IPG Politécnico da Guarda
Escola Superior de Tecnologia e Gestão

RELATÓRIO DE ESTÁGIO

Curso Técnico Superior Profissional em
Testes de Software

Cátia Sofia Simão Regalado

junho | 2016

Escola Superior Tecnologia e Gestão

Instituto Politécnico da Guarda

RELATÓRIO DE ESTÁGIO

CÁTIA SOFIA SIMÃO REGALADO

Relatório para a Obtenção do Diploma de Técnico Superior Profissional

em Testes de *Software*

junho/2016

“Just because you've counted all the trees doesn't mean you've seen the forest."

Rameshwar Vyas

Agradecimentos

Com o culminar desta importante etapa, o presente relatório apenas foi possível devido ao apoio imensurável prestado pelas pessoas que me são mais próximas. Deste modo, quero prestar o meu enorme apreço e profundo agradecimento:

À minha mãe que esteve sempre presente, com a sua paciência e compreensão, com os seus conselhos e amor, tornando assim possível a concretização deste objetivo.

Ao César pelo seu amor, companheirismo e amizade, pela sua paciência e compreensão, pela sua ajuda em vários momentos deste relatório.

À Professora Doutora Maria Clara Silveira, orientadora deste relatório, pela sua sapiência, dedicação, incentivo, disponibilidade, empenho, conselhos, orientações, apoio e compreensão, que foram uma base importante e que contribuiram para a qualidade deste projeto.

Aos supervisores da Altran, Ângela Lázaro e João Ferreira, por todo o apoio e incentivo, pela transmissão dos seus conhecimentos, pelos seus conselhos e pelas suas advertências, que em muito irão contribuir para a minha futura carreira profissional.

A todos os professores pela sua disponibilidade, sabedoria e conhecimentos, que foram de extrema importância ao longo do curso e nesta etapa final.

A todos os colegas pela amizade e ajuda.

A todos os amigos pessoais que sempre tiveram uma palavra amiga, de força e encorajamento e pela compreensão da minha ausência.

A todos o meu sincero bem haja, que retribuo com a minha amizade.

Resumo

O presente relatório reporta o trabalho desenvolvido durante o estágio curricular no Curso Técnico Superior Profissional Testes de *Software*, que foi realizado na Altran Fundão, onde se desenvolveram as atividades descritas neste documento. O principal objetivo desta etapa foi realizar tarefas relacionadas com testes de *software*, nomeadamente a construção de casos de teste, realização de testes, bem como registar e reportar *bugs*.

Os testes de *software* são cada vez mais uma etapa essencial no ciclo de vida de um produto de *software*. Ao longo do desenvolvimento de um *software* podem surgir erros que apenas são detetados quando se realizam testes. É desta forma que se constroem *softwares* com qualidade e rigor.

O estágio curricular foi realizado em dois projetos diferentes: RSI, *Rural Servicios Informáticos*, e *AstraZeneca*. O cliente RSI, na área da Banca, pretende a construção de *WebServices*. Enquanto a *AstraZeneca*, cliente na área Farmacêutica, pretende o desenvolvimento de uma aplicação *mobile*. O modelo de desenvolvimento de *software* utilizado em ambos os projetos foi o modelo ágil.

Ao longo da realização do estágio curricular desenvolveram-se atividades de teste com o objetivo de verificar a existência de erros. Para tal, as tarefas desenvolvidas foram no âmbito da qualidade e testes com diferentes ferramentas e em diferentes níveis do processo. Neste sentido, foram realizados testes unitários, testes de carga, testes funcionais, testes de usabilidade e testes visuais. Ao longo destas atividades foram, igualmente, realizadas diversas reuniões com o intuito de gerir o processo de desenvolvimento, assim como partilhar informação e definir tarefas.

Palavras-chave: Testes de *Software*, Casos de Teste, Ferramentas de Testes, Automatização e Erros.

Índice

Índice de Imagens

Capítulo I – Introdução

1. Introdução

Este relatório reporta o trabalho desenvolvido durante o estágio curricular realizado na Altran Fundão compreendido no período de 1 de março a 30 de junho de 2016. São apresentadas e descritas as atividades desenvolvidas no projeto *Rural Servicios Informáticos* (RSI) e no projeto *AstraZeneca*.

Neste capítulo serão apresentados o enquadramento do estágio, os seus objetivos, bem como a estrutura do presente relatório. Este documento visa apresentar o trabalho desenvolvido no decorrer do estágio, salientando os pontos essenciais e que contribuíram para um crescimento progressivo.

1.1. Enquadramento

O estágio na Altran é parte integrante do currículo do Curso Técnico Superior Profissional Testes de *Software*. Esta fase é crucial na vida de um estagiário, é a etapa em que se tem a possibilidade de concretizar todo o conhecimento adquirido ao longo do curso, assim como consolidar aprendizagens e conceitos. É neste momento que se coloca em prática tudo o que foi ganho na teoria. É um processo de experimentação e concretização, assim como um período que possibilita o aumento do leque de novas aprendizagens profissionais e técnicas.

O estágio efetuado na Altran Portugal na sucursal do Fundão, decorreu do dia 1 de março ao dia 30 de junho de 2016.

1.2. Objetivos

O objetivo inicial do estágio é o de realizar testes de *software*. Este tipo de testes são de extrema importância, uma vez que erros existentes em determinados *softwares* podem causar problemas.

Após a chegada a esta etapa os objetivos a ter em conta são:

- ✓ Desenvolver e consolidar as aprendizagens adquiridas ao longo do curso Testes de *Software*.
- ✓ Conhecer a metodologia de desenvolvimento utilizada no projeto.
- ✓ Perceber quais as ferramentas de testes a serem utilizadas, conhecendo as linguagens de programação e a fase de desenvolvimento do projeto.
- ✓ Trabalhar com ferramentas de teste de *software* abordadas no curso.
- ✓ Aprender a trabalhar com novas ferramentas.
- ✓ Construir casos de teste.
- ✓ Realizar testes.
- ✓ Reportar possíveis erros.
- ✓ Perceber a verdadeira importância de testar e a sua eficácia.

1.3. Estrutura do Relatório

Este relatório apresenta uma introdução e encontra-se dividido em cinco capítulos:

A **Introdução** reflete o conteúdo deste relatório, nomeadamente o enquadramento do estágio, assim como aponta os seus objetivos.

O capítulo 2 **apresenta a empresa** que possibilitou a realização do estágio, assim como os projetos integrados durante o mesmo.

No capítulo 3, **trabalho desenvolvido no projeto RSI**, indica-se o trabalho executado ao longo do estágio no designado projeto, abordando momentos essenciais e incluindo exemplos, bem como apontando estratégias e lógicas utilizadas.

O capítulo 4 **trabalho desenvolvido no projeto *AstraZeneca*** apresenta o trabalho executado ao longo do estágio no referido projeto, indicando momentos essenciais e incluindo exemplos, assim como apontando estratégias e lógicas utilizadas.

Por fim, apresentam-se as **conclusões** retiradas ao longo do estágio, nomeadamente pontos importantes e dificuldades encontradas.

Capítulo II – Apresentação da Empresa e dos Projetos

2. Apresentação da Empresa e dos Projetos

Neste capítulo são apresentados a empresa e os projetos onde se desenvolveu o estágio curricular.

2.1. Caraterização da Empresa

Altran é uma multinacional francesa com mais de 24.000 colaboradores em mais de 20 países e com um leque de 500 importantes clientes em todo o mundo (Altran, 2016). A Altran surgiu em Portugal em 1998 e consolidou a sua marca em 2009. Atualmente, a Altran Portugal conta com mais de 800 colaboradores. A empresa tem sede em Lisboa e está presente no Porto e no Fundão, sendo que nesta última é apresentada como um *Global Delivery Centre*. Na Altran do Fundão existem mais de 250 colaboradores.

Este conceito consiste na prestação de serviços para determinadas empresas no estrangeiro, tendo em conta a distância e o fuso horário. O principal elemento diferenciador de Portugal, remete para a conjugação de vários fatores, nomeadamente baixo custo, qualidade das equipas, proximidade & acessibilidade, facilidade de comunicação em diferentes línguas e afinidade cultural. Todos estes fatores contribuem de forma decisiva para a deslocalização de projetos para o nosso país. Neste sentido, para os clientes Altran, a aposta no *Global Delivery Centre* Altran permite-lhes conjugar a redução de custos acompanhado por uma maior eficiência operacional, adquirindo, desta forma, benefícios.

A Altran apresenta 5 valores, dos quais faz uso perante os seus clientes de forma a prestar serviços com maior qualidade e rigor, sendo estes: ***Dynamism*** (dinamismo),

Excellence (excelência), ***Innovation*** (inovação), ***Care*** (consideração) e ***Responsibility*** (responsabilidade) (imagem 1). Os seus colaboradores, ao ingressarem na Altran, são imediatamente informados acerca destes valores de forma a contribuírem para o sucesso da empresa, bem como para completarem o seu perfil profissional, garantindo, igualmente, o seu sucesso e crescimento.

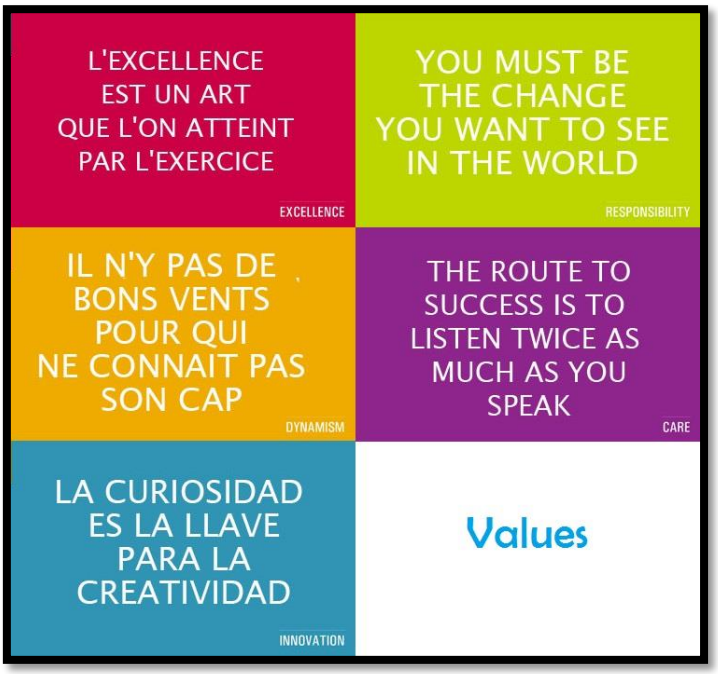

***Imagem 1 – Valores Altran (adaptado** (Altran, 2016))*

A Altran Portugal é, nos dias de hoje, um dos principais *players* na Consultoria de Inovação e Tecnológica em Portugal. São diversos os sectores de atividade em que atua, tais como: Financeiro, Telecomunicações & Media, Administração Pública, Indústria, *Energy & Life Sciences*, *Intelligent Systems* e *Utilities*. A atividade aqui desenvolvida tem uma estrutura assente na venda de soluções inovadoras.

A empresa possui um modelo de negócio diferenciado, sendo que a sua oferta assenta em quatro linhas de negócio: *Intelligent Systems*, *Information Systems*, *Lifecycle Experience e Mechanical Engineering*.

A Altran Portugal, no Fundão, sendo um centro de *Global Delivery Centre*, com competências para sistemas de informação, tem como objetivo servir o mercado nacional e internacional. Atualmente, na Altran Fundão existem dezenas de projetos em distintas

áreas, nomeadamente na área das telecomunicações (*Bouygues Telecom - Bytel*), da banca (RSI, *Axa Banque*), da saúde (*AstraZeneca*), entre outras.

2.1.1. Direção/Comité Executivo

A Altran Portugal é constituída pelo seguinte comité executivo Altran Portugal, tal como se pode verificar na imagem seguinte (imagem 2):

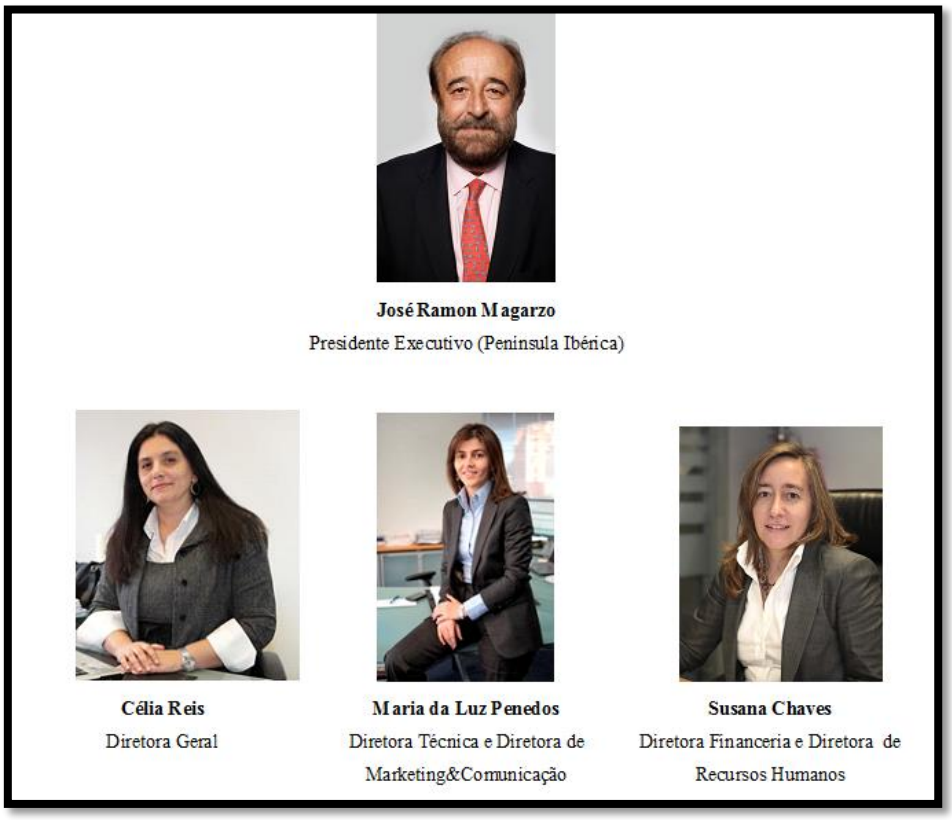

Imagem 2 – Comité Executivo Altran Portugal

2.1.2. Meio envolvente

A Altran situa-se no Centro de Negócios e Serviços do Fundão. O Fundão é uma cidade concelho, pertencente ao distrito de Castelo Branco. Situa-se na região centro, mais propriamente na Cova da Beira, com uma área de 700 Km^2, na qual se distribuem 23 freguesias, possuindo 30.000 habitantes no seu total.

As suas atividades económicas são mais direcionadas para a agricultura, comércio e serviços. No entanto, os executivos que têm governado esta cidade têm criado condições

de forma a chamar população para aqui trabalharem e viverem, nomeadamente a integração da Altran na cidade, acompanhada com a renovação da cidade construindo condições e proporcionando qualidade de vida.

2.2. Contexto dos Projetos

Neste ponto procede-se à apresentação, sucinta, dos projetos em que foi desenvolvido o estágio: *Rural Servicios Informáticos* (RSI) e *AstraZeneca*.

2.2.1. RSI – *Rural Servicios Informáticos*

RSI é um cliente da área da banca espanhola que, presentemente, pretende construir *WebServices* para uma melhor qualidade e facilitação na prestação dos seus serviços.

Neste projeto participam a Altran Portugal, Altran Espanha, *Inversis*, RSI – Cliente. O cliente RSI empresa na área da banca espanhola, invocou serviços à Altran Espanha que, por sua vez, solicitou colaboração à Altran Portugal. Este projeto tem uma dinâmica própria, sendo que os colaboradores interagem da forma como se verifica na imagem 3.

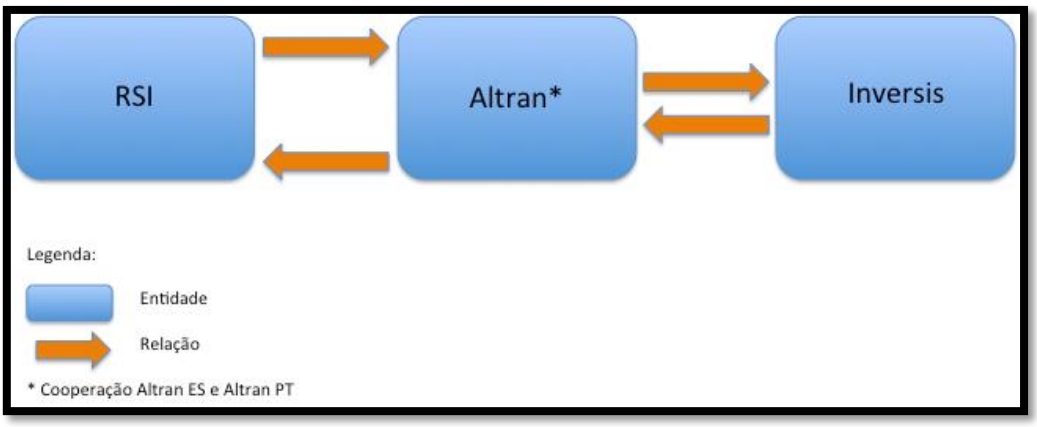

Imagem 3 – Dinâmica do Projeto

Neste sentido, percebeu-se que o principal objetivo da Altran é a criação de *SOAP WebServices (SOAP - Simple Object Access Protocol,)*, sendo que a Altran por sua vez, recorre aos serviços da *Inversis* que nos envia dados necessários. Neste seguimento,

podemos aferir que a RSI invoca os nossos serviços (Altran) e, seguidamente, nós invocamos os serviços da *Inversis* de forma a dar resposta à RSI.

2.2.2. AstraZeneca

AstraZeneca é um cliente da área farmacêutica que se encontra a desenvolver uma aplicação (app) *mobile, para Android e iOS,* para monitorizar a toma de um determinado medicamento, por parte de doentes com cancro do pulmão.

O trabalho a desenvolver neste projeto reflete-se ao nível do desenvolvimento da app *mobile* em sistema *Android* e em sistema *iOS*, assim como realizar testes relativos à mesma e reportá-los. A dinâmica da aplicação está ilustrada na imagem seguinte (imagem 4):

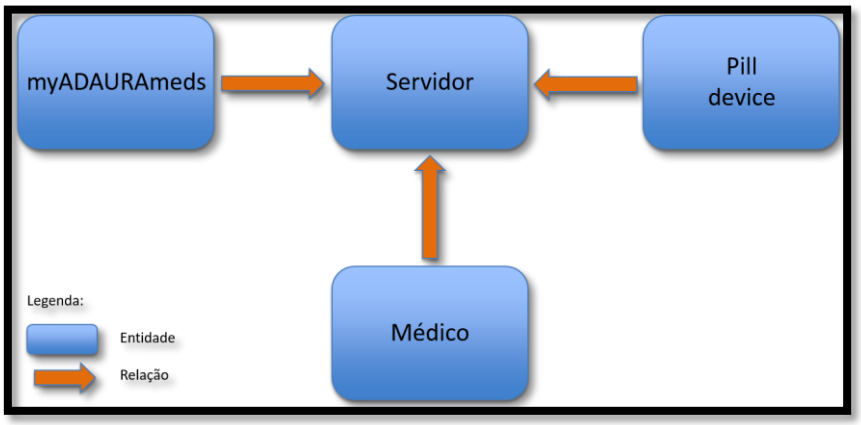

Imagem 4 – Dinâmica do Projeto

Neste sentido, a app *myADAURAmeds* faz comunicação com o servidor de forma a informar se o paciente tomou o medicamento ou não. Por outro lado, o *pill device* comunica, igualmente, com o servidor com o intuito de indicar se a caixa foi ou não aberta, supondo, assim, que o medicamento foi tomado ou não. No entanto, a resposta que será validada e considerada pelo médico, é sempre a que é dada na app mobile. Ou seja, se o doente abrir o *pill device* e tomar o medicamento, mas não registar a toma na aplicação, o médico vai interpretar que o paciente não tomou o medicamente. Caso contrário, se a toma for registada na app, o médico vai considerar que o medicamento foi tomado. Outro caso a ser considerado é se o paciente não abrir o *pill device* e na aplicação informar que o medicamento foi tomado, esta última é a resposta a ser validada.

2.3. Metodologia seguida nos Projetos – Modelo *Agile*

No projeto RSI e no projeto *AstraZeneca* a metodologia seguida é a metodologia ágil (Modelo *Agile*). O Modelo *Agile* foi definido por Edmons em 1974, no entanto, esta metodologia ficou aprovada em 2001 com o manifesto ágil (Manifesto for Agile Software Development, 2001). A metodologia ágil tem como principal visão a entrega (rápida) do produto.

O modelo ágil é iterativo e incremental, ou seja, trabalha por iterações, por curtos períodos. O desenvolvimento do *software* é feito em curtos períodos de tempo, são gastas, tipicamente, entre 1 a 4 semanas, minimizando assim possíveis riscos.

Em cada incremento existe, de igual forma, análise de requisitos, desenho e análise, implementação e desenvolvimento, testes e avaliação/priorização (imagem 5).

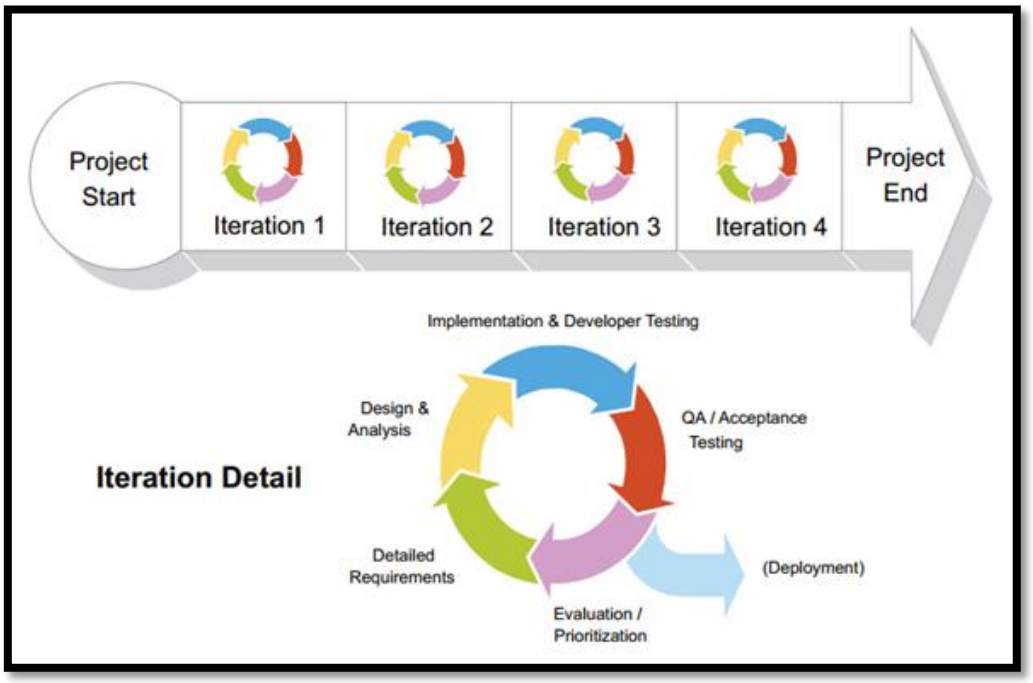

Imagem 5 – Modelo Agile *(James, 2016)*

Todas estas fases são incrementadas pelo menos uma vez. Contudo, podem ser realizadas mais que uma vez, já que este modelo é implementado em pequenas iterações. A primeira fase é a **definição de requisitos**, a segunda **análise e desenho**, a terceira fase **implementação e desenvolvimento**, a quarta fase é relativa à realização de **testes** e por último, a quinta fase, a **avaliação**.

No que se refere à fase de definição de requisitos, nesta definem-se os requisitos, indicando os objetivos, as funcionalidades e as restrições de um *software*, de acordo com as necessidades do utilziador. Os requisitos podem ser **funcionais** ou **não funcionais**. Relativamente aos testes funcionais, estes referem-se ao que o sistema deve fazer, ou seja, indicam as suas funcionalidades. Quanto aos requisitos não funcionais, definem qualidades globais ou atributos do sistema, assim como as restrições. Podem ser requisitos de integridade, segurança, usabilidade, fiabilidade e desempenho.

Quanto à fase de análise e desenho do *software*, são analisados os requisitos e construído um protótipo do que virá a ser o *software*, implementando todos os requisitos definidos. Contudo, este protótipo terá que ter já um aspeto de acordo com o que o cliente pretende.

Relativamente à fase de implementação e desenvolvimento, é nesta etapa que o *software* é desenvolvido. Neste sentido, os *developers* devem escrever código, construindo assim o *software* e devem, igualmente, efetuar testes ao mesmo. Este tipo de testes executados são testes unitários.

Referente à fase de testes, os *testers* têm ao seu encargo realizar os testes necessários ao *software* já desenvolvido, de forma a encontrar erros. Com esta atividade pretende-se detetar os *bugs* existentes com o objetivo de serem corrigidos e ultrapassados, desenvolvendo assim um *software* com qualidade.

Quanto à fase de avaliação, aqui o cliente vai avaliar o que está feito, dando o seu feedback, indicando se aprova o que se encontra realizado até àquele momento, ou, se por outro lado, reprova algum ponto ou vários, e o processo terá de se iniciar novamente de forma a garantir a sua satisfação.

Uma vez que, este modelo atribui especial atenção à fase de testes, assim como é de todo o interesse satisfazer as necessidades dos utilizadores, e realçando aqui essa fase, é necessário evidenciar a importância das atividades de teste. Desta forma, importa aqui refletir sobre a importância de testar.

No mundo atual estamos rodeados por tecnologia que depende de um *software* para funcionar. Sempre que ouvimos falar em *software* surge-nos a ideia de algo que vai funcionar de forma correta e com o intuito de nos facilitar. No entanto, nem sempre verificamos isso. Quando isso acontece é sinal de que algo falhou e que o *software* não

tem qualidade. Isto verifica-se quando as empresas não dão a devida importância à realização de testes ao *software*.

Os testes de *software* fazem parte de um processo de garantia de qualidade em que é possível medir a qualidade do *software* em termos de defeitos relativamente a requisitos funcionais e não funcionais. Um erro cometido no desenvolvimento de um *software* resulta num defeito nesse produto.

Cada vez mais é necessário lançar para o mercado aplicações de qualidade e eficazes, de forma a satisfazer os seus utilizadores. Isso apenas é possível se as aplicações não tiverem erros no seu *software*. Para tal, é necessário executar testes ao *software* com o intuito de encontrar erros e seguidamente corrigi-los para garantir a qualidade e o sucesso da aplicação. E quanto mais cedo forem detetados os erros, menores serão os custos. Por conseguinte, é aconselhável realizar os testes logo nas primeiras fases do desenvolvimento.

No entanto, as atividades de teste não são isoladas, elas estão relacionadas com as atividades de desenvolvimento de *software*. Os testes estão inseridos num modelo de desenvolvimento onde se indicam todos os passos a ter em conta, nomeadamente os testes a serem realizados a nível dos requisitos, ou seja a nível das caraterísticas funcionais e não funcionais e o seu resultado. Para testar são necessárias ferramentas de teste que permitem analisar o *software*, e é igualmente necessário construir casos de teste para definir os testes a realizar e a que funcionalidades ou campos, indicando valores (válidos e inválidos). Uma das grandes vantagens de testar é o facto de ser muito mais viável realizar testes, encontrar erros e corrigir os mesmos, do que ter de os corrigir já na fase de produção.

Existem dois tipos de testes: **caixa branca** utilizada maioritariamente pelos *developers* e que incide apenas no código e **caixa preta** em que os *testers* não têm acesso ao código, tendo em conta apenas a *interface*. Por conseguinte, existem quatro níveis de testes: **testes unitários** que testam se cada funcionalidade especificada na codificação e desenho do *software* foi implementada corretamente; **testes de integração** que são utilizados para garantir que a comunicação entre estes dois produtos seja implementada como especificado na codificação, desenho e arquitetura do sistema; **testes de sistema** realizados após todos os testes de integração, uma vez que validam o sistema como um todo, podendo analisar os requisitos funcionais e não funcionais, como desempenho,

volume, documentação, robustez; **testes de aceitação** que auxiliam na verificação do *software* para que o mesmo entre ou não em produção. É através destes procedimentos que se garante o sucesso de uma aplicação, testando ao máximo, detetando erros e corrigindo-os.

Após esta referência, essencial, à neccessidade da execução de testes, importa ainda referir que a metodologia ágil assenta sobre os seguintes valores (Pinheiro, RepositóriUM, 2015):

- **indivíduos e interações mais** do que processos e ferramentas;
- ***software* funcional mais** do que documentação abrangente;
- **colaboração com o cliente mais** do que negociação contratual;
- **responder à mudança mais** do que seguir um plano.

No entanto, isto não significa que serão colocadas de lado a documentação, as negociações ou um plano, mas sim que se deve dar maior importância ao *software* e à necessidade de dar resposta imediata ao cliente.

Um dos pontos fulcrais na equipa que desenvolva *software* com metodologia ágil prende-se com a auto-organização e motivação constante por parte dos colaboradores. O facto de não existir um plano rígido acontece devido às constantes mudanças que ocorrem ao longo do desenvolvimento de um projeto. Neste sentido, fica facilitada a resposta imediata ao cliente de forma a satisfazer as suas necessidades sem comprometer o *software*.

Para que esta metodologia funcione é de extrema importância que a equipa e o cliente comuniquem na perfeição. Isto devido ao facto de não existir muita documentação e uma vez que as mudanças são constantes e imprevisíveis será sempre necessário, continuamente, diálogo e *feedback* de ambas as partes.

Da mesma forma que apresenta valores, este modelo também ostenta princípios, sendo estes doze (Costa, s.d.):

1) A nossa maior prioridade no desenvolvimento é satisfazer o cliente através da **entrega contínua e adiantada de *software*** com valor agregado.

2) **Mudanças nos requisitos são bem-vindas**, mesmo tardiamente no desenvolvimento. Processos ágeis tiram vantagem das mudanças visando vantagem competitiva para o cliente.

3) **Entregar frequentemente *software* operacional**, com uma frequência de poucas semanas a poucos meses, com preferência à menor escala de tempo.

4) Pessoas de negócio e programadores devem **trabalhar diariamente em conjunto**.

5) Construir projetos em torno de **indivíduos motivados**. Disponibilizar-lhes o ambiente e o suporte necessário e confiar neles para fazer o seu trabalho.

6) O método mais eficiente e eficaz de transmitir informações para e entre uma equipa de desenvolvimento é através de **conversa cara a cara**.

7) ***Software* a funcionar** é a medida primária de **progresso**.

8) Os processos ágeis promovem **desenvolvimento sustentável**. Todos os intervenientes devem ser capazes de manter um ritmo constante indefinidamente.

9) Atenção contínua à **excelência técnica e um bom *design*** aumenta a agilidade.

10) **Simplicidade**, a arte de maximizar a quantidade de trabalho não realizado, é essencial.

11) As melhores arquiteturas, requisitos e *designs* emergem de **equipas flexíveis**.

12) Em **intervalos regulares**, a equipa reflete sobre como se tornar mais eficaz, **ajustando o seu comportamento**.

Capítulo III – Trabalho Desenvolvido no Projeto RSI

3. Trabalho Desenvolvido no Projeto RSI

Neste ponto procede-se à apresentação de como se realizou a integração no projeto, assim como se descrevem as atividades desenvolvidas no percurso feito no estágio curricular.

3.1. Contexto de Integração no Projeto RSI

RSI – *Rural Servicios Informáticos* – é um cliente da área da Banca em Espanha. Este cliente pretende construir *WebServices*. Na área de testes, as atividades a executar são:

- ✓ Testes unitários, testes de integração e testes funcionais de *Web Services SOAP*;
- ✓ Utilização de ferramentas como:
  - *SoapUI* para invocação direta dos serviços e criação de casos de testes automatizados;
  - *JUnit* integrado no *IDE Eclipse* para automatização de testes.
- ✓ Em *WebSites*, testes funcionais e automatização de testes por meio de uso de ferramentas como *Selenium* e *JUnit*.

Os objetivos principais deste projeto remetem para a construção de *WebServices* com ligação a outros serviços, assim como testar os mesmos. Com a pretensão de construir *WebServices*, o cliente solicitou o desenvolvimento dos mesmos. Para tal, é necessário recorrer à Inversis para disponibilizar os dados necessários.

O primeiro passo dado nesta etapa foi tomar conhecimento do projeto, iniciando os trabalhos com uma breve conversa, de forma a fazer a integração no projeto e conhecer qual o principal objetivo. Em termos de documentação, apenas foi disponibilizada a análise funcional, a qual foi lida, percebida e, posteriormente, esclarecidas dúvidas junto da *team leader* Ângela Lázaro. Neste documento constam os serviços que o cliente pretende desenvolver, bem como as suas funcionalidades e mapeamento. Neste sentido, pode-se depreender, tal como já referido, que o projeto RSI remete para a criação de vários *WebServices*. *WebService* é uma tecnologia baseada em XML e HTTP cuja principal função é disponibilizar serviços interativos na *Web* que podem ser acedidos ou consumidos por qualquer outra aplicação independente da linguagem ou plataforma em que a aplicação foi construída (Caetano, s.d.). Os *WebServices* permitem às aplicações enviar e receber dados em formato XML. Para as empresas, os *WebServices* podem trazer agilidade para os processos e eficiência na comunicação entre cadeias de produção ou de logística. Toda e qualquer comunicação entre sistemas passa a ser dinâmica e principalmente segura, pois não há intervenção humana (O que é Web Service?, 2007).

Os *WebServices* que estão em construção para o cliente RSI correspondem aos serviços *historicoMovimientos*, *operacionesPendientes*, *cancelacionOrden*, *configuracion* e *posicionGlobal*. Cada serviço é dependente de um ou mais serviços da *Inversis*.

O serviço *historicoMovimientos* retorna os movimentos de um cliente e/ou entidade e é dependente dos serviços *consultarOrdenesMB* e *buscadorTraspasosMB* da *Inversis*, (imagem 6).

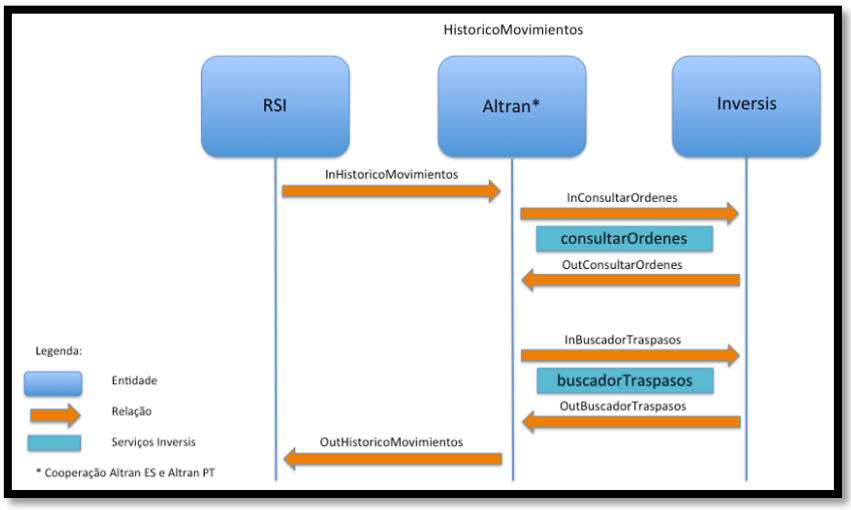

Imagem 6 – Dinâmica Serviço historicoMovimientos

O serviço *operacionesPendientes* retorna operações pendentes de um cliente e/ou entidade e depende dos serviços *consultarOrdenesMB* e *buscadorTraspasosMB* da *Inversis* (imagem 7).

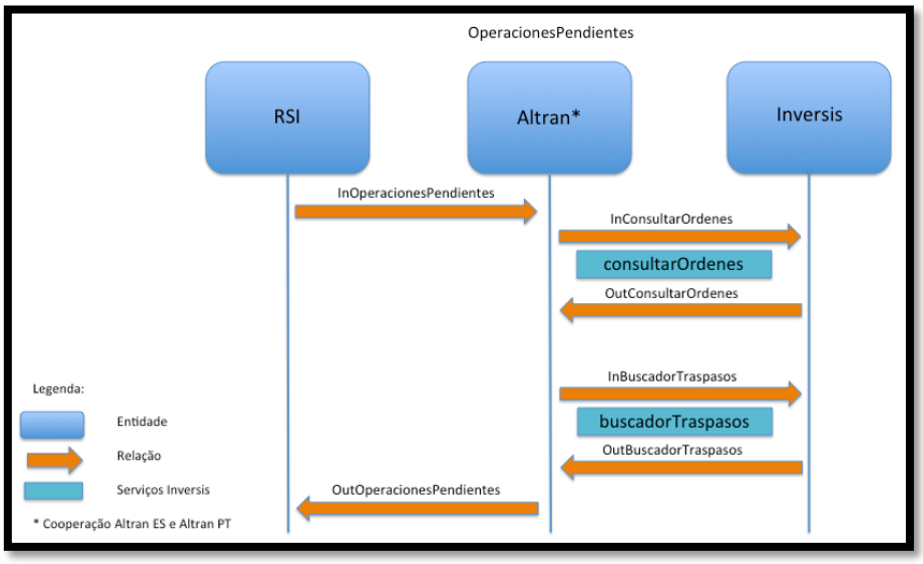

Imagem 7 - Dinâmica Serviço operacionesPendientes

O serviço *cancelacionOrden* retorna o cancelamento de uma ordem/pedido, enviando 3 possibilidades: *Suscripcion*, *Reembolso* e *Rebalanceo* e este serviço depende da *entradaOrdenesMB* para *suscripción* e *reembolso* e da *altaTraspasos* para *rebalanceo* (imagem 8).

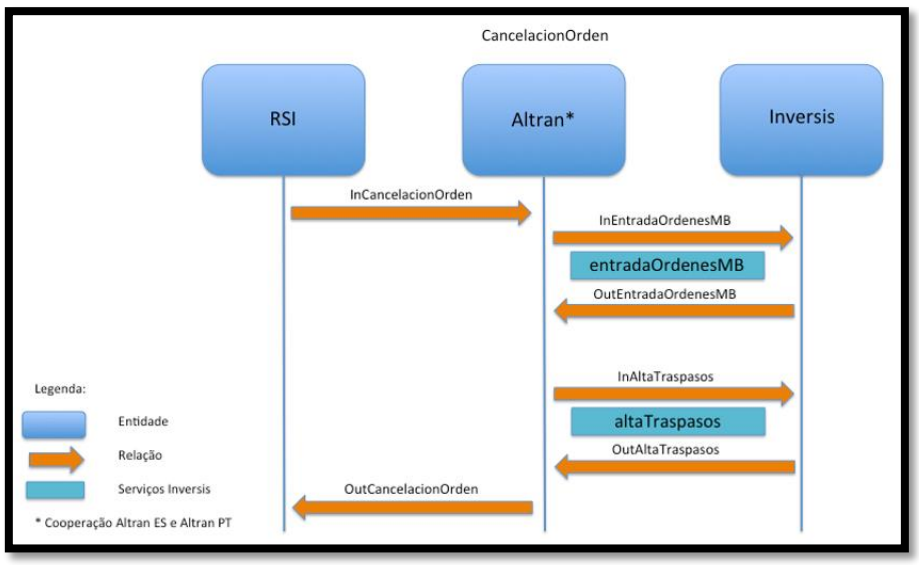

Imagem 8 - Dinâmica Serviço cancelacionOrden

O serviço *configuracion* retorna um conjunto de dados/informações sobre a configuração existente no *Fondo Inteligente* e depende do serviço *detalleFondoInteligente* da *Inversis* (imagem 9).

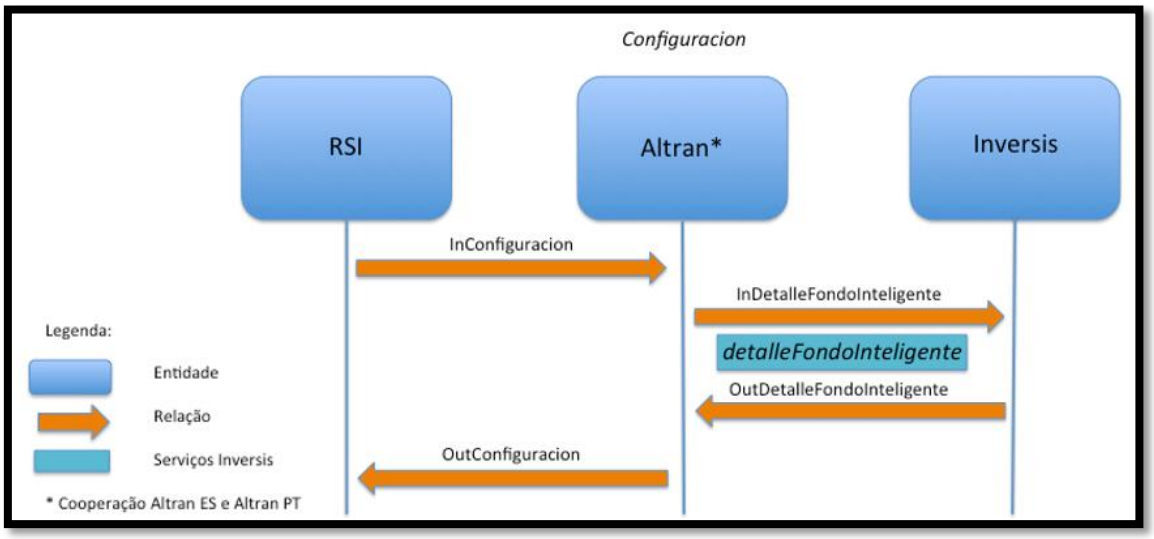

Imagem 9 - Dinâmica Serviço configuracion

O serviço *posicionGlobal* e depende dos serviços *consultarDetallePosicionSaldoMB* e *simulacion_Orden_FI* da *Inversis* (imagem 10).

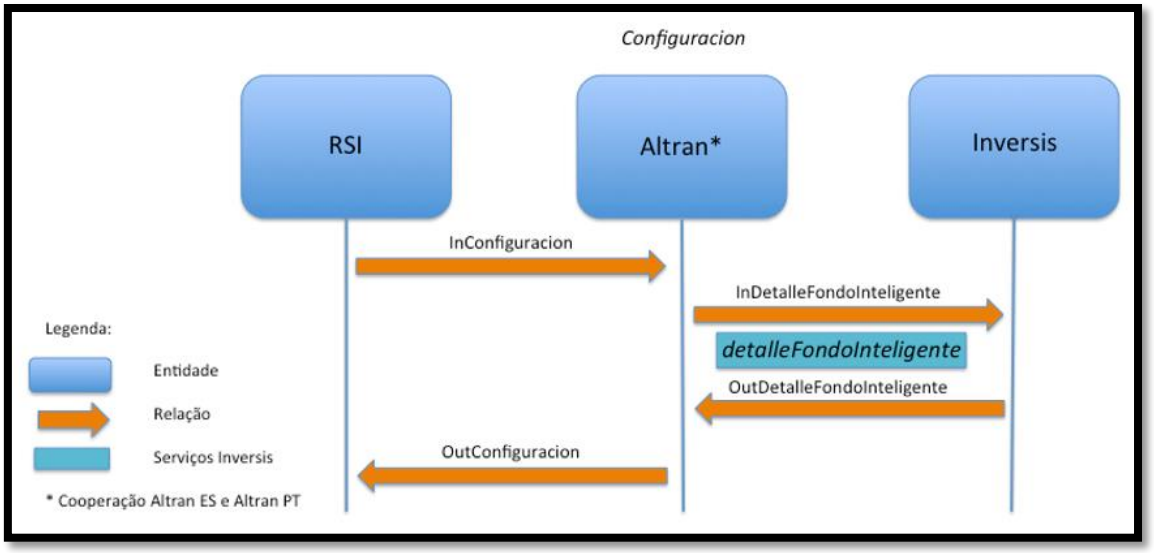

Imagem 10 - Dinâmica Serviço posicionGlobal

A equipa afeta a este projeto é composta por 8 colaboradores: 1 gestor de projeto, 4 *developers* e 1 *tester* na Altran Portugal, sendo que 1 dos programadores é *team leader* e 2 *developers* na Altran Espanha.

3.2. Atividades Desenvolvidas

O estágio neste projeto decorreu desde o dia 1 de março e terminou no dia 06 de maio do presente ano. Na primeira semana de estágio as atividades cingiram-se à integração na empresa através do *Induction Day* e ao conhecimento do projeto. *Induction Day* é uma atividade com duração de 1 dia em que se apresenta a empresa: a sua história, estrutura, organização e resultados. Desta forma foi possível conhecer a missão da empresa, a sua dinâmica, assim como alguns colaboradores da Altran.

A etapa seguinte foi o conhecimento do projeto a ser integrado. RSI – *Rural Servicios Informáticos* – é uma empresa do grupo *Caja Rural* da área da banca espanhola responsável pela definição e implementação da estratégia comum em todas as questões relativas ao tratamento automatizado de informações dos bancos rurais.

Após integração no projeto e tomada de consciência dos seus objetivos, procedeu-se à instalação do *SoapUI* uma vez que, inicialmente, esta é a ferramenta a utilizar nos testes a realizar. A versão instalada foi *SoapUI OpenSource* 5.2.1 (imagem 11).

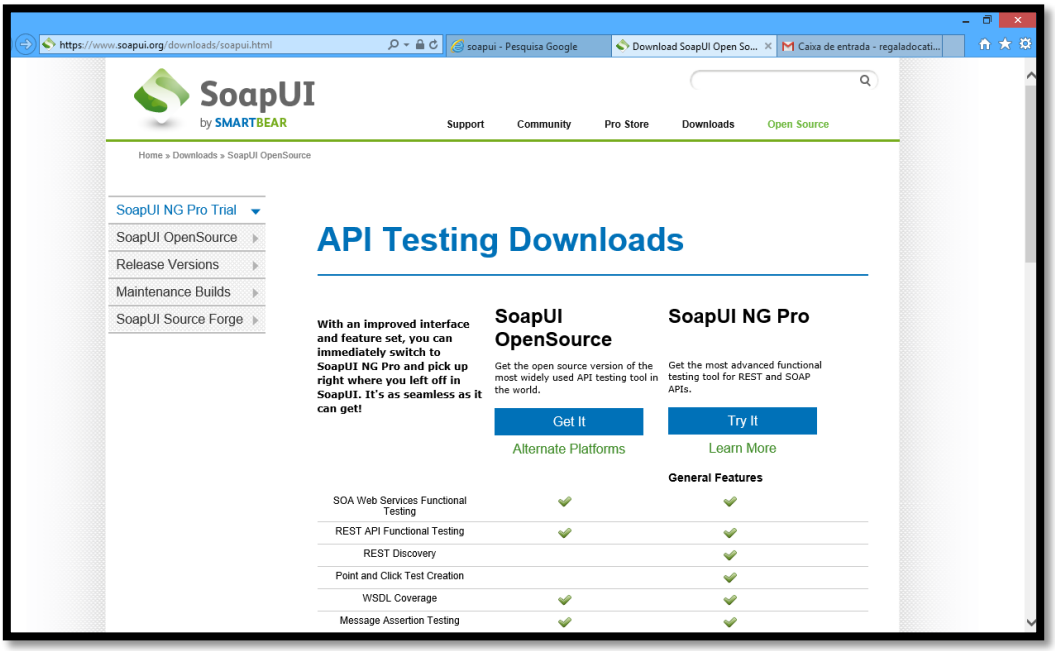

Imagem 11 - SoapUI OpenSource *5.2.1*

Passos da instalação:

1) Descarregar executável
    a) Aceder à página oficial do SoapUI (https://www.soapui.org/)
    b) Clicar no menu *Downloads*
    c) Clicar em *Download SoapUI*
    d) Escolher *SoapUI OpenSource*
    e) Clicar em *Get it*
2) Guardar o executável na pasta *downloads*
3) Correr o executável e clicar sempre em ok.

Após a instalação abriu-se a ferramenta e surgiu uma janela em que mostra apenas menus, *links* que remetem para informação sobre o *SoapUI* e surgiu ainda a árvore de *Projects* vazia. Começou-se por explorar os menus que existiam, bem como as opções existentes a partir da opção *Project.* Foi a partir daqui que se iniciou a criação de um novo projeto. Para tal, clicou-se com o botão direito do rato em cima de *Project* e escolheu-se a opção *New Soap Project*. Atribuiu-se um nome ao *project*, neste caso TesteCriarProjeto, e colou-se no *inicial WSDL* o endereço disponibilizado pela *team leader* (que está publicado no servidor local) e ok.

Com o SoapUI instalado e devidamente explorado realizaram-se testes funcionais, com o intuito de verificar se o mapeamento estava correto, relativamente aos serviços *historicoMovimientos*, *operacionesPendientes*, *cancelacionOrden*, *configuracion* e *posicionGlobal*. Neste seguimento, foram executados testes em que se chamavam os serviços e se verificava que eles respondiam, e quando se inseria um valor num determinado campo iria surgir o mesmo valor no outro campo correspondente. Para tal, eram inseridos valores, válidos e/ou inválidos, nos respetivos campos de forma a verificar os resultados. Um dos exemplos foi no serviço *historicoMovimientos*, relativamente ao campo *descripcionError* a mensagem ali escrita teria de aparecer no campo *mensajeMostrar* (imagem 12).

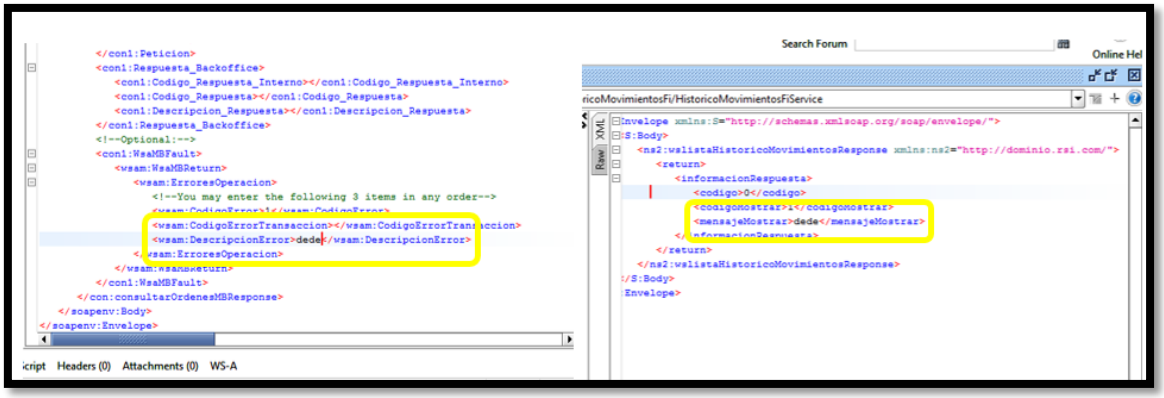

Imagem 12 – Exemplo de Mapeamento: Serviço historicoMovimientos

Um outro exemplo de mapeamento, é que se pode verificar na imagem seguinte (imagem 13), relativo ao serviço *positionGlobal*. Quando se atribuia valores válidos nos campos de entrada, recebiam-se valores nos respetivos campos de resposta. Neste sentido, para a conta com *codigoInterno* = 1575794, *codigoCuentaValoresExterno* = 02391000395580, *entiEntrCom* = 1987723, *codigoIsin* = 1234567, *tipoInstrumento* = MODERADO e *codigoIsin* = 0987654, *tipoInstrumento* = DINAMICO, são apresnetados os valores correspondentes a essa conta.

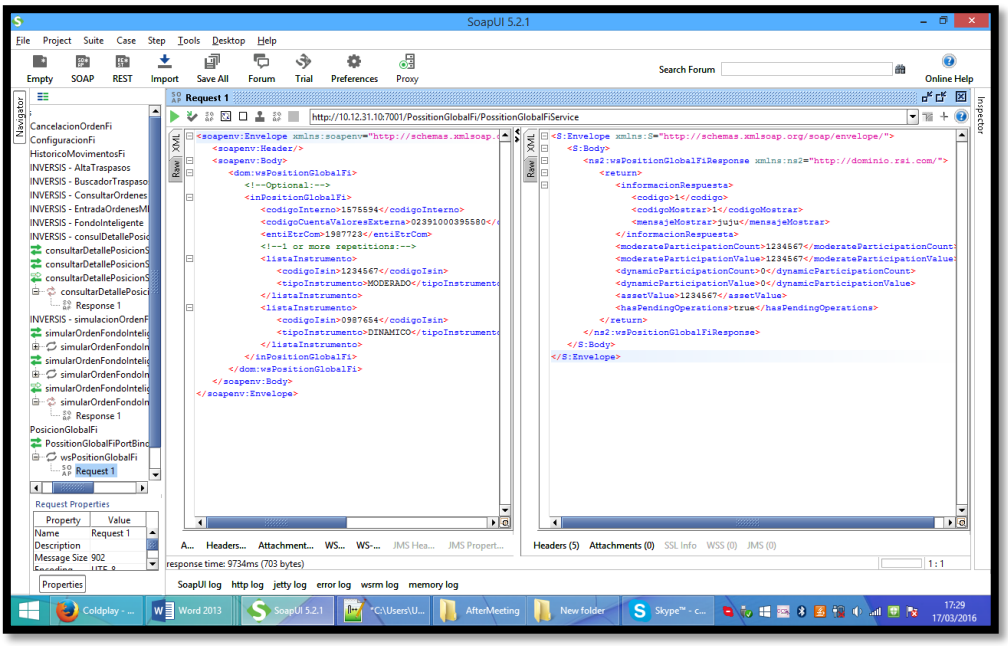

Imagem 13 - Exemplo de Mapeamento : Serviço positionGlobal

Para a realização dos ditos testes foi necessário proceder à construção de casos de teste. No decorrer dos testes foram detetados alguns erros, sendo que um deles foi verificado na criação do projeto do serviço *operativaAlta*. O que se verificou foi que a ligação desse serviço não estava correta uma vez que abria o serviço *posicionGlobal* (imagem 14).

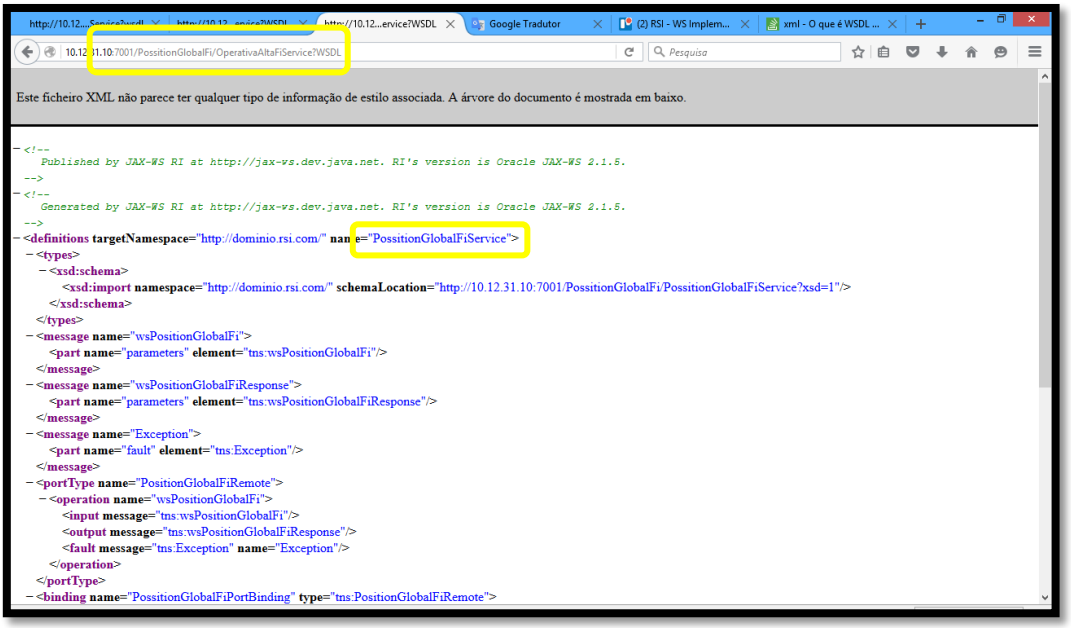

Imagem 14 – Erro no Serviço operativaAlta

Este erro foi reportado aos *developers* e corrigido.

Depois de verificar que o mapeamento estava correto e após o desenvolvimento do projeto passou-se à automatização dos testes. Ao longo dos trabalhos houve alterações na análise funcional realizadas pela Altran ES e a *Inversis*. Desta ação foi necessário reanalisar este documento de forma a perceber quais as mudanças feitas, assim como entender as novas interações. Com estas alterações houve a necessidade de serem feitas alterações no código. Nesse seguimento, também os testes foram novamente realizados de forma a testar as modificações. Como exemplo, efetuou-se a alteração da designação do campo *datosOperacion* para *ordenAnulable* do serviço *operacionesPendientes* (imagem 15).

Imagem 15 – Alteração dos Campos na Análise Funcional

Ao longo da realização dos testes foram feitas algumas observações, tais como no preenchimento do campo *fechaDesde* com valor superior ao valor do campo *fechaHasta* o que surgiu foi uma lista vazia e não erro. Esta situação acontece porque tem mais lógica surgir erro na camada superior, isto com o intuito de evitar tráfego desnecessário neste nível. Contudo, foram colocadas nas observações uma questão para o cliente: na nossa camada é para validar? Ou é só na camada superior (*front-end*)? (apêndice 1). Este exemplo está retratado no exemplo seguinte (imagem 16).

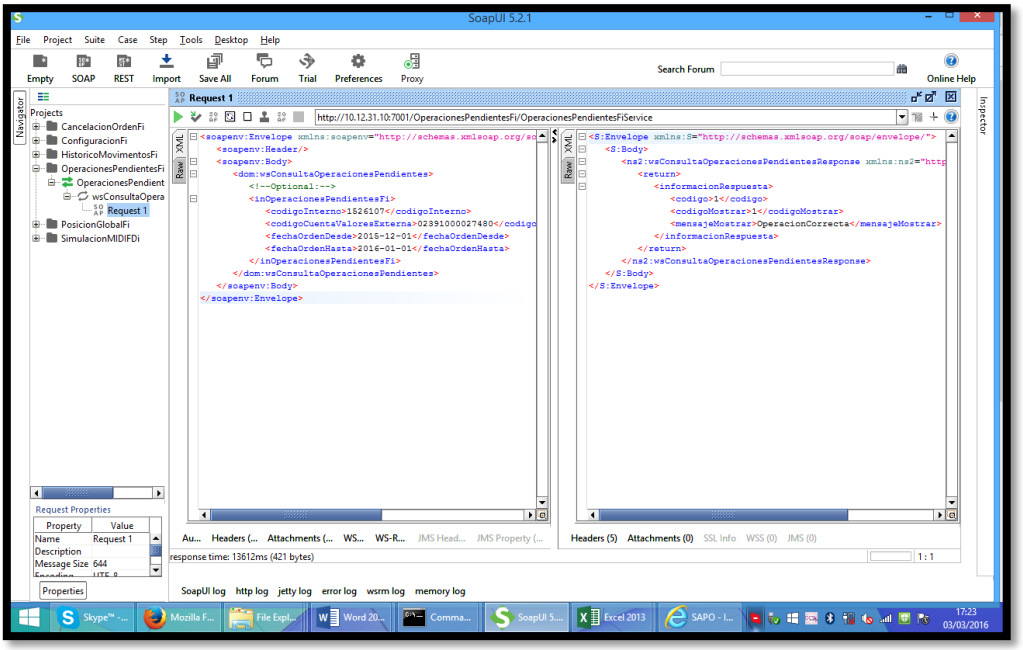

Imagem 16 – Exemplo de Teste no SoapUI

Outros testes realizados foram testes unitários e funcionais, mais concretamente nos serviços *historicoMovimientos* e *operacionesPendientes*, em que foram testados o campo *estado* e *estadoOrden*, tendo sido atribuídos todos os valores válidos que recebem, assim como valores inválidos.

Foram igualmente realizados testes aos campos *Date* e *Cancelable* dos mesmos serviços, sendo que foi testado se as datas estavam a ser retornadas por ordem da mais atual para a mais antiga e verificar se o campo *anulable* (resposta ao campo *Cancelable*) devolve o valor correto.

Durante a realização dos testes foram detetados alguns erros e reportados de imediato aos *developers.* Um desses erros encontrados foi no preenchimento do campo *anulable* do *buscadorTraspasos* em que ao preencher o mesmo campo com S deveria retornar *true* e não foi esse o resultado. A resposta foi *false* (imagem 17). Após a sua correção pelos *developers* repetiram-se os testes, sendo verificado que os erros ficaram corrigidos.

Imagem 17 – Erro no Serviço buscadorTraspasos

No decorrer do trabalho foram feitas algumas pesquisas sobre alguns conceitos necessários e presentes no desenvolvimento do mesmo, tais como testes funcionais e testes unitários. Quanto aos primeiros pode-se afirmar que os testes funcionais permitem realizar testes de uma forma mais rápida e eficiente, possibilitando encontrar as não

conformidades do *software* em relação aos requisitos do sistema. Estes testes permitem testar as funcionalidades, requisitos presentes na documentação, de forma a validar as funcionalidades descritas.

Quanto aos testes unitários, estes testam uma única unidade do sistema de maneira isolada, geralmente simulando as prováveis dependências que aquela unidade tem. Estes testes são realizados com o intuito de validar dados válidos e inválidos através de I/O (*In/Out* – entrada/saída). Um teste unitário, especificamente, testa aquele e somente aquele método, de forma a evitar acesso a outros recursos como sistema de arquivos, banco de dados, rede, entre outros. Para tal devemos fazer uso de *mocks*.

Durante este período foram realizadas pesquisas sobre conceitos importantes e necessários ao desenvolvimento do trabalho, tais como: *WebServices*, WSDL, XSD, XML, *SoapUI*, *Mock*, servidor, entre outros. Realizou-se também uma pesquisa mais aprofundada acerca do funcionamento do *SoapUI*.

Com o evoluir do trabalho surgiu a necessidade de utilizar *assertions* para realizar testes de automação mais específicos. Neste seguimento, desenvolveu-se uma pesquisa com o intuito de perceber como realizar testes no *SoapUI* utilizando *assertions*.

A seguir, iniciou-se a exploração e experimentação com diferentes tipos de *assertions* de forma a perceber a sua funcionalidade e propósito. Depois de concluído o reconhecimento iniciaram-se os testes relativos ao projeto, utilizando *assertions*, nomeadamente *contains* e *XPath Match*.

Foram realizados testes a todos os serviços, tendo em conta os campos que cada um contém. No decorrer dos mesmos foram detetados alguns erros, os quais foram reportados, de imediato, aos *developers* e corrigidos instantaneamente. Um dos erros identificados cingiu-se ao preenchimento correto do *request* e do *response*, no serviço *configuracion*, tendo em conta a análise funcional, sendo que o resultado não foi o esperado, uma vez que surgiu uma lista de código com erros (imagem18). Depois de revisto pelo *developer* o erro encontrava-se numa variável que não estava inicializada.

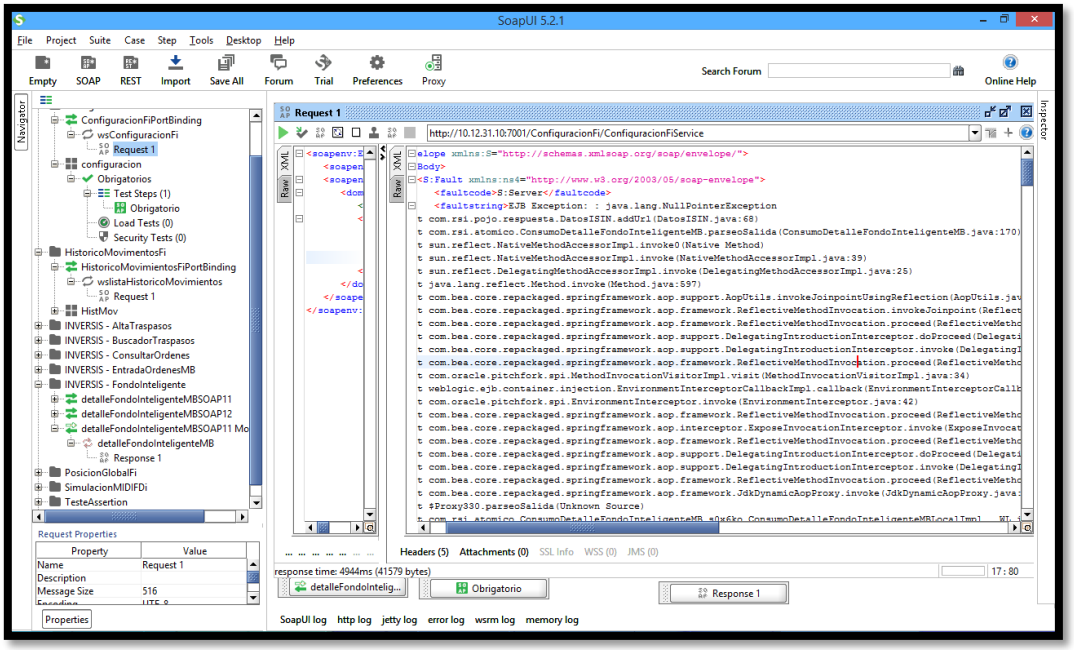

Imagem 18 – Erro no Serviço configuracion

Os restantes testes executados correram conforme o esperado, não se verificando erros.

As assertions são utilizadas para comparar partes ou o todo do que é obtido com o que é expectável, tanto a nível de valores como de formato. Ou seja, as *assertions* são validações feitas para confirmar se a resposta de uma requisição contém as informações esperadas. As *assertions* utilizadas foram: *contains* e *XPath Match configuration*. *Contains* especifica a condição prevista baseada na resposta, ou seja verifica se contém o valor, se aquele valor está presente. *XPath Match configuration* valida de acordo com uma expressão *XPath* o formato do campo, sendo que é indicado o caminho para o campo. As *assertions* foram criadas em todos os serviços da RSI (imagens 19 e 29).

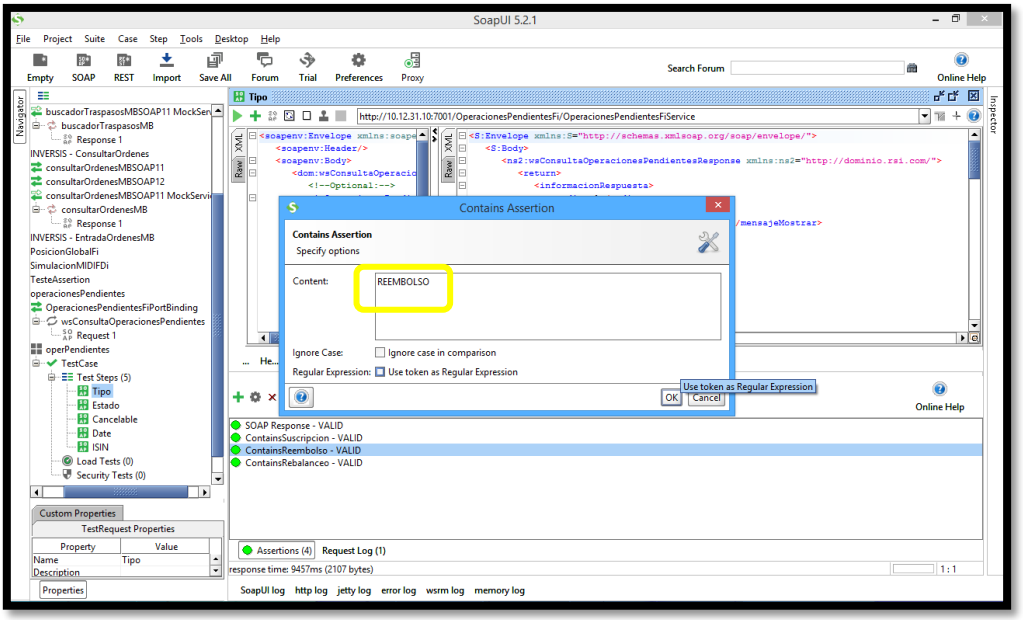

Imagem 19 – Exemplo Assertion: Contains

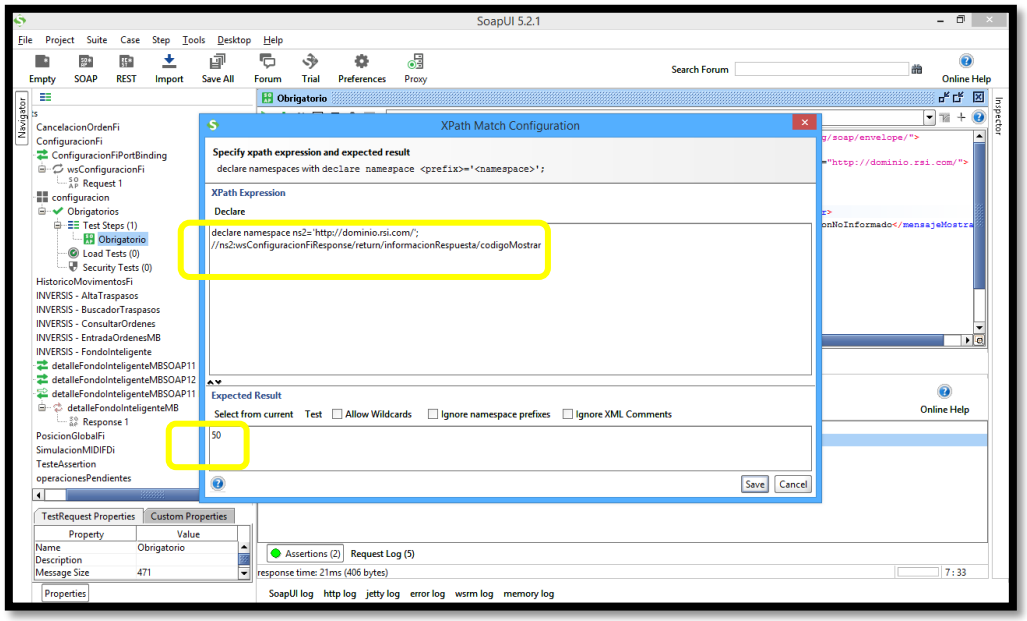

Imagem 20 – Exemplo Assertion: Xpath Expression

Com a conclusão da utilização das *assertions* passou-se a um outro nível de automação. Neste sentido, a partir dos projetos já criados, foram alteradas as propriedades dos mesmos, mais concretamente as *Custom Properties*. Assim, foram adicionadas duas propriedades: IP e *PORT*. Desta forma, atribuímos um IP e um *Port* fixo para todos os projetos. Assim, pode-se aferir que as *Custom Properties* têm a função de armazenar

valores para serem utilizados posteriormente, dentro de um determinado projeto. Após esta alteração foram realizados testes em todos os serviços, segundo os respetivos campos, os quais correram como esperado.

Com o desenrolar do trabalho surgiu a necessidade de trabalhar com o *JUnit*. Para tal, foi necessário trabalhar a partir de um computador com sistema operativo *Mac*. Isto devido à pouca qualidade do computador que me foi disponibilizado pela empresa, no sentido de disponibilizar pouca memória RAM. Primeiramente, foi necessário proceder à instalação do *SoapUI* nessa mesma máquina (imagem 21).

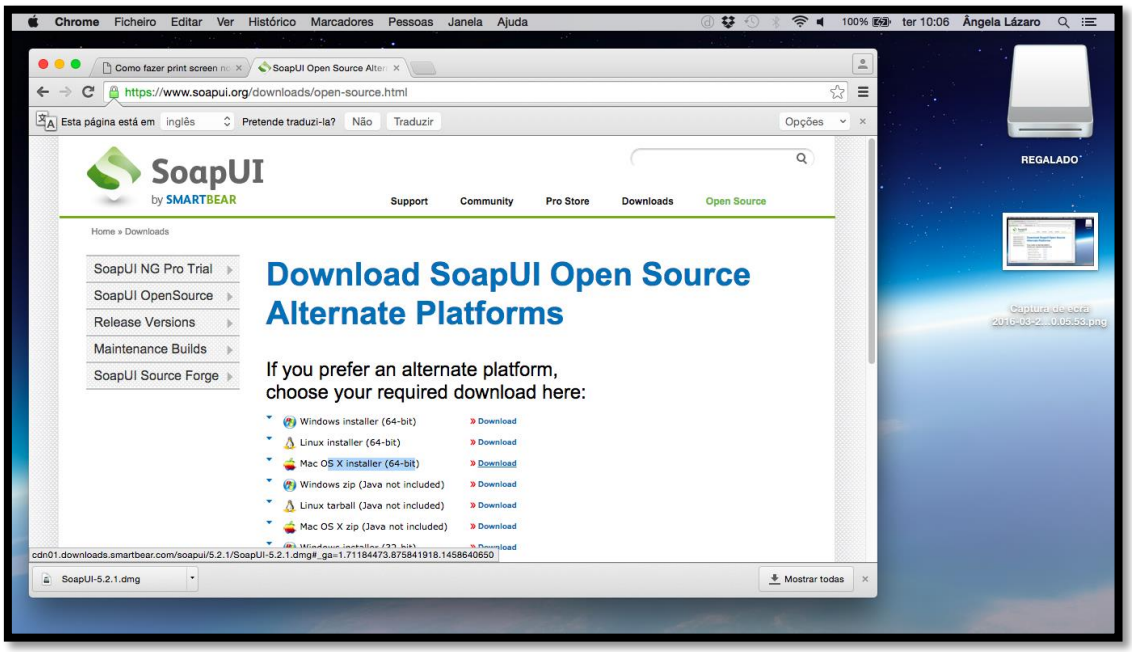

Imagem 21 – SoapUI OpenSource 5.2.1

Após a sua instalação houve a necessidade de importar os projetos já criados (imagem 22).

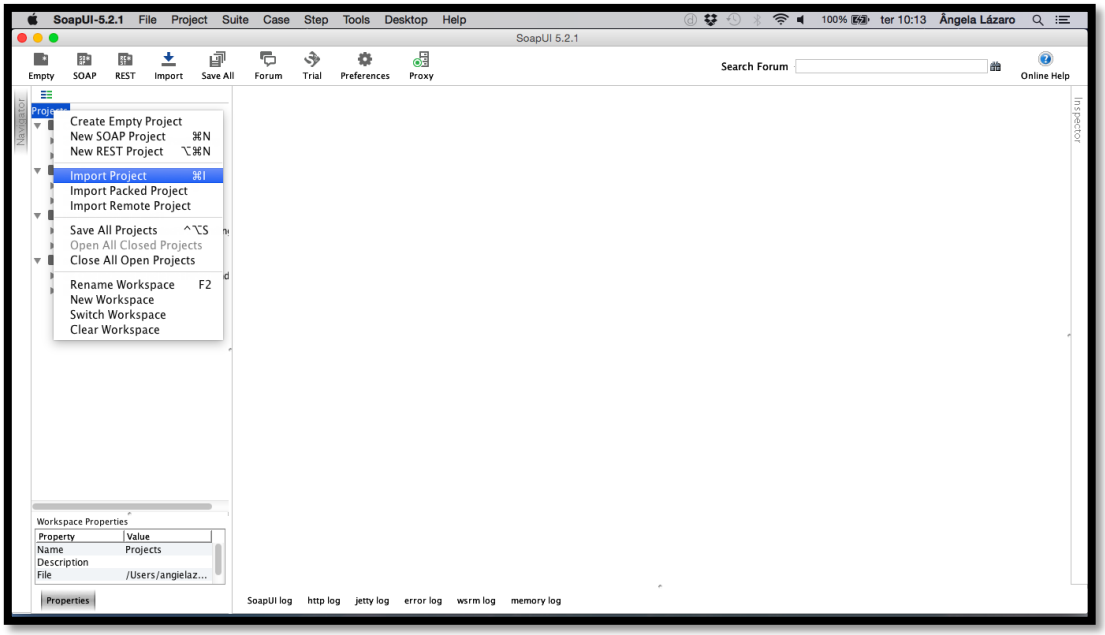

Imagem 22 – Importação de Projetos

Após este pequeno estudo foram realizados alguns testes, utilizando alguns exemplos no *JUnit*. Iniciou-se com a criação de um projeto, seguido da criação de uma classe, posteriormente um *JUnitTestSuite* e depois *JUnitTestCase* (imagem 23).

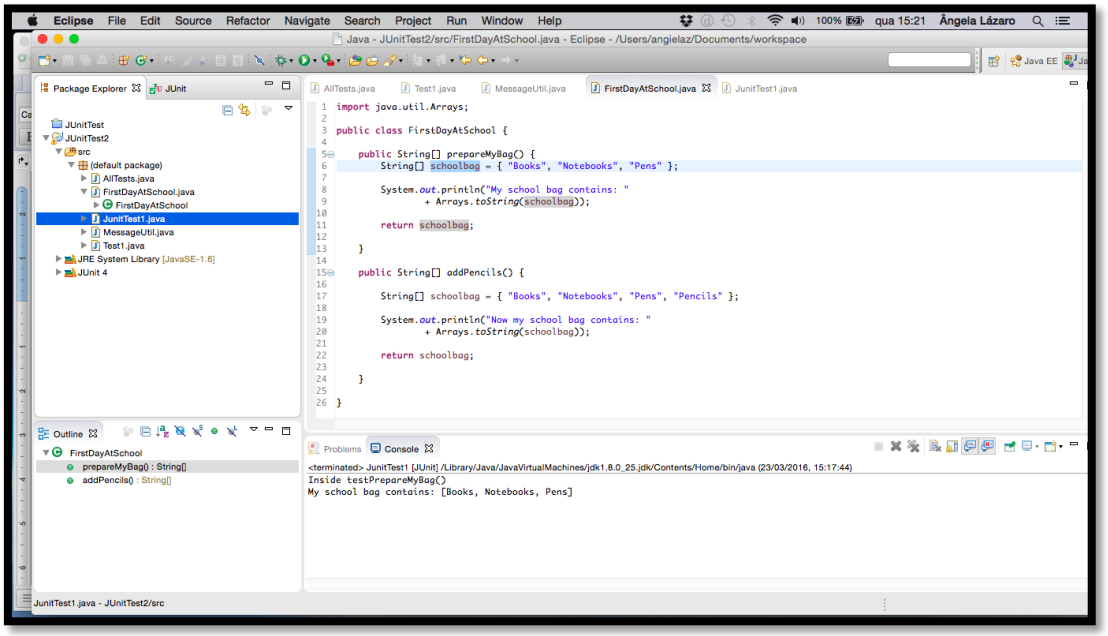

Imagem 23 – Exemplo Teste na Ferramenta Junit

No que se refere à ferramenta *JUnit* foram executados testes utilizando *asserts* (imagem 24), os quais correram como o determinado. Para tal foram criadas entradas, chamadas e validações. Com esta ferramenta pode-se verificar se os valores e formatações estão corretas e, sobretudo, os tempos que demoram a serem executados.

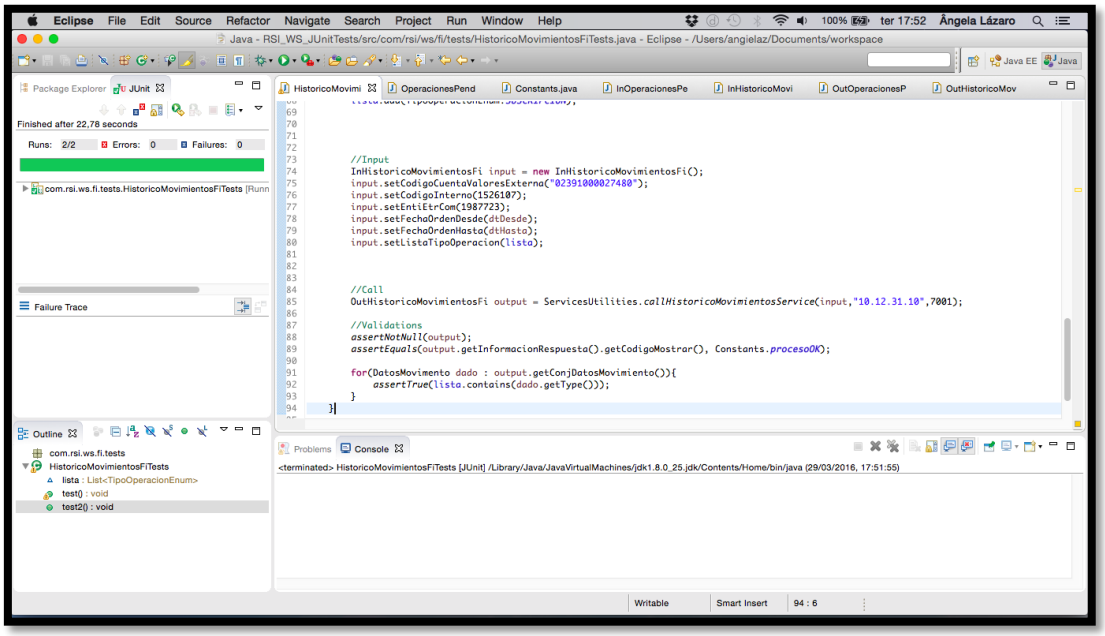

Imagem 24 – Exemplo Teste na Ferramenta JUnit

Os resultados surgiram nos diferentes tipos de visualização, de forma a informar se há erros ou não, e para possibilitar uma melhor leitura dos tempos. Os resultados corresponderam ao esperado.

No decorrer da realização dos testes, e com a utilização de diferentes ferramentas de testes, surgiu a oportunidade de utilizar uma nova ferramenta – *Selenium* – em interação com outra ferramenta já utilizada anteriormente – *Junit*.

Neste sentido, foram escolhidas diversas páginas *Web* onde se realizou uma gravação com passos simples (em cada página) e, posteriormente, se exportou como *Java* e se correu com o *JUnit*. Para tal, foi aberto o navegador Firefox e iniciado o *Selenium*. Para iniciar o teste clicou-se no botão "gravar" e inseriu-se o endereço do *site* Sapo (www.sapo.pt) e clicou-se na opção "Restaurantes", sendo eu foi aberta uma nova janela. Nessa nova janela foi escolhida um distrito e um prato de eleição. De seguida surgem as diferentes opções segundo as escolhas realizadas. Após este passo o resultado é exportado como *Java* e corrido no *JUnit*.

Após a realização destes testes foi feito, passo a passo, o contrário. Foi escrito, em *Java*, código de forma a dar ordem para se abrir uma determinada página e a partir dessa se abrisse um *link*, dentro da mesma. Neste sentido, foram criadas variáveis, foi ordenado que antes do teste iria abrir-se a página dos CTT (http://www.ctt.pt/home/index.html), que durante o teste seria aberto um novo *link* dentro da página dos CTT e após o teste seria fechada a página. O código foi executado e correu como o esperado. Ou seja, após a execução verificou-se o resultado, sendo a página dos CTT surgiu e posteriormente o link e observou-se que o teste não apresentava erros (imagem 25).

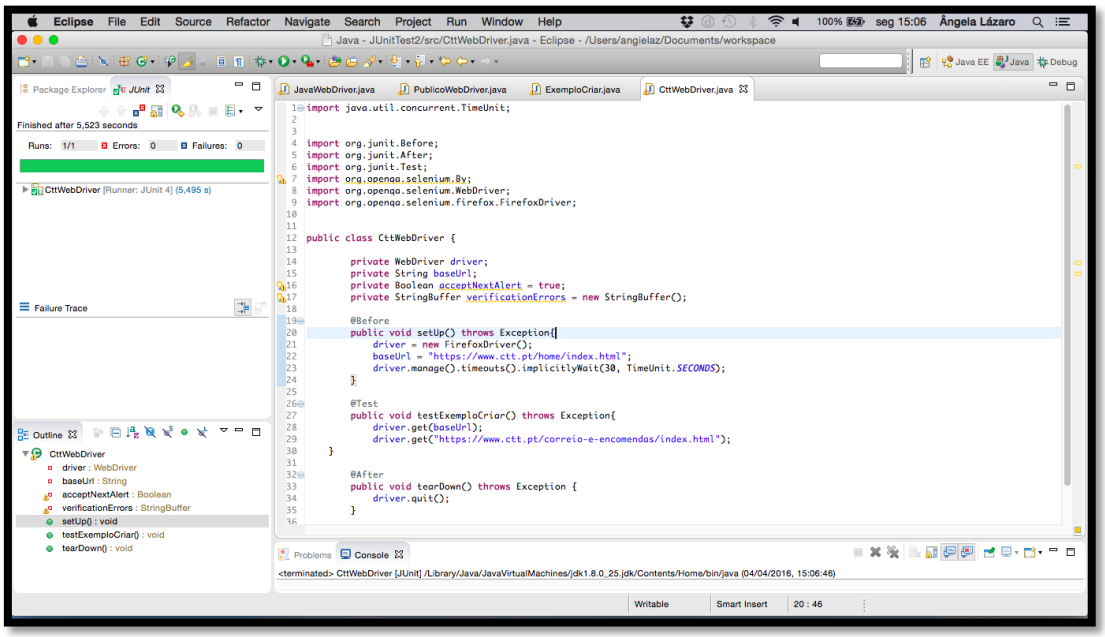

Imagem 25 – Exemplo Teste na Ferramenta JUnit

Outra atividade realizada foi a leitura da nova documentação do cliente RSI relativa a um outro projeto "*Recuperación de Formularios*". A aplicação "*Recuperación de Formularios*" é uma aplicação *Web* em que os utilizadores de Evo Banco (banco pertence à RSI) poderão registar os pedidos de contratação da "*Cuenta e Hipoteca Inteligente*" e da "*Cuenta Joven*". Nesta aplicação serão recolhidos:

- formulários iniciados na *Web* que serão terminados num registo no IRIS;
- formulários que foram abandonados nalgum momento da contratação, sem gerar a alta de um registo no IRIS;
- formulários que cuja contratação foi interrompida por algum erro no processo.

Neste sentido, esta aplicação permite a consulta/gestão destas candidaturas.

3.3. Ferramentas Utilizadas

Durante o estágio desenvolvido na Altran foram utilizadas diversas ferramentas de testes. A utilização de diversas ferramentas de testes para testar os *WebServices* da RSI tem o objetivo de permitir a exploração de diferentes ferramentas de forma a desenvolver conhecimento, capacidades e visão quer sobre os serviços que estão a ser testados, quer sobre as ferramentas passíveis de serem utilizadas, abrangendo assim um maior leque de competências técnicas. E uma vez que as ferramentas realizam diferentes tipos de testes, é de todo o interesse utilizá-las.

As ferramentas de teste de *software* são um aliado na avaliação do mesmo, de forma a precaver vulnerabilidades e garantir a sua qualidade. Servem, igualmente, para minimizar riscos e inconformidades. Ou seja, durante a realização dos testes são detetadas falhas que, ao serem corrigidas antes do lançamento do *sofware*, a sua correção vai garantir que este se conclua sem erros e que funcione como o desejável.

De seguida são apresentadas as ferramentas que foram utilizadas durante este período de estágio.

3.3.1. SoapUI

O ***SoapUI*** é uma ferramenta *OpenSource*, termo inglês que significa, na nossa língua, código aberto, escrita em *Java* cuja principal função é consumir e testar *WebServices*. É uma ferramenta de execução de testes, nomeadamente testes funcionais, unitários, carga e segurança. É também um facilitador de todo o processo de criação e depuração dos testes por meio de uma *interface* gráfica visual e intuitiva (Caetano, s.d.).

Esta ferramenta apresenta vantagens como:

- ✓ licença grátis (algumas versões);
- ✓ *interface* gráfica simples;
- ✓ permite criar e executar diferentes tipos de teste;
- ✓ favorece o suporte à maioria dos protocolos e diferentes tecnologias
- ✓ permite a maximização do trabalho (Caetano, s.d.).

De seguida pode-se observar um exemplo de teste realizado com esta ferramenta (imagem 26):

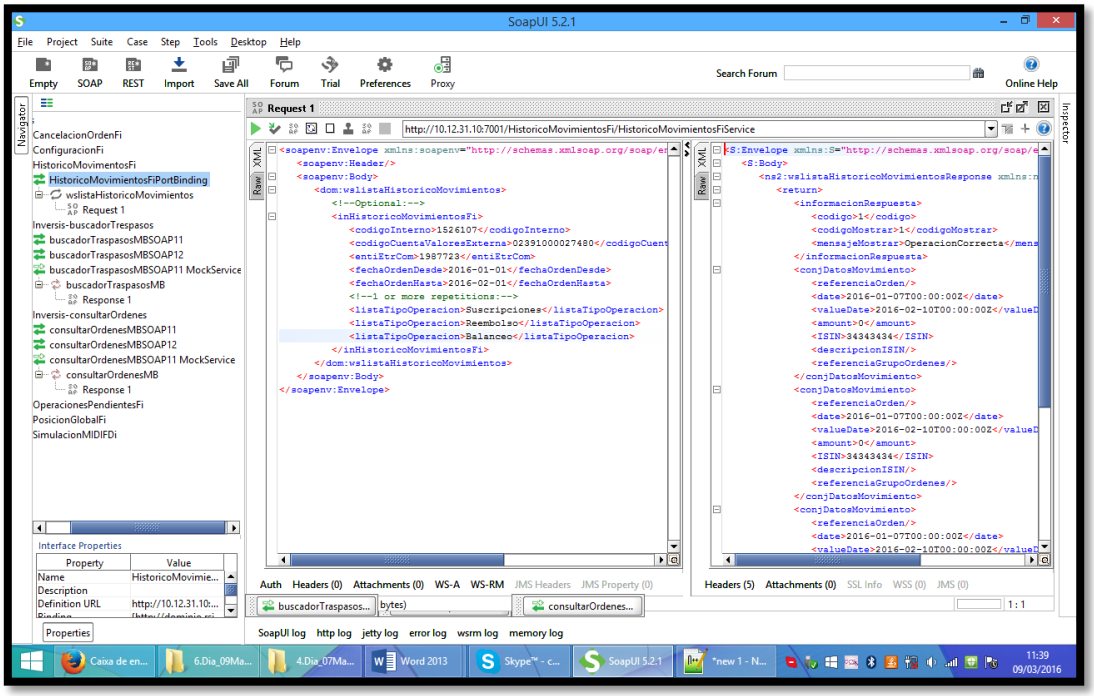

Imagem 26 – Exemplo de Teste Realizado com a Ferramenta SoapUI

Para a execução deste teste foi necessário inserir dados válidos no *request* (dados lado esquerdo), de seguida correr o teste e posteriormente verificar a resposta, respetiva, no *response* (dados lado direito). Ao inserir na request determinados dados, vai permitir receber no response os dados correspondentes ao inseridos.

3.3.2.JUnit

JUnit é uma ferramenta simples que escreve testes repetitivos. É um exemplo da arquitetura *xUnit* de ferramentas de teste à unidade. O *JUnit* é uma ferramenta *OpenSource* com suporte à criação de testes automatizados na linguagem de programação *Java*. Esta ferramenta facilita a criação de código para a automação de testes com apresentação dos resultados. A sua principal função é verificar se cada método de uma classe funciona da forma esperada, exibindo possíveis erros ou falhas.

Algumas vantagens de se utilizar a ferramenta *JUnit*:

- ✓ permite a criação rápida de código de teste enquanto possibilita um aumento na qualidade do sistema, sendo desenvolvido e testado;
- ✓ não é necessário escrever o próprio *framework*;
- ✓ amplamente utilizado pelos programadores da comunidade código-aberto;
- ✓ uma vez escritos, os testes são executados rapidamente sem que, para isso, seja interrompido o processo de desenvolvimento;
- ✓ verifica os resultados dos testes e fornece uma resposta imediata;
- ✓ pode-se criar uma hierarquia de testes que permitirá testar apenas uma parte do sistema ou todo ele;
- ✓ permite que o programador perca menos tempo a depurar seu código;
- ✓ *JUnit* é livre.

Da pesquisa realizada percebeu-se que existem vários métodos que permitem correr os testes com as suas especificidades. Alguns destes métodos (JUnit, 2015):

- ✓ *fail(message)* → permite ao método falhar;
- ✓ *assertTrue([message,] boolean condition)* → verifica se a condição *boolean* é verdade;
- ✓ *assertEquals([message,] expected, actual)* → testa que dois valores são iguais.

Na imagem seguinte pode-se verificar um teste realizado com a ferramenta *JUnit* (imagem 27).

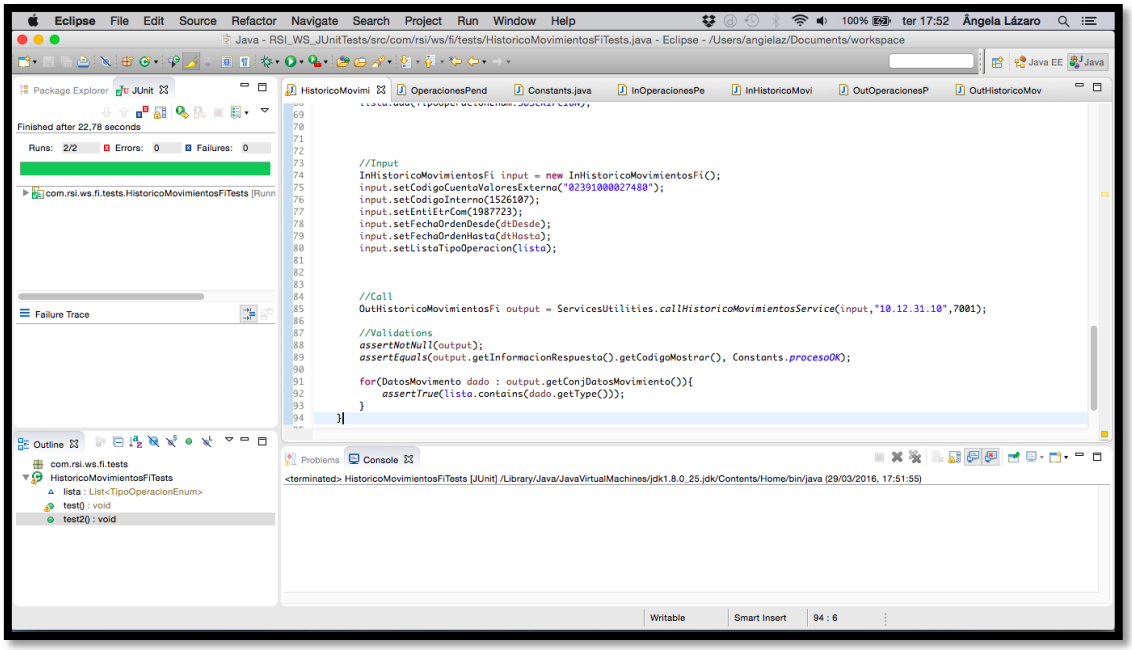

Imagem 27 – Exemplo de Teste Realizado com a Ferramenta JUnit

Para executar este teste foi preciso utilizar alguns métodos, nomeadamente o método *assertNotNull* e o *assertEquals*, surgindo assim os respetivos valores. Os resultados neste tipo de testes vão estar sempre de acordo com o método escrito no teste, possibilitando assim a escolha do tipo de dados que se quer verificar. Neste caso, o teste executado foi ao *WebService historicoMovimientos*.

3.3.3.JMeter

JMeter é uma aplicação *Java desktop*, *OpenSource*, que simula o acesso de vários utilizadores ao mesmo tempo a um determinado sistema *Web*.

Tal como as outras ferramentas o *JMeter* possui vantagens, sendo estas:

- ✓ simplifica a procura de erros;
- ✓ *interface* amigável (*userfriendly*);
- ✓ permite configurar o número de *threads*, a quantidade de vezes que cada *thread* será executada e o intervalo entre cada execução.

Na ferramenta *JMeter* foram criados *Tests Plan* para testar os *WebServices* e verificar o tempo que demora a aceder. Para a utilização desta ferramenta é necessário colocar o *url* do *WebService* a testar, o código do mesmo e criar modos de vista de forma a facilitar a leitura dos testes, nomeadamente *View Results Tree*, *View Results in Table* e *Summary Report* (JMeter, 2014).

JMeter é uma aplicação *desktop* de código aberto, 100% feita em *Java* e é utilizada para realizar testes de carga. Neste sentido, pode-se verificar se a aplicação tem capacidade de utilização, em simultâneo, por vários utilizadores. Uma daa principais vantagens prende-se com o facto de ser possível verificar o tempo em que o acesso é feito. De seguida, encontra-se um exemplo de um teste realizado com esta ferramenta (imagem 28).

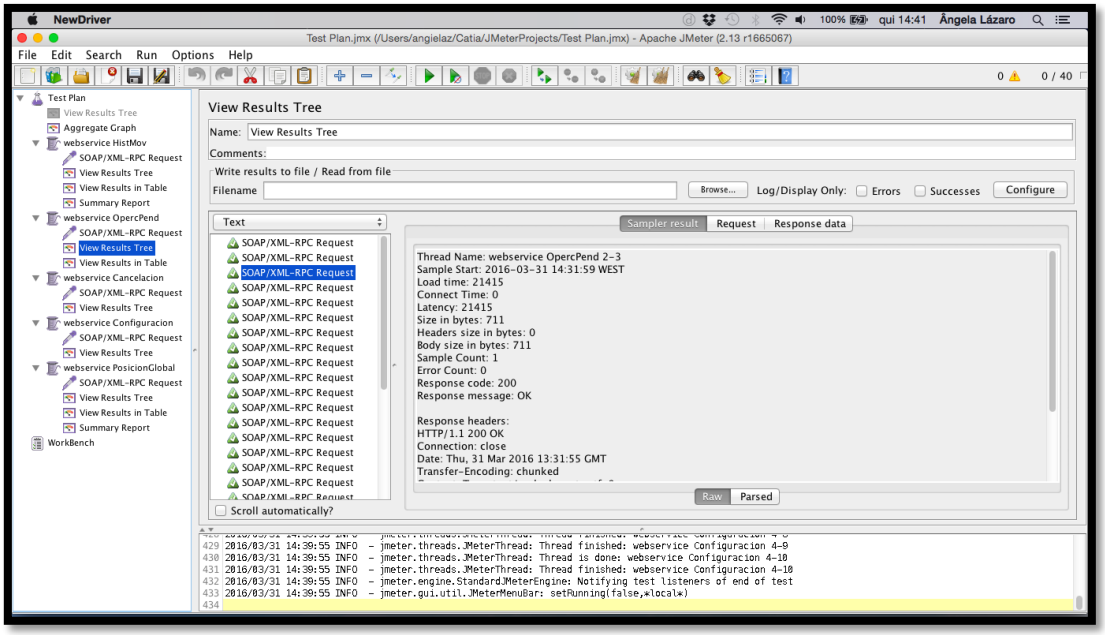

Imagem 28 – Exemplo de Teste Realizado com a Ferramenta JMeter

Com a execução deste teste verificou-se que o sistema suporta, com sucesso, a adesão de vários utilizadores em simultâneo. Neste teste pode-se ainda observar o tempo de resposta do mesmo, bem como a ausência de erros na sua execução. O teste em questão foi executado ao *WebService operacionesPendientes*.

3.3.4. Selenium IDE

Selenium é uma ferramenta de testes de *software record & play*, ou seja, é uma ferramenta que grava e reproduz/corre o teste. É dirigida a aplicações *Web* e apresenta-se como um *software OpenSource*. O *Selenium* corre no *Firefox*.

O *Selenium* em questão, *Selenium IDE* (*Integrated Development Environment*), permite gravar, editar e depurar testes. Neste sentido, o *tester* executa uma ação a qual é gravada através desta ferramenta, ficando todos os passos dados gravados num script. Esta ferramenta simplifica os testes de regressão, já que a qualquer momento pode-se realizar um mesmo teste nas novas versões do sistema. Por outro lado, permite igualmente, com a ajuda do Eclipse, executar o teste no sentido contrário, ou seja escrever o código de forma a executar sozinho as ações pretendidas.

O Selenium IDE permite:

- ✓ gravar as ações a serem feitas para se construir um caso de teste;
- ✓ controlar a velocidade de gravação;
- ✓ executar a(s) gravação(ões) feita(s);
- ✓ executar o caso de teste passo a passo
- ✓ exportar as sessões de teste como arquivos fonte *Java*.

As grandes vantagens que esta ferramenta apresenta prendem-se sobretudo com a facilidade na instalação, sendo local e simples e com o facto de ser muito fácil de usar (Selenium, 2016). Desta forma, pode-se observar na imagem seguinte um exemplo de um teste executado, utilizando o Selenium (imagem 29).

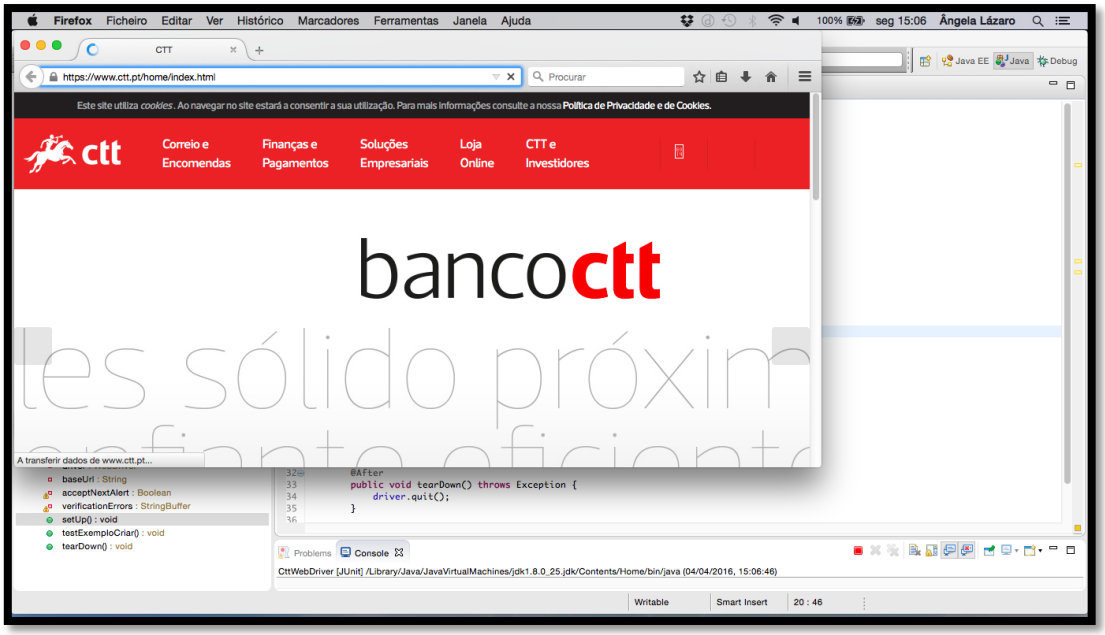

Imagem 29 – Exemplo de Teste Realizado com o Selenium

Neste teste em questão, foi escrito código no Eclipse para a execução das ações pretendidas. Neste caso, foi pedido que fosse aberto o site dos CTT e de seguida fosse aberta uma nova página incluída nesse mesmo site.

Capítulo IV – Trabalho Desenvolvido no Projeto *AstraZeneca*

4. Trabalho Desenvolvido no Projeto *AstraZeneca*

Neste capítulo faz-se alusão à integração no projeto, bem como à descrição das atividades desenvolvidas no mesmo.

4.1. Contexto de Integração no Projeto RSI

Inserção num projeto na área Farmacêutica, em *Nearshore*, irá ter à sua responsabilidade as seguintes atividades:

- ✓ Planeamento de testes.
- ✓ Criação e análise de casos de teste.
- ✓ Testes unitários, testes de integração e automação de testes funcionais na área de *Mobile* (*Android & iOS*).
- ✓ Documentação e *reporting*.
- ✓ Utilização de ferramentas como:
  - *Android Studio* (testes unitários)
  - *RobotFramework* para automação de casos de testes.

Tal como já referido neste documento, o cliente *AstraZeneca* é da área farmacêutica. O projeto *myADAURAmeds* a desenvolver tem como objetivo a criação de uma aplicação *mobile*, para *Android* e *iOS*, com a função de monitorizar a toma de um medicamento específico por parte de doentes com cancro do pulmão. Neste sentido, as

atividades a desenvolver remetem para o desenvolvimento do *software*, bem como testar o mesmo ao longo de várias etapas.

A aplicação, com uma *interface clear* e simples, dispõe de um *layout* inicial que inclui o *login* (*username* e *password*), o botão *forgot*, a *checkbox remember me* e o botão *login* (imagem 30). Ao efetuar o login com credenciais corretas a aplicação inicia, caso contrário surge uma mensagem em que indica que as credenciais estão erradas (imagem 31).

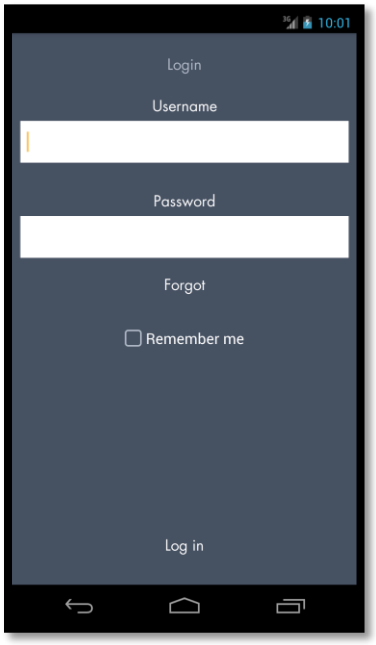

Imagem 30 – Login myADAURAmeds

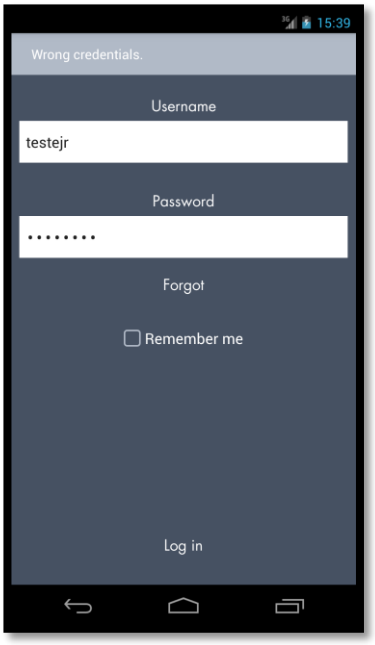

Imagem 31 – Mensagem Credenciais Erradas

O *forgot* reporta para uma página que nos solicita o *username*, que, ao ser escrito, o servidor enviará uma mensagem para o *e-mail* com um *link* que remete para criar uma nova *password*. Depois de criada a nova palavra passe o utilizador pode voltar à página inicial do login e efetuar o mesmo com valores corretos.

A *checkbox remember me*, ao ser ativada, envia uma mensagem *warning* para o ecrã, indicando que ao relembrar as credenciais qualquer pessoa que tenha acesso ao telemóvel do utilizador pode ter, igualmente, acesso à aplicação (imagem 32).

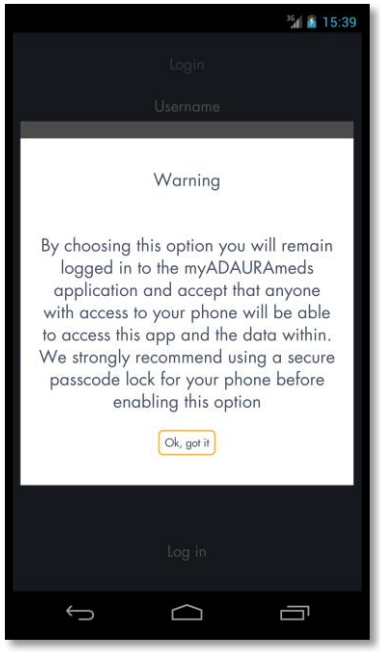

Imagem 32 – Mensagem Warning

Após o *login* efetuado corretamente, surge um *layout* com o menu de navegação, o menu *progress* e o menu *record medicine* (imagem 33).

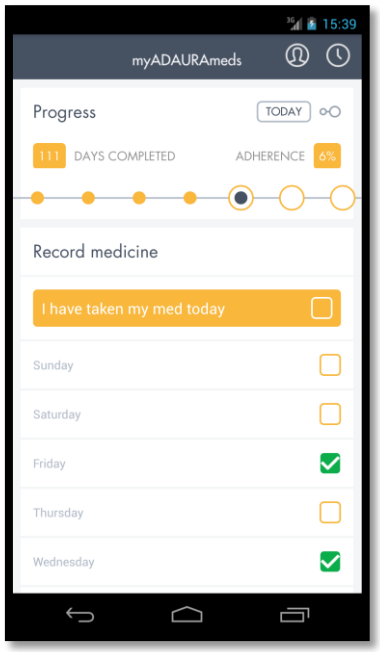

Imagem 33 – Layout com Menu Navegação, Menu Progress e Menu Record Medicine

O primeiro, menu de navegação, contém o nome da aplicação, o botão *logout* e o botão *set reminder*. O *logout* efetua a saída da aplicação de forma a voltar à página inicial

do *login* (imagem 34). Enquanto o botão *set reminder* permite definir a hora em que a aplicação alertará o utilizador para a toma do medicamento (imagem 35).

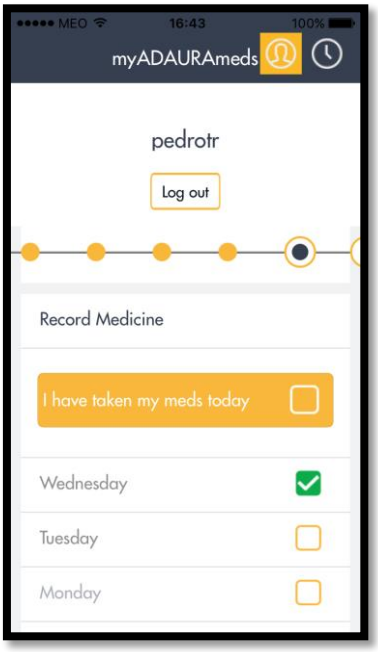

Imagem 34 – Menu Logout

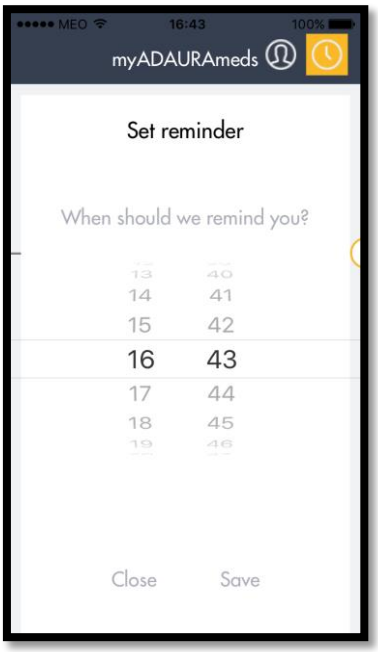

Imagem 35 – Menu Set Reminder

O menu *progress* (imagem 36) dispõe do botão *today*, uma imagem, a indicação dos *days completed* assim como da *adherence* e a *timeline*. O botão *today* remete o utilizador para a semana atual. A imagem ao lado do *today* dá acesso à informação da *timeline*. A indicação *days completed* informa os dias completos em que se efetuou o tratamento, assim como *adherence* indica a aderência da toma do mesmo. A *timeline* indica a semana em que estamos e mostra, ainda, as semanas transatas e as posteriores. A *timeline* tem outra função, sendo esta de extrema importância: informa o doente da data da próxima consulta (*Study Visit*).

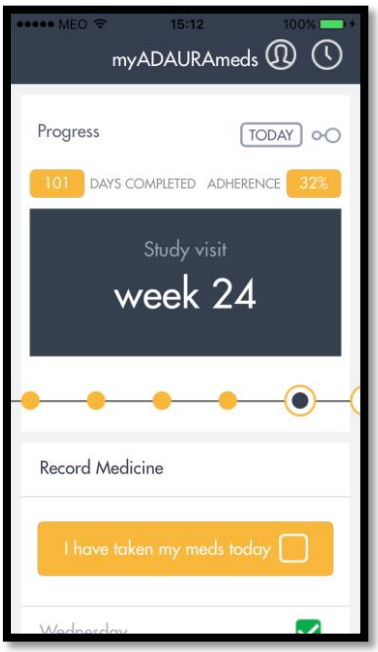

Imagem 36 – Menu Progress

O menu *record medicine* (imagem 37) mostra os dias completos, até ao momento, da toma do medicamento, apontando se este foi tomado, não tomado ou se simplesmente o utilizador não registou nenhuma das opções (*meds taken*, *meds not taken*, *empty*). Este menu também possibilita alterar a toma do medicamento, mas apenas, dos últimos sete dias. É importante referir que ao iniciar a aplicação e ao clicar pela primeira vez na checkbox, indicando a toma do medicamento no dia de hoje, surge uma mensagem de felicitações pelo ato. O aparecimento destas mensagens tem o objetivo de encorajar os doentes a continuar o tratamento.

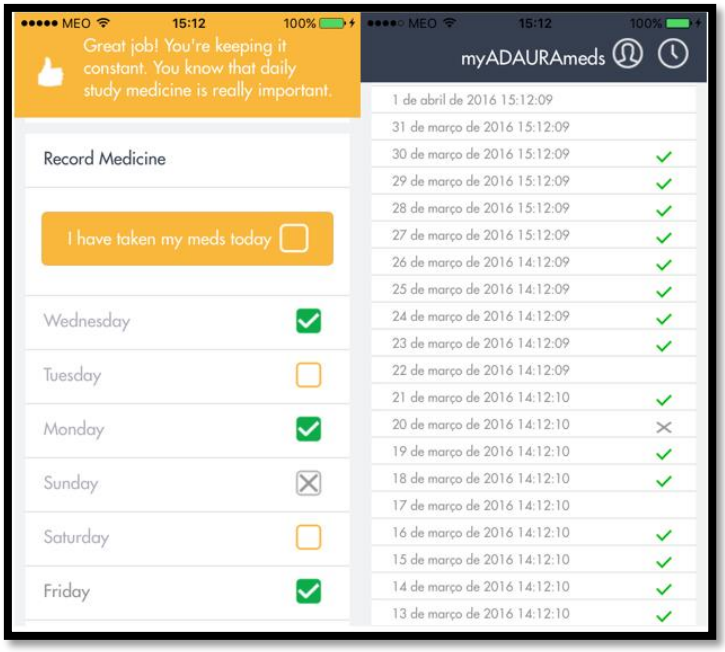

Imagem 37 – Menu Record Medicine

Como já foi referido, em certos momentos surgem mensagens para o utilizador. Estas dividem-se em dois tipos: Reminders e Rewards. As mensagens Reminders servem para relembrar o paciente para a toma do medicamento. Enquanto as Rewards são mensagens de encorajamento e de incentivo à toma e à continuação do tratamento. Estas são diversas e funcionam da seguinte maneira: a primeira vez que a app é corrida as mensagens surgirão por ordem, posteriormente, vão aparecer aleatoriamente.

A equipa afeta a este projeto é composta por múltiplos colaboradores, especificamente elementos da Altran Suécia, França, Inglaterra e Portugal, a empresa InVision (que desenvolveu os *Layouts*) e o cliente. Na Altran Portugal, atualmente, são 8 os colaboradores, nomeadamente: 1 gestor de projeto, 4 *developers* e 3 *testers*.

4.2. Atividades Desenvolvidas no Projeto *AstraZeneca*

A integração neste projeto foi feita a 09 de maio do ano corrente e teve términus a 30 de junho. Ao longo da permanência no projeto da *AstraZeneca* foram realizados vários testes, tais como: testes funcionais, testes unitários, testes de usabilidade, testes visuais e automação de testes. Alguns destes já tinham sido experienciados e realizados no projeto anterior e outros apenas foram desenvolvidos neste. Assim, referem-se os

testes de usabilidade, sendo que estes têm o objetivo de verificar a facilidade com que o utilizador interage com a aplicação. Ou seja, indica-nos se o *software* é intuitivo e de fácil compreensão por parte do *user*. Os testes visuais permitem detetar falhas ao nível visual de todos os elementos que compõem o *layout*.

No decorrer destas atividades foram encontradas algumas falhas e reportadas aos *developers*. Para tal trabalho foi construído um documento em Excel de forma a registar e reportar os *bugs* (apêndice 2). Um exemplo retirado do mesmo apêndice é o que se pode verificar na imagem seguinte (imagem 38).

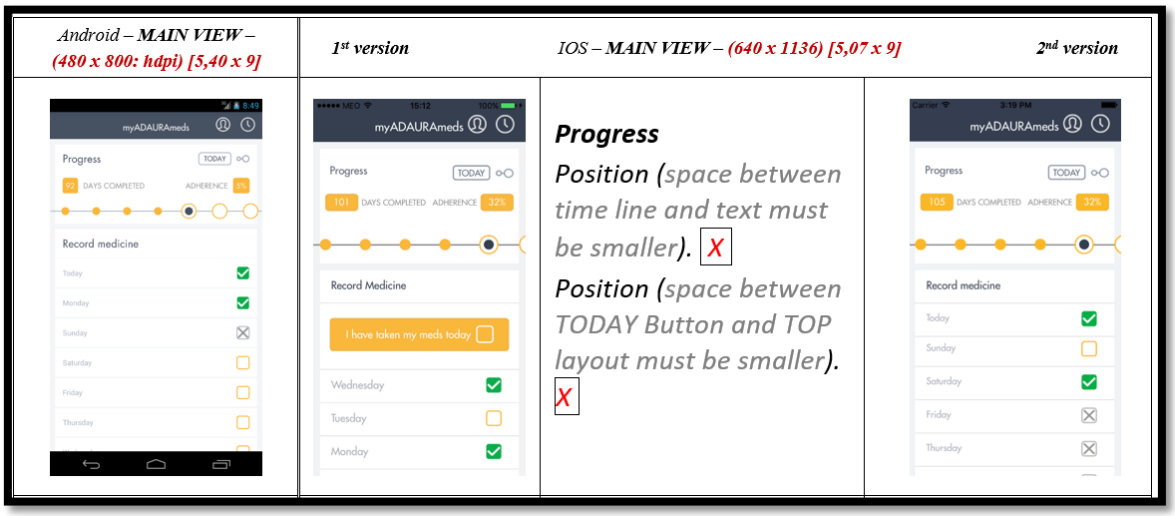

Imagem 38 – Exemplo de Falhas Detetadas

No projeto *AstraZeneca* eram feitas *stand-up meetings* com o intuito de, cada colaborador, referir as atividades realizadas no dia anterior, assim como as atividades a realizar no próprio dia. Com estas reuniões pretendia-se mostrar o trabalho de cada um, bem como rever em que ponto do processo se encontra o projeto, indicando os pontos já executados e os a executar com maior urgência.

Outro tipo de reuniões realizadas eram as reuniões dos *sprints reviews* em que participavam todos os colaboradores das diferentes empresas sucursais da Altran (a nível internacional) e o cliente *AstraZeneca*. Estas reuniões eram feitas com o objetivo de dar por terminado o sprint, apresentando o trabalho realizado em cada sprint, assim como prever e indicar o trabalho a realizar no *sprint* seguinte, até à conclusão do projeto.

4.3. Ferramentas Utilizadas

As ferramentas utilizadas neste projeto são: *Adroid Sutdio, Ride, Appium, Notepad++, UIAutomatorViewer*.

4.3.1. Android Studio

Android Studio é um ambiente de desenvolvimento integrado (IDE) para desenvolver para a plataforma *Android*. As funções deste *software* incluem a edição inteligente de códigos, recursos para design de interface de utilizador e análise de performance, entre outras coisas. Algumas das características mais atraentes apresentadas pelo *Google* são:

- visualização dos recursos como *strings*, ícones e cores. Por exemplo, na IDE quando utilizar uma *string* o editor irá apresentar o conteúdo do valor da *string*;
- análise de código baseado nas anotações da API do *Android*;
- pré-visualização do *layout* – o *layout* da aplicação pode ser visualizado simultaneamente para todas as resoluções de ecrã e idiomas suportados, aplicando as mudanças simultaneamente;
- construtor de *layout* – um editor gráfico com arrastar e soltar (*Drag and Drop*) (Avram, 2013).

No entanto, na área de testes, o *Android Studio* disponibiliza *devices* de forma a realizarmos testes em sistema *Android* . Assim, apresenta-se de seguida um exemplo de um teste realizada à aplicação através de um *device* (imagem 39).

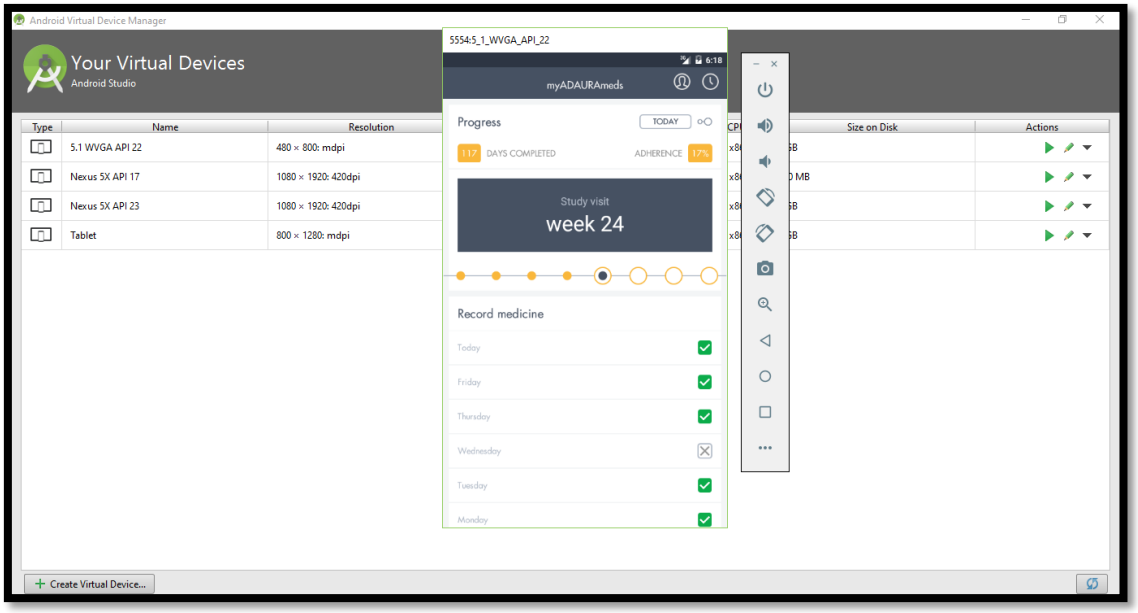

Imagem 39 – Exemplo Utilização da Ferramenta Android Studio

4.3.2. Ride

Ride é implementado com *Python* e suporta também *Jython* (JVM) e *IronPython* (NET). É um *framework* de automação de teste genérico para testes de aceitação. Os testes são criados através da utilização de palavras-chave. Para tal é necessário utilizar *libraries*. O *Ride* já inclui uma biblioteca API simples para criar bibliotecas de teste personalizados que podem ser implementadas de forma nativa com *Python* ou *Java*. Disponibiliza, igualmente, capacidade de criar palavras-chave de nível superior reutilizáveis a partir das palavras-chave existentes. Apresenta relatórios de resultados de fácil leitura e *logs* em formato HTML, com o intuito de verificar logo as falhas (caso existam), localizando-as e dando indicações. Uma outra vantagem do *Ride* prende-se com o facto de fornecer a marcação para categorizar e selecionar casos de teste a ser executado (Robot Framework, 2015).

De seguida verifica-se um exemplo de um teste corrido no *Ride* (imagem 40).

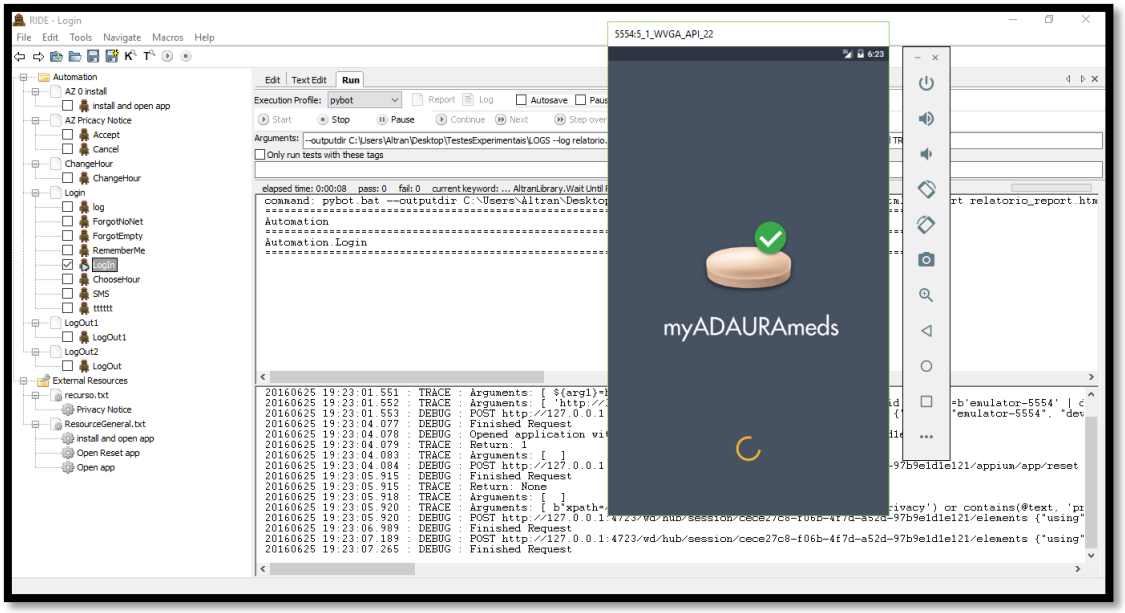

Imagem 40 – Exemplo de Teste Realizado com a Ferramenta Ride

4.3.3. Appium

Appium é uma ferramenta *OpenSource* e multiplataforma para automação de aplicações nativas, híbridas e sites *mobile* para os principais sistemas operacionais: *Android*, *iOS* (e agora para *FirefoxOS*). Este *software* utiliza os próprios *frameworks* de teste de cada plataforma. As principais funcionalidades do *Appium* são:

- os desenvolvedores podem testar aplicações nativas, híbridas e *web* mobile;
- pode ser simulado em dispositivos reais, emuladores ou simuladores;
- aplicações *iOS* e *Android* podem ser testadas com um script simples;
- as aplicações *web* mobile normalmente precisam de um script separado de teste, diferente do utilizado para aplicações nativas, pois há grandes diferenças entre os elementos gráficos contidos em uma página *web* e na estrutura de tais páginas;
- para executar os testes, o *Appium* interage com a biblioteca *UIAutomation* da *Apple* e o *framework UIAutomator* nos *Android* com versão superior a 16. Para as versões anteriores do *Android*, o *Appium* usa o *Selendroid*;
- o *Appium* usa a API *Web*Driver do *Selenium* para enviar os comandos de teste.

A utilização do *Appium* neste projeto cinge-se à questão de o colocar a correr sempre que se correm/executem testes no *Ride* (Costa M. , 2015). Neste sentido apresenta-se uma imagem desse exemplo (imagem 41).

Imagem 41 – Exemplo de Utilização da Ferramenta Appium

4.3.4. Notepad++

Notepad++ é um pequeno e rápido editor de texto de código aberto, para *Windows*, que permite trabalhar com arquivos de textos simples e código-fonte de diversas linguagens de programação. A sua utilização é uma mais-valia, uma que ele tem suporte à diferenciação de comandos através de cores, facilitando a leitura e correção de possíveis erros. Este programa já tem embutido o reconhecimento para linguagens com C, C++, *Java*, *HTML*, *XML*, *PHP*, *JavaScript* e várias outras. Neste sentido, a utilização do *Notepad++* no projeto é de todo importante, constituindo este um auxiliar à construção das variáveis a serem utilizadas na criação dos *test cases* no *Ride* (TechTudo, 2014). Asim, pode-se verificar na imagem apresentada a seguir como esta ferramenta ajuda em muito no trabalho executado (imagem 42).

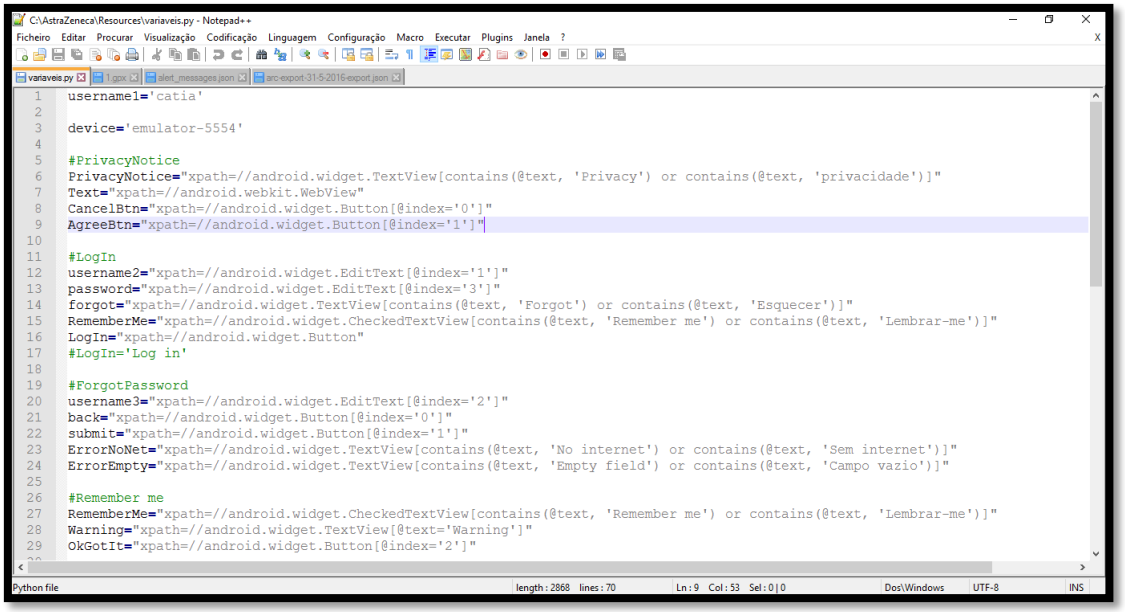

Imagem 42 – Exemplo de Utilização do Notepad++ na Criação de Variáveis

4.3.5. UIAutomatorViewer

O ***UIAutomatorViewer*** faz parte das ferramentas do *Android* SDK para a criação de testes em *interface* gráfica, pertencendo a uma sessão específica dentro do universo *Android*. Neste sentido, é uma *interface* visual para inspecionar componentes *Android* e visualizar as propriedades destes componentes em emuladores ou dispositivos. Uma vez que fornece as propriedades, nomeadamente o *path*, pode-se construir o xPath de qualquer elemento com o objetivo de criar testes mais robustos e menos suscetíveis a erros por conta da localização destes. Assim, o *UIAutomatorViewer* torna-se numa ferramenta essencial já que para automatizar é necessário saber como localizar os elementos/componentes no ecrã para escrever o código interagindo com o mesmo. A função do *UIAutomatorViewer* neste projeto é mostrar os componentes da app *Android* que está a ser executada no emulador, de forma a inspecionar as suas propriedades e determinar a forma de localização (Nogueira, 2016). Como tal, temos a seguir uma imagem que reflete esse ponto (imagem 43).

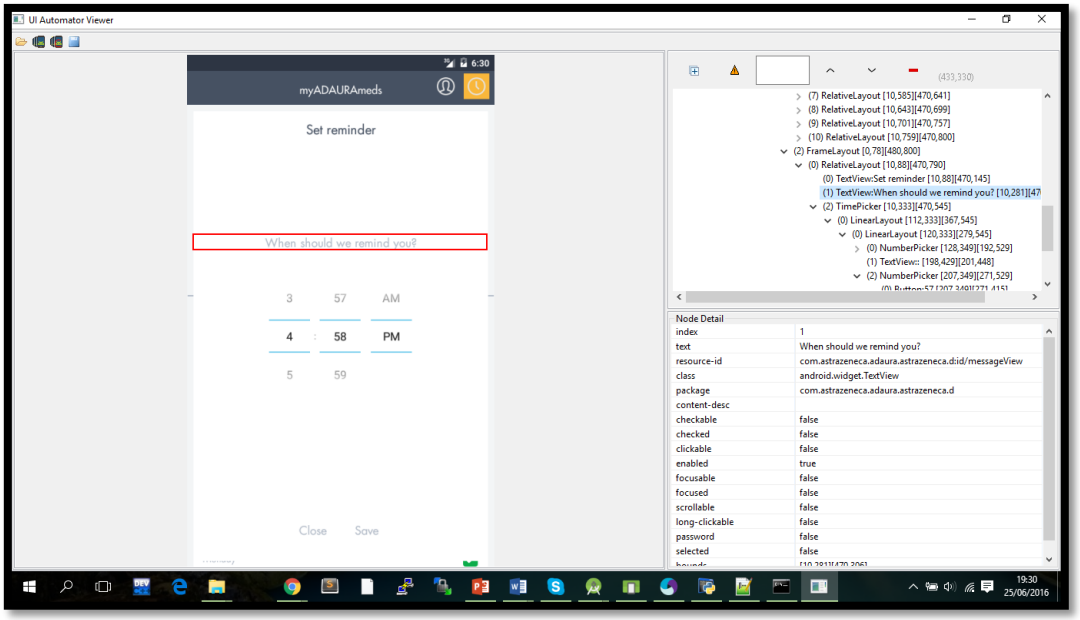

Imagem 43 – Exemplo de Captura com a Ferramenta UIAutomator e Path de Elemento

Capítulo IV – Conclusão

Conclusão

Com o colmatar do estágio são algumas as ilações a tirar. A mensagem que ressalta é a importância de executar testes de *software*. Esta importância é fundamental, repercutindo-se no lançamento de *software* com qualidade. E uma vez que, cada vez mais a exigência dos clientes é maior, é de todo necessário apostar nos testes de *software*. Apenas assim as empresas podem dar respostas eficazes e satisfatórias.

Ao se executarem testes desde o início do ciclo de vida do *software*, torna este processo mais barato e eficaz, ou seja, os erros encontrados logo no início da produção do *software* mais cedo serão corrigidos, garantindo uma maior qualidade do mesmo. Contudo, nem todas as empresas dão valor aos testes de *software*, apresentando alguma resistência.

No decorrer do estágio proporcionado pela Altran, foi possível colocar em prática as aprendizagens adquiridas em unidades curriculares do plano de estudos do Curso Testes de *Software*. Esta fase de desenvolvimento de competências é fulcral, tendo sido importante realizar testes num ambiente profissional e concreto. Com esta experiência pode-se concluir que efetivamente é de extrema importância realizar testes de *software*, verificando-se que cada vez mais se aposta na automatização. Este novo conceito que surge lentamente, reflete-se num trabalho feito cada vez mais por máquinas, deixando a mão humana de lado. Desta forma, torna os testes mais rápidos e com custos reduzidos, tal como refere (Fernandes, 2014) "Sendo a etapa de testes uma das mais "caras" em termos de recursos e tempo no processo de desenvolvimento de *software*, a automatização de processos que compõem esta área tornou-se um dos principais desafios e interesses para as organizações." Contudo, importa ressalvar que nem todos os testes são passíveis de automatizar.

Assim, conclui-se que a aposta futura passará cada vez mais por esta opção, gerando assim *software* com qualidade em menor tempo, respondendo às espectativas do cliente.

Bibliografia

(s.d.). Obtido de JUnit: http://junit.org/junit4/

(s.d.). Obtido de Altran: www.altran.com

(s.d.). Obtido de Altran: www.altran.pt

Alves, P. (20 de 02 de 2014). *O que é HTTPS e como ele pode proteger a sua navegação na Internet*. Obtido de Techtudo: http://www.techtudo.com.br/noticias/noticia/2014/02/o-que-e-https-e-como-ele-pode-proteger-sua-navegacao-na-internet.html

Apache JMeter: Testes de software com SoapUI. (s.d.). Obtido de DevMedia: http://www.devmedia.com.br/apache-jmeter-testes-de-software-com-soapui/32087

Assertion (software development). (04 de 02 de 2016). Obtido de Wikipedia: https://en.wikipedia.org/wiki/Assertion_%28software_development%29

Automação de teste. (25 de 11 de 2014). Obtido de Wikipedia: https://pt.wikipedia.org/wiki/Automa%C3%A7%C3%A3o_de_teste

Brito, E. (18 de 09 de 2013). *O que é o IP? Descubra para que serve e qual é seu número*. Obtido de Techtudo: http://www.techtudo.com.br/artigos/noticia/2013/05/o-que-e-o-ip-descubra-para-o-que-serve-e-qual-e-seu-numero.html

Caetano, C. (s.d.). *SoapUI: Testes de Web Services rápido e descomplicado*. Obtido de Linha de Código: http://www.linhadecodigo.com.br/artigo/1286/soapui-testes-de-web-services-rapido-e-descomplicado.aspx

Caracterização do Fundão. (s.d.). Obtido de Câmara Municipal do Fundão: https://www.cm-fundao.pt/movetofundao/caracterizacao_fundao

Caroline, A., & Pessotti, M., & Ventura, M., & & Rodrigues, R. (24 de 10 de 2012). *JMeter*. Obtido de Slideshare: http://pt.slideshare.net/rodcoura/jmeter-14864546

Cipriano, P. A. (2007). *Plataforma de testes de sistemas de software.* Porto.

Escrevendo Testes JUnit no NetBeans IDE. (s.d.). Obtido de NetBeans: https://netbeans.org/kb/docs/java/junit-intro_pt_BR.html

Galves, M., & & Junior, O. A. (07 de 02 de 2008). *Testes automatizados de aplicações web com Selenium.* Obtido de Dicas-L: http://www.dicas-l.com.br/arquivo/testes_automatizados_de_aplicacoes_web_com_selenium.php

Goncalmar, C. (04 de 06 de 2013). *O que significa http, e para que serve?* Obtido de Cahmar123: http://cahmar123.blogspot.pt/2013/06/o-que-significa-http-e-para-que-serve.html

JMeter. (16 de 10 de 2014). Obtido de Wikipedia: https://pt.wikipedia.org/wiki/JMeter

JUnit. (09 de 03 de 2015). Obtido de Wikipedia: https://pt.wikipedia.org/wiki/JUnit

Lima, D. (s.d.). *Qualidade de software e Selenium IDE em testes funcionais.* Obtido de DevMedia: http://www.devmedia.com.br/qualidade-de-software-e-selenium-ide-em-testes-funcionais/33225

Lopes, M. B., & & Carneiro, A. G. (s.d.). *A Importância do Processo de Teste de Software em TI.* Obtido de Univiçosa: http://www.univicosa.com.br/arquivos_internos/artigos/ImportanciadoProcessodeTestedeSoftwareemTI.pdf

Marins, P. (s.d.). *O que é front-end?* Obtido de Mobgeek: http://mobgeek.com.br/blog/o-que-e-front-end

O que é open source? (s.d.). Obtido de Canaltech: http://canaltech.com.br/o-que-e/o-que-e/O-que-e-open-source/

O que são os arquivos XSD? (28 de 12 de 2011). Obtido de Virtualgroup: http://www.virtualgroup.com.br/o-que-sao-os-arquivos-xsd/

Objeto Mock. (18 de 09 de 2015). Obtido de Wikipedia: https://pt.wikipedia.org/wiki/Objeto_Mock

Pereira, A. P. (18 de 03 de 2009). *O que é XML?* Obtido de Techmundo: http://www.tecmundo.com.br/programacao/1762-o-que-e-xml-.htm

Porta (redes de computadores). (19 de 10 de 2015). Obtido de Wikipedia: https://pt.wikipedia.org/wiki/Porta_%28redes_de_computadores%29

Selenium (software). (05 de 04 de 2016). Obtido de Wikipedia: https://en.wikipedia.org/wiki/Selenium_%28software%29

Selenium IDE. (s.d.). Obtido de SeleniumHQ: http://www.seleniumhq.org/projects/ide/

Servidor. (11 de 04 de 2016). Obtido de Wikipedia: https://pt.wikipedia.org/wiki/Servidor

SoapUI: Testes de Web Services rápido e descomplicado. (s.d.). Obtido de LINHA DE CODIGO: http://www.linhadecodigo.com.br/artigo/1286/soapui-testes-de-web-services-rapido-e-descomplicado.aspx

Souza, E. (07 de 03 de 2013). *Automação de testes de software*. Obtido de Slideshare: http://pt.slideshare.net/eduardofsouza9/automacao-de-testes-de-softwares

Steppat, N. (28 de 02 de 2007). *Testes de aceitação com o Selenium*. Obtido de Caelum: http://blog.caelum.com.br/testes-de-aceitacao-com-o-selenium/

Teste de carga. (25 de 04 de 2015). Obtido de Wikipedia: https://pt.wikipedia.org/wiki/Teste_de_carga

Teste de software. (08 de 03 de 2016). Obtido de Wikipedia: https://pt.wikipedia.org/wiki/Teste_de_software

Understanding Assertions in SoapUI – SoapUI Tutorial #5. (17 de 03 de 2016). Obtido de Software Testing Help: http://www.softwaretestinghelp.com/soapui-tutorial-5-soapui-assertions/

Urbano. (27 de 08 de 2012). *Tutorial JMeter: Uso do JMeter para Testes de Desempenho na Web*. Obtido de iMobilis: http://www.decom.ufop.br/imobilis/metodologia-de-testes-tutorial-jmeter-para-testes-de-performance-em-plataforma-web/

Using XPath Assertion in SOAP UI. (s.d.). Obtido de Groovy in SOAP UI: https://groovyinsoapui.wordpress.com/tag/xpath-assertion-in-soap-ui/

Web Services Description Language. (13 de 11 de 2015). Obtido de Wikipedia: https://pt.wikipedia.org/wiki/Web_Services_Description_Language

Working with Properties. (s.d.). Obtido de SoapUI: https://www.soapui.org/functional-testing/properties/working-with-properties.html

Working with Properties. (s.d.). Obtido de SoapUI: https://www.soapui.org/scripting---properties/working-with-properties.html

XML. (08 de 03 de 2016). Obtido de Wikipedia: https://pt.wikipedia.org/wiki/XML

Glossário

Análise Funcional

é um documento onde constam as principais funções de um sistema de modo a que a arquitetura funcional seja aprofundada com uma descrição detalhada de cada uma delas.

Android

é um sistema operativo, da Google, para dispositivos móveis, nomeadamente *SmartPhones*.

Android Studio

é um ambiente de desenvolvimento integrado (IDE) para desenvolver para a plataforma *Android.*

Appium

é uma ferramenta *OpenSource* e multiplataforma para automação de aplicações nativas, híbridas e sites *mobile* para os principais sistemas operacionais: *Android, iOS* (e agora para *FirefoxOS*).

Aplicação *Mobile*

normalmente designado como app, é um *software* desenvolvido para ser instalado em dispositivos móveis.

Automatização (automação)

é um processo que consiste na utilização de *software* para controlar a execução do teste de *software*, a comparação dos resultados esperados com os resultados reais, a configuração das pré-condições, relatório de teste. A grande vantagem da automatização é a minimização de tarefas repetitivas, permitindo que os *testers* se foquem na construção de casos de teste mais pormenorizados e abrangentes.

Assertions

é uma declaração que indica que um determinado predicado deve ser verdadeiro num dado ponto do código.

Front-end

é toda a parte da apresentação visual de um site. A forma como o conteúdo se apresenta no ecrã, a estrutura hierárquica das informações e a aplicação do design para a exibição das informações.

HTTP

é a sigla de *Hyper Text Transfer Protocol* (Protocolo de Transferência de Hipertexto). HTTP é um protocolo, ou seja, uma determinada regra que permite ao seu computador trocar informações com um servidor que abriga um site. Neste sentido, uma vez conectados sob esse protocolo, as máquinas podem receber e enviar qualquer conteúdo textual – os códigos que resultam na página acedida pelo navegador.

iOS

é um sistema operativo móvel, da Apple, desenvolvido originalmente para o *iPhone.*

IP

Internet Protocol (protocolo de internet) é uma identificação única de um dispositivo (computador, impressora, etc) conectado a determinada rede.

JMeter

é uma aplicação *Java desktop*, *OpenSource*, que simula o acesso de vários utilizadores, ao mesmo tempo, a um determinado sistema *Web*.

JUnit

é uma ferramenta *OpenSource* com suporte à criação de testes automatizados na linguagem de programação *Java*.

Mock

em desenvolvimento de *software*, *mocks* são objetos que simulam o comportamento de objetos reais de forma controlada. São normalmente criados para testar o comportamento de outros objetos. Ou seja, os objetos *mock* são objetos "falsos" que simulam o comportamento de uma classe ou objeto "real" para que possamos focar o teste na unidade a ser testada.

Notepad++

é um pequeno e rápido editor de texto de código aberto, para *Windows*, que permite trabalhar com arquivos de textos simples e código-fonte de diversas linguagens de programação.

OpenSource

em português código aberto. É código que fica disponível pelos *developers* de forma a poder ser alterado.

Pill Device

dispositivo (caixa) onde são guardados comprimidos e que se encontra ligado através de tecnologia, nomeadamente a um servidor.

Port

em português porta, é um ponto físico (*hardware*) ou lógico (*software*), no qual podem ser feitas conexões. Neste sentido, é um canal que permite a transferência de dados entre um dispositivo de entrada e o processador ou entre o processador e um dispositivo de saída.

Ride

é um *framework* de automação de teste genérico para testes de aceitação. Os testes são criados através da utilização de palavras-chave.É implementado com *Python* e suporta também *Jython* (JVM) e *IronPython* (NET).

Selenium

é uma ferramenta para testar aplicações *web* pelo browser de forma automatizada.

Servidor

é um sistema de computação centralizada que fornece serviços a uma rede de computadores.

Soap

Simple Object Access Protocol, em português protocolo simples de acesso a objetos. *Soap* é um protocolo para troca de mensagens entre as aplicações consumidoras e o *WebService*. Baseado na linguagem XML.

SoapUI

é uma ferramenta *OpenSource* escrita em *Java* cuja principal função é consumir e testar *WebServices*.

Sprint

o trabalho a executar é dividido em iterações, ou seja, em pequenos ciclos, num curto período de tempo.

Stand-up Meeting

é uma breve reunião realizada diariamente, por norma da parte da manhã, pelos colaboradores de uma equipa de desenvolvimento com o intuito de partilhar informações sobre o projeto, de forma a definir as suas atividades, atribuindo prioridades.

Team Leader

é uma pessoa que orienta, instrui, lidera e motiva um grupo de outros indivíduos com a finalidade de atingir objetivos e resultados específicos. Interliga as necessidades dos clientes àquelas que são as potencialidades da sua equipa.

Testes de *Software*

envolvem ações que vão do levantamento de requisitos até à execução do teste propriamente dito. É a investigação, exploração, utilização do *software* de forma a verificar se funciona como o determinado ou se apresenta erros.

Testes de Carga

usado para verificar o limite de dados processados pelo *software* até que ele não consiga mais processá-lo.

Testes de Usabilidade

verificam a facilidade da interação entre o utilizador e a aplicação, indicando se o *software* é intuitivo e de fácil compreensão.

Testes Funcionais

permitem testar as funcionalidades, requisitos, presentes na documentação, de forma a validar as funcionalidades descritas.

Testes Unitários

testam uma única unidade do sistema de maneira isolada, geralmente simulando as prováveis dependências que aquela unidade tem. São realizados com o intuito de validar dados válidos e inválidos através de I/O.

A grande vantagem destes testes é que sempre que surjam alterações no código, possibilita uma rápida deteção e eficiência no tratamento de erros que possam aparecer.

Testes Visuais

permitem detetar falhas ao nível visual de todos os elementos que compõem o *layout*.

UIAutomatorViewer

é uma ferramenta que faz parte das ferramentas do *Android* SDK para a criação de testes em *interface* gráfica, pertencendo a uma sessão específica dentro do universo *Android*.

WebService

é uma tecnologia baseada em XML e HTTP cuja principal função é disponibilizar serviços interativos na *Web* que podem ser acedidos (ou consumidos) por qualquer outra aplicação independente da linguagem ou plataforma em que a aplicação foi construída.

WSDL

Web Services Description Language (WSDL) é uma linguagem baseada em XML utilizada para descrever *Web Services*. Trata-se de um documento escrito em XML que além de descrever o serviço, especifica como acedê-lo e quais as operações ou métodos disponíveis.

XML

eXtensible Markup Language é um tipo de Linguagem Padronizada de Marcação Genérica, recomendada pela W3C, para a criação de documentos com dados organizados hierarquicamente, tais como textos, banco de dados ou desenhos vetoriais. O seu principal propósito é a facilidade de compartilhamento de informações através da internet.

XSD

arquivos XSD (*XML Schema Definition*) são usados para descrever o formato/padrão que um arquivo XML deve seguir, ou seja, ele tem que indicar quais *nodes* ele pode conter, quais *subnodes* e seus atributos.

Os elementos são declarados utilizando-se a *tag "element"*. Os principais atributos da *tag* são:

- *name*: nome do elemento;
- *type*: tipo de dados do elemento;

- *minOccurs*: mínimo de vezes que o elemento pode aparecer;
- *maxOccurs*: máximo de vezes que o elemento pode.

Apêndices

Apêndice 1 – Tabela de Testes Projeto RSI

Step Name	Descrição do step	Resultado esperado	Resultado Obtido	Observ
Step 1	Preencher com valores codigo-interno = 1526107 (fixo) codigoCuentaValorExterna = 02391000027480 (fixo) fechaOrdenDesde = 2015-12-01 fechaOrdenHasta = 2016-01-01	Retorna lista com dados	NOK	Como a lista foi retornada vazia por não existirem dados neste período, a resposta poderia ser mais clara e avisar que a lista está vazia.
Step 2	Preencher com valores codigo-interno = 1526107 (fixo) codigoCuentaValorExterna = 02391000027480 (fixo) fechaOrdenDesde = 2015-12-01 fechaOrdenHasta = 2016-02-01	Retorna lista com dados	OK	
Step 3	Preencher com valores codigo-interno = 1526107 (fixo) codigoCuentaValorExterna = 02391000027480 (fixo) fechaOrdenDesde = 2016-01-01 fechaOrdenHasta = 2016-02-01	Retorna lista com dados	OK	
Step 4	Preencher com valores codigo-interno = 1526107 (fixo) codigoCuentaValorExterna = 02391000027480 (fixo) fechaOrdenDesde = 2015-12-01 fechaOrdenHasta = 2016-04-01	Retorna lista com dados	OK	
Step 5	Preencher com valores codigo-interno = 1526107 (fixo) codigoCuentaValorExterna = 02391000027480 (fixo) fechaOrdenDesde = 2016-01-01 fechaOrdenHasta = 2016-02-01	Retorna lista com dados	OK	
Step 6	Preencher com valores codigo-interno = 1526107 (fixo) codigoCuentaValorExterna = 02391000027480 (fixo) fechaOrdenDesde = 2016-02-01 fechaOrdenHasta = 2016-03-01	Retorna lista com dados	OK	
Step 7	Preencher com valores codigo-interno = 1526107 (fixo) codigoCuentaValorExterna = 02391000027480 (fixo) fechaOrdenDesde = 2016-05-01 fechaOrdenHasta = 2016-07-01	Retorna lista vazia	OK	
Step 8	Preencher com valores codigo-interno = 1526107 (fixo) codigoCuentaValorExterna = 02391000027480 (fixo) fechaOrdenDesde = 2016-02-01 fechaOrdenHasta = 2016-01-01	Retorna lista vazia	OK	Na resposta surge erro 50 - por campos obrigatórios não preenchidos. A resposta deveria dizer mesmo que a fechaOrdenDesde é superior À fechaOrdenHasta.
Step 9	Valores codigo-interno = codigoCuentaValorExterna = fechaOrdenDesde = fechaOrdenHasta =	Retorna mensagem de erro 50 - campos obrigatórios não preenchidos	OK	
Step 10	Valores codigo-interno = 1526107 codigoCuentaValorExterna = fechaOrdenDesde = fechaOrdenHasta =	Retorna mensagem de erro 50 - campos obrigatórios não preenchidos	OK	
Step 11	Valores codigo-interno = 1526107 codigoCuentaValorExterna = 02391000027480 fechaOrdenDesde = fechaOrdenHasta =	Retorna mensagem de erro 50 - campos obrigatórios não preenchidos	OK	

Step 12	Valores codigo-interno = 1526107 codigoCuentaValorExterna = 02391000027480 fechaOrdenDesde = 2016-02-01 fechaOrdenHasta =	Retorna mensagem de erro 50 - campos obrigatórios não preenchidos	OK	
Step 13	Valores codigo-interno = 1526107 (fixo) codigoCuentaValorExterna = 02391000027480 (fixo) fechaOrdenDesde = 2016-02-01 fechaOrdenHasta = 2016-01-01	Retorna lista com dados	OK	Retorna erro 50 - campos obrigatórios não preenchidos. Poderia retornar erro mas indicando que a data está inválida ou indicar que fechaOrdenDesde é superior a fechaOrdenHasta.
Step 14	Datas inválidas Valores codigo-interno = 1526107 (fixo) codigoCuentaValorExterna = 02391000027480 (fixo) fechaOrdenDesde = 2016/01/01 fechaOrdenHasta = 2016/02/01	Retorna erro	OK	Retorna erro 50 - campos obrigatórios não preenchidos. Poderia retornar erro mas indicando que a data está inválida.
Step 15	Datas inválidas Valores codigo-interno = 1526107 (fixo) codigoCuentaValorExterna = 02391000027480 (fixo) fechaOrdenDesde = 01-01-2016 fechaOrdenHasta = 01-02-2016	Retorna erro	OK	Retorna erro 50 - campos obrigatórios não preenchidos. Poderia retornar erro mas indicando que a data está inválida.

Apêndice 2 – Tabela de Testes Projeto *AstraZeneca*

Process	Screen	"Visual Tests"	Sprint2_v1 (F/P) 08/06/2016	Observation	Sprint2_v2 (F/P) 13/06/20162	Observation2
Main View	Record Medicine	Title font	P		P	
Main View	Record Medicine	Title size	P		P	
Main View	Record Medicine	Title colour	F	must be dark blue	P	
Main View	Record Medicine	Left text view font	P		P	
Main View	Record Medicine	Left text view size	F	Must be smaller???	P	
Main View	Record Medicine	Left text view colour	P		P	
Main View	Record Medicine	Right image button rounded conners	P		P	
Main View	Record Medicine	Right image button size	P		P	
Main View	Record Medicine	Right image button empty colour line TODAY	P		P	
Main View	Record Medicine	Right image button empty colour line other days	P		P	
Main View	Record Medicine	Right image button taken	P		P	
Main View	Record Medicine	Right image button not taken	P		P	
Main View	Record Medicine	Historical Right "image" empty	P		P	
Main View	Record Medicine	Historical Right "image" taken	F	Size X	P	
Main View	Record Medicine	Historical Right "image" not taken	F	Size X	P	
Main View	Record Medicine	Size cell today - not taken	F	must be smaller	P	
Main View	Record Medicine	Size orange today - not taken	F	must be smaller	P	

Main View	Record Medicine	Text colour cell today - not taken	P		P	
Main View	Record Medicine	Text font cell today - not taken	P		P	
Main View	Record Medicine	Last 7 days Size cells day	P		P	
Main View	Record Medicine	Last 7 days Text Align	F	Not vertically centered	P	
Main View	Record Medicine	Historical Size cells day	F	must be bigger	P	
Main View	Record Medicine	Historical icon align	F	align with icon 7 days	P	
Main View	Record Medicine	When Tap a day colour change to dark blue	P		P	
Main View	Record Medicine	Correct Text	F	"**M**edicine" must be "**m**edicine"	P	
Main View	Progress	Man icon size	P		P	
Main View	Progress	Man icon colour	P		P	
Main View	Progress	myADAURAmeds text colour	P		P	
Main View	Progress	myADAURAmeds text font	F	wrong font	P	
Main View	Progress	myADAURAmeds position	F	base line align	P	
Main View	Progress	myADAURAmeds text size	F	??because text font is wrong...	P	
Main View	Progress	Clock icon size	P		P	
Main View	Progress	Clock icon colour	P		P	
Main View	Progress	Text font username	F	wrong font	P	
Main View	Progress	Text colour username	F	must be dark blue	P	

Main View	Progress	Text size username	F	??because text font is wrong…	P	
Main View	Progress	Button Logout Text font username	F	wrong font	P	
Main View	Progress	Button Logout Text colour username	F	must be dark blue	P	
Main View	Progress	Button Logout Text size username	F	??because text font is wrong…	P	
Main View	Progress	Layout Title size	P		P	
Main View	Progress	Title Text Font	P		P	
Main View	Progress	Title Text Size	P		P	
Main View	Progress	Title Text Colour	P		P	
Main View	Progress	"TODAY" Button size	P		P	
Main View	Progress	"TODAY" Button border colour	P		P	
Main View	Progress	"TODAY" Button colour	P		P	
Main View	Progress	"TODAY" Button Text Font	P		P	
Main View	Progress	"TODAY" Button Text Size	P		P	
Main View	Progress	"TODAY" Button Text Colour	P		P	
Main View	Progress	Right icon "Colour" near "TODAY" Button	P		P	
Main View	Progress	Right icon "Size" near "TODAY" Button	P		P	
Main View	Progress	Left text view rounded conners	P		P	
Main View	Progress	Left text view colour ambar	P		P	

Main View	*Progress*	*Left text view size*	***P***		***P***	
Main View	*Progress*	*Left text view text size*	***P***		***P***	
Main View	*Progress*	*Left text view text font*	***P***		***P***	
Main View	*Progress*	*Left text view colour*	***P***		***P***	
Main View	*Progress*	*Text font*	***P***		***P***	
Main View	*Progress*	*Text size*	***P***		***P***	
Main View	*Progress*	*Text colour*	***P***		***P***	
Main View	*Progress*	*Right text view rounded conners*	***P***		***P***	
Main View	*Progress*	*Right text view colour ambar*	***P***		***P***	
Main View	*Progress*	*Right text view text size*	***P***		***P***	
Main View	*Progress*	*Right text view text font*	***P***		***P***	
Main View	*Progress*	*Right text view text colour*	***P***		***P***	
Main View	*Progress*	*Time Line colour line*	***P***		***P***	
Main View	*Progress*	*Time Line bullet colour selected*	***F***	*must be dark blue*	***P***	
Main View	*Progress*	*Time Line bullet colour not selected*	***P***		***P***	
Main View	*Progress*	*Time Line circules colour line*	***P***		***P***	
Main View	*Progress*	*Time Line circules selected bullet colour*	***P***		***P***	
Main View	*Progress*	*Time Line circule selected "Today" bullet colour*	***P***		***P***	

Main View	*Progress*	*Time Line bullet size*	***F***	*must be smaller*	***P***	
Main View	*Progress*	*Correct Text*	***F***	*"Log **O**ut" must be "Log **o**ut"*	***P***	
Main View	*Progress*	*Positions*	***F***	*_- space between time line and text must be smaller* *_- space between TODAY Button and TOP layout must be smaller*	***F***	*_- space between time line and text must be smaller* *_- space between TODAY Button and TOP layout must be smaller*