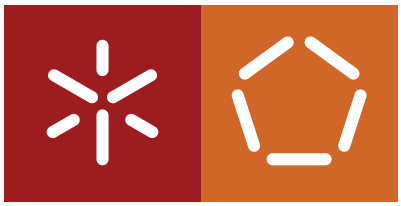
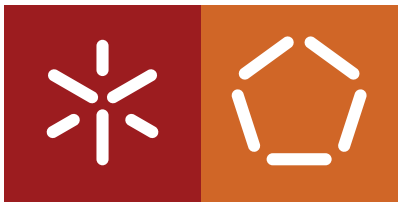


Universidade do Minho
Escola de Engenharia
Departamento de Informática

Catarina Araújo Machado

Modelação de Frameworks de Desenvolvimento Web

February 2021



Universidade do Minho
Escola de Engenharia
Departamento de Informática

Catarina Araújo Machado

Modelação de Frameworks de Desenvolvimento Web

Master dissertation
Integrated Master's in Informatics Engineering

Dissertation supervised by
José Creissac Campos

February 2021

ABSTRACT

Web Development is currently one of the main areas of Software Development (Ignacio Fernández-Villamor et al., 2008), covering both the development of websites, web services and web applications (Mehrab et al., 2018). In the early 90's, most web pages were static HTML documents (Vuksanovic and Sudarevic, 2011). Today, websites are complex web applications that perform transactions, present real-time data and provide interactive experiences to users (Plekhanova, 2009).

There are several possible approaches to developing a web application. On one extreme there is programming "from scratch", where the programmer has to program every detail using some programming language. On the other hand, it is possible to use a "low code" platform, or even a "no code" platform, and develop an application without actually programming (Marvin, 2018). This way, in one case the development is done "manually"(too laborious) and in the other one the code is generated/configured automatically (less flexibility). However, there is a middle ground between these two extremes, which is to use a web framework (Liawatimena et al., 2018).

The first framework was developed in the late 90's and since then over 5000 have been released (Mehrab et al., 2018). This proliferation of languages, frameworks and libraries demonstrates the current state of immaturity of web development technologies, which creates difficulties in the development and maintenance of applications (Ignacio Fernández-Villamor et al., 2008). A model-based approach could help to solve this problem.

In a model-based approach it is possible to increase the level of abstraction of the development process (Meixner et al., 2011). In this way, the model can be executed for simulation or testing purposes at any stage of the development process, which means that the behaviour of the system can be evaluated from the requirements phase to the production phase, without the need to change the description of the system (Bergmann, 2014).

This dissertation thus arises with the general objective of studying how the web area can take advantage of this type of approach, with a particular focus on the interface layer.

KEYWORDS Web Development, Web Frameworks, Modeling, Model-based.

RESUMO

O Desenvolvimento Web é atualmente uma das principais áreas do Desenvolvimento de Software (Ignacio Fernández-Villamor et al., 2008), abrangendo tanto o desenvolvimento de websites, como de web services e de aplicações web (Mehrab et al., 2018). No início dos anos 90, a maioria das páginas web eram documentos HTML estáticos (Vuksanovic and Sudarevic, 2011). Atualmente, os websites são aplicações web complexas que realizam transações, apresentam dados em tempo real e proporcionam experiências interativas aos utilizadores (Plekhanova, 2009).

Existem várias abordagens possíveis ao desenvolvimento de uma aplicação web. Pode optar-se por programar tudo "de raiz", onde o programador tem que programar todos os detalhes recorrendo a alguma linguagem de programação. Por outro lado, pode recorrer a alguma plataforma *low code*, ou até mesmo *no code*, e desenvolver uma aplicação sem efetivamente programar (Marvin, 2018). Desta forma, num dos casos o desenvolvimento é feito de forma "manual" (demasiado trabalhoso) e no seguinte o código é gerado/configurado automaticamente (pouca flexibilidade). No entanto, há um meio termo entre estes dois extremos, nomeadamente utilizar uma framework web (Liawatimena et al., 2018).

A primeira framework foi desenvolvida no final dos anos 90 e desde então mais de 5000 foram lançadas (Mehrab et al., 2018). Esta proliferação de linguagens, frameworks e bibliotecas demonstra o atual estado de imaturidade das tecnologias de desenvolvimento web, que cria dificuldades no desenvolvimento e manutenção das aplicações (Ignacio Fernández-Villamor et al., 2008), que uma abordagem baseada em modelos poderia ajudar a solucionar.

Numa abordagem baseada em modelos é possível aumentar o nível de abstração do processo de desenvolvimento (Meixner et al., 2011). Desta forma, o modelo pode ser executado para fins de simulação ou teste em qualquer etapa do processo de desenvolvimento, o que significa que o comportamento do sistema pode ser avaliado desde a fase de requisitos até à fase de produção, sem a necessidade de alteração da descrição do sistema (Bergmann, 2014).

Esta dissertação surge, assim, com o objetivo geral de estudar como é que a área web poderá tirar partido deste tipo de abordagem, com um foco particular na camada de interface.

PALAVRAS-CHAVE Desenvolvimento Web, Web Frameworks, Modelação, Metodologias de Desenvolvimento baseadas em Modelos.

CONTEÚDO

LISTA DE FIGURAS

LISTA DE TABELAS

Parte I

MATERIAL INTRODUTÓRIO

1

INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO

Desde a criação da *world wide web*, nos anos 90, a internet tem crescido de forma exponencial (Vossen and Hagemann, 2007). Em 2002 atingiu os 500 milhões de utilizadores e, quatro anos depois, chegou a 1 milhar de milhões de utilizadores[1]. Atualmente, são mais de 4 milhares de milhões de pessoas que acedem quase diariamente à internet, cerca de 59% da população mundial[2]. A internet desencadeou assim a Quarta Revolução Industrial, levando o mundo a se estabelecer na Era da Informação.

Este crescimento veio acompanhado de várias mudanças no paradigma das aplicações web. No início, com o surgimento do *browser*, as páginas web eram estáticas, desenvolvidas apenas através de HTML. Consequentemente, o princípio cliente/servidor, abordado na Secção 2.1, foi sendo cada vez mais aproveitado, quer do ponto de vista do cliente com o aumento da qualidade e interatividade das interfaces gráficas, quer do ponto de vista do servidor com o aumento da lógica associada às aplicações (Vossen and Hagemann, 2007).

Em seguida, com o surgimento dos motores de pesquisa, as vantagens da web foram se intensificando passando a ser bastante comercializável, seja através de plataformas de e-commerce, blogs, redes sociais, comunidades, plataformas de *streaming*, serviços no geral e muitas outras opções que estão atualmente à disposição (Vossen and Hagemann, 2007).

A experiência de utilização, isto é, a usabilidade das plataformas, também tem sido um aspeto central de preocupação no desenvolvimento web. O conceito de UX *design*, presente de forma inconsciente já há milhares de anos, tem como principal objetivo proporcionar uma experiência o mais intuitiva e *user-friendly* possível aos utilizadores (Stevens, 2019).

A complexidade e a necessidade de robustez cada vez mais presentes na web levaram à constante necessidade de simplificar o processo de desenvolvimento de aplicações web. As frameworks web nasceram nesse sentido, oferecendo uma alternativa ao extremo de desenvolver aplicações do zero (*"from scratch"*).

No entanto, o número de frameworks web existentes tem aumentado substancialmente nos últimos anos (Mehrab et al., 2018), o que também comprova o atual estado de imaturidade das tecnologias de desenvolvimento web (Ignacio Fernández-Villamor et al., 2008). Atualmente, grande percentagem do tempo de desenvolvimento é utilizado na respetiva manutenção (Meixner et al., 2011). Desta forma, a web, e em particular a camada de interface, pode beneficiar de uma abordagem baseada em modelos.

1 https://phys.org/news/2009-08-key-milestones-internet.html. Retrieved January 27, 2021.
2 https://www.statista.com/statistics/617136/digital-population-worldwide. Retrieved January 27, 2021.

Numa abordagem baseada em modelos é possível aumentar o nível de abstração do processo de desenvolvimento (Meixner et al., 2011). Para além disso, o modelo pode ser executado para fins de simulação ou teste em qualquer etapa do processo de desenvolvimento, o que significa que o comportamento do sistema pode ser avaliado desde a fase de requisitos até à fase de produção, sem a necessidade de alteração da descrição do sistema (Bergmann, 2014).

1.2 MOTIVAÇÃO

O desenvolvimento de interfaces de utilizador para sistemas interativos tornou-se uma tarefa que requer muito tempo e, portanto, dispendiosa com a difusão das interfaces gráficas. Através da análise de diferentes aplicações de software, verificou-se que aproximadamente 48% do código fonte, 45% do tempo de desenvolvimento, 50% do tempo de implementação e 37% do tempo de manutenção são necessários para aspetos relacionados com as UI (Meixner et al., 2011). Segundo Meixner et al. (2011), estes valores parecem continuar válidos desde que esses estudos foram feitos, uma vez que a difusão dos sistemas interativos, bem como as suas necessidades, continuaram a aumentar drasticamente nos últimos anos.

Atualmente, os programadores de UI enfrentam essencialmente os seguintes quatro problemas: heterogeneidade dos utilizadores finais, heterogeneidade das plataformas computacionais e das formas de *input* (teclado, rato, *touch*, entre outros), heterogeneidade das linguagens de programação/linguagens de *markup* e conjuntos de ferramentas *widget* e ainda a heterogeneidade dos ambientes de trabalho (Meixner et al., 2011).

O desenvolvimento de aplicações web, muitas vezes baseado em metodologias de desenvolvimento ágeis (14), representa um desafio para as metodologias baseadas em modelos. No entanto, o atual estado de imaturidade das tecnologias de desenvolvimento web, visível na proliferação de linguagens, frameworks e bibliotecas, cria dificuldades no desenvolvimento e manutenção das aplicações (Ignacio Fernández-Villamor et al., 2008), que a abordagem baseada em modelos poderia ajudar a solucionar.

O desenvolvimento da interface do utilizador baseada em modelos (MBUID) é uma abordagem que tem como objetivo enfrentar os desafios anteriormente mencionados e diminuir o esforço necessário para desenvolver as interfaces gráficas (Meixner et al., 2011). A ideia do desenvolvimento baseado em modelos é que o desenvolvimento seja feito a partir da definição de modelos da solução que se pretende, sendo depois o código gerado automaticamente. No fundo, o intuito é aumentar o nível de abstração do processo de desenvolvimento (Meixner et al., 2011).

As metodologias de desenvolvimento baseadas em modelos são uma mais-valia na medida em que oferecem uma maior consistência na documentação e implementação, eliminam os erros de código, contribuem para uma verificação e validação contínua e permitem um maior número de mudanças em estados avançados dos projetos. No entanto, estas metodologias podem limitar os programas a desenvolver (Bergmann, 2014). Esta limitação é especialmente relevante no desenvolvimento da camada de interface das aplicações.

Este tema surge, assim, com o objetivo geral de estudar como é que a área web poderá tirar partido deste tipo de abordagem, com um foco particular na camada de interface.

1.3 OBJETIVOS

O primeiro objetivo passa por compreender e ter uma visão o mais alargada possível do estado atual da área de desenvolvimento web. Para isso, é importante ter bem assente alguns conceitos fundamentais, como o de aplicação web e o de framework web, e assimilar bem todo o caminho e evolução desta área de desenvolvimento. Consequentemente, é também essencial compreender os padrões arquiteturais que as frameworks seguem.

Em seguida, o objetivo é entrar em detalhe na caracterização de uma framework web, recolhendo as suas principais características e os fatores imprescindíveis que cada uma deve contemplar. Posteriormente, o objetivo passa por fazer um estudo das frameworks web mais recomendadas, segundo várias métricas, para que no fim possa ser feita uma análise mais objetiva dessas frameworks, com base nos critérios anteriormente encontrados.

Desta forma, depois de ficar mais explícito o que caracteriza uma framework, é possível avançar-se para o próximo passo, nomeadamente representá-la através de um modelo, para assim se culminar num Modelo de Framework, tão completo quanto possível, de modo a suportar a posterior modelação de qualquer framework web. Em seguida, deve ser definida uma solução para a geração de código da interface de utilizador, a partir de protótipos, com base no Modelo.

Posteriormente, o objetivo é que seja possível analisar um mockup e conseguir abstraí-lo para o Modelo de Framework web previamente definido. Desta forma, no final conseguimos converter o *mockup* no próprio código da framework.

A ideia de utilizar um modelo visa tornar a abordagem mais genérica, de modo a suportar a evolução de novas frameworks web que possam, entretanto, surgir (Walker and Orooji, 2011).

Com esta dissertação pretende-se dar um contributo no desenvolvimento de soluções de engenharia baseadas em modelos para o desenvolvimento web, através da construção de um modelo da framework, um estudo de caso, um artigo científico e a dissertação.

1.4 ESTRUTURA DO DOCUMENTO

O presente Relatório de Pré-Dissertação está dividido em três diferentes capítulos.

No Capítulo 1 é feita uma introdução ao tema da dissertação, começando por uma breve contextualização, seguida de uma motivação e, por fim, os objetivos gerais.

O Capítulo 2 debruça-se sobre o estado da arte do desenvolvimento web em geral, focando aspetos como o surgimento das frameworks web e as vantagens e desvantagens da sua utilização. Neste capítulo, encontra-se também um estudo, tanto das principais características das frameworks web, como das principais frameworks web da atualidade, culminando num quadro comparativo destas frameworks. Para além disso, é feita também uma introdução ao tema de Desenvolvimento de Interfaces de Utilizador baseados em modelos, utilizando como referência a *Cameleon Reference Framework*.

Por fim, o Capítulo 3 é dedicado à conclusão e trabalho futuro, através da apresentação do plano de trabalho para a realização da dissertação. Neste capítulo, é apresentada uma descrição para cada uma das tarefas a realizar e a respetiva calendarização prevista através de um Diagrama de Gantt.

2

ESTADO DE ARTE

Neste capítulo, começa-se por fazer uma revisão da literatura em relação ao contexto geral da área de desenvolvimento web (Secção 2.1), onde se começa por apresentar as definições de aplicação web (Secção 2.1.1) e de framework web (Secção 2.1.2). Em seguida, são expostas vantagens e desvantagens da utilização de frameworks web (Secção 2.1.3). Posteriormente, são apresentadas as definições e diferenças entre frameworks *client-side* e *server-side* (Secção 2.1.4) e é feita uma revisão de algumas arquiteturas de software, nomeadamente de padrões arquiteturais utilizados pelas frameworks web (Secção 2.1.5). Esta secção termina realçando a importância da escolha da framework web para o desenvolvimento de uma aplicação web (Secção 2.1.6).

Na Secção 2.2 são apresentadas as características ou propriedades mais importantes que as frameworks web devem ter, segundo diferentes autores. Nesta secção, é apresentada ainda uma análise das características mais mencionadas pelos diversos autores e é feita uma exposição mais detalhada dessas características (Secção 2.2.1).

Posteriormente, na Secção 2.3, é apresentado um levantamento das frameworks mais relevantes da atualidade, segundo alguns rankings que têm por base diferentes critérios de seleção (Secção 2.3.1).

Em seguida, na Secção 2.4 é feito um quadro comparativo das frameworks melhor classificadas da Secção 2.3, bem como uma apresentação mais detalhada dessas frameworks, segundo as métricas mais relevantes encontradas na Secção 2.2.

Por fim, na Secção 2.5 é feita uma introdução ao tema de Desenvolvimento de Interfaces de Utilizador baseados em modelos, utilizando como ponto de partida a Cameleon Reference Framework.

2.1 CONTEXTO GERAL DA ÁREA DE DESENVOLVIMENTO WEB

O Desenvolvimento Web é atualmente uma das principais áreas do Desenvolvimento de Software (Ignacio Fernández-Villamor et al., 2008), abrangendo tanto o desenvolvimento de websites, como de *web services* e de aplicações web (Mehrab et al., 2018). No início dos anos 90, a maioria das páginas web eram documentos HTML estáticos (Vuksanovic and Sudarevic, 2011). Atualmente, os websites são aplicações web complexas que realizam transações, apresentam dados em tempo real e proporcionam experiências interativas aos utilizadores (Plekhanova, 2009).

Durante os primórdios da evolução da web, os websites eram programados manualmente, o que conduzia muitas vezes a muitos erros de código e a muito trabalho humano (Curie et al., 2019). Para além disso,

verificava-se que uma grande quantidade de código era duplicada entre projetos, o que levou ao surgimento das frameworks web (Walker and Orooji, 2011). Estas frameworks vieram colmatar estes problemas e trazer inúmeras vantagens, tal como apresentado na Secção 2.1.3.

A primeira framework foi desenvolvida no final dos anos 90 e desde então mais de 5000 foram lançadas (Mehrab et al., 2018). Uma das possíveis justificações para a existência de tantas frameworks reside no facto de quando um programador não está satisfeito com nenhuma framework já existente, desenvolve a sua própria (Walker and Orooji, 2011).

2.1.1 *Definição de Aplicações Web*

Uma aplicação web requer um servidor web para gerir os pedidos do cliente, um servidor da aplicação web para executar as tarefas pedidas e, por vezes, uma base de dados para armazenar a informação[1].

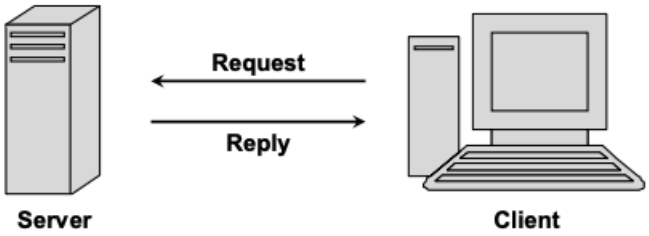


Figura 1: Princípio Cliente/Servidor. Retirado de Stevens (2019).

Essas aplicações podem ser acedidas através de um navegador web, através de uma rede, e são desenvolvidas utilizando linguagens suportadas por navegadores (por exemplo, HTML e JavaScript) (Al-Fedaghi, 2011). Se a aplicação for muito complexa e dinâmica, pode exigir mais processamento do lado do servidor[1]. Para desenvolver aplicações web, os programadores utilizam tanto tecnologias de *frontend* (*client-side*), como de *backend* (*server-side*) (Pop and Altar Samuel, 2014).

As tecnologias de *frontend* têm como principal objetivo apresentar os dados aos utilizadores e permitir a interação deste com o serviço. Por outro lado, as tecnologias de *backend* referem-se a tecnologias de armazenamento e de processamento de dados (Pop and Altar Samuel, 2014).

Existem várias abordagens possíveis ao desenvolvimento de uma aplicação web, desde programar tudo "de raiz", onde o programador tem que programar todos os detalhes recorrendo a alguma linguagem de programação (irrealista na maior parte dos casos, dado o esforço). Por outro lado, pode recorrer a alguma plataforma *low code*, ou até mesmo *no code*, e desenvolver uma aplicação sem efetivamente programar Marvin (2018) (pouca flexibilidade, o que acaba por tirar um pouco de liberdade). Assim, num dos casos o desenvolvimento é feito de forma demasiado "manual" e no seguinte o código é gerado/configurado automaticamente. No entanto, há um meio termo entre estes dois extremos, nomeadamente utilizar uma framework web Liawatimena et al. (2018).

1 https://blog.stackpath.com/web-application. Retrieved January 26, 2021.

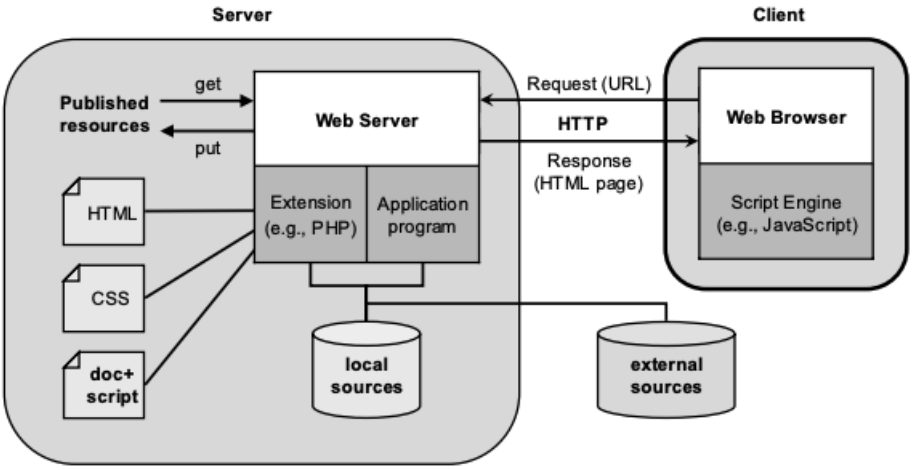


Figura 2: *Client-side* vs. *Server-side*. Retirado de Stevens (2019).

2.1.2 *Definição de Framework Web*

Em ambiente de sistemas de informação, uma framework é uma estrutura de apoio na qual outras aplicações de software podem ser organizadas e desenvolvidas. Uma framework pode incluir programas de apoio, bibliotecas de código, uma linguagem de *scripting*, serviços comuns, interfaces ou outros pacotes de software/*utilities* para ajudar a desenvolver e juntar os diferentes componentes de uma aplicação de software (Shan and Hua, 2006). Uma framework web é basicamente uma ferramenta que ajuda a construir um website, seja ele estático ou dinâmico (Curie et al., 2019).

As *web application frameworks* (WAF) são definidas como um conjunto de classes que compõem um design reutilizável para uma aplicação ou, mais frequentemente, para uma camada de uma aplicação (Ignacio Fernández-Villamor et al., 2008). Uma *web application framework* é normalmente desenvolvida numa arquitetura de n-camadas (Shan and Hua, 2006), tipicamente a camada de interface, a camada de dados e a camada da lógica de negócio (mais em detalhe na Secção 2.1.5).

2.1.3 *Vantagens e desvantagens das frameworks*

As frameworks web ajudam os programadores ao trazerem uma forma padrão de desenvolver e implementar aplicações web (Mehrab et al., 2018). Essa abordagem reduz drasticamente a rotatividade e o risco de tecnologia, uma vez que as frameworks *open source* são ativamente mantidas e aprimoradas por profissionais altamente qualificados em todo o mundo (Shan and Hua, 2006). Para além disso, adotar uma framework como a infraestrutura de desenvolvimento padrão para aplicações web é a melhor maneira de garantir que o desenvolvimento não fica preso a nenhuma arquitetura proprietária (Shan and Hua, 2006).

Os efeitos positivos das frameworks nos projetos estão também relacionados com a diminuição de erros de código (Curie et al., 2019), redução do tempo de desenvolvimento, complexidade reduzida, aumento da produ-

tividade, fiabilidade (Mehrab et al., 2018), extensibilidade (Okanovic, 2014)(Mehrab et al., 2018), modularidade (Okanovic, 2014), qualidade (Shan and Hua, 2006)(Salas Zarate et al., 2015) e performance (Salas Zarate et al., 2015). Outras vantagens são o facto de proporcionarem um ambiente completo para o desenvolvimento, interoperabilidade, consistência, código exaustivamente testado nas bibliotecas, classes e funções, bem como código bem estruturado, utilizando padrões arquiteturais. Para além destas, outras vantagens são a segurança e manutenção, para que os programadores não tenham de construir sistemas personalizados a partir do zero cada vez que lançam um novo website e ainda o facto de possuírem componentes de software ou blocos de construção de código, de modo a que os programadores possam partilhar e reutilizar o código (Vuksanovic and Sudarevic, 2011).

Estes benefícios proporcionam o que todos os programadores desejam, nomeadamente o rápido desenvolvimento de aplicações, reutilização tanto do código como da concepção, custos de manutenção reduzidos e uma maior facilidade de personalização (Okanovic, 2014).

No entanto, existem também algumas desvantagens no uso de frameworks web. Em algumas situações, o desempenho da aplicação pode ser bastante prejudicado devido à complexidade e sobrecarga do código da framework, que acaba por criar uma maior carga para o hardware subjacente (Vuksanovic and Sudarevic, 2011). Para além disso, existem algumas frameworks que possuem uma curva de aprendizagem elevada e convenções rígidas que dificultam a flexibilidade da aplicação e a criatividade do programador (Vuksanovic and Sudarevic, 2011).

Essa curva de aprendizagem é o reflexo da framework definir uma série de regras que é necessário cumprir e o facto de encapsular uma série de conceitos que é necessário compreender.

2.1.4 *Client-side e Server-side*

As frameworks podem ser classificadas em duas diferentes categorias: *client-side* e *server-side*. As frameworks *client-side* são responsáveis pela implementação e melhoria das interfaces de utilizador que vêm sob a forma de características animadas e *layouts*. Alguns exemplos dessas frameworks são as seguintes: Vue.js[2], Angular.js[3] e Ember.js[4] (Curie et al., 2019).

Devido ao facto de existirem tantos dispositivos com acesso à web, é importante que quando se está a desenvolver uma interface gráfica se tenha em consideração a sua responsividade. Esta é uma característica que permite que a aplicação web se adapte automaticamente a qualquer tipo de dispositivo, desde um telemóvel com um pequeno ecrã até um computador com um grande monitor (Voutilainen et al., 2015). Essas frameworks são uma mais-valia nesse sentido, uma vez que incorporam bibliotecas e *guidelines* que facilitam esse trabalho.

2 https://vuejs.org
3 https://angularjs.org
4 https://emberjs.com

As frameworks *server-side* têm também regras e arquiteturas bem definidas e permitem criar diferentes tipos de páginas. Essas frameworks podem ainda fornecer fatores de segurança às páginas web. Exemplos deste tipo de frameworks incluem Django[5], Zend[6] e Ruby on Rails[7] (Curie et al., 2019).

Desta forma, as *client-side* permitem uma experiência de utilização mais próxima de uma aplicação nativa, enquanto que as *server-side* estão mais presas ao modelo de navegação na web.

2.1.5 *Arquiteturas de software*

Como forma de facilitar o desenvolvimento de aplicações web, as frameworks podem seguir diferentes padrões arquiteturais (*architectural patterns*). Estas arquiteturas são soluções gerais, reutilizáveis e bem estabelecidas para problemas comuns na conceção de software e ajudam a documentar as decisões de conceção arquiteturais, bem como facilitam a comunicação entre os *stakeholders* através de um vocabulário comum (Avgeriou and Zdun, 2005).

A *three-tier arquitecture* é uma abordagem que defende a organização da aplicação em três camadas, nomeadamente a camada de apresentação, a camada de negócio e a camada de acesso a dados. Essas camadas são uma forma de encapsular a lógica que cada parte executa (Kouraklis, 2016).

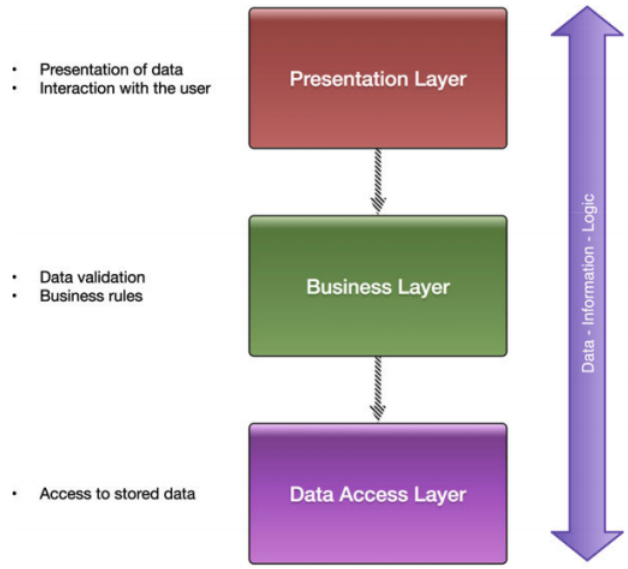


Figura 3: Estrutura da arquitetura de três camadas. Retirado de Kouraklis (2016).

O *design pattern Model-View-Controller* (MVC) combina várias tecnologias normalmente divididas num conjunto de camadas (Pop and Altar Samuel, 2014) e é normalmente utilizado para enriquecer a camada de interface (Kouraklis, 2016). Desta forma, este padrão ajuda a organizar melhor o código do programa (Nassourou,

5 https://www.djangoproject.com
6 https://www.zend.com
7 https://rubyonrails.org

2010), promovendo a separação da lógica de domínio (*controller*), interface de utilizador (*view*) e processamento de dados (*model*). Este padrão é frequentemente implementado através de uma estrutura de pastas (Vuksanovic and Sudarevic, 2011). O *Model* suporta basicamente o *backend* e contém todas as camadas da lógica de dados (Curie et al., 2019), sendo os ficheiros que o compõe responsáveis pela busca, modificação, inserção e remoção de dados da base de dados. A *View* é responsável pelo aspecto visual da página (*frontend*) (Curie et al., 2019), mostrando os dados aos utilizadores da aplicação (Vuksanovic and Sudarevic, 2011). Por fim, o *Controller* é responsável pela invocação de páginas apropriadas de acordo com o pedido do utilizador (Nassourou, 2010). Desta forma, o *Controller* aciona e recolhe dados do *Model*, carrega os dados e encaminha-os para as *Views*, que apresentam os resultados ao utilizador (Vuksanovic and Sudarevic, 2011).

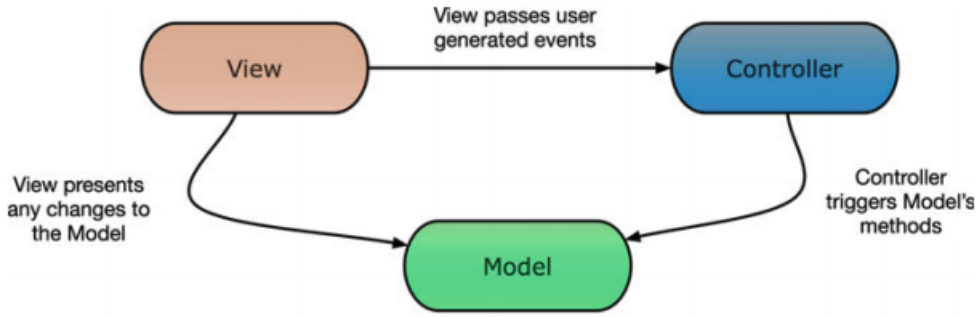


Figura 4: Estrutura da arquitetura MVC. Retirado de Kouraklis (2016).

Através da figura seguinte, é possível perceber a relação entre a arquitetura de três camadas e a arquitetura MVC (Kouraklis, 2016).

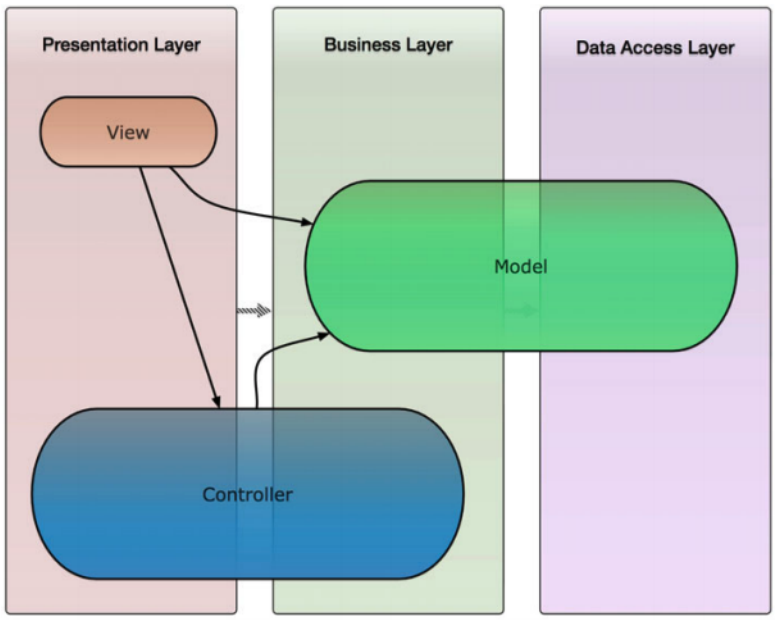


Figura 5: Relação entre arquitetura de três camadas e MVC. Retirado de Kouraklis (2016).

A arquitetura MVP (*Model-View-Presenter*) surgiu numa tentativa de tentar combater algumas insuficiências da arquitetura MVC. Neste tipo de abordagem, o *Controller* foi substituído pelo *Presenter* e as funções, responsabilidades e capacidades de cada parte foram alteradas. Desta forma, passa a existir uma evidente separação entre a *View* e o *Model* e a sincronização é realizada pelo *Presenter* (Kouraklis, 2016).

Nesta versão do padrão MVP não há comunicação entre a *View* e o *Model*, o que normalmente é intitulado como *Passive View* (Kouraklis, 2016).

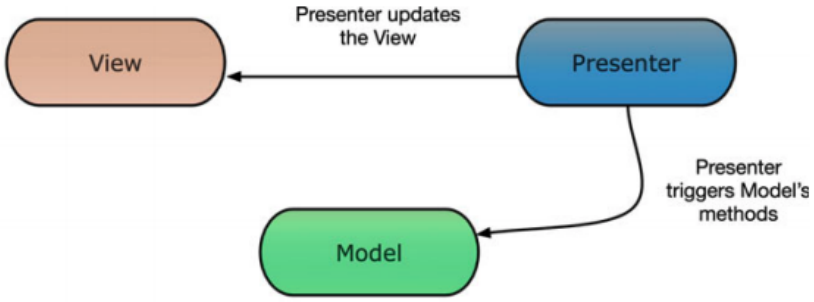


Figura 6: Estrutura da arquitetura MVP. Retirado de Kouraklis (2016).

A relação entre a arquitetura de três camadas e o padrão MVP, segundo Kouraklis (2016), pode ser apresentada através da imagem seguinte.

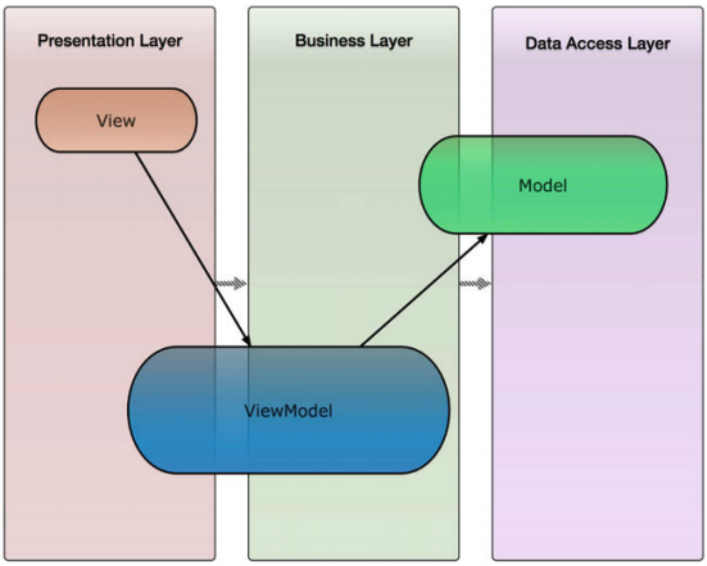


Figura 7: Relação entre arquitetura de três camadas e MVP. Retirado de Kouraklis (2016).

Consequentemente, a arquitetura MVVM (*Model-View-ViewModel*) surgiu com uma alternativa aos padrões MVC e MVP. Nesta arquitetura, o *ViewModel* substitui o *Presenter* e o *Controller* (Kouraklis, 2016).

O *Model* mantém-se com uma lógica semelhante à do padrão MVP, sendo responsável pelo acesso a diferentes fontes de dados (por exemplo, bases de dados, ficheiros ou servidores). A *View* representa os dados no

formato apropriado (gráfico ou não gráfico), refletindo o estado dos dados e recolhe a interação e eventos do utilizador. Na arquitetura MVVM, a maior parte do código encontra-se no *ViewModel*. Este componente representa a forma como se espera que a *View* seja, de acordo com as interações do utilizador. É no *ViewModel* que são descritos vários princípios e estruturas que apresentam dados específicos recolhidos através do *Model* (Kouraklis, 2016).

O *ViewModel* trata também da comunicação entre a *View* e o *Model*, encaminhando todos os dados necessários da *View* para o *Model* de uma forma que o *Model* possa facilmente interpretar (Kouraklis, 2016).

Figura 8: Estrutura da arquitetura MVVM. Retirado de Kouraklis (2016).

Segundo Kouraklis (2016), este padrão pode ser relacionado com a arquitetura de três camadas da seguinte forma.

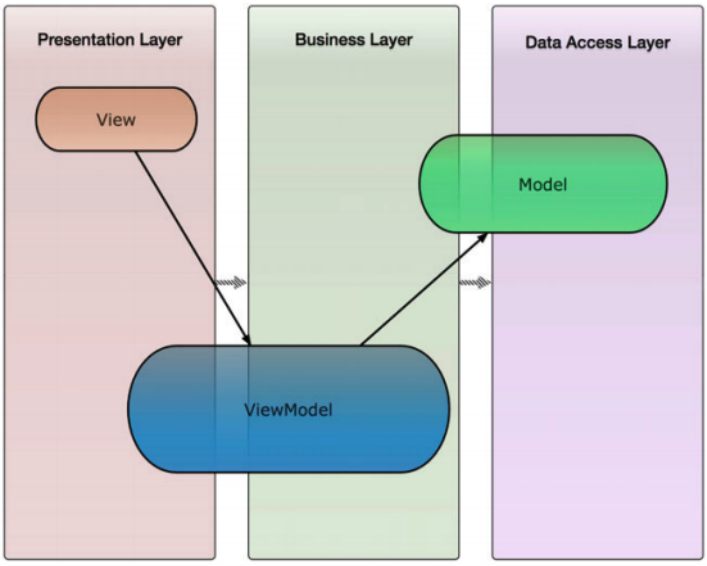


Figura 9: Relação entre arquitetura de três camadas e MVVM. Retirado de Kouraklis (2016).

2.1.6 *A escolha da framework*

A escolha da framework mais adequada para um determinado projeto é uma etapa-chave no desenvolvimento de aplicações web (Ignacio Fernández-Villamor et al., 2008), podendo até ser considerada a fase mais importante. Como atualmente existem muitas frameworks baseadas em diferentes linguagens, a escolha da framework a utilizar pode não ser uma tarefa fácil (Curie et al., 2019)(Salas Zarate et al., 2015) e não existe uma framework

que seja a melhor para qualquer projeto (Plekhanova, 2009). Escolher uma framework inadequada pode prejudicar todo o desenvolvimento da aplicação web, quer seja pela falta de documentação existente, por causarem dificuldades aos programadores ao fazerem quaisquer alterações ao comportamento do núcleo da framework, quer por afetarem o desempenho e a velocidade dos websites (Curie et al., 2019). Para além disso, uma má escolha pode levar a vários problemas e atrasos no desenvolvimento de uma determinada aplicação web, uma vez que há uma perda de tempo ao estudar os detalhes de uma nova linguagem que não é a melhor para o projeto, há uma falha no cumprimento dos prazos, uma vez que os programadores não estão habituados à framework e há também o desperdício de tempo a tomar medidas correctivas para escolher uma framework diferente. Para evitar todos estes problemas, é muito importante conhecer e identificar as melhores práticas de desenvolvimento web (Salas Zarate et al., 2015) e perceber previamente quais são as características que pretendemos que o nosso projeto reúna e escolher uma framework que vá ao encontro desses aspetos (Ignacio Fernández-Villamor et al., 2008).

2.2 CARACTERÍSTICAS DAS FRAMEWORKS DE DESENVOLVIMENTO WEB

Existem vários artigos que tentam enumerar e caracterizar as características mais importantes nas frameworks e que tentam definir modelos de comparação das frameworks web. Em Ignacio Fernández-Villamor et al. (2008) é mencionado que as principais propriedades para categorizarmos uma framework são as seguintes:

- Descrição de Domínio e Persistência;
- Apresentação;
- Segurança;
- Usabilidade;
- Testes;
- Orientação do Serviço;
- Orientação dos Componentes;
- Adopção.

Para cada um desses aspetos, é também apresentado um conjunto de perguntas que devem ser respondidas do ponto de vista de cada uma das frameworks que pretendemos analisar. Respondendo a todas essas questões, é possível retirar importantes conclusões em relação à interoperabilidade, maturidade, usabilidade e à capacidade de realizar testes de cada uma das frameworks (Ignacio Fernández-Villamor et al., 2008).

Em Curie et al. (2019) são apresentadas sete diferentes características que as frameworks podem ter ou suportar, sendo elas:

- Computação em Nuvem;
- HTML 5;
- *Template Framework*;
- Mapeamento Relacionado a Objetos;
- Segurança;
- Suporte de Plataforma;
- *Debugging*.

Já em Walker and Orooji (2011) são focados os seguintes aspetos:

- Instalação;
- Curva de Aprendizagem;
- *Core Library*;
- ORM;
- *Unit Testing*;
- JavaScript incluído;
- Documentação;
- Comunidade;
- Atualizações;
- Reutilização de partes.

No entanto, os autores acabam por concluir que conhecer bem a linguagem é a parte mais importante ao usar-se uma framework.

Os aspetos realçados em Vosloo and Kourie (2008) são os seguintes:

- Apresentação;
- Validação de Formulários;
- Validação;

- Fluxo das Páginas;
- Estado da Sessão;
- Autenticação;
- Integração do *Backend*;
- Tratamento de Eventos;
- Concorrência;
- Uso de Recursos;
- *Miscellaneous*.

No artigo Salas Zarate et al. (2015) são apresentadas as melhores práticas no desenvolvimento web, nomeadamente:

- Suporte AJAX;
- Computação em Nuvem;
- Suporte *Comet*;
- Mensagens de Erro Personalizadas;
- Personalização e Extensibilidade;
- *Debugging*;
- Documentação;
- Validação de Formulários;
- Suporte HTML5;
- Internacionalização;
- Suporte de frameworks baseadas em JavaScript;
- ORM;
- Renderização paralela;
- Suporte de plataforma;
- Suporte REST;
- *Scaffolding*;

- Segurança;
- *SiteMap*;
- *Template framework*;
- Testes;
- Uso de Atores;
- Uso de *Lazy Loading*;
- Usar *pattern matching* e *Wiring*.

No entanto, o autor realça três características que considera fundamentais para o sucesso de uma aplicação web, sendo elas a Fiabilidade, a Usabilidade e a Segurança.

Já no artigo Plekhanova (2009) é mencionado que as métricas tradicionais de avaliação de uma framework, tais como *benchmark* performance e a qualidade técnica da linguagem de programação, são cada vez mais menosprezadas, uma vez que atualmente as diferenças entre estes aspetos são cada vez menores e cada vez menos relevantes para o desenvolvimento web. Desta forma, neste artigo são mencionadas sete métricas e cada uma delas deve ser avaliada numa escala de 1.00 (fraco) a 5.00 (excelente). Posteriormente, é apresentada uma fórmula que atribui diferentes pesos a cada uma destas métricas, culminando assim numa fórmula para a avaliação de frameworks. As métricas apresentadas são as seguintes:

- Desenvolvimento da interface do utilizador (peso de 0.2);
- *Maintainability* (peso de 0.15);
- Gestão e migração de dados (peso de 0.2);
- Testabilidade (peso de 0.15);
- Popularidade (peso de 0.1);
- Comunidade e Maturidade (peso de 0.1);
- *Marketability* (peso de 0.1).

2.2.1 *Análise das Características das Frameworks de Desenvolvimento Web*

Para se avaliar e comparar as diferentes frameworks, pode seguir-se diferentes critérios, mas existem algumas características que acabam por ser mais comuns em diversos autores que fazem esta análise, como se pode constatar a partir de Ignacio Fernández-Villamor et al. (2008), Curie et al. (2019), Walker and Orooji (2011), Vosloo and Kourie (2008), Salas Zarate et al. (2015) e Plekhanova (2009), apresentado anteriormente (Secção 2.2).

Desta forma, tendo em consideração o número de vezes que cada parâmetro aparece nesses artigos, pode concluir-se que as características mais relevantes de uma framework são as seguintes:

1. Apresentação (2 vezes);
2. Segurança (3 vezes);
3. Computação em Nuvem (2 vezes);
4. Suporte HTML 5 (2 vezes);
5. *Template Framework* (2 vezes);
6. Testes (4 vezes);
7. Suporte de Plataforma (2 vezes);
8. *Debugging* (2 vezes);
9. ORM (2 vezes);
10. Suporte de frameworks baseadas em JavaScript (2 vezes);
11. Documentação (2 vezes);
12. Comunidade (2 vezes);
13. Validação (3 vezes);
14. Usabilidade/Curva de Aprendizagem (2 vezes).

Apresentação

A apresentação implica tudo o que uma aplicação web precisa para entregar a interface gráfica no browser do utilizador, utilizando para isso *hypertext markup language* (HTML) ou uma linguagem de *markup* semelhante (Vosloo and Kourie, 2008).

A apresentação é provavelmente o requisito funcional mais simples e mais bem compreendido que uma framework web deve proporcionar. É o primeiro requisito reconhecido, uma vez que as primeiras frameworks web concentraram-se exclusivamente na apresentação (é de salientar que a apresentação está intimamente relacionada com as preocupações do padrão de design MVC) (Vosloo and Kourie, 2008).

Segurança

As aplicações na web estão sob constante ataque e é necessário o uso de mecanismos de proteção contra os problemas mais comuns, tais como o *cross site scripting* (XSS), as falhas de injeção ou a execução de ficheiros maliciosos. No entanto, não há uma forma clara de resolver estes problemas e normalmente as frameworks recomendam boas práticas ou fornecem ferramentas úteis para a sua gestão, como mecanismos para lidar com a autenticação, autorização e sessões de utilizadores (Ignacio Fernández-Villamor et al., 2008).

Computação em Nuvem

A computação em nuvem é a prestação de serviços tais como o armazenamento, servidores, análises, entre outros, através da Internet. Existem vários benefícios associados ao uso de computação em *cloud*, como segurança, desempenho, produtividade, flexibilidade e custo. Uma vez que existem vários tipos de *cloud deployments*, como nuvens públicas, privadas e híbridas, o utilizador pode optar pela opção que melhor satisfaz as necessidades da sua aplicação (Curie et al., 2019).

Suporte HTML 5

HTML significa *Hyper Text Markup Language* e define as propriedades e os comportamentos da página web. Com HTML é possível incorporar imagens, animações, áudios, entre muitos outros elementos, sem utilizar programas de terceiros (Curie et al., 2019).

Template Framework

Os templates podem ser utilizados para melhorar a uniformidade, a disposição e a navegação de qualquer website. Os templates são basicamente páginas web pré-desenhadas, constituídos por conjuntos de peças de código HTML, que cortam o processo de desenvolvimento da aplicação web. Os templates de websites podem ser uma forma rápida e fácil de diminuir a carga de trabalho (Curie et al., 2019).

Testes

Os programadores frequentemente dedicam mais tempo e esforço a encontrar e corrigir erros de código do que efetivamente a escrever novo código. Os testes de software são uma atividade destinada a avaliar a qualidade de um programa e a tentar melhorá-lo através da identificação de defeitos e problemas (Salas Zarate et al., 2015).

Suporte de Plataforma

É importante que antes dos programadores se dedicarem ao desenvolvimento de novo código, avaliem as exigências que serão colocadas ao hardware e ao sistema operativo. Em suma, seria pouco produtivo investir muito tempo e dinheiro na configuração e codificação para descobrir que o desempenho do servidor é fraco devido ao facto da plataforma não ser adequada. As frameworks web foram desenvolvidas para funcionar numa variedade de plataformas, seja Windows, Linux ou Os X (Salas Zarate et al., 2015).

Debugging

O *debugging* é o processo de identificar, localizar e eliminar erros de código (Curie et al., 2019).

ORM

Object–relational mapping (ORM) permite que os programadores definam declarativamente o mapeamento entre o modelo do objetivo e a base de dados e permite também que as operações de acesso à base de dados sejam expressas em termos de objetos (Salas Zarate et al., 2015). A principal função da ORM é reduzir a complexidade do código (Curie et al., 2019).

Suporte de frameworks baseadas em JavaScript

A incorporação de JavaScript numa página web permite melhorar a experiência dos utilizadores através da conversão de uma página estática numa página interativa. O código JavaScript corre do lado do cliente, ou seja, corre no processador do utilizador e não no servidor web. Desta forma, diminui-se a largura de banda e a tensão do lado do servidor web (Salas Zarate et al., 2015).

Documentação

A documentação de uma framework pode cobrir todos os detalhes da framework. Uma vez que as frameworks se baseiam em alguma linguagem de programação, a documentação da framework também se baseia na documentação da própria linguagem de programação (Walker and Orooji, 2011).

A documentação pode incluir a API, isto é, a lista de todos os métodos de cada biblioteca da framework, tutoriais com exemplos de tarefas mais comuns e ainda tutoriais de terceiros (Walker and Orooji, 2011).

Comunidade

A comunidade de apoiantes de uma framework é constituída por pessoas que interagem e comunicam sobre a framework em websites e blogs. A comunidade incentiva o crescimento da framework e simultaneamente ajuda a esclarecer dúvidas de outros programadores (Walker and Orooji, 2011).

Validação

A validação está relacionada com o input do utilizador e frequentemente é vista como parte do tratamento de formulários. Assim, a validação refere-se à criteriorização da entrada de dados por parte do utilizador em relação a algumas restrições. A validação pode também depender de conhecimentos de domínio mais complexos que podem ter de ser verificados num sistema de *backend* (Vosloo and Kourie, 2008).

O programador deve especificar as validações que devem ser efetuadas, como é que os erros de validação devem ser transmitidos ao utilizador (para que este os possa corrigir) e que influência devem ter esses erros no comportamento dinâmico da interface gráfica apresentada (Vosloo and Kourie, 2008).

Usabilidade/Curva de Aprendizagem

A usabilidade mede a facilidade de interação de um utilizador com um sistema. Para tal, são consideradas todas as funcionalidades do sistema, bem como a sua interface gráfica. A usabilidade depende de aspetos como o tipo

de utilizadores que vão trabalhar com o sistema, quais são os seus objetivos e qual é o contexto de utilização. No âmbito das frameworks web, a usabilidade mede a experiência de desenvolvimento dos programadores ao utilizarem a framework (Ignacio Fernández-Villamor et al., 2008). Essa experiência acaba por estar intimamente relacionada com a curva de aprendizagem da framework.

2.3 FRAMEWORKS DE DESENVOLVIMENTO WEB

Tal como mencionado na Secção 2.1, desde o início da era digital foram já desenvolvidas mais de 5000 frameworks de desenvolvimento web (Mehrab et al., 2018). Todas essas frameworks possuem diferentes características e são reconhecidas por diferentes particularidades.

Existem vários estudos que tentam fazer *rankings* das frameworks web, tendo por base diferentes fatores como o tamanho da comunidade da framework, a popularidade ou até mesmo a quantidade de utilizações bem-sucedidas. Em seguida, serão apresentados alguns desses *rankings*.

2.3.1 *Rankings de Frameworks Web*

Em [8] é apresentado um ranking de frameworks tendo por base duas diferentes métricas de popularidade, nomeadamente o GitHub *score* e o Stack Overflow *score*. O GitHub *score* é baseado no número de estrelas que o repositório tem no GitHub. Estas estrelas representam o número de utilizadores que colocaram esse determinado repositório como favorito, sendo que os repositórios favoritos de um utilizador aparecem no seu próprio perfil público. O Stack Overflow *score* é baseado no número de perguntas realizadas no Stack Overflow que contêm a tag da framework.

Uma vez que estas duas medidas de popularidade estão em escalas diferentes, as pontuações finais são normalizadas a uma escala de 0-100 e é feita uma média entre os dois resultados obtidos, culminando assim numa pontuação final. Caso a framework não possua um repositório no GitHub ou não tenha uma tag inequívoca no Stack Overflow, a métrica em falta é desconsiderada, ficando assim a pontuação final igual ao valor da métrica existente.

Segundo [8], neste ranking são consideradas todas as frameworks capazes de terminar a frase "Acabei de construir esta aplicação web em [inserir nome da framework aqui]" (as frameworks podem ser "*full-stack*", "*server-side*" ou "*client-side*").

Como se pode ver na Figura 10 e 11, atualmente React é a framework melhor posicionada tendo em conta estas duas métricas. Em segundo lugar está ASP.NET MVC, que apesar de não ter repositório no GitHub, consegue o segundo lugar por ser bastante popular no Stack Overflow. Em seguida, apresenta-se Angular, Ruby on Rails, AngularJS, Vue.js e Django que surgem empatadas. É de realçar que a framework com melhor GitHub *Score* é Vue.js e a framework com melhor Stack Overflow *Score* é ASP.NET.

8 https://hotframeworks.com. Retrieved January 25, 2021.

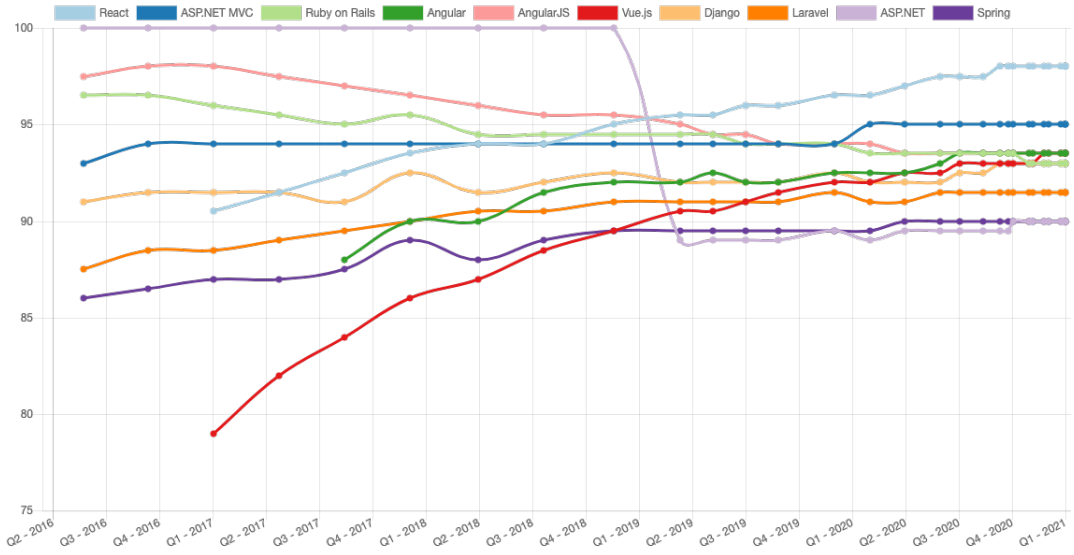


Figura 10: Ranking das frameworks (formato gráfico). Retirado de [8].

Rankings

Framework	Github Score	Stack Overflow Score	Overall Score
React	99	97	98
ASP.NET MVC		95	95
Angular	91	96	93
Ruby on Rails	87	99	93
AngularJS	90	97	93
Vue.js	100	87	93
Django	89	97	93
Laravel	90	93	91
ASP.NET	80	100	90
Spring	86	94	90
Express	88	87	87
Flask	89	83	86
Meteor	86	80	83
Symfony	81	86	83
CodeIgniter	79	86	82
JSF		81	81
Ember.js	80	78	79
.NET Core	77	80	78

Figura 11: Ranking das frameworks (formato tabela). Retirado de [8].

Já em Salas Zarate et al. (2015) são selecionadas as seguintes frameworks: Struts, JSF, CakePHP, Grails, Ruby on Rails, Catalyst e Django. Os autores referem que esta seleção foi feita tendo em consideração o nível de maturidade e a utilização bem-sucedida destas frameworks em vários projetos (incluindo websites muito conhecidos). Para além disso, mencionam também que essas conclusões foram deduzidas com base num ranking que tem em conta a experiência de programadores que estão habituados a trabalhar com estas frameworks e

que estes afirmam tê-las selecionado por terem uma grande e ativa comunidade de programadores e porque possuem uma curva de aprendizagem muito baixa. No entanto, considero que o critério de escolha não é muito claro na globalidade e não fica explícito como é que as frameworks foram efetivamente escolhidas.

Desde 2011, o Stack Overflow tem vindo a fazer anualmente um estudo onde questiona os programadores em relação a quais são as suas tecnologias preferidas, os seus hábitos ao programar e as suas preferências de trabalho (Overflow, 2017). No último estudo, em 2020, foram inquiridos cerca de 65.000 programadores e uma das questões consistia em selecionar as frameworks web que utilizam no seu dia a dia. Desta forma, segundo os resultados desse estudo, que podem ser consultados em Overflow (2020) e na Figura 12, as dez frameworks web mais utilizadas em 2020 foram as seguintes:

1. jQuery [9] - https://jquery.com;
2. React.js - https://reactjs.org;
3. Angular - https://angular.io;
4. ASP.NET - https://dotnet.microsoft.com/apps/aspnet;
5. Express - https://expressjs.com;
6. ASP.NET Core - https://docs.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-5.0;
7. Vue.js - https://vuejs.org;
8. Spring - https://spring.io;
9. Angular.js - https://angularjs.org;
10. Django - https://www.djangoproject.com;
11. Flask - https://flask.palletsprojects.com/en/1.1.x;
12. Laravel - https://laravel.com;
13. Ruby on Rails - https://rubyonrails.org;
14. Symfony - https://symfony.com;
15. Gatsby - https://www.gatsbyjs.com;
16. Drupal - https://www.drupal.org.

Segundo Overflow (2020), jQuery ainda está a liderar a tabela, mas tem vindo a perder utilizadores para React.js e Angular ano após ano.

9 jQuery é considerada uma biblioteca de JavaScript e não tanto uma framework web, mas foi incluída por fazer parte do estudo.

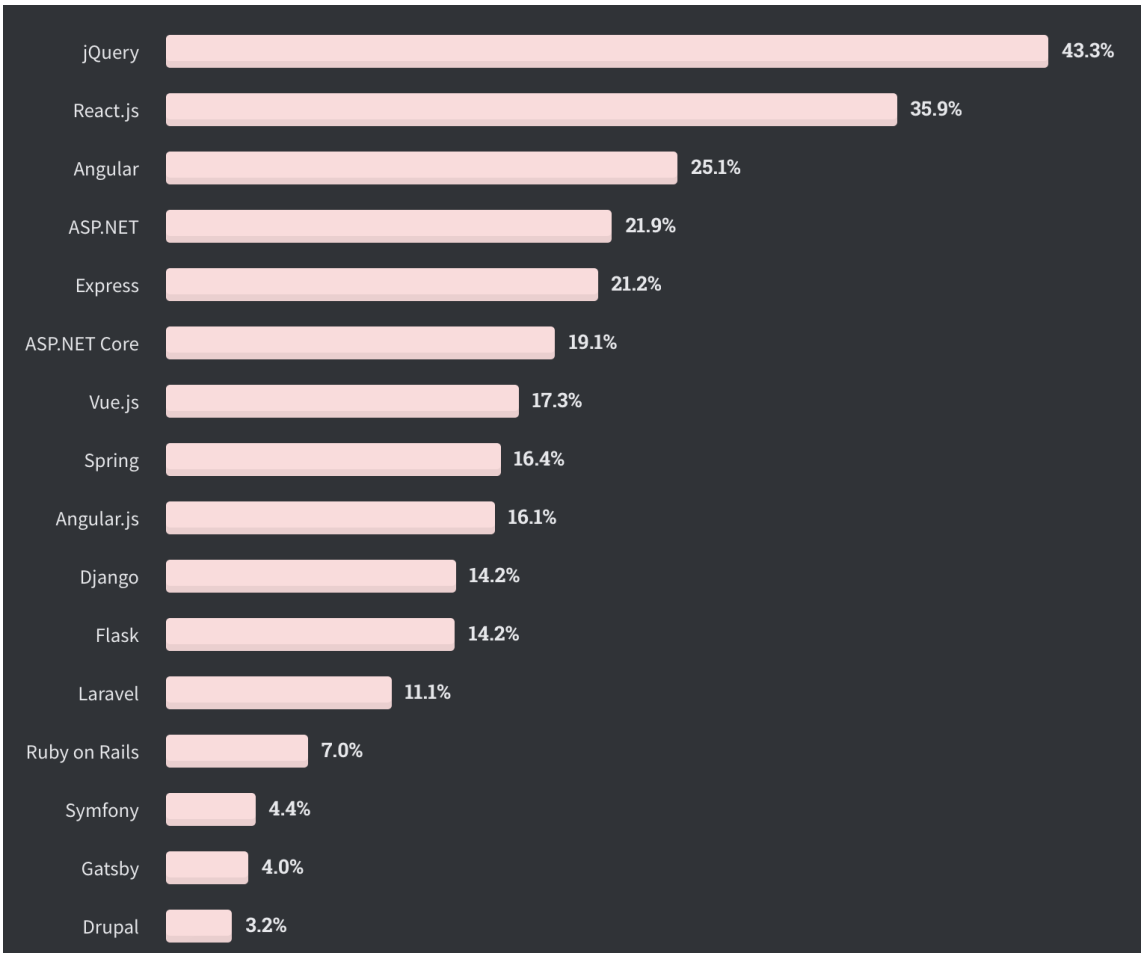


Figura 12: Ranking das frameworks segundo Stack Overflow. Retirado de Overflow (2020).

2.4 ANÁLISE DE MÉTRICAS E FRAMEWORKS WEB

Nesta secção pretende-se comparar frameworks web tendo por base diferentes métricas. Para tal, numa primeira fase selecionaram-se oito frameworks web, considerando os três *rankings* apresentados na Secção 2.2 e foi feita uma pequena apresentação de cada uma destas. Em seguida, foram selecionadas algumas métricas, tendo em consideração as características recolhidas na Secção 2.1. Por fim, foram feitos quadros comparativos dessas métricas nas frameworks anteriormente selecionadas.

2.4.1 *Seleção de Frameworks Web*

A necessidade de se fazer uma pré-seleção das frameworks a analisar surge por existirem tantas frameworks web e comparar todas elas seria demasiado trabalhoso e pouco produtivo.

Desta forma, analisando os rankings apresentados na Secção 2.3.1, foram selecionadas as seguintes frameworks web:

- Spring;
- ASP.NET MVC;
- Angular;
- Django;
- Vue.js;
- React;
- Ruby on Rails.

Spring

Spring é uma framework em camadas de Java EE, que inclui um "*lightweight container*" para configuração automática e cablagem de objetos de aplicação através da inversão de controlo (injeção de dependência), uma camada de abstração para gestão de transações, uma camada de abstração JDBC, funcionalidade AOP e integração com mapeadores O-R. A framework Spring pode também utilizar o Spring IDE, uma interface gráfica do utilizador para os ficheiros de configuração (Shan and Hua, 2006).

ASP.NET MVC

ASP.NET MVC permite desenvolver aplicações web dinâmicas com uma separação clara das preocupações e que permite um controlo total sobre a markup para um desenvolvimento agradável e ágil. Esta framework utiliza o padrão Model-View-Controller (MVC) e é considerada uma framework leve ("*lightweight*") e altamente testável que está integrada com características ASP.NET existentes, tais como *master pages* e autenticação baseada em membros (Islam et al., 2011).

Angular

Angular é uma framework web baseada em TypeScript, segue o padrão *Model-View-Viewmodel* (MVVM)[10] e baseia-se numa hierarquia de componentes. Desta forma, os componentes são os principais blocos de construção e podem exibir informação, renderizar modelos e realizar ações nos dados. Angular possui vários tipos de *data binding* dentro de um componente, como o *property binding*, *event binding* e *two-way binding* (Wohlgethan, 2018).

Django

Django é uma full-stack framework web baseada em Python (Plekhanova, 2009) e baseia-se no princípio MVC (Liawatimena et al., 2018). Django foca-se em automatizar o máximo possível e caracteriza-se pelo lema "não se

10 https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93viewmodel

repetir a si mesmo" (Plekhanova, 2009) (Liawatimena et al., 2018). Para além disso, concentra-se na ineficiência, dando ao programador a possibilidade de realizar quase todas as tarefas com o menor esforço possível de programação (código). Django é também escalável, maduro e rápido, com uma numerosa comunidade de programadores e um enorme conjunto de componentes incorporados. Django pode aceder ou criar instâncias de dados JSON ou XML e lidar com sistemas de gestão de bases de dados relacionais, tais como Oracle, MySQL, SQLite, e PostgreSQL. No *deployment*, Django é totalmente apoiado pela plataforma de nuvem AWS Elastic Beanstalk e Heroku (Liawatimena et al., 2018).

Vue.js

Vue.js é uma framework web de JavaScript, focada no desenvolvimento de interfaces gráficas. A arquitetura de Vue.js é normalmente descrita como *Model-View-Viewmodel* (MVVM). Vue.js destaca-se pelo seu elevado grau de escalabilidade e é caracterizada por ser baseada em componentes e usar diretivas nos *templates*, como para *data binding* e *event handling* (Wohlgethan, 2018).

React

React é uma framework web em JavaScript que tem como objetivo desenvolver interfaces gráficas. React é caracterizado por ser orientado por componentes e pela sua composição. O objetivo geral é transformar o estado atual da aplicação numa vista que possa ser apresentada ao utilizador (através da DOM). É possível escrever componentes através de duas diferentes abordagens: componentes como funções (mais recomendada) e componentes como classes ES6 (Wohlgethan, 2018).

Ruby on Rails

Ruby on Rails é uma framework server-side e caracteriza-se por seguir o princípio de "convenção sobre configuração" e de "não se repetir a si mesmo". As aplicações desenvolvidas em Rails têm que seguir o padrão *Model-View-Controller* (MVC) (Ignacio Fernández-Villamor et al., 2008).

Além disso, a arquitetura Rails oferece ao programador web um conjunto de serviços disponíveis, tais como a persistência de bases de dados independentes do fornecedor, geração automática de código para criação, leitura, atualização e eliminação de recursos ou testes integrados. A maioria das críticas da Ruby on Rails deve-se à dactilografia dinâmica da Ruby e à imaturidade da Ruby on Rails (Ignacio Fernández-Villamor et al., 2008).

2.4.2 *Comparação de Frameworks Web*

Por fim, apresenta-se um quadro comparativo com as frameworks web selecionadas na Secção 2.4.1 e com todas as características salientadas na Secção 2.2.1.

Tal como se pode ver na Tabela 1, no caso de Spring, no que diz respeito à Segurança, uma dos possíveis ferramentas é a *Spring Security* e de Computação em Nuvem é *Spring Cloud*. Em ASP.NET MVC, um exemplo para utilização de Computação em Nuvem é Azure Cloud Services.

Em relação a Angular, para *debug* é normalmente utilizado Augury e em Wohlgethan (2018) é feita uma referência em relação à existência de Documentação, de uma Comunidade e da possibilidade de integração de JavaScript.

A Segurança de Django é mencionada em Curie et al. (2019) e alguns exemplos para Computação em Nuvem são referidos em Salas Zarate et al. (2015), sendo eles dotCloud, Google App Engine e Amazon EC2. Pelo mesmo autor, para Suporte HTML5 é referido HTML5 Boilerplate e H5BP, para *Debugging* a Django Debug Toolbar, para ORM a Django ORM e para JavaScript é mencionado jQuery, ExtJS e Dojo. Por fim, ainda pelo mesmo autor, é mencionada a Documentação de Django, nomeadamente Sphinx. O Template Framework e as Plataformas Suportadas (Windows, Linux, OSX) são referidas em Curie et al. (2019).

No caso de Ruby on Rails, a possibilidade do desenvolvimento da camada de apresentação é mencionada em Ignacio Fernández-Villamor et al. (2008) e de Segurança em Curie et al. (2019). Em Salas Zarate et al. (2015), é mencionado para Computação em Nuvem Amazon EC2, Linode, Rackspace e Heroku. Para *Debugging*: *debug*, *to_yaml* e *inspect* e para ORM é ActiveRecord. Por fim, pelo mesmo autor, para JavaScript é referido jQuery, *script.aculo.us* e Dojo e o nome da Documentação é Rdoc. A inexistência de HTML5 é mencionada em Curie et al. (2019) e Salas Zarate et al. (2015) e as plataformas suportadas (Windows, Linux, OSX) em Curie et al. (2019).

	Spring	ASP.NET MVC	Angular	Django	Vue.js	React	Ruby on Rails
Apresentação	Sim	Sim	Sim	Sim	Sim	Sim	Sim
Segurança	Sim	Sim	Sim	Sim	Sim	Sim	Sim
Computação em Nuvem	Sim	Sim	Sim	Sim	Sim	Sim	Sim
Suporte HTML5	Sim	Sim	Sim	Sim	Sim	Sim	Não
Template Framework	Sim	Sim	Sim	Sim	Sim	Sim	Sim
Testes	Sim	Sim	Sim	Sim	Sim	Sim	Sim
Suporte de Plataforma	Windows, Linux, OSX	Windows, Linux, OSX	Windows, Linux, OSX	Windows, Linux, OSX	Windows, Linux, OSX	Windows, Linux, OSX	Windows, Linux, OSX
Debugging	Sim	Sim	Sim	Sim	Sim	Sim	Sim
ORM	Sim	Sim	Sim	Sim	Sim	Sim	Sim
JavaScript	Sim	Sim	Sim	Sim	Sim	Sim	Sim
Documentação	Sim	Sim	Sim	Sim	Sim	Sim	Sim
Comunidade	Sim	Sim	Sim	Sim	Sim	Sim	Sim
Validação	Sim	Sim	Sim	Sim	Sim	Sim	Sim
Usabilidade/ Curva de Aprendizagem	Boa	Boa	Muito boa	Boa	Muito boa	Muito boa	Boa

Tabela 1: Quadro Comparativo de Frameworks.

2.5 DESENVOLVIMENTO DE INTERFACES DE UTILIZADOR BASEADOS EM MODELOS

A atual diversidade de dispositivos disponíveis e dos respetivos tamanhos, formas e estilos ("form factors") aumenta a necessidade de abordagens baseadas em modelos para apoiar a adaptação de aplicações de um dispositivo para outro. Isto é particularmente relevante na área do desenvolvimento web, uma vez que existe um elevado leque de browsers e de tipos de dispositivos (Silva and Campos, 2012).

O Desenvolvimento de Interfaces de Utilizador baseados em modelos (MBUID) é uma base para a resolução deste problema. Os modelos e as transformações dos modelos permitem repensar o desenvolvimento do design da interface gráfica, independente dos detalhes de implementação, e redefinir modelos para concretizar essas interfaces gráficas. As UIDLs, Linguagens de descrição das Interfaces de Utilizador, suportam a descrição de uma interface a vários níveis de abstração e a realização de transformações entre esses níveis (Silva and Campos, 2012).

A *Cameleon Reference Framework* (G. Calvary, 2002) para o desenvolvimento baseado em modelos de interfaces de utilizador *multi-target* identifica quatro desses níveis (Silva and Campos, 2012):

- *Concepts and Task model* - descreve as tarefas a executar e as entidades que os utilizadores manipulam no seu cumprimento;
- *Abstract User Interface* (AUI) - descreve a interface do utilizador independentemente de qualquer modalidade concreta de interação e da plataforma informática;
- *Concrete User Interface* (CUI) - descreve uma instanciação da AUI para um conjunto concreto de modalidades de interação;
- *Final User Interface* (FUI) - corresponde à interface do utilizador que está a correr numa plataforma, seja por ser executada ou interpretada.

Resumidamente, a *Cameleon Reference Framework* decompõe a concepção da interface do utilizador numa série de diferentes componentes que procuram reduzir o esforço em atingir os múltiplos contextos de utilização[11].

11 https://www.wx3.org/community/uad/wiki/Cameleon_Reference_Framework

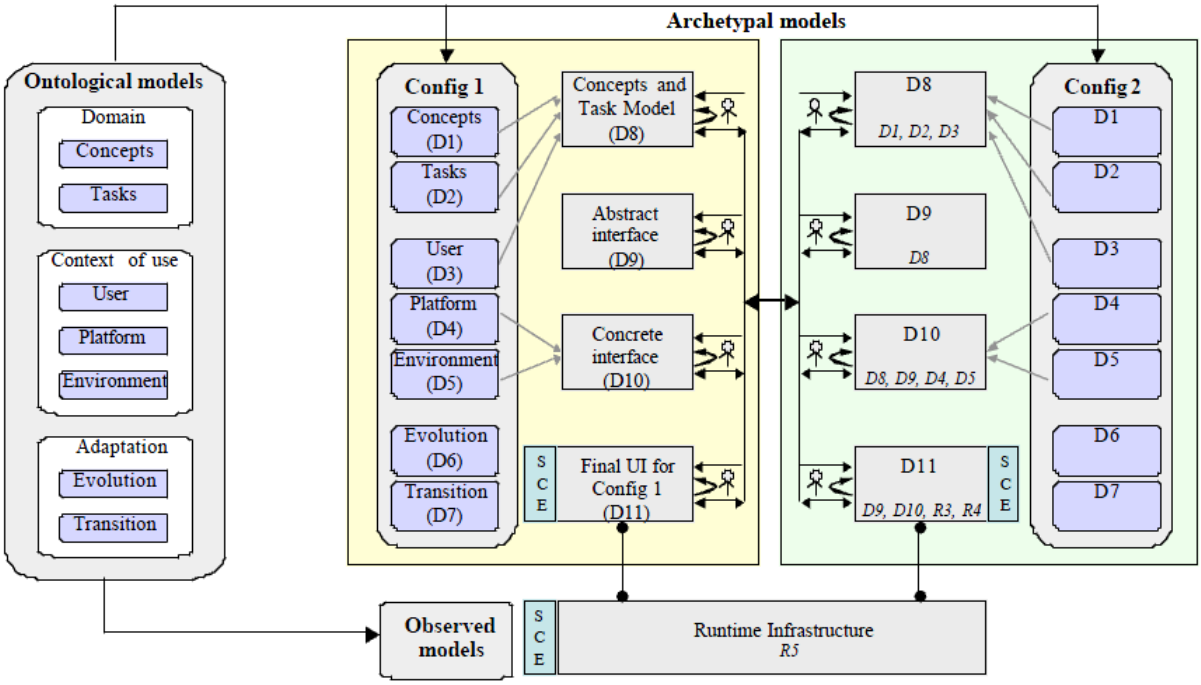


Figura 13: Estrutura da *Cameleon Reference Framework*. Retirado de G. Calvary (2002).

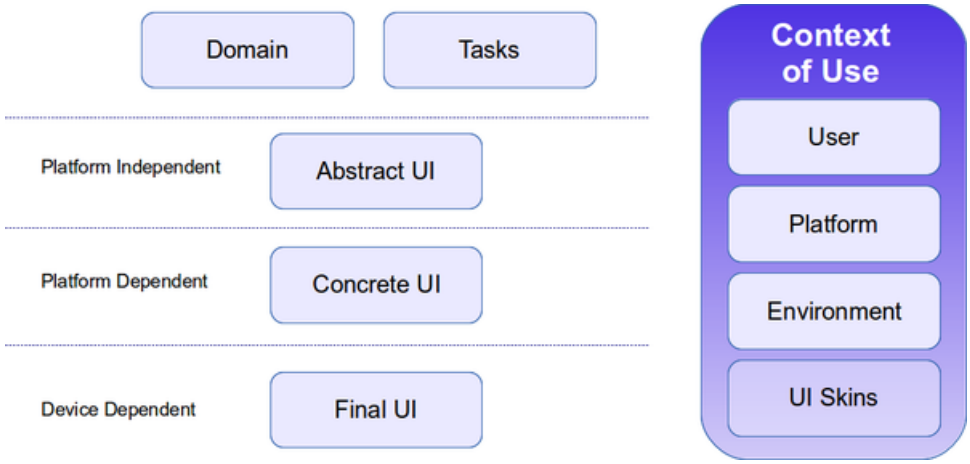


Figura 14: *Cameleon Reference Framework*. Retirado de [11].

3

CONCLUSÃO E TRABALHO FUTURO

Ao longo do presente documento, foi possível entrar em detalhe em relação ao estado atual da área de desenvolvimento web. Para isso, recorreu-se à definição de alguns conceitos, como o de aplicação web, framework web, *client-side* e *server-side* e analisou-se as vantagens e desvantagens associadas à utilização de frameworks. Para além disso, foram apresentados vários padrões de arquiteturas de software para o desenvolvimento de aplicações web, nomeadamente a arquitetura de três camadas, MVC, MVP e MVVM.

Desta forma, foi possível obter uma visão mais alargada do estado atual da área de desenvolvimento web e assimilar bem todo o caminho e evolução desta área de desenvolvimento.

Consequentemente, as características mais relevantes das frameworks web foram estudadas e foi feito um resumo das características mais mencionadas. Desta forma, foi possível perceber melhor o que caracteriza uma framework e quais são os aspetos ou propriedades que devemos ter em consideração aquando da escolha da framework a utilizar no próximo projeto web.

Em seguida, foram estudadas várias frameworks web, através da análise de diferentes rankings que tinham por base diferentes premissas. As frameworks melhores classificadas nesses rankings foram selecionadas, culminando assim numa comparação dessas frameworks com base nas características previamente recolhidas.

Por fim, foi feita uma introdução ao tema de Desenvolvimento de Interfaces de Utilizador baseados em modelos, utilizando como referência a *Cameleon Reference Framework*.

3.1 TAREFAS

- Tarefa 1: Estado da Arte (4 meses).
  - Tarefa 1.1: Contexto geral da área de Desenvolvimento Web;
  - Tarefa 1.2: Características das Frameworks de Desenvolvimento Web;
  - Tarefa 1.3: Frameworks de Desenvolvimento Web;
  - Tarefa 1.4: Análise de Métricas e Frameworks Web;
  - Tarefa 1.5: Desenvolvimento de Interfaces de Utilizador baseados em modelos.
- Tarefa 2: Criar um modelo genérico de framework (2 meses).
- Tarefa 3: Utilização de modelo genérico para auxiliar a geração de interface (2 meses).

- Tarefa 4: Implementação mockup para modelo genérico que gerará a interface (2 meses).
- Tarefa 5: Escrita de artigo científico (1 mês).
- Tarefa 6: Escrita da dissertação (6 meses).
- Tarefa 7: Preparação da defesa final (1 mês).

3.1.1 *Tarefa 1*

A Tarefa 1 consiste no Estado da Arte da área de desenvolvimento web. Desta forma, na Tarefa 1.1 o objetivo é fazer um enquadramento geral da área, apresentando algumas definições cruciais como de aplicação web, framework web (e respetivas vantagens e desvantagens da sua utilização), *server-side*, *client-side* e arquiteturas de software. Na Tarefa 1.2 o objetivo é apresentar as características mais importantes que as frameworks web devem concretizar, seguido de uma análise das características mais mencionadas na literatura.

Em seguida, a Tarefa 1.3 visa apresentar as frameworks mais relevantes da atualidade, segundo alguns rankings que têm por base diferentes critérios de seleção. Posteriormente, na Tarefa 1.4 o objetivo é comparar algumas frameworks através da apresentação de um quadro comparativo.

Por fim, na Tarefa 1.5 é feita uma introdução ao tema de Desenvolvimento de Interfaces de Utilizador baseados em modelos, utilizando como ponto de partida a *Cameleon Reference Framework*.

3.1.2 *Tarefa 2*

Na Tarefa 2 é pretende-se a realização de um modelo genérico de framework. Para tal, numa fase inicial deve ser feito o modelo de apenas uma framework (Vue.js) e, consequentemente, proceder-se à análise sequencial de outras frameworks e tentar enquadrá-las no modelo construído.

Desta forma, iterativamente, serão feitas revisões e alterações ao modelo inicialmente proposto, de modo a que este se torne genérico o suficiente para conseguir modelar todas as frameworks analisadas.

3.1.3 *Tarefa 3*

Na Tarefa 3 o objetivo passa pela utilização do modelo genérico previamente construído para a geração do código da interface gráfica.

3.1.4 *Tarefa 4*

A Tarefa 4 visa desenvolver o processo de conversão de um mockup (protótipo desenhado por um designer de software) no modelo genérico. Desta forma, será possível converter o próprio mockup no código da framework.

3.1.5 *Tarefa 5*

Um dos objetivos da presente dissertação é a elaboração de um artigo científico que será realizado na Tarefa 5.

3.1.6 *Tarefa 6*

A Tarefa 6 pode ser traduzida como a conclusão da escrita da dissertação.

3.1.7 *Tarefa 7*

Por fim, a Tarefa 7 consiste na preparação da defesa final.

3.2 CALENDARIZAÇÃO DO TRABALHO DA DISSERTAÇÃO

Através do seguinte Diagrama de Gantt é possível identificar os meses em que cada uma das tarefas anteriormente descritas irá ser desenvolvida.

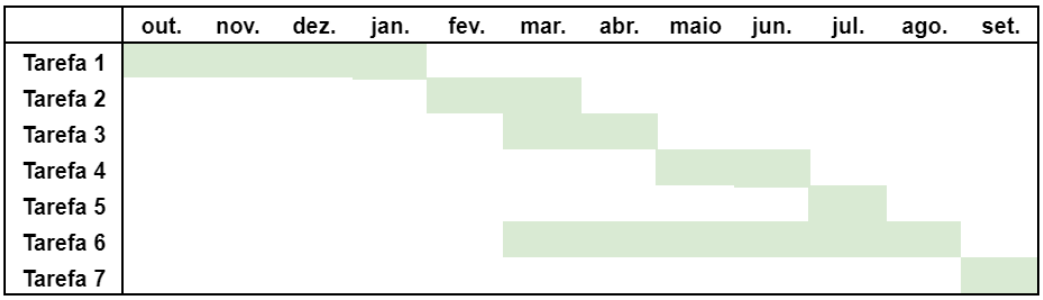


Figura 15: Diagrama de Gantt.

Desta forma, após a conclusão de todas as tarefas, dar-se-á por concluída a dissertação proposta.

BIBLIOGRAFIA

Princípios do manifesto Ágil. URL https://agilemanifesto.org/iso/ptpt/principles.html.

Sabah Al-Fedaghi. Developing web applications. *International Journal of Software Engineering and Its Applications*, 5, 05 2011. doi: 10.1007/978-1-4302-3531-6_12.

Paris Avgeriou and Uwe Zdun. Architectural patterns revisited - a pattern language. volume 81, pages 431–470, 01 2005.

Arno Bergmann. Benefits and drawbacks of model-based design. *KMUTNB International Journal of Applied Science and Technology*, 7:15–19, 09 2014. doi: 10.14416/j.ijast.2014.04.004.

Dasari Curie, Joyce Jaison, Jyoti Yadav, and J Fiona. Analysis on web frameworks. *Journal of Physics: Conference Series*, 1362:012114, 11 2019. doi: 10.1088/1742-6596/1362/1/012114.

et al. G. Calvary. The cameleon reference framework, 2002. D1.1 of the CAMELEON Project R&D Project IST-2000-30104.

José Ignacio Fernández-Villamor, Laura Díaz-Casillas, and Carlos Á. Iglesias. A comparison model for agile web frameworks. In *Proceedings of the 2008 Euro American Conference on Telematics and Information Systems*, EATIS '08, New York, NY, USA, 2008. Association for Computing Machinery. doi: 10.1145/1621087.1621101.

Md Islam, Md Islam, and Tasneem Halim. A study of code cloning in server pages of web applications developed using classic asp.net and asp.net mvc framework. pages 497–502, 12 2011. ISBN 978-1-61284-907-2. doi: 10.1109/ICCITechn.2011.6164840.

John Kouraklis. *MVVM as Design Pattern*. 10 2016. ISBN 978-1-4842-2213-3. doi: 10.1007/978-1-4842-2214-0_1.

Suryadiputra Liawatimena, Harco Leslie Hendric Spits Warnars, Agung Trisetyarso, Edi Abdurahman, Benfano Soewito, Antoni Wibowo, Ford Gaol, and Bahtiar Abbas. Django web framework software metrics measurement using radon and pylint. pages 218–222, 09 2018. doi: 10.1109/INAPR.2018.8627009.

Rob Marvin. The best low-code development platforms, Aug 2018. URL https://www.pcmag.com/roundup/353252/the-best-low-code-development-platforms. (Retrieved January 25, 2021).

Zakaria Mehrab, Raquib Bin Yousuf, Ibrahim Asadullah Tahmid, and Rifat Shahriyar. Mining developer questions about major web frameworks. pages 191–198, 01 2018. doi: 10.5220/0006929501910198.

Gerrit Meixner, Fabio Paternò, and Jean Vanderdonckt. Past, present, and future of model-based user interface development. *i-com*, 10:2–11, 11 2011. doi: 10.1524/icom.2011.0026.

Mohamadou Nassourou. Java web frameworks which one to choose? 01 2010.

Vensada Okanovic. Web application development with component frameworks. pages 889–892, 05 2014. ISBN 978-953-233-077-9. doi: 10.1109/MIPRO.2014.6859693.

Stack Overflow. Developer survey results 2017, 2017. URL https://insights.stackoverflow.com/survey/2017. (Retrieved January 12, 2021).

Stack Overflow. 2020 developer survey, 2020. URL https://insights.stackoverflow.com/survey/2020#technology-programming-scripting-and-markup-languages. (Retrieved January 12, 2021).

J. Plekhanova. Evaluating web development frameworks: Django, ruby on rails and cakephp. 2009.

Dragos-Paul Pop and Adam Altar Samuel. Designing an mvc model for rapid web application development. *Procedia Engineering*, 69, 12 2014. doi: 10.1016/j.proeng.2014.03.106.

María Salas Zarate, Giner Alor-Hernández, Rafael Valencia-García, Lisbeth Rodríguez, Alejandro González, and José Cuadrado. Analyzing best practices on web development frameworks: The lift approach. *Science of Computer Programming*, 102, 05 2015. doi: 10.1016/j.scico.2014.12.004.

Tony Shan and Winnie Hua. Taxonomy of java web application frameworks. pages 378–385, 10 2006. doi: 10.1109/ICEBE.2006.98.

Carlos Eduardo Silva and José Creissac Campos. Can gui implementation markup languages be used for modelling? In Marco Winckler, Peter Forbrig, and Regina Bernhaupt, editors, *Human-Centered Software Engineering*, pages 112–129, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. ISBN 978-3-642-34347-6.

Emily Stevens. The fascinating history of ux design: A definitive timeline, July 2019. URL https://careerfoundry.com/en/blog/ux-design/the-fascinating-history-of-ux-design-a-definitive-timeline/. (Retrieved January 24, 2021).

Iwan Vosloo and Derrick Kourie. Server-centric web frameworks: An overview. *ACM Comput. Surv.*, 40, 04 2008. doi: 10.1145/1348246.1348247.

Gottfried Vossen and Stephan Hagemann. From version 1.0 to version 2.0: A brief history of the web. 01 2007.

Jari-Pekka Voutilainen, Jaakko Salonen, and Tommi Mikkonen. On the design of a responsive user interface for a multi-device web service. pages 60–63, 05 2015. doi: 10.1109/MobileSoft.2015.16.

Irena Vuksanovic and Bojan Sudarevic. Use of web application frameworks in the development of small applications. pages 458–462, 01 2011.

D. Walker and A. Orooji. Metrics for web programming frameworks. 2011.

Eric Wohlgethan. Supporting web development decisions by comparing three major javascript frameworks: Angular, react and vue.js, 2018.