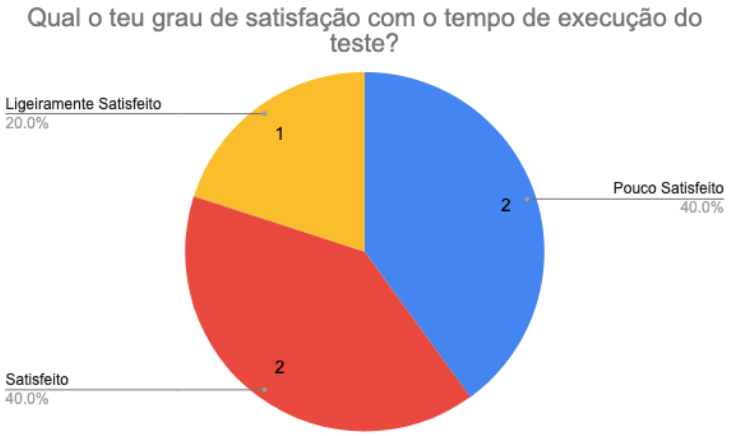


Figura 54. Grau de satisfação do tempo de execução do teste na experiência nº3

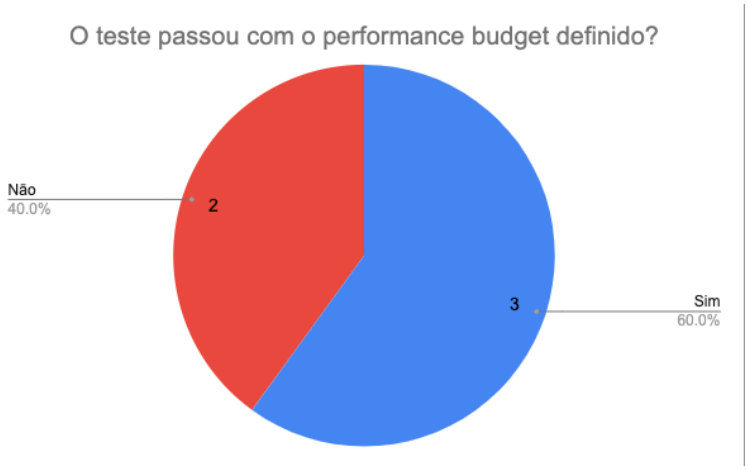


Figura 55. Número de testes que cumpriram o *performance budget* na experiência nº3

5.3.5 Experiência 4: Utilização do *WebPageTest* num website *online* com a instância pública

Para as questões efetuadas relacionadas com a experiência nº 4, dedicada à utilização do comando `webpagetest` aplicado a um website *online*, utilizando para tal a instância pública do *WebPageTest*:

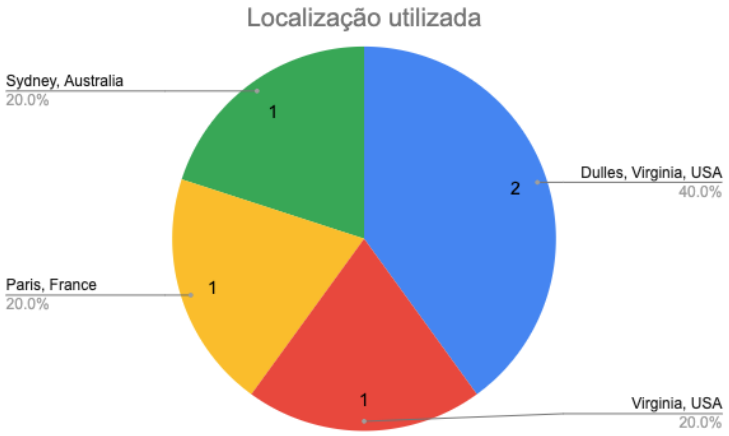


Figura 56. Localizações utilizadas na experiência nº4

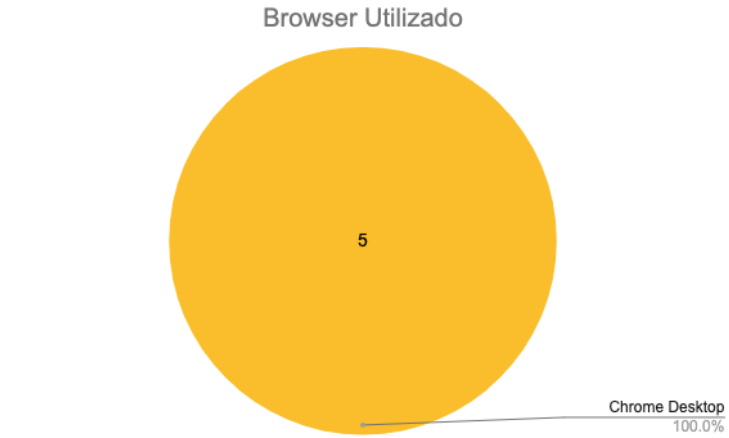


Figura 57. *Browsers* utilizados na experiência nº4

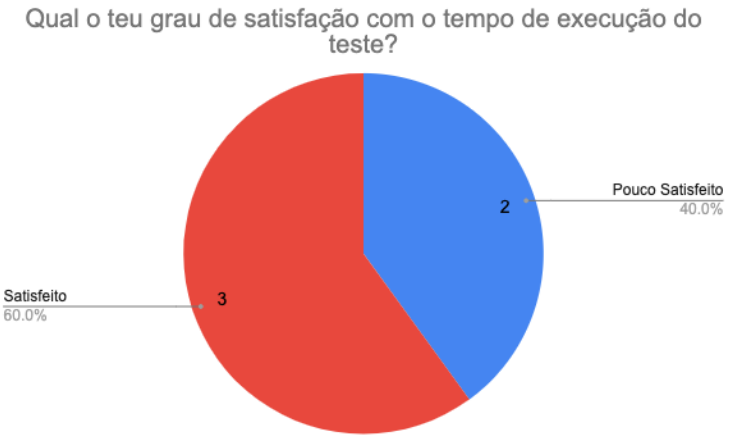


Figura 58. Grau de satisfação com o tempo de execução do teste da experiência nº4

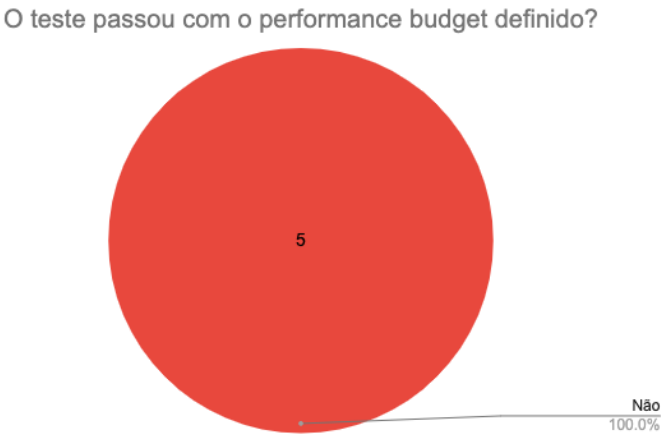


Figura 59. Número de testes que cumpriram o *performance budget* da experiência nº4

5.3.6 ***Feedback* geral *WebPageTest***

Relativamente às questões efetuadas associadas às experiências nº 3 e nº 4, dedicadas à utilização do comando `webpagetest` nos contextos idealizados, foram obtidos os seguintes resultados:

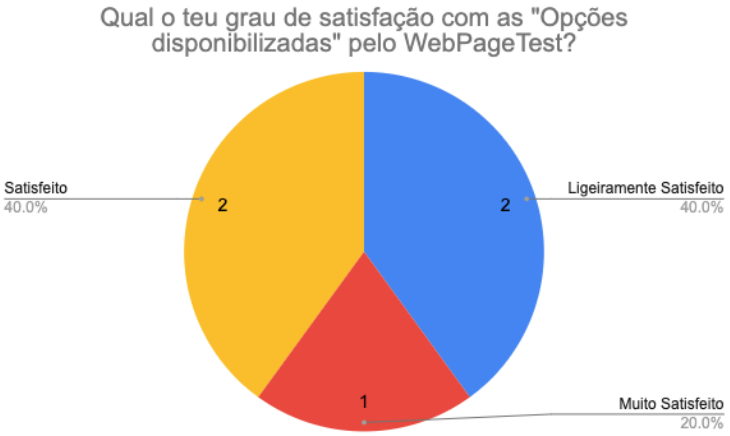


Figura 60. Grau de satisfação com as opções disponibilizadas pelo *WebPageTest*

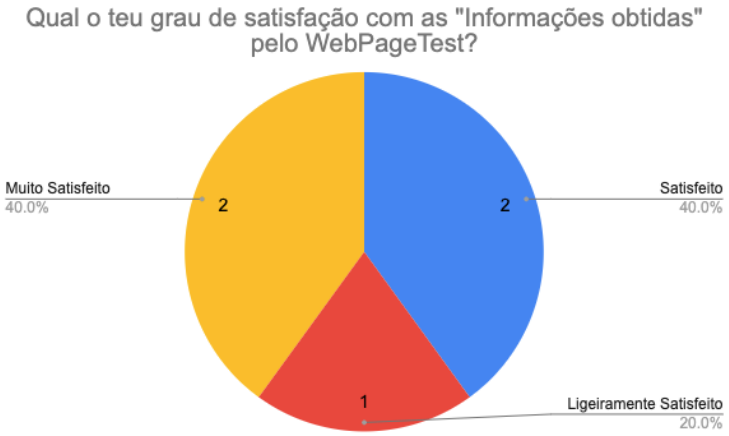


Figura 61. Grau de satisfação com as informações obtidas do *WebPageTest*

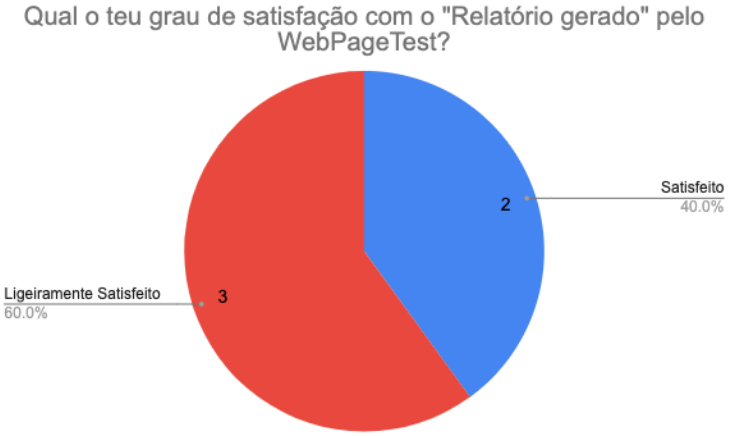


Figura 62. Grau de satisfação com o relatório gerado pelo *WebPageTest*

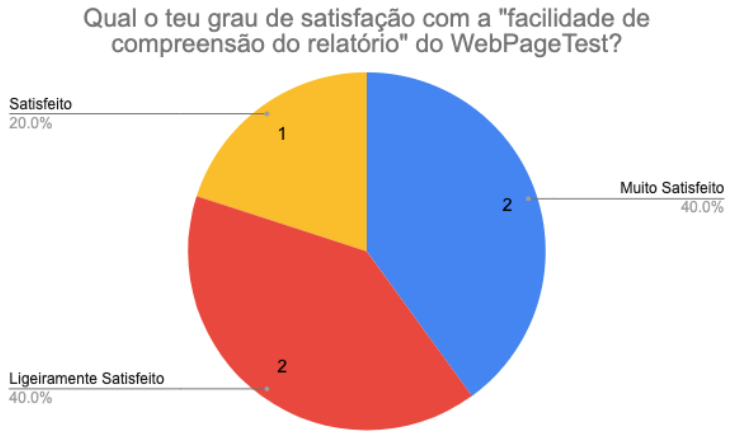


Figura 63. Grau de satisfação com a facilidade de compreensão do relatório do *WebPageTest*

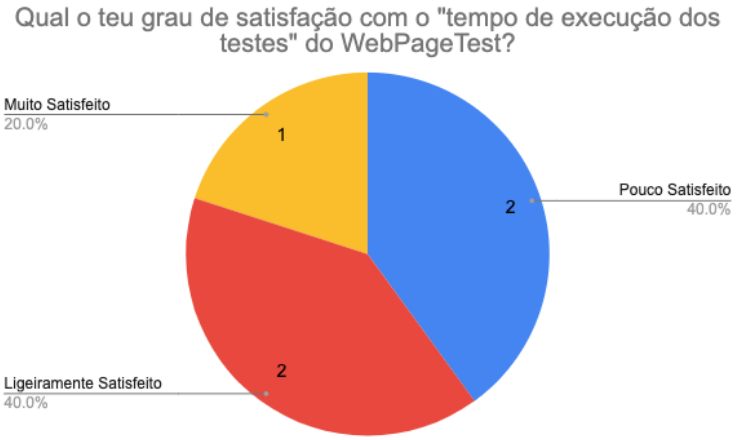


Figura 64. Grau de satisfação com o tempo de execução dos testes do *WebPageTest´*

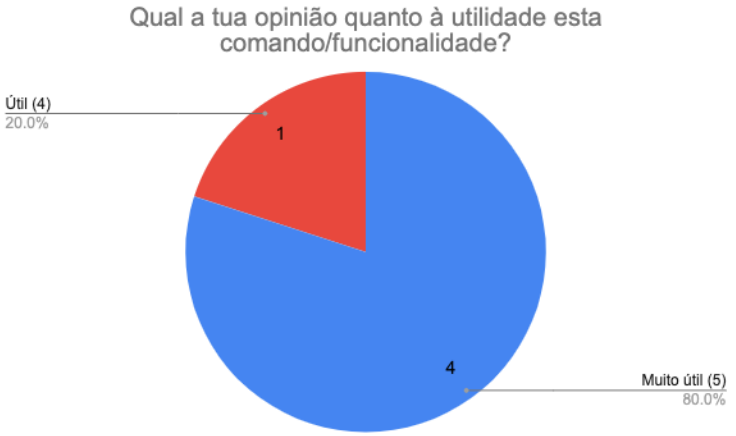


Figura 65. Grau de satisfação relativamente à utilidade do comando *webpagetest*

5.3.7 ***Feedback*** **geral protótipo**

Após as questões relativas a cada experiência e comando utilizado, foram efetuadas questões de modo a avaliar o grau de satisfação sobre o protótipo de uma forma global. Foram obtidos os seguintes resultados:

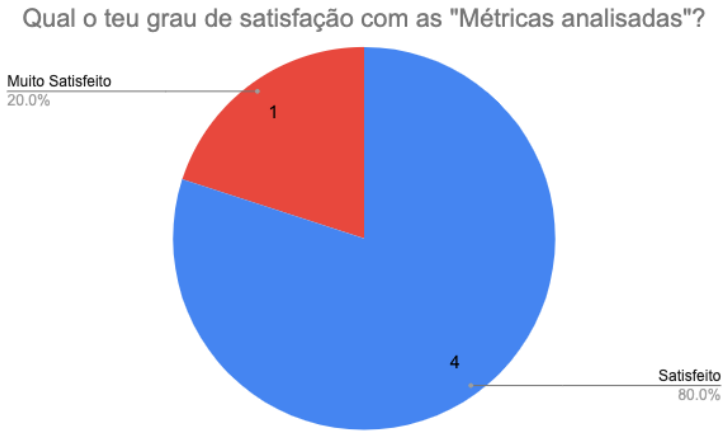


Figura 66. Grau de satisfação com as métricas analisadas pelo protótipo

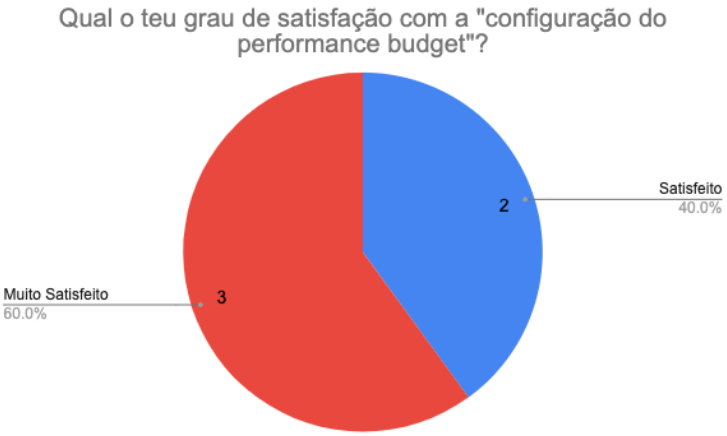


Figura 67. Grau de satisfação com a configuração do *performance budget* do protótipo

Figura 68. Grau de satisfação com a seleção de dispositivos do protótipo

Figura 69. Grau de satisfação com a seleção de localizações no protótipo

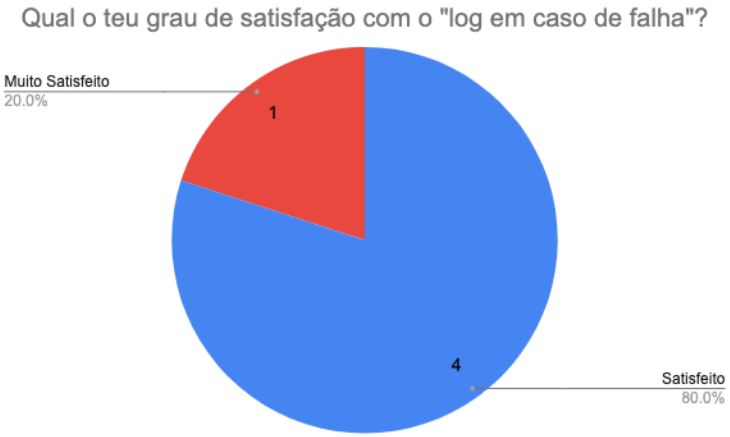


Figura 70. Grau de satisfação com o *log* em caso de falha do protótipo

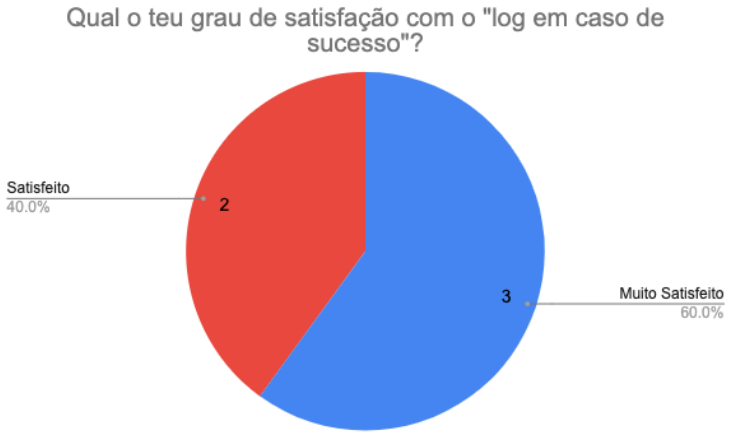


Figura 71. Grau de satisfação com os *logs* em caso de sucesso no protótipo

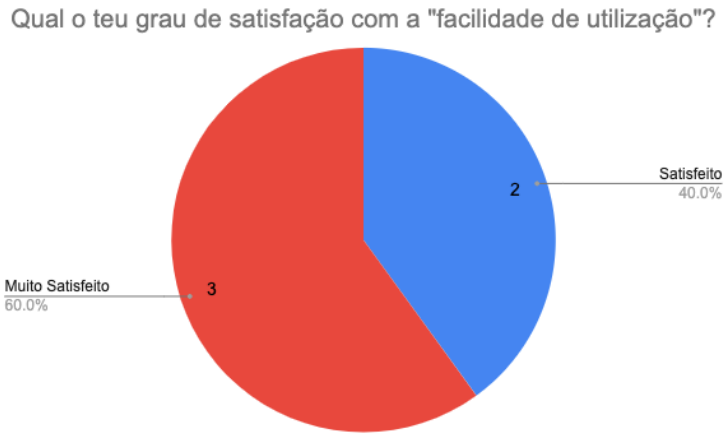


Figura 72. Grau de satisfação com a facilidade de utilização do protótipo

Figura 73. Grau de satisfação com a disponibilização de acesso aos relatórios gerados pelas ferramentas

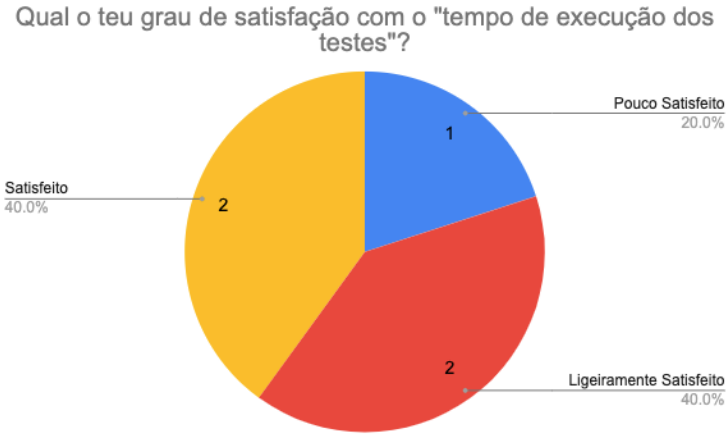


Figura 74. Grau de satisfação com o tempo de execução dos testes no protótipo

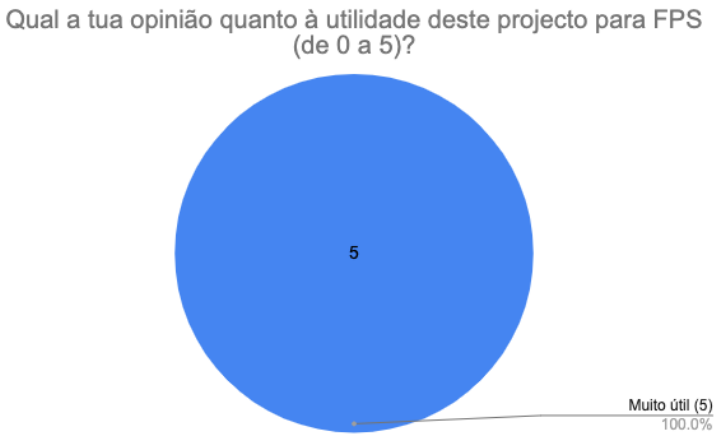


Figura 75. Opinião relativamente à utilidade do protótipo para FPS

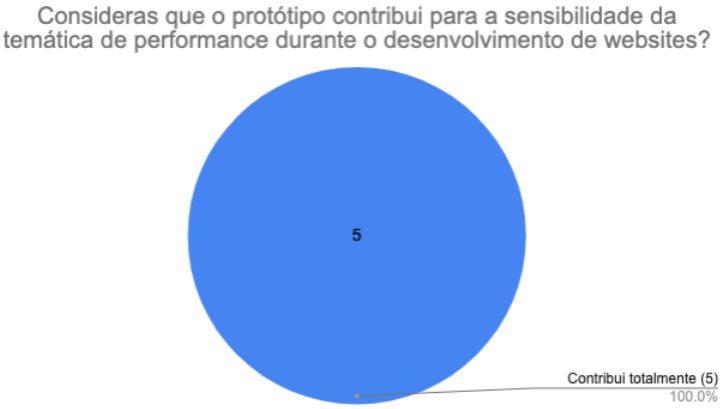


Figura 76. Opinião relativamente à sensibilização da importância do desempenho criada pela utilização do protótipo

5.3.8 **Questões de resposta aberta**

Relativamente à questão para dar uma opinião geral sobre a ferramenta em questão, os inquiridos responderam:

- "Penso que seja uma excelente iniciativa que pode prevenir problemas no futuro e melhorar a experiência dos nossos utilizadores se for tido em consideração as indicações apresentadas pela ferramenta.";

- "Considero que este projeto é bastante útil, contribuindo para a sensibilização do tema performance na equipa e conseguindo explicar todas as métricas de forma explícita e bem construída. Ter em conta a performance nos projetos é uma preocupação que deve ser tida em conta desde o seu desenvolvimento, por isso, o facto de esta aplicação permitir ter todas as possibilidades juntas num único projeto leva à facilitação de trabalho e a que o seu uso seja mais facilitado. Considero que este projeto é uma mais-

valia e que deve ser bem partilhado para chegar ao maior número de pessoas da equipa. Parabéns pelo projeto."

- “Acho que isto tem um enorme potencial para FPS, nomeadamente com a integração nas *pipelines*, que vai permitir "forçar" uma *performance* aceitável.”

- “Acho que é uma ferramenta imprescindível para a entrega de produtos com uma maior atenção à qualidade de performance.”

À questão para fornecer sugestões de melhoria para ferramenta em questão, os inquiridos responderam:

- “Uma aplicação com uma experiência melhor para os testes”;

- "Como melhoria considero que enquanto está a correr os testes a ferramenta pode demonstrar alguma informação de carregamento e que está a calcular valores e apresentar alguns valores finais à medida que são calculados, ao invés de apresentar tudo no final.”

- “Tenho apenas 3 sugestões:

    - apresentar no relatório final o tempo decorrido durante o teste (a duração do teste);

    - dar *feedback* visual enquanto o teste está a correr (tipo um *loader*) para o utilizador saber que o CLI não está “*freeze*”;

    - quando temos que introduzir *password* para correr um teste, deveria ser via *prompt* de password para não se ver o que estamos a escrever.”

- “Era interessante criar mais hipóteses de exportação de dados, talvez criar um PDF e na pipeline acabar por enviar esse PDF a responsáveis do projeto (nomeadamente quando forem entregas em master). Se possível também era interessante incorporar isso no *GitLab*. Outra sugestão é que ao incorporar procurar não meter esta ferramenta na pipeline principal, podíamos despoletar outra pipeline exclusiva para fazer estes testes visto que são mais vagarosos e dependentes de serviços externos como no exemplo do *WebPageTest*, visto isso se incorporamos na *pipeline* principal acabará por ser um entrave para a entrega rápida que muitas vezes temos de fazer. Acho que é uma ferramenta que nos informa e deverá decidir ações futuras no nosso *backlog* mas que não deverá impedir uma *feature* de ir para a frente.”

Para o campo disponibilizado para fornecer algumas observações, os inquiridos responderam:

- “Os testes demoram um bocado, mas acredito que seja porque os dispositivos e localização escolhida não sejam as que disponibilizam maior performance. Ainda assim, considero que o facto de o protótipo medir todos os parâmetros de performance budget e ainda os avaliar leva a que o tempo demorado a correr os testes seja bastante bem conseguido.”;

- “Nada mais a assinalar.”;

- “Devemos incorporar esta ferramenta nas nossas pipelines e procurar notificar o criador do *Merge Request* com os resultados dos testes, não acho que a restrição de entrega na conceção de novos *tenants* deva ser feita, mas poderá ser uma ferramenta restritiva na manutenção dos já entregues. Poderá ser uma ferramenta útil para perceber como os *tenants* pós deixarem a equipa que os criou e sendo entregues a outras que são encarregues da sua manutenção e algumas implementações estarão a beneficiar ou a prejudicar o produto inicialmente entregue.”

5.3.9 **Satisfação com a ferramenta *Calibre***

Nas experiências realizadas com o *Calibre*, foi a ferramenta em que os utilizadores mais experienciaram funcionalidades e opções diversas, como é o cado das várias localizações e dispositivos móveis.

De um modo geral, os utilizadores afirmaram estar satisfeitos com o tempo de execução dos testes nesta ferramenta, especialmente no caso da experiência nº 2, que atingiu um grau de satisfação de 60%.

Contudo, nenhum dos testes executados nesta ferramenta passou no *performance budget* definido, mesmo considerando que o *performance budget* fornecido para cada uma das experiências variava ligeiramente, pelo que se pode constatar que os *websites* utilizados para os testes (*websites* de FPS) possuem métricas de desempenho que devem ser melhoradas.

De uma forma geral:

- 60 % dos inquiridos encontram-se satisfeitos (nível 4 de 5) com as opções disponibilizadas e com a facilidade de compreensão do relatório, enquanto que os restantes 40% encontram-se muito satisfeitos;

- 60% dos inquiridos encontram-se muito satisfeitos (nível 5 de 5) com a informação obtida pela ferramenta ao correr o comando `calibre` e com o relatório gerado pelo *Calibre* (os restantes 40% afirmaram encontrarem-se satisfeitos – nível 4 de 5);

- 80% dos inquiridos revelaram estar satisfeitos com o tempo de execução dos testes executados, enquanto que 20% revelou estar muito satisfeito;

- 80% dos inquiridos consideram que este comando este comando é muito útil.

Estes resultados mostram que os inquiridos revelaram estar satisfeitos com as funcionalidades disponibilizadas pelo *Calibre* e que não encontraram problemas de maior a executar esta experiência. No entanto, o resultado obtido no tempo de execução revela que pode existir aqui a oportunidade de otimizar o tempo de execução.

5.3.10 Satisfação com a ferramenta *WebPageTest*

Relativamente às experiências com a ferramenta *WebPageTest*, as opiniões revelam ser mais díspares.

Na experiência nº3, 40% dos inquiridos revelaram estar pouco satisfeitos com o tempo de execução do teste, outros 40% revelaram estar satisfeitos e apenas 20% revelaram estar ligeiramente satisfeitos. Este motivo pode-se dever ao facto de os testes estarem a correr nos computadores dos inquiridos, e não disporem de recursos suficientes para poderem utilizar a instância privada da melhor forma. No entanto, isto revela que é necessário pensar em formas de otimizar e acelerar o tempo deste tipo de testes, visto a instância privada utilizada na experiência nº3 ser o tipo de instância necessária utilizar para se testar os *websites* internos.

No entanto, e ao contrário do *Calibre*, 60% dos inquiridos revelaram que o seu teste na experiência nº3 cumpriu o *performance budget* definido, mas tal se deve ao facto de terem utilizado a configuração `tolerance` no ficheiro de configuração.

Na experiência nº4, relacionada com a utilização da instância pública do *WebPageTest*, 40% dos inquiridos utilizaram a localização por defeito – Dulles, Virginia, EUA – enquanto que os restantes 60% utilizaram diferentes localizações. No entanto, todos utilizaram o *browser* Google Chrome para esta experiência.

Em termos de grau de satisfação com o tempo de execução do teste da experiência nº4, 40% revelou estar pouco satisfeito (nível 2/5), enquanto que os outros 60% revelaram estar satisfeitos (nível 4/5). Esta divisão de opinião pode ser devido ao tempo de processamento por parte da instância pública do *WebPageTest*, que devido ao ser uma instância pública, podia ter outros testes a decorrer, e as experiências podem ter ficado em fila de espera, provocando *timeouts* na aplicação.

De um modo geral:

- 40% dos inquiridos revelam estar ligeiramente satisfeitos (nível 3/5) com as opções disponibilizadas, outros 40% encontram-se satisfeitos (nível 4/5) e apenas 20% se encontra muito satisfeito (nível 5/5) com as opções disponibilizadas;

- Quanto à informação obtida, 40% dos inquiridos revelam estar muito satisfeitos (nível 5/5), outros 40% encontram-se satisfeitos (nível 4/5) e apenas 20% se encontra ligeiramente satisfeito (nível 3/5) com as opções disponibilizadas;

- 60% dos inquiridos revelam estar ligeiramente satisfeitos (nível 3/5) com o relatório gerado pelo *WebPageTest*, enquanto os restantes se encontram satisfeitos (nível 4/5). Tal se pode dever ao relatório do gerado pelo *WebPageTest* não ser tão agradável de se consultar, navegar e analisar face ao relatório do *Calibre*;

- Relativamente à facilidade de compreensão do relatório, as opiniões estão divididas, podendo demonstrar a veracidade da afirmação anterior relativamente ao relatório produzido, pois 40% revelam estar ligeiramente satisfeitos (nível 3/5) e outros 40% revelam estar muito satisfeitos (nível 5/5);

- Sem sombra de dúvidas os inquiridos revelam não estar totalmente satisfeitos com o tempo de execução de testes do *WebPageTest*, quando comparado com o *Calibre*, onde 40% dos inquiridos revelam estar pouco satisfeitos (nível 2/5) e outros 40% ligeiramente satisfeitos (nível 3/5);

- 80% dos inquiridos consideram que este comando e as funcionalidades disponibilizadas são bastante úteis (nível 5/5).

Com base nos resultados obtidos, verifica-se aqui a necessidade de apostar em melhorar o tempo de execução dos testes com a ferramenta *WebPageTest*, tendo em conta as experiências não tão satisfatórias quanto ao tempo de execução dos testes desta ferramenta.

5.3.11 Satisfação global da ferramenta

Relativamente às funcionalidades gerais disponibilizadas pela ferramenta:

- 80% dos inquiridos revelam estar satisfeitos (nível 4/5) com as métricas analisadas e suportadas pela aplicação;

- 60% dos inquiridos revelam estar muito satisfeitos (nível 5/5) com o tipo de configuração fornecida pelo *performance budget*, demonstrando a grande flexibilidade conseguida no ficheiro de configuração, conforme foi possível verificar pelas experiências;

- 60% dos inquiridos mostram-se satisfeitos (nível 3/5) com as opções de configuração disponibilizadas de uma forma geral bem como as localizações fornecidas para seleção;

- Na seleção de dispositivos, a opinião encontra-se mais dividida – 40% encontram-se satisfeitos (nível 4/5), os outros 40% mostram-se muito satisfeitos (nível 5/5) e 20% revelam estar ligeiramente satisfeitos (nível 3/5);

- No caso da informação fornecida em caso de falha, 80% revelam estar satisfeitos (nível 4/5) com o tipo de *log* fornecido, nomeadamente o detalhe das métricas que falham;

- No caso da informação fornecida em caso de sucesso, 60% dos inquiridos revelam estar satisfeitos com o *log* fornecido, enquanto que os restantes manifestaram estar satisfeitos;

- Relativamente à facilidade de utilização da ferramenta e a disponibilização dos relatórios originais, 60% dos inquiridos mostram estar muito satisfeitos (nível 5/5) enquanto que os restantes mostram estar satisfeitos (nível 4/5);

- Quanto ao tempo de execução, 20% dos inquiridos mostraram estar pouco satisfeitos (nível 2/5), 40% ligeiramente satisfeitos (nível 3/5) e os restantes 40% satisfeitos (nível 4/5);

Relativamente à utilidade do projeto, 100% dos inquiridos consideram este projeto como extremamente útil para FPS, bem como revelaram que a utilização deste protótipo contribui para terem mais sensibilidade para a temática do desempenho durante o desenvolvimento de *websites*, algo que era um dos objetivos principais do projeto.

Nos campos de resposta aberta, de um modo geral, os inquiridos revelaram que este projeto irá indubitavelmente contribuir para melhorar a experiência dos utilizadores e permitir evoluir os nossos processos de desenvolvimento, principalmente com uma correta integração nas *pipelines* de desenvolvimento.

Não obstante, existem ligeiras melhorias a aplicar em termos de experiência de utilização e o fornecimento de mais mecanismos de obter a informação, como relatórios PDF com os resultados obtidos, mas que são sugestões muito pertinentes que vão contribuir para o desenvolvimento de futuras versões desta ferramenta dentro da organização.

6 Conclusões e trabalho futuro

Neste capítulo procede-se à apresentação das conclusões retiradas do projeto realizado, assim como a descrição dos objetivos atingidos no projeto, as respetivas limitações e trabalho futuro, possibilitando assim melhorias e o crescimento da aplicação protótipo desenvolvida.

Não obstante, é feita uma apreciação final relativamente ao projeto, assim como os conhecimentos apreendidos a nível pessoal.

6.1 Objetivos alcançados

O objetivo primordial associado a este projeto foi a promoção da melhoria do desempenho das aplicações *web* desenvolvidas por FPS.

Perante a necessidade de definir um processo para medir e monitorizar o desempenho das aplicações *web e-commerce*, com especial foco na fase do desenvolvimento dos projetos, a implementação do protótipo permitiu responder à necessidade do desenvolvimento ferramentas especializadas que permitam garantir a fácil manutenção da plataforma de FPS, conforme mencionado na secção 1.2, de forma a poder contribuir para a prevenção dos os problemas adjacentes a um ambiente de desenvolvimento distribuído que se assiste em FPS.

Assim, com o desenvolvimento deste projeto, foi reforçada a importância do desempenho dentro de FPS, bem como rentabilizar a informação e potencialidades de ferramentas já disponíveis. A documentação produzida e pesquisa efetuada no âmbito da escrita do presente documento associada a este projeto contempla a necessidade de disponibilizar mais informação sobre os recursos disponíveis, além de possibilitar uma perspetiva mais clara e detalhada de como monitorizar a performance e que práticas devem ser adotadas de modo a poder construir soluções que visem contemplar um bom desempenho.

De forma sistemática, as concretizações dos objetivos propostos pela realização deste projeto encontram-se descritos na Tabela 13. Sistematização dos objetivos do projeto:

Tabela 13. Sistematização dos objetivos do projeto

Objetivo	**Grau de cumprimento**
Promover a melhoria do desempenho das aplicações web desenvolvidas por FPS	Parcial
Desenvolvimento protótipo que fornece fornecer informação sobre o estado de desempenho de aplicações e conteúdos *web* de FPS	Completo
Reunir indicadores mais adequados para a análise de desempenho de aplicações *web e-commerce*	Completo
Definir processo para a medição e monitorização do desempenho de aplicações *web* e-*commerce* na fase de desenvolvimento	Parcial

Em suma, pode-se afirmar que, de um modo geral, os objetivos definidos para este projeto foram alcançados, onde o desenvolvimento do protótipo e a recolha dos indicadores mais adequados irão permitir a promoção da melhoria do desempenho das aplicações *web* desenvolvidas por FPS, logo que este protótipo seja integrado por completo nos ciclos de desenvolvimento e respetivas ferramentas associadas, conforme indicado pelas sugestões apresentadas nas avaliações realizadas, como se pode verificar na secção 5.3.8.

6.2 Limitações

O presente projeto, aliado ao protótipo desenvolvido, pretende dar resposta aos casos de uso (cf. secção 3.1.2) fundamentais para permitir a sua utilização. No entanto, para uma melhor implantação desta solução no ambiente de CI/CD de FPS, existe a necessidade de novos desenvolvimentos por parte das ferramentas internas de FPS, nomeadamente na aplicação CMS – *Content Management System* – aplicação em que é possível definir qual a versão do *frontend* de um dado *website* de FPS que se pretende que seja apresentada, isto quer para ambiente de desenvolvimento como de produção.

No CMS, sempre que existe uma nova versão, há a necessidade de se especificar manualmente qual a versão a utilizar. No entanto, e de modo a possibilitar uma correta interação na *pipeline* de CI/CD nas várias etapas no ciclo de desenvolvimento, é necessário que seja possível definir via *pipeline* a versão disponível numa das máquinas destinadas a testes durante o desenvolvimento, para que o protótipo possua um URL válido a que possa aceder e executar os respetivos testes de desempenho.

Relativamente ao protótipo, existem outras limitações, sendo elas:

- a alta dependência do *browser Google Chrome*, nomeadamente para efetuar auditorias do *Google Lighthouse*, onde as classificações das várias categorias avaliadas só são possíveis de se obter e processar apenas quando o *browser* definido para testes seja baseado em *Chromium*, o único que possibilita a execução do *Google Chrome*;

- a inexistência de dispositivos móveis possíveis de serem utilizados para executarem testes de desempenho com recurso à instância privada do *WebPageTest*.

6.3 Trabalho futuro

Apesar de o protótipo desenvolvido já disponibilizar as funcionalidades mais importantes para os objetivos traçados, existe um conjunto de funcionalidade a serem implementadas e/ou melhoradas, que surgiram ao longo do projeto e que podem ser consideradas uma mais valia para a unidade de negócio ao longo prazo, sendo elas:

- A disponibilização de uma aplicação *web* em que:

    - Seja possível verificar os resultados dos testes realizados, podendo verificar as métricas sob a forma de gráficos, e observar a variação das métricas comuns às várias ferramentas utilizadas;

    - Os dados recolhidos pelas ferramentas utilizadas sejam persistidos e disponíveis para consulta sob as interfaces descritas no ponto anterior (para permitir esta persistência, foi criado num protótipo um mecanismo de *log* para todos os relatórios gerados, em que é possível depois aceder a esses relatórios e gravar os seus dados na futura camada de persistência);

    - Fornecer indicações de possíveis melhorias e otimizações de desempenho com base nos resultados medidos, de modo a que um próximo teste apresente uma melhoria nos resultados medidos.

- Disponibilizar mais agentes para a instância interna do *WebPageTest*, de modo a poder fornecer um leque mais variado de dispositivos para correr testes de desempenho para os *websites* apenas acessíveis pela Intranet da organização;

- Criar uma interface *web* destinada a definição automática do *performance budget* a utilizar, com integração com as ferramentas disponíveis na organização para proceder à monitorização;

- Integrar o protótipo por completo nas *pipelines* de CI/CD, sendo para tal necessário desenvolvimentos adicionais conforme identificado na secção 6.2, referente às limitações. Com a resolução dessa limitação, é possível o desenvolvimento de *scripts* que permitam tornar uma versão do *frontend* de um *website* automaticamente

disponível num servidor interno de testes, de forma a obter um URL válido para executar testes de desempenho e, no caso de não cumprir o *performance budget*, as *pipelines* que integrem a ferramenta desenvolvida irão falhar, não possibilitando o lançamento de uma versão até que os problemas de desempenho sejam endereçados.

6.4 Apreciação pessoal

No decorrer do projeto, considero que o meu conhecimento associado à temática de desempenho de aplicações *web* aumentou de forma considerável por ter aprendido como deve ser efetuada a análise e monitorização de desempenho, as métricas existentes e como estas evoluíram de modo a cada vez dar informação mais útil e precisa aos utilizadores e desenvolvedores bem como as melhores práticas a aplicar de modo a poder desenvolver soluções *web* que privilegiem um bom desempenho, algo de extrema importância para envergadura dos projetos desenvolvidos em FPS.

Não obstante, este projeto permitiu conhecer um leque variado de ferramentas com o mesmo fim – a análise do desempenho de *websites* – além de ter permitido contribuir com uma pequena correção para uma falha existente no módulo *Node.js* do *Calibre*, detetada durante o desenvolvimento da integração do *Calibre* com o protótipo. Existem diversas ferramentas disponíveis, mas grande parte delas eram pagas e, grande parte delas dispunham de documentação e eram baseadas no *WebPageTest*. No caso específico do *WebPageTest*, existem alguns recursos disponíveis, mas não suficientemente claros ou pagos - no caso do livro sobre como utilizar esta ferramenta (Viscomi et al., 2015), não estando completamente disponível para consulta. Foi necessário a análise do código-fonte da ferramenta, de modo a compreender alguns conceitos inerentes. Além disso, a documentação por parte dos autores do projeto está bastante desatualizada, demonstrada pela criação de algumas questões em diversos fóruns, mas que se compreende tendo em consideração a idade do projeto. Para as ferramentas utilizadas, existiu uma curva de aprendizagem, de forma a poder encontrar uma solução que desse resposta ao tipo de integração destas ferramentas inicialmente definida por este projeto, principalmente por falta de exemplos e casos de utilização real.

Relativamente à temática de desempenho, sinto que a equipa está consciente da importância que o desempenho ocupa, justificando pelo investimento numa solução como o *Calibre* para efetuar monitorização e análise diária ao desempenho do conjunto de *websites* de FPS, e mesmo ao desenvolvimento de *dashboards* no *New Relic* com base na recolha de dados dos utilizadores dos *websites*, em que é possível obter uma clara visão do estado do *website*, quer no lado do cliente como no lado do servidor. Mas esta preocupação ficou sempre mais dos intervenientes na infraestrutura e chefia de FPS, pelo que com este projeto foi possível criar sensibilidade para este tópico para o resto da equipa, mais em específico os desenvolvedores *frontend*, ao facilitar o acesso a estas ferramentas e disponibilizar um protótipo que é facilmente integrado nas suas rotinas diárias de desenvolvimento dos projetos.

Com esta prova de conceito foi possível constatar-se que o desenvolvimento de aplicações *web* com um bom desempenho exige esforço e um compromisso em manter esse estado desde do início do desenvolvimento, assumido sob a forma de um *performance budget*, por exemplo. No entanto, com as ferramentas e o *mindset* corretos tal é possível.

Em suma, considero que este projeto foi um contributo significativo quer para mim como para toda a equipa de FPS.

Referências

Alexandrova, J. (2018), «Continuous Integration with TeamCity - TeamCity 10.x and 2017.x Documentation - Confluence», 6 Abril, disponível em: https://confluence.jetbrains.com/display/TCD10/Continuous+Integration+with+TeamCity (acedido 24 Fevereiro 2019).

Arsenault, C. (2017), «Web Performance Trends: Speeding Up the Web», *KeyCDN*, 6 Junho, disponível em: https://www.keycdn.com/blog/web-performance-trends (acedido 24 Fevereiro 2019).

Atwood, J. (2005), «Progressive Image Rendering», *Coding Horror*, 14 Dezembro, disponível em: https://blog.codinghorror.com/progressive-image-rendering/ (acedido 4 Fevereiro 2019).

Awasthi, R. (2018), «What exactly does Checkmarx do and how is it different from competitors? - Quora», 11 Fevereiro, disponível em: https://www.quora.com/What-exactly-does-Checkmarx-do-and-how-is-it-different-from-competitors (acedido 13 Outubro 2019).

AWWARDS e Google. (2018), «Speed Matters: Designing for mobile performance», *Google Awwwarads*, disponível em: http://www.awwwards.org/brainfood-mobile-performance-vol3.pdf (acedido 13 Fevereiro 2019).

Bachuk, A. (2017), «What is tree shaking?», *Alex Bachuk*, 22 Novembro, disponível em: https://medium.com/@netxm/what-is-tree-shaking-de7c6be5cadd (acedido 20 Fevereiro 2019).

Barinka, A. (2018), «Online Luxe Goods Hub Farfetch Climbs After $855 Million IPO», *Bloomberg*, 21 Setembro, disponível em: https://www.bloomberg.com/news/articles/2018-09-21/online-luxury-goods-hub-farfetch-s-ipo-raises-885-million (acedido 17 Janeiro 2019).

Bevacqua, N. (2015a), «Fixing Performance in the Web Stack», *Pony Foo*, 6 Agosto, disponível em: https://ponyfoo.com/articles/fixing-web-performance (acedido 31 Janeiro 2019).

Bevacqua, N. (2015b), «Let's talk about Web Performance», *Pony Foo*, 5 Agosto, disponível em: https://ponyfoo.com/articles/talk-about-web-performance (acedido 31 Janeiro 2019).

Bidelman, E. (2019), *Run Lighthouse in CI, as a web service, using Docker. Pass/Fail GH pull requests.: ebidel/lighthousebot*, JavaScript, , disponível em: https://github.com/ebidel/lighthousebot (acedido 7 Fevereiro 2019).

Borins, M., Miller, A., Ameen, S. e Trott, R. (2019), «Introduction to Node.js», *S*, disponível em: https://nodejs.dev/ (acedido 8 Outubro 2019).

Brown, S. (2019), «The C4 model for visualising software architecture», *C4 Model*, disponível em: https://c4model.com/ (acedido 10 Outubro 2019).

BS Web Team. (2018), «India improves internet speed but still fares poorly in global rankings», *Business Standard India*, 19 Dezembro, disponível em: https://www.business-standard.com/article/current-affairs/here-is-how-indian-mobile-fixed-broadband-internet-speeds-fared-in-2018-118121900288_1.html (acedido 17 Fevereiro 2019).

Buchanan, J. e Gardiner, L. (2003), «A comparison of two reference point methods in multiple objective mathematical programming», *European Journal of Operational Research*, Vol. 149 No. 1, pp. 17–34.

Calibre. (2019a), «Paint based metrics: First Contentful Paint, First Meaningful Paint, Visually Complete, Speed Index», *Calibre*, disponível em: https://calibreapp.com/docs/metrics/paint-based-metrics (acedido 28 Agosto 2019).

Calibre. (2019b), «Time to Interactive», *Calibre*, disponível em: https://calibreapp.com/docs/metrics/time-to-interactive (acedido 28 Agosto 2019).

Calibre. (2019c), «Calibre's CLI and Node.JS API», *Calibre*, disponível em: https://calibreapp.com (acedido 7 Fevereiro 2019).

Cambridge University Press. (2019), «PERFORMANCE | meaning in the Cambridge English Dictionary», disponível em: https://dictionary.cambridge.org/dictionary/english/performance (acedido 7 Fevereiro 2019).

Chatterjee, I., Küpper, J., Mariager, C., Moore, P. e Reis, S. (2018), «The decade ahead: Trends that will shape the consumer goods industry», *McKinsey & Company*, p. 18.

Clark, J., Faris, R., Morrison-Westphal, R., Noman, H., Tilton, C. e Zittrain, J. (2017), «The Shifting Landscape of Global Internet Censorship», *Internet Monitor*, 29 Julho, disponível em: https://thenetmonitor.org/research/2017-global-internet-censorship (acedido 15 Fevereiro 2019).

Creative Bloq Staff. (2013), «Accessibility expert warns: stop using carousels», *Creative Bloq*, 10 Julho, disponível em: https://www.creativebloq.com/accessibility-expert-warns-stop-using-carousels-7133778 (acedido 4 Fevereiro 2019).

Dalal, M. (2015), «Flipkart moves towards becoming app-only platform», *Https://Www.Livemint.Com*, 20 Março, disponível em: https://www.livemint.com/Industry/J9VeQxowSOlHU8ZMUParUL/Flipkart-moves-towards-becoming-apponly-platform.html (acedido 7 Fevereiro 2019).

Dayan, S. (2018), «How I dropped 250KB of dead CSS weight with PurgeCSS», *freeCodeCamp.org*, 3 Julho, disponível em: https://medium.freecodecamp.org/how-i-dropped-250kb-of-dead-css-weight-with-purgecss-28821049fb (acedido 31 Janeiro 2019).

Duran, M. (2017), «WebPageTest API: Test specs», *GitHub*, 9 Agosto, disponível em: https://github.com/marcelduran/webpagetest-api (acedido 2 Outubro 2019).

Duran, M. (2019), *WebPageTest API wrapper for NodeJS.*, JavaScript, , disponível em: https://github.com/marcelduran/webpagetest-api (acedido 2 Outubro 2019).

Economy, E.C. (2018), «The great firewall of China: Xi Jinping's internet shutdown», *The Guardian*, 29 Junho, disponível em: https://www.theguardian.com/news/2018/jun/29/the-great-firewall-of-china-xi-jinpings-internet-shutdown (acedido 17 Fevereiro 2019).

Enright, A. (2010), «Web accelerator revs up conversions, cart size and sales for AutoAnything.com», *Digital Commerce 360*, 19 Agosto, disponível em: https://www.digitalcommerce360.com/2010/08/19/web-accelerator-revs-conversion-and-sales-autoanything/ (acedido 18 Janeiro 2019).

Escardo, D. (2018), «Time to First Byte: The Critical SEO Metric You Aren't Measuring», *Impact*, 20 Novembro, disponível em: https://www.impactbnd.com/blog/time-to-first-byte-seo (acedido 12 Outubro 2019).

Farfetch UK Limited. (2018a), «What is Farfetch? - About Farfetch», disponível em: https://aboutfarfetch.com/about/farfetch/ (acedido 17 Janeiro 2019).

Farfetch UK Limited. (2018b), «Black & White - About Farfetch», disponível em: https://aboutfarfetch.com/about/black-white/ (acedido 18 Janeiro 2019).

Farfetch UK Limited. (2019), «Farfetch - Pressroom», disponível em: https://farfetch.prezly.com/ (acedido 17 Janeiro 2019).

Fowler, M. (2006), «bliki: CodeSmell», *martinfowler.com*, 9 Fevereiro, disponível em: https://martinfowler.com/bliki/CodeSmell.html (acedido 13 Outubro 2019).

Gasparyan, A. (2016), «Most Important Metrics for Your Website Performance», *Monitis Blog*, 22 Dezembro, disponível em: http://www.monitis.com/blog/most-important-metrics-for-your-website-performance/ (acedido 6 Fevereiro 2019).

Goh, J.M. (2018), «process.env: What it is and why/when/how to use it effectively», *Medium*, 20 Junho, disponível em: https://codeburst.io/process-env-what-it-is-and-why-when-how-to-use-it-effectively-505d0b2831e7 (acedido 9 Outubro 2019).

Google. (2018a), «Minify Resources (HTML, CSS, and JavaScript) | PageSpeed Insights», *Google Developers*, 23 Novembro, disponível em: https://developers.google.com/speed/docs/insights/MinifyResources (acedido 4 Fevereiro 2019).

Google. (2018b), «Lighthouse | Tools for Web Developers», *Google Developers*, 21 Agosto, disponível em: https://developers.google.com/web/tools/lighthouse/ (acedido 28 Janeiro 2019).

Google. (2019a), «A new image format for the Web | WebP», *Google Developers*, 19 Janeiro, disponível em: https://developers.google.com/speed/webp/ (acedido 31 Janeiro 2019).

Google. (2019b), «First CPU Idle | Tools for Web Developers», *Google Developers*, disponível em: https://developers.google.com/web/tools/lighthouse/audits/first-cpu-idle (acedido 3 Outubro 2019).

Google. (2019c), «How To Think About Speed Tools | Web Fundamentals», *Google Developers*, 29 Janeiro, disponível em: https://developers.google.com/web/fundamentals/performance/speed-tools/ (acedido 28 Janeiro 2019).

Google. (2019d), *GoogleChrome/Lighthouse*, JavaScript, GoogleChrome, disponível em: https://github.com/GoogleChrome/lighthouse (acedido 3 Outubro 2019).

Google. (2019e), «Lighthouse Scoring Guide | Tools for Web Developers», disponível em: https://developers.google.com/web/tools/lighthouse/v3/scoring (acedido 12 Outubro 2019).

Gosha, G. e Farley, J. (2017), «GIF, PNG, JPG or SVG. Which One To Use?», *SitePoint*, 2 Maio, disponível em: https://www.sitepoint.com/gif-png-jpg-which-one-to-use/ (acedido 31 Janeiro 2019).

Grigsby, J. (2011), «Native is easy. Web is essential.», 11 Abril, disponível em: https://www.slideshare.net/grigs/native-is-easy-web-is-essential/51 (acedido 4 Fevereiro 2019).

Handley, L. (2019), «Smartphones: 72% of people will use only mobile for internet by 2025», 24 Janeiro, disponível em: https://www.cnbc.com/2019/01/24/smartphones-72percent-of-people-will-use-only-mobile-for-internet-by-2025.html (acedido 17 Fevereiro 2019).

Hedley, M. (2019), «Chinese Market Entry», *B2B International*, disponível em: https://www.b2binternational.com/publications/china-market-entry/ (acedido 16 Fevereiro 2019).

Hlushko, E., Khodakivskyi, B., Villanueva, C., Christensen, M. e Venech, G. (2018), «Lazy Loading», 12 Outubro, disponível em: https://webpack.js.org/guides/lazy-loading/ (acedido 6 Fevereiro 2019).

HTTP Archive. (2019), «State of the Web», disponível em: https://beta.httparchive.org/reports/state-of-the-web#bytesTotal (acedido 18 Janeiro 2019).

Hunt, T. (2016), «I wanna go fast: HTTPS' massive speed advantage», 22 Junho, disponível em: https://www.troyhunt.com/i-wanna-go-fast-https-massive-speed-advantage/ (acedido 31 Janeiro 2019).

Interactive Accessibility, Inc. (2019), «ADA Compliance | Interactive Accessibility», disponível em: http://www.interactiveaccessibility.com/services/ada-compliance (acedido 14 Fevereiro 2019).

John, F. (2018), «Local WebPagetest Using Docker», *Medium*, 7 Fevereiro, disponível em: https://medium.com/@francis.john/local-webpagetest-using-docker-90441d7c2513 (acedido 2 Setembro 2019).

Kadlec, T. (2013), «Setting a performance budget», *TimKadlec.com*, 28 Janeiro, disponível em: https://timkadlec.com/2013/01/setting-a-performance-budget/ (acedido 31 Janeiro 2019).

Kadlec, T. (2014a), «Fast Enough», *TimKadlec.com*, 14 Janeiro, disponível em: https://timkadlec.com/2014/01/fast-enough/ (acedido 5 Fevereiro 2019).

Kadlec, T. (2014b), «Performance Budget Metrics», *TimKadlec.com*, 18 Novembro, disponível em: https://timkadlec.com/2014/11/performance-budget-metrics/ (acedido 5 Fevereiro 2019).

Kadlec, T. (2015), «Choosing performance», *TimKadlec.com*, 14 Maio, disponível em: https://timkadlec.com/2015/05/choosing-performance/ (acedido 31 Janeiro 2019).

Kansara, V.A. (2015), «BoF Exclusive | Farfetch Opens E-Commerce Platform to Brands», *The Business of Fashion*, 9 Setembro, disponível em: https://www.businessoffashion.com/articles/bof-exclusive/farfetch-e-commerce-fashion-brands-omnichannel (acedido 18 Janeiro 2019).

Kearney, M., Osmani, A., Basques, K. e Miller, J. (2019), «Measure Performance with the RAIL Model | Web Fundamentals», *Google Developers*, Janeiro, disponível em: https://developers.google.com/web/fundamentals/performance/rail (acedido 1 Fevereiro 2019).

Kim, J. (2019), «What internet search is like behind China's Great Firewall», *Marketplace®*, 15 Fevereiro, disponível em: http://www.marketplace.org/2019/02/15/tech/look-internet-life-behind-great-firewall-china (acedido 15 Fevereiro 2019).

Koen, P.A., Ajamian, G.M., Boyce, S., Clamen, A., Fisher, E., Fountoulakis, S., Johnson, A., et al. (2002), «Effective Methods, Tools, and Techniques», *The PDMA ToolBook for New Product Development*, p. 32.

Kruchten, P. (1995), «Architectural Blueprints—The "4+1" View Model of Software Architecture», p. 15.

Lenox, J. (2019), «How Improving Website Performance Can Help Save The Planet — Smashing Magazine», 15 Janeiro, disponível em: https://www.smashingmagazine.com/2019/01/save-planet-improving-website-performance/ (acedido 28 Janeiro 2019).

Lever, C. (2016), «Operationalizing Performance Budgets», *Rigor*, 14 Junho, disponível em: https://rigor.com/blog/2016/06/5978 (acedido 5 Fevereiro 2019).

Maslen, T. (2012), «Moving Swiftly: The story of how BBC News fell in love with Responsive Web Design», *Speaker Deck*, 25 Setembro, disponível em: https://speakerdeck.com/tmaslen/moving-swiftly-the-story-of-how-bbc-news-fell-in-love-with-responsive-web-design (acedido 4 Fevereiro 2019).

Mateus, J. (2017), «A importância dos questionários de satisfação | LinkedIn», *LinkedIn*, 23 Fevereiro, disponível em: https://www.linkedin.com/pulse/import%C3%A2ncia-dos-question%C3%A1rios-de-satisfa%C3%A7%C3%A3o-joana-mateus/ (acedido 17 Fevereiro 2019).

McDonald, N. (2018), «Digital in 2018: World's internet users pass the 4 billion mark», *We Are Social USA*, 30 Janeiro, disponível em: https://wearesocial.com/us/blog/2018/01/global-digital-report-2018 (acedido 15 Fevereiro 2019).

Meenan, P. (2018), «WebPageTest: Private Instances», *GitHub*, 4 Abril, disponível em: https://github.com/WPO-Foundation/webpagetest-docs (acedido 2 Outubro 2019).

Meenan, P. (2019), «Quick Start Guide - WebPagetest Documentation», disponível em: https://sites.google.com/a/webpagetest.org/docs/using-webpagetest/quick-start-quide (acedido 7 Fevereiro 2019).

Menascé, D.A. (2002), «Load Testing, Benchmarking and Application Performance Management for the Web», *Computer Measurement Group (CMG)*, p. 11.

Metz, E. e Lencevicius, R. (2003), «Efficient Instrumentation for Performance Profiling», *arXiv:cs/0307058*, disponível em: http://arxiv.org/abs/cs/0307058 (acedido 22 Fevereiro 2019).

Mihajlija, M. (2018), «Your first performance budget», *Web.Dev*, 5 Novembro, disponível em: https://web.dev/your-first-performance-budget (acedido 4 Outubro 2019).

Mishunov, D. (2015), «Why Perceived Performance Matters, Part 1: The Perception Of Time», *Smashing Magazine*, 25 Setembro, disponível em: https://www.smashingmagazine.com/2015/09/why-performance-matters-the-perception-of-time/ (acedido 8 Outubro 2019).

Mozilla Corporation. (2019), «JavaScript», *MDN Web Docs*, 30 Setembro, disponível em: https://developer.mozilla.org/en-US/docs/Web/JavaScript (acedido 8 Outubro 2019).

Mozilla Foundation. (2018), «The Internet uses more electricity than...», 12 Março, disponível em: https://internethealthreport.org/2018/the-internet-uses-more-electricity-than/ (acedido 30 Janeiro 2019).

New Relic, Inc. (2019a), «Introduction to New Relic APM | New Relic Documentation», disponível em: https://docs.newrelic.com/docs/apm/new-relic-apm/getting-started/introduction-new-relic-apm (acedido 24 Fevereiro 2019).

New Relic, Inc. (2019b), «Synthetics | New Relic Documentation», disponível em: https://docs.newrelic.com/docs/synthetics (acedido 24 Fevereiro 2019).

New Relic, Inc. (2019c), «Insights | New Relic Documentation», disponível em: https://docs.newrelic.com/docs/insights (acedido 24 Fevereiro 2019).

O'Dell, J. (2013), «New Relic now lets you make plug-ins for any kind of data you've got», *VentureBeat*, 19 Junho, disponível em: https://venturebeat.com/2013/06/19/new-relic-now-lets-you-make-plugins-for-any-kind-of-data-youve-got/ (acedido 24 Fevereiro 2019).

Oracle. (2019), «What Is a Socket? (The Java™ Tutorials > Custom Networking > All About Sockets)», disponível em: https://docs.oracle.com/javase/tutorial/networking/sockets/definition.html (acedido 1 Outubro 2019).

Orendorff, A. (2019), «What Is the Future of Ecommerce? 10 Insights on the Evolution of an Industry», *Enterprise Ecommerce Blog - Enterprise Business Marketing, News, Tips & More*, 31 Janeiro, disponível em: https://www.shopify.com/enterprise/the-future-of-ecommerce (acedido 16 Fevereiro 2019).

Osmani, A. (2018), «Start Performance Budgeting», *Addy Osmani*, 8 Outubro, disponível em: https://medium.com/@addyosmani/start-performance-budgeting-dabde04cf6a3 (acedido 16 Janeiro 2019).

Oxford University Press. (2019), «metric | Definition of metric in English by Oxford Dictionaries», *Oxford Dictionaries | English*, disponível em: https://en.oxforddictionaries.com/definition/metric (acedido 6 Fevereiro 2019).

Petrungaro, D., D'Ianni, L., Mao, S. e Reagent, C.M. (2019), «Conventional Commits», *Conventional Commits*, disponível em: https://www.conventionalcommits.org/en/v1.0.0-beta.4/ (acedido 24 Fevereiro 2019).

Pickering, H. (2016), «CSS Inheritance, The Cascade And Global Scope: Your New Old Worst Best Friends», *Smashing Magazine*, 21 Novembro, disponível em: https://www.smashingmagazine.com/2016/11/css-inheritance-cascade-global-scope-new-old-worst-best-friends/ (acedido 4 Fevereiro 2019).

RisingStack. (2017), «Node.js Performance Monitoring with Prometheus», *Node.js Collection*, 27 Junho, disponível em: https://medium.com/the-node-js-collection/node-js-performance-monitoring-with-prometheus-c3d50c2d5608 (acedido 1 Fevereiro 2019).

Saaty, T.L. (2008), «Decision making with the analytic hierarchy process», *International Journal of Services Sciences*, Vol. 1 No. 1, p. 83.

Schwarz, B. (2017), «Beyond the Bubble: Real world performance», *Building Calibre*, 25 Maio, disponível em: https://building.calibreapp.com/beyond-the-bubble-real-world-performance-9c991dcd5342 (acedido 18 Janeiro 2019).

Sevilleja, C. (2018), «FMP: First Meaningful Paint», *Scotch*, 2 Julho, disponível em: http://scotch.io/courses/10-web-performance-audit-tips-for-your-next-billion-users-in-2018/fmp-first-meaningful-paint (acedido 3 Outubro 2019).

Singh, M. (2019), «India's smartphone market grew by 10% in 2018, bucking global slowdown», *VentureBeat*, 25 Janeiro, disponível em: https://venturebeat.com/2019/01/25/indias-smartphone-market-grew-by-10-in-2018-bucking-global-slowdown/ (acedido 17 Fevereiro 2019).

Smartbear. (2019), «Fundamentals of Performance Profiling», disponível em: https://smartbear.com/learn/code-profiling/fundamentals-of-performance-profiling/ (acedido 7 Fevereiro 2019).

SonarSource S.A,. (2019), «Documentation | SonarQube Docs», disponível em: https://docs.sonarqube.org/latest/ (acedido 13 Outubro 2019).

Statista. (2019), «Global retail e-commerce market size 2014-2021», *Statista*, disponível em: https://www.statista.com/statistics/379046/worldwide-retail-e-commerce-sales/ (acedido 15 Fevereiro 2019).

Staveley, A. (2011), «Dublin Tech: The "4+1" View Model of Software Architecture», *Dublin Tech*, 7 Maio, disponível em: http://dublintech.blogspot.com/2011/05/41-view-model-of-software-architecture.html (acedido 17 Fevereiro 2019).

The Editors of Encyclopaedia Britannica. (2019), «World Wide Web (WWW) | History, Definition, & Facts», *Encyclopedia Britannica*, disponível em: https://www.britannica.com/topic/World-Wide-Web (acedido 13 Outubro 2019).

ThemeFusion. (2019), «Performance Testing using WebPageTest», *ThemeFusion | Professional Website Tools*, 14 Janeiro, disponível em: https://theme-fusion.com/documentation/avada/technical/performance-testing-using-webpagetest/ (acedido 1 Outubro 2019).

TNN. (2018), «India is fastest growing e-commerce market: Report - Times of India», *The Times of India*, 29 Novembro, disponível em: https://timesofindia.indiatimes.com/business/india-business/india-is-fastest-growing-e-commerce-market-report/articleshow/66857926.cms (acedido 16 Fevereiro 2019).

Tuli, S. (2018), «Learn How to Set Up a CI/CD Pipeline From Scratch - DZone DevOps», *Dzone.Com*, 10 Agosto, disponível em: https://dzone.com/articles/learn-how-to-setup-a-cicd-pipeline-from-scratch (acedido 11 Fevereiro 2019).

Uptrends B.V. (2019), «What is Web Performance Monitoring?», *Uptrends*, disponível em: https://www.uptrends.com/what-is/Web-Performance-Monitoring (acedido 6 Fevereiro 2019).

Viscomi, R., Davies, A. e Duran, M. (2015), *Using WebPageTest: Web Performance Testing for Novices and Power Users*, O'Reilly Media, Inc., disponível em: (acedido 10 Setembro 2019).

Wagner, J. (2018), «Why Performance Matters | Web Fundamentals», *Google Developers*, 18 Dezembro, disponível em: https://developers.google.com/web/fundamentals/performance/why-performance-matters/ (acedido 18 Janeiro 2019).

Walton, P. (2019), «User-centric Performance Metrics | Web Fundamentals», *Google Developers*, 29 Janeiro, disponível em: https://developers.google.com/web/fundamentals/performance/user-centric-performance-metrics (acedido 1 Fevereiro 2019).

«WebPageTest - About». (2019), , disponível em: https://www.webpagetest.org/about (acedido 7 Fevereiro 2019).

«yargs/yargs». (2019), *GitHub*, disponível em: https://github.com/yargs/yargs (acedido 26 Setembro 2019).

Anexo A. Conteúdos organizacionais

Este anexo pretende contextualizar sobre recursos e ferramentas disponíveis para as equipas de desenvolvimento de FPS, que permitiram e contribuíram para o modo como este projecto foi realizado, nomeadamente:

- Os processos onde o protótipo desenvolvido será integrado;
- Ferramentas existentes que serão utilizadas para:
    - Integrar o protótipo de modo a poder validar se as soluções *web* de organização cumprem os requisitos mínimos de desempenho definidos sob a forma de um *performance budget* (cf. secção 2.3);
    - recolher informação relevante para o projeto, nomeadamente métricas de desempenho de dados reais de utilizadores.

A.1 Processo de desenvolvimento de FPS

Em FPS, o desenvolvimento de projetos utiliza o sistema distribuído para controlo de versões *Git*, utilizando uma instância interna da plataforma *GitLab CE* (*GitLab Community Edition*) para o efeito, que disponibiliza vários mecanismos para a colaboração entre desenvolvedores e equipas de desenvolvimento.

O típico processo de desenvolvimento de uma tarefa técnica para um *website* passa pelas seguintes fases no seu repositório *Git*:

1. Análise dos requisitos da tarefa, para verificar a sua complexidade, a existência de dependências de outros projetos ou a necessidade de efetuar novos desenvolvimentos nas dependências identificadas.

2. A codificação/implementação associada à tarefa técnica, normalmente efetuada por um ou mais desenvolvedores, recorrendo a práticas de desenvolvimento ágil. Estes desenvolvimentos decorrem num *branch* de *master*, ou seja, um ramo de desenvolvimento à parte com base no ramo principal do projeto em questão. Todas os ficheiros alterados que contêm os desenvolvimentos associados à tarefa são adicionados ao *branch* sob a forma de um ou mais *commits*.

3. Quando o desenvolvimento inicial da tarefa técnica se encontra terminado, procede-se à abertura de um *merge request*, um pedido para integrar a implementação efetuada no *branch* principal do projeto. Antes de proceder a esta abertura, é necessário garantir que:

    a. O *commit*, ou seja, um conjunto de alterações feitas, respeita um conjunto de normas previamente definidas dentro da organização – *Conventional Commits* (Petrungaro et al., 2019).

    b. Os desenvolvimentos efetuados possuem testes unitários que possam garantir o correto funcionamento da implementação realizada.

    c. O *merge request* contém toda a descrição e detalhe necessário sobre a implementação realizada, nomeadamente se esta contém alterações com grande impacto onde as suas dependências possam necessitar de um processo de migração ao utilizar a funcionalidade implementada – *breaking change*.

    d. Assim que o *merge request* esteja aberto, este executa a *pipeline* de *merge requests* do projeto para a respetiva *branch*, de modo a assegurar que a compilação é executada e os testes passam todos com sucesso, e possuem uma cobertura de testes sobre o código desenvolvido no mínimo de 80%. Todas as consequentes alterações (ou seja, novos *commits*) originam a uma nova execução da *pipeline*.

4. Estando o *merge request*, a tarefa passa a fase de *Code Review*, onde o código desenvolvido na *branch* é revisto por uma ou várias equipas de desenvolvimento de FPS, de modo a assegurar a qualidade do desenvolvimento e poder detetar possíveis falhas. O *Gitlab* disponibiliza uma interface para o efeito onde é possível criar comentários no código a sugerir alternativas ou a identificar partes que sejam necessárias melhorar. Os desenvolvedores fazem as correções de uma forma proactiva nesta fase.

5. Com a fase de *Code Review* concluída e tendo obtido uma aprovação por parte dos revisores do *merge request*, a tarefa passa para a fase de testes funcionais/manuais por parte dos QA, de forma a garantir que os desenvolvimentos efetuados cumprem os critérios de aceitação definidos e que a sua integração com as restantes funcionalidades do *website* não levanta quaisquer problemas, caso contrário, o desenvolvedor tem de efetuar as respetivas correções.

6. Uma vez que a tarefa foi testada e aprovada com sucesso por parte do gestor de produto na equipa, este desenvolvimento pode ser incluído no *branch master*, ou seja, a *branch* estável com os conteúdos mais atuais do *website* em desenvolvimento, um processo denominado de *merge*. Este *merge* só é autorizado por parte do *GitLab* caso a *pipeline* tenha passado, ou seja, se a sua compilação e testes tenham executado e passado com sucesso, sem o levantamento de quaisquer erros.

Após a repetição deste conjunto de passos, representando a concretização de várias tarefas de desenvolvimento associadas a um projeto, surge o processo de implantação, no qual é possível lançar uma dada versão do *website* com um conjunto de funcionalidades e/ou correções executadas. Antes de ser disponibilizada ao público, esta versão é alvo de testes de regressão por parte dos QA, que asseguram o funcionamento de todas as anteriores e novas funcionalidades, assim como a respetiva integração. Todo este processo tem ferramentas de apoio que permitem automatizar e facilitar partes destes processos.

A.2 *TeamCity*: sistema de *Continuous Integration/Continuous Delivery*

Como sistema de apoio ao processo de *Continuous Integration* e *Continuous Delivery*, as equipas de desenvolvimento de FPS utilizam a ferramenta comercial *TeamCity*, uma aplicação que permite executar verificações e executar processos de compilação capazes de assegurar que novas funcionalidades consigam ser integradas com a base do *software* existente e permitir a deteção de problemas em fases mais iniciais do ciclo de desenvolvimento de *software* (Alexandrova, 2018).

A Figura 77 apresenta a configuração do *TeamCity* aplicada a um projeto interno (denominado *whitelabel*), em que é possível verificar as várias *pipelines* configuradas, nas quais se destacam:

- "*Build and Publish Merge Requests*", responsável por compilar e correr os testes de forma automática para os *merge requests* que foram abertos associados aos novos desenvolvimentos do projeto;

- "*Build and Publish Master*", responsável por compilar e correr os testes unitários associados ao projeto no *branch master*, sempre associado ao último *commit*. Esta *pipeline* executa uma vez por dia, ou sempre que é feito um *merge* para *master*;

- "*Build and Publish New Versions*", *pipeline* responsável por disponibilizar uma nova versão do *website whitelabel* num sistema de disponibilização de versões de *software* a nível interno (neste caso, é utilizado o sistema *ProGet*) para posterior instalação ou execução.

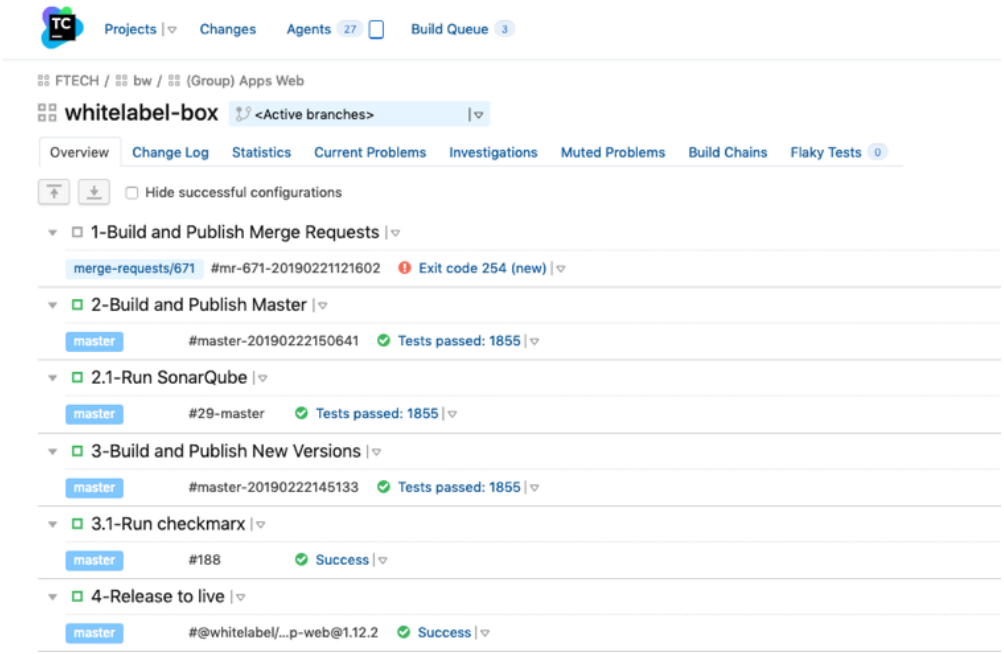


Figura 77. *Pipelines* configuradas no *TeamCity* para um projeto interno de FPS

Cada uma destas *pipelines* possui a sua configuração, que neste caso são um conjunto de passos definidos sobre a forma de *scripts* que representa os vários passos a executar que define cada *pipeline* identificada anteriormente. Estas configurações estão gravadas sob a forma de *templates*, o que significa que estas configurações são partilhadas por todas as *pipelines* associadas aos *websites* e *software* desenvolvidos por FPS, estando assim centralizada a configuração original, facilitando assim a sua manutenção.

Existe também duas *pipelines* dedicadas à execução de *software* que visam promover uma melhor qualidade do código desenvolvido, bem como promover a segurança do mesmo e das bibliotecas externas utilizadas. Elas são:

- O *software SonarQube*, uma ferramenta *open-source* que procede à análise estática do código realizado pelos desenvolvedores. Este tipo de análise, aliado a esta ferramenta permite detetar possíveis *bugs*, vulnerabilidades ou *code smells*[44] (SonarSource S.A, 2019) e com este tipo de configuração, esta verificação é executada a em todos os *branches Git* de um projeto e novas execuções deste tipo de análise podem ser lançadas por via de um novo *Git commit*;

- O *software Checkmarx*, que procede à análise estática do código de um projeto de forma a avaliar se este identifica possíveis vulnerabilidades de segurança (Awasthi, 2018).

[44] Característica no código indicadora de um problema maior no *software* ou sistema (Fowler, 2006).

Uma *pipeline* é executada num agente, ou seja, uma máquina (tipicamente virtual, instalada e executada separadamente da instância do servidor *TeamCity*) que executa todos os passos configurados da *pipeline*, onde é possível no *TeamCity* acompanhar o seu progresso e verificar os *logs* emitidos por esse agente.

É possível integrar as *pipelines* do *TeamCity* com diversas ferramentas, neste caso existe uma integração com o *GitLab*, que esta só permite que um *merge request* só seja integrado com o *branch master* quando esta *pipeline* seja executada com sucesso. Também é possível verificar e obter relatórios sobre os testes unitários executados, bastando para isso configurar a respetiva *pipeline* para esse efeito.

A.3 *New Relic*: sistema de monitorização de desempenho

Em FPS, existem duas ferramentas utilizadas de modo a poder monitorizar o desempenho das soluções *web* criadas, com fins distintos – a obtenção de dados sintéticos e dados reais de utilização (cf. secção 2.5). Para esse efeito são utilizadas as ferramentas *Calibre* (cf. secção 2.8.2) e o *New Relic*.

O *Calibre* é utilizado para poder efetuar testes comparativos de desempenho entre dispositivos emulados por esta plataforma, de forma a simular e obter informação do estado de desempenho dos *websites*, de acordo com as diferentes condições e características de acessos aos mesmos. Também é utilizada a integração do *Slack* com o *Calibre* de modo a poder receber alertas de quando alguma das métricas mais importantes a controlar de um *website* ultrapassa o seu limite.

O *New Relic* é um sistema de *software* destinado a análise e monitorização de aplicações móveis e *web* em tempo real, com suporte à integração de múltiplos *plugins* customizáveis para a recolha de dados arbitrários (O'Dell, 2013). Esta ferramenta integra múltiplas funcionalidades, das quais se destacam:

- *APM* (*Application Performance Monitoring*), destinado à medição e monitorização de métricas de desempenho em tempo real para diversas aplicações em diversos ambientes (New Relic, Inc., 2019a);

- *Synthetics*, sistema onde é possível monitorizar *websites*, transações de negócio críticas e de *endpoints* de uma API, através de serviços de verificação do estado de um *website* e de simulação de atividade real de utilizadores (New Relic, Inc., 2019b);

- *Insights*, onde é possível utilizar e tratar os dados recolhidos nos restantes serviços para analisar comportamentos do utilizador, transações de negócio e informação dos clientes, através da criação de múltiplos gráficos (*dashboards*) (New Relic, Inc., 2019c).

Através da combinação das diversas funcionalidades que o *New Relic* fornece, é possível dentro das equipas de desenvolvimento:

- Monitorizar o desempenho dos ambientes de produção (servidores) onde operam os *websites* dos vários clientes;
- Analisar os diversos pedidos efetuados pelos *browsers* aos servidores;
- Criar alertas para quando algum critério possível de avaliar ultrapassa um dado limite definido;
- Proceder ao rastreamento dos tempos e métricas dos utilizadores dos *websites*, através de código injetado através desta ferramenta nos *websites* criados por FPS;
- Criar ecrãs de monitorização, com várias métricas de desempenho relevantes de acompanhar, como é possível observar na Figura 78.

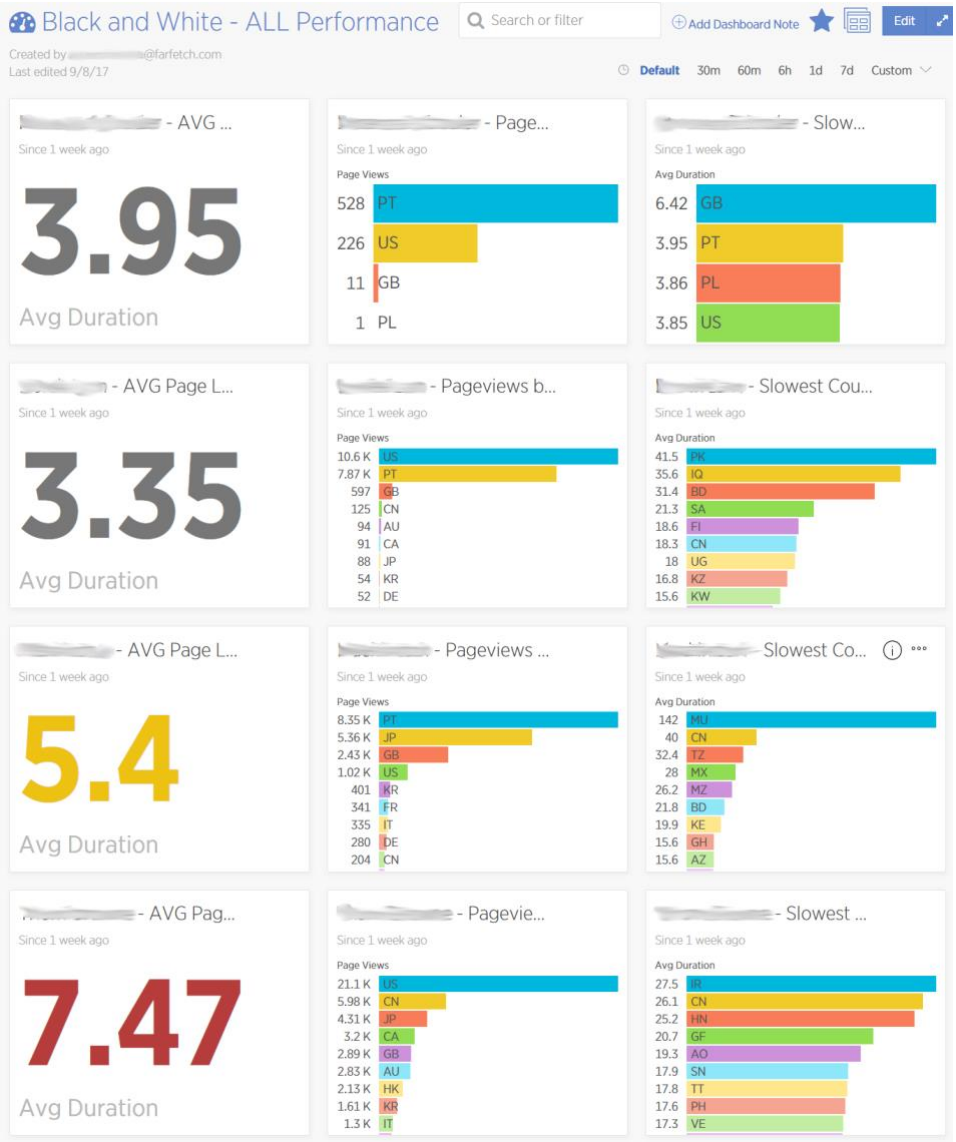


Figura 78. Exemplo de um *dashboard* de monitorização do desempenho de *websites* FPS através do *New Relic Insight*

Anexo B. Análise de Valor

O presente capítulo apresenta a análise de valor realizada no âmbito do projeto, de modo a poder comprovar o seu valor para o mercado, para a organização e para os seus clientes

A análise será feita com recurso ao modelo *New Concept Development* (NCD) de modo a que se possa compreender quais as necessidades do cliente, assim como as ideias geradas e selecionadas para atingir o conceito de negócio final. Recorre-se ainda ao modelo CANVAS de modo a permitir obter uma visão geral do negócio, bem como é utilizado o método de avaliação hierárquica (AHP) para permitir a utilização de critérios qualitativos e quantitativos no processo da avaliação de ideias e desenvolvimento, de modo a auxiliar a compreensão e avaliação do projeto.

A organização e a equipa onde o presente projeto se encontra enquadrado dedica-se ao desenvolvimento de aplicações *web e-commerce* destinados a clientes que pretendam ter soluções para operacionalizar o seu negócio *online*. Dada a respetiva presença *online*, está inerente o acesso às suas aplicações *web* por parte de utilizadores com as mais variadas condições, como a ligação de rede e dispositivos a serem utilizados, assim como a localização geográfica de onde o acesso é proveniente. Como as condições de acesso são muito díspares, é relevante assegurar uma experiência de utilização adequada, pela disponibilização de aplicações com bom desempenho, visto que o problema da falta de desempenho das aplicações de hoje em dia se trata de uma questão de cultura no desenvolvimento da mesma e não tecnológica (cf. secção 1.1.3). Por este motivo, é relevante definir um processo e guias de desenvolvimento que permitem privilegiar e considerar o desempenho como um critério de aceitação para qualquer desenvolvimento realizado dentro da organização e das equipas, havendo assim a necessidade de inovar e gerar ideias, bem como novos processos que permitam monitorizar e assegurar o bom desempenho das aplicações *web* desenvolvidas.

B.1 Modelo *New Concept Development* (NCD)

O *New Concept Development* (NCD) é um modelo iterativo capaz de fornecer uma linguagem e visão comuns, que permite elucidar a visão e cultura de uma organização/projeto. Ao recorrer a este modelo, é possível conhecer quais os fatores de influência do ambiente externo – as capacidades organizacionais, estratégia de negócio e do mundo exterior (como canais de distribuição, clientes e concorrentes).

Este modelo é caracterizado por cinco elementos que definem as atividades a realizar:

- Identificação da oportunidade (*Opportunity Identification*);
- Análise da Oportunidade (*Opportunity Analysis*);
- Geração e Enriquecimento de Ideias (*Idea Generation & Enrichment*);
- Seleção de Ideias (*Idea Selection*);
- Desenvolvimento do conceito (*Concept Definition*).

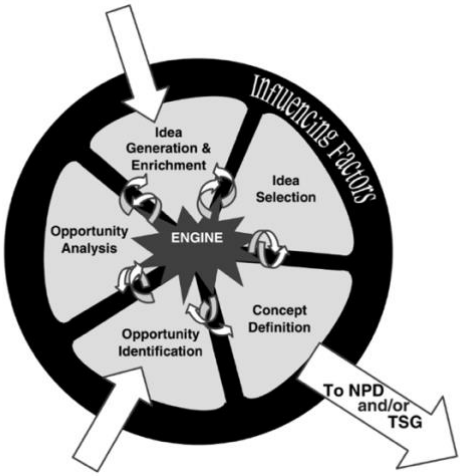


Figura 79. Representação do modelo *New Concept Development* (Koen et al., 2002)

A figura anterior representa graficamente o modelo NCD, em que este adota um formato circular de modo a transmitir a fluidez e iteração de ideias entre os cinco elementos previamente identificados. As setas indicadoras cuja direção está orientada para o interior do modelo representam pontos de partida e sugerem que os projetos e as respetivas soluções devem iniciar pela identificação de oportunidades ou geração e enriquecimento de ideias. A seta com orientação para o exterior do diagrama representa o modo como os conceitos resultantes da aplicação do modelo NCD entram num novo processo: o Novo Processo de Produto (NPD) ou o processo *Technology Stage-Gate™*[45] *(TSG)* (Koen et al., 2002).

[45] Processo estruturado destinado à gestão de projetos tecnológicos de alto-risco.

B.2 Identificação da oportunidade

Atualmente, assiste-se a um crescimento significativo do mercado *online* (Statista, 2019), graças à rápida evolução e desenvolvimento da tecnologia acessível à população mundial, o que permite que mais pessoas disponham de acesso à *Internet* (McDonald, 2018), muito devido à democratização do acesso a dispositivos móveis (*smartphones*, *tablets*) e a planos de dados mais económicos.

De modo a acompanhar o progresso nas operações *online* de um negócio, as organizações procuram incrementar e evoluir as suas funcionalidades para as aplicações *web Business-to-Consumer*[46] (B2C), tanto no *design* como na tecnologia utilizada. No entanto, devido ao aumento de utilizadores com acesso à *Internet*, assim como o aumento da exigência de conteúdos mais rápidos e com melhor qualidade por parte destes, torna-se relevante a aplicação e consideração de práticas que privilegiem o desempenho das aplicações *web* durante a sua implementação.

Existem três tipos de problemas importantes, que surgem pelas aplicações *web* não possuírem um desempenho adequado:

- Os mercados mais emergentes ou com um poder mais reduzido de compra – como por exemplo a Índia ou países do continente Africano – não dispõem de uma infraestrutura de rede moderna e não têm a possibilidade de adquirir dispositivos tecnologicamente mais avançados, pelo que as suas características de armazenamento e processamento são reduzidas face aos requisitos das atuais aplicações *web*, que por sua vez são necessários de modo a permitir uma melhor experiência de utilização (Schwarz, 2017);

- A existência de países e culturas que dispõem de um acesso mais limitado à *Internet*, devido a políticas internas governamentais, das quais podem nascer diversas restrições. Estas podem ser a conteúdos políticos, sociais, de conflitos armados e segurança nacional ou até mesmo a ferramentas disponíveis na *Internet* (plataformas, *websites*) (Clark et al., 2017). Tome-se por exemplo o caso da China – a sua *firewall*[47] privilegia o acesso a conteúdos a *websites* no seu território, enquanto bloqueia acesso a ferramentas exteriores (redes sociais internacionais – *Facebook* e *Twitter*, o acesso ao motor de pesquisa *Google*) assim como reduz a velocidade de acesso a *websites* e conteúdos estrangeiros que não estejam *alojados* em servidores Chineses (Kim, 2019);

- Nos países ocidentais, locais onde existem utilizadores que dispõem de melhores ligações à *Internet*, assim como têm acesso a dispositivos tecnologicamente mais avançados e/ou recentes, estes exigem conteúdos com menores tempos de

[46] Expressão inglesa destinada a categorizar um modo de funcionamento de negócio, em que as transações são realizadas diretamente entre uma organização e os consumidores.

[47] Sistema de segurança de uma rede, que monitoriza e controla tráfego de entrada e saída, permitindo definir um conjunto de regras de segurança que define permissões e/ou bloqueios de tráfego.

carregamento. A sua tolerância para aguardar pela apresentação de conteúdo útil ainda pode ser mais reduzida dependendo do contexto em que está a tentar aceder a esta informação. Segundo um estudo da organização *Google*, existem quatro fatores externos que condicionam o modo como a velocidade da *web* é percecionada – o momento em que uma página pode ser efetivamente usada, a idade do utilizador, o seu estado de espírito e o local de atividade (AWWARDS e Google, 2018). Graças a estes quatro fatores foi possível perceber que:

- os utilizadores mais jovens são os mais exigentes por tempos de carregamento mais rápidos;
- um *website* é percecionado como mais lento quando se acede "em movimento" face quando se acede ao mesmo estando sentado (em casa, por exemplo);
- a velocidade percecionada é interpretada como mais rápida quando um utilizador se sente calmo e/ou relaxado, enquanto que é tida como mais lenta quando este se sente ansioso e/ou está apressado.

Assim, e de modo a endereçar os problemas enumerados anteriormente, é importante disponibilizar e desenvolver soluções *web* com um bom desempenho, através da adoção de boas práticas destinadas a melhorar o desempenho e as experiências de utilização que procuram apresentar a informação crítica/mais útil ao utilizador final, o mais rapidamente possível – por colocar em segundo plano o carregamento de conteúdo secundário, utilizando práticas de *progressive rendering* (cf. secção 2.2).

Posto isto, e para assegurar que as soluções *web* desenvolvidas consigam fornecer uma experiência de utilização adequada nos diversos cenários descritos anteriormente, é necessário existir processos de avaliação, análise e de monitorização do desempenho contra algumas métricas definidas num *performance budget* (cf. secção 2.3), antes de estas sejam implantadas e disponibilizadas aos clientes de FPS e, consequentemente, disponibilizadas aos utilizadores finais.

B.3 Análise da oportunidade

Conforme já foi mencionado anteriormente, o desempenho das aplicações é uma mais-valia para um negócio responsável por operar *e-commerce*, que permite atrair mais visitantes, a reter mais utilizadores e, consequentemente, a gerar mais receitas (cf. secção 2.1). A implementação de um mecanismo que permita a análise e monitorização de desempenho de aplicações *web* no contexto deste projeto traduz-se numa mais-valia para a organização, dado que vai permitir que esta consiga assegurar para todos os seus projetos um padrão mínimo que garanta um desempenho adequado das soluções desenvolvidas antes de estas serem entregues aos seus clientes.

Considerando a necessidade da existência de um processo interno que vise a garantia do desempenho das aplicações desenvolvidas (cf. secção 1.2), com a pesquisa efetuada e a análise das necessidades da organização será possível:

- Tirar partido da utilização de ferramentas de monitorização e análise de desempenho (cf. secção 2.8) no projeto, em que algumas já se encontram em utilização na organização – o *Calibre* (cf. secção 2.8.2);

- Permitir desenvolver um *mindset* e procurar sensibilizar sobre esta temática a todos os intervenientes no desenvolvimento de projetos para clientes;

- Aumentar a potencialidade de rentabilidade das soluções *web*.

As seguintes subsecções explicam alguns pontos-chave que irão beneficiar com a implementação deste projeto.

B.3.1 Novos mercados com restrições

Conforme já foi mencionado em secções anteriores, existem novos mercados que podem ser alvo de um maior investimento e que pode resultar em mais receitas associadas ao *e-commerce*. De acordo com um estudo realizado pela McKinsey afirma que "1,4 mil milhões de pessoas irão fazer parte da classe média em 2020, onde 85% dessas pessoas serão da região Ásia/Pacífico (APAC)" (Chatterjee et al., 2018). Não obstante, "o foco e maior volume de vendas em *e-commerce* tem vindo a afastar-se dos países Ocidentais, uma tendência que pode agravar considerando o maior desenvolvimento dos países Asiáticos" (Orendorff, 2019).

O mercado chinês é um exemplo de aposta para o crescimento de inúmeros "negócios de países ocidentais, muito relacionado por este possuir uma população que ascende a mais 1,3 mil milhões de pessoas, num local em que a sua demografia tem rapidamente vindo a ser alterada, com maiores remunerações auferidas e maior consumo por parte da sua população". Contribuindo para este facto, verifica-se um "ambiente favorável para desenvolvimento e introdução de negócios neste país, o que é especialmente útil para organizações que procuram obter mais lucros, mas que os seus mercados atuais não o permitem" (Hedley, 2019).

A Índia também se revela como um país a considerar investir para expandir um negócio, tendo em conta que é "o mercado que mais cresceu em termos de retalho *online* face ao topo das economias globais, com um crescimento a rondar os 53% desde 2013 a 2017" (TNN, 2018). Este crescimento deve-se à "criação e desenvolvimento de vários *marketplace de e-commerce*, assim como uma maior proliferação do acesso a *smartphones* e a dados móveis" (TNN, 2018).

No entanto, para as organizações que pretendam operar nestes mercados, existem algumas limitações que devem ser consideradas de modo a poder definir a sua estratégia e o modo como estabelecer a sua presença neste tipo de mercados.

No mercado chinês, a combinação de ações legislativas e tecnologias forçadas pelo país permitiram criar a *Great Firewall of China*, uma "barreira interna que visa regular a utilização da *Internet* dentro do seu território". Esta *firewall* tem como principais objetivos "o bloqueio de acesso, assim como reduzir o tráfego a *websites* estrangeiros (através do aumento do tempo de carregamento e de execução destes), mas que privilegia o acesso à sua rede interna, promovendo a utilização de conteúdo nacional e o desenvolvimento das empresas no seu território" (Economy, 2018). Para uma organização estrangeira poder operar convenientemente em território chinês deve obter uma licença *ICP* [48] (*Internet Content Provider*), o que lhes "permite operar um *website* regional, não sofrendo assim das políticas tão restritivas aplicadas à *Internet* no território chinês, mas que requer que os servidores para esse *website* sejam alojados no seu território". No entanto existem organizações que não dispõem de recursos para a disponibilização de um *website* regional, pelo que se torna relevante a aplicação de diversas práticas nas suas aplicações, de modo a que permita um grau de utilização adequado para a população chinesa, como é o caso da:

- agregação de pedidos a um servidor, de modo a reduzir o tempo de comunicação com um servidor externo;

- não integração de mecanismos de recolha de informação sobre a utilização de *websites* que necessitem de comunicar com possíveis serviços bloqueados em território chinês, como é o caso de alguns serviços das empresas *Google* e *Facebook*.

No mercado Indiano, o "acesso a tecnologia mais avançada é algo recente, pelo que os *smartphones* bem como a infraestrutura disponível não é tão avançada quando comparada com as condições existentes nos países ocidentais" (Singh, 2019), pelo que conteúdos desenvolvidos não se encontram preparados para ser utilizados de uma forma adequada por a população Indiana. Para potenciar a utilização de *websites* produzidos por organizações ocidentais neste mercado é importante "assegurar e testar que as soluções realizadas se encontram preparadas para serem executadas num ambiente cujos dispositivos disponíveis são de gama média/baixa e as velocidades de acesso à *Internet* são consideradas baixas, quando comparadas com outros países" (BS Web Team, 2018).

Em suma, existem vários mercados a emergir e que possuem potencial para permitir o crescimento do volume de negócios. Por outro lado, é necessário analisar a sua cultura e restrições que estes apresentam e confrontar com as soluções desenvolvidas por uma organização.

B.3.2 Aumento da exigência dos utilizadores

Os utilizadores são, e estão, cada vez mais exigentes no que diz respeito à utilização de soluções *web*. O rápido desenvolvimento e evolução da tecnologia é um dos grandes fatores que

[48] Licença que permite a operação de *websites* na rede interna da China, emitida pelo ministério da Indústria e da Tecnologia da informação do país.

contribui para o aumento desta exigência, onde o acesso à *Internet* e a dispositivos que permitam o acesso a esta estão mais democratizados, especialmente os dispositivos móveis como *smartphones* e *tablets* (Handley, 2019), local de onde provêm muitos acessos atualmente, segundo um estudo da Google (AWWARDS e Google, 2018).

Dado o aumento de dispositivos móveis e o respetivo acesso à *Internet* por sua via, é natural que agora existam vários locais onde uma pessoa possa navegar na *Internet* – seja em casa, no local do trabalho, em viagem ou outro qualquer lugar, como nas viagens diárias. No entanto, o local de acesso pode influenciar o estado de espírito de um utilizador – se este está mais relaxado (quando está em casa, onde tipicamente dispõe de uma ligação estável à *Internet*), ou se está mais agitado (em viagem ou trabalho, onde recorre a dados móveis) – pelo que afeta a forma como interpreta a velocidade de resposta de um *website*.

No estudo realizado pela Google (AWWARDS e Google, 2018), foi feito um inquérito de modo a compreender a relação entre o desempenho real e o desempenho que os utilizadores experienciam (*perceived performance*), com foco na navegação *web* em dispositivos móveis. Com base nos resultados obtidos, representados nos gráficos do estudo apresentados na Figura 80, grande parte inquiridos afirmaram que o tempo de carregamento (*Load Time)* de uma página fora de casa (*Out & About*), em que exige um acesso via dados móveis com tecnologia 3G ou 4G, lhes demorou mais de 4 segundos, enquanto que em locais como a sua residência (*Work*) ou trabalho (*At Home*), o tempo de carregamento da mesma página *web* variou entre 1 e 4 segundos.

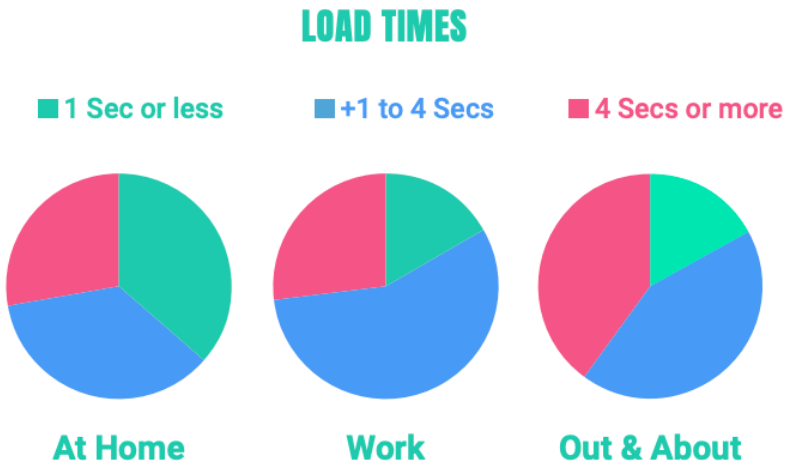


Figura 80. Gráficos representativos da variação dos tempos de carregamento por tipos de localização do estudo realizado pela Google (AWWARDS e Google, 2018)

Esse estudo afirma ainda que 53% das visitas a *websites* oriundas de dispositivos móveis demorem mais de 3 segundos a carregar são abandonadas (AWWARDS e Google, 2018). Estes números comprovam a afirmação inicial – a construção de *websites* com um melhor desempenho deve ser tida uma prioridade.

Em termos de experiência de utilização de um *website*, a velocidade de carregamento de uma página é um dos aspetos mais valorizados, seguido da facilidade para encontrar um conteúdo que o utilizador pretende, o que é justificável, pois este só consegue utilizar um *website* até que este esteja carregado e que apresente conteúdo. Tipicamente um *website* é considerado rápido caso seja carregado em menos de 4 segundos, de acordo com o referido estudo da Google (AWWARDS e Google, 2018). No entanto, o estudo concluiu que 29% das pessoas não consideraram que a experiência de carregamento foi rápida. Esta experiência de acesso a uma página por um utilizador é denominada de *perceived speed* e é afetada pelos seguintes factores externos (Figura 81): uso efetivo do *website* (*effective use*), idade (*age*) estado de espírito (*state of mind*), e local de acesso (*place of activity*).

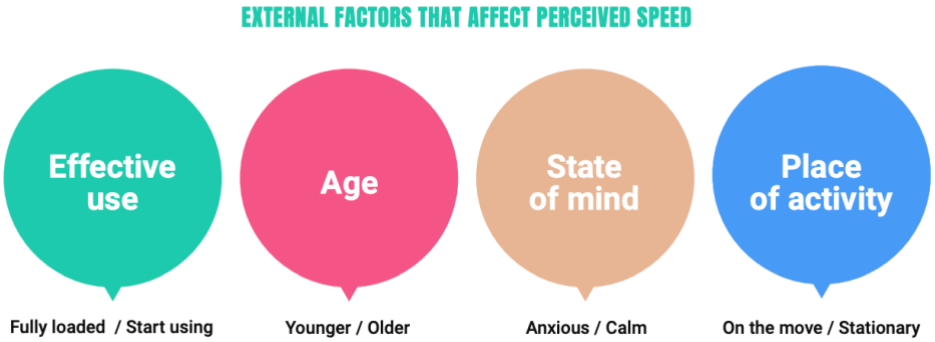


Figura 81. Fatores externos que afetam a *perceived speed* (AWWARDS e Google, 2018)

O fator externo do tempo necessário para a utilização efetiva de um *website* é importante, pois apesar de uma página não estar totalmente carregada, pode permitir ao utilizador interagir e ver o conteúdo útil que requisitou, enquanto outras operações estão a ser efetuadas em segundo plano. Isto é particularmente útil para os *websites e-commerce* que necessitam de apresentar listas com um elevado número de produtos, em que para o utilizador parece que a página já está totalmente carregada, tendo este assim percecionado a velocidade de carregamento como rápida (AWWARDS e Google, 2018).

A idade é um fator importante a considerar quando se define a audiência para um *website*, pois diferentes faixas etárias têm diferentes níveis de tolerância para aguardar pelo carregamento de uma página, onde a audiência mais jovem, situada entre os 18 e os 24 anos são os mais exigentes, enquanto que pessoas com 25 ou mais anos interpretam esse mesmo tempo de carregamento como mais rápido (AWWARDS e Google, 2018).

A perceção positiva da velocidade e resposta de um *website*, além de ter um impacto positivo sobre como um utilizador se sente durante a sua visita, pode incentivar a visitas futuras ao mesmo, bem como moldar a perceção que este tem com a marca/organização por detrás do *website*. Ao conseguir estas perceções positivas, uma organização pode esperar que os seus KPI

de negócio (como a taxa de conversão[49]) aumentem, o que contribui para um aumento das suas receitas.

B.3.3 Vantagens do projeto

O presente projeto traz diversas vantagens, para além das mencionadas nas secções anteriores – entrar em novos mercados com restrições e contribuir para a melhoria da experiência de utilização através do aumento da velocidade percecionada – que revelam ser importantes por contribuir para o aumento do conhecimento das equipas de desenvolvimento e da qualidade das soluções desenvolvidas. Essas vantagens são:

- A redução de recursos utilizados pelos servidores (cf. secção 2.2) e pelo cliente (cf. secção 2.1);
- Aumentar o número de clientes e os segmentos de mercado;
- Melhorar os KPI de negócio (como a contribuição para o aumento da taxa de conversão de utilizadores);
- Aumentar as receitas obtidas (*revenue*);
- Sensibilizar para a temática de desempenho junto das equipas de desenvolvimento;
- Melhorar a experiência do programador (*developer experience*), ao facilitar a sua manutenibilidade;
- Garantir um padrão mínimo de desempenho nos *websites* desenvolvidos pela adoção de um *performance budget*;
- Permitir o crescimento e aumento do volume de negócio das marcas e retalhistas que possuam soluções *e-commerce* desenvolvidas por FPS.

B.3.4 Vantagens de um melhor desempenho

Melhorar o desempenho de um *website e-commerce* é sinónimo de trazer um maior tráfego e visitantes regulares ao mesmo, que pode significar um aumento do volume de vendas e o aumento da taxa de conversão.

O desempenho não adequado de um *website* nos dias de hoje é um problema proveniente da não-existência de uma cultura que privilegie o desempenho (cf. secção 1.1.3), e dado que o estado atual da tecnologia permite a criação de soluções *web* adequadas, é importante garantir

[49] Taxa que identifica a percentagem de possíveis clientes que podem executar uma determinada ação desejada pelo negócio, como é o caso de efetuar uma compra *online*.

que estas estejam disponíveis para os mais diversos tipos de pessoas poderem utilizarem e, assim, contribuir para uma equidade de acesso a nível mundial.

Na perspetiva de uma organização com custos operacionais para suportar a sua base tecnológica, os conteúdos que privilegiem de melhorias de desempenho permitem a redução de gastos com servidores, de forma a que estes possam consumir menos recursos e menos energia. Para um cliente, também permite que este tenha menos custo, por gastar menos dados do seu *plafond* ao visitar um *website* alvo de melhorias de desempenho.

B.3.5 Desvantagens de um melhor desempenho

Como foi possível verificar, a melhoria de desempenho de um *website* oferece diversas vantagens ao ser corretamente aplicada.

No entanto, a implementação de soluções que privilegiem o desempenho desde do início do desenvolvimento (cf. secção 2.3) exige um maior cuidado com os conteúdos a colocar numa página, o que pode ir contra os *designs web* idealizados para determinadas páginas, assim como exige uma maior colaboração entre a equipa de desenvolvimento e de *design web*.

A implementação de *websites* com bom desempenho implica um maior conhecimento por parte das equipas de desenvolvimento (*know how*[50]), de forma a ser possível a aplicação de boas práticas, técnicas ou configurações específicas que visem um melhor desempenho. O processo para o teste de desempenho de um *website* é algo moroso e complexo, por exigir a sua execução em diversos contextos – dispositivos, condições de rede – com várias repetições, de modo a obter valores de métricas que refletem o real estado de desempenho do mesmo.

B.4 Geração e enriquecimento de ideias

Face à necessidade de garantir o desempenho adequado das soluções *web* desenvolvidas, foram geradas algumas ideias que pudessem ser o foco deste projeto, sendo elas:

- **Ideia 1:** mecanismo de *profiling* (cf. secção 2.7) de desempenho para a organização, que permita analisar o código-fonte das soluções *web* existentes em que, ao identificar possíveis pontos de melhoria na sua análise, proceda à criação de tarefas de uma forma automática na ferramenta interna de gestão de tarefas da organização – o *Jira* (cf. secção 1.5), a serem endereçadas no trabalho futuro das equipas de desenvolvimento;

- **Ideia 2:** o desenvolvimento de um sistema de alertas e alarmística para problemas de desempenho com base na definição de um *performance budget* (cf. secção 2.3), em que este sistema procede à monitorização do desempenho das aplicações *web* em

[50] Saber como. Expressão inglesa que se traduz no conjunto de conhecimentos práticos (como fórmulas, informações tecnologias, técnicas, procedimentos) adquiridos.

execução pelos clientes e que, no caso de alguma métrica cumprir o *budget* definido, emite um alerta para um conjunto de meios definidos (via *e-mail* ou uma ferramenta para comunicação interna);

- **Ideia 3:** a criação de uma ferramenta destinada a medir métricas (cf. secção 2.4) de desempenho, que possa caracterizar o estado de desempenho de um *website* alvo de análise e medição por parte desta;

- **Ideia 4:** a criação de um mecanismo de análise e monitorização de desempenho a aplicar a projetos em desenvolvimento dentro da organização, que vise a integração de múltiplas ferramentas de análise de desempenho e que permita dar informação do estado de desempenho de uma solução *web* e que possíveis correções ou melhorias possam ser feitas de modo a poder contribuir para a melhoria do desempenho da mesma.

De modo a tirar proveito das oportunidades identificadas e analisadas anteriores, com estas ideias geradas e identificadas nesta secção é possível endereçar a garantia do desempenho das soluções *web* desenvolvidas na organização. No entanto, é necessário avaliar estas ideias de modo a assegurar que a ideia a adotar transparece uma proposta de valor maior, assim como permite ir mais de encontro às necessidades reais da organização.

B.5 Seleção de ideias

Para este projeto, após a avaliação de todas as ideias geradas de acordo com a análise hierárquica, de modo a verificar a ideia que fornecia mais vantagens a um curto prazo, a ideia selecionada foi a que visa a criação de um protótipo que procede à análise e monitorização de desempenho de aplicações *web* durante o seu desenvolvimento.

Para a realização deste protótipo, devem ser utilizadas ferramentas já existentes na organização que permitem analisar o estado de desempenho de uma dada aplicação, permitindo obter essa informação de uma forma mais clara, além de possibilitar a sua integração na *pipeline* (cf. secção 2.8.1) de desenvolvimento com um *performance budget* aplicado (cf. secção 2.3), de modo a garantir um nível mínimo de desempenho oferecido em cada versão da aplicação alvo de teste.

Um método de decisão multicritério recorre a técnicas numéricas e matemáticas que auxiliam os intervenientes num processo de decisão a selecionarem uma opção num conjunto discreto de alternativas. Este processo é realizado com base no cruzamento de alternativas com os critérios existentes, permitindo a priorização de alternativas numa situação de critérios conflituosos, de modo a procurar satisfazer restrições, com objetivos conflituantes – ou seja, uma solução de compromisso. Assim, estes métodos fornecem o apoio à negociação e/ou decisão em grupo (Buchanan e Gardiner, 2003).

O método de decisão multicritério a utilizar para determinar qual a solução gerada a selecionar será o método de análise hierárquica – *Analytic Hierarchy Process* (AHP). Este método permite utilizar critérios qualitativos como quantitativos associados ao processo de avaliação e visa dividir um problema de decisão em níveis hierárquicos, de modo a facilitar a sua compreensão e avaliação.

Conforme mencionado na secção B.2, a proposta de valor para este projeto é a melhoria do desempenho das aplicações *web* desenvolvidas dentro da organização, através da análise e monitorização das mesmas, assim como da disseminação de conhecimento associado a esta temática.

De acordo com as ideias geradas na secção B.4, procedeu-se à criação da seguinte árvore hierárquica de decisão.

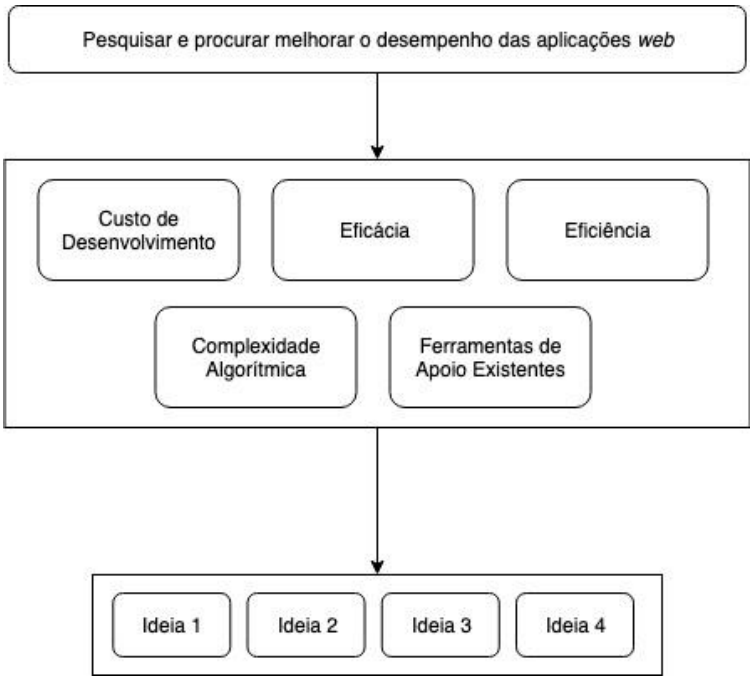


Figura 82. Árvore hierárquica de decisão do projeto

Como se pode constatar na figura anterior, a primeira camada ilustra o objetivo principal do projeto – a pesquisa e a necessidade de procurar melhorar o desempenho das aplicações *web* desenvolvidas pela organização. Os critérios aplicados para avaliar qual a melhor ideia para desenvolver a solução são:

- Eficácia, se a solução cumpre os requisitos de melhoria de desempenho das aplicações *web*;
- Eficiência, como e de que forma é que a solução permite a melhoria das aplicações *web*;
- Complexidade algorítmica, se a solução é fácil ou complexa de ser desenvolvida de modo a atingir o objetivo enunciado;

- Custo de desenvolvimento, a relação tempo/recursos necessários para a implementação de determinada solução.

- Ferramentas de apoio existentes, os recursos existentes de apoio à obtenção de dados de desempenho de uma aplicação *web*.

Com recurso a estes critérios, é viável proceder a avaliação da melhor ideia/solução para atingir o objetivo principal. A seguinte tabela ilustra a matriz de comparação utilizada, de modo a atribuir as prioridades, segundo a Escala Fundamental de Saaty para o método AHP, a fim de tornar a matriz consistente (Saaty, 2008).

Tabela 14. Tabela de avaliação do método de análise AHP

Critérios de avaliação	**Eficácia**	**Ferramentas de apoio existentes**	**Eficiência**	**Complexidade algorítmica**	**Custos de desenvolvimento**
Eficácia	1	2	3	7	6
Ferramentas de apoio existentes	$\frac{1}{2}$	1	2	7	6
Eficiência	$\frac{1}{3}$	$\frac{1}{2}$	1	6	5
Complexidade algorítmica	$\frac{1}{7}$	$\frac{1}{7}$	$\frac{1}{6}$	1	3
Custos de desenvolvimento	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{5}$	$\frac{1}{3}$	1
Total	$2\ \frac{1}{7}$	$3\ \frac{17}{21}$	$6\ \frac{11}{30}$	$21\ \frac{1}{3}$	21

De seguida, procede-se à normalização dos valores obtidos em todos os critérios, de modo a igualar todos os critérios na mesma unidade e ser possível identificar a ordem de importância de cada critério, conforme é ilustrado na seguinte tabela.

Tabela 15. Matriz normalizada do método de avaliação AHP

Critérios de avaliação	Eficácia	Ferramentas de apoio existentes	Eficiência	Complexidade algorítmica	Custos de desenvolvimento
Eficácia	0,4667	0,5250	0,4712	0,3281	0,2857
Ferramentas de apoio existentes	0,2333	0,2625	0,3141	0,3281	0,2857
Eficiência	0,1556	0,1313	0,1571	0,2813	0,2381
Complexidade algorítmica	0,0667	0,0375	0,0262	0,0469	0,1429
Custos de desenvolvimento	0,0778	0,0438	0,0314	0,0156	0,0476
Soma	1,0000	1,0000	1,0000	1,0000	1,0000

Com base nos resultados obtidos na matriz normalizada apresentada na tabela anterior, é possível calcular a prioridade relativa associada a cada critério por via da média aritmética dos valores de cada linha da matriz, permitindo conhecer assim o nível de prioridade associado a cada critério.

Tabela 16. Prioridade relativa dos critérios de avaliação da análise hierárquica

Nº	Critério de Avaliação	Prioridade Relativa	Percentagem
1	Eficácia	0,4153	41,5%
2	Ferramentas de apoio existentes	0,2847	28,5%
3	Eficiência	0,1929	19,3%
4	Complexidade Algorítmica	0,0640	6,4%
5	Custos de desenvolvimento	0,0432	4,3%

Com base nos resultados finais apresentados na tabela anterior, é possível constatar-se que dos cinco critérios influenciadores da seleção da ideia a aplicar, os que mais importância auferiram na seleção foram a eficácia, as ferramentas de apoio existentes e a eficiência, dado impactarem diretamente os resultados e a implementação da solução. Não obstante, a complexidade algorítmica e os custos de desenvolvimento são critérios que apoiaram no processo de tomada de decisão, mas que não revelaram um peso significativo no processo face aos outros critérios.

De todas as ideias geradas, e numa relação de pesquisa/aplicação de conhecimentos, a Ideia 4 – que sugere a criação e implementação de um mecanismo protótipo para análise e monitorização de desempenho a aplicar em projetos em fase de desenvolvimento – é das quatro ideias geradas a que atinge mais vantagens enumeradas (cf. secções B.3.3 e B.3.4) de acordo com a avaliação hierárquica elaborada.

Esta ideia transparece uma maior proposta de valor face aos critérios identificados, como é o caso da utilização de ferramentas já existentes no mercado (cf. secção 2.8), que permite garantir a veracidade e fiabilidade dos resultados obtidos pelo projeto, assim como permite responder mais eficaz e eficientemente às necessidades da organização.

B.6 Definição do conceito

O presente projeto e conceito inerentes devem compreender o modo como o fraco desempenho de uma aplicação afeta os utilizadores de *websites*, e como o desempenho deve ser medido e monitorizado, de modo a poder analisar o resultado destas medições, de modo a garantir um padrão mínimo adequado de desempenho para aplicar a diversos *websites* produzidos.

O projeto será focado em aplicar a monitorização e análise do desempenho de aplicações *web* em fase de desenvolvimento, e integrar numa *pipeline* de desenvolvimento com a definição de um *performance budget*, uma vez que é a fase onde é possível iterar e melhorar mais proactivamente a solução desenvolvida, o que permite trazer mais valor para a organização e aplicar assim uma área ainda não explorada devidamente pelas equipas de desenvolvimento.

É também importante referir que uma parte integrante do conceito deste projeto são os *stakeholders* identificados para o mesmo (cf. secção 1.4). Apenas através da identificação destes intervenientes foi possível definir quais os requisitos necessários de forma a poder produzir e trazer o valor expectável que cada um destes almeja obter do projeto em mãos.

Para tal, existe um conjunto de passos que devem ser realizados de modo a produzir uma solução que forneça informação sobre o estado de desempenho de um *website*:

1. Deve ser especificado *à priori* um *performance budget* com o qual a aplicação *web* alvo de teste deve cumprir;

2. Correr a *pipeline* associada à aplicação *web* (idealmente o conjunto de passos de compilação para disponibilizar uma nova versão para *deployment* num ambiente de produção);

3. Correr testes de desempenho da aplicação *web* alvo de teste com recurso às ferramentas de teste integradas no protótipo;

4. Disponibilizar um relatório que classifique e informe o estado de desempenho da aplicação *web* e que indique se este cumpre ou não o *performance budget* previamente definido;

5. Persistir os dados da análise efetuada para permitir comparar com outros testes efetuados e analisar tendências/padrões entre execuções de teste;

6. O não cumprimento de um *performance budget* durante a execução de testes de desempenho a uma dada versão de uma solução *web* que está a ser *deployed*, deve alertar desse não cumprimento e abortar o processo de *deployment*.

Os requisitos enumerados descrevem os requisitos mínimos deste projeto, onde para além disto, devem ser selecionadas métricas para análise e ser criada a documentação necessária de apoio à implementação e recomendação de boas práticas associadas ao desempenho de aplicações.

B.7 Modelo CANVAS

O *Business Model Canvas*, mais conhecido como modelo CANVAS, trata-se de uma ferramenta de planeamento estratégico que permite auxiliar na melhoria ou criação de um modelo de negócio, de modo a ser possível conhecer todos os intervenientes do processo.

Este modelo, apresentado na seguinte tabela, possui nove elementos que, em conjunto, permitem conhecer melhor o negócio:

- Parcerias Principais (*Key Partners*), em que são incluídas as atividades-chave realizadas por terceiros e os recursos principais adquiridos fora da organização;
- Atividades-chave (*Key Activities*), onde são identificadas as atividades essenciais que possibilitam a entrega da proposta de valor;
- Recursos Principais (*Key Resources*), onde se procede à identificação dos recursos necessários que possibilitam a realização das atividades-chave;
- Proposta de Valor (*Value Proposition*), onde é estabelecido o que a organização vai oferecer ao mercado envolvente e qual o valor que será entregue aos clientes;
- Relacionamento com o Cliente (*Customer Relationships*), em que se estabelece como a organização se relaciona com cada segmento de clientes;
- Canais (*Channels*), onde se estabelece como o cliente compra ou recebe o seu produto/serviço;
- Segmentos de Clientes (*Customer Segments*), onde são identificados os segmentos de clientes que serão o foco da organização;
- Estrutura de Custos (*Cost Structure*), em que são revelados os custos relevantes e necessários de modo a que a estrutura proposta no CANVAS possa funcionar;
- Fluxos de Receita (*Revenue Streams*), onde são estabelecidas as possibilidades para obtenção de receitas pelas proposições de valor.

Tabela 17. Modelo CANVAS do projeto

<table>
<tr><th colspan="5">Análise e monitorização de desempenho de aplicações web e-commerce</th></tr>
<tr><td>Parcerias Principais
(Key Partners)</td><td>Atividades-chave
(Key Activities)</td><td>Proposta de Valor
(Value Proposition)</td><td>Relacionamento com os clientes
(Customer Relationships)</td><td>Segmento de clientes
(Customer Segments)</td></tr>
<tr><td rowspan="4">- Unidade de negócio FPS e respetivas equipas de desenvolvimento

- Projetos internos de FPS (white-label)</td><td>- Conhecer conceitos inerentes ao desempenho e respetivas aplicações

- Conhecer e aplicar métricas de desempenho

- Efetuar o levantamento das necessidades de análise e monitorização

- Implementação com base nos requisitos da organização

- Definir um processo de monitorização de desempenho</td><td rowspan="4">- Disponibilizar informação que permite saber o estado de desempenho de uma versão de uma aplicação implementada e se está de acordo com os parâmetros definidos internamente pela organização</td><td>- Departamento comercial</td><td rowspan="4">- Organização, que pretende oferecer aplicações web com o desempenho otimizado

- Marcas e retalhistas que pretendem possuir websites com desempenho adequado para oferecer aos seus utilizadores

- Utilizadores, que pretendem usufruir de uma melhor experiência de utilização (UX)</td></tr>
<tr><td>Recursos Principais
(Key resources)</td><td>Canais
(Channels)</td></tr>
<tr><td>- Desenvolvedores (para implementação e aprovação de desenvolvimentos)

- Ferramentas de apoio à análise, medição e monitorização</td><td>- Aplicação web</td></tr>
<tr><td colspan="3">Estrutura de Custos
(Cost Structure)</td><td colspan="2">Fontes de Receita
(Revenue Streams)</td></tr>
<tr><td colspan="3">- Desenvolvedores
- Ferramentas de análise, medição e monitorização de desempenho de aplicações web
- Formação em áreas de desempenho aplicadas ao desenvolvimento de aplicações</td><td colspan="2">- Comissões das parcerias com as marcas e retalhistas, provenientes do aumento de vendas de produtos e visitas ao website</td></tr>
</table>

Como é possível observar-se no modelo *CANVAS* apresentado na tabela anterior, ao utilizar ferramentas já existentes (cf. secção 2.8) que permitam a obtenção de informação relativa ao estado de desempenho de uma aplicação *web* e a definição de um *performance budget* (cf. secção 2.3), a entrega da proposta de valor é viável, por permitir a existência e a criação de um protótipo que consiga atuar durante o desenvolvimento das soluções, tornando possível proceder a análise e monitorização de desempenho. De modo a proceder à criação deste protótipo, é necessário obter conhecimento inerente a esta temática, como é o caso das métricas a utilizar (cf. secção 2.4), assim como ir de encontro às necessidades da organização, refletidos como atividades-chave deste projeto.

As fontes de receita expectáveis deste projeto surgem pelo aumento do valor das comissões obtidas por via das aplicações *web* das marcas/retalhistas que, ao usufruírem de soluções desenvolvidas que privilegiem um melhor desempenho destinado a cativar e fornecer uma melhor experiência aos utilizadores finais, conseguem obter uma maior taxa de conversão e, consequentemente, aumentar o volume de vendas efetuadas. Não obstante, a proposta de valor definida permite aumentar a carteira de clientes da unidade de negócio FPS – por atrair um maior número de marcas/retalhistas interessadas em utilizar o serviço de *e-commerce* disponibilizado por esta, assim como chamar a atenção de clientes que possuam um maior volume de vendas (ou seja, grandes marcas e retalhistas).

As principais parcerias estão diretamente relacionadas com a unidade de negócio FPS, dado que inclui os elementos da equipa, os projetos internos em desenvolvimento e ferramentas já implementadas, que servem como fonte de conhecimento bem como local para aplicação do projeto desenvolvido.