Patrícia Isabel Cunha de Jesus

Impactos da Automatização de Testes no Meio Empresarial

Relatório de Estágio submetido como requisito parcial para obtenção do grau de **Mestre em Engenharia de Software**

PRESIDENTE
Professor José António Moinhos Cordeiro, ESTSetúbal/IPS

ARGUENTE
Professora Patrícia Alexandra Pires Macedo, ESTSetúbal

ORIENTADOR
Professor Nuno Miguel Vicente de Pina Gonçalves, ESTSetúbal/IPS

SUPERVISOR
Eng.º Pedro Oliveira - Atos It Solutions And Services Unipessoal

Novembro 2022

Para os meus pais e para a minha irmã.

Sem vocês, tenho a certeza que não tinha conseguido.

Agradecimentos

Para já ficam os seguintes agradecimentos:

À instituição que me acolheu durante estes dois últimos anos, Escola Superior de Tecnologia de Setúbal, do Instituto Politécnico de Setúbal, onde realizei este mestrado. A todos os professores, com quem privei e com quem aprendi muito ao longo destes anos e sempre estiveram disponíveis para me ajudar em qualquer questão.

Aos meus colegas de curso, que em muitos momentos fomos o apoio uns dos outros, foi um privilégio passar esta etapa da minha vida ao vosso lado.

À Atos, por mais uma vez me deixarem realizar um projeto de estágio e ao fim de 3 anos continuarem a apostar em mim para fazer parte da grande empresa que são.

Um agradecimento, em particular, ao meu Supervisor Pedro Oliveira, por todo o apoio e disponibilidade dados ao longo de todo o estágio. A todos os meus colegas de equipa, o meu muito obrigada, por toda a ajuda, por tudo o que me ensinam e por toda a colaboração e união diária.

Ao meu orientador de estágio da Escola Superior de Tecnologia de Setúbal, Professor Nuno Pina Gonçalves, um agradecimento especial por no meio de tantos alunos e tanto trabalho ter aceitado orientar o meu estágio. Agradeço ainda pela sua disponibilidade ao ajudar-me a esclarecer as dúvidas que me ocorreram durante o estágio.

A todos os meus amigos, que com muito orgulho, acompanharam esta fase da minha vida, sempre presentes manifestando sempre o seu apoio.

A todos os meus familiares, que estão e aos que infelizmente já não estão presentes, pelo apoio incondicional e por tudo o que fizeram por mim, para que pudesse chegar a este momento. Principalmente aos meus pais, por toda a paciência, carinho e força que me foram dando para que eu pudesse estudar e seguir os meus sonhos.

À minha irmã Mariana, por acreditar sempre em mim e nas minhas capacidades, como ninguém desde o primeiro dia, pela força e carinho, a ela o meu muito obrigada.

Obrigada a todos, por tudo!

Resumo

O presente documento é referente à experiência no estágio, que foi a opção escolhida para se realizar no âmbito da Unidade Curricular de Estágio/Projeto do Mestrado de Engenharia de Software, da Escola Superior de Tecnologia de Setúbal (ESTSetúbal), pertencente ao Instituto Politécnico de Setúbal (IPS).

Este período de estágio contou com a integração de vários projetos, em dois clientes da empresa que é a entidade empregadora, com o objetivo de automatização de testes. Esta experiência permitiu uma integração, observação e análise daquilo que é o meio empresarial em Portugal e daquela que é a vida de um profissional na área de QA.

Este trabalho propõe uma solução que envolve o estudo e apresentação de alternativas às ferramentas utilizadas no mercado, com especial foco em testes automatizados.

Foi elaborado um estudo de mercado que avalia que tipo de propostas de emprego existem na área de QA, quais as necessidades do mercado, questões salariais envolvidas e se os planos de formação a nível académico estão adequados ou são suficientes no que toca a formar ou consciencializar profissionais a enveredar por áreas relacionadas com a qualidade software.

Palavras-chave: QA, Mercado, Testes Automatizados, Ferramentas, Qualidade e Empresas

Abstract

This document refers to the internship experience, which was the option chosen to be carried out within the scope of the Internship/Project of the Master in Software Engineering, at the Escola Superior de Tecnologia de Setúbal (ESTSetúbal), belonging to the Instituto Politécnico de Setúbal (IPS).

This internship period included the integration of several projects, in two customers of the company that is the employer, with the objective of automating tests. This experience allowed an integration, observation and analysis of what is the business environment in Portugal and what is the life of a professional in the QA area.

This work proposes a solution that involves the study and presentation of alternatives to the tools used in the market, with a special focus on automated tests.

A market study was carried out to assess what type of job offers there are in the QA area, what the market needs are, salary issues involved and whether academic training plans are adequate or sufficient in terms of training or raising awareness professionals to embark on areas related to software quality.

Keywords: QA, Market, Automated Tests, Tools, Quality and Companies

Índice

Agradecimentos ..... **iv**

Resumo ..... **v**

Abstract ..... **vi**

Índice ..... **vii**

Lista de Figuras ..... **ix**

Lista de Tabelas ..... **xi**

Lista de Siglas e Acrónimos ..... **xii**

Capítulo 1 ..... **1**

1. Introdução **1**

1.1. Estrutura do documento **1**

1.2. Motivação **2**

1.3. Contexto **2**

1.4. Solução **3**

Capítulo 2 ..... **5**

2. Enquadramento Organizacional do Estágio **5**

2.1. Planeamento **6**

2.1.1. Acompanhamento de Projeto(s) *6*

2.1.2. Preparação *7*

2.1.3. Desenvolvimento *7*

Capítulo 3 ..... **8**

3. Trabalho Relacionado **8**

3.1. *Quality Assurance* (QA) e a sua importância num projeto de software **8**

3.1.1. Quality Assurance no meio empresarial *8*

3.1.2. Atividades de um grupo Software Quality Assurance *10*

3.1.3. O papel de um profissional de QA *11*

3.1.4. Vantagens da implementação do Quality Assurance *12*

3.2. Os vários tipos de testes e a sua importância **13**

3.2.1. Teste vs Debugging *14*

3.2.2. Processo de teste de Software *15*

3.2.3. Tipos de Testes *16*

3.2.4. Testes Manuais vs Testes Automáticos *18*

3.3. Testes Automáticos **20**

3.3.1. Vantagens da automatização de Testes *21*

3.3.2. Desvantagens e limitações da automação de testes 22
3.3.3. O papel das ferramentas de automação 23
3.4. Plataformas *Low Code* **25**
3.4.1. Característica de uma Plataforma Low Code 28
3.4.2. Risco de efeitos de bloqueio 31
Capítulo 4 ....**32**
4. Trabalho Realizado **32**
4.1. Cliente 1 ....**32**
4.1.1. TestPath 35
4.1.2. Resultados e conclusões 59
4.2. Cliente 2 ....**63**
4.2.1. FRAT 66
4.2.2. Resultados e conclusões 75
4.3. TestPath vs FRAT **77**
4.4. Estudo de mercado **79**
4.4.1. Procura no mercado 79
4.4.2. Questões salariais 83
4.4.3. Formação Académica 85
Capítulo 5 ....**97**
5. Conclusão **97**
5.1. Trabalho Futuro **99**
Referências **100**

Lista de Figuras

Figura 1- Diagrama de processo de testes 16
Figura 2- Diagrama da integração das várias tecnologias com o TestPath no ciclo de teste 37
Figura 3- Página de inserção de credenciais 39
Figura 4- Escolha de Menu 39
Figura 5- Opções rápidas 40
Figura 6- Exemplo de painéis com dados das opções rápidas 41
Figura 7- Lista de Testes por Projeto 41
Figura 8- Exemplo de criação de uma pasta de projeto 41
Figura 9- Exemplo de criação de um teste 42
Figura 10- Opções a realizar durante um teste 42
Figura 11- Detalhes de um teste 43
Figura 12- Submenu Test Resources 44
Figura 13- Submenu *Applications* 44
Figura 14- Opções do submenu *Applications* 44
Figura 15- Exemplo de sessão *Spy* 45
Figura 16- Exemplo de elementos mapeados 46
Figura 17- Exemplo de localização de objeto 46
Figura 18- Exemplo de *Steps* de uma *Keyword* e respetivos argumentos 47
Figura 19- Exemplo de dados inseridos no *Data Driven* 48
Figura 20- Exemplo da criação de registo na BD 49
Figura 21- Submenu de *Webservices* 50
Figura 22- Exemplo de uma lista de variáveis 51
Figura 23- Lista de opções de ações a aplicar sob as variáveis 51
Figura 24- Exemplo de uma lista de *defects* 52
Figura 25- Exemplo de uma *RunSuite* e respetivas execuções 52
Figura 26- Exemplo de várias execuções de teste em tempo real 53
Figura 27- Opção que permite visualizar evidências em vários formatos 53
Figura 28- Exemplo de visualização de um *Runner* 54
Figura 29- Exemplo de agendamento de execuções através do calendário 55
Figura 30- Exemplo de inserção de dados para agendamento 55
Figura 31- Menu de *Dashboard* 55
Figura 32- Exemplo do *dashboard* de *Test Cases* 56
Figura 33- Exemplo do *dashboard* da lista de execuções 56
Figura 34- Exemplo do *dashboard* de *Keywords* 57
Figura 35- Exemplo da lista de estatísticas dos utilizadores 57
Figura 36- Exemplo do *dashboard* da lista de *RunSuites* 57
Figura 37- Exemplo da lista de *Defects* 58

Figura 38- Diagrama da integração das várias tecnologias no ciclo de teste 68
Figura 39- Estrutura de um projeto com utilização de Cucumber 69
Figura 40- Diagrama de arquitetura simples do Selenium Grid 71
Figura 41- Diagrama de arquitetura simples do Selenium WebDriver 72
Figura 42- Resultados da pesquisa feita pelo termo “qa tester” 80
Figura 43- Resultados da pesquisa feita pelo termo “qa manual tester” 81
Figura 44- Resultados da pesquisa feita pelo termo “qa automation tester” 81
Figura 45- Resultados das pesquisas feitas pelo termo “qa automation tester” nas plataformas Glassdoor e Indeed 82
Figura 46- Resultados da pesquisa feita pelo termo “software *developer*” 83
Figura 47- Gráfico do intervalo de valores salariais para um QA Tester 84
Figura 48- Gráfico do intervalo de valores salariais para um Software Developer 84
Figura 49- Plano curricular do Mestrado em Engenharia Informática da UNL-FCT 87
Figura 50- Plano curricular do Mestrado em Engenharia de Software do IPS-ESTS 87
Figura 51- Plano curricular da Licenciatura em Engenharia Informática da UFP 94
Figura 52- Plano curricular da Licenciatura em Engenharia Informática da U. Portucalense 95
Figura 53- Plano curricular do Mestrado em Engenharia de Software da FEUP 95
Figura 54- Plano curricular do Mestrado em Engenharia Informática do ISEP 96

Lista de Tabelas

Tabela 1- Plano de trabalho 6
Tabela 2- Requisitos de Software e Hardware para instalação do TestPath 38
Tabela 3- Diferenças entre o TestPath e FRAT 78
Tabela 4- Cursos da área de TI nos estabelecimentos de Ensino Superior em Setúbal 86
Tabela 5- Cursos da área de TI nos estabelecimentos de Ensino Superior em Lisboa 89
Tabela 6- Cursos da área de TI nos estabelecimentos de Ensino Superior em Braga 91
Tabela 7- Cursos da área de TI nos estabelecimentos de Ensino Superior no Porto 93

Lista de Siglas e Acrónimos

4GL	Quarta geração de programação
AI	Inteligência artificial
APIs	Interfaces de programação de aplicações
BI	Business intelligence
BPM	Gestão de processos de negócios
BPMN	Business Process Model and Notation
CISQ	Consórcio da qualidade do software de TI
CLOUD	Espécie de dispositivo de armazenamento de dados colocado na Internet
DBMS	Sistemas de gestão de base de dados
FRAT	Framework para automação de testes
GUI	Interface gráfica de utilizador
IDEs	Ambientes de desenvolvimento integrados
IOS	iPhone Operating System
IPS	Instituto Politécnico de Setúbal
JDBC	Java Database Connectivity
LCDP	Plataformas de Desenvolvimento Low-Code
LCPs	Plataformas Low-Code
MDD	Desenvolvimento orientado a modelos
MVC	Model-View-Controller
QA	Quality Assurance
RAD	Desenvolvimento Rápido de Aplicações
RESTful	Representational state transfer
RPA	Robotic Process Automation
SQA	Software Quality Assurance
TI	Tecnologias de Informação
TP	TestPath
UE	União Europeia
WfMS	Sistemas de gestão de fluxo de trabalho

Capítulo 1

1. Introdução

Este documento tem como objetivo detalhar todo o trabalho realizado no âmbito da unidade curricular Projeto/Estágio do Mestrado em Engenharia de Software, da Escola Superior de Tecnologia de Setúbal do Instituto Politécnico de Setúbal no ano letivo 2021/2022.

Para esta Unidade Curricular optou-se pela realização de um estágio. O estágio contou com uma duração de 1458 horas, tendo este decorrido entre as datas 10 de janeiro de 2022 e 16 de setembro de 2022, na empresa Atos IT Solutions and Services, Unipessoal, LDA, sob a supervisão de Pedro Oliveira, como supervisor de estágio na organização e do Professor Nuno Pina Gonçalves, como professor orientador por parte do IPS.

1.1. Estrutura do documento

Nesta secção apresentam-se algumas considerações gerais relativas à estrutura deste documento.

O presente documento está dividido em 5 capítulos, ordenados de modo a tornar mais fácil a compreensão, havendo sempre uma ligação entre estes:

- Introdução
- Enquadramento Organizacional do Estágio
- Trabalho Relacionado
- Trabalho Realizado
- Conclusão

Em primeiro lugar, contamos com uma parte introdutória que incluí uma breve introdução, seguida da exposição das motivações, objetivos e soluções propostas para esta dissertação.

Posteriormente, no 2º capítulo, é feita a descrição do enquadramento organizacional, os clientes para os quais foram prestados serviços, função nas equipas a integrar e os projetos que serviram de análise. É também apresentado todo o planeamento e metodologias utilizadas durante todo o período de estágio.

No capítulo seguinte, encontramos todo o trabalho de pesquisa e estudos relacionados com a temática escolhida para ser abordada nesta dissertação.

De seguida, no capítulo número 4, é feita toda a descrição do trabalho realizado, os fundamentos tecnológicos aprendidos e aperfeiçoados, referências às ferramentas utilizadas para automatização de testes nos vários projetos e os resultados obtidos. Este capítulo é talvez o mais importante de toda a dissertação uma vez que é aqui que é pretendido provar que a solução proposta para o problema inicial, faz sentido.

Por fim, serão expostas as conclusões e reflexões de toda a experiência vivida e conhecimento adquirido, assim como as perspetivas e trabalhos futuros.

Após o capítulo conclusivo, são ainda apresentadas todas as referências bibliográficas e webgráficas, consultadas ao longo da realização desta dissertação de estágio.

Em seguida, é descrita a motivação para a realização deste estágio, a descrição do contexto em que este se insere e a solução para o problema apresentado, prevista no âmbito desta dissertação. Esta secção está dividida nas seguintes subsecções:

- Motivação;
- Contexto;
- Solução.

1.2. Motivação

A área de testes tem vindo a crescer muito nos últimos anos e a valorização dessas atividades tem vindo a ser algo muito necessário tendo em conta que as empresas precisam da cultura de qualidade nos seus produtos e nos seus colaboradores em geral.

Com o crescimento desta área, tem vindo a existir uma tremenda falta de profissionais, num mercado com muita oferta para pouca “mão-de-obra”. Esta também se deve ao desinteresse existente pela área, que entre muitos fatores está relacionada com a falta de formação a nível académico e ao desconhecimento da existência da área de *Quality Assurance* (QA) e do que consiste a atividade de um profissional nos vários cargos da mesma.

É certo que um profissional na área de QA tem cada vez mais espaço no mercado e com uma maior valorização, mas ainda existe muito por fazer e conquistar.

É importante demonstrar as consequências que estes factos, anteriormente mencionados, estão e irão provocar a longo prazo, na qualidade do desenvolvimento de um software, ou seja, a falta de qualidade no software produzido.

1.3. Contexto

Esta dissertação tem como objetivo explorar e desenvolver soluções para melhor resolver aquilo que é a perspetiva e realidade de um profissional de QA, no meio empresarial e em

Portugal.

Com especial foco, no alerta para a importância que os testes automáticos têm num projeto de engenharia de software e os benefícios de estes acompanharem todo o processo de desenvolvimento desde início. Com isto, salientar a temática dos testes de regressão ou de não regressão que, à medida que novas *releases* e novas funcionalidades vão sendo testadas e entregues, é necessário garantir através destes testes que o que já foi entregue nas versões anteriores continua a funcionar corretamente e que não existem danos colaterais com origem nas novas funcionalidades.

Historicamente um dos maiores desafios é tentar com uma certa frequência provar a importância de um *Tester* no processo *end-to-end* do desenvolvimento de um projeto software.

As empresas procuram por profissionais com determinados conhecimentos como:

Testes Funcionais, Testes Automatizados, Testes de Performance, Testes de *Backend, Test Manager* e etc.

É cada vez mais difícil encontrar profissionais qualificados com a experiência que as empresas exigem.

Em Portugal esta realidade não seria diferente, porém a essência do *tester* funcional manual continua intacta na maioria dos cenários. O que muitas vezes acrescenta complexidade dos testes automáticos com ferramentas tradicionais de testes, devido à maior parte de os profissionais de testes terem mais *skills* funcionais do que técnicas.

Também a nível académico, o facto de não se atribuir muita importância à qualidade do que se é produzido, mas apenas ao que é produzido. Existe um foco, de uma forma geral, muito maior para o desenvolvimento e não para todas as restantes áreas dentro do mundo das tecnologias de informação.

1.4. Solução

É percetível, com a contextualização feita anteriormente, que todos estes pontos conduzem ao problema maior, que é a falta de qualidade de um software a médio/longo prazo.

De modo a solucionar este problema, é importante ter conhecimento das várias ferramentas/tecnologias/plataformas existentes no mercado e quais poderão trazer mais vantagens, das metodologias associadas e mais proveitosas, e sobretudo, a integração das equipas de QA juntamente com as restantes equipas das outras áreas envolvidas nos projetos (*developers*, analistas, etc.).

Apesar de se ter em conta todo o tipo de testes e de todos terem uma importância significativa, nesta dissertação, optou-se por dar enfâse aos testes automáticos e ao que os seus benefícios trazem num projeto de engenharia de Software.

É sabido que algumas empresas já apostam neste tipo de testes, mas a grande maioria continua a apostar só e unicamente nos testes funcionais devido a grande parte dos *testers* não terem conhecimentos técnicos que permitam trabalhar com as ferramentas já existentes no mercado. Essa falta de conhecimentos também se deve ao facto de os profissionais

contratados não terem uma formação (licenciaturas, mestrados e etc.) que passe pelas áreas das tecnologias de informação. É um entrave, de certa forma, que uma formação inicial não resolve e envolve custos.

Como tal, a Atos desenvolveu uma ferramenta de automatização de testes, denominada *TestPath*, semelhante às já existentes e bastante utilizadas no mercado, mas com a particularidade de não ser necessário proceder à codificação para desenvolvimento e execução dos testes.

Através da apresentação e utilização desta ferramenta para automatizar os testes, nos projetos associados a este período de estágio, juntamente com as metodologias associadas e conjunto de boas práticas, é pretendido avaliar os impactos resultantes e solucionar o problema que motivou o desenvolvimento desta dissertação.

Capítulo 2

2. Enquadramento Organizacional do Estágio

Neste capítulo irá ser feito um enquadramento daquilo que foram as funções desempenhadas, os clientes para os quais foram prestados serviços, funcionamento e organização das equipas durante o período de estágio. Vão ser também descritos o planeamento e a metodologia utilizados durante este mesmo período.

Como referido no início deste documento, o estágio foi realizado na empresa Atos It Solutions. Porém as tarefas desenvolvidas no estágio foram realizadas em dois dos clientes da empresa.

Por motivos de confidencialidade, ao longo deste documento nunca serão revelados os nomes destas duas entidades, nem os respetivos nomes ou temáticas dos projetos associados às mesmas. Sendo que para uma fácil identificação estes clientes irão ser sempre referidos como Cliente 1 e Cliente 2. Os respetivos projetos seguirão a mesma nomenclatura sendo sempre associados e identificados junto do cliente para o qual estavam a ser desenvolvidos.

A ideia inicial seria a integração num projeto, de um dos clientes da empresa, cujo papel desempenhado seria o de *tester* de automação. Como tal o objetivo seria o acompanhamento desse mesmo projeto no que diz respeito à fase do ciclo de vida de um teste como:

- Planeamento de testes.
- Monitorização e controlo de teste.
- Análise de teste
- Conceção de teste
- Implementação de teste
- Execução de teste
- Conclusão de teste

Tudo isto utilizando as ferramentas e metodologias definidas pelo cliente, uma vez que a integração iria ser numa equipa que já estava “dentro” do projeto e já tinha a sua própria metodologia estipulada e ferramentas definidas, não dando outras hipóteses de escolha, apenas de adaptação. No que diz respeito aos testes, na equipa, já existiria apenas um único *tester*, sendo este um *tester* funcional manual.

Ao contrário do que foi descrito para a ideia inicial, o objetivo não diferiu, apenas no período de estágio, que por questões de mercado e de necessidade da empresa, foi dividido entre dois clientes e em cada cliente o acompanhamento foi de 2 ou mais projetos, cada um com as suas ferramentas e metodologias aplicadas. Esta alteração pode ser explicada pelo facto referido anteriormente que é também um dos problemas apresentados para a elaboração desta dissertação. No capítulo 1 deste documento, devido a ser uma área relativamente recente no mercado não existem profissionais qualificados suficientes para fazer a automação de testes apenas num único projeto.

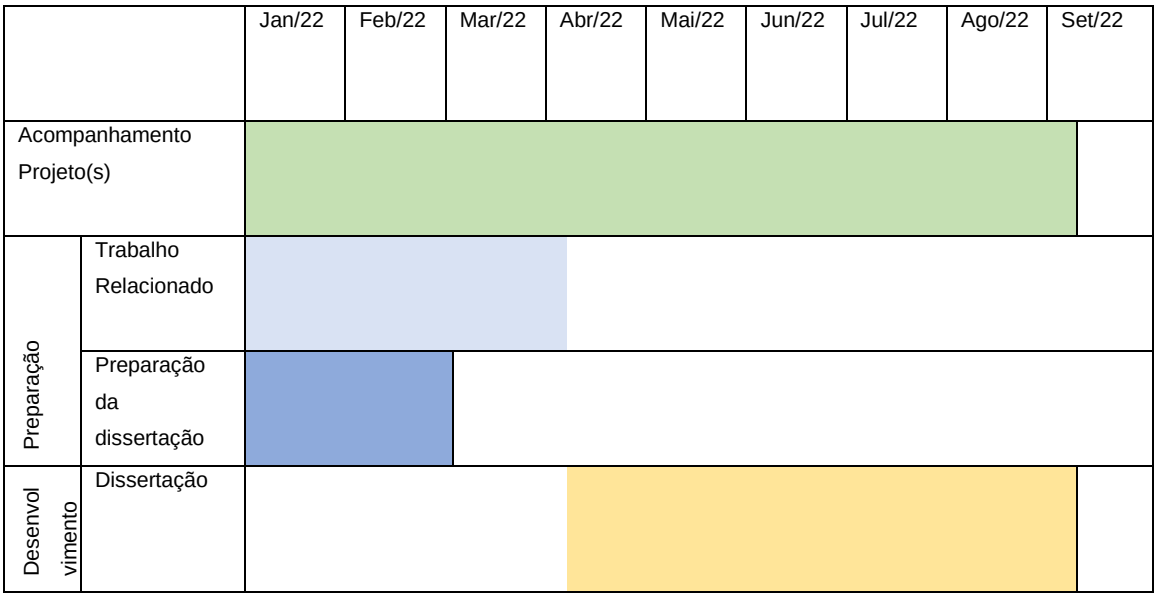
2.1. Planeamento

Nesta secção, o foco principal é apresentar o plano de trabalho desenhado para desenvolver a solução apresentada anteriormente, no âmbito desta dissertação, como a calendarização.

Como referido no capítulo introdutório, esta dissertação foi baseada num estágio com a duração de 9 meses, com início no dia 10 de janeiro de 2022. De modo a obter um planeamento, foram divididos estes 9 meses (1458 horas), por semanas, em que a cada semana correspondem tarefas.

Posto isto, e com o desenvolvimento da solução em mente, é mostrado o plano de trabalho feito com a adição de um cronograma de desenvolvimento desta dissertação, visto na seguinte tabela:

Tabela 1- Plano de trabalho



Com o anterior cronograma (Tabela 1) em mente, o plano de trabalho foi dividido em 3 fases:

- Fase de Acompanhamento de Projeto(s)
- Fase de Preparação
- Fase de Desenvolvimento

2.1.1. Acompanhamento de Projeto(s)

Esta fase consistiu no acompanhamento/participação de um ou mais projetos, conforme as necessidades da entidade empregadora de colocação em um ou mais clientes.

Foi a fase que acompanhou todo o tempo de estágio, e teve como objetivo colocar em prática todos os conceitos aprendidos e aproveitar a perspetiva de um profissional da área de QA para tirar algumas conclusões daquilo que é praticado no meio empresarial de modo a ajudar no

desenvolvimento do estudo e desenvolvimento desta dissertação.

2.1.2. Preparação

Esta fase foi composta pelo estudo de trabalhos relacionados na área de foco desta dissertação, de modo a conseguir elaborar uma estrutura de todos os pontos a abordar. Como tal, houve o desenvolvimento de uma pesquisa bibliográfica para obter uma sustentação teórica sobre o assunto através de artigos, internet, trabalhos e experiências anteriores, formando as referências da pesquisa.

2.1.3. Desenvolvimento

Esta fase foi constituída pela redação contínua do documento de dissertação. Foi a fase que ocupou as restantes semanas do estágio em simultâneo com a fase de integração em projeto. Nesta etapa foi descrito e detalhado todo o trabalho realizado dentro dos projetos envolventes e respetivos estudos com base no que já existia, e foi trabalhado no Capítulo 3 de trabalho relacionado, e nos resultados que foram obtidos durante a automatização de testes nos projetos.

Ao contrário do que foi planeado para a estrutura inicial desta dissertação, vão existir capítulos dedicados exclusivamente ao estágio, nomeadamente o Capítulo 4 – Trabalho realizado, por questões de melhor entendimento e clarividência. Não colocando em causa algumas questões de confidencialidade.

Sendo que o plano inicial seria descrever o tempo experienciado dentro dos capítulos com o tema respetivo de estudo, ou seja, cada capítulo contaria com uma secção para descrição da experiência profissional dentro do contexto da temática do capítulo.

Porém, era algo de que já existia consciência, uma vez que 9 meses muita coisa muda e dentro do mercado na área das TI as alterações são uma constante, como foi o caso.

Deixar apenas como nota que todo este período foi realizado em regime remoto, ou seja, em teletrabalho.

Capítulo 3

3. Trabalho Relacionado

Neste capítulo apresentam-se alguns estudos e pesquisas relacionadas com o âmbito desta dissertação. O capítulo está dividido nas seguintes seções:

- *Quality Assurance* (QA) e a sua importância num projeto de software
- Os vários tipos de testes e a sua importância
- Testes Automáticos
- Plataformas *Low Code*

3.1. *Quality Assurance* (QA) e a sua importância num projeto de software

A qualidade de software é um conceito que traz algum desafio, alguns *developers* de software continuam a acreditar que é algo com o qual nos devemos preocupar depois de o código ser gerado. Nada poderia estar mais errado. (Pressman, 2015)

Antes de qualquer produto ser lançado, a qualidade desse mesmo produto deve ser assegurada, garantindo, a satisfação do cliente, no mínimo. É exatamente isso que torna o trabalho de um profissional de *Quality Assurance* tão importante para todo o processo de desenvolvimento. (Lacerda, 2019)

Quality Assurance é muitas vezes confundido com um "*Tester*". No entanto, QA é mais do que um *tester*, é um processo de engenharia que inclui um conjunto de atividades que tentam garantir que o produto ou serviço final oferecidos estejam de acordo com o nível de qualidade exigido. (Pombo, 2020)

Tem como propósito atingir os objetivos do projeto avaliando o desempenho com base em padrões de qualidade relevantes e também nos requisitos do cliente. Fornecendo visibilidade à gestão sobre os processos e padrões que estão a ser seguidos e cumpridos, e os produtos de trabalho que estão a ser produzidos na organização, eficazmente ou não. É uma investigação sistemática sobre a forma como as coisas são feitas na organização e envolve a realização de auditorias de projetos, fornecedores e departamentos. (O'Regan, 2002)

3.1.1. *Quality Assurance* no meio empresarial

Além de saber como funciona o *Quality Assurance* e a sua importância, é essencial saber também como implementá-lo a nível empresarial. Afinal, sem uma adesão adequada, os resultados podem ficar aquém do esperado, sem aproveitar o máximo de eficiência que se poderia obter. (Gaea,

2018) A garantia de qualidade de software abrange uma ampla gama de preocupações e atividades que se concentram na gestão da qualidade de software:

Padrões- O trabalho de QA é garantir que os padrões que foram adotados sejam seguidos e que todos os produtos de trabalho estejam em conformidade com eles. (O'Regan, 2002)

Revisões e auditorias- As revisões técnicas são uma atividade de controlo de qualidade realizada por engenheiros de software para engenheiros de software, a principal função é descobrir erros.

As auditorias têm como finalidade o relato de resultados para os grupos afetados. Estas auditorias, geralmente são conduzidas pelo grupo *Software Quality Assurance* (SQA), grupo este que é independente dos grupos auditados. O grupo SQA atua como representante interno do cliente. Ou seja, as pessoas que executam SQA devem olhar para o software do ponto de vista do cliente. (Pressman, 2015)

A preocupação do planeamento num processo de auditoria é selecionar os projetos/áreas a serem auditadas, determinar quem precisa de estar envolvido e lidar com a logística. Uma reunião de auditoria foca-se numa reunião formal com os participantes da mesma para discutir responsabilidades específicas no projeto, os processos seguidos e assim por diante. Por fim, um relatório de auditoria detalha os resultados detalhados da mesma e inclui ações de auditoria do que é preciso para ser resolvido. Assim que o relatório de auditoria é publicado, o auditor acompanhará as ações atribuídas até à conclusão e uma vez que todas as ações tenham sido abordadas a auditoria pode então ser encerrada. (O'Regan, 2002)

Teste- O teste de software é um controlo de qualidade cuja principal função é encontrar erros. O trabalho da SQA é garantir que os testes sejam adequadamente planeados e conduzidos com eficiência para que tenham a maior probabilidade de atingir o seu objetivo principal.

Recolha e análise de erros/defeitos- A única maneira de melhorar é medir os resultados. A SQA recolhe e analisa dados de erros e defeitos para entender melhor como os erros são introduzidos e quais as atividades de engenharia de software são mais adequadas para eliminá-los.

Mudança de gestão- A mudança é um dos aspetos mais disruptivos de qualquer projeto de software. Se não for gerida adequadamente, a mudança pode levar à entropia, que quase sempre leva à má qualidade. SQA garante que as práticas adequadas de gestão de mudanças são instituídas.

Educação- Toda as empresas de software querem melhorar as suas práticas, um dos principais contribuintes para essa melhoria é a educação de engenheiros de software. A SQA deve assumir a liderança na melhoria do processo de software e é um dos principais proponentes e patrocinadores de programas educacionais. (Lacerda, 2019)

Gestão de fornecedores- É importante que exista, uma gestão de fornecedores. Estes são uma parte importante do processo de desenvolvimento de soluções e têm implicações diretas na qualidade do produto desenvolvido. Com uma gestão eficiente, realizando avaliações constantes sobre a capacidade de eles suprirem as necessidades internas numa empresa. O trabalho da organização SQA é garantir resultados de software de alta qualidade, sugerindo práticas de qualidade

específicas que o fornecedor deve seguir (quando possível) e incorporar padrões de qualidade como parte de qualquer contrato com um fornecedor externo. (Gaea, 2018)

Gestão de segurança- Com o aumento do cyber crime e as novas regulamentações governamentais relacionadas com a privacidade, todas as empresas de software devem instituir políticas que protejam os dados em todos os níveis, estabelecer proteção de firewall para *WebApps* e garantir que o software não seja adulterado internamente. A SQA garante que o processo e a tecnologia apropriados sejam usados para alcançar a segurança do software.

Segurança- Como o software é quase sempre um componente de sistemas de classificação humana, o impacto de defeitos ocultos pode ser catastrófico. SQA pode ser responsável para avaliar o impacto da falha de software e para iniciar essas etapas necessárias para reduzir riscos.

Gestão de riscos- Embora a análise e mitigação de riscos seja preocupação dos engenheiros de software, a organização SQA garante que as atividades de gestão de risco sejam conduzidas adequadamente e que sejam estabelecidos planos de contingência relacionados ao risco.

3.1.2. Atividades de um grupo *Software Quality Assurance*

Além de cada uma dessas preocupações e atividades, o SQA trabalha para garantir que atividades de suporte de software (por exemplo, manutenção, linhas de ajuda, documentação e manuais) são conduzidos ou produzidos com a qualidade como preocupação dominante. (Gaea, 2018)

O grupo SQA ajuda a equipa de software a alcançar um produto final de alta qualidade. Um conjunto de atividades SQA recomendado, que abordam o planeamento de garantia de qualidade, supervisão, registo manutenção, análise e relatórios.

Planeamento- O plano é desenvolvido como parte do projeto e é revisto por todas as partes interessadas. O plano identifica avaliações a serem realizadas, auditorias e revisões a serem conduzidas, padrões que são aplicáveis ao projeto, procedimentos para relatório e rastreamento de erros, produtos de trabalho que são produzidos pelo grupo SQA e feedback que será fornecido à equipa de software.

Descrição do processo- A equipa de software seleciona um processo para o trabalho a ser executado. O grupo SQA revê a descrição do processo para conformidade com a política organizacional, padrões internos de software, padrões impostos externamente e outras partes do plano do projeto de software.

Revisão- O grupo SQA identifica, documenta e rastreia desvios do processo e verifica se as correções foram feitas. (O'Regan, 2002)

Auditoria- As revisões do grupo SQA selecionadas para os produtos de trabalho: identifica, documenta e rastreia desvios; verifica que foram feitas correções; e reporta periodicamente os resultados do seu trabalho para o gestor de projeto. (O'Regan, 2002)

Documentação de procedimentos- Garante que os desvios nos trabalhos do software e nos produtos de trabalho sejam documentados e tratados. Os desvios podem ser encontrados no plano

do projeto, descrição do processo, padrões ou produtos de trabalho de engenharia de software. (Gaea, 2018)

Registo- Qualquer não conformidade é relatada à administração. Os itens de não conformidade são rastreados até que sejam resolvidos.

Qualidade de requisitos- A exatidão, integridade e consistência do modelo de requisitos terá uma forte influência na qualidade de todos os produtos de trabalho. O SQA deve garantir que a equipa de software revê adequadamente o modelo de requisitos para atingir um alto nível de qualidade.

Qualidade do projeto- Cada elemento definido na fase de desenho deve ser avaliado pela equipa de software para garantir que existe alta qualidade e que o projeto em si está em conformidade com os requisitos. O SQA procura métricas que sejam indicadores de qualidade.

Qualidade do código- O código-fonte e os produtos de trabalho relacionados devem estar em conformidade com os padrões de codificação locais e exibir características que facilitem a manutenção. SQA deve isolar aqueles atributos que permitem uma análise razoável da qualidade do código.

Eficácia do controle de qualidade- Uma equipa de software deve aplicar recursos limitados de maneira que tenha uma maior probabilidade de alcançar um resultado de alta qualidade. SQA analisa a alocação de recursos para revisões e testes para avaliar se eles estão a ser alocados da maneira mais eficaz.

Quando a QA é uma parte ativa de todo o ciclo de desenvolvimento, os testes são integrados a toda a estratégia de criação. Isso permite que possíveis bugs sejam descobertos no início e corrigidos imediatamente, evitando perda de tempo e dinheiro. No entanto é importante saber que os procedimentos e abordagens SQA que funcionam num ambiente de software podem não funcionar tão bem noutros. Mesmo que uma empresa adote uma abordagem consistente à engenharia de software, diferentes produtos de software podem apresentar diferentes níveis de qualidade. (Pressman, 2015)

A solução para esse dilema é entender os requisitos específicos de qualidade para um produto de software e, em seguida, selecionar o processo e ações específicas de SQA e tarefas que serão usadas para atender a esses requisitos.

3.1.3. O papel de um profissional de QA

O trabalho do profissional de QA envolve um processo sistemático de realização de testes, focado no processo de desenvolvimento, que visa detetar falhas e erros no software, assegurando que tudo funciona em conformidade. Isto é necessário para garantir que o produto final não chega às mãos do cliente com erros ou problemas de mau funcionamento. (Pombo, 2020)

Durante o processo de desenvolvimento de software, a equipa de desenvolvimento pode deparar-se com algumas dúvidas, como, se algo que não seja alterado pode vir a ser afetado e portar-se de forma estranha, por exemplo. O papel de um profissional de QA é crucial nesta fase,

permitindo à equipa de desenvolvimento responder com mais confiança a qualquer questão que inevitavelmente vá acontecendo à medida que o projeto evolui.

Além do domínio destes conceitos e tipologias de testes, um bom profissional de QA deve possuir certas características que lhe permitam um desempenho eficiente e um papel decisivo no processo de desenvolvimento. (Pombo, 2020) Deve ter boas capacidades de comunicação, deve ser um bom "caçador" (procura de bugs), um bom repórter, um bom colega de equipa, e claro, um bom examinador (tester) atento aos detalhes.

Para aplicar o *Quality Assurance*, é essencial contar com um profissional de qualidade para auxiliar no processo, formar uma equipa que seja responsável pela gestão desse processo internamente. O primeiro passo é encontrar o gestor que será responsável por esse processo, deve ser especialista na área e será responsável por aplicar os testes de qualidade durante todo o processo de criação, produção e implementação das soluções criadas numa empresa de TI. (Gaea, 2018) A busca por este tipo de profissionais neste momento é um enorme desafio, por se tratar de uma área relativamente recente.

É importante também que exista uma constante formação dos recursos humanos, os colaboradores envolvidos no processo de desenvolvimento precisam de incorporar uma mentalidade de qualidade total e aplicá-la em todas as etapas da elaboração do software. Para que isso ocorra, é necessário realizar formações que ensinem os conceitos e estratégias de *Quality Assurance*. (Gaea, 2018)

Estas devem ser feitas com a presença do gestor, que será responsável por essa área e devem ser repetidas periodicamente para reforçar essas questões e trazer novas tendências e ferramentas para a garantia da qualidade nos procedimentos específicos.

3.1.4. Vantagens da implementação do *Quality Assurance*

A implementação do *Quality Assurance*, segundo (Lacerda, 2019), traz uma série de benefícios para qualquer empresa de TI, tais como:

- Entrega de produtos com maior qualidade, de acordo com a "declaração de qualidade";
- Melhora a experiência do cliente;
- O cliente fica mais satisfeito, já que recebe um produto que pode não só atender, mas superar as suas expectativas, com o mínimo de falhas possível;
- Os profissionais conseguem ficar mais satisfeitos, já que veem um resultado mais proeminente das suas atividades;
- Auxilia a antecipar problemas e saber como resolvê-los de antemão;
- Eleva o nome de uma empresa no mercado;
- Auxilia a reduzir custos operacionais, já que minimiza a necessidade de *refactoring*;
- Cria referências positivas, o que auxilia a aumentar a base de clientes;
- Com clientes fidelizados e novas indicações, é possível conseguir receitas constantes para o seu negócio.

Com estes benefícios e respetivo feedback, é possível perceber que a adesão ao *Quality Assurance* pode ser bastante benéfica para o qualquer negócio, conseguindo alavancar resultados e fazer com que uma empresa cresça no mercado. (Pressman, 2015)

Como toda esta realidade é muito nova, as equipas de QA nem sempre conseguem contar com padrões e ferramentas de testes eficientes. As mais comuns no mercado ainda apresentam limitações no seu funcionamento e, por isso, não atendem a todas as necessidades. (Pressman, 2015) Por se tratar de um segmento que, apesar de muito novo, está em constante evolução, outro problema é a falta de profissionais qualificados para atuação e gestão da área de QA.

A alternativa que se tem apresentado para a solução deste problema é a automação do controlo de qualidade. É uma opção que pode atender à demanda do mercado de apps, promover melhorias no processo de desenvolvimento ágil e viabilizar a criação de novos padrões e procedimentos para testes de QA. (Pressman, 2015)

Para entrar ainda mais no mundo QA é importante reter também os seguintes conceitos:

Test Plan - Documento que descreve a estratégia de teste, objetivos, estimativa e recursos necessários para os testes. O Plano de Teste ajuda-nos a determinar o esforço necessário para validar a qualidade do software em teste. O Plano de Testes terá uma série de conjuntos de testes.

Test Suite- Parte do Plano de Testes. É um conjunto de casos de teste.

Test Case- Uma execução do software que pode expor um bug.

Edge Case- É um caso de teste que ocorre em condições raras.

Happy Path- Trata-se de um caso de teste bem definido, cujo resultado é o esperado.

Test Instructions- Instruções necessárias, passo a passo, para executar um caso de teste.

Acceptance Criteria- Conjunto de condições que devem ser verificadas, para que a característica possa ser aceite para ir para lançamento.

3.2. Os vários tipos de testes e a sua importância

Os sistemas de software são parte integrante da vida atual, das aplicações de negócio aos produtos de consumo. A maioria das pessoas já teve uma experiência com software que não funcionou conforme esperado. Software que não funciona corretamente pode provocar muitos problemas, incluindo perdas de tempo, dinheiro ou reputação de negócio.

Os testes de software são uma forma de avaliar a qualidade do software e de reduzir o risco de falha de software em funcionamento. (Sánchez-Gordón et al., 2018)

Um equívoco comum sobre os testes é que estes consistem apenas na execução de testes, ou seja, na execução do software e verificação dos resultados. O teste de software é um processo que inclui muitas atividades diferentes. (Sánchez-Gordón et al., 2018)

O processo de teste também inclui atividades como planeamento de testes, análise, conceção e

implementação de testes, informação acerca do progresso e de resultados dos testes e avaliação da qualidade dum objeto de teste. Inclui também a verificação/revisão do resultado de outras fases do processo de trabalho, tais como, requisitos, *user stories* e código-fonte.

Outro equívoco comum em relação aos testes é que estes se concentram inteiramente na verificação de requisitos, *user stories* ou outras especificações. Embora os testes impliquem verificar se o sistema cumpre os requisitos especificados, também implica validação, que consiste em verificar se o sistema irá satisfazer as necessidades do utilizador e de outros *stakeholders* no(s) seu(s) ambiente(s) operacional(is). (Sánchez-Gordón et al., 2018)

As atividades de teste são organizadas e efetuadas de forma diferente em diferentes ciclos de vida.

Para um determinado projeto, os objetivos de teste podem incluir:

- Prevenir defeitos, ao avaliar produtos de trabalho, como requisitos, *user stories*, conceção e código.
- Verificar se todos os requisitos especificados foram cumpridos.
- Verificar se o objeto de teste está completo e validar se funciona como os utilizadores e outros *Stakeholders* esperam.
- Aumentar a confiança no nível de qualidade do objeto de teste.
- Encontrar defeitos e falhas, reduzindo assim o nível de risco de qualidade inadequada do software.
- Fornecer informações suficientes aos *Stakeholders* para que estes tomem decisões informadas, especialmente no que respeita ao nível de qualidade do objeto de teste.
- Cumprir os requisitos ou normas contratuais, legais ou regulamentares e/ou verificar a conformidade do objeto de teste com esses requisitos ou normas.

Os objetivos de teste podem variar, dependendo do contexto do componente ou sistema que está a ser testado, do nível de teste e do modelo de ciclo de vida de desenvolvimento de software. Por exemplo, estas diferenças podem incluir:

- Durante o teste de componentes, um objetivo pode ser encontrar o maior número possível de falhas para que os defeitos subjacentes sejam identificados e corrigidos atempadamente. Outro objetivo pode ser aumentar a cobertura dos testes de componentes face ao código desenvolvido.
- Durante o teste de aceitação, um objetivo pode ser confirmar que o sistema funciona conforme esperado e satisfaz os requisitos. Outro objetivo deste teste pode ser fornecer informações aos *Stakeholders* sobre o risco de lançar o sistema num determinado momento.

3.2.1. Teste vs Debugging

O teste e o *debugging* são atividades diferentes. A execução de testes pode demonstrar falhas causadas por defeitos no software. O *debugging* é a atividade de desenvolvimento que deteta, analisa e corrige esses defeitos. Testes de confirmação subsequentes verificam se as correções

resolveram os defeitos. Em alguns casos, os *testers* são responsáveis pelo teste inicial e pelo teste de confirmação final, enquanto os programadores realizam o *debugging*, testes de componentes e testes de integração de componentes associados (integração contínua). (Sánchez-Gordón et al., 2018)

No entanto, no desenvolvimento Agile, e nalguns outros ciclos de vida de desenvolvimento de software, os *testers* podem estar envolvidos no *debugging* e nos testes de componentes.

3.2.2. Processo de teste de Software

Não existe um processo universal de teste de software, mas existem conjuntos comuns de atividades de teste, sem os quais os testes terão menos probabilidade de atingir os seus objetivos estabelecidos. Estes conjuntos de atividades de teste formam um processo de teste.

O processo de teste de software adequado e específico numa determinada situação depende de muitos fatores. Questões como que atividades de teste estão envolvidas neste processo de teste, como estas atividades são implementadas e quando estas atividades ocorrem podem ser debatidas na estratégia de teste duma organização.

Os fatores contextuais que influenciam o processo de teste duma organização, incluem, mas não se limitam a:

- Modelo de ciclo de vida de desenvolvimento de software e metodologias de projeto utilizadas.
- Níveis de teste e tipos de teste considerados.
- Riscos de produto e riscos de projeto.
- Área de negócio.
- Limitações operacionais, incluindo, mas não se limitando a:
    - Orçamentos e recursos.
    - Prazos.
    - Complexidade.
    - Requisitos contratuais e regulamentares.
- Políticas e práticas organizacionais.
- Normas internas e externas aplicáveis.

Um processo de teste consiste nos seguintes grupos de atividades principais:

- Planeamento de testes.
- Monitorização e controlo de teste.
- Análise de teste.
- Conceção de teste.
- Implementação de teste.
- Execução de teste.
- Conclusão de teste.

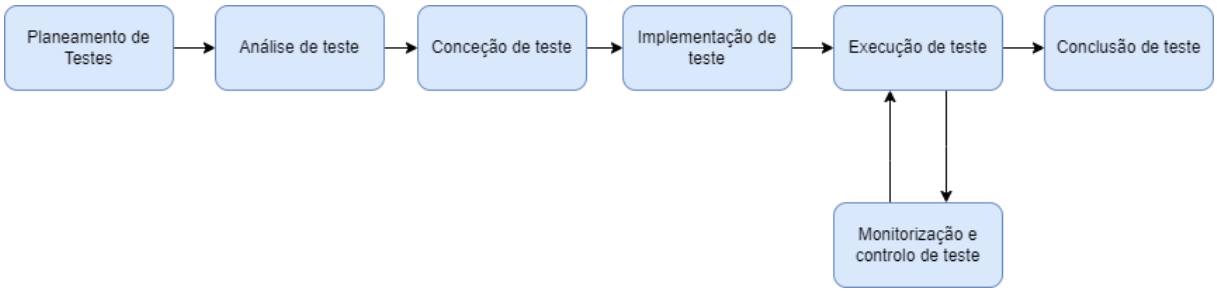


Figura 1- Diagrama de processo de testes

Cada grupo de atividades principal (Figura 1) é composto por atividades constituintes, cada uma consiste em várias tarefas individuais, que podem variar dum projeto ou lançamento (*release*) para outro.

Além disso, embora muitos destes grupos de atividades principais possam parecer logicamente sequenciais, são muitas vezes implementados iterativamente. Assim, as atividades de teste também ocorrem numa base iterativa e contínua nesta abordagem de desenvolvimento de software. Mesmo no desenvolvimento sequencial de software, a sequência lógica faseada dos grupos de atividades principais implicará sobreposição, combinação, simultaneidade ou omissão, pelo que é geralmente necessário adaptar estes grupos de atividades principais ao contexto do sistema e do projeto.

3.2.3. Tipos de Testes

Existem essencialmente quatro tipos de testes: Testes Funcionais, Testes não Funcionais, Testes Estruturais e Testes Relacionados com Alterações.

Testes Funcionais- são realizados para assegurar que o software cumpre os requisitos pretendidos. Os recursos são testados através da introdução de inputs e da análise dos resultados.

O teste funcional dum sistema implica testes que avaliam funções que o sistema deve executar. Os requisitos funcionais podem ser descritos em produtos de trabalho, como especificações de requisitos de negócio, *epics, user stories*, casos de uso ou especificações funcionais, ou podem não estar documentados. As funções são “o que” o sistema deve fazer.

Os testes funcionais devem ser realizados em todos os níveis de teste, embora o foco seja diferente em cada nível. (Sánchez-Gordón et al., 2018)

O teste funcional considera o comportamento do software, pelo que podem ser utilizadas técnicas caixa-preta para determinar as condições de teste e os casos de teste para a funcionalidade do componente ou sistema.

A exaustividade dos testes funcionais pode ser medida através da cobertura funcional. A cobertura funcional é a medida em que uma determinada funcionalidade foi executada pelos testes e é expressa em percentagem do(s) tipo(s) de elementos a ser abrangido(s).

A conceção e execução de testes funcionais podem envolver competências ou conhecimentos especiais, tais como o conhecimento do problema de negócio específico que o software resolve. (Sánchez-Gordón et al., 2018)

Testes Não Funcionais- destinam-se a verificar os aspetos não funcionais do software.

O teste não funcional dum sistema avalia as características de sistemas e software, como usabilidade, eficiência de desempenho ou segurança. O teste não funcional é o teste de "quão bem" o sistema se comporta.

Contrariamente às perceções erradas comuns, os testes não funcionais podem e devem ser realizados em todos os níveis de teste e realizados o mais cedo possível. A descoberta tardia de defeitos não funcionais pode ser extremamente perigosa para o sucesso dum projeto. As técnicas caixa-preta podem ser utilizadas para determinar as condições de teste e os casos de teste para testes não funcionais. (Sánchez-Gordón et al., 2018)

A exaustividade dos testes não funcionais pode ser medida através da cobertura não funcional. A cobertura não funcional é a medida em que um determinado tipo de elemento não funcional foi executado pelos testes e é expressa em percentagem do(s) tipo(s) de elementos abrangido(s). (Sánchez-Gordón et al., 2018)

A conceção e execução de testes não funcionais pode envolver competências ou conhecimentos especiais, tais como o conhecimento das deficiências inerentes a um projeto ou tecnologia ou a base de utilizadores específica.

Testes Estruturais ou Caixa-branca- O teste caixa-branca produz testes baseados na estrutura interna ou na implementação do sistema. A estrutura interna pode incluir códigos, arquitetura, fluxos de trabalho e/ou fluxos de dados no sistema.

A exaustividade dos testes caixa-branca pode ser medida através da cobertura estrutural. A cobertura estrutural é a medida em que um determinado tipo de elemento estrutural foi executado pelos testes e é expressa em percentagem do tipo de elemento abrangido. (Sánchez-Gordón et al., 2018)

No nível de teste de componentes, a cobertura de código baseia-se na percentagem de código de componente que foi testada e pode ser medida em termos de diferentes aspetos do código (itens de cobertura), como a percentagem de instruções executáveis testadas no componente ou a percentagem de resultados de decisão testados. Estes tipos de cobertura são coletivamente chamados de cobertura de código. No nível de teste de integração de componentes, o teste caixa-branca pode basear-se na arquitetura do sistema, como interfaces entre componentes, e a cobertura estrutural pode ser medida em termos da percentagem de interfaces executadas por testes.

A conceção e a execução do teste caixa-branca podem envolver competências ou conhecimentos especiais, tais como a forma como o código é criado, como os dados são armazenados e como usar ferramentas de cobertura e interpretar corretamente os seus resultados. (Sánchez-Gordón et al., 2018)

Testes Relacionados com Alterações- Este tipo de testes serve para quando são feitas alterações num sistema, para corrigir um defeito ou devido a funcionalidades novas ou alteradas, o teste deve ser feito para confirmar que as alterações corrigiram o defeito ou implementaram a

funcionalidade corretamente e não causaram consequências adversas imprevistas.

 Teste de confirmação: Depois dum defeito ser corrigido, o software pode ser testado com todos os casos de teste que falharam devido ao defeito, que deve ser executado novamente na nova versão do software. O software também pode ser testado com novos testes para cobrir as alterações necessárias para corrigir o defeito. No mínimo, as etapas para reproduzir a(s) falha(s) causada(s) pelo defeito devem ser novamente executadas na nova versão do software. O objetivo dum teste de confirmação é confirmar se o defeito original foi corrigido com sucesso.

 Teste de regressão: É possível que uma alteração efetuada numa parte do código, seja uma correção ou outro tipo de alteração, possa afetar acidentalmente o comportamento doutras partes do código, dentro do mesmo componente, noutros componentes do mesmo sistema ou mesmo noutros sistemas. As alterações podem incluir alterações no ambiente, como uma nova versão dum sistema operativo ou sistema de gestão de bases de dados. (Sánchez-Gordón et al., 2018)

Tais efeitos colaterais não intencionais são chamados de regressões. O teste de regressão implica a execução de testes para detetar tais efeitos secundários não intencionais.

Os testes de confirmação e de regressão são realizados em todos os níveis de teste.

Especialmente em ciclos de vida de desenvolvimento iterativo e incremental (p. ex., Agile), novas funcionalidades, alterações nas funcionalidades existentes e reestruturação (*refactoring*) de código resultam em alterações frequentes no código, o que também requer testes relacionados com alterações. Devido à natureza evolutiva do sistema, os testes de confirmação e regressão são muito importantes. Isto é particularmente relevante para sistemas *Internet of Things* em que objetos individuais (p. ex., dispositivos) são frequentemente atualizados ou substituídos. (Sánchez-Gordón et al., 2018)

3.2.4. Testes Manuais vs Testes Automáticos

Existem duas principais práticas em QA: Testes Manuais e Testes Automáticos.

Os testes manuais são realizados por seres humanos de forma manual e têm como finalidade reproduzir as ações dos utilizadores para validar se determinada funcionalidade ou correção está a funcionar conforme aquilo que é espetável pelo utilizador e abrangem o seguinte:

Testes de Compatibilidade- Asseguram que o software é compatível com diferentes browsers ou computadores/dispositivos. (Pombo, 2020) O teste de compatibilidade pode ajudar a entender os critérios que um sistema / aplicação precisam para atingir e cumprir objetivos, para serem aceites pelos utilizadores.

Smoke Tests- É um conjunto de testes que visam assegurar as funções mais importantes de um software. (Pombo, 2020) Devem ser executados na fase inicial do ciclo de vida de um teste, pois verifica muito brevemente a produtividade da construção e garante que os requisitos elementares podem estar a ser cumpridos ou não. Sempre que um novo *deployment* é efetuado, ou qualquer mudança é feita, o teste de fumo deve ser realizado.

O objetivo dos testes de fumo é testar as funcionalidades importantes de um sistema e não executar os cenários de teste de forma exaustiva.

Testes de Regressão- O teste de regressão é um tipo de teste de software que consiste na aplicação de versões mais recentes do software, para garantir que não surgiram novos defeitos em componentes já analisados. Asseguram que uma alteração recente do código não afetou negativamente as características existentes. (Pombo, 2020) Se, ao juntar o novo componente ou as suas alterações com os componentes restantes do sistema surgirem novos defeitos em componentes inalterados, então considera-se que o sistema regrediu.

Muitas vezes são usadas ferramentas específicas para os testes de regressão, ferramentas de automação. Estas conseguem um resultado mais exato do teste, ao executar exatamente os mesmos passos seguidos para o teste das primeiras versões, já que é permitida a gravação do teste. (Korel et al., 1998)

Testes Beta/Alpha- Realizado pelo utilizador final ou pelo cliente para verificar e aceitar o software. (Pombo, 2020) O teste beta ajuda na obtenção de feedback por parte dos utilizadores finais e em tempo real.

O teste alfa, por outro lado, ajuda a simular um ambiente de utilizador em tempo real antes de enviar o software para o teste beta, além de corrigir possíveis erros para que se possa enviar um produto qualificado para o teste beta.

Os testes alfa e beta são inseparáveis um do outro e desempenham um papel importante no ciclo de vida de um teste.

Testes de Usabilidade- O Teste de Usabilidade tem como objetivo avaliar a usabilidade da aplicação, determinando até que ponto a interface do software é fácil e intuitiva de utilizar.

Este tipo de teste possibilita detetar todas as ações dos utilizadores, analisar as suas preferências, ajudando a determinar o que pode ser melhorado na aplicação.

Com a realização destes testes é possível analisar o tempo de resposta do servidor para cada requisição, e até mesmo a satisfação, mesmo que subjetiva, do utilizador, em relação ao que está a ser apresentado. Analisar características de usabilidade numa aplicação em desenvolvimento, ou até mesmo pronta, possibilita que muitos problemas sejam diagnosticados antes do software ser disponibilizado para o cliente. (Renata, 2013)

Isto evita, no mínimo, a insatisfação do cliente numa fase avançada do ciclo de vida do desenvolvimento do software, onde as correções se tornam mais caras e impactantes.

Os testes automatizados são realizados por robôs e consistem no seguinte:

Unit Testing- O teste unitário é uma verificação feita com uma pequena porção de código, uma unidade de um software. Ou seja, é diferente do teste geral, que se dedica a atestar o fluxo do sistema, com as funcionalidades principais.

O objetivo é ajudar a rastrear os bugs e impedir que eles retornem depois das alterações feitas no produto. (Noleto, 2020)

Os principais benefícios de adotar testes unitários para o desenvolvimento de software:

- Menos erros
- Maior satisfação dos clientes
- Mais documentação
- Redução de custos com testes manuais
- Menor perda de tempo com a escrita de testes

System Testing- O teste de sistema concentra-se no comportamento e nas capacidades de todo um sistema ou produto, muitas vezes considerando as tarefas de ponta-a-ponta (início ao fim "edn-to-end") que o sistema pode executar e os comportamentos não funcionais que exibe ao executar essas tarefas. (Sánchez-Gordón et al., 2018)

Os objetivos do teste de sistema incluem:

- Reduzir o risco.
- Verificar se os comportamentos funcionais e não funcionais do sistema estão conforme concebidos e especificados.
- Validar se o sistema está completo e funcionará como esperado.
- Criar confiança na qualidade do sistema no seu conjunto.
- Encontrar defeitos.
- Evitar que os defeitos passem para níveis de teste mais altos ou para produção.

Para determinados sistemas, a verificação da qualidade dos dados pode também ser um objetivo. Tal como acontece com os testes de componentes e testes de integração, nalguns casos, os testes automatizados de regressão ao sistema fornecem a confiança de que as alterações não afetaram as funcionalidades existentes ou as capacidades completas (*end-to-end*). O teste de sistema produz geralmente informações que são usadas pelos *Stakeholders* para tomar decisões de lançamento (*release*). Os testes de sistema também podem satisfazer requisitos ou normas legais ou regulamentares.

Idealmente, o ambiente de teste deve corresponder ao objetivo final ou ao ambiente de produção.

3.3. Testes Automáticos

Automação de testes, por definição, consiste em utilizar um software externo (que não faça parte do software que está a ser testado) para controlar a execução dos testes e a comparação dos resultados encontrados com os resultados esperados. (Schveitzer, 2018)

Basicamente fazendo uso de algum outro software os testes até então manuais e os seus resultados são mapeados e parametrizados para que sejam executados e validados sem que haja a necessidade direta de interferência humana. (Sánchez-Gordón et al., 2018)

A automação de testes é mais do que um mecanismo para executar um conjunto de testes sem interação humana. Envolve um processo de criação do *testware*, incluindo: Software, Documentação, Casos de teste, Ambientes de teste e Dados de teste. (Schveitzer, 2018)

A automação de testes tem como objetivo ajudar a executar muitos casos de testes de forma consistente e repetidamente em diferentes versões do sistema ou ambientes, permitindo um melhor escalonamento dos casos de testes. Também é um objetivo a melhoria da qualidade do software como um todo, pois testes automatizados proveem uma melhor cobertura quando se trata de testes que procuram validar se defeitos antigos e que já foram resolvidos não ressurgiram (testes de regressão). (Lacerda, 2019)

Segundo (Schveitzer, 2018), existem três abordagens possíveis para automação de testes, são elas:

Testes de API- Quando usamos um software ou Framework específico para validar o funcionamento das interfaces públicas para classes, módulos, métodos e bibliotecas do sistema, esta abordagem é focada no *Back-end* do sistema.

Teste de interface de utilizador- Nesta abordagem é testada e validada a interface gráfica do sistema (GUI), para além do funcionamento, é também testada a integridade do *Front-end* do sistema.

- Teste de serviços ou protocolos- É uma abordagem mais específica que procura testar e validar um serviço específico oferecido por um sistema ou um protocolo.

3.3.1. Vantagens da automatização de Testes

Adquirir simplesmente uma ferramenta não garante o sucesso. Cada nova ferramenta introduzida numa organização exigirá esforço para alcançar vantagens reais e duradouras. Existem vantagens e oportunidades potenciais com a utilização de ferramentas de teste, mas também existem riscos. Isto é particularmente verdadeiro em ferramentas de execução de teste automáticos. [Lacerda, 2019]

As vantagens que se tem com a automação de testes são muitas, como por exemplo:

- Mais testes podem ser executados a cada versão nova do software. Cada vez que é lançada uma nova versão do software podemos executar cada vez mais testes, ao incluir novos testes automatizados e fazer mais testes manuais de forma mais desafogada devido à automação.

- Possibilidade de criação de testes que não podem ser feitos manualmente.

- Possibilita o aumento da complexidade dos testes, uma vez que o teste é automatizado fica mais fácil de aumentar a complexidade do mesmo seja por adição de novas regras ou validações, também a execução do teste não terá um custo muito maior de tempo mesmo com essas adições.

- Reduzir os erros humanos na execução dos testes, quando usamos testes automatizados tudo é mapeado e parametrizado o que acaba por eliminar fatores como erros técnicos e ambiguidade a qual uma pessoa está suscetível na hora de criar e executar os testes de forma manual.

- Elevar a eficácia e eficiência dos recursos de teste, os testes passam a ser mais assertivos e otimizados e fornecem mais tempo para os testes manuais.

- Feedback mais rápido e organizado sobre a qualidade do software, isto é, é possível ao término de uma execução de minutos obter um relatório completo e descritivo da qualidade do software.

- Maior confiabilidade do sistema, fica muito claro se o sistema possui consistência através de repetição exaustiva de vários cenários de testes (p. ex., os dados de teste são criados de forma coerente, os testes são executados por uma ferramenta numa determinada ordem e com a mesma frequência, e os testes são consistentemente derivados de requisitos).

- Redução do trabalho manual repetitivo (p. ex., execução de testes de regressão, tarefas de configuração/destruição de ambientes, reintrodução dos mesmos dados de teste e verificação face a normas de codificação), economizando tempo.

- Avaliação mais objetiva (p. ex. medidas estáticas, cobertura).

- Acesso mais fácil à informação sobre os testes (p. ex. estatísticas e gráficos sobre o progresso de testes, taxas de defeitos e desempenho).

3.3.2. Desvantagens e limitações da automação de testes

Como foi descrito anteriormente, a automação de testes traz inúmeras vantagens, porém também temos algumas desvantagens e riscos quando adotamos a automação de testes:

- Custos adicionais envolvidos, que incluem custos desde ferramentas a profissionais e equipas qualificadas.

- Investimento inicial para configurar o *testware*, leva-se um tempo e esforço até que o ambiente para testes automáticos seja montado, além disso, algumas vezes é preciso adquirir máquinas ou recursos específicos, isto ainda é mais válido para quem trabalha com infraestrutura em *cloud*.

- Necessidade de tecnologias adicionais, serão sempre inseridos novos softwares, *frameworks* e até mesmo linguagens de programação novas ao projeto.

- A equipa precisa de ter habilidades de desenvolvimento e automação, não são muitos os profissionais de testes no mercado que possuem este conhecimento.

- Pode existir um desvio dos objetivos de teste, por exemplo, existir uma concentração na automatização de casos de testes às custas da execução dos testes manuais.

- Os testes podem ficar mais complexos, devido a requisitos e regras que os façam encaixar-se na automação.

- Erros adicionais podem ser introduzidos pela automação, existirão testes que iram acusar falha, porém a falha às vezes será no teste automatizado e não no recurso que foi testado.

- Manutenção constante dos testes automatizados e do ambiente de testes automáticos, será sempre necessário tempo e pessoas a fim de corrigir eventuais problemas nos testes e nos ambientes.

- As expectativas sobre a ferramenta podem ser irrealistas (incluindo funcionalidade e facilidade

de uso).

- O tempo e o esforço necessários para alcançar vantagens significativas e contínuas da ferramenta pode ser subestimado (incluindo a necessidade de mudanças no processo de teste e na melhoria contínua da forma como a ferramenta é usada).

- O esforço necessário para manter os produtos de trabalho de teste gerados pela ferramenta pode ser subestimado.

- A ferramenta pode ser utilizada em excesso (considerada como uma substituição da conceção ou execução de testes, ou a utilização de testes automatizados em casos em que os testes manuais seriam melhores).

- O controlo de versão dos produtos de trabalho de teste pode ser negligenciado.

- Os problemas de relações e interoperabilidade entre ferramentas críticas, tais como ferramentas de gestão de requisitos, ferramentas de gestão de configurações, ferramentas de gestão de defeitos e ferramentas de vários fornecedores, podem ser negligenciados.

- O fornecedor da ferramenta pode encerrar a atividade, descontinuar a ferramenta ou vender a ferramenta a outro fornecedor.

- O fornecedor pode dar uma resposta insatisfatória para suporte, atualizações e correções de defeitos.

- Um projeto de código aberto pode ser suspenso.

- Uma nova plataforma ou tecnologia pode não ser suportada pela ferramenta.

- Pode não existir uma propriedade clara da ferramenta (p. ex., para mentoria, atualizações, etc.).

A automação de testes possui algumas limitações, segundo (Schveitzer, 2018), que devem ser levadas em conta, são elas:

- Nem todos os testes manuais podem ser automatizados: Alguns testes não podem ser automatizados, um exemplo são testes de usabilidade ou alguns testes que possuem dependência de outros softwares e informações externas.

- A automação só pode verificar os resultados interpretáveis pela máquina: Ou seja, testes cujos resultados não sejam resultados exatos não podem ser automatizados.

- Não é um substituto para testes exploratórios: Como a cada versão de um software a sua complexidade tende a aumentar os testes automatizados por serem parametrizados e mapeados não dão conta de testar todos os casos possíveis, incluindo nesse contexto os novos casos de testes, sendo assim, o teste automatizado não substitui os testes exploratórios.

3.3.3. O papel das ferramentas de automação

No campo das ferramentas, o primeiro passo é considerar qual é a ferramenta que responde às nossas necessidades. É importante ter em conta as nossas necessidades e observar os tipos de validação, arquitetura e soluções que nos permitem construir, gerir e relatar o projeto de automação.

O processo de selecionar uma ferramenta de automação para web ou mobile não é o mesmo que deveremos ter em conta. Por isso, nesse caso, há que fazer uma abordagem mais específica no que toca ao suporte técnico e às ferramentas que se adequam ao tipo de sistema escolhido. (Amaral, 2021)

Naturalmente, no contexto da necessidade de se recorrer à automação de testes, as ferramentas de automação ocupam um papel de destaque, pois são as grandes responsáveis pela execução deste tipo de tarefa.

Em seguida são apresentadas algumas das ferramentas de automação mais utilizadas pelo mercado:

Selenium- é a ferramenta mais popular e utilizada no mercado. Esta ferramenta *open source* é usada principalmente para aplicações web. (PrimeControl, 2022)

A estrutura de testes do Selenium atua em vários browsers e plataformas como Linux, Mac, Windows, Firefox, Chrome, Internet Explorer, assim como em *Headless Browsers*.

O Selenium IDE, um *add-on* de browser, permite gravar e reproduzir, o que significa que o *tester* pode usar estas funcionalidades até mesmo se estiver a aprender a utilizar o Selenium IDE. O Selenium WebDriver ajuda a criar scripts de automação de testes mais complexos e avançados.

Além disso, os *testers* podem escrever em várias linguagens de programação como Java, Perl, JavaScript, PHP, Python, C#, Ruby e Groovy.

TestComplete- Desenvolvido pela SmartBear Software, o TestComplete é uma plataforma comercial que pode automatizar testes do desktop, mobile e aplicações web. Permite também a utilização de diversas linguagens, como JavaScript, VBScript e Python, além também de ter as funções de teste orientado por dados (DDT), teste por palavras-chaves, teste de regressão e teste distribuído. (PrimeControl, 2022)

O TestComplete possibilita a criação de testes automatizados para aplicações iOS, Web Microsoft Windows e Android. A ferramenta possui as seguintes funções:

- Testes de GUI
- Suporta JavaScript, Python, VBScript, JScript, DelphiScript, C++Script e C#Script
- Visualizador de testes
- Testes com scripts
- Gravação e reprodução de testes

Telerik Test Studio- é uma ferramenta de automação abrangente. O TestStudio pode ser usado manualmente e também oferece teste funcional de UI, teste de performance, teste exploratório, teste mobile, teste de carga e teste no Visual Studio. São compatíveis no Test Studio aplicações de automação como Angular, ASP.NET, HTML5, JavaScript, AJAX, WPF, Silverlight, MVC, Ruby e iOS, PHP e Android. (PrimeControl, 2022) Também suporta linguagens de script como VB.Net e C#.

Outras funções do Telerik Test Studio são:

- Gravar e reproduzir

- Integração com Visual Basic Studio 2010 e 2012
- Testes cross-browser
- Integração com ferramentas de depuração
- Testes manuais

Unified Functional Testing (UFT) - uma das ferramentas de automação de testes cross-platform de ponta, que proporciona testes funcionais automatizados e testes de regressão. Faz uso de VBscripts para especificar procedimentos de testes. (PrimeControl, 2022)

Algumas das características exclusivas do UFT são:

- Automatiza testes em Web, PowerBuilder, Desktop, ActiveX, SAP, Delphi, Net, Flex, Java, Oracle, Siebel, Mobile, PeopleSoft, Stingray, Visual Basic…
- Usa VBScript como linguagem de script
- Oferece uso simplificado para GUI
- Altamente integrado com a HP ALM (ferramenta de gestão de testes) e HP LoadRunner (ferramenta de testes de performance)
- Integração com Mercury Business Process Testing e Mercury Quality Center
- *Smart Object Recognition*
- Mecanismo de tratamento de erros
- Criação de parâmetros para objetos, pontos de verificação e tabelas de dados
- Documentação automatizada

Cucumber- Programado em Ruby, é uma ferramenta de testes *open source* desenvolvido com o conceito de desenvolvimento orientado por comportamento (BDD). (PrimeControl, 2022) Este conceito é usado para escrever testes de aceitação para aplicações web. Anteriormente, era restrito apenas ao Ruby, mas atualmente suporta outras linguagens como Java, .NET, Scala, Groovy, etc. As principais funcionalidades são:

- Usado para testes automáticos de aceitação
- Fornece documentos únicos, que inclui especificações e testes
- Também suporta múltiplos sistemas operativos.
- O código do Cucumber pode ser executado em diferentes *frameworks* como Selenium, Ruby, etc.

3.4. Plataformas *Low Code*

Uma plataforma *low-code* (LCP) é um conjunto de ferramentas para programadores e não programadores. Estas permitem a geração e entrega rápida de aplicações com o mínimo esforço de escrita através de uma linguagem de programação e requer o menor esforço possível para a instalação e configuração de ambientes, treino e implementação. (Bock et al. 2021)

Com um número de empresas em rápido crescimento, o uso de soluções *low-code* pode ser um

passo à frente na criação de aplicações de negócios essenciais, isto é, até agora, o método mais rápido e provavelmente também o mais barato de desenvolvimento de software.

As Plataformas de Desenvolvimento *Low-Code* (LCDP) são baseadas numa interface gráfica do utilizador ao projetar a aplicação como oposição às técnicas de programação codificadas.

A característica do LCDP concentra-se no desenvolvimento do seguinte:

• Bases de dados

• Processos de negócios

• Interface do utilizador (aplicações baseadas na web).

A técnica *Low-Code Programming* é derivada da quarta geração de programação (4GL), ideologia que junta conceitos de Desenvolvimento Rápido de Aplicações (RAD). A Programação *LowCode* permite que o programador gaste menos tempo a pensar na sintaxe do código e colocar mais ênfase em projetar a estética e a funcionalidade da aplicação, reduzindo assim o tempo gasto na solução de problemas e implementação. A ideia de *Low-code*, que foi introduzido em 2011, é considerado um romance e conceito de ponta no campo da programação. (Bock et al. 2021)

As abordagens que criaram a Programação *Low-Code* são do seguinte modo:

• Abordagem de desenvolvimento de software orientada a modelos

• Desenvolvimento de Aplicação Rápida

• Geração automática de código

• Programação visual

Estima-se que existam cerca de 275.000 vagas não preenchidas no sector das TI na UE e, até 2025, esta escassez certamente terá passado de 1 milhão. De acordo com a empresa de pesquisa Gartner, até 2021, a demanda por sistemas de informação aumentou cinco vezes mais rápido do que a capacidade de os fornecer pelos departamentos de TI, porque o número de funcionários não está a crescer a ritmo suficiente. Esta situação afeta a pontualidade da entrega do software. Além disso, as soluções de TI modernas não são baseadas em sistemas. As empresas usam cada vez mais soluções personalizadas.

Plataformas *low-code* permitem que sejam criadas aplicações sem codificação, enquanto aumentam a velocidade de introdução a mudanças.

Alguns exemplos de fornecedores globais de plataformas *low-code* são: Outsystems, Salesforce, Microsoft PowerApps, Mendix, Google App Maker, TrackVia e Appian.

No mercado português, cada vez mais empresas oferecem plataformas *low-code*. Essas soluções são construídas principalmente em tecnologias europeias ou americanas existentes, adicionando elementos e módulos personalizados.

O exemplo mais flagrante neste caso é a Outsystems, que é uma plataforma *low code* que fornece ferramentas para as empresas desenvolverem, implementarem e gerirem aplicações, como aplicações móveis e empresariais da web, que são executados em *cloud*, local ou em ambientes híbridos. (Madrinha et al., 2018) Esta foi fundada em 2001 em Lisboa, Portugal, e é um membro do consórcio da qualidade do software de TI (CISQ). (Marvin, 2017)

Em 2014, a Outsystems lançou uma versão gratuita da plataforma que fornece aos *developers* ambientes de *cloud* pessoais para criar e implementar aplicações web e móveis sem cobrança. A versão atual é 11, para ambas as versões, pagas e não pagas. (Atlanta, 2014)

As plataformas *low-code* para automatizar processos de negócios em fabricação é um desafio difícil e exigente que permitem a solução de muitos problemas de pesquisa e tecnológicos.

Desde a redução de custos associados com mão-de-obra e manutenção para garantir um maior rendimento líquido dos produtos através de melhorias de qualidade, a automação é a chave para administrar uma fábrica lucrativa nos dias de hoje. (Waszkowski, 2019)

Infelizmente, para muitos gestores, a automação não é uma questão de primeiro plano. Para muitas empresas, isso exige entrar num mundo inteiramente novo de desenvolvimento, de dados e enfrentar desafios completamente novos, como:

• Adquirir e pagar por habilidades de desenvolvimento especializado,

• Compreender a experiência do utilizador para criar formulários,

• Manutenção manual de aplicações e automação de processos para garantir o mínimo de tempo de inatividade.

Esses desafios técnicos e de desenvolvimento foram a razão por que as empresas não decidiram automatizar processos. Isto quando surgiu a nova opção que pode simplificar todos esses desafios e dar mesmo aos mais “limitados” experiência de desenvolvimento, a chance de criar aplicações e sistemas. (Waszkowski, 2019)

No entanto, não está claro o que distingue as plataformas *low code* das instalações de desenvolvimento de software existentes, como ambientes de desenvolvimento integrados (*IDEs*) clássicos e ferramentas para desenvolvimento orientado a modelos (MDD). Em conjunto, a tendência *low-code* apresenta-se como uma contribuição para os objetivos centrais da pesquisa de sistemas de informação de negócios, mas necessita urgentemente de esclarecimento.

Enquanto isso, organizações com e sem fins lucrativos enfrentam a questão de saber se, e em que condições, os investimentos em projetos de desenvolvimento *low-code* são justificados. A necessidade de tomar essas decisões é mais um motivo para que pesquisas em sistemas de informação empresarial analisem o assunto e contribuam para o esclarecimento do termo. (Zolotas et al. 2018)

Na falta de descrições informativas e consistentes de LCPs, a única maneira de obter um conceito ou mesmo uma ideia de LCPs é uma análise das plataformas reais oferecidas sob este rótulo no mercado atual. Se essas plataformas tiverem características comuns, pode ser possível derivar indutivamente delas uma possível conceituação de desenvolvimento *low-code*.

Um facto impressionante sobre quase todos os produtos considerados *low-code* é que eles já existiam, de uma forma ou de outra, antes que o rótulo “*low-code*” fosse inventado. Com exceção de uma plataforma, as soluções agora comercializadas como LCPs têm sido os produtos únicos ou primários que as empresas têm como ofertas por anos, se não décadas. Por exemplo, alguns

sistemas já foram dados como soluções para “desenvolvimento rápido de aplicações”, “plataforma como serviço” (PaaS) ou como uma “plataforma de aplicações orientadas por modelos”. (Robert et al. 2016) Outra plataforma, que tem um histórico como ferramenta para 'gestão de processos de negócios' (BPM) em geral e como ferramenta para BPM 'móvel' e BPM 'centrado nos humanos' em particular. A conclusão é que as soluções *low-code* atuais raramente são produtos novos, na maioria das vezes, têm uma longa história no mercado e simplesmente foram renomeados. (Sanchis et al. 2019)

Outra descoberta significativa é que as plataformas disponíveis no setor *low-code* variam drasticamente em várias dimensões. Embora certas características técnicas sejam comuns a muitos LCPs, os ambientes diferem substancialmente em scope funcional, propósito principal, gama de tecnologias envolvidas, amplitude de aplicabilidade, meios de design e especificação e outros aspetos. Por exemplo, as quatro categorias aproximadas de LCPs que podemos distinguir incluem:

(1) Plataformas simples de gestão de dados,

(2) Sistemas clássicos de gestão de fluxo de trabalho (WfMS),

(3) Interface gráfica de utilizador (GUI)

(4) Plataformas multiuso complexas para configuração, integração e desenvolvimento de aplicativos de negócios

O resultado é que as plataformas oferecidas sob o rótulo “*low-code*” são extremamente heterogéneas e não constituem uma classe bem definida de ambientes tecnológicos.

3.4.1. Característica de uma Plataforma Low Code

Características comuns

Os recursos mais comuns dos LCPs enquadram-se na perspetiva estática. Cada produto considerado possui um componente para a definição de estruturas de dados. Esse componente é quase sempre oferecido na forma de uma ferramenta de modelação conceitual, implementando uma linguagem clássica de modelação de dados (ou seja, uma variante do Modelo Entidade-Relação) ou uma linguagem proprietária simplificada. Às vezes, as estruturas de dados só podem ser definidas em caixas de diálogo ou listas baseadas na interface do utilizador. Um recurso comum relacionado é a capacidade de aceder a fontes de dados externas usando uma variedade de interfaces de programação de aplicações (APIs). Por exemplo, todas as plataformas permitem o uso de APIs padrão como JDBC, bem como vários conectores para outros tipos de arquivos e sistemas. Por regra, as plataformas são projetadas para que os dados possam ser armazenados num sistema de bases de dados interno ou em sistemas externos existentes. (Chang et al. 2017)

Outro recurso fornecido por todas as plataformas *low-code* estudadas é um designer de GUI. Sem exceção, as plataformas incorporam um componente para desenvolver interfaces gráficas de utilizador e integrá-las com outros artefactos de implementação. Todos os designers de GUI revistos trazem paletas de *widgets* pré-definidos, embora o seu âmbito varie muito. Definir o acoplamento de GUIs e estruturas de dados é bastante conveniente na maioria dos ambientes, pois não é necessário

implementar o padrão *model-view-controller* (MVC) manualmente. (Chang et al. 2017) Além disso, a maioria dos sistemas oferece suporte distinto na adaptação das GUIs a diferentes ambientes de destino (por exemplo, navegadores de desktop, tablets e smartphones).

Além disso, todos os sistemas oferecem especificações funcionais básicas. A abordagem mais comum aqui são linguagens de expressão simples para regras de decisão e formas baseadas em diálogo para especificar as condições de fluxo do programa. Da mesma forma, cada solução fornece uma biblioteca de operações padrão genéricas, como funções matemáticas. Além disso, todas as soluções permitem, de várias maneiras, invocar e integrar funções externas através de APIs. Por exemplo, quase todos os sistemas permitem o uso de abordagens padrão, como serviços Web e serviços RESTful (*representational state transfer*); e muitos sistemas possibilitam o uso de uma longa lista de APIs de provedores individuais, como APIs da Google e outras APIs de "social media". (Kłosowski et al. 2016)

Passando para uma área diferente, quase todas as soluções consideradas oferecem suporte avançado na implementação de aplicações, embora isso assuma formas bastante diferentes. Por exemplo, alguns sistemas exigem a instalação do ambiente da plataforma *low-code* num servidor web, para que aplicações individuais possam ser implementadas. Outros sistemas, pelo contrário, permitem implementar as soluções desenvolvidas como aplicações autocontidas em vários dispositivos e máquinas. Por fim, outra característica comum é que quase todas as soluções consideradas possuem um componente para definir funções e direitos do utilizador. As funções e o sistema do utilizador geralmente estão contidos na arquitetura de controlo da plataforma, que será implementada juntamente com a aplicação personalizada. (Robert et al. 2016)

Características ocasionais

Um recurso ocasional é a disponibilidade de um componente de modelação de fluxo de trabalho e um mecanismo de fluxo de trabalho. Vários dos LCPs maiores e mais complexos incorporam um WfMS na sua arquitetura; outros sistemas agora rotulados como LCPs, conforme indicado anteriormente, são realmente WfMS clássicos. As plataformas usam uma linguagem de modelação conceitual como o Business *Process Model and Notation* (BPMN), ou uma estrutura de representação proprietária. Noutros sistemas, a perspetiva é mais genérica e técnica. Essas plataformas fornecem principalmente componentes para definir a interação e a transição entre os formulários de interface do utilizador definidos pelo próprio utilizador. Algumas plataformas contam com componentes proprietários de modelação de processos para essa finalidade, outras em menus básicos e scripts de captura de eventos.

Outro recurso ocasional é a disponibilidade de componentes de codificação avançados ou tradicionais. Alguns sistemas envolvem um ou vários componentes explícitos onde as especificações procedimentais podem ser feitas usando linguagens de programação tradicionais e tecnologias relacionadas. Mais frequentemente, os sistemas usam Java e JavaScript. De facto, num nível mais ou

menos oculto da arquitetura, quase todas as plataformas *low-code* permitem o recurso ao código de programação tradicional. As plataformas dos principais fornecedores oferecem uma variedade de unidades de aplicações configuráveis e semelhantes a blocos de construção. Por exemplo, essas unidades fornecem, em âmbito limitado de funcionalidades pré-implementadas para as áreas de *business intelligence* (BI), inteligência artificial (AI) e RPA. (Kłosowski et al. 2016)

Características raras

Apenas uma plataforma de grande escala fornece uma biblioteca considerável de modelos de dados de referência específicos de domínio, abordando conceitos comuns de negócios e comunicação, como “cliente”, “endereço” e “e-mail”. (Jasiulewicz et al. 2015) Alguns sistemas oferecem pequenos conjuntos de estruturas de referência simples, e a maioria dos LCPs existentes não traz nenhum modelo de referência de dados. Quando se trata de funções de referência específicas de domínio e artefactos reutilizáveis de natureza dinâmica, ainda menos artefactos são encontrados. Uma plataforma de grande escala possibilita a reutilização de várias funções prontas de várias aplicações de negócios do mesmo fornecedor. A maioria dos outros sistemas oferece catálogos de funções reutilizáveis ou exemplos de processos predefinidos, mas estes são na sua grande maioria genéricos por natureza, ou muito limitados em âmbito e profundidade.

As plataformas *low-code* integram vários componentes clássicos de desenvolvimento num ambiente. A maneira em que os LCPs diferem das infraestruturas clássicas de desenvolvimento de software é que eles incorporam todas ou a maioria das ferramentas e componentes necessários para uma classe limitada de projetos de desenvolvimento de software num único ambiente. Ao usar uma infraestrutura clássica de desenvolvimento de software, é preciso lidar com uma variedade considerável de ferramentas separadas, como IDEs, ferramentas de modelação, sistemas de gestão de base de dados (DBMS), estruturas de mapeamento objeto-relacional (O-R), editores de GUI, assistentes de implementação e compilação, e assim por diante. (Amoroso, 1998) Nos LCPs, essas ferramentas são integradas num único sistema, de modo que exista menos necessidade de alternar entre diferentes sistemas e, mais importante, menos necessidade de manter a consistência e integrar os artefactos de implementação produzidos pelas tecnologias distintas. (Jasiulewicz et al. 2015)

A reutilização é abordada num nível de arquitetura genérico, não num nível específico de domínio. As plataformas *low-code* raramente oferecem artefactos reutilizáveis num nível específico de domínio. Em vez disso, o que é reutilizado por essas plataformas são estruturas arquitetónicas bastante genéricas e muitas vezes implícitas para determinadas classes de sistemas de aplicações. Isso envolve implementações de referência para áreas como design de GUI, mapeamentos O-R, operações MVC, acesso a fontes de dados externas e outros serviços. Dentro dessas estruturas, então, soluções individuais podem ser configuradas e desenvolvidas.

Os ganhos de produtividade decorrem principalmente da redução dos esforços das tarefas rotineiras. Os LCPs produzem ganhos de produtividade principalmente pela redução dos esforços de

tarefas rotineiras em projetos de desenvolvimento de software de complexidade baixa a moderada. (Waszkowski, 2017) Isto aplica-se de várias maneiras. Uma delas é que, através do fornecimento de um ambiente integrado e pré-definido, há menos esforço para sincronizar os artefactos produzidos por componentes de desenvolvimento previamente separados. A produtividade também é promovida pelas implementações, de referência mencionadas acima, para tarefas genéricas como design de GUI, mapeamento O-R, implementação de MVC e implementação em diferentes ambientes. (Hall, 1998)

3.4.2. Risco de efeitos de bloqueio

Dependência com o fornecedor

Quanto mais uma empresa investe no uso de uma plataforma específica *low-code*, mais difícil é para ela mudar para uma plataforma diferente. Isso aumenta a dependência do negócio num fornecedor individual. Na maioria dos casos, as representações geradas num LCP não podem ser utilizadas noutro diferente. (Hall, 1998) Isto representa uma séria ameaça à proteção do investimento e pode levar a becos sem saída quando um fornecedor deixa de oferecer suporte a uma plataforma (como é o caso do AppMaker, que o Google abandonou recentemente). (Ribeiro, 2022)

Flexibilidade limitada

Com os códigos tradicionais, os desenvolvedores podem personalizar o software em qualquer grau que desejarem, enquanto o *low-code* depende da funcionalidade integrada das plataformas e dos plugins. Se uma determinada funcionalidade não for fornecida, a equipa de desenvolvimento poderá ter dificuldade em cumprir os requisitos relacionados. (Ribeiro, 2022)

Segurança e risco

Quando se trata de segurança, os projetos *low-code* dependem muito dos seus provedores de plataforma para resolver quaisquer falhas de segurança que possam ser descobertas devido à falta de controlo sobre o código-fonte. Se esses provedores de plataforma forem invadidos, poderá não haver atualizações de segurança e as empresas não poderão corrigir os problemas por conta própria. Além disso, se as empresas dependerem de provedores de plataforma *low-code* como *hosts*, os seus dados e sistemas podem ficar expostos e vulneráveis a violações de segurança. (Ribeiro, 2022)

Capítulo 4

4. Trabalho Realizado

Neste capítulo é abordado todo o trabalho realizado no decorrer do estágio, e os aspetos relacionados com os projetos nos quais houve uma integração, nomeadamente as funcionalidades testadas e ferramentas/linguagens que foram utilizadas e, por fim, uma apresentação dos resultados obtidos.

Nesta primeira fase, após todo o estudo preliminar realizado é importante reter que num projeto de software a obtenção de produtos de software com qualidade aprimorada é mais confiável, constitui uma atividade em que a demanda cresce significativamente, pois os utilizadores exigem cada vez mais qualidade, eficiência e eficácia.

Posto isto, na próxima secção deste capítulo está descrita toda a experiência vivenciada no primeiro cliente.

4.1. Cliente 1

Após a entrada no cliente foi feita uma reunião para breve contextualização do que era necessário para a integração nas equipas e nos projetos. Desta pudemos extrair os seguintes dados:

- A integração do *tester* de automação iria ser feita em dois projetos distintos da casa, uma vez que existia falta de profissionais especializados para trabalhar com as ferramentas necessárias, com possibilidade de se poder estender a um terceiro projeto.
- Os dois projetos em questão eram projetos que já tinham algum tempo considerável de desenvolvimento, portanto 1 e 2 anos, sendo que o projeto que já tinha 2 anos de desenvolvimento se encontrava em produção.
- Ambos os projetos estavam a ser desenvolvidos em Outsystems.
- Um dos projetos teria apenas componente web e o outro componente web e mobile.
- A metodologia adotada pelas duas equipas de ambos os projetos, era uma metodologia Agile, nomeadamente SCRUM.
- As tecnologias/ferramentas a utilizar para a automação de testes seriam o TestPath com a integração do JIRA.

Dadas estas informações, o primeiro passo, uma vez que os projetos já contavam com um tempo considerável de existência, seria inteirar do âmbito dos projetos, o que estava desenvolvido, o que existia a nível de testes, uma vez que estes existiam devido a já existir um *tester* funcional manual na equipa, e aquilo que poderia ser automatizado ou não.

E é precisamente nesta primeira etapa que surgiu a primeira limitação, falta de qualquer tipo de documentação. Sabemos que neste meio e só de alguns anos para cá existe uma maior preocupação

com o facto de ter tudo documentado, principalmente quando estamos a trabalhar utilizando uma metodologia ágil, porém dentro de todas as plataformas utilizadas pelo cliente (Confluence e JIRA) não existia qualquer tipo de descrição, ficheiro ou página relacionadas com nenhum dos projetos em questão.

Pelo que, se optou pela segunda hipótese, que foi ir procurar o analista funcional de cada projeto. O que aconteceu foi que nenhum dos projetos possuía um analista funcional ou alguém que desempenhasse uma função semelhante. Sabemos que em meio empresarial é muito difícil seguir uma metodologia à risca, mas no caso da metodologia SCRUM não possuir um analista funcional acabou por ser um pouco limitante.

Como última alternativa, e uma vez que a automação iria ser de testes e iria existir uma certa parceria com o *tester* funcional manual de cada projeto, foi procurada a informação necessária junto destes.

Após o entendimento do âmbito de cada projeto, seria importante perceber que tipo de trabalho já tinha sido desenvolvido a nível dos testes e que tipos de testes eram realizados. Pelo que se chegou à conclusão de que apenas eram realizados testes funcionais de forma manual e por vezes, não em todos os sprints, existia uma equipa que realizaria testes de *cybersegurança*.

Seguidamente, tentou-se perceber dentro dos testes que já estavam então desenhados, quais seriam possíveis de automatizar, tendo em mente que uma vez já começados os projetos e já existindo bastante desenvolvimento o mais fácil seria começar pela automatização dos testes de regressão de cada projeto. Pois estes são os testes fundamentais de serem executados em todos os Sprints uma vez que contêm as funcionalidades principais que já não são passíveis de sofrer alterações e que não podem ser afetadas pelos novos desenvolvimentos entregues em cada *release*, e na automatização, convém sempre automatizar testes onde consigamos perceber que não irão existir grandes alterações, pois aí perde-se tempo e se forem alterações drásticas é apenas desperdício de trabalho.

Para descrever a análise que foi feita aos desenhos e aos casos de teste já existentes vamos separar os projetos. Chamemos-lhes Projeto 1 e Projeto 2. Importante salientar que todo este trabalho estava alojado na plataforma JIRA, mas como referido algures nos parágrafos anteriores, a falta de documentação era uma realidade e nos testes não foi exceção.

Em relação ao Projeto 1, a nível de *test cases* dedicados a este projeto, encontrámos uma pasta no JIRA com cerca de quatro mil, porém, após a análise e leitura destes se, se aproveitaram cem, foi muito. Em primeiro lugar, seja em que área for, a organização é chave, ou seja, os casos de teste não são todos iguais, não estão todos dentro da mesma categoria, os testes para mobile não são o mesmo que testes para a web, dentro dos testes em mobile, um teste em Android não é um teste IOS, e por aí adiante.

Antes do processo da automação propriamente dito, houve muito trabalho para ser feito, pois nós só retiramos benefícios de algo, seja processo ou tecnologia, quando tudo está devidamente equilibrado e a “caminhar” no mesmo sentido. O objetivo a partir daqui seria organizar toda a entropia envolvida com os casos de teste e só aí depois começar a automatizar. Contudo, estamos a falar de

um projeto já em produção e com *releases* de 3 em 3 semanas, pelo que as *guidelines* eram extremamente apertadas e era necessário começar a automatizar o quanto antes. E se tivesse ficado por esta justificação, era algo considerado normal neste meio, porém a justificação apresentada foi a de se ter de acelerar o processo de automação para facilitar o trabalho do *tester* funcional porque estava com muito trabalho.

Como já referido anteriormente, a automação de testes traz inúmeros benefícios dentro de um projeto de engenharia de software, porém nenhum deles é facilitar o trabalho a um dos elementos da equipa, isto mostra algum descrédito pelo trabalho do profissional que está a automatizar e pode levar à desmotivação do mesmo, não falando que demonstra algum desconhecimento do trabalho pretendido em si e a vontade de ter testes automáticos pelos motivos errados.

Posto isto, ficou acordado que seriam automatizados menos testes por Sprint, mas que se iria fazer um *refactoring* aos casos de teste, tarefa esta dividida pelos dois elementos responsáveis pela qualidade, no projeto. Em suma, a maioria dos casos de teste apenas apresentava o título não possuindo sequer *step definitions*, pelo que ficou acordado que grande parte iria para uma pasta denominada de "*Deprecated*" e o processo de desenho e plano de testes iria começar de novo.

Assim sendo, para a automação foi seguido o plano que seria a ideia inicial, desenhar os testes de regressão e passar à automatização.

Em relação ao Projeto 2, o cenário encontrado foi idêntico, porém, houve uma maior flexibilidade por parte da equipa, também por o projeto estar longe de ir para produção, e pôde-se fazer o procedimento da conceção dos testes de forma correta, acompanhando também o trabalho que era desenvolvido pelos restantes elementos da equipa.

Sendo que a estratégia de começar por automatizar os testes de regressão também foi a adotada neste projeto.

Em ambos os projetos, foi precisa muita atenção, no sentido em que nem todos os testes poderiam ser automatizados, em ambas as aplicações existiam funcionalidades que para funcionarem e serem testadas, em algum momento no fluxo do teste eram necessários dispositivos externos que exigiam manuseamento do próprio *tester* e logicamente este tinha de ser presencial. Também existiam algumas limitações em questões de acessos, pelo que também não foi possível a automatização de funcionalidades que exigissem confirmações de dados num sistema de *backoffice* que era externo. Todos os testes com estas características e testes mobile em IOS no caso do Projeto 1, ficou definido que continuariam a ser testados manualmente. Todos os restantes, até impedimento em contrário seriam automatizados.

Para fazer a descrição daquele que foi o processo de desenvolvimento dos testes automáticos, é importante que conheçamos bem a ferramenta que foi utilizada para este efeito, o TestPath. Na seguinte subsecção, irá ser descrita o que é esta ferramenta, para que serve, as suas principais funcionalidades, uma breve *overview* pela sua interface e as vantagens da sua utilização.

4.1.1. TestPath

Projeto desenvolvido pela Atos que consiste no desenvolvimento de uma Framework para automação dos testes, TestPath (TP), que normaliza a implementação de testes automáticos e facilita o processo de testes dos mesmos.

A nível de capacidades de Automação de teste o TP é uma ferramenta feita essencialmente para *Testers* Funcionais. É o que podemos considerar uma tecnologia agnóstica no sentido em que facilmente conseguimos funcionar/trabalhar com ele sem conhecer alguns detalhes básicos, é, portanto, uma ferramenta extremamente intuitiva.

É também fácil de escalonamento e o processo de padronização permitem uma curva de aprendizagem rápida, ao ponto de não ser necessário uma formação muito extensiva e exaustiva e no que toca a nível de documentação, a própria aplicação possuí um manual de utilizador que permite auxiliar em qualquer dúvida que o utilizador tenha.

É também uma ferramenta colaborativa, no sentido em que não funciona como um simples IDE, permite também a integração com qualquer ambiente de desenvolvimento e com várias outras tecnologias, como irá ser descrito posteriormente.

Posto isto é interessante ver onde podemos aproveitar a utilização do TP e que tipos de serviços de teste são efetuados pela Atos com este:

Atualizações da plataforma

- Automatização do *backlog* atual dos casos de teste
- Criação da primeira versão de teste de regressão automatizada

Migração para Ambientes Cloud

- Automatização do *backlog* atual dos casos de teste
- Criação da primeira versão de teste de regressão automatizada
- Garantia de teste funcional do *Cloud Migration*

Automação de *Backlog* de Casos de Teste

- Automação de casos de teste
- Criação da primeira versão de teste de regressão automatizada

Ciclo de Vida de Desenvolvimento de Sprints Contínuos

- Nova automação de casos de teste
- Atualização automática da sua versão de teste de regressão automatizada

Teste de regressão

- Automatização do seu *backlog* de teste de regressão

4.1.1.1. ADERÊNCIA DE TECNOLOGIAS

No que diz respeito a tecnologias com que o TP pode colaborar/funcionar, podemos dividi-las nos seguintes grupos:

- Cognitive Systems- Dando como exemplo de utilização em *chatbots*, inteligência artificial e etc…
- Corporative Systems- Dando como exemplo alguns dos mais utilizados como Outsystems, Java ou SAP.
- Web Services- Essencialmente SOAP e REST
- Databases- Todas as que funcionem com modelos relacionais como Oracle, SQL server ou MySql e não relacionais como por exemplo MongoDB
- Web Browsers- É possível testar em todos, pelo menos os mais conhecidos como o Chrome, Edge, Safari ou Mozilla
- Mobile Devices- Neste momento apenas é possível realizar testes no SO mobile Android, sendo que a possibilidade de testes em SO IOS está a ser desenvolvida
- Command Line- Pode ser manuseado através de várias linhas de comandos que utilizam protocolos como SSH ou FTP.

4.1.1.2. INTEGRAÇÕES

O TP permite também várias integrações tecnológicas (Figura 2) divididas pelos 3 seguintes grupos:

Ferramentas de gestão de testes- O objetivo destas ferramentas de gestão de testes do cliente é o alojamento dos resultados dos testes realizados. Dando como exemplo o JIRA ou o Azure DevOps.

Ferramentas CI/CD (práticas combinadas de integração contínua e entrega contínua) - Ferramentas que irão permitir a criação de processos automáticos para executar os testes, se bem que só por si o TP consegue fazê-lo. Dando como exemplo o Jenkins.

Ferramentas Engine- Nestas ferramentas podemos dar exemplos como o Selenium ou o SoapUI que é necessário para os testes efetuados em webservices (apesar do TP já o permitir nativamente).

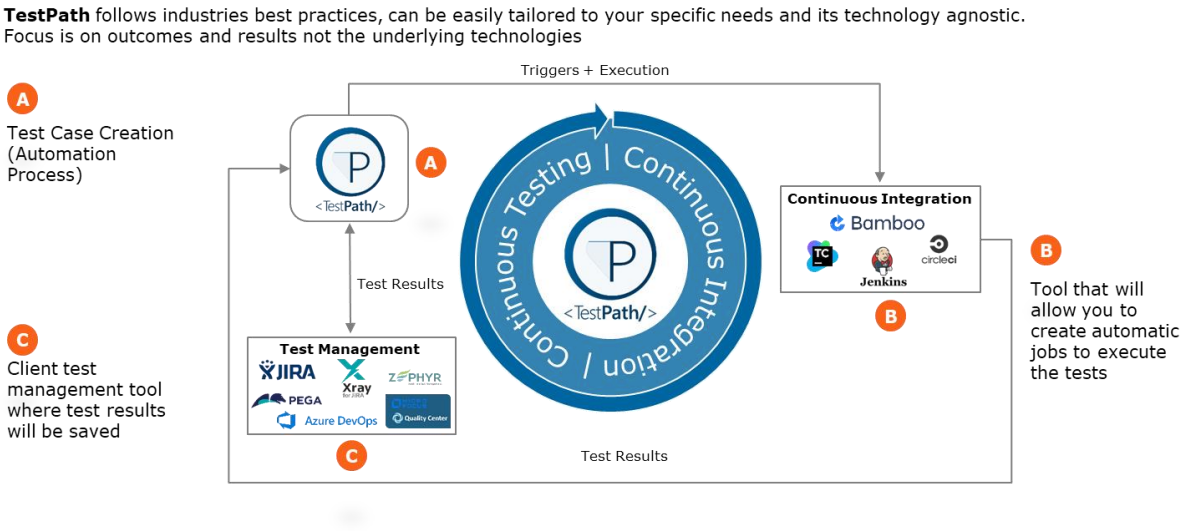


Figura 2- Diagrama da integração das várias tecnologias com o TestPath no ciclo de teste

4.1.1.3. REQUISITOS DE HARDWARE/SOFTWARE

TestPath foi desenvolvido para ser fácil de instalar e testar, mas dependendo da tecnologia em teste, mais configurações podem ser necessárias.

O primeiro passo é fazer login no TestPath, clicar no seu nome de utilizador e baixar o Runner e em seguida executar o ficheiro Runner.App.exe.

Após a instalação, as tecnologias padrão (Web, APP, Android, IOS, Database) estarão disponíveis.

Para teste de Web Service, pode ser necessário instalar o SoapUI e para testes em Android, o único requisito é ter um telefone Android conectado ao PC.

A nível de Software e Hardware encontram-se mais alguns requisitos na Tabela 2:

Tabela 2- Requisitos de Software e Hardware para instalação do TestPath

Requisitos da máquina do servidor	Requisitos da máquina do cliente
CPU 8 Core	Pode ser usada uma máquina de teste comum, não há requisitos de hardware específicos
Memória RAM de 16 GB+	
Disco rígido de 500 GB+ partição única de preferência	
Instalação de software de 50 Gb	
Filesystem – Histórico de evidências e logs de execução (~10 a 20mb cada log/relatório)	
SO: Windows Server 2012 ou superior Windows 10 Professional Edition	
SQL Server 2012 ou superior	
Se a máquina SQL não for igual ao SO, na máquina OS é necessário o SQL Management Studio	
Perfil do proprietário da base de dados	
Excel 2007 - .xlsx ou superior	Instalação rápida e direta do agente do Windows
Microsoft IIS ativado	
Portas usadas: 80	
Direitos de administrador local	
Área de trabalho remota com direitos de administrador	
NodeJS instalado ou acesso ao site para baixá-lo	
.Net framework 4.6.1 (versão mínima) instalado	
Pacote de recursos de média instalado	
Visual C++ Redistribuível 2015-2019	

4.1.1.4. OVERVIEW DA APLICAÇÃO

Para começar a trabalhar com o TestPath deparamo-nos com o login, cuja interface é semelhante à da Figura 3. Para realizar o login é necessário ter credenciais, escrever o nome de utilizador, a password e depois clicar em “Entrar”.

Figura 3- Página de inserção de credenciais

Após o processo de “Login”, o passo seguinte é escolher o que se pretende fazer dentro das seguintes opções disponíveis, representadas na Figura 4, tendo em conta que as duas primeiras opções apenas são disponibilizadas a utilizadores que tenham permissões de administração.

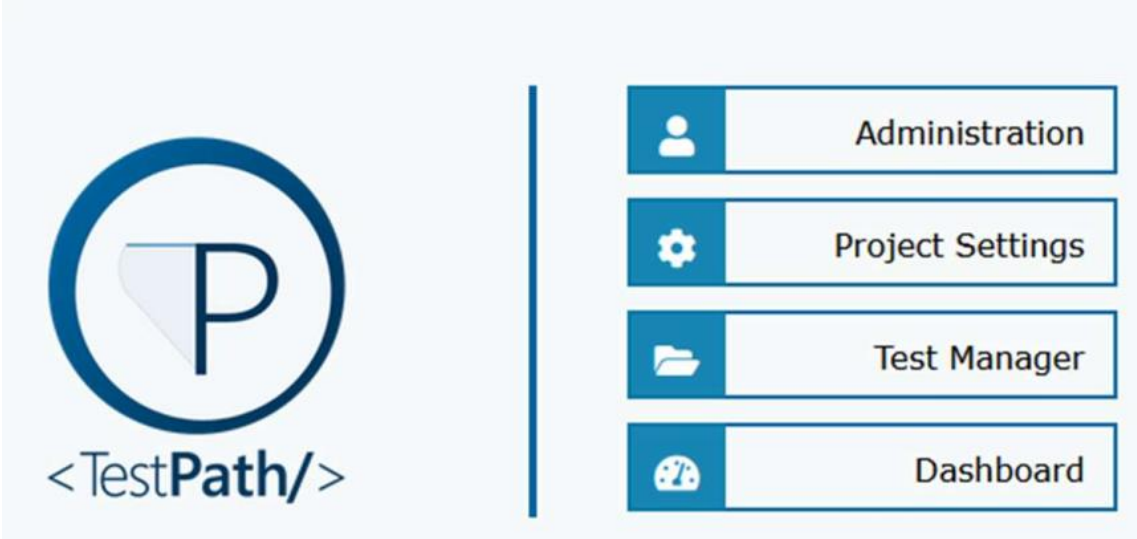


Figura 4- Escolha de Menu

O menu de administração possui algumas opções de submenu como:

- Tipos de utilizador, Utilizadores, Projetos, Permissões e Pools de agentes, onde são geridas as listas dos tipos de utilizador, utilizadores, projetos, permissões e pools existentes, respetivamente. Cada lista conta com as típicas opções de criar, eliminar ou editar as informações.
- *Logs* de erros, como o nome indica, é onde podemos observar os erros que ocorreram no TestPath.

❖ SQL, onde são realizadas consultas à base de dados.

O menu de Definições de projeto está constituído pelo submenu “Details” e é aqui que é feita a gestão das configurações do projeto, para o projeto selecionado após o login. Particularmente é aqui que podemos adicionar módulos de integração ou gerir a cache de integrações.

Dando como exemplo o Jira XRay, que foi a única integração utilizada dentro das ferramentas de gestão de teste durante o estágio, caso seja a *tool* selecionada, apenas é necessário fornecer o URL, Utilizador, Senha, Projeto, ID do Cliente e outras informações consideradas secretas.

O Menu de Test Manager é a parte integrante do TP à qual vamos dar um pouco mais de ênfase, pois é aqui que se concentram a maior parte das funcionalidades e se realizam mais ações relacionadas com o desenvolvimento e execução dos testes.

Quando entramos neste menu, vemos o nosso painel com o login efetuado através do nome de utilizador (Figura 5). Na primeira utilização nada é mostrado, mas temos a possibilidade de clicar nas seguintes opções:

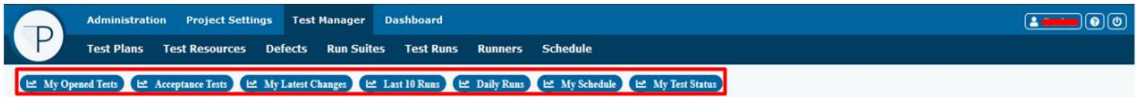


Figura 5- Opções rápidas

– Meus testes abertos, aqui encontra-se a lista de testes que não estão concluídos.

– Testes de aceitação, possui a lista de testes cujo status está em aceitação.

– Minhas últimas alterações, possui a lista das últimas alterações feitas.

– Últimas 10 execuções, obtemos a lista das 10 últimas execuções com o status e outras informações.

– Execuções diárias, lista das execuções feitas no próprio dia.

– Minha agenda, consiste num cronograma com uma lista de execuções de teste.

– Status do meu teste, apresenta um gráfico com a representação do status dos meus testes.

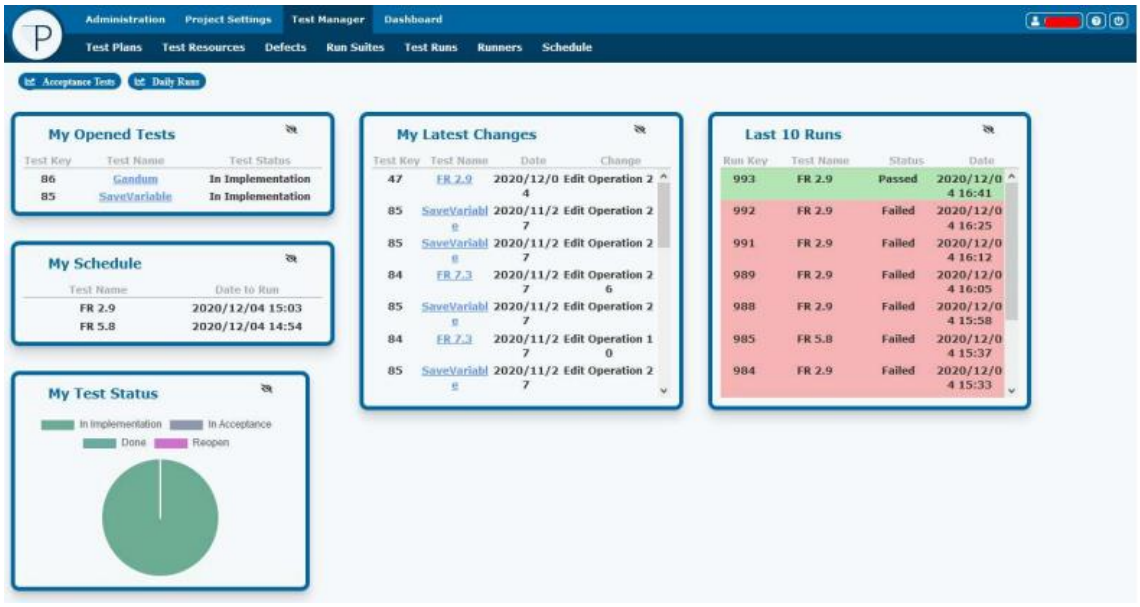


Figura 6- Exemplo de painéis com dados das opções rápidas

Passando para a lista de submenus:

1- Test plan

Aqui encontramos a principal janela onde fazemos a gestão dos testes. A primeira coisa com que nos deparamos ao clicar neste submenu é uma lista dos testes no projeto escolhido, como podemos observar na Figura 7, sendo que cada lista de testes pode ser agrupada ou filtrada pelo seu nome.

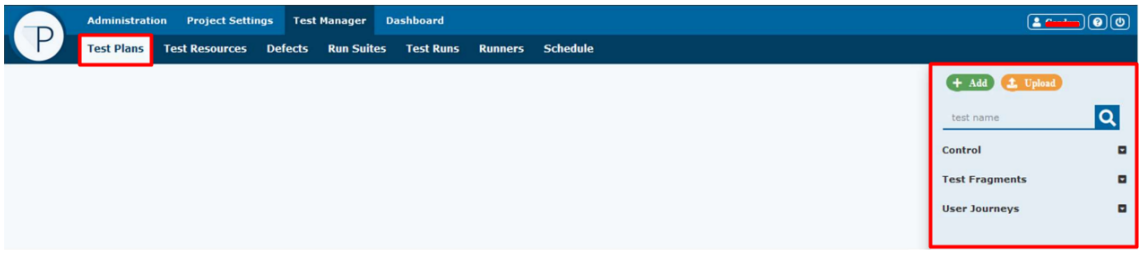


Figura 7- Lista de Testes por Projeto

Para criar um grupo basta clicar no botão “Adicionar” na parte superior (a verde) e em seguida clicar no botão “+ Criar Plano de Teste” (Figura 8), após a escolha do Nome do novo grupo é só clicar em criar e surgirá uma mensagem de sucesso.

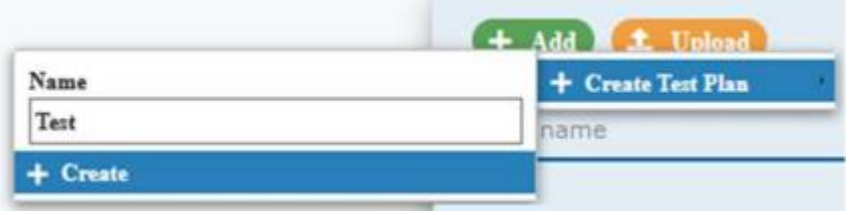


Figura 8- Exemplo de criação de uma pasta de projeto

Para criar um teste basta clicar no botão “Adicionar” na parte superior (a verde) e em seguida clicar no botão “+ Criar Teste” (Figura 9), após a escolha do Nome do novo teste é só clicar em criar e surgirá uma mensagem de sucesso.

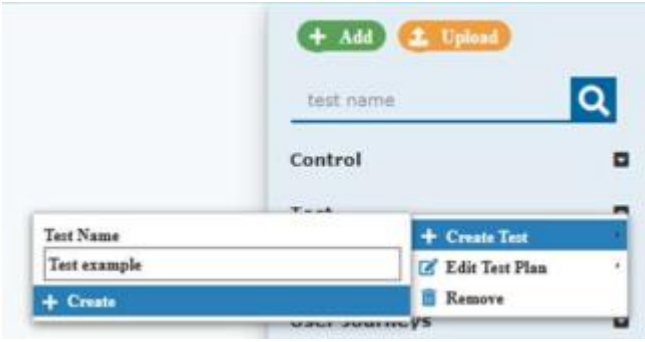


Figura 9- Exemplo de criação de um teste

Figura 10- Opções a realizar durante um teste

Existem também algumas opções que podemos realizar (Figura 10):

– Salvar: para salvar as propriedades

– Adicionar Data Driven: para criar um *data driven* que esteja vinculado a este teste específico

– Desabilitar: desabilita este teste

– Exportar: permite exportar o teste (excel, json ou java)

– Mover: para mover o teste, seja para um plano de teste existente ou para um novo plano de teste

– Excluir: exclui este teste

– Máquina do tempo: para ver as alterações que foram feitas neste teste

– Fluxo: para ver uma representação gráfica deste teste

– *Analytics*: mostra as últimas 10 execuções e o último detalhe da execução

– Mensagens: serve como uma discussão de mensagens onde os utilizadores podem escrever mensagens importantes

– Copiar: copiar algumas ou todos os passos para outro teste/pasta

– Executar: para executar o teste, seja no próprio PC, num agente ou numa pool de agentes

– *Debug*: para executar o teste passo a passo e verificar ao vivo a execução, ou seja, permite que o utilizador crie em tempo real um teste usando a aplicação em teste.

O *Debug* requer que o *Runner* para o utilizador autenticado esteja no estado ONLINE.

Enquanto uma sessão de *debug* está ativa, o utilizador não pode executar testes no mesmo *runner*.

Os seguintes recursos estão disponíveis no *debug*:

- Criar Sessão: cria uma Sessão de *Spy*
- Finalizar Sessão: finaliza a Sessão de *Spy* atual
- Atualizar ecrã: atualiza a captura de ecrã da aplicação aberta
- Alternar tamanho: alterna o tamanho da captura de ecrã
- Mostrar/ocultar *logs*: alterna a exibição dos *logs* de sessão
- Iniciar Gravação: Inicia o recurso de gravação

O *Debug* também contém um recurso de registo onde o utilizador pode simular ações, permite inserir um novo passo, permitindo até mesmo executá-lo antes de adicionar ao teste.

Para a opção de mapeamento de zona, o utilizador pode selecionar uma área e todos os objetos dentro dela serão mapeados.

Quando uma etapa é gerada usando o recurso Record, o objeto também será gerado, a menos que já exista, nesse cenário, usará o objeto existente.

Quando as novas etapas são criadas, elas são colocadas numa lista temporária, onde o utilizador pode escolher quais adicionar ao teste.

Os testes também podem ser criados ao fazer upload de um arquivo JSON (no mesmo formato do *Export*).

A nível de detalhes de um teste, cada um tem um conjunto de propriedades (Figura 11):

– Nome: o nome do teste, que pode ser editado
– Descrição: uma breve descrição do objetivo do teste
– Criador: o nome do utilizador que criou o teste
– Chave: a chave interna do teste
– Prioridade: a prioridade do teste (baixa, média, alta)
– Risco: o risco do teste (baixo, médio, alto)
– Status: o status do teste

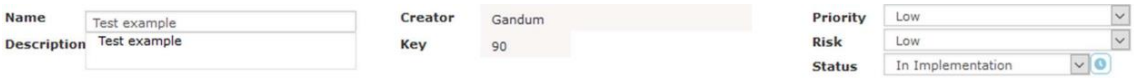


Figura 11- Detalhes de um teste

Os testes são o principal recurso do TestPath, é aqui que o utilizador define as etapas a serem executadas.

Os testes podem conter várias etapas de teste, onde as operações são colocadas, as operações são as ações usadas para avançar o teste, essas operações interagem diretamente com a aplicação, verificam determinadas condições ou criam variáveis para salvar valores.

2- *Test resources*

Dentro deste submenu possuímos outros vários submenus (Figura 12) com um conjunto de recursos para usar nos testes que são os seguintes:

Figura 12- Submenu Test Resources

a) *Applications*

Esta é a janela onde é feita a gestão dos objetos a serem usados no teste. Ao entrarmos podemos ver uma lista de aplicações no projeto escolhido após o login, como demonstrado na Figura 13:

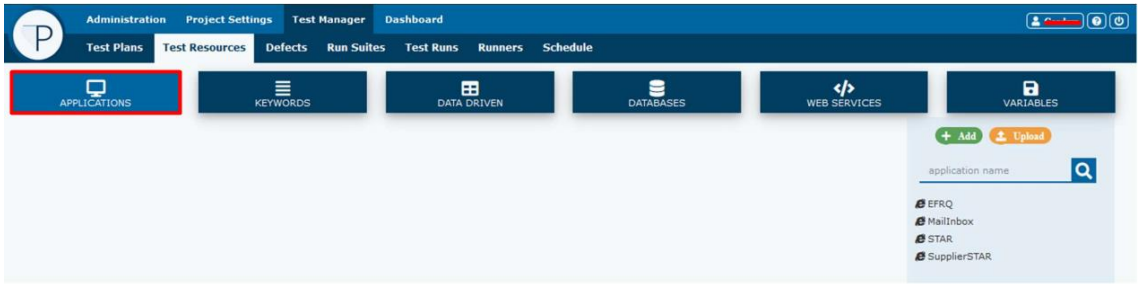


Figura 13- Submenu *Applications*

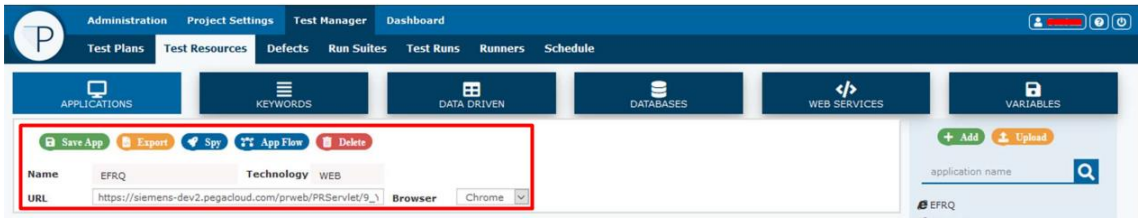


Figura 14- Opções do submenu *Applications*

Quando escolhemos uma aplicação, os seus atributos aparecerão na área de conteúdo superior (Figura 14), onde podemos ver então o Nome da aplicação, o URL que nos redireciona para a mesma, a Tecnologia em que esta está desenvolvida, no exemplo da imagem anterior temos uma aplicação web e por fim o browser onde pretendemos abrir a aplicação, que podemos escolher outros aqui ou no próprio teste, mas será o padrão, que por definição vem o Chrome.

A nível de opções temos alguns botões para:

– Salvar aplicação, que salva as alterações nas propriedades

– Exportar, exporta a aplicação e todos os objetos, somente para o formato Json

– Excluir, exclui a aplicação e todos os objetos

– *Spy*, aqui é utilizado o URL para ir até ao site e lá dentro podemos salvar objetos apenas escolhendo/clicando neles.

O *Spy* exige que o Runner para o utilizador autenticado esteja no estado *ONLINE*.

Enquanto uma sessão *Spy* está ativa, o utilizador não pode executar testes no mesmo *runner*.

Quando clicamos e entramos no modo *Spy*, o navegador padrão será iniciado e no TestPath veremos uma cópia do ecrã. O nome do objeto resultante será uma concatenação do nome da página com "nome do objeto" (Page1_BtnLogin, por exemplo).

Se formos ao navegador e clicarmos em "Atualizar" o ecrã será atualizado para o ecrã que estiver na app.

A opção "*validate*" é usado para verificar se um Xpath é válido ou não.

A partir daqui podemos clicar nos objetos que queremos adicionar à aplicação e após o TP reconhecer, eles irão aparecer na lista, como podemos ver na Figura 15:

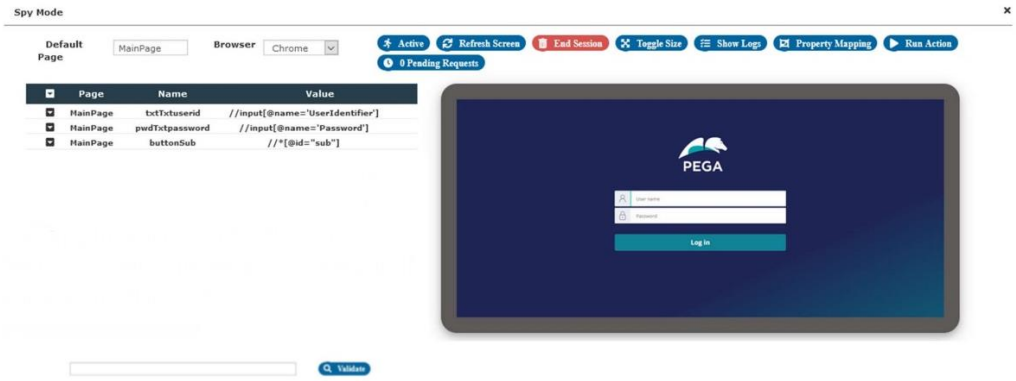


Figura 15- Exemplo de sessão *Spy*

Quando quisermos salvar os objetos basta clicar na seta que aponta para baixo e depois na opção "Adicionar todos".

É aconselhável mapear com a funcionalidade *Spy and Debug*, pois será a própria aplicação que gerará o valor do objeto de mapeamento, em vez de ser manualmente, tornando os objetos duplicados uma ocorrência menos provável.

Quando escolhemos uma aplicação, os objetos dessa aplicação aparecerão na área de conteúdo (Figura 16), a partir daqui podemos criar, editar ou excluir objetos e localizar onde o objeto específico está a ser utilizado (Figura 17).

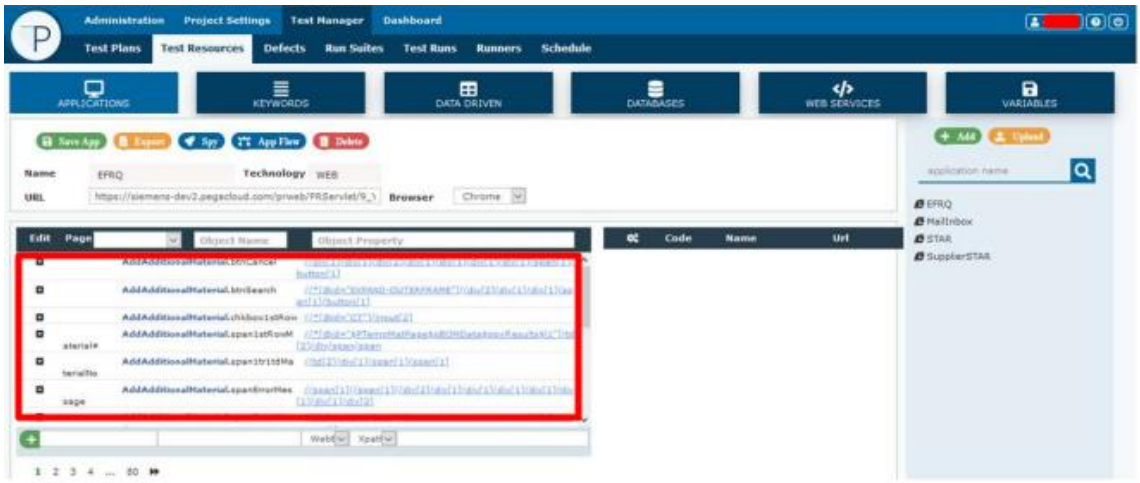


Figura 16- Exemplo de elementos mapeados

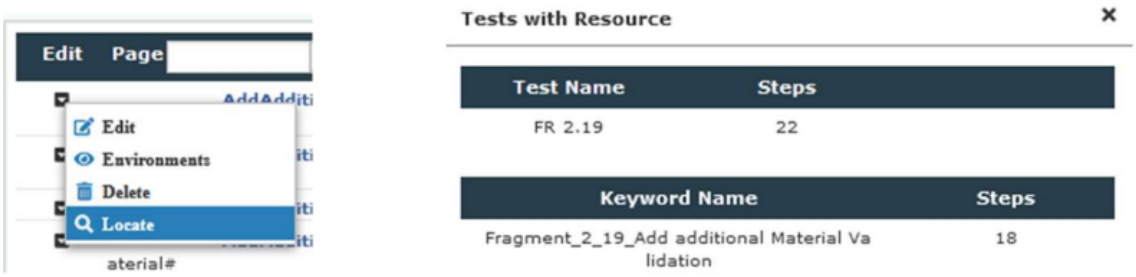


Figura 17- Exemplo de localização de objeto

As *Aplications* também suportam Ambientes, na lista secundária podem ser adicionados Ambientes (DEV, QUAL, etc.), que permitem uma URL diferente (em Aplicações Web por exemplo), e cada objeto pode ter uma propriedade diferente por ambiente.

b) *KeyWords*

A opção *Keywords* é onde podemos gerir um conjunto de etapas que podem ser chamadas de um teste ou de outra *keyword* para executar um processo, ou seja, são recursos que encapsulam operações para que os testes sejam chamados, chamando a palavra-chave em vez de todas as operações.

Qualquer conjunto de operações que normalmente são replicadas em vários testes deve se tornar uma palavra-chave (podem ser criadas a partir do próprio teste). Um teste pode chamar uma palavra-chave usando a operação *RunKeyword*.

Isto é excelente para quando temos um processo que é executado mais de uma vez, como um processo de Login ou Logout, devemos criar uma *Keyword*, isto porque é mais fácil atualizar uma *Keyword* do que atualizar variáveis e utilizar muito a repetição de passos.

Algumas vantagens das palavras-chave são:

- Redução do tamanho dos testes
- Redução da quantidade de combinações duplicadas de operações
- Facilidade de manutenção

As *keywords* permitem argumentos, isso significa que valores colocados nos campos Objeto ou Dados não precisam de ser fixos, eles podem ser espaços reservados para argumentos que o teste envia. Para colocar um argumento em qualquer campo, basta escrever &(X)& no respetivo campo, onde X é o número do argumento (começa em 1), isso significa que um campo contém &(1)&, o primeiro argumento quando um test chama uma palavra-chave, substituirá este *token*.

Para facilitar o entendimento, os argumentos podem ter nomes, na tabela lateral, isso atualizará a KW para ver o nome do argumento em vez do token, o teste também mostrará o argumento e o valor atual.

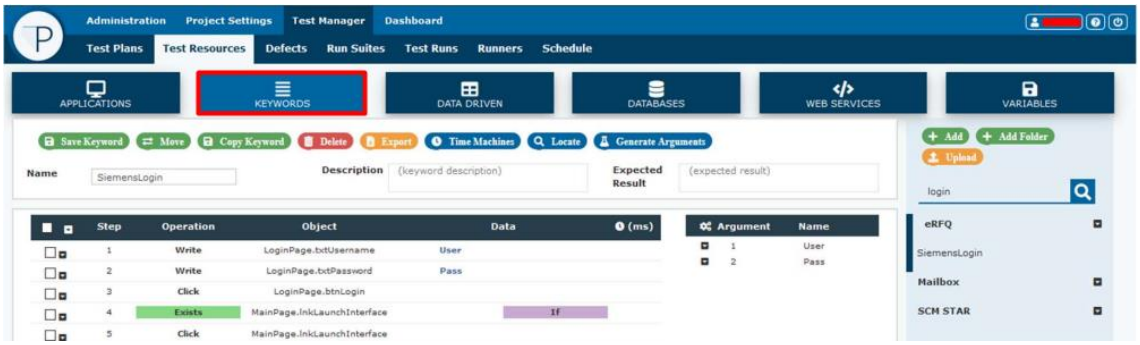


Figura 18- Exemplo de *Steps* de uma *Keyword* e respetivos argumentos

As KW são compatíveis com as seguintes ações (Figura 18):

Salvar palavra-chave: atualiza os campos de cabeçalho da kw (nome, descrições, resultado esperado)
Copiar palavra-chave: cria uma kw com base na anterior, com o novo nome selecionado
Excluir: exclui a kw
Exportar: exporta a kw em formato JSON
Time Machine: Exibe um histórico de alterações da kw
Localizar: Localiza os testes que usam a kw
Gerar Argumentos: Gera uma lista de argumentos na lista de argumentos

As KW podem chamar outras KW, mas não se podem chamar a si mesmas, caso contrário os testes correm o risco de entrar num *loop* infinito.

c) Data Driven

O *Data Driven* é onde podemos gerir um conjunto de dados que se pode usar em 1 ou mais testes, ou seja, permite que o utilizador salve campos de formulário fora do próprio teste, de modo a melhorar a escalabilidade do teste.

O *Data Driven* funciona como se fosse uma tabela em que o número de linhas e colunas são ilimitados (Figura 19). Os dados em questão são distribuídos em colunas e podemos ter vários valores para os mesmos dados, um para cada linha.

O *Data Driven* também suporta arquivos, não apenas texto e oferece algumas vantagens:

- Todos os argumentos podem ser armazenados fora do teste
- Facilidade de manutenção
- Argumentos (credenciais por exemplo) não precisam de ser repetidos

Isto significa que, por exemplo, se as credenciais de uma plataforma forem alteradas, a única alteração que o utilizador deve fazer é atualizar o *Data Driven*.

Para chamar um valor do *Data Driven*, no teste, em vez do valor normal, o utilizador chama esse valor com a seguinte fórmula: $(DATADRIVEN.COLUMNNAME. ROWNUMBER)$

O número da linha é opcional, por padrão será “1”, sendo que também pode ser substituído pelo *RunSuites*, mas se o número for selecionado no teste, esse será o valor.

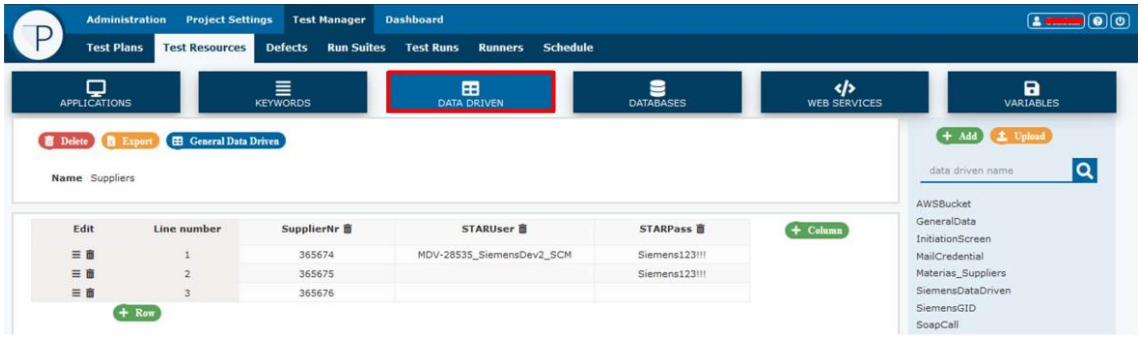


Figura 19- Exemplo de dados inseridos no *Data Driven*

d) *Databases*

A opção *Databases*, como o nome indica é onde podemos gerir e criar um conjunto de conexões de bases de dados para usar nos testes. Podemos também gerar consultas, estas consultas podem ser chamadas nos testes e os seus resultados salvos em variáveis. O utilizador também pode obter uma célula específica da tabela de resultados.

Para adicionar uma Base de Dados, basta criar um registo na lista, os seguintes campos serão obrigatórios (Figura 20):

- Nome de objeto para a BD, para que os testes possam conhecer cada BD a ser usada
- Tecnologia da BD
- Servidor da BD
- Nome da BD
- Nome de utilizador da BD
- Password

Para executar um Teste de DB, no caso de teste, é só selecionar a operação *RunQuery* e o resultado será salvo numa variável que conterá a tabela de resultados

Ao utilizar a operação *ReadCell*, o utilizador pode obter o valor de uma determinada célula da tabela (pesquisando a coluna e o número da linha)

O recurso *Spy* também está disponível para testes de BD, permitindo ao utilizador testar consultas e verificar o esquema da base de dados.

Figura 20- Exemplo da criação de registo na BD

e) *WebServices*

Nesta opção podemos gerir um conjunto de *webservices* que podem ser usados nos testes (Figura 21). Podem ser adicionados serviços REST ou Soap, estes podem vir de um projeto SoapUI, que deve conter *Test Suites* e *Test Cases*, e serão estes os objetos a ser utilizados.
O utilizador pode também selecionar qual o tipo de retorno dos que estão disponíveis (JSON, XML, RawText).

Os *WebServices* estão disponíveis no recurso *Spy*, permitindo que o utilizador teste as suas chamadas.

REST

Para criar e executar *REST Web Services*, basta criar um *WebService* e escolher REST como tecnologia. No botão "+ Nova Chamada REST" o utilizador pode criar chamadas REST, preenchendo os campos:

- Nome: o nome da chamada, que será utilizado nos testes (não pode ser duplicado)
- Método: o tipo de chamada (GET, POST, PATCH)
- URL: o URL da chamada
- BODY: o corpo da chamada (xml, json), este campo é opcional
- Header: os cabeçalhos REST necessários para fazer essa chamada (estes campos são opcionais)

As chamadas REST são suportadas por variáveis, podem ser colocadas nos campos URL, BODY e Header.

SOAP

Para criar e executar *SOAP Web Services*, basta criar um *WebService* e escolher SOAP como tecnologia. O WSDL pode ser adicionado no campo de texto e salvo usando o botão “+ Atualizar WSDL.”

Para gerar as novas chamadas, é salvar o WSDL seguindo as instruções acima referidas, e executar o recurso SPY, um botão chamado "Gerar Estrutura SOAP" estará disponível, clicando nele, todas as chamadas dentro do WSDL serão adicionadas ao *WebService*, e o corpo preenchido automaticamente com um modelo.

Não podem ser feitas novas chamadas, mas o utilizador pode modificar e copiar as chamadas existentes. Podem preencher os seguinte campos:

- Nome: o nome da chamada, que será utilizado nos testes (não pode ser duplicado)
- Ação SOAP: a ação SOAP que será executada (não é recomendável alterar)
- Serviço SOAP: a URL da chamada SOAP (não é recomendado alterar)
- BODY: o corpo da chamada (xml), este campo é opcional

As chamadas SOAP são suportadas por variáveis e podem ser colocadas no campo BODY.

Integração SOAPUI

O TestPath suporta testes de *Web Service*, com integração com projetos SoapUI. Para adicionar um projeto, o utilizador carrega um projeto .xml SoapUI, e serão exibidas listas de *TestSuites* e *TestCases* do projeto.

A Integração SOAPUI suporta as seguintes operações:

- Upload: carrega um projeto SoapUI

Para executar um *WebService Test*, no caso de teste, selecione a operação *RunWebServiceTest*, os dados serão *TestCase*; Property1=X; Property2=Y; (as propriedades são opcionais).

As propriedades globais devem ser definidas no SoapUI, usando o Menu Propriedades Globais, e nas etapas de teste no SoapUI, o parâmetro será ${PropertyName}.

Figura 21- Submenu de *Webservices*

f) *Variables*

A opção de variáveis é onde gerimos e vemos um conjunto de todas as variáveis, que contém um valor, criadas e usadas nos testes.

Temos uma tabela onde vemos o nome da variável, o último valor no final de um teste e se a variável é global ou não (Figura 22). Se uma variável for global, isso significa que podemos salvar uma variável num teste e usar esse mesmo valor noutro teste.

Existem várias ações que terminam com a criação de uma Variável (*SaveObjectVariable, SaveDataVariable, ReadExcel*), essas operações costumam ter o último campo Data aberto com o seguinte *placeholder*: nome da variável.

Ao escolher a opção de histórico de verificação podemos ver os valores que aquela variável já teve. O histórico mostra também a execução, o teste e o valor associados.

Podemos escolher outras opções como (Figura 23):

– Alternar entre local e global

– Limpar o histórico

–Eliminar a variável

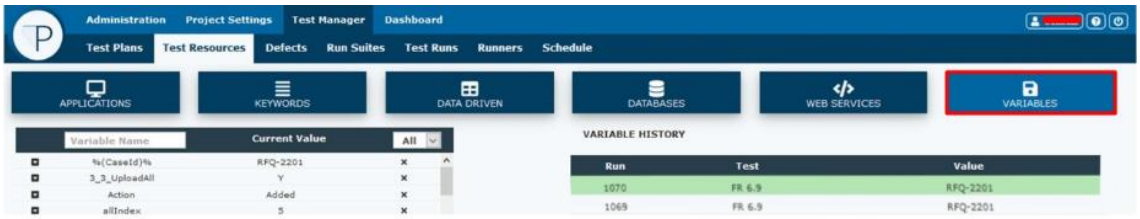


Figura 22- Exemplo de uma lista de variáveis

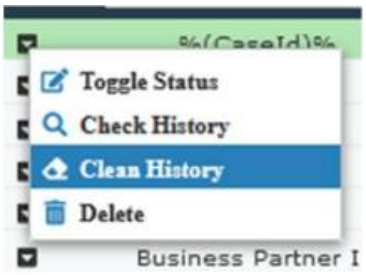


Figura 23- Lista de opções de ações a aplicar sob as variáveis

3- *Defects*

No menu de defeitos, como o nome indica, sempre que um teste é executado e encontra um erro, esse erro é registado como um defeito e é nesta opção que podemos verificar esses logs (Figura 24).

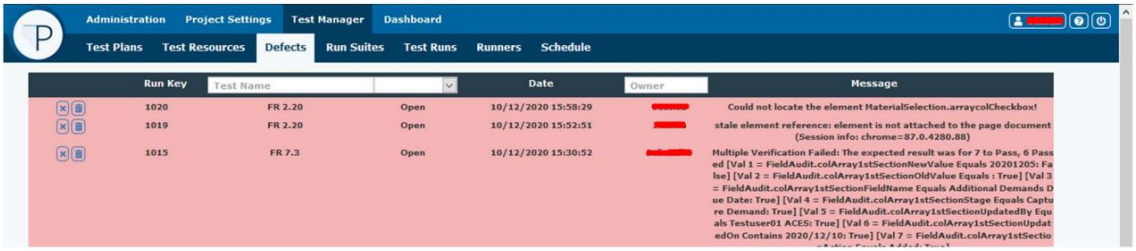


Figura 24- Exemplo de uma lista de *defects*

4- Run suites

Aqui temos a opção onde construímos o processo *End2End*, ou por outras palavras, a ou as baterias de testes.

Aqui temos acesso à lista de testes que serão executados cada vez que “correrMos” a *suite* de execução.

Podemos ver também as últimas dez execuções com um esquema de cores para distinção do status, aprovado a verde e não aprovado a vermelho.

Existe também a possibilidade de promover algumas substituições de valores de dados (somente) quando os testes são executados numa *runsuite* (Figura 25).

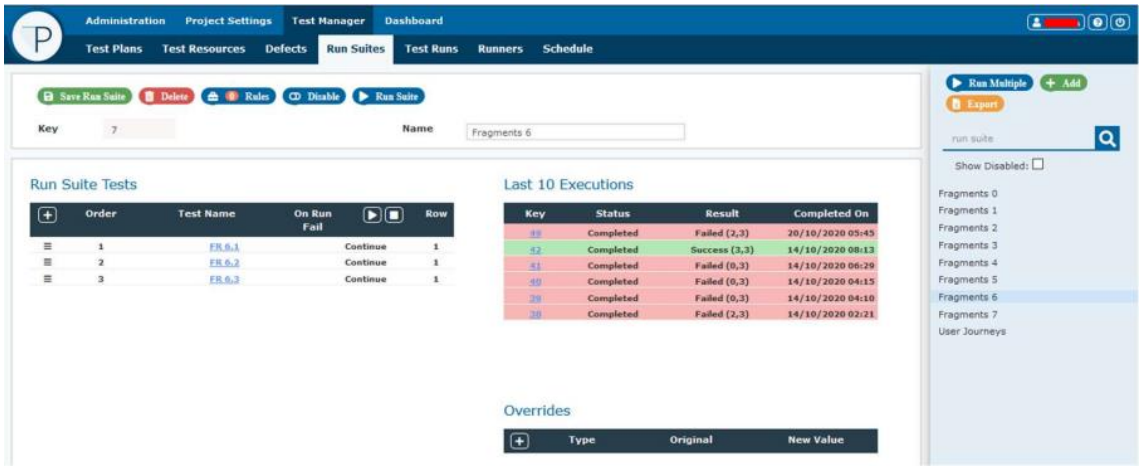


Figura 25- Exemplo de uma *RunSuite* e respetivas execuções

RunSuite permite as seguintes ações:

- *Save Run Suite*: atualiza o nome da RS
- Excluir: remove a RS
- Regras: Mostra a lista de regras da RS
- Desabilitar/Habilitar: Alterna o status ativo da RS
- *Run Suite*: Prepara a execução da RS

As RS podem ser configuradas para execução simultânea ou sequencial (a simultaneidade aplica-se apenas a execuções em agentes).

Para a execução da RS, algumas ações podem ser definidas antecipadamente. O que acontece em *On Run Fail*, se estiver definido como *Stop*, a *Run Suite* irá parar aqui depois que um teste falhar, se estiver em continue, ele continuará para o próximo teste

A ordem dos testes pode ser alterada reordenando os testes (arrastar e soltar) e estes podem ser desativados ou removidos.

5- *Test runs*

Nesta opção é onde vemos todas as execuções com os seus detalhes, quer finalizadas, quer durante a própria execução, ou seja, as informações são atualizadas em tempo real. Podemos filtrar as execuções por nome ou status do teste, por exemplo, ou outras opções presentes na Figura 26.

Também será possível reprogramar e verificar evidências dos testes executados nesta página através da visualização dos *logs*, visualização do relatório online, baixar em PDF ou ver o vídeo de teste, ou simplesmente efetuar a execução novamente (Figura 27).

Figura 26- Exemplo de várias execuções de teste em tempo real

Figura 27- Opção que permite visualizar evidências em vários formatos

6- *Runners*

Na opção *Runners* podemos ver a lista de utilizadores e a lista de agentes associados ao projeto. Estes possuem um esquema de cores de status, verde para online e vermelho para *offline*. Ao clicarmos num utilizador ou agente podemos ver as últimas execuções com um esquema de cores de status, verde para os casos de sucesso e vermelho para os casos de falha (Figura 28).

O utilizador pode reprogramar uma execução, verificar as evidências e acompanhar um rastreador em tempo real das *runsuites* em execução.

Figura 28- Exemplo de visualização de um *Runner*

7- *Schedule*

Esta opção permite a gestão do agendamento de testes ou *runsuites*, pois estes podem ser programados para serem executados num determinado horário, em vez de um modelo *Run ASAP*.

Ao clicar num dia, representado pelo calendário da Figura 29, podemos ver o teste agendado para execução naquele dia. Para fazer um agendamento basta clicar na opção desejada, preencher os dados e clicar em “agendar” (Figura 30).

Esta página também é o local para cancelar qualquer execução que ainda não tenha começado.

Ao agendar o utilizador pode:

- Selecionar o conjunto de teste/execução
- Selecionar a data/hora
- Selecionar onde executar (Utilizador ou Agente)
- Definir as configurações de recorrência (com que frequência repetir e quantas vezes)
- Agendamento da API
- Uma *RunSuite* pode ser acionada para executar usando a API

Isto permite integrações com ferramentas de CI/CD como Jenkins.

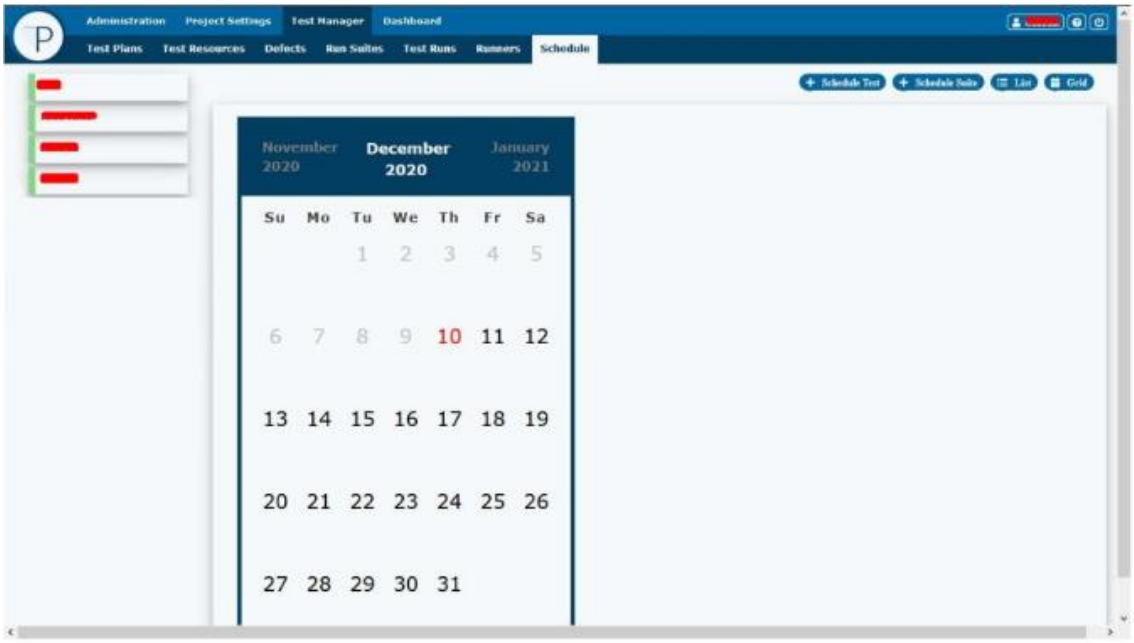


Figura 29- Exemplo de agendamento de execuções através do calendário

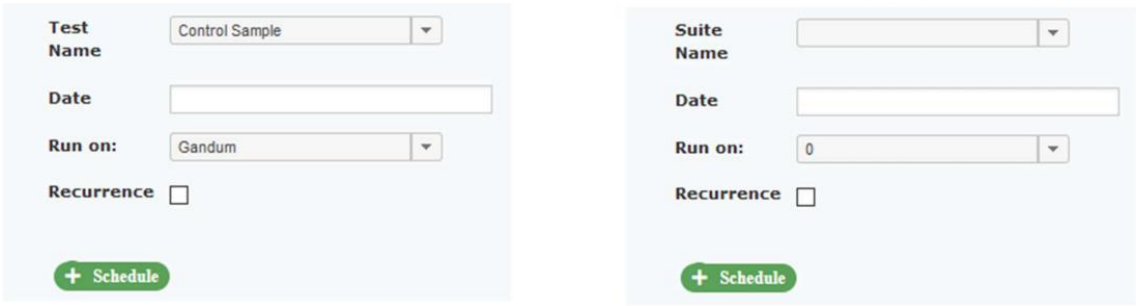


Figura 30- Exemplo de inserção de dados para agendamento

Dashboard

Neste menu (Figura 31) temos um conjunto de *dashboards* agrupados por:

Figura 31- Menu de *Dashboard*

1- Casos de teste

Aqui podemos ver todos os casos de teste e o status da última execução. É possível filtrar os mesmos pela Chave de teste, pelo Nome do teste, Status do teste e/ ou Criador.

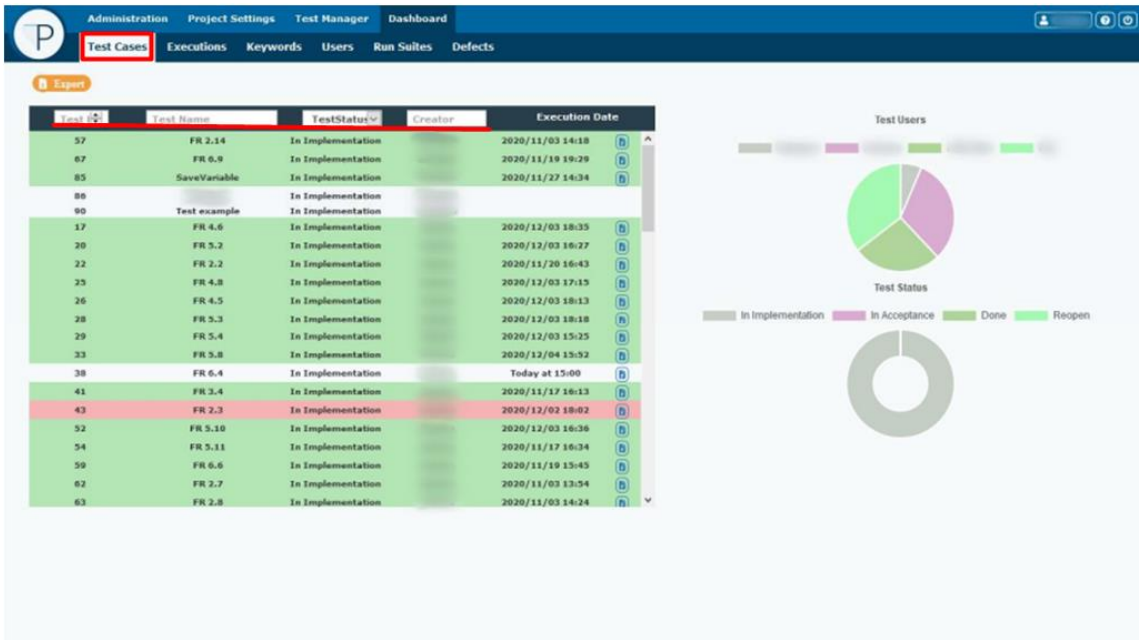


Figura 32- Exemplo do *dashboard* de *Test Cases*

2- Execuções

Aqui podemos ver a lista de execuções de teste entre as datas selecionadas e filtrar as mesmas através da chave e conjunto de execução, chave e nome do teste, status da execução e utilizador.

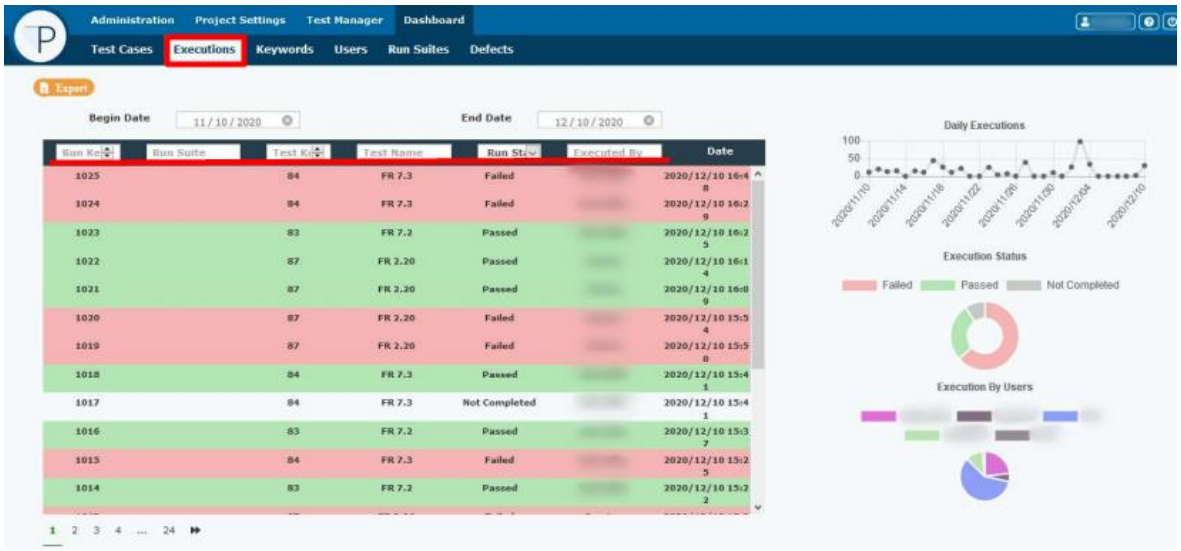


Figura 33- Exemplo do *dashboard* da lista de execuções

3- Palavras-chave

Aqui é possível observar a lista de palavras-chave, número de etapas e uso, filtrando pelo Nome da palavra-chave.

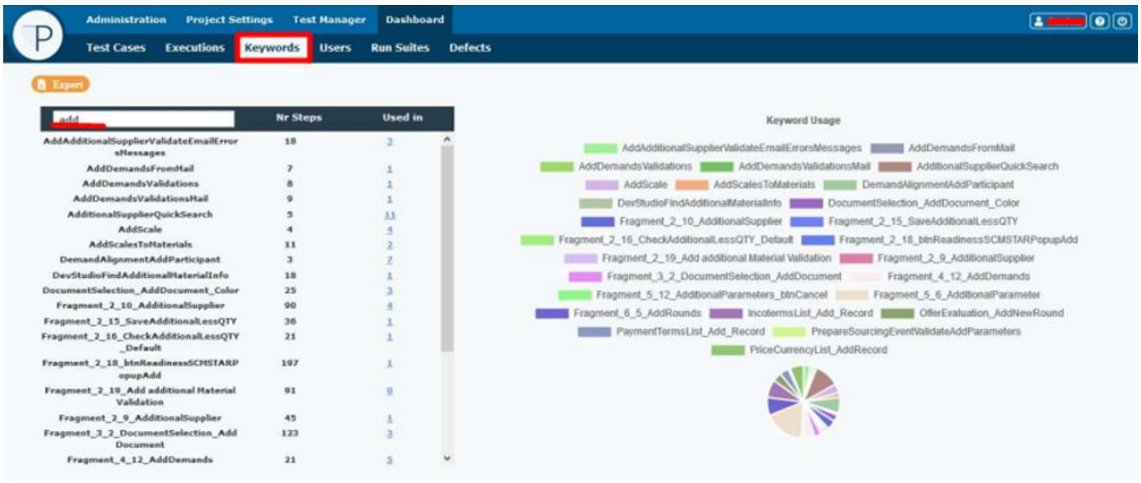


Figura 34- Exemplo do *dashboard* de *Keywords*

4- Utilizadores

É possível ver a lista de algumas estatísticas relacionadas com os utilizadores.

Figura 35- Exemplo da lista de estatísticas dos utilizadores

5- *RunSuites*

Aqui podemos ver a lista dos conjuntos de execução com o status e data da última execução, é possível filtrar pela chave e nome da suite.

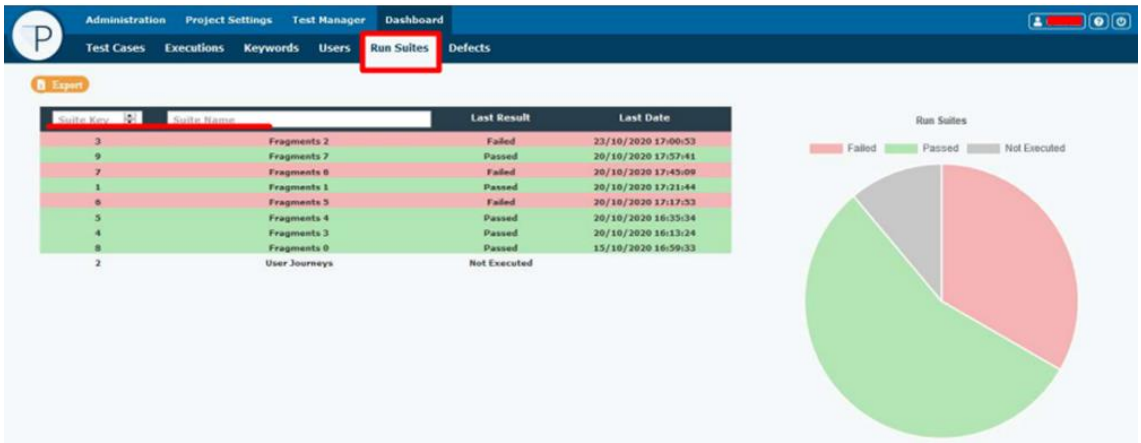


Figura 36- Exemplo do *dashboard* da lista de *RunSuites*

Nota: Alguns gráficos, como os que foram representados anteriormente (Figura 32, Figura 33, Figura 34, Figura 35 e Figura 36), apenas surgem quando os dados são selecionados, tal como nos conjuntos de

características acima.

6- Defeitos

Podemos ver a lista de defeitos (Figura 37) registados entre duas datas, podendo ainda filtrar pelo nome do teste, pelo proprietário e/ou status do teste.

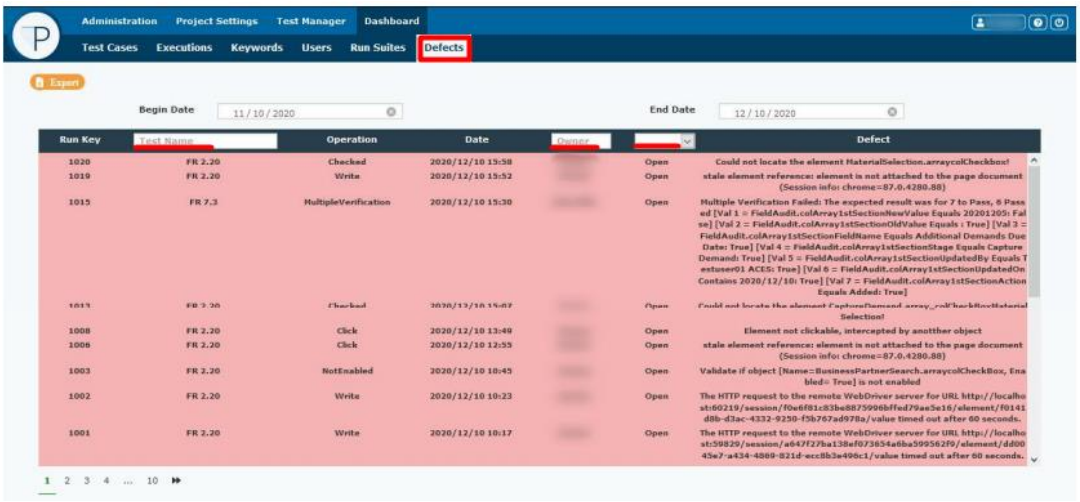


Figura 37- Exemplo da lista de *Defects*

4.1.1.5. BENEFÍCIOS

No que diz respeito a benefícios ou vantagens o TP oferece imensas, é uma ferramenta que acima de tudo compensa o seu investimento através da reutilização.

Por exemplo, a reutilização de evidências retiradas de um *Test Case* que permite fortalecer a documentação ao contribuir para a criação de manuais de utilizador, ou até mesmo fornecer o suporte aos Testes de Aceitação, são situações em que o TP pode agregar valor a um projeto.

Na vida de um *tester*, seja ele de que tipo for, o mais importante são as Evidências de teste e nesse aspeto o TP permite a gravação de vídeo e todas as etapas de um teste e as evidências em relatórios, isto permite ao *tester* a visibilidade total, rastreabilidade e auditabilidade de cada execução da etapa de teste.

Não só os relatórios, mas também os *dashboards* fornecem Informações de teste e resultados de forma mais gráfica, através desses *dashboards* existe a possibilidade de acompanhamento e monitorização em tempo real dos progressos em qualquer lugar.

A característica que podemos considerar mais vantajosa no TP é sem dúvida a fácil criação ou manutenção de um teste, ou seja, o facto da não existência de scripts e a não necessidade de desenvolvimento ou interpretação de código facilita imenso o trabalho e traz a vantagem adicional do foco nas regras de negócio, portanto profissionais com perfis mais funcionais ou não tão inteirados no assunto podem criar casos de teste automatizados, claro que características como um sentido crítico e raciocínio lógico não são dispensáveis.

Também a facilidade da automatização de testes de regressão que são de extrema

importância.

O TP é um facilitador rápido de atualizações e novas versões, para além da permissão de execuções em paralelo, de agentes de execução múltipla e regras e a capacidade para lidar com vários projetos simultâneos.

Outro facto que também é vantajoso é o de os casos de teste ficarem alojados na base de dados e serem fáceis de exportar para vários formatos como foi explicado na *overview* da aplicação, na anterior subsecção.

Uma grande eficiência dos testes também é possível devido ao processo de criação modular, isto vai permitir a alta reutilização de casos de teste, dando como exemplo a utilização de *KeyWords* ou *Data Driven* permite reutilizar vários steps e não torna o teste demasiado extenso e impercetível, ou até mesmo o facto de ser possível a cópia de vários steps de uns testes para os outros.

Esta reutilização de ativos de teste é possível em vários projetos.

Também a criação de cenários de teste *End2End*, ou seja, tudo num conjunto de testes com várias tecnologias.

O TP permite estandardização implementação de uma prática “standardizada”, ou seja, uma abordagem geral para a automatização de teste, quer-se com isto dizer que os testes podem na sua grande maioria ser muito diferentes, mas a forma em como os abordamos para proceder à automatização é sempre igual, desde o mapeamento de elementos até ao tratamento destes através das várias opções possíveis de ações.

Como referido anteriormente o facto de o TP permitir integrações com várias tecnologias, principalmente ferramentas de gestão e controlo de versões como Pega Agile Studio, Jira, Azure DevOps, Quality Center e etc., é muito vantajoso por várias razões, mas principalmente pelo facto do aumento de produtividade, maior segurança de dados, fim das falhas de comunicação, a utilização de *dashboards* e um controlo total de todos os processos. O facto de ser uma ferramenta colaborativa para teste facilita não só a gestão de testes, mas também a existência de equipas de automação *Onshore-Neasrhore-Offshore* que utilizam e exploram os ativos de teste de forma concorrente.

Por fim, o seu modelo altamente escalável, permite que a curva de aprendizagem seja reduzida e o processo de automação padrão permite que exista uma maior velocidade de implementação, e as ações de manutenção dos testes sejam ágeis e eficientes, o que faz com que o tempo de disponibilização de uma aplicação para o mercado seja menor.

4.1.2. Resultados e conclusões

Aquando do começo da automatização de testes para o Projeto 1, para a primeira *release* em que iriam existir testes automáticos, uma vez que iriam ser feitos os desenhos de teste, os mapeamentos de elementos através do TP e o próprio desenvolvimento dos testes optou-se pela realização de um número mais baixo destes, de modo a conseguir avaliar se seria possível aumentar o número de testes num seguinte *Sprint*.

Estas tarefas foram executadas sem qualquer contratempo, até efetivamente chegar a semana da *release*. O suposto seria após o *Sprint Planning Meeting* desenvolver o que ficara planeado, tanto

na parte de desenvolvimento como na parte dos testes, sendo que as primeiras 3 semanas seriam para esse efeito e a 4ª semana, a semana destinada à *solution release* seria para execução de testes, tanto os manuais como os automáticos, de modo a verificar se estaria tudo em condições para seguir para produção. Porém, surgiram os contratempos que vieram sempre a acontecer nos Sprints seguintes:

- Tarefas atribuídas em excesso aos *developers*, pelo que as 3 semanas destinadas ao desenvolvimento nunca eram cumpridas, fazendo com que na semana dedicada aos testes, ainda estavam desenvolvimentos a decorrer e *deploys* a ocorrer nos períodos de execução dos testes, fazendo com que estes falhassem e não fosse possível fazer o trabalho em condições e no período suposto.
- O facto de apenas existirem 2 ambientes antes de Produção, ou seja, apenas existia o ambiente de Desenvolvimento e o ambiente de Qualidade, sendo que era proibido executar testes em Desenvolvimento, com a justificação de ser um ambiente instável. O que está completamente errado, pois os testes apenas eram executados em Qualidade.
- A existência de alterações a certas funcionalidades da aplicação que não eram comunicadas à equipa de qualidade, com resultado na falha da execução da bateria de testes, sendo que muitas destas funcionalidades estavam presentes em vários fluxos de testes, grande maioria eram testes de regressão que acabaram por ter de ser "eliminados", devido às alterações mudarem o fluxo ao ponto de não se conseguir proceder a alterações, na maioria dos casos.
- Foi pedido ao *tester* de automação que realizasse um teste *End2End* que seria mais fácil, ou seja, o objetivo seria a criação de um teste único em que fosse testada toda a aplicação web e outro teste dentro dos mesmos moldes para mobile Android. Em primeiro lugar, esta é uma definição errada do que são testes *End2End*, estes testes *End2End*, estavam sim a ser feitos, mas nos moldes corretos, vários testes de modo a cobrir todas as funcionalidades possíveis que no final resultavam em duas *runsuites*, uma para a aplicação web e a outra para a aplicação em mobile android. Isto porque da maneira como os testes estavam a ser pedidos iriam dar imensos problemas como o teste ficar mais lento e muito demorado, em caso de um eventual *timeout* ou na existência de alguns bugs, o teste falhava todo por completo em vez de falhar um ou dois numa *runsuite* inteira, e dificultava imenso a manutenção desse mesmo teste ou até mesmo dificultar alterações. Pelo que, estes factos foram todos mencionados e explicados a toda a equipa, sendo que ainda assim houve uma insistência para que isto acontecesse até efetivamente se perceber que não dava certo.

Após estas ocorrências, ao fim de 2 Sprints, na *Sprint Review Meeting* foi mostrada a insatisfação por parte dos dois elementos da qualidade, pois não estavam reunidas quaisquer condições para serem exercidas as suas funções, ao que foram prometidas novas soluções. Soluções essas que simplesmente, não só pioraram este cenário, como ainda agravaram um pouco

mais a situação.

Uma das soluções implementadas foi aumentar o tempo do desenvolvimento, porém esse aumento existiu à conta do corte do tempo para a qualidade, portanto, o tempo de *solution release* que tinha um período temporal de uma semana passou a contar com apenas 3 dias, sendo que o último dia não poderia ser completo, pois a passagem para produção tinha de ser feita.

Logicamente isto trouxe sérios problemas, em primeiro lugar, porque se uma semana para proceder à execução dos testes, fossem eles manuais ou automáticos, era pouco, 3 dias era impensável, em primeiro lugar porque os testes manuais só por si são mais demorados e existiam numa quantidade considerável, em segundo lugar, os testes automáticos na quantidade existente para a altura não eram poucos, a sua execução não demoraria mais de 45 minutos, porém é irresponsável achar que ao serem executados todos os testes que não existam bugs e que vai correr tudo bem e que os desenvolvimentos estão automaticamente prontos a ir para produção, ainda para mais quando os desenvolvimentos eram passados diretamente do ambiente de Desenvolvimento para o ambiente de Qualidade sem qualquer teste realizado pela equipa de desenvolvimento, uma vez que os *testers* não estavam autorizados a testar em desenvolvimento.

Todos estes acontecimentos trouxeram constrangimentos e limitações que obviamente resultaram naquilo que não era o objetivo, evitar um software sem qualidade em produção e nas “mãos” do cliente.

A juntar a estas situações, existiram também muitas situações de desconsideração profissional e sempre com os dois elementos da equipa de qualidade como:

- Nas reuniões diárias de equipa (*dailys*), cujo período temporal era de 15 minutos, que nunca eram respeitados, sempre que chegava à equipa de qualidade nunca era concedida a palavra com a justificação de que o tempo tinha sido excedido e o essencial estava dito.
- Nos planeamentos dos Sprints, nos *backlogs* e restantes *boards*, os testes e as tarefas dos *testers* eram sempre as atividades que nunca eram contempladas em nada, sendo que os elementos de qualidade eram sempre os únicos cujas *tasks* não tinham *story points* atribuídos, com a justificação de não valer a pena.
- Com o decorrer do Sprint, existiam alterações que eram decididas pelo PO do projeto, sem justificação aparente e sem serem documentadas, sendo que a maior parte das alterações afetava o trabalho em andamento.
- O desinteresse e profundo desconhecimento daquilo que estava a ser feito a nível de testes automáticos.
- Os vários pedidos por parte do *tester* de automação para juntar aos testes de regressão, testes de *happy path, non happy path*, entre outros, que eram sempre recusados devido a não serem uma prioridade.
- Após muita insistência por parte do *tester* de automação, é que os gráficos com os resultados das execuções dos testes automáticos passaram a constar nos relatórios de estatísticas finais dos Sprints.

- Vários momentos de desautorização, em que os dois elementos da equipa de qualidade mostram evidências em como os testes não passam e a aplicação não tem condições para passar para o ambiente de Produção e o PO em conjunto com o GP autorizam essa mesma passagem porque tinham de ter algo para apresentar ao cliente.
- Após o *report* de alguns bugs, a culpa de desenvolvimento questionava sempre a equipa de qualidade das ações que tinham praticado para "estragar o programa" e que o problema poderia estar na ferramenta de automação, o TestPath, mesmo mostrando evidências que os mesmos bugs ocorriam caso fossem feitos testes manuais, quando a equipa de qualidade estava simplesmente a cumprir as suas funções e a colaborar com a equipa de modo que o produto ficasse da melhor forma possível.

Considerando todas estas limitações e fatores, os resultados ficaram aquém do esperado e apenas resultou na desmotivação e monotonia do *tester* de automação no projeto.

No que diz respeito ao Projeto 2, a situação não foi tão drástica, porém os resultados obtidos não foram muito diferentes.

Primeiramente não existia um consenso por parte da equipa no que diz respeito ao ter testes automatizados no projeto, isto porque a maior parte da equipa achava que os testes automatizados faziam sentido, mas o PO queria mais um *tester* manual para libertar trabalho ao colega que já integrava a equipa nessa função sendo que durante muito tempo os testes automáticos estiveram suspensos e a equipa de qualidade deste projeto apenas executava testes manuais.

Entretanto, por motivos externos, o PO deste projeto foi substituído, pelo que com o novo PO houve um entendimento e por fim começou-se a cumprir aquilo que seria o plano inicial, automatizar os testes de regressão, que eram bastantes e muito extensos. Após estarem quase todos os testes de regressão automatizados, foi tomada uma decisão a nível interno de que a aplicação deixaria de funcionar nos moldes em que estava a funcionar até à data e esta iria ser integrada dentro de outra aplicação, pelo que iriam mudar todos os processos. Em suma, todos os testes já automatizados iriam ser "deitados ao lixo".

No fim de descritas as situações e experiências destes dois projetos, houve a tentativa por parte de outro projeto em início de desenvolvimento de integrar testes automáticos, mas por motivos de suspensão dos testes automáticos em muitos dos projetos do cliente e da necessidade, por parte da Atos, da colocação da *tester* de automação noutro cliente e noutros projetos, terminou assim a colaboração com o Cliente 1.

Em suma, desta experiência fica a aprendizagem de que o ciclo de vida do teste faz parte do ciclo de vida do software, é muito importante que os testes devam ser iniciados num período equivalente. O processo de design e desenvolvimento de testes é tão complexo quanto o processo de desenvolvimento do software em si, nunca o devendo desvalorizar. Se os testes não forem iniciados juntamente com as primeiras *releases* executáveis do software, o esforço de teste retardará a descoberta de muitos problemas no ciclo de desenvolvimento, sejam os testes manuais, automáticos

ou de outra natureza.

Embora as tarefas de planeamento e definição de teste com início antecipado possam expor falhas ou imperfeições no trabalho de especificação inicial, é recomendável escolher o trabalho de teste com cuidado e antecedência. Além da possibilidade de retrabalho já mencionado, a equipa de teste precisa de ter cuidado para manter a sua função como orientadora de qualidade imparcial e não se desviar dos requisitos e das tarefas de design iniciais, atuando como "guardiã da qualidade" em todo o projeto.

Esta experiência foi também a prova de que não adianta ter profissionais qualificados, motivados a cumprir o seu trabalho, e bons recursos tecnológicos, se depois por parte de toda uma organização não se praticar uma cultura de qualidade.

A forma de como os testes serão executados depende de vários fatores:

- o domínio da aplicação;
- o orçamento;
- a política da empresa;
- a tolerância de risco;
- a equipa.

Coisas que na sua generalidade falharam desde o início e que por alguma resistência à mudança, não se reverteram.

O objetivo fundamental nesta dissertação de provar os impactos dos testes automáticos não foi conseguido nesta experiência com este cliente, uma vez que não se conseguiu provar se a existência destes foi positiva ou negativa. Mas fica a certeza de que se houvesse organização, respeito e conhecimento por aquele que é o trabalho de um profissional de QA, o impacto tinha sido positivo, pois a ferramenta utilizada tinha muito potencial e funcionalidades que não foram aproveitadas nem em metade, e dadas as suas características, permitia executar o trabalho rapidamente e eficazmente. Mas dadas as circunstâncias mencionadas anteriormente, em nada resultou.

4.2. Cliente 2

Após entrada no Cliente 2, foram feitas várias reuniões, não só para efeitos de apresentação às várias equipas, mas para apresentar os vários projetos, normas do cliente, os vários processos e esclarecimentos de qualquer questão que se pudesse vir a colocar. Dessas reuniões iniciais resultou que:

- A *tester* de automação iria integrar 3 projetos em simultâneo sendo que em dois iria apenas automatizar testes e num deles, por necessidade do cliente, era necessário alguém que executasse testes manuais.
- Dos três projetos, o Projeto 1, que apenas requeria testes manuais era um projeto que já

contava com 6 meses de desenvolvimento. O Projeto 2 era um projeto que consistia em executar testes de regressão de um portal de forma automática, sendo que esses testes já estavam desenvolvidos, porém seria sempre necessária a manutenção desses testes e o desenvolvimento de novos, conforme fossem acrescentadas mais funcionalidades, embora algumas funcionalidades não pudessem ser automatizadas pelo que na equipa existiam mais 3 colegas a executar testes manuais. O Projeto 3 seria um projeto novo para integrar o portal referido no Projeto 2, que ainda estaria na fase da definição de requisitos, mas já se tinha o objetivo de que os testes para esse projeto seriam todos automatizados e era importante todas as partes envolvidas (GP, Analista funcional, Designer, *Developer* e *Tester*) acompanharem todo o processo desde início e todos participarem nas tomadas de decisão.

- Todos os projetos do Cliente 2 tem o seu desenvolvimento na linguagem de programação JAVA.
- A metodologia utilizada em todos os projetos é o *Kanban*.
- Todos os projetos passam por vários ambientes desde o ambiente de Desenvolvimento ao ambiente final de Produção, sendo a ordem Desenvolvimento-Piloto-Formação-Qualidade-Pré-Produção-Produção.
- As semanas iniciais iriam ser dedicadas à leitura de toda a documentação existente relacionada com os 3 projetos em questão e a uma formação para aprender a trabalhar com toda a tecnologia envolvida em todos os projetos e utilizada pelo Cliente 2, com especial foco na *framework* utilizada para a automação de testes, a FRAT.

Em seguida irão ser descritas a metodologia e todas as tecnologias utilizadas pelo Cliente 2, que são necessárias para o processo de desenvolvimento de testes, sejam eles manuais ou automatizados.

1 - Kanban

Kanban é um termo japonês que significa “cartão” ou “sinalização”. Assim sendo, o método Kanban resume-se a uma forma de gestão visual que permite controlar o fluxo de trabalho através da utilização de cartões e colunas coloridas.

A ideia é que, conforme as atividades sejam executadas, os cartões são movidos de coluna, permitindo ao gestor de projeto e restante equipa acompanhar o progresso do projeto e identificar a capacidade de novas entregas.

O Kanban possui 5 principais benefícios:

- Autonomia;
- Priorização de tarefas;
- Aumento da produtividade;
- Redução de custos;
- Colaboração.

2 - WIKI

Este é um software colaborativo que permite a edição coletiva dos documentos usando um sistema que não necessita que o conteúdo tenha de ser revisto antes da sua publicação. Portanto, é aqui que se encontra toda a documentação relacionada com tudo o que se passa no Cliente, que vai desde documentação relacionada com os projetos, com as tecnologias utilizadas, processos, equipas envolvidas, etc.

O funcionamento é muito fácil e intuitivo, a partir do momento em que a estrutura de página e até o próprio layout é muito semelhante ao site da Wikipédia.

3 - JENKINS

Jenkins é um servidor de automação de código aberto, que ajuda a automatizar as partes do desenvolvimento de software relacionadas à construção, teste e implementação, facilitando a integração e a entrega contínuas.

Sendo que, no cliente, o único objetivo do uso do Jenkins é o de simplificar a execução dos testes automatizados. As principais características e vantagens são:

- Não requer instalação ou configuração do ToolKit (Posteriormente será explicado o que é o Toolkit);
- Qualquer pessoa pode meter os testes a correr;
- Execução em todos os ambientes;
- Todos os testes que são executados são geradas evidências no CukeReports;
- É importante que ao realizar a entrega, a versão atual do código de teste seja incluída.

4 - CUKEReports

Tem por objetivo guardar e exibir o resultado da execução dos cenários de testes executados localmente ou através do Jenkins.

• As execuções realizadas em versão SNAPSHOT (localmente) ficam disponíveis por 30 dias, após esse período elas são apagadas;

• Vermelho: Com Falha;

• Verde: Com sucesso;

• Laranja: Não implementado;

• Para aceder às evidências carregar no botão *Screenshot*.

5 - Bugzila

O Bugzila é um programa de gestão de erros através de uma página Web.

O programa é usado para acompanhar os projetos do cliente. O software permite que os utilizadores, neste caso os *testers*, enviem tickets nos quais os desenvolvedores possam atuar, ou seja, permite que os utilizadores enviem tickets e que os membros do projeto atribuam aos bugs um nível de gravidade e atribuam esses mesmos bugs a desenvolvedores específicos para serem analisados e corrigidos. Sendo que no final do desenvolvimento e repetição do teste para verificação

da correção desses bugs, é ao *tester* que cabe a função de controlar o estado da entrega que inclui os bugs e os dá como fechados ou não. Como em muitos outros projetos, o Bugzila possui uma licença de código aberto.

6 - SVN

O Apache Subversion (SVN) é um software livre e de código aberto desenvolvido pela Apache Software Foundation que atua como um sistema de controlo para rastrear alterações em arquivos, pastas e diretorias. É usado para ajudar na recuperação de dados e no registo do histórico de alterações feitas ao longo do tempo.

Funciona como um repositório onde está alojado todo o código de todos os projetos, organizados pelas várias fases como a de Análise e requisitos, Desenho de aplicação, Desenvolvimento e implementação e por fim a pasta de testes, em projetos com testes manuais existe uma pasta para a colocação de evidências.

7 - IDE (Eclipse)

É o IDE para desenvolvimento Java, porém suporta várias outras linguagens a partir de plugins como C/C++, PHP, ColdFusion, Python, Scala e Kotlin.

Foi feito em Java e segue o modelo open source de desenvolvimento de software.

O Eclipse é o IDE escolhido devido a ser o que tem mais compatibilidade com a ferramenta de automação e por motivos de todas as equipas utilizarem os mesmos IDE’s sendo que é possível optar pelo Netbeans ou o IntelliJ. É no Eclipse que vai ser extraída a pasta do projeto com a pasta do projeto FRAT incorporada, para ser desenvolvido todo o código dos testes e os Feature Files, que serão posteriormente descritos.

4.2.1. FRAT

Projeto desenvolvido por um dos departamentos do Cliente 2 que tem como objetivo o desenvolvimento de uma Framework para automação dos testes (FRAT), que normaliza a implementação de testes automáticos e facilita o processo de testes dos mesmos.

A FRAT acelera a implementação de testes automáticos para um projeto de desenvolvimento, ao proporcionar um sub-projeto de testes reconfigurável.

Neste contexto os testes automatizados são um conjunto de scripts que descrevem o comportamento esperado, os "Feature Files", juntamente com um projeto de software escrito na linguagem de programação JAVA.

Nos "Feature Files" é descrito o comportamento esperado de um determinado cenário de teste usando Cucumber de forma a produzir a documentação, mas também o "*source code*" do que vai ser testado.

O projeto de software JAVA usa os "Feature Files" para gerar protótipos de código, que são depois executados de forma a validar todos os passos do teste.

Este projeto de software está integrado em cada projeto e reside num repositório SVN dentro da pasta destinada aos testes.

Os pré-requisitos para usar esta framework de testes são:

- *Skills* de developer JAVA, bem como sentido crítico para poder organizar todo o projeto de forma coerente;
- Configuração do Ambiente de Desenvolvimento necessário para um projeto de testes, disponível num *toolkit*. Esse *toolkit* contém uma máquina virtual (VM) com o sistema operativo CentOS (Linux) e as ferramentas necessárias para o desenvolvimento de testes, incluindo o IDE que preferir (Eclipse, NetBeans ou IntelliJ Idea), Subversion como repositório e Maven para compilação, execução e gestão de dependências. Este conjunto de ferramentas pode ser utilizado por equipas de projeto ou pelos programadores ou *testers* nas suas máquinas locais. O *toolkit* é também reutilizável e garante que todos os programadores/*testers* terão o mesmo ambiente de desenvolvimento, e não haverá conflito de versões de software.

A partir do momento em que existam as condições e o desejo de que o projeto possua testes automatizados, as etapas/páginas para desenvolver um projeto de testes são as seguintes:

- Checkout do projeto FRAT
  Após a configuração do ambiente de desenvolvimento é necessário efetuar o checkout da FRAT no repositório de SVN. A intenção é usar o código da FRAT para servir de base para a criação do projeto de testes de um projeto de desenvolvimento.
- Criar Projeto de Testes
  Precisa-se construir um projeto de teste (e também módulos de teste) no projeto de desenvolvimento para aproveitar o que FRAT oferece. Para isso, podemos aproveitar do projeto de testes desenvolvido de FRAT para construir o nosso projeto de teste.
- Desenvolvimento no Projeto de Testes
  Quando já existir um projeto de testes, será possível desenvolver testes automáticos aproveitando as funcionalidades de FRAT.

  Para o desenvolvimento de testes automáticos serão necessárias tarefas como:
  - Iniciar um ambiente local para que este seja usado no desenvolvimento/execução;
  - Efetuar *check-in* e *check-out* do projeto;
  - Efetuar *update* do projeto.

- Verificação da qualidade do código fonte com Sonar

A nível daquilo que a FRAT proporciona para desenvolver cenários num projeto de

desenvolvimento e as tecnologias que usa nos seus módulos. Em seguida são detalhados todos os módulos que a FRAT oferece:

- **Módulos de Implementação da FRAT**

Cada módulo de implementação contém funcionalidades com objetivos diferentes que auxiliam na validação do comportamento de um sistema. Essa validação é desenvolvida nos módulos de testes do projeto (a FRAT também contém os seus próprios módulos de teste).

- **Módulos de Testes da FRAT**

Nesta página temos uma descrição de alto nível de projetos (ou módulos) de testes disponíveis na *framework* de testes (FRAT).

Cada módulo de testes contém cenários em Cucumber que descrevem o comportamento esperado de funcionalidades implementadas em FRAT.

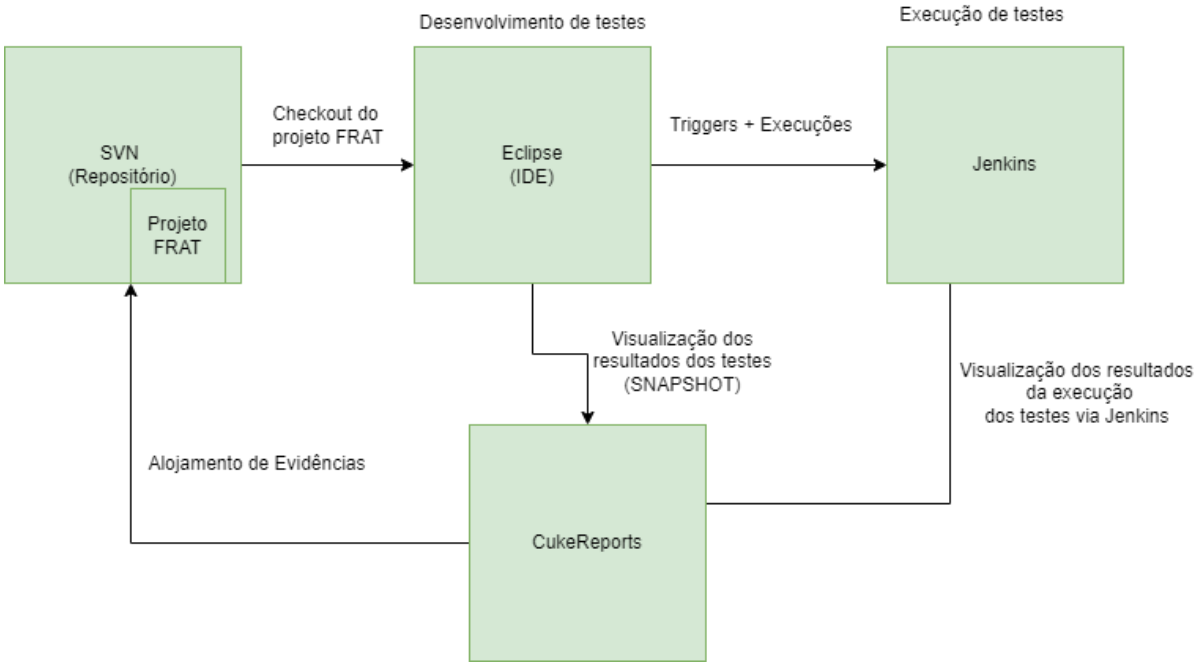


Figura 38- Diagrama da integração das várias tecnologias no ciclo de teste

4.2.1.1 TECNOLOGIAS USADAS EM INTEGRAÇÃO COM A FRAT

1) Cucumber

O Cucumber é uma *framework* de automação de testes que utiliza a técnica *Behavior Driven Development* (BDD). A técnica BDD utiliza metodologias de desenvolvimento orientado ao comportamento, isto é, o desenvolvimento dos testes é feito baseado no comportamento que o sistema tem ou deve ter. O objetivo é escrever cenários de comportamento que ilustrem as regras de negócio, em vez de interface, o que permite um envolvimento de analistas na escrita dos cenários e a

partir daí são implementados os testes em Java.

A imagem seguinte ilustra a estrutura de funcionamento de um Projeto com Cucumber:

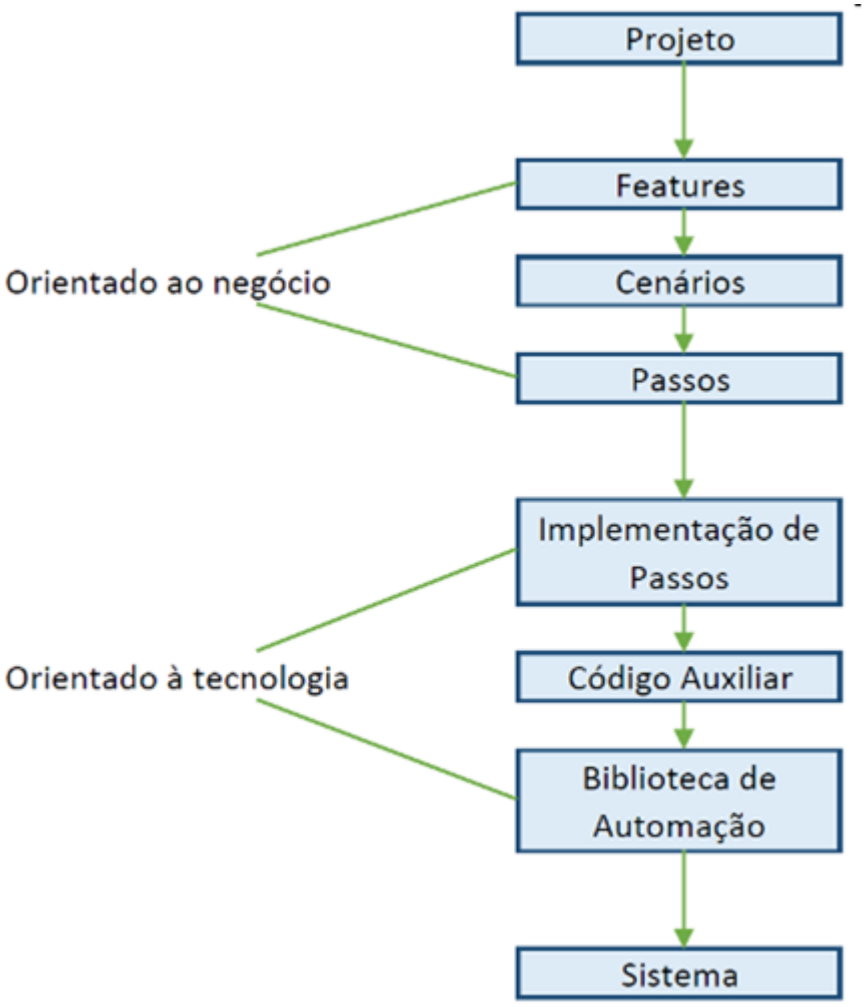


Figura 39- Estrutura de um projeto com utilização de Cucumber

Utilizando as especificações do Projeto, são escritos os Features em português num texto simples. O Cucumber lê os Features, correndo os Cenários contra o Sistema, sendo que cada Cenário é composto por uma lista de Passos.

Junto com os Features é dado ao Cucumber o *Step Definitions*, que mapeia cada Passo em código Java de forma a executar a ação por ele descrita. Para auxiliar as ações muitas vezes é usado e criado Código Auxiliar, que poderá também usar uma Biblioteca de Automação, como é o caso dos testes em Web que usam o Selenium Webdriver para interagir com o Browser.

Se o código do *Step Definitions* executar sem erros, o Cucumber executa o próximo Passo do Cenário. Se executar um Cenário até ao fim sem erros, marca-o como *Passed*, se um dos Passos falhar, marca o Cenário como *Failed* e passa para o próximo Cenário sem executar os Passos que faltavam. Enquanto decorre a execução do teste, o Cucumber vai imprimindo os resultados de cada Passo.

1.1) Gherkin

Na criação do texto descritivo do comportamento, o Cucumber utiliza uma linguagem chamada Gherkin. O Gherkin descreve de forma não técnica e de leitura fácil o comportamento do software, permitindo a analistas e managers já terem uma descrição do negócio. Esta descrição irá ser diretamente usada por developers pois serão estes ficheiros, escritos em Gherkin, que o Cucumber vai ler para executar os testes.

1.2) Feature Files

Como todas as linguagens de programação, o Gherkin também tem a sua própria sintaxe e um conjunto de regras que quem escreve tem de seguir. Todos os ficheiros Gherkin têm a extensão .feature e cada ficheiro não é nada mais que um conjunto de comportamentos que uma funcionalidade do sistema deve ter, por exemplo um Registo de Utilizador - registo_utilizador.feature.

2) Selenium

O Selenium é um conjunto de ferramentas '*open source*' e uma aplicação portátil que ajuda os '*testers'* a automatizar eficientemente aplicações '*web-based'*. Permite testar '*broken links'*, a existência de algum objeto na UI, funcionalidades Ajax, alertas, janelas, mensagens de erro, lista de opções de um '*dropdown* menu', e muito mais.

Para desenvolver os scripts de automação o utilizador tem de ter o *toolkit*, na sua versão mais atual, instalado.

2.1) Selenium Grid

A Selenium Grid é a ferramenta que distribui os testes através de múltiplas máquinas virtuais ou físicas, de forma que possamos correr vários scripts de automação em paralelo. São executadas várias instâncias dos testes do WebDriver que usam a mesma base de código, portanto o *source code* não precisa de estar presente no sistema onde os testes correm.

A grande vantagem de executar os scripts em simultâneo é que acelera drasticamente o processo de testes em diferentes browsers e diferentes plataformas de uma forma altamente eficaz, permitindo-nos obter um feedback preciso de forma praticamente imediata.

O Selenium Grid é composto por um Hub e vários Nodes.

O Hub atua como ponto central a partir do qual todos os testes vão ser *triggered* e os Nodes são as instâncias que estão conectadas ao Hub que executa os testes.

Pode haver um ou mais Nodes e estes são independentes do sistema operativo onde correm e podem usar qualquer um dos navegadores suportados.

Em seguida apresenta-se a arquitetura do Selenium Grid:

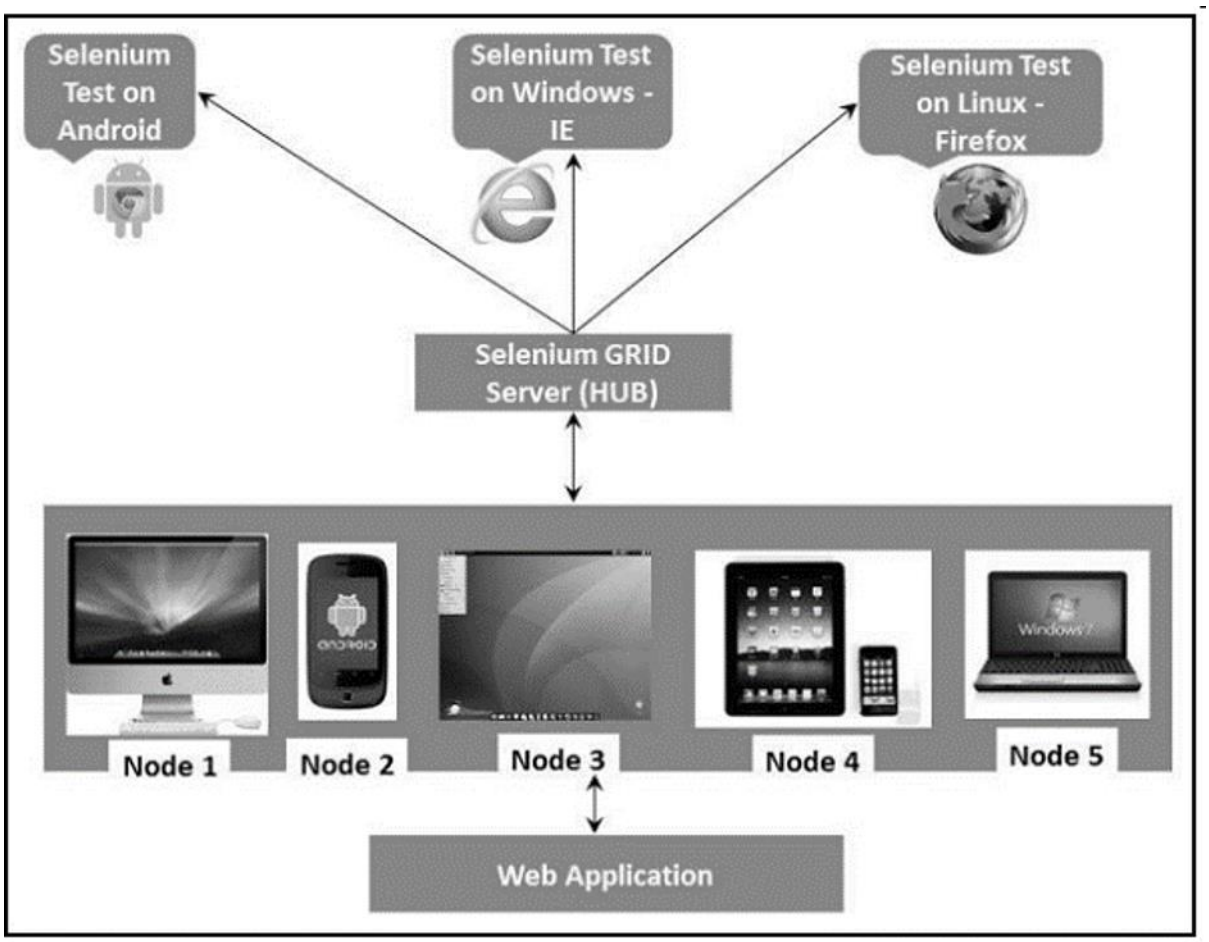


Figura 40- Diagrama de arquitetura simples do Selenium Grid

2.2) Selenium WebDriver

A Selenium WebDriver é a ferramenta mais utilizada entre todas as disponíveis, e é utilizada para auxiliar na automação de testes no browser. O WebDriver controla o browser ao nível do sistema operativo o que lhe permite interagir diretamente com o mesmo, de uma forma mais rápida, e usar o seu motor próprio para o controlar. Também permite executar testes em *headless browsers*. Por este motivo é importante perceber como funciona para que se possa tirar total partido das suas capacidades.

Pode ser usado para testes em diferentes tipos de browser, testes de elementos UI baseados em AJAX, manipulação de múltiplos quadros, múltiplas janelas do navegador, assim como pop-ups e alertas, testar a navegação em páginas complexas, e funções de navegação avançada do utilizador (ex: *drag-and-drop*).

Em seguida apresenta-se a arquitetura do Selenium WebDriver:

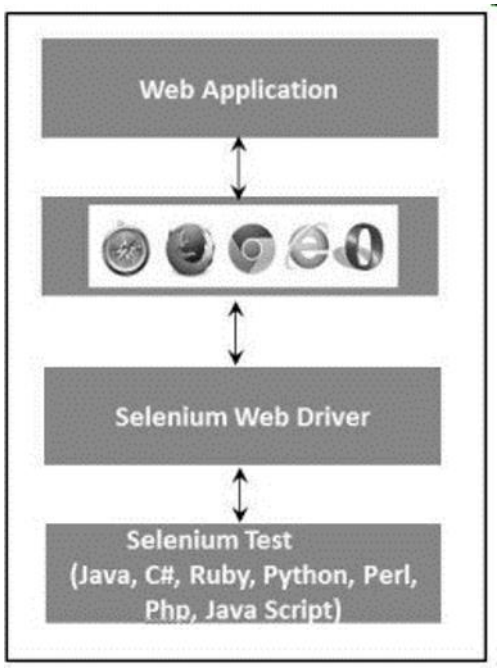


Figura 41- Diagrama de arquitetura simples do Selenium WebDriver

Apesar da arquitetura simples mostrada pela figura acima o WebDriver é constituído por uma API complexa e relativamente grande.

Para cada browser existe um driver e um conjunto de configurações específicas para ser possível interagir com ele. Esta abstração do browser e outras operações já estão implementadas pela FRAT.

4.2.1.2 PLANO DE FORMAÇÃO PARA UTILIZAÇÃO DA FRAT

De modo a obter mais conhecimentos teóricos e práticos sobre automação e aprender a utilizar a FRAT para a automação de testes, foi dada uma formação para automação de testes.

O processo de formação, passa por um conjunto de aquisição de conhecimentos teóricos e práticos sobre automação de testes, com o objetivo de que no final da formação o formando esteja habilitado a desenvolver e manter testes de automação para qualquer projeto.

4.2.1.3 TESTES AUTOMATIZADOS NO ÂMBITO DOS PROJETOS DO CLIENTE 2

A automação de testes é efetuada através de software especializado, que executa um conjunto de testes (*Test Scenario*). De um ponto de vista abstrato, espera-se conseguir inserir dados no sistema que está a ser testado, comparar os resultados esperados com os resultados reais (*Assertion*), verificar que os resultados reais foram persistidos e, por fim, gerar relatórios detalhados sobre a execução.

As vantagens pretendidas com este processo são que pelo facto de os ciclos de

desenvolvimento de uma aplicação requererem a execução do mesmo *Test Scenario* repetidamente. Ao automatizar, é possível executar um conjunto de testes de forma rápida e consistente, reduzindo a margem de erro e sem que seja necessária intervenção humana.

De modo a seguir todo o processo da automatização de testes, segundo boas práticas e para todas as equipas terem um trabalho uniforme dentro do que é possível, é seguido o seguinte modelo com as respetivas fases:

Estratégia de Testes Automatizados

- Ter acompanhamento das atividades que estão a ser realizadas pela equipa de testes automatizados utilizando o Kanban;
- Incentivar a comunicação entre as equipas para que esse processo de entrega do código de teste siga naturalmente;
- Comunicar a entrega da *release* à equipa de testes automáticos com antecedência de 2 dias;
- Rever o âmbito de teste que será entregue com base nos critérios de identificação dos cenários de teste;
- Registar uma tarefa no bugzila associada à entrega para entrega do código dos testes automáticos.

Planeamento & Análise

• Funcionalidades priorizadas para automatização são encaminhadas por e-mail;
• Análise da documentação (UC e Regra de negócio);
• Mapeamento dos fluxos de negócio que serão implementados no âmbito de testes de regressão;
• Mapeamento do conjunto de dados (*query*);
• Definição da estratégia de implementação;
• Construção do Feature File.

Construção do Script & Execução

• Construção dos scripts automatizados;
• Reuso de scripts (métodos e componentes).

Execução

- Execução dos testes;
- ROTA - Envio do relatório operacional de testes Automatizados;
    - O ROTA é o Relatório Operacional de Testes Automáticos e tem por objetivo passar as seguintes informações:
    - Actividades realizadas;
    - Próximas actividades;
        - Constragimentos;
        - Catálogo de testes (descrição dos cenários de teste com as suas respetivas variantes de acordo com a funcionalidade);

        - Estado da construção do script;
        - Resultado de execução (Separador execução);
        - Evidências de teste (Link CukeReports);
        - Report de bugs.

        Benefícios:

        - Apresenta uma visão de catálogo dos testes (âmbito de testes automatizados);
        - Permite uma visão rápida de como está o estado de construção e execução do script.
- Re-teste;
- Report Bugs;
- Reteste Bugs;
- Alinhamento com desenvolvimento.

Manutenção

• Manutenções dos scripts nas possíveis mudanças, incremento do âmbito;

• Manter ativos os scripts já desenvolvidos;

• Análise/correção das críticas reportadas pela esteira de CI.

Execução Pós *Deploy*

• Execução dos testes – Jenkins;

• Apresentação do resultado de execução pelo Cukereports.

Entrega do script de teste

É imprescindível que haja a comunicação das equipas de desenvolvimento e testes automatizados, para que o código de teste siga atualizado na entrega. Quando essa comunicação não ocorre facilmente encontramos os seguintes problemas:

- Módulos de análise e de testes comentados no pom.xml do projeto, ou seja, o código de teste não é entregue;
- Código de teste desatualizado em relação a novas funcionalidades;
- Código de teste desatualizado em relação ao âmbito *regressive.*

Entregáveis

Atualmente para os projetos os entregáveis são:

- Feature file;
- Código de teste (Script de testes automáticos);
- ROTA - Catálogo de teste;
- Evidências de teste (Cuke Report);
- Script de teste incluído na entrega.

4.2.2. Resultados e conclusões

Após começar a executar os testes no Projeto 1, a título de curiosidade, foi feita uma verificação e colocação da questão, por que motivo não existiam testes automatizados para aquele projeto, pelo que foi concluído que devido a ter sido uma migração de uma aplicação local, para uma aplicação em web browser, não é possível a nenhuma ferramenta de automatização inspecionar elementos para poder fazer um mapeamento dos mesmos. É então esse o motivo pelo qual todos os testes funcionais deste projeto terem de ser executados de forma manual.

O processo, logicamente é um pouco diferente do de automatizar testes, porém é menos complexo, uma vez que existe no início de todas as entregas um ficheiro que é o ROT (igual ao ROTA dos testes automáticos, mas adaptado com as informações dos testes manuais), este ficheiro é nada mais que um ficheiro Excel onde são preenchidos vários dados relacionados com a entrega e com o que vai ser testado nessa mesma entrega.

A parte mais desvantajosa do processo de executar testes manuais em relação aos automatizados é precisamente a documentação, a execução demorada nos vários ambientes, e a recolha de evidências. Em primeiro lugar, é correto e legítimo ficar tudo documentado, porém existem ferramentas mais avançadas neste momento que o permitem de uma forma mais simples, ora por cada teste declarado no ROT, seja de bugs ou de regressão, por cada teste tem que existir um ficheiro Word onde é descrita toda a informação relacionada com o teste, desde a sua descrição, o ambiente, no caso de ser um bug a descrição do bug, o resultado esperado e o obtido do teste, e por fim colocar os prints relativos às evidências.

Visto assim, parece um processo superorganizado, e não se pode dizer que não seja, porém não é minimamente prático e provoca uma enchente de ficheiros e ocupação de memória, pelo que tem sempre de existir um tempo extra despendido para o preenchimento correto de toda a informação.

O facto de um teste manual ser mais demorado, também não é uma ajuda, pois a atenção tem de ser redobrada e por cada passo do ciclo de teste as evidências (prints) também têm de ser recolhidas e editadas de forma manual. Este facto, num projeto em que apenas se testa em dois ambientes no mínimo e tenhamos poucos testes por entrega e entregas num período regular espaçado de semanas, não é muito trabalhoso, aborrecido ou impossível.

Porém estamos a falar de um projeto que já conta com praticamente 200 testes de regressão, devido a ser uma aplicação muito extensa e com muitas funcionalidades, e por cada entrega tem sempre 5 ou mais bugs associados, pelo que testar praticamente 200 testes por 4 ambientes (DEV-PLT-FRM-QUA), em PRE e Produção existe um *tester* por parte do cliente, recolher evidências, preencher a respetiva documentação , com uma *guideline* apertada e sendo que existem alturas em que existem 2 entregas por semana, torna-se humanamente impossível, pelo que na maior parte das situações apenas são efetuados os testes dos bugs em todos os ambientes e os testes de regressão apenas efetuados no ambiente de Qualidade, devido a ser o ambiente onde é efetuada a acreditação.

Relativamente ao Projeto 2, aquando alguma entrega, são executados automaticamente os

testes de regressão do projeto, de modo a acompanhar os testes manuais dos restantes colegas de equipa.

Durante este período de estágio, por algumas vezes, devido a atualizações da FRAT foram efetuadas algumas reestruturações e *refactoring* do código já existente, houve também a necessidade do desenvolvimento de mais 2 testes de regressão que foram desenvolvidos no espaço de duas semanas e estão a ser executados normalmente sempre que necessário. Existe o cuidado da execução destes, diariamente, por precaução e para estarem sempre atualizados e prontos nos momentos de acreditação.

Passando para o Projeto 3, é um projeto que até à data do término do estágio não está concluído, ainda se encontra numa fase de desenvolvimento e desenvolvimento dos testes automáticos muito precoce.

Porém foi um projeto que foi acompanhado desde o início, houve uma forte colaboração entre todos os elementos da equipa para colaborar na análise de requisitos e no processo de design. No que diz respeito ao desenvolvimento dos testes, foram documentados todos os planos de teste e a escrita dos Feature Files, estes eram uma tarefa exclusiva da analista funcional do projeto, mas devido a um fluxo de trabalho elevado por parte desta e a existência de *skills* por parte da *tester* de automação para este projeto e o acompanhamento desde o começo do projeto, excecionalmente esta tarefa ficou delegada ao *tester*. O desenvolvimento dos scripts de teste em si, foram construídos com base naquilo que é a estrutura proposta pela FRAT, utilizando todos os seu módulos e métodos necessários conforme as funcionalidades a serem testadas. Neste momento, apenas estão automatizados os testes de regressão definidos até à data pelo projeto.

Em suma, é obtida uma experiência satisfatória de todos os projetos, foi permitida ter uma perspetiva diferente em todos e poder fazer uma comparação da experiência de atuar como *tester* manual e *tester* de automação.

O impacto da automatização de testes é extremamente positivo, foi uma mais-valia em todas as execuções de teste e facilitou o trabalho no que diz respeito ao tempo despendido.

A apontar, como sugestão, nos processos de acreditação, projetos em que existem testes manuais existe um processo de acreditação formal em que é enviado um email com todos os detalhes em como os testes passaram e podem seguir para produção aquando da autorização do gestor de projeto. Porém se o projeto apenas tiver testes automatizados este processo não existe e apenas é perguntado via teams se os testes foram executados com sucesso e conforme a resposta prosseguem. No caso de uma execução mista em que existem testes manuais e automáticos, a acreditação apenas é feita baseada nos testes manuais sendo que apenas é questionado se os testes automáticos foram executados com sucesso e os resultados não constam no email de acreditação. Seria mais justo e correto existir um processo semelhante para os testes automatizados.

No que toca a tecnologia, não era despropositada a substituição do ROTA e ROT por uma integração com uma plataforma como o JIRA, por exemplo. Atualmente e com a existência dessas

ferramentas que automatizam muitos dos processos de documentação e de agilidade entre ferramentas, não existe necessidade de estes processos serem feitos de forma manual e serem preenchidos ficheiros de Word e Excel. Logicamente isto são processos que envolvem custos, mas são investimentos proveitosos e proporcionam autonomia e flexibilidade aos profissionais.

4.3. TestPath vs FRAT

Uma vez que houve uma experiência em dois clientes diferentes com duas ferramentas de automação também elas diferentes. É interessante fazer uma comparação entre as duas, verificar qual das duas a mais vantajosa, a mais intuitiva, a mais versátil e em que pontos estas diferem.

Na seguinte tabela, observamos as principais características e diferenças entre as duas ferramentas de automação utilizadas:

Tabela 3- Diferenças entre o TestPath e FRAT

	TestPath	FRAT
Skills de programação		X
Integração com várias tecnologias	X	X
Facilitador rápido de atualizações e novas versões	X	X
Criação de automação de testes fácil e manutenção	X	X
Reutilização de componentes	X	X
Gravação de vídeo em todas as etapas	X	
Evidências em Relatórios	X	
Visibilidade total, rastreabilidade e auditabilidade de cada execução da etapa de teste	X	X
Ferramenta colaborativa para testes	X	X
Equipas de automação *Onshore-Neasrhore-Offshore*	X	X
Informações de teste e resultados em *Dashboards*	X	
Acompanhamento e monitorização em tempo real de qualquer lugar	X	X
Agentes de Execução Múltipla	X	
Capacidade para lidar com vários projetos simultâneos	X	X
Execução de cenários de teste *End2End*	X	X
Abordagem geral para testes de automação	X	
Curva reduzida de aprendizagem	X	
Rapidez na velocidade de implementação	X	

Conforme observado na Tabela 3 conseguimos chegar à conclusão de que se tivéssemos de optar por uma das ferramentas optaríamos pelo TP. Uma vez que, de forma geral, acabaria por resolver a maior parte dos problemas existentes e anteriormente especificados no âmbito desta dissertação. Principalmente, pelo facto de para se trabalhar com o TP não serem necessárias *skills* de programação e o desenvolvimento dos testes ser mais intuitivo. Também o facto de o TP possuir uma interface e a maior parte das funcionalidades estarem nele concentradas ajuda a que não exista dispersão de tempo a andar entre aplicações.

A FRAT apenas pode ser mais vantajosa para quem possuir *skills* e gostar muito de programação, porque tirando esse ponto, o TP é uma ferramenta mais independente e completa.

4.4. Estudo de mercado

Como temos vindo a perceber ao longo deste estudo, um dos principais problemas dentro do mundo de QA é precisamente a falta de profissionais qualificados, já foi anteriormente referido que é uma área recente dentro do mundo da informática e que a falta de profissionais não é algo exclusivo desta área, é um problema geral. Porém, ao contrário das outras áreas na informática, esta falta tem sido colmatada por profissionais cuja sua área de formação nem sequer está muito perto das áreas das TI.

Esta questão pode colocar-nos algumas dúvidas como se de facto existe uma procura acentuada e adequada no mercado, se a nível salarial é justo ou vantajoso enveredar por este ramo e se a nível académico, será que a formação é adequada ou fornece conhecimento suficiente para e ou sobre esta área.

Em torno destes pontos, as próximas secções destinam-se a resultados de pesquisa que pretendem dar resposta a todas estas questões.

4.4.1. Procura no mercado

Começando pela oferta e propostas existentes no mercado português, uma vez que este estudo apenas se restringe à realidade a nível nacional, foi investigado através de plataformas, como o LinkedIn e semelhantes, que tipo de propostas de emprego existem dentro da área de QA, para que tipo de cargos existem mais propostas, que tipo de *testers* são necessários e se comparativamente com outras áreas dentro das TI existe mais procura.

Das várias plataformas existentes foram selecionadas 3:

- LinkedIn, devido a ser uma das plataformas mais utilizadas no recrutamento de profissionais e de entre as várias redes sociais relacionadas com a procura/oferta de emprego, ser a mais completa e popular;
- Glassdoor, é um site americano onde funcionários atuais e ex-funcionários avaliam anonimamente as empresas, mas entre as muitas funcionalidades, uma delas é a procura de

anúncios de emprego;

- Indeed, é um motor de busca de empregos criado nos Estados Unidos, que funciona como *searcher* vertical.

A escolha resumiu-se a estas 3 plataformas, não só por serem bastante populares, mas porque todas as restantes serem muito semelhantes.

A pesquisa começou pelo LinkedIn, onde foi procurado de uma forma mais geral pelos empregos como "QA *tester*" com a localização em Portugal (Figura 42), a nível de resultados obtivemos cerca de 270 propostas, sendo que estavam abrangidas diversas áreas dentro de QA, como *testers* funcionais, *testers* manuais, *testers* de automação, engenheiros de QA ou software *testers*. Pelo que foi decidido filtrar a própria pesquisa precisamente por esses termos.

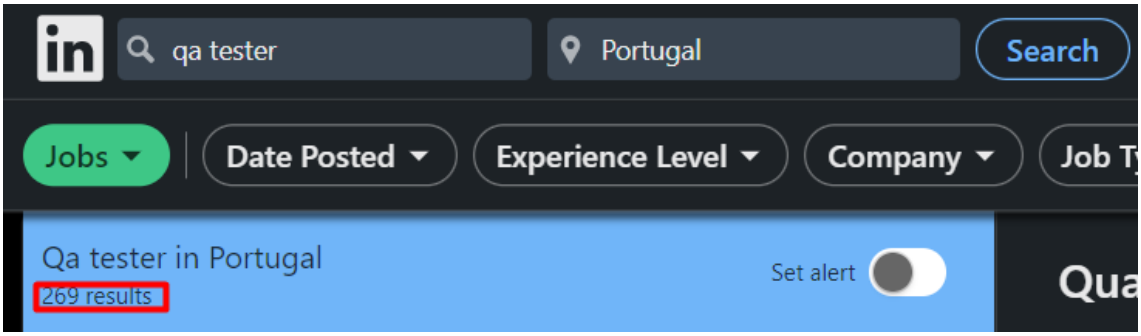


Figura 42- Resultados da pesquisa feita pelo termo "qa tester"

Pesquisando por "QA manual *tester*" (Figura 43), obtemos 247 resultados sendo que muitas das propostas devolvidas incluem propostas que não são para *testers* manuais, mas um pouco do que foram os resultados à procura anterior. Das propostas em que efetivamente era solicitado um *tester* manual, os requisitos pedidos eram os seguintes:

- Experiência demonstrada em testes manuais para a Web/*backend*;
- Forte conhecimento de metodologias de teste de software;
- Experiência com SQL;
- Detalhista com capacidade de ver a imagem maior;
- Capacidade de reproduzir problemas de funcionalidade complexos;
- Revisão e análise dos requisitos do projeto para clareza, compreensão e testabilidade;
- Desenvolvimento de planos de teste do sistema e casos de teste manuais;
- Execução dos casos de teste e scripts de teste;
- Fornecer entrada ou criar artefactos de teste para incluir, mas não se limitar a: Planos de Teste, Resultados de Teste, Métricas e Relatórios de Defeitos.

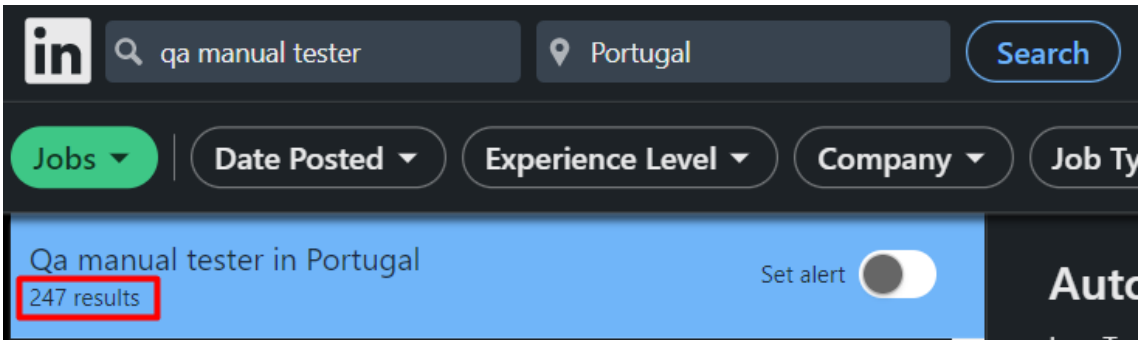


Figura 43- Resultados da pesquisa feita pelo termo “qa manual tester”

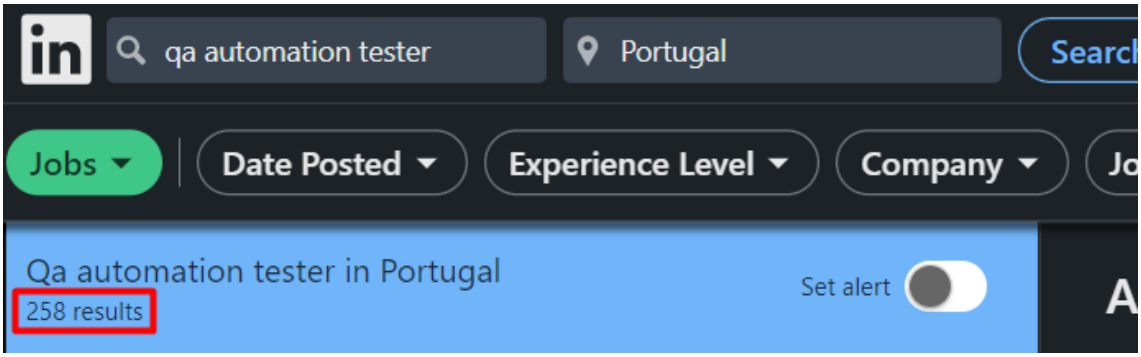


Figura 44- Resultados da pesquisa feita pelo termo “qa automation tester”

Mas se o termo de pesquisa for “QA *automation tester*” (Figura 44), aumenta significativamente o número de resultados e são claramente a função mais solicitada dentro dos vários elementos do QA, porém os requisitos mantêm grande parte dos anteriormente referidos para *testers* manuais, no entanto acrescenta mais alguns que certamente acarretam mais alguma dificuldade como:

- Produção/ Desenvolvimento de scripts de teste; - Execução de scripts de teste; - Possibilidade de automação de casos de teste em estágio posterior;
- Experiência /Conhecimento em Java, JavaScript, Python ou C#;
- Experiência testes em API;
- Experiência na utilização de ferramentas de automatização de testes, por exemplo Selenium, Cypress ou outra;
- Experiência na utilização de ferramentas de testes, por exemplo, Cucumber, Azure Test Plan, ou outra;
- Experiência na utilização de ferramentas de DevOps, por exemplo Git, Azure DevOps, ou outra;
- Compreensão profunda das melhores práticas na criação de scripts de teste de sistema, execução de teste e relatório/rastreamento de defeitos;
- Experiência em testes de aplicações web em diferentes plataformas e navegadores
- Fortes habilidades de solução de problemas e análise de causa raiz;
- Pensamento crítico e capacidade de colaborar com uma equipa na resolução de problemas urgentes;
- Compreensão das seguintes tecnologias Web: HTTP(S), S/FTP, Web Services (SOAP,

RESTful);

- Licenciatura ou Mestrado em Ciência da Computação, Engenharia ou áreas afins.

Ou seja, são pedidos requisitos que envolvem *skills* muito mais técnicas, como não poderia deixar de ser.

Passando às duas outras plataformas selecionadas (Figura 45), foram feitas exatamente as mesmas pesquisas com os mesmos termos, porém os resultados não foram muito diferentes, pelo que a maior parte das propostas, senão todas as propostas encontradas, foram as já encontradas quando feitas as pesquisas no LinkedIn, logicamente podem diferir num número de resultados ou outro, mas na essência bastava ter sido feita a primeira análise.

Figura 45- Resultados das pesquisas feitas pelo termo “qa automation tester” nas plataformas Glassdoor e Indeed

Para além dos requisitos acima verificados, percebe-se que a maioria das propostas são para a área metropolitana de Lisboa e as restantes para o Porto, se bem que a localização no pós-pandemia deixou der ser um entrave a partir do momento que quase todas as propostas de emprego prometem um sistema híbrido ou completamente remoto.

De forma a comparar e por mera curiosidade foi pesquisada a necessidade de *developers* independentemente da área em Portugal (Figura 46) e a diferença é abismal sendo que a falta destes profissionais é muito superior, embora nestas propostas estivessem algumas relacionadas com testes automáticos. Certamente é uma comparação “injusta” a partir do momento em que dentro da área de desenvolvimento existem muitas subáreas e muitas outras *skills*, mas muitas dessas *skills* são comuns com as que são pedidas para automatizar testes, ora não fossem os *testers* de automação *developers* também, contradizendo a ideia errada que muitas pessoas têm.

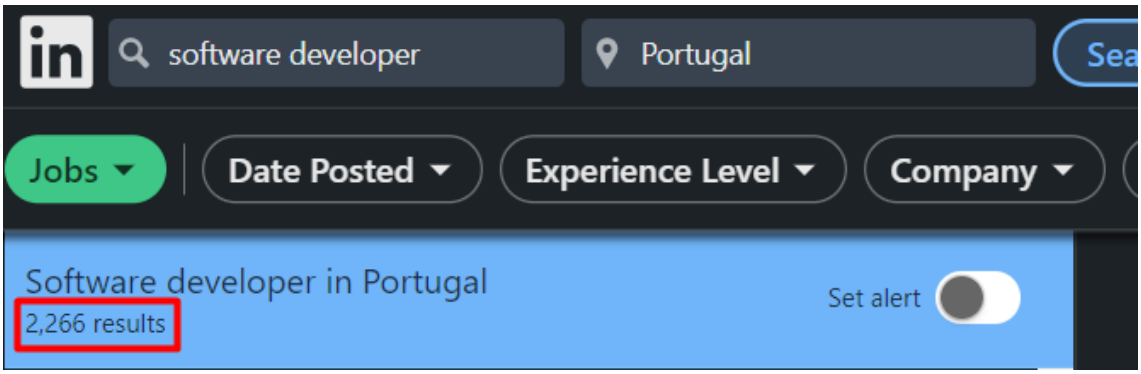


Figura 46- Resultados da pesquisa feita pelo termo “software *developer*”

Em suma, conseguimos concluir que efetivamente o motivo de falta de profissionais na área de QA não se deve à falta de procura e oportunidades.

4.4.2. Questões salariais

Após a primeira análise e concluirmos que existe muita procura na área de QA, continua a dúvida do porquê de isto acontecer, apesar da falta de profissionais qualificados, mas estes existem e a área da Qualidade de Software não é uma primeira escolha. Uma das possíveis respostas poderá ser a questão salarial.

Sabemos que, apesar de a maioria das pessoas colocar o salário como ponto principal para definir um bom emprego, não está correto, porém é um ponto que não deixa de ser importante, uma vez que os trabalhadores são pessoas e como tal têm contas para pagar e despesas das quais não podem fugir. Sendo Portugal um país onde os salários são dos mais baixos da Europa e o estilo de vida que a maioria dos salários proporciona não é propriamente confortável, talvez seja interessante analisar quanto recebe, em média, um profissional de QA em Portugal.

Para fazer essa análise, recorreu-se ao site Glassdoor, como foi feito anteriormente para avaliar algumas propostas de emprego e também no site talent.com. Importante informar que estes sites, assim como a grande maioria dos sites deste género faz os cálculos baseados em pequenas amostras, ou seja, baseado apenas nos valores e dados que funcionários de várias empresas disponibilizam anonimamente.

Começou-se por pesquisar quanto seria o salário médio de um *Tester* pelo que os resultados (Figura 47) foram os seguintes:

€ 1,200/ Mês - Brutos
Baseado em 11 salários

O salário médio de um qa *tester* em Portugal é de € 14,400 anuais ou € 7.38 por hora. As posições de nível inicial começam nos € 6,780 anuais, enquanto os trabalhadores mais experientes podem chegar a ganhar € 35,250 anuais.

Figura 47- Gráfico do intervalo de valores salariais para um QA Tester

Postos estes valores optou-se por fazer uma comparação com a média de salários de profissionais de outras áreas, nomeadamente desenvolvimento de software (Figura 48). Os resultados foram os seguintes:

€ 2,167/ Mês - Brutos
Baseado em 190 salários

O salário médio de software *developer* em Portugal é de € 26,000 anuais ou € 13.33 por hora. As posições de nível inicial começam nos € 18,250 anuais, enquanto os trabalhadores mais experientes podem chegar a ganhar € 50,000 anuais.

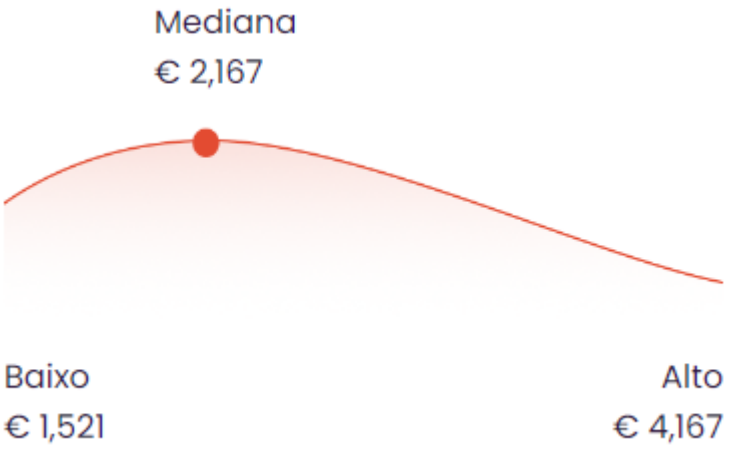


Figura 48- Gráfico do intervalo de valores salariais para um Software Developer

Logicamente estes valores são calculados com dados muito gerais e podem não ter em consideração fatores essenciais como os descontos aplicados, a formação, os anos de experiência e dados demográficos, por exemplo. Já para não falar que os números das amostras utilizados não são propriamente equilibrados.

Porém conseguimos tirar uma ideia geral de que o salário médio de um profissional de QA em Portugal, comparativamente à média nacional, não é um mau salário. Mas se formos comparar com outras áreas como a de desenvolvimento de software, a diferença ainda é considerável, pelo que se um profissional na área das TI com uma boa formação e *skills*, tiver como prioridade o salário, obviamente não envereda pela área de qualidade.

4.4.3. Formação Académica

O ponto da formação académica surge devido ao facto de, se formos ter com uma equipa de QA numa qualquer empresa e questionarmos cada profissional a nível da sua formação académica, obtemos que a grande maioria não teve a sua formação dentro das áreas das tecnologias de informação.

Por exemplo, temos profissionais formados nas áreas de Matemática ou Gestão empresarial, e até mesmo em áreas totalmente discrepantes como Direito ou Educação Básica, mas poucos profissionais com licenciaturas/mestrados ou semelhantes em Engenharia Informática ou similar.

O que nos leva a questionar o porquê, de dentro dos profissionais supostamente qualificados e relacionados com a área, não existir o interesse para desempenhar este papel. Declinando, por vezes, propostas, em detrimento de ir apenas para desenvolvimento, quando na verdade, na maioria dos casos para elaborar testes é necessário desenvolver/programar.

Colocando hipóteses de propostas de mercado e questões salariais, anteriormente analisadas, de parte. Resta-nos averiguar se isto acontece devido a alguma lacuna a nível da formação académica. Como tal, foi elaborada uma pesquisa que tem como amostra todos os estabelecimentos de ensino superior, quer do sector público como do sector privado, ensino universitário ou politécnico, dos distritos de Setúbal, Lisboa, Braga e Porto. Com o objetivo de avaliar os planos de estudos e respetivos conteúdos programáticos, dos cursos relacionados com as Tecnologias de informação, abrangendo licenciaturas e mestrados. De modo a concluir que destaque tem o QA nestes conteúdos e o valor atribuído à qualidade do software produzido, mesmo que de forma académica.

No distrito de Setúbal, como indicado na Tabela 4, encontramos 3 estabelecimentos de ensino, 2 do sector público e apenas 1 do setor privado, onde foram avaliados os planos de estudos de 9 cursos diferentes entre os quais 6 licenciaturas e 3 mestrados.

Tabela 4- Cursos da área de TI nos estabelecimentos de Ensino Superior em Setúbal

<table>
<tr><th colspan="3">Ensino Superior no Distrito de Setúbal</th></tr>
<tr><td colspan="2">Estabelecimentos de Ensino Público</td><td>Estabelecimentos de Ensino Privado</td></tr>
<tr><td colspan="2">Universidade Nova de Lisboa</td><td>Instituto Politécnico Jean Piaget do Sul</td></tr>
<tr><td colspan="2">Faculdade de Ciências e Tecnologia</td><td>Escola Superior de Tecnologia e Gestão Jean Piaget</td></tr>
<tr><td colspan="2">Licenciatura em Engenharia Informática
Mestrado em Engenharia Informática
Licenciatura em Engenharia Eletrotécnica e de Computadores
Mestrado em Engenharia Eletrotécnica e de Computadores</td><td>Licenciatura em Engenharia Informática</td></tr>
<tr><td colspan="2">Instituto Politécnico de Setúbal</td><td></td></tr>
<tr><td>Escola Superior de Tecnologia de Setúbal</td><td>Escola Superior de Ciências Empresariais</td><td></td></tr>
<tr><td>Licenciatura em Engenharia Eletrotécnica e de Computadores
Licenciatura em Engenharia Informática
Mestrado em Engenaria de Software</td><td>Licenciatura em Gestão de Sistemas de Informação</td><td></td></tr>
</table>

Ao analisar todos os planos de estudos dos cursos representados em cima, a conclusão era o já expectável, apenas dois destes planos possuem uma unidade curricular dedicada à qualidade de software e cujos propósitos são:

- Compreender os conceitos de qualidade de software;
- Conhecer e aplicar as técnicas de revisão de software;
- Compreender e pôr em prática as normas e abordagens de Software Quality Assurance (SQA);
- Conhecer e utilizar um plano de SQA;
- Compreender e aplicar as estratégias de testes de software nos vários ambientes existentes;
- Saber aplicar os testes em aplicações convencionais, em aplicações orientadas a objetos e em aplicações Web;
- Compreender a modelação formal e compreender as linguagens de especificação formal;
- Compreender e utilizar as diversas métricas de produto nas fases do desenvolvimento de software.

Porém, estas duas UC's são lecionadas a nível de mestrado, nomeadamente nos mestrados de Engenharia Informática (Figura 49) e Engenharia de Software (Figura 50), da UNL-FCT e do IPS- ESTS,

respetivamente. Sendo que a UC de Qualidade de Software no Mestrado de Engenharia Informática é opcional, já a mesma UC no Mestrado de Engenharia de Software é obrigatória.

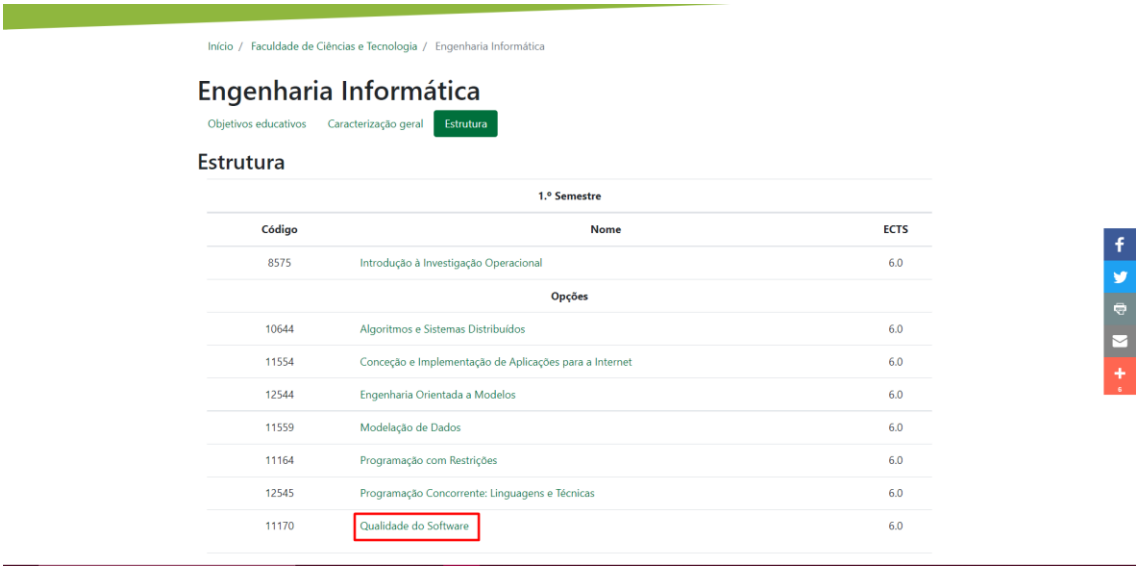


Figura 49- Plano curricular do Mestrado em Engenharia Informática da UNL-FCT

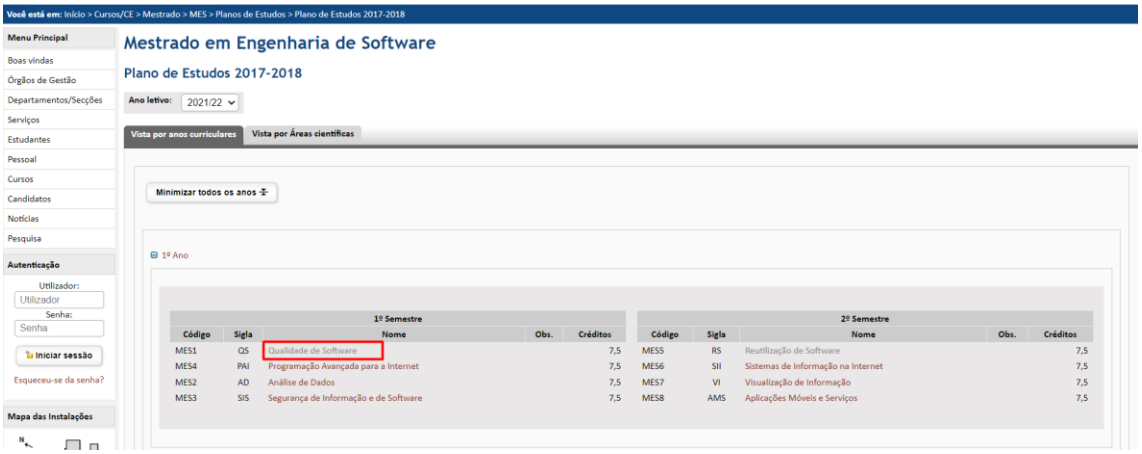


Figura 50- Plano curricular do Mestrado em Engenharia de Software do IPS-ESTS

No que diz respeito aos restantes conteúdos programáticos, neste caso, das licenciaturas apesar de não existirem UC's exclusivamente dedicadas a esta temática, nas UC's de Programação Orientada para objetos e Engenharia de Software, pertencentes às licenciaturas de Engenharia Informática da UNL-FCT e do IPS-ESTS, existem conteúdos dedicados à qualidade de software e desenvolvimento de testes.

É importante frisar que estes resultados são baseados nos dados a que foi possível ter acesso, através dos meios oficiais das instituições em questão, podem não estar totalmente detalhados e pouco atualizados, e, no entanto, existir alguma relevância na qualidade do software produzido, em especial, nas UC's com componente prática/laboratorial. Sendo que de muitas UC's não se conseguiu encontrar qualquer detalhe quanto aos conteúdos programáticos, apenas a sua denominação.

Fazendo exatamente a mesma análise, mas para o distrito de Lisboa, a amostra aumenta significativamente, como podemos observar na Tabela 5. Pois existem mais estabelecimentos de ensino, quer dos setores público e privado, mas também existe uma maior variedade e quantidade de cursos dentro da área das TI.

No que diz respeito a resultados desta pesquisa, apesar de a amostra ser maior, não são encontradas grandes diferenças em relação aos resultados obtidos para o distrito de Setúbal.

Tabela 5- Cursos da área de TI nos estabelecimentos de Ensino Superior em Lisboa

<table>
<tr><th colspan="5">Ensino Superior no Distrito de Lisboa</th></tr>
<tr><td colspan="4">Estabelecimentos de Ensino Público</td><td>Estabelecimentos de Ensino Privado</td></tr>
<tr><td colspan="4">Universidade de Lisboa</td><td>Universidade Autónoma de Lisboa Luís de Camões</td></tr>
<tr><td>Instituto Superior Técnico</td><td colspan="2">Instituto Superior Técnico (Tagus Park)</td><td>Faculdade de Ciências</td><td></td></tr>
<tr><td>• Licenciatura em Engenharia Eletrotécnica e de Computadores
• Licenciatura em Engenharia Informática e de Computadores
• Mestrado Bolonha em Engenharia Informática e de Computadores</td><td colspan="2">• Licenciatura em Engenharia Informática e de Computadores
• Licenciatura em Engenharia de Telecomunicações e Informática
• Mestrado Bolonha em Engenharia Informática e de Computadores</td><td>• Licenciatura em Engenharia Informática
• Licenciatura em Tecnologias de Informação
• Mestrado em Engenharia Informática
• Mestrado em Informática
• Mestrado em Segurança Informática</td><td>• Licenciatura em Engenharia Informática
• Licenciatura em Informática de Gestão</td></tr>
<tr><td colspan="3">ISCTE</td><td colspan="2" rowspan="2">Universidade Europeia</td></tr>
<tr><td colspan="2">Instituto Universitário de Lisboa</td><td>Instituto Universitário de Lisboa (Sintra)</td></tr>
<tr><td colspan="2">• Licenciatura em Engenharia Informática
• Licenciatura em Engenharia de Telecomunicações e Informática
• Licenciatura em Informática e Gestão de Empresas
• Mestrado em Engenharia Informática
• Mestrado em Engenharia de Telecomunicações e Informática</td><td>• Licenciatura em Desenvolvimento de Software e Aplicações
• Licenciatura em Tecnologias Digitais e Inteligência Artificial
• Licenciatura em Tecnologias Digitais e Segurança</td><td>• Licenciatura em Engenharia Informática
• Licenciatura em Informática de Gestão</td></tr>
<tr><td colspan="4">Universidade Nova de Lisboa</td><td>Universidade Lusíada -</td></tr>
<tr><td colspan="4">Instituto Superior de Estatística e Gestão de Informação</td><td>Centro Universitário Lusíada - Lisboa</td></tr>
<tr><td colspan="4">• Licenciatura em Ciência de Dados
• Licenciatura em Gestão de Informação
• Licenciatura em Sistemas e Tecnologias de Informação</td><td>• Licenciatura em Engenharia Informática</td></tr>
</table>

Ensino Superior no Distrito de Lisboa	
Estabelecimentos de Ensino Público	Estabelecimentos de Ensino Privado
Instituto Politécnico de Lisboa	Universidade Lusófona de Humanidades e Tecnologias
Instituto Superior de Engenharia de Lisboa	
• Licenciatura em Engenharia Eletrónica e Telecomunicações e de Computadores • Licenciatura em Engenharia Informática e de Computadores • Licenciatura em Engenharia Informática e Multimédia • Licenciatura em Engenharia Informática, Redes e Telecomunicações • Mestrado em Engenharia Informática e de Computadores • Mestrado em Engenharia Informática e Multimédia	• Licenciatura em Informática de Gestão • Licenciatura em Engenharia Informática, Redes e Telecomunicações • Licenciatura em Engenharia Informática • Mestrado em Engenharia Informática e Sistemas de Informação
Escola Superior Náutica Infante D. Henrique	Atlântica - Instituto Universitário
• Licenciatura em Engenharia Informática e de Computadores	• Licenciatura em Gestão de Sistemas e Computação
	Instituto Superior de Tecnologias Avançadas de Lisboa
	• Licenciatura em Engenharia Informática

Começando pelas instituições de ensino privado, onde quer nas Licenciaturas quer nos Mestrados não foi possível retirar grandes resultados uma vez que ou não era possível obter os planos de estudos, ou os planos de estudo apenas apresentavam a designação da unidade curricular e não apresentavam os conteúdos programáticos ao detalhe. Posto isto, fica em aberto se Unidades Curriculares como Gestão da Qualidade com carácter opcional, pertencente à Licenciatura em Gestão de Sistemas e Computação da universidade Atlântica, ou a UC de Engenharia de Requisitos e Testes, pertencente às Licenciaturas em Informática de Gestão e Engenharia Informática da Universidade Lusófona estarão relacionadas com aquilo que é pretendido neste estudo, uma vez que não conseguimos ver os conteúdos programáticos.

As únicas universidades onde é possível ver os conteúdos programáticos são a Universidade Lusíada e o Instituto Superior de Tecnologias Avançadas de Lisboa, sendo que a única UC que contém matéria relacionada com qualidade de software e testes é Engenharia de Software da Licenciatura de Engenharia Informática no ISTEC.

Nas instituições de ensino público, foi possível retirar mais informação apesar de que no caso da Escola Superior Náutica Infante D. Henrique também só foi possível ter acesso ao nome das disciplinas lecionadas.

No caso das Licenciaturas e Mestrados pertencentes ao Instituto Politécnico de Lisboa e Universidade Nova de Lisboa, apenas algumas UC's de programação e as UC's de Engenharia de Software referem como conteúdo a testagem aos programas desenvolvidos.

Dos cursos lecionados no ISCTE, apenas algumas UC's de programação, de Desenvolvimento

Ágil de Software e Engenharia de Software possuem conteúdos relacionados com Metodologias de teste e controlo da qualidade do software, Testes unitários, de integração (uso e desenho), funcionais, de carga, e de cobertura de código, métricas de qualidade e melhoria do produto.

Dentro da Universidade de Lisboa, na faculdade de ciências, no que diz respeito aos mestrados analisados, existe a UC de Verificação e Validação de Software que contém as matérias relacionadas com qualidade de software, porém esta é sempre uma UC opcional. Passando para as Licenciaturas, a UC de Projeto de Sistemas de Informação da Licenciatura de Engenharia Informática e a UC Planeamento e Gestão de Projeto da Licenciatura em Tecnologias de Informação são as únicas cujos conteúdos possuem um ponto relacionado com Testes de Software.

Por fim, falando no Instituto Superior Técnico, no que diz respeito ao mestrado analisado, apenas existe uma UC relacionada com qualidade de software, Teste e Validação de Software, sendo que esta é opcional dentro das UC's existentes para a área de Engenharia de Software.

A nível de Licenciaturas, apenas na cadeira de Engenharia de Software, existem conteúdos relacionados com testes de software.

Tabela 6- Cursos da área de TI nos estabelecimentos de Ensino Superior em Braga

Ensino Superior no Distrito de Braga	
Estabelecimentos de Ensino Público	Estabelecimentos de Ensino Privado
Universidade do Minho	Universidade Lusíada
Faculdade de Ciências e Tecnologia	Centro Universitário Lusíada - Norte - Vila Nova de Famalicão
• Licenciatura em Engenharia Informática • Mestrado em Engenharia Informática • Licenciatura em Engenharia de Telecomunicações e Informática • Licenciatura em Engenharia e Gestão de Sistemas de Informação • Mestrado em Engenharia de Telecomunicações e Informática • Mestrado em Engenharia e Gestão de Sistemas de Informação	• Licenciatura em Engenharia Eletrónica e Informática • Mestrado em Engenharia Eletrónica e Informática
Instituto Politécnico do Cávado e do Ave Escola Superior de Tecnologia	Escola Superior de Tecnologias de Fafe
• Licenciatura em Engenharia Eletrónica e de Computadores • Licenciatura em Engenharia de Sistemas Informáticos • Mestrado em Engenharia Informática • Mestrado em Engenharia Eletrónica e de Computadores	• Licenciatura em Tecnologias e Gestão de Sistemas Informação

A nível de universidades no distrito de Braga (Tabela 6), começando pelas duas instituições de ensino privado, nos três cursos analisados, nenhum, dentro dos seus programas inclui conteúdos relacionados com qualidade de software, pelo menos, daquilo que foi possível obter. Na Licenciatura em Tecnologias e Gestão de Sistemas Informação da Escola Superior de Tecnologias de Fafe e o mestrado em Engenharia Eletrónica e Informática da Universidade Lusíada, contém nos seus programas Engenharia de Software, porém como apenas conseguimos ter acesso ao nome da UC não podemos assumir que os conteúdos são todos iguais aos lecionados nas restantes universidades.

Nos dois estabelecimentos de ensino público, no Instituto Politécnico do Cávado e do Ave, apenas o Mestrado em Engenharia Informática possui uma UC dedicada à qualidade de software e testes, denominada de Qualidade e Testes de Software. Os restantes cursos, não possuem estes conteúdos ou relacionados nos seus programas.

Na Universidade do Minho, nas três licenciaturas e nos 3 mestrados analisados, contanto com os mestrados integrados também, apenas existe uma UC opcional cujos conteúdos são de qualidade de software, Teste e Validação de Sistemas de Informação, sendo que nas várias UC's relacionadas com programação ou engenharia e desenvolvimento de software, existe quase sempre um tópico relacionado com testes, sejam eles de aceitação ou regressão.

Tabela 7- Cursos da área de TI nos estabelecimentos de Ensino Superior no Porto

Ensino Superior no Distrito do Porto		
Estabelecimentos de Ensino Público		Estabelecimentos de Ensino Privado
Universidade do Porto		Universidade Fernando Pessoa
Faculdade de Ciências	Faculdade de Engenharia	
• Licenciatura em Ciência de Computadores • Mestrado em Ciência de Computadores	• Licenciatura em Engenharia Informática e Computação • Mestrado em Engenharia Informática e Computação • Mestrado em Engenharia de Software	• Licenciatura em Engenharia Informática • Mestrado em Engenharia Informática
Instituto Politécnico do Porto		Universidade Lusófona do Porto
Escola Superior de Tecnologia e Gestão	Instituto Superior de Engenharia do Porto	
• Licenciatura em Engenharia Informática • Licenciatura em Sistemas de Informação para a Gestão • Mestrado em Engenharia Informática	• Licenciatura em Engenharia de Telecomunicações e Informática • Licenciatura em Engenharia Informática • Mestrado em Engenharia Informática	• Licenciatura em Engenharia Informática
		Universidade da Maia
		• Licenciatura em Informática
		Universidade Portucalense Infante D. Henrique
		• Licenciatura em Engenharia Informática • Licenciatura em Sistemas de Informação para Gestão • Mestrado em Informática
		Instituto Superior Politécnico Gaya
		Escola Superior de Ciência e Tecnologia
		• Licenciatura em Engenharia Informática
		Instituto Superior de Tecnologias Avançadas do Porto (ISTEC Porto)
		• Licenciatura em Engenharia Informática

Ensino Superior no Distrito do Porto	
Estabelecimentos de Ensino Público	Estabelecimentos de Ensino Privado
	ISLA
	Instituto Politécnico de Gestão e Tecnologia - Escola Superior de Tecnologia
	• Licenciatura em Engenharia Informática Mestrado em Engenharia de Tecnologias e Sistemas Web

No que diz respeito aos resultados obtidos para a pesquisa realizada para o distrito do Porto (Tabela 7) e começando pelos estabelecimentos de ensino privado, que são a grande maioria, temos a questão comum de a maioria apenas apresentar o nome das UC's no programa, pelo que conseguimos ver que quase todas as licenciaturas têm a disciplina de Engenharia de Software ou Gestão de Projeto de Software, porém não podemos afirmar que possuam os mesmos conteúdos que as já analisadas anteriormente.

Na Universidade Fernando Pessoa na Licenciatura de Engenharia Informática, na UC de Engenharia de Software, na parte do programa dedicada aos testes de software, ao contrário de todas as outras universidades, até agora não aborda os testes de uma forma geral, mas dá ênfase aos testes automatizados (Figura 51). Já na Licenciatura de Engenharia Informática da Universidade Portucalense existe mesmo a UC de Qualidade de Software, cujo programa está relacionado com a qualidade e testes de software (Figura 52).

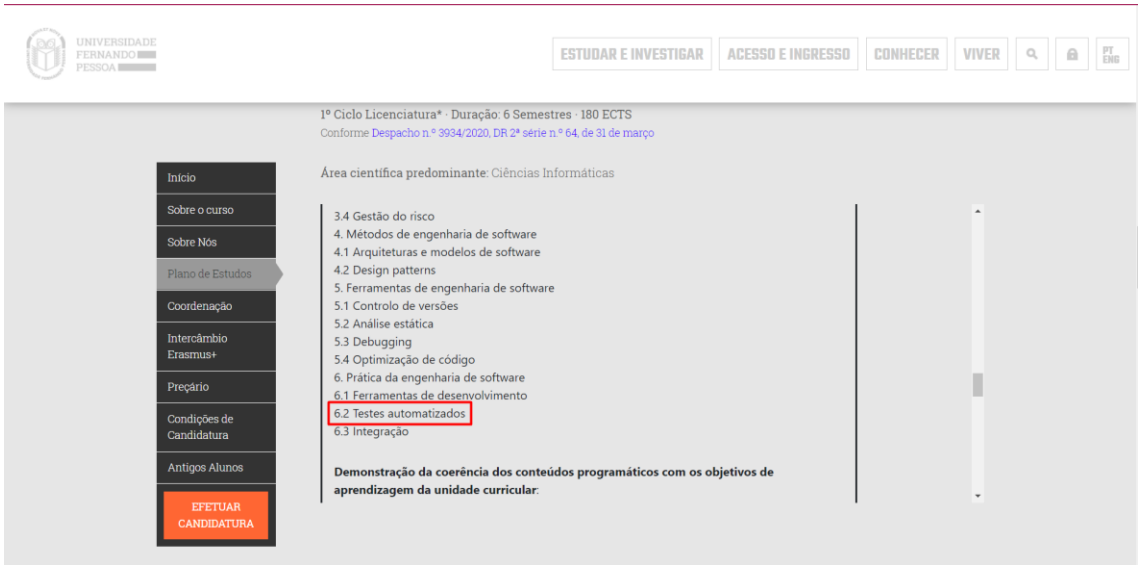


Figura 51- Plano curricular da Licenciatura em Engenharia Informática da UFP

Estatística Aplicada	6	Bases de Dados	6
Sistemas Operativos	6	Laboratório de Tecnologias Web	6
Laboratório de Programação	6	Engenharia de Software	6
Engenharia de Requisitos	6	Laboratório de Aplicações Multimédia	6

3ºano

1º Semestre	ECTS	2º Semestre	ECTS
Laboratório de Aplicações Móveis	6	Sistemas Distribuídos	6
Inteligência Artificial	6	Laboratório de Análise de Dados	6
Qualidade de Software	6	Segurança Informática	6
Empreendedorismo	6	Estágio	8
Gestão de Projetos	6	OP - Internet das Coisas	4
		OP - Visão por Computador	4

Figura 52- Plano curricular da Licenciatura em Engenharia Informática da U. Portucalense

Passando às instituições de ensino público, apenas no mestrado de Engenharia de Software da Faculdade de Engenharia da Universidade do Porto e no mestrado de Engenharia Informática do ISEP no Politécnico do Porto apresentam UC's relacionadas com qualidade de Software, sendo estas Teste, Verificação e Validação de Software (Figura 53) e Qualidade na Engenharia de Software (Figura 54), respetivamente, sendo estas obrigatórias. Todos os outros cursos vão ao encontro do que já foi descrito anteriormente para outras instituições, ou não existe informação relevante ou existem algumas partes nos programas de UC's, como Engenharia de Software ou semelhante, onde são abordados os testes de software.

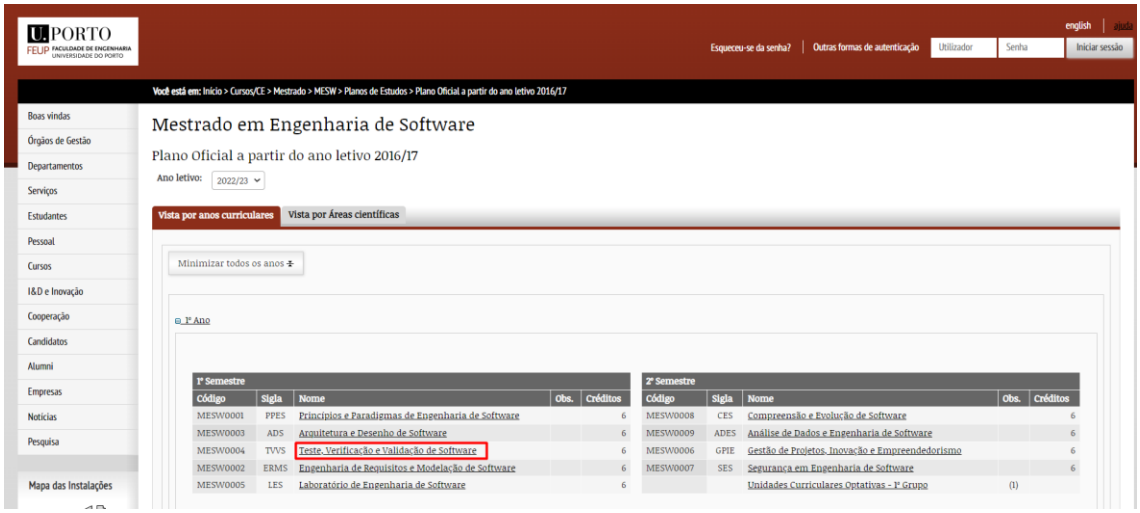


Figura 53- Plano curricular do Mestrado em Engenharia de Software da FEUP

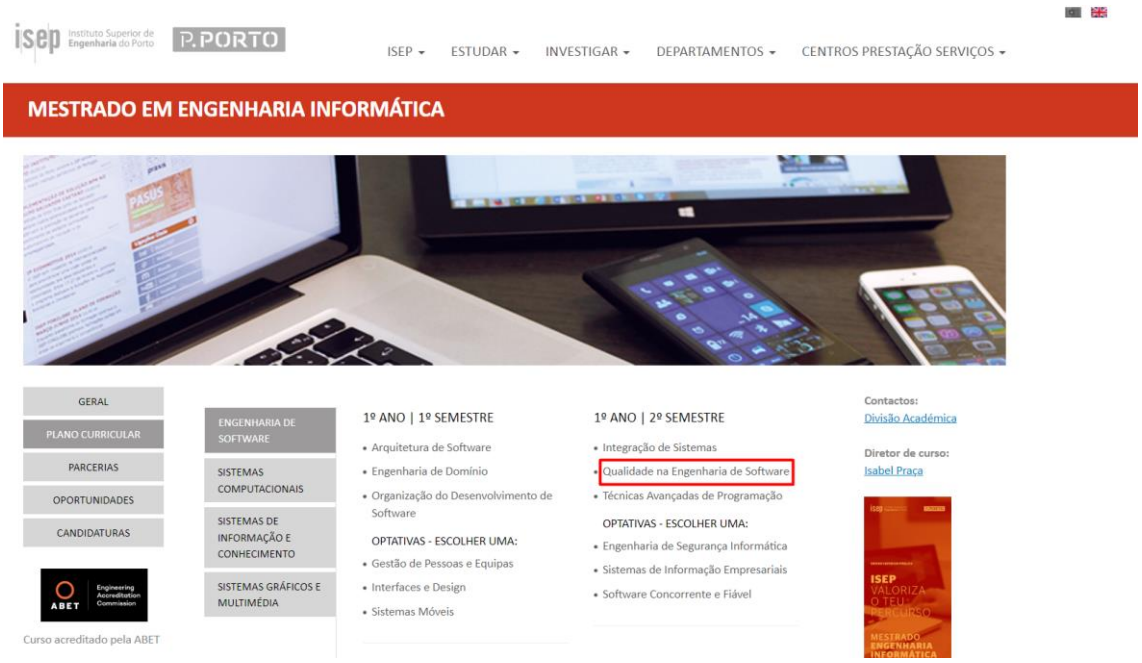


Figura 54- Plano curricular do Mestrado em Engenharia Informática do ISEP

Em suma, podemos concluir que pelo menos dentro destas zonas geográficas as ofertas a nível de formação são muito semelhantes, todos os cursos analisados eram de áreas relacionadas com tecnologias de informação, porém os cursos mais aproximados daquilo que era pretendido são os cursos em que a maioria das UC's estão relacionadas com informática, nomeadamente da área de Engenharia de Software. Daí o facto de só as licenciaturas ou mestrados integrados que tivessem a UC de Engenharia de Software possuíam conteúdos no programa que fossem de encontro ao que era procurado.

No que diz respeito a UC's cujos conteúdos sejam exclusivamente dedicados a Qualidade de Software estas apenas existem a nível de mestrado, retirando duas exceções, sendo que, excetuando três casos, todas estas UC's são opcionais, estando na maior parte das vezes, nos sites consultados, no fim da página, pelo que dá a sensação que poucas são as cadeiras escolhidas para serem lecionadas. Relembrando que estes dados são apenas baseados nas informações que as próprias instituições disponibilizam e algumas pecam por escassez de informação, mas naquilo que foi possível retirar, esta é uma área que para além de pouco explorada no mundo empresarial, é pouco explorada a nível da própria formação académica.

Logicamente, é positivo o facto de estes conteúdos serem abordados, mas dada a quantidade muito extensa de conteúdos da UC de Engenharia de Software, obviamente só permite o fornecimento de uma ideia muito superficial e geral em relação a esta temática. E sendo que a grande maioria dos alunos que envereda pelas áreas tecnológicas termina as suas licenciaturas e poucos são os que prosseguem estudos para um mestrado, devido a rapidamente ficarem vinculados ao mercado de trabalho, não ajuda muito. Se bem que esta é uma tendência que está a melhorar, talvez daqui a alguns anos, estes dados possam ser relativamente diferentes e sendo que nas várias ofertas formativas estão sempre UC's relacionadas com as áreas que estão na "moda" é possível que um dia a qualidade do software esteja verdadeiramente na "moda."

Capítulo 5

5. Conclusão

Em jeito de conclusão ao estudo elaborado ao longo deste documento e estágio realizado, pode ser retirado que não adianta ter mentalidade de que o produto final, neste caso um software, tem de ter qualidade e ter uma equipa de testes para garantir essa mesma qualidade, quando na verdade não passa de um fundamento teórico que não é posto em prática na sua totalidade.

Dentro das equipas de um projeto de desenvolvimento de software, continua a ser muito valorizado o facto de termos um produto desenvolvido, mas quanto à sua qualidade pouco se diz ou pensa. Continuamos em muitos casos a ouvir a frase “está com problemas ou bugs em produção”. Atualmente, não é porque não existem equipas para garantir essa mesma qualidade, mas porque ainda existe uma certa “desvalorização”, muitas vezes inconsciente, para com esse tipo de atividades e funções. As equipas e os profissionais de QA são muitas vezes “desvalorizados” entre as equipas e o seu trabalho visto como um incómodo. Continuam a existir *guidelines* irrealistas e apertadas para aquilo que se pode efetivamente produzir, deixando um espaço temporal ainda mais reduzido no final para serem executadas todas as atividades de testes.

O que se pretende transmitir com o anterior parágrafo não é um enaltecimento da área de QA em relação a outras áreas como o desenvolvimento, dando esta como exemplo, sabemos que é de extrema importância e sem desenvolvimento não existe produto final, nem o que testar. Mas é importante uma equipa num projeto de software funcionar como um todo, e que a cultura existente ainda em muitas empresas tem de evoluir neste sentido. É sabido também que esta não é uma questão linear, existem muitos fatores que contribuem para que esta questão cultural persista, como orçamentação e os próprios processos de gestão. O fundamental é a perceção de que numa área fundamentalmente tecnológica, não pode ser só a tecnologia a evoluir, mas sim as mentalidades e os processos utlizados.

Ao longo deste estudo apercebemo-nos que seja qual for a metodologia utilizada ou qual seja o projeto é muito importante que os testes acompanhem o desenvolvimento do início ao fim e não apenas no final do processo. Os testes de regressão são fundamentais e facilitam perdas de tempo, trabalho e custos desnecessários no final de todo o processo. É neste aspeto que cada vez mais é considerado importante a automatização de testes, como pudemos observar na experiência no 2º cliente, a automatização de testes trouxe inúmeros benefícios aos projetos onde foram aplicados tais como:

✓Execuções em diversas vezes e em diversos ambientes com mínimo de esforço;

✓Feedback mais rápido, antecipando a identificação de eventuais não conformidades;

✓Reutilização de componentes (sejam eles scripts ou KW);

✓Maior cobertura dos testes;

✓Os dados utilizados sempre de forma aleatória.

Logicamente a existência destes nunca deve abolir ou retirar em definitivo a execução de testes manuais, até porque para o desenvolvimento de um bom teste automático é importante estar por trás um bom teste manual. Mas existem critérios para a identificação dos fluxos a serem automatizados dos quais foram identificados alguns para auxiliar na definição e prioridade dos fluxos a serem automatizados como:

Ciclos repetitivos -Todos os testes que envolvam a repetição dos mesmos passos numerosas vezes com ligeiras variações são candidatos à automação. Testar funcionalidades como login, registos e consultas ao sistema são candidatos a automatizar.

Consumidores de tempo - Testes relativos a processos de negócio cuja execução manual consome muito tempo são bons candidatos à automação (Ex: validações)

Relevância para o negócio - São os fluxos que podem ser muito ou pouco utilizados, porém apresentam risco para o negócio.

Taxa de erro por funcionalidade - São mapeadas as funcionalidades que geram mais bugs para o negócio e, a partir deste ponto, será definido o fluxo a ser automatizado.

Fluxos críticos - Funcionalidades que apresentam maior vulnerabilidade / sensibilidade ao negócio.

Isso é um desafio porque todos dentro de uma equipa são responsáveis pela qualidade e, dependendo da composição da equipa, diferentes aspetos de qualidade podem ser tratados por pessoas diferentes.

Para além da equipa de qualidade, apesar de que quem desenvolve não deve fazer testes exaustivos, não é despropositada a realização de testes unitários ao seu código, por exemplo.

No que diz respeito a fatores externos ao meio empresarial como a formação académica, talvez fosse importante não menosprezar a qualidade do código produzido, mesmo que este seja desenvolvido em meio académico, nas UC's dedicadas ou que possuam conteúdos dedicados à qualidade de software, para além de ensinar todo um processo teórico, que é muito importante, mas ao mesmo tempo aproximar daquilo que é o ambiente empresarial e salientar aquilo que efetivamente faz falta no mercado de trabalho.

Inclusive, porque o que vemos a nível de propostas de emprego, na área de QA, os requisitos são extremamente parecidos e facilmente colmatáveis.

No que diz respeito a questões salariais, é um caso mais difícil de analisar, no sentido em que o mercado "está ao rubro" e temos vários tipos de empresas, com vários tipos de projetos e filosofias, mas o que podemos apontar em comum, é que os salários propostos na área de QA são sempre inferiores aos salários propostos em outras áreas das tecnologias de informação, uma forma de atrair profissionais para essas áreas seria as propostas de salários mais competitivos.

Existe muito trabalho a percorrer, no que toca a garantir condições e entusiamo por enveredar na área de QA, mas como foi experienciado neste estágio, tudo depende da organização empresarial, do respeito e cooperação entre os vários profissionais e pela própria formação e conhecimentos que são transmitidos. A perspetiva desejada para um futuro breve é a de que o profissional de QA é um elemento da equipa que colabora com o projeto e com os seus elementos para que o resultado deste, seja o desejado e tenha qualidade.

5.1. Trabalho Futuro

No que diz respeito a um possível trabalho futuro, seria mais interessante realizar o mesmo estudo mas fazendo parte de um projeto que estivesse prestes a iniciar e fazer o acompanhamento completo até ao seu término, algo que não foi possível, devido ao tempo limitado para realização do estágio e a questões de mercado, que fizeram com que houvesse uma passagem por vários projetos diferentes e resultados pertinentes mas não houve foco apenas num projeto e numa única *framework* de modo a aprofundar e detalhar mais.

Também a nível de estudos de mercado aprofundar um pouco mais de modo a alargar a zona geográfica para obter mais amostras e consultar mais plataformas, para alargar a quantidade e variedade de resultados, verificar se existiria uma discrepância tão grande face ao que já se concluiu neste estudo. E, ao invés de ser um estudo baseado em pesquisa e em apenas uma única experiência profissional, existir questionários a vários profissionais da área que com outras visões e experiências deem a sua contribuição para resultados mais ricos e sugestivos quanto a possíveis soluções.

Claro que, profissionalmente será sempre objetivo contribuir com novas sugestões e alternativas que facilitem a vida e trabalho de um profissional de QA, em especial *Testers* de automação, no que diz respeito à evolução das *frameworks* (novas funcionalidades e melhoramento das já existentes) utilizadas no desenvolvimento de testes automáticos.

Referências

Amaral, E. (2021, Junho 15), Primeiros passos para a Automatização de Testes de Software, ComputerWorld, https://www.computerworld.com.pt/2021/06/15/primeiros-passos-para-a-automatizacao-de-testes-de-software/

Amoroso, D. L. (1988). Organizational issues of end-user computing. ACM SIGMIS Database: the DATABASE for Advances in Information Systems, 19(3-4), 49-58.

Atlanta, GA. (2014, Maio 28), Thousands of Developers Worldwide Empowered by OutSystems Free Cloud Offering, Outsystems, https://www.outsystems.com/news/free-cloud-offering/

Bock, A. C., & Frank, U. (2021). Low-code platform. *Business & Information Systems Engineering*, *63*(6), 733-740.

Chang, Y. H., & Ko, C. B. (2017). A study on the design of low-code and no code platform for mobile application development. International journal of advanced smart convergence, 6(4), 50-55.

Gaea, (2018), Afinal, o que é Quality Assurance?, Gaea, https://gaea.com.br/afinal-o-que-e-quality-assurance/

Hall, E. M. (1998). Managing risk: Methods for software systems development. Pearson Education.

Jasiulewicz-Kaczmarek, M., & Piechowski, M. (2015). Improvement of the process of information management in maintenance-a case study. In Applied Mechanics and Materials (Vol. 795, pp. 99-106). Trans Tech Publications Ltd.

Kłosowski, G., & Gola, A. (2016, September). Risk-based estimation of manufacturing order costs with artificial intelligence. In 2016 Federated Conference on Computer Science and Information Systems (FedCSIS) (pp. 729-732). IEEE.

Korel, B. e Al-Yami, A. (1998), Automated Regression Test Generation, Department of Computer Science Illinois Institute of Technology e College of Technology at Dammam, Chicago, http://portal.acm.org/citation.cfm?id=271771.271803

Lacerda, A. (2019, novembro 8), Ser QA em Portugal - Vale ou não vale a pena?, LinkedIn, https://www.linkedin.com/pulse/ser-qa-em-portugal-vale-ou-n%C3%A3o-pena-amanda-lacerda/?originalSubdomain=pt

Madrinha, J. e Leitao, L. (2018), Is Outsystems Portugal's Latest Unicorn?, Forbes, https://www.forbes.com/sites/forbesinternational/2018/07/13/is-outsystems-portugals-latest-unicorn/?sh=32ce95443633

Marvin, R. (2017), OutSystems Review, Pcmag, https://www.pcmag.com/reviews/outsystems

Media de salário de um Software Developer. (15 de 09 de 2022). Obtido de Talent.com: https://pt.talent.com/salary?job=software+developer

Media de salário de um Software Tester. (15 de 09 de 2022). Obtido de Talent.com: https://pt.talent.com/salary?job=software+tester

Media de salário de um QA Tester. (15 de 09 de 2022). Obtido de Talent.com: https://pt.talent.com/salary?job=qa+tester

Noleto, C. (2020, julho 23), Teste unitário: o que são, por que usar e por onde começar?, Trybe, https://blog.betrybe.com/tecnologia/testes-unitarios/

O'Regan, G. (2002). Introduction to software quality. In *A Practical Approach to Software Quality* (pp. 1-48). Springer, New York, NY.

Plano de estudos da Licenciatura em Engenharia Informática. (16 de 09 de 2022). Obtido de Instituto Piaget: https://ipiaget.org/licenciaturas/engenharia-informatica/#1523959897308-29174b42-1aa7

Plano de estudos da Licenciatura em Engenharia Informática. (16 de 09 de 2022). Obtido de Faculdade de Ciências e Tecnologias da Universidade Nova de Lisboa: https://www.fct.unl.pt/ensino/curso/licenciatura-em-engenharia-informatica

Plano de estudos da Licenciatura em Engenharia Eletrotecnica e de Computadores. (16 de 09 de 2022). Obtido de Faculdade de Ciências e Tecnologias da Universidade Nova de Lisboa: https://www.fct.unl.pt/ensino/curso/licenciatura-em-engenharia-eletrotecnica-e-de-computadores

Plano de estudos do Mestrado em Engenharia Informática. (16 de 09 de 2022). Obtido de Faculdade de Ciências e Tecnologias da Universidade Nova de Lisboa: https://www.fct.unl.pt/ensino/curso/mestrado-em-engenharia-informatica

Plano de estudos do Mestrado em Engenharia Eletrotecnica e de Computadores. (16 de 09 de 2022). Obtido de Faculdade de Ciências e Tecnologias da Universidade Nova de Lisboa:

https://www.fct.unl.pt/ensino/curso/mestrado-em-engenharia-eletrotecnica-e-de-computadores

Plano de estudos da Licenciatura em Gestão de Sistemas de Informação. (16 de 09 de 2022). Obtido de Escola Superior de Ciências Empresariais do Instituto Politécnico de Setúbal: http://www.si.ips.pt/esce_si/planos_estudos_geral.formview?p_Pe=1459

Plano de estudos do Mestrado em Engenharia de Software. (16 de 09 de 2022). Obtido de Escola Superior de Tecnologia de Setúbal do Instituto Politécnico de Setúbal: https://portal.ips.pt/ests/pt/cur_geral.cur_planos_estudos_view?pv_plano_id=21867&pv_ano_lectivo=2021&pv_tipo_cur_sigla=M&pv_origem=CUR

Plano de estudos do Mestrado Engenharia Eletrotécnica e de Computadores. (16 de 09 de 2022). Obtido de Escola Superior de Tecnologia de Setúbal do Instituto Politécnico de Setúbal: https://portal.ips.pt/ests/pt/cur_geral.cur_planos_estudos_view?pv_plano_id=22032&pv_ano_lectivo=2021&pv_tipo_cur_sigla=M&pv_origem=CUR

Plano de estudos da Licenciatura em Engenharia Informática. (16 de 09 de 2022). Obtido de Escola Superior de Tecnologia de Setúbal do Instituto Politécnico de Setúbal: http://portal.ips.pt/ests/pt/cur_geral.cur_planos_estudos_view?pv_plano_id=21847&pv_ano_lectivo=2021&pv_tipo_cur_sigla=L&pv_origem=CUR

Plano de estudos da Licenciatura em Engenharia Informática. (16 de 09 de 2022). Obtido de Universidade Autónoma de Lisboa: https://autonoma.pt/cursos/engenharia-informatica/

Plano de estudos da Licenciatura em Engenharia Informática e de Telecomunicações. (16 de 09 de 2022). Obtido de Universidade Autónoma de Lisboa: https://autonoma.pt/cursos/engenharia-informatica-e-de-telecomunicacoes/

Plano de estudos da Licenciatura em Informática de Gestão. (16 de 09 de 2022). Obtido de Universidade Autónoma de Lisboa: https://autonoma.pt/cursos/informatica-de-gestao/

Plano de estudos da Licenciatura em Informática de Gestão. (16 de 09 de 2022). Obtido de Universidade Europeia - IADE: https://www.iade.europeia.pt/cursos/licenciaturas/licenciaturas/informatica-de-gestao

Plano de estudos da Licenciatura em Engenharia Informática. (16 de 09 de 2022). Obtido de Universidade Europeia - IADE: https://www.iade.europeia.pt/cursos/licenciaturas/licenciaturas/engenharia-de-informatica

Plano de estudos da Licenciatura em Engenharia Informática. (16 de 09 de 2022). Obtido de Universidade Lusíada Lisboa: https://www.lis.ulusiada.pt/pt-pt/cursos/2022-2023/1%C2%BAciclo%E2%80%93licenciaturasemestradosintegrados/engenhariainform%C3%A1tica.aspx

Plano de estudos da Licenciatura em Gestão de Sistemas e Computação. (16 de 09 de 2022). Obtido de Universidade Atlântica: https://www.uatlantica.pt/licenciatura-em-gestao-de-sistemas-e-computacao/

Plano de estudos da Licenciatura em Informática de Gestão. (16 de 09 de 2022). Obtido de Universidade Lusófona de Humanidades e Tecnologias: https://www.ulusofona.pt/licenciaturas/informatica-de-gestao

Plano de estudos da Licenciatura em Engenharia Informática Redes e Telecomunicações. (16 de 09 de 2022). Obtido de Universidade Lusófona de Humanidades e Tecnologias: https://www.ulusofona.pt/licenciaturas/engenharia-informatica-redes-e-telecomunicacoes

Plano de estudos da Licenciatura em Engenharia Informática. (16 de 09 de 2022). Obtido de Universidade Lusófona de Humanidades e Tecnologias: https://www.ulusofona.pt/licenciaturas/engenharia-informatica

Plano de estudos do Mestrado em Engenharia Informática e Sistemas de Informação. (16 de 09 de 2022). Obtido de Universidade Lusófona de Humanidades e Tecnologias: https://www.ulusofona.pt/mestrados/engenharia-informatica-e-sistemas-de-informacao

Plano de estudos da Licenciatura em Engenharia Informática. (16 de 09 de 2022). Obtido de Instituto Superior de Tecnologias Avançadas de Lisboa: http://www.istec.pt/index.php/licenciatura-em-informatica/?doing_wp_cron=1662759987.2217290401458740234375

Plano de estudos do Mestrado em Engenharia Informática Redes e de Computadores. (17 de 09 de 2022). Obtido de Instituto Superior de Engenharia de Lisboa: https://www.isel.pt/curso/mestrado/mestrado-em-engenharia-informatica-e-de-computadores

Plano de estudos do Mestrado em Engenharia Informática e Multimédia. (17 de 09 de 2022).

Obtido de Instituto Superior de Engenharia de Lisboa: https://www.isel.pt/curso/mestrado/mestrado-em-engenharia-informatica-e-multimedia

Plano de estudos da Licenciatura em Engenharia Eletronica e Telecomunicações e de Computadores. (17 de 09 de 2022). Obtido de Instituto Superior de Engenharia de Lisboa: https://www.isel.pt/curso/licenciatura/licenciatura-em-engenharia-eletronica-e-telecomunicacoes-e-de-computadores

Plano de estudos da Licenciatura em Engenharia Informática e de Computadores. (17 de 09 de 2022). Obtido de Instituto Superior de Engenharia de Lisboa: https://www.isel.pt/curso/licenciatura/licenciatura-em-engenharia-informatica-e-de-computadores

Plano de estudos da Licenciatura em Engenharia Informática e Multimédia. (17 de 09 de 2022). Obtido de Instituto Superior de Engenharia de Lisboa: https://www.isel.pt/curso/licenciatura/licenciatura-em-engenharia-informatica-e-multimedia

Plano de estudos da Licenciatura em Engenharia Informática Redes e Telecomunicações. (17 de 09 de 2022). Obtido de Instituto Superior de Engenharia de Lisboa: https://www.isel.pt/curso/licenciatura/licenciatura-em-engenharia-informatica-redes-e-telecomunicacoes

Plano de estudos da Licenciatura em Engenharia Informática e de Computadores. (17 de 09 de 2022). Obtido de Escola Náutica Infante D. Henrique: https://www.enautica.pt/pt/cursos-4/licenciaturas-9/engenharia-informatica-e-de-computadores-/engenharia-informatica-e-de-computadores--1187

Plano de estudos do Mestrado em Engenharia Informática. (17 de 09 de 2022). Obtido de ISCTE Instituto Universitário de Lisboa: https://www.iscte-iul.pt/curso/12/mestrado-engenharia-informatica

Plano de estudos do Mestrado em Engenharia de Telecomunicações Informática. (17 de 09 de 2022). Obtido de ISCTE Instituto Universitário de Lisboa: https://www.iscte-iul.pt/curso/11/mestrado-engenharia-de-telecomunicacoes-informatica

Plano de estudos do Mestrado em Gestão de Sistemas de Informação. (17 de 09 de 2022).

Obtido de ISCTE Instituto Universitário de Lisboa: https://www.iscte-iul.pt/curso/18/mestrado-gestao-de-sistemas-de-informacao

Plano de estudos da Licenciatura em Ciências de Dados. (17 de 09 de 2022). Obtido de ISCTE Instituto Universitário de Lisboa: https://www.iscte-iul.pt/curso/291/licenciatura-ciencia-de-dados

Plano de estudos da Licenciatura em Desenvolvimento de Software de Aplicações. (17 de 09 de 2022). Obtido de ISCTE Instituto Universitário de Lisboa: https://www.iscte-iul.pt/curso/378/licenciatura-desenvolvimento-de-software-aplicacoes

Plano de estudos da Licenciatura em Engenharia de Telecomunicações Informática. (17 de 09 de 2022). Obtido de ISCTE Instituto Universitário de Lisboa: https://www.iscte-iul.pt/curso/2/licenciatura-engenharia-de-telecomunicacoes-informatica

Plano de estudos da Licenciatura em Engenharia Informática. (17 de 09 de 2022). Obtido de ISCTE Instituto Universitário de Lisboa: https://www.iscte-iul.pt/curso/3/licenciatura-engenharia-informatica

Plano de estudos da Licenciatura em Informática e Gestão de Empresas. (17 de 09 de 2022). Obtido de ISCTE Instituto Universitário de Lisboa: https://www.iscte-iul.pt/curso/6/licenciatura-informatica-gestao-de-empresas

Plano de estudos da Licenciatura em Tecnologias Digitais e Inteligência Artificial. (17 de 09 de 2022). Obtido de ISCTE Instituto Universitário de Lisboa: https://www.iscte-iul.pt/curso/376/licenciatura-tecnologias-digitais-inteligencia-artificial

Plano de estudos da Licenciatura em Tecnologias Digitais e Segurança. (17 de 09 de 2022). Obtido de ISCTE Instituto Universitário de Lisboa: https://www.iscte-iul.pt/curso/379/licenciatura-tecnologias-digitais-seguranca

Plano de estudos da Licenciatura em Engenharia Informática. (17 de 09 de 2022). Obtido de Faculdade de Ciências da Universidade de Lisboa:

https://fenix.ciencias.ulisboa.pt/degrees/engenharia-informatica-564500436615277/curriculo

Plano de estudos da Licenciatura Informática. (17 de 09 de 2022). Obtido de Faculdade de Ciências da Universidade de Lisboa: https://fenix.ciencias.ulisboa.pt/degrees/informatica-564500436615356/curriculo

Plano de estudos da Licenciatura em Engenharia Informática. (17 de 09 de 2022). Obtido de Faculdade de Ciências da Universidade de Lisboa:

https://fenix.ciencias.ulisboa.pt/degrees/seguranca-informatica-564500436615446/curriculo

Plano de estudos do Mestrado em Engenharia Informática. (17 de 09 de 2022). Obtido de Faculdade de Ciências da Universidade de Lisboa:

https://fenix.ciencias.ulisboa.pt/degrees/engenharia-informatica-564500436615278/curriculo

Plano de estudos do Mestrado em Tecnologias de Informação. (17 de 09 de 2022). Obtido de Faculdade de Ciências da Universidade de Lisboa:

https://fenix.ciencias.ulisboa.pt/degrees/tecnologias-de-informacao-564500436615450/curriculo

Plano de estudos da Licenciatura em Sistemas e Tecnologias de Informação. (17 de 09 de 2022). Obtido de Nova IMS da Universidade Nova de Lisboa:

https://www.novaims.unl.pt/pt/ensino/cursos/licenciaturas/sistemas-e-tecnologias-de-informacao/plano-de-estudos-sistemas-e-tecnologias-de-informacao/#

Plano de estudos da Licenciatura em Gestão de Informação. (17 de 09 de 2022). Obtido de Nova IMS da Universidade Nova de Lisboa:

https://www.novaims.unl.pt/pt/ensino/cursos/licenciaturas/gestao-de-informacao/plano-de-estudos-gestao-de-informacao/#

Plano de estudos da Licenciatura em Ciência de Dados. (17 de 09 de 2022). Obtido de Nova IMS da Universidade Nova de Lisboa:

https://www.novaims.unl.pt/pt/ensino/cursos/licenciaturas/ciencia-de-dados/plano-de-estudos-ciencia-de-dados/#

Plano de estudos da Licenciatura em Engenharia Informática e de Computadores. (17 de 09 de 2022). Obtido de Instituto Superior Técnico: https://fenix.tecnico.ulisboa.pt/cursos/leic-a/curriculo

Plano de estudos do Mestrado em Engenharia de Telecomunicações e Informática. (17 de 09 de 2022). Obtido de Instituto Superior Técnico: https://fenix.tecnico.ulisboa.pt/cursos/merc/curriculo

Plano de estudos do Mestrado em Engenharia Informática e de Computadores. (17 de 09 de 2022). Obtido de Instituto Superior Técnico- Alameda: https://fenix.tecnico.ulisboa.pt/cursos/meic-a/curriculo

Plano de estudos do Mestrado em Engenharia Informática e de Computadores. (17 de 09 de 2022). Obtido de Instituto Superior Técnico- Taguspark: https://fenix.tecnico.ulisboa.pt/cursos/meic-t/curriculo

Plano de estudos da Licenciatura em Engenharia Electrotécnica e de Computadores. (17 de 09 de 2022). Obtido de Instituto Superior Técnico:

https://fenix.tecnico.ulisboa.pt/cursos/leec21/curriculo

Plano de estudos da Licenciatura em Engenharia de Telecomunicações e Informática. (17 de 09 de 2022). Obtido de Instituto Superior Técnico: https://fenix.tecnico.ulisboa.pt/cursos/lerc/curriculo

Plano de estudos da Licenciatura em Tecnologias e Gestão de Sistemas de Informação. (20 de 09 de 2022). Obtido de Instituto de Estudos Superiores de Fafe:

https://www.iesfafe.pt/index.php?option=com_content&view=article&id=1239

Plano de estudos da Licenciatura em Ciência de Computadores. (20 de 09 de 2022). Obtido de Faculdade de Ciências da Universidade do Porto:

https://sigarra.up.pt/fcup/pt/cur_geral.cur_planos_estudos_view?pv_plano_id=31845&pv_ano_lectivo=2021&pv_tipo_cur_sigla=L&pv_origem=CUR#div_3_id_415171

Plano de estudos do Mestrado em Ciência de Computadores. (20 de 09 de 2022). Obtido de Faculdade de Ciências da Universidade do Porto:

https://sigarra.up.pt/fcup/pt/cur_geral.cur_planos_estudos_view?pv_plano_id=10621&pv_ano_lectivo=2021&pv_tipo_cur_sigla=M&pv_origem=CUR

Plano de estudos da Licenciatura em Engenharia Eletrotécnica e de Computadores. (20 de 09 de 2022). Obtido de Faculdade de Engenaria da Universidade do Porto:

https://sigarra.up.pt/feup/pt/cur_geral.cur_planos_estudos_view?pv_plano_id=31024&pv_ano_lectivo=2022&pv_tipo_cur_sigla=L#div_id_408994

Plano de estudos do Mestrado em Engenharia de Software. (20 de 09 de 2022). Obtido de Faculdade de Engenaria da Universidade do Porto:

https://sigarra.up.pt/feup/pt/cur_geral.cur_planos_estudos_view?pv_plano_id=15801&pv_ano_lectivo=2022&pv_tipo_cur_sigla=M&pv_origem=CUR#div_id_367447

Plano de estudos da Licenciatura em Segurança Informática em Redes de Computadores. (20 de 09 de 2022). Obtido de Escola Superior de Tecnologia e Gestão do Instituto Políteсnico do Porto: https://www.estg.ipp.pt/cursos/licenciatura/557

Plano de estudos da Licenciatura em Sistemas de Informação para a Gestão. (20 de 09 de 2022). Obtido de Escola Superior de Tecnologia e Gestão do Instituto Políteсnico do Porto: https://www.estg.ipp.pt/cursos/licenciatura/630

Plano de estudos do Mestrado em Engenharia Informática. (20 de 09 de 2022). Obtido de Escola Superior de Tecnologia e Gestão do Instituto Políteсnico do Porto:

https://www.estg.ipp.pt/cursos/mestrado/807

Plano de estudos da Licenciatura em Engenharia Informática. (20 de 09 de 2022). Obtido de Universidade Fernando Pessoa:

https://www.ufp.pt/inicio/estudar-e-investigar/licenciaturas/engenharia-informatica/#plano

Plano de estudos do Mestrado em Engenharia Informática. (20 de 09 de 2022). Obtido de Universidade Fernando Pessoa:

https://www.ufp.pt/inicio/estudar-e-investigar/mestrados/engenharia-informatica/#plano

Plano de estudos da Licenciatura em Engenharia Informática. (20 de 09 de 2022). Obtido de Universidade Lusófona do Porto: https://www.ulp.pt/licenciaturas/engenharia-informatica

Plano de estudos da Licenciatura em Informática. (20 de 09 de 2022). Obtido de Universidade da Maia:

https://www.ismai.pt/pt/ensino/oferta-formativa/licenciaturas/informatica

Plano de estudos do Mestrado em Informática Especialização em Engenharia de Software. (20 de 09 de 2022). Obtido de Universidade Portucalense:

https://www.upt.pt/inicio/departamentos/ciencia-e-tecnologia/cursos_dct/mestrados-em-informatica/mestrado-em-informatica-especializacao-em-engenharia-de-software/

Plano de estudos da Licenciatura em Sistemas de Informação. (20 de 09 de 2022). Obtido de Universidade Portucalense:

https://www.upt.pt/inicio/departamentos/ciencia-e-tecnologia/cursos_dct/licenciatura-em-sistemas-de-informacao-para-gestao/

Plano de estudos do Cibersegurança e Auditoria de Sistemas Informáticos. (20 de 09 de 2022). Obtido de Instituto Superior Politécnico de Gaya:

https://ispgaya.pt/ensino/oferta-formativa/mestrados/ciberseguranca-e-auditoria-de-sistemas-informaticos

Plano de estudos da Licenciatura em Engenharia Informática. (20 de 09 de 2022). Obtido de Instituto Superior Politécnico de Gaya:

https://ispgaya.pt/ensino/oferta-formativa/licenciaturas/engenharia-informatica/plano-curricular

Plano de estudos da Licenciatura em Engenharia Informática. (20 de 09 de 2022). Obtido de Instituto Superior de Tecnologias Avançadas do Porto:

https://istec-porto.pt/licenciatura-em-engenharia-informatica/

Plano de estudos da Licenciatura em Engenharia Informática. (20 de 09 de 2022). Obtido de ISLA Gaia:

https://www.islagaia.pt/pt/ensino/licenciaturas/652-engenharia-informatica.html

Plano de estudos do Mestrado em Engenharia de Tecnologias e Sistemas Web. (20 de 09 de

2022). Obtido de ISLA Gaia:

https://www.islagaia.pt/pt/ensino/mestrados/658-engenharia-de-tecnologias-e-sistemas-web-novo.html

Plano de estudos da Licenciatura em Engenharia Eletronica e Informática. (20 de 09 de 2022). Obtido de Universidade Lusíada Famalicão:

https://www.fam.ulusiada.pt/faculdades/curso.php?codplano=153

Plano de estudos da Licenciatura em Engenharia de Sistemas informáticos. (20 de 09 de 2022). Obtido de Escola Superior de Tecnologia do Instituto Politécnico do Cávado e do Ave:

https://est.ipca.pt/curso/engenharia-de-sistemas-informaticos/

Plano de estudos da Licenciatura em Engenharia de Eletrónica e de Computadores. (20 de 09 de 2022). Obtido de Escola Superior de Tecnologia do Instituto Politécnico do Cávado e do Ave:

https://est.ipca.pt/curso/engenharia-eletronica-e-de-computadores/

Plano de estudos da Licenciatura em Engenharia Informática. (20 de 09 de 2022). Obtido de Escola Superior de Tecnologia do Instituto Politécnico do Cávado e do Ave: https://est.ipca.pt/curso/engenharia-informatica/

Plano de estudos da Licenciatura em Engenharia de Telecomunicações e Informática. (20 de 09 de 2022). Obtido de Escola de Engenharia da Universidade do Minho:

https://www.eng.uminho.pt/pt/Estudar/_layouts/15/UMinho.PortaisUOEI.UI/Pages/CatalogoCursoDetail.aspx?itemId=4353&catId=13

Plano de estudos do Mestrado em Engenharia Informática. (20 de 09 de 2022). Obtido de Escola de Engenharia da Universidade do Minho:

https://www.eng.uminho.pt/pt/Estudar/_layouts/15/UMinho.PortaisUOEI.UI/Pages/CatalogoCursoDetail.aspx?itemId=4358&catId=13

Plano de estudos do Mestrado em Gestão de Sistemas de Informação. (20 de 09 de 2022). Obtido de Escola de Engenharia da Universidade do Minho:

https://www.eng.uminho.pt/pt/Estudar/_layouts/15/UMinho.PortaisUOEI.UI/Pages/CatalogoCursoDetai

l.aspx?itemId=4184&catId=13

Plano de estudos do Mestrado em Engenharia de Telecomunicações e Informática. (20 de 09 de 2022). Obtido de Escola de Engenharia da Universidade do Minho:

https://www.eng.uminho.pt/pt/Estudar/_layouts/15/UMinho.PortaisUOEI.UI/Pages/CatalogoCursoDetail.aspx?itemId=4300&catId=13

Plano de estudos da Licenciatura em Engenharia Informática. (20 de 09 de 2022). Obtido de Escola de Engenharia da Universidade do Minho:

https://www.eng.uminho.pt/pt/Estudar/_layouts/15/UMinho.PortaisUOEI.UI/Pages/CatalogoCursoDetail.aspx?itemId=4346&catId=13

Plano de estudos da Licenciatura em Engenharia e Gestão de Sistemas de Informação. (20 de 09 de 2022). Obtido de Escola de Engenharia da Universidade do Minho:

https://www.eng.uminho.pt/pt/Estudar/_layouts/15/UMinho.PortaisUOEI.UI/Pages/CatalogoCursoDetail.aspx?itemId=4345&catId=13

Pombo, D. (2020, novembro 18), A importância do QA no processo de desenvolvimento, WhithAffinity, https://with.affinity.pt/a-importancia-do-qa-no-processo-de-desenvolvimento/

Pressman, R. S. (2015). Software engineering: a practitioner's approach. Palgrave macmillan.

PrimeControl (2022), 10 ferramentas de automação de testes mais usadas, https://www.primecontrol.com.br/10-ferramentas-de-automacao-de-testes-mais-usadas/

Renata, (2013), Introdução aos diferentes tipos de teste, DEVMEDIA, https://www.devmedia.com.br/introducao-aos-diferentes-tipos-de-teste/29799

Resultados para a pesquisa de propostas de emprego para QA Tester em Portugal. (13 de 09 de 2022). Obtido de LinkedIn:
https://www.linkedin.com/jobs/search/?currentJobId=3080231711&geoId=100364837&keywords=qa%20tester&location=Portugal&refresh=true&start=25

Resultados para a pesquisa de propostas de emprego para QA Manual Tester em Portugal. (13 de 09 de 2022). Obtido de LinkedIn:
https://www.linkedin.com/jobs/search/?currentJobId=3100403202&geoId=100364837&keywords=qa%

20manual%20tester&location=Portugal&originalSubdomain=pt&refresh=true

Resultados para a pesquisa de propostas de emprego para QA Automation em Portugal. (13 de 09 de 2022). Obtido de LinkedIn: https://www.linkedin.com/jobs/search/?currentJobId=3100403202&geoId=100364837&keywords=qa%20automation%20tester&location=Portugal&originalSubdomain=pt&refresh=true

Resultados para a pesquisa de propostas de emprego para Software Developer em Portugal. (13 de 09 de 2022). Obtido de Glassdoor: https://www.glassdoor.com/Job/portugal-software-developer-jobs-SRCH_IL.0,8_IN195_KO9,27.htm?context=Jobs&clickSource=searchBox

Resultados para a pesquisa de propostas de emprego para QA Automation Tester em Portugal. (13 de 09 de 2022). Obtido de Glassdoor: https://www.glassdoor.com/Job/portugal-qa-automation-tester-jobs-SRCH_IL.0,8_IN195_KO9,29.htm?clickSource=searchBox

Resultados para a pesquisa de propostas de emprego para QA Manual Tester em Portugal. (13 de 09 de 2022). Obtido de Glassdoor: https://www.glassdoor.com/Job/portugal-qa-manual-tester-jobs-SRCH_IL.0,8_IN195_KO9,25.htm?clickSource=searchBox

Resultados para a pesquisa de propostas de emprego para Tester em Portugal. (13 de 09 de 2022). Obtido de Glassdoor: https://www.glassdoor.com/Job/portugal-qa-tester-jobs-SRCH_IL.0,8_IN195_KO9,18.htm?clickSource=searchBox

Resultados para a pesquisa de propostas de emprego para QA Tester em Portugal. (13 de 09 de 2022). Obtido de Indeed: https://pt.indeed.com/ofertas?q=qa%20tester&l=Portugal&from=searchOnHP&vjk=2f05b28964a3d75c

Resultados para a pesquisa de propostas de emprego para QA Manual Tester em Portugal. (13 de 09 de 2022). Obtido de Indeed: https://pt.indeed.com/ofertas?q=qa%20manual%20tester&l=Portugal&vjk=2f05b28964a3d75c

Resultados para a pesquisa de propostas de emprego para QA Automation Tester em Portugal. (13 de 09 de 2022). Obtido de Indeed: https://pt.indeed.com/ofertas?q=qa%20automation%20tester&l=Portugal&vjk=1bd821edd788fbc3

Ribeiro, Marcos J. (2022, Janeiro 12) , Prós e contras do Low Code/No Code, LinkedIn, https://www.linkedin.com/pulse/pr%C3%B3s-e-contras-do-low-codeno-code-marcos-j-ribeiro-msc/?originalSubdomain=pt

Robert, W., Agata, C., Maciej, K., Tadeusz, N., Zbigniew, W., & Kazimierz, W. (2016). Data flow between RFID devices in a modern restricted access administrative office. In MATEC Web of Conferences (Vol. 76, p. 04004). EDP Sciences.

Sánchez-Gordón, M. L., & Colomo-Palacios, R. (2018, September). From certifications to international standards in software testing: mapping from ISQTB to ISO/IEC/IEEE 29119-2. In *European Conference on Software Process Improvement* (pp. 43-55). Springer, Cham.

Sanchis, R., García-Perales, Ó., Fraile, F., & Poler, R. (2019). Low-code as enabler of digital transformation in manufacturing industry. Applied Sciences, 10(1), 12.

Schveitzer, A. (2018, outubro 3), Automação de testes: objetivos, abordagens, vantagens, desvantagens e limitações, LinkedIn, https://www.linkedin.com/pulse/automa%C3%A7%C3%A3o-de-testes-objetivos-abordagens-vantagens-e-alan-schveitzer/

Selenium Webdriver. (07 de 07 de 2022). Obtido de Tutorials Point: https://www.tutorialspoint.com/selenium_webdriver_with_java_novice_to_ninja_plus_interview/index.asp

Waszkowski, R. (2019). Low-code platform for automating business processes in manufacturing. *IFAC-PapersOnLine*, *52*(10), 376-381.

Waszkowski, R. (2017, July). Spread Page approach to document management. In International Conference on Applied Human Factors and Ergonomics (pp. 52-61). Springer, Cham.

Zolotas, C., Chatzidimitriou, K. C., & Symeonidis, A. L. (2018). RESTsec: a low-code platform for generating secure by design enterprise services. Enterprise Information Systems, 12(8-9), 1007-1033.