IPG Politécnico da Guarda
Escola Superior de Tecnologia e Gestão

RELATÓRIO DE ESTÁGIO

Curso Técnico Superior Profissional
em Desenvolvimento de Aplicações Informáticas

Sérgio José dos Santos Guelho

julho | 2018

INSTITUTO POLITÉCNICO DA GUARDA

ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO

SÉRGIO JOSÉ DOS SANTOS GUELHO

RELATÓRIO PARA A OBTENÇÃO DO DIPLOMA DE TÉCNICO SUPERIOR PROFISSIONAL EM DESENVOLVIMENTO DE APLICAÇÕES INFORMÁTICAS

2018

RELATÓRIO FINAL DE ESTÁGIO CURRICULAR DESENVOLVIDO NA EMPRESA PET UNIVERSAL - LVS

Relatório de Estágio apresentado à
Escola Superior de Tecnologia e
Gestão da Guarda com vista à
obtenção do grau de Técnico
Superior de Desenvolvimento de
Aplicações Informática**s**

Orientador: Professor José Quitério Figueiredo

GUARDA

2018

IDENTIFICAÇÃO

Estagiário: Sérgio José dos Santos Guelho
Número: 1012496
Email: sergioguelho@gmail.com

Curso: Técnico Superior Profissional de Desenvolvimento de Aplicações Informáticas

Docente Orientador: José Alberto Quitério Figueiredo
UTC: Informática

Entidade onde decorreu o estágio

Nome: Pet Universal - LVS
Morada: IEUA - Incubadora de Empresas da Universidade de Aveiro - PCI.Creative Science Park Aveiro Region - Via do Conhecimento, Edifício Central
Código Postal: 3830-352
Localidade: Ílhavo
Telefone: 234 243 820

Tutor

Nome: Luís Pinto
Cargo/Função: CEO

Período de Estágio

26 de Fevereiro a 15 de Junho

Guelho, S. J. S. (2018). *Relatório de Estágio Curricular desenvolvido na empresa Pet Universal - LVS.* Relatório de Estágio Curricular, Escola Superior de Tecnologia e Gestão do Instituto Politécnico da Guarda, Guarda, Portugal.

À minha mãe, a minha referência,
por todo o apoio incondicional prestado,
ao longo da minha vida académica.

AGRADECIMENTOS

Agradeço antes de mais ao orientador de empresa Eng.º Luís Pinto, pelo apoio prestado durante todo o processo de estágio, assim como por todos os ensinamentos partilhados. Agradeço igualmente aos demais colegas da empresa que me tão bem me acolheram e integraram desde o início deste processo de estágio.

Gostaria igualmente de agradecer ao professor e orientador da Escola Superior de Tecnologia e Gestão do Instituto Politécnico da Guarda, o professor José Quitério Figueiredo, por toda a informação partilhada e por todos os conselhos transmitidos.

Por fim, quero agradecer à minha família, e à minha namorada por todo o apoio prestado durante a realização deste documento, assim como por todo o apoio prestado ao longo da realização deste curso.

RESUMO

O principal objetivo da apresentação e elaboração do relatório de estágio curricular passa pela análise reflexiva de todas as atividades relevantes desenvolvidas durante o estágio curricular. Tendo sempre como base o contexto onde estive inserido, tentei que através de uma análise reflexiva de todo o processo fosse possível expor as principais dificuldades sentidas e a forma como foram superadas. Quanto às aprendizagens realizadas durante o processo de estágio, estas são também pormenorizadamente descritas neste documento, assim como todas as situações que contribuíram para o meu desenvolvimento tanto a nível profissional como a nível pessoal. Para além da reflexão de todo o processo, é também contemplada uma dimensão descritiva do processo de estágio, de modo a realizar uma breve mas adequada contextualização da minha ação. Na parte final do documento, optei por aprofundar mais os trabalhos desenvolvidos durante o estágio, uma aplicação móvel concebida do zero para servir de extensão ao software já existente, conhecido por "Hospi", e o novo software desenvolvido pela empresa que irá ter o seu lançamento no final do mês de Agosto.

Palavras-Chave: Estágio Curricular. Desenvolvimento de Aplicações Informáticas. Reflexão. Relatório Final de Estágio Curricular. Hospi mobile.

Índice

1. Introdução

A necessidade de elaboração do presente documento, relatório de estágio curricular, surge no âmbito da unidade curricular de estágio a realizar durante o 2º semestre do 2º ano do Curso Técnico Superior Profissional de Desenvolvimento de Aplicações Informáticas do Instituto Politécnico da Guarda, referente ao ano letivo de 2017/2018. Este estágio marca o final de todo um processo de aprendizagem, e marca também o início de uma carreira na área das tecnologias da informação. Esta não foi apenas a última etapa da minha preparação como aluno antes do início do desempenho profissional na área da informática, mas sim a primeira de um longo processo de aprendizagem que dificilmente terá fim.

O estágio curricular foi realizado sob orientação do Eng.º Luís Pinto, CEO da empresa Pet Universal - LVS, sediada na incubadora de empresas de Universidade de Aveiro. O estágio como já foi referido, é uma das unidades curriculares previstas no plano de estudos do Curso Técnico Superior Profissional de Desenvolvimento de Aplicações Informáticas para o segundo semestre do segundo ano.

O relatório de estágio será organizado em quatro capítulos, mais concretamente num primeiro em que será realizada uma contextualização da prática desenvolvida, um segundo capítulo em que será descrito, o plano de estágio e o seu cumprimento, um terceiro em que serão relatadas, analisadas e refletidas todas as ferramentas e tecnologias aprendidas, e um quarto e último capítulo onde será feita referência aos projetos em que estive diretamente envolvido ao longo destes quatro meses. No final foi feita uma conclusão sobre o trabalho desenvolvido e as aprendizagens desenvolvidas. O presente relatório deverá descrever claramente todos os acontecimentos relevantes ocorridos durante o processo de estágio.

O objetivo principal da elaboração deste relatório de estágio será sempre o de realizar uma reflexão o mais acurada possível sobre todo o processo, procurando sempre novas pistas para poder melhorar no futuro.

Devido ao carácter evolutivo que qualquer tipo de aprendizagem deve assumir, sob consequência de ser bem sucedida, vou optar por fazer referência a esse mesmo processo evolutivo, sempre na tentativa de perceber qual o caminho percorrido desde o ponto de partida até ao final, avaliando ao mesmo tempo a qualidade de todo o processo, acreditando sempre que o meu processo formativo ainda não terminou.

2. Contextualização da Prática Desenvolvida

2.1. Expectativas Iniciais

A partir do momento em que iniciei a minha formação em Desenvolvimento de aplicações informáticas, comecei a desenvolver o gosto pela área da programação. Defini então como objetivo pessoal, desenvolver e adquirir conhecimentos e aptidões para que pudesse no futuro desenvolver software. Desde então que todas as minhas expectativas se focaram no momento em que iria poder desempenhar essa função, marcando o início do estágio curricular o momento de início no desempenho dessas funções. Como tal, certas expectativas inerentes a todo este processo foram surgindo, sendo relevante analisar as mesmas antes de abordar todo o processo de estágio.

O ano de estágio marca o primeiro momento de aplicação prática, dos conteúdos teóricos adquiridos ao longo de todo um processo de formação inicial orientado para este momento, daí a importância de todo o processo de estágio, sendo o local e empresa em que o mesmo iria decorrer um dos fatores determinantes para o sucesso do mesmo.

A entidade de acolhimento para realizar o meu estágio curricular foi a empresa LVS - Pet Universal Lda. A escolha desta empresa deveu-se à sua localização na cidade de Ílhavo, próxima da cidade de Aveiro e da sua respetiva Universidade, conhecida nacionalmente pelo seu excelente trabalho na área da engenharia de computadores. Outro dos fatores que pesou no momento de escolher a empresa de estágio foi o facto da Pet Universal ser uma empresa de desenvolvimento de software, assumindo-se assim como o local ideal para quem quer trabalhar no futuro a desenvolver software poder estagiar e preparar-se para o futuro. Por fim, outro dos fatores relevantes que me levou a optar pela LVS - Pet Universal Lda. para realizar o meu estágio, foi o facto de a mesma ser uma *startup*. Ponderei este fator, e a conclusão a que cheguei foi que poderia realizar funções mais variadas e com maior responsabilidade numa *startup* do que numa empresa maior, uma vez que uma *startup*, sendo uma empresa nova e de dimensão menor, todos os recursos interessam e são aproveitados para as mais diversas tarefas, permitindo assim desenvolver um maior leque de competências que não seria possível noutro contexto.

Após ter concluído um ciclo de cerca de um ano e meio de estudos esta foi também a primeira oportunidade de colocar em prática todos os conhecimentos adquiridos, sempre com

o objetivo de desenvolver as minhas competências técnicas desta forma de uma maneira mais prática. Tive sempre a consciência de que teria de assumir um espírito aberto a novos conteúdos e continuar a assumir-me como um autodidata ativo, mesmo ciente das dificuldades que poderiam advir dessa postura. Todas as dificuldades inerentes a este processo de aprendizagem, transformaram-se ao longo do tempo em oportunidades de crescer e me tornar mais autónomo e capaz.

A estratégia teria também um papel fundamental no sucesso, ou não, do processo de estágio, e com base num plano de estágio simples, mas bem orientado, parti na expectativa de através da divisão do mesmo em pequenas metas alcançáveis e mensuráveis, de modo a ter um fio condutor que guiasse todo o meu processo. A expectativa era grande, uma vez que os objetivos iniciais constantes no plano de estágio eram extremamente desafiadores e aliciantes para qualquer jovem que inicia a sua experiência no mundo profissional de desenvolvimento de software.

Depois de um percurso académico de quase 18 anos penso que a crítica passa a ser encarada de forma natural. Durante todo o meu percurso académico, o sentido crítico dos meus professores influenciou diretamente a minha aprendizagem, contando com o professor orientador e o meu tutor da empresa como responsáveis por desempenhar essa tarefa durante este semestre de estágio. Considero esse papel indispensável para que haja uma constante reciclagem profissional aliada a uma constante renovação de conhecimento e correção de hábitos antigos que não seriam os mais corretos.

Parti na expectativa de transformar a interação com os meus colegas de empresa, mais concretamente em relação à sua capacidade de análise crítica e reflexiva, em informação relevante acerca de todo o trabalho por mim realizado. Para que tal acontecesse, era necessário que a interação entre mim e todos os elementos da empresa fosse desenvolvida e estabelecida num clima harmonioso proporcionando um bom ambiente caracterizado pela entre ajuda e cooperação no processo de formação individual.

A empresa e mais concretamente a equipa de desenvolvimento era composta por profissionais bastante experientes, já com muitos anos de experiência a trabalhar na área. Este fator fez com que no momento do início desta aventura, que é o estágio curricular, partisse na expectativa de poder contar sempre com a sua opinião, influenciando a mesma direta ou indiretamente a orientação da minha atividade, através da partilha de experiências e conselhos extremamente relevantes sobre o contexto da empresa, e sobre as especificidades da função a desempenhar.

2.2. Caracterização da Empresa

Após escolha da empresa na qual deveria realizar o estágio curricular foi iniciado esse mesmo processo, com uma primeira tarefa de caracterização da empresa e do meio em que a mesma está inserida. Esta tarefa enquadrada nas tarefas preparatórias é de elevada importância para que seja realizado um correto planeamento da ação, adequado sempre às características específicas da empresa. A empresa Pet Universal - LVS foi fundada em 2015 e está sediada no parque da ciência e inovação de Ílhavo, inaugurado no dia 6 de março de 2018, e no qual se encontra inserida a incubadora de empresas da Universidade de Aveiro, da qual a Pet Universal faz parte. A empresa dispõe de uma sala principal onde toda a equipa desenvolve o seu trabalho, e de outra sala mais pequena onde são realizadas as reuniões com clientes e reuniões de equipa mais prolongadas.

A equipa está organizada em três áreas, divididas espacialmente na sala por "ilhas". As áreas são a tecnológica/desenvolvimento, administração e gestão de recursos humanos e marketing, vendas e comunicação. Cada uma das três áreas tem uma pessoa responsável por coordenar toda a atividade dos membros inseridos nessa pequena equipa.

No meu caso, fiquei inserido na equipa de tecnológica e de desenvolvimento de soluções de TI, sendo o responsável pela minha ação o Engenheiro Luís Pinto e o coordenador técnico da equipa de desenvolvimento o Sr. Vítor Martins. Uma das grandes vantagens deste tipo de organização da empresa é a proximidade com os demais membros da mesma área de ação e de trabalho, que permite uma mais fácil articulação e consequente aumento de produtividade.

É importante referir também que tanto o Eng. Luís Pinto como o Vítor Martins são duas pessoas com mais de 5 anos de experiência em desenvolvimento de software, e como tal toda a sua experiência foi extremamente importante para esclarecer as dúvidas que foram surgindo ao longo do processo de estágio.

Segundo Robehmed (2013), uma *startup* é uma empresa empreendedora que acaba de surgir com o intuito de colmatar uma necessidade específica de mercado, desenvolvendo um modelo de negócio normalmente associado a um produto específico ou serviço ou aliado a algum processo ou plataforma já existentes. A escalabilidade é algo extremamente importante para uma *startup*, uma vez que o principal objetivo de uma *startup* é estabilizar o modelo de negócio e acentuar o seu crescimento ao longo de um curto espaço de tempo.

Sendo a Pet Universal uma *startup* que tem como principal objetivo colmatar uma necessidade específica relacionada com o mercado veterinário, a mesma enfrenta todos os desafios comuns a qualquer *startup*. Como seria previsível, o número de elementos que constitui a equipa é pequeno, cerca de 15 pessoas, pelo que todos os recursos são necessários e valorizados.

3. Análise Reflexiva sobre o planeamento

3.1. Planeamento

No início do estágio foi definido, em conjunto com o tutor da empresa o seguinte plano de estágio:

Objetivos para o estágio: Desenvolver uma aplicação informática possível de implementar em plataformas web e móveis. Aquisição dos diferentes conhecimentos necessários para a conceção de aplicações desse género.

Atividades principais:
» Planear e projetar sistemas de bases de dados de acordo com os requisitos
» Desenvolver uma aplicação para web
» Desenvolver uma aplicação para dispositivos móveis
» Planear todas as fases de desenvolvimento de uma aplicação informática

Linguagens a abordar:
» JavaScript
» Java
» SQL

Ordem de eventos:
1- Desenvolvimento de uma aplicação móvel;
2- Desenvolvimento de uma aplicação para web;
3- Orientação para os processos de manutenção das aplicações.

3.2. Execução do plano de estágio

Após concluir o estágio é possível verificar que o plano foi cumprido quase na sua totalidade, tendo apenas ficado uma lacuna quanto a questões relacionadas com programação web e mais concretamente na abordagem a javascript. Essa falha no cumprimento do planeamento inicial, deve-se essencialmente à falta de tempo originada pelos prazos de lançamento do novo produto da Pet Universal, que obrigaram toda a equipa a trabalhar no seu desenvolvimento de modo a poder entregar o produto ao cliente a tempo, e fizeram com que este processo de aprendizagem não tivesse sido possível de iniciar.

O estágio teve duas fases distintas, sendo que numa primeira fase do estágio, aprofundei os conhecimentos sobre programação para dispositivos móveis, mais concretamente para android, de modo a poder desenvolver uma solução móvel inexistente na gama de produtos da Pet Universal. Esta fase de aprendizagem foi extremamente produtiva, tendo aprofundado todos os conteúdos já interiorizados e adquirido novos conceitos essenciais para o desenvolvimento de qualquer tipo de aplicação móvel. Como resultado foi possível desenvolver uma aplicação móvel, simples e intuitiva, que irei descrever mais detalhadamente no capítulo 4.

Na segunda fase, foi-me lançado o desafio de colaborar com os restantes membros da equipa de desenvolvimento no projeto da nova solução modular da Pet Universal. Nesta fase, foi possível aplicar muitos dos conhecimentos de Java, adquiridos ao longo dos três semestres anteriores. Este desafio obrigou-me a sair da minha zona de conforto, e a procurar novos conteúdos sobre programação em Java, SQL e até sobre o desenvolvimento de WebServices REST. Todo este processo será igualmente descrito de forma mais completa e detalhada no capítulo 4.

4. Ferramentas e tecnologias utilizadas

Neste capítulo irei mencionar todas as ferramentas e tecnologias que foram utilizadas ao longo do estágio. Algumas foram utilizadas diariamente, e algumas apenas foram utilizadas para complementar funcionalidades que outras tecnologias ou ferramentas poderiam não oferecer.

4.1. Android

Android é um sistema operativo móvel, atualmente desenvolvido pela google, e que tem como base Linux. É o sistema operativo mais utilizado em todo o mundo, e está disponível em diversos smartphones e tablets. Como a Google é a principal responsável pelo seu desenvolvimento, os serviços como Gmail são normalmente facilmente acedidos através de dispositivos com este sistema operativo. É projetado principalmente para dispositivos móveis, sendo que o interface, na maioria dos casos, é feito através do contato manual com o ecrã do dispositivo que dispõe do sistema operativo. A última versão é a 8.1, Oreo, que foi lançada em 5 de dezembro de 2017.

Logotipo do sistema operativo Android

4.1.1 Android Studio

O Android Studio é um ambiente de desenvolvimento integrado especificamente criado para desenvolver aplicações para a plataforma android. Foi lançado no ano de 2013 e é gratuito pelo que se assume como uma excelente ferramenta para qualquer programador que pretenda desenvolver para Android. Para compilar os projetos é utilizado o Gradle, que serve como sistema de automação de compilação.

Logotipo da ferramenta Android Studi

4.1.1.1 Drawer Layout

Segundo a documentação disponível para *developers* de Android, o Drawer Layout é "um painel que exibe as principais opções do painel de navegação do aplicativo na borda esquerda da tela. Fica oculto a maior parte do tempo, mas é revelado quando o utilizador desliza um dedo a partir da borda esquerda da tela, ou, no nível superior do aplicativo, o usuário toca no ícone do aplicativo na barra de ações."

4.2. XML

XML é uma linguagem de marcação, e que tem como principal objetivo facilitar a troca de dados através da internet. Serve como facilitador da interação entre diversas linguagens de programação, sendo que estabelece um formato hierárquico para a criação de documentos. É extremamente importante quando falamos em comunicação entre bases de dados, isto porque através desta linguagem de marcação duas bases de dados diferentes conseguem comunicar com maior facilidade.

4.3. Java

Java é uma linguagem orientada a objetos diferente de outras linguagens de programação por ser "compilada para um *bytecode* que é interpretado por uma máquina virtual". Começou a ser desenvolvida em 1991 pela Sun Myscrosystems, e desde então que tem vindo a crescer, até se tornar na linguagem de programação mais popular. Características como a orientação a objetos específicos do dia a dia e a sua independência da plataforma são algumas das suas vantagens quando comparamos com outras linguagens de programação. É igualmente possível associar *frameworks* como HIbernate ou Spring para facilitar o desenvolvimento de aplicações, sendo que os IDE's mais populares são o Eclipse e o NetBeans, sendo o segundo *open source*. A última versão de java é a SE 11, sendo que as anteriores como a 7 e 8 são ainda muito populares no desenvolvimento de software.

4.3.1. NetBeans

O NetBeans é um IDE, ou seja, um ambiente de desenvolvimento integrado *open-source*, e que permite desenvolver software em diferentes tipos de linguagens. O projeto foi iniciado em 1996 tendo sido depois adquirido pela Sun Microsystems, para colmatar a falta de uma solução para um IDE. Para além de facilitar o trabalho de todos os programadores na escrita e compilação de código, o NetBeans possui diversas bibliotecas, e a sua documentação é grande pelo que se torna fácil encontrar informação sobre este ambiente de desenvolvimento.

Logotipo da ferramenta NetBeans

4.3.2. Hibernate

Hibernate é uma *framework* para mapear as relações entre objetos que está escrito na linguagem java, entre outras. A grande utilidade desta ferramenta prende-se na facilidade com que são mapeados os atributos dos de uma bases de dados relacional habitual e os objetos criados na linguagem java. A utilização de anotações facilita todo este processo e a complexidade que possa vir a existir entre os programas com base em Java orientado a objetos e que utilizem bases de dados relacionais. Transforma classes de Java para poderem ser criadas tabelas de dados, ajudando igualmente a passar os dados escritos em java para SQL.

4.4. SQL

SQL é a linguagem de consulta estruturada de dados que facilita a pesquisa de informação em bases de dados relacionais. Uma consulta SQL tem como objetivo "especificar" uma resultado e não o método para o alcançar. Existem várias bases de dados que utilizam esta linguagem como Oracle, MySql, SQLite entre outras. É uma linguagem considerada relativamente simples de aprender, isto devido às suas características declarativas, em que o utilizador não estabelece um procedimento, mas procura obter um resultado para a sua pesquisa ou seleção de dados de uma base de dados.

Imagem alusiva à linguagem SQL

4.4.1. MariaDB

MariaDB foi desenvolvida pelos responsáveis do MySQL e é uma das bases de dados mais populares do mundo, muito devido às suas características *open source*, ou seja, sem custos e com código aberto disponível para todos. MariaDB assume-se como uma alternativa viável ao MySQL, especialmente pela sua rapidez e robustez, possuindo ainda muitas ferramentas adicionais que lhe conferem uma grande versatilidade e adaptabilidade. É uma base de dados relacional, que utiliza um interface SQL para manipular e aceder aos seus dados. MariaDB também já inclui recursos em JSON, o que abre novas portas.

4.5. Git

Git é uma ferramenta de controlo de versões, que serve para controlar as alterações feitas num computador local num projeto em que diversas pessoas participam ativamente. A sua principal função passa pela gestão de todo o código desenvolvido para uma aplicação, e a estabilização do mesmo, através da criação de diferentes versões dos desenvolvimentos realizados por diferentes elementos integrantes de um projeto. Git pode ser utilizado através da sua própria consola, "Git bash" ou através de um IDE que suporte Git, como é o caso do NetBeans. É *open source* e encontra-se neste momento na versão 2.17, e é atualmente uma das ferramentas de controlo de versões de projetos mais populares.

Logotipo da ferramenta de controlo de versões git

4.6 Visual Studio Team Services

O Visual Studio Team Services oferece um serviço de gestão de projetos para diferentes linguagens de programação. Permite juntar Git e a metodologia Agile num mesmo serviço o que facilita a gestão dos projetos de equipa. Dispõe de várias ferramentas de planeamento aliadas à gestão e controlo de versões com base em Git. Existem versões base desta ferramenta e versões avançadas. Permite igualmente uma partilha fácil, rápida e eficaz de projetos entre colaboradores de uma equipa, ainda que localizados remotamente.

4.7. Agile

Os princípios de uma metodologia de desenvolvimento de *software* ágil, em inglês *Agile*, prendem-se com a garantia da satisfação do consumidor, entrega frequente de soluções de *software* já em estado funcional em vez de demorar um período maior de tempo a serem entregues. Este tipo de metodologia trabalha através da disponibilização de versões aos clientes, que por sua vez, através da sua opinião, poderão contribuir no momento da conceção de futuros planos de desenvolvimento.

4.7.1. Scrum

Scrum é uma das técnicas mais conhecidas da metodologia de desenvolvimento de *software* ágil, e também uma das mais popularmente utilizadas pelas empresas deste sector. Os projetos pertencentes a uma scrum são divididos em Sprints, que tendem a durar entre uma semana a um mês, em que a relação esforço tempo é fulcral para a conclusão com sucesso de cada uma das sprints. Todas as atividades que a equipa terá de desenvolver ao longo de uma sprint constam numa lista de atividades, *Backlog*, definidas em reunião de planeamento realizada antes do início de cada sprint. Estas atividades são também ordenadas através de níveis de prioridade, do ponto de vista da prioridade demonstrada pelo cliente final. As *daily scrums* são curtas reuniões realizadas diariamente com o objetivo de realizar um ponto de situação sobre a execução de cada sprint, e sobre as tarefas a executar por cada membro da equipa, e os impedimentos para que as mesmas sejam executadas com sucesso.

4.8. Slack

Slack é uma ferramenta de comunicação para equipas, muito utilizada para facilitar a comunicação entre membros de equipas de empresas. Tem como base *cloud*, e regista todas as conversas realizadas entre membros de um projeto ou empresa, e em canais específicos de conversação que podem ser iniciados por colaboradores de modo a organizar os temas de conversação por tópicos. Representa assim uma boa solução para facilitar a interação dos membros de uma equipa num local de trabalho, e promove a articulação dos mesmos com o intuito duma mais eficaz colaboração em projetos.

Logotipo do Slack

4.9. Jenkins

O Jenkins foi desenvolvido em Java e serve para a automação da compilação de código novo ou editado, desenvolvido e inserido num repositório já existente. Facilita uma solução de integração contínua das funcionalidades desenvolvidas pela equipa de desenvolvimento, sendo que após cada alteração feita no repositório em que um membro da equipa está a trabalhar, o Jenkins inicia uma nova tentativa de compilação, alertando para eventuais erros ou falhas que poderão surgir no momento de compilar o código com as alterações submetidas.

4.10. Postman

A aplicação Postman serve de "braço direito" aos programadores no momento de testar os interfaces desenvolvidos pelos programadores (APIs - *Application Programming Interfaces*). A sua facilidade de utilização no momento de fazer pedidos através do protocolo HTTP é extremamente importante no momento de desenvolver serviços para aplicações, podendo testar a sua capacidade de execução e fiabilidade.

Logotipo da ferramenta Postman

4.11. Serviços REST

O "*Representational State Transfer* (REST)"é um estilo de arquitetura que define um conjunto de restrições a serem usadas para criar serviços da web. Os Serviços da Web que estão em conformidade com o estilo arquitetural REST, ou serviços da Web RESTful, fornecem interoperabilidade entre sistemas de computador na Internet." Num serviço REST, um recurso é acedido através de um URI específico que depois produz uma resposta num formato que vai desde HTML a JSON. Os métodos mais comuns para realizar operações e pedidos são so métodos GET, POST, PUT e DELETE, conhecidos como os "métodos HTTP CRUD predefinidos".

5. Trabalho desenvolvido

Neste capítulo irei abordar o trabalho desenvolvido ao longo do estágio, abordando de forma mais pormenorizada os dois projetos em que participei ativamente.

5.1. Aplicação móvel

Ainda antes de ter iniciado o estágio, o Engenheiro Luís Pinto, o meu tutor na empresa, lançou-se o desafio de pensar e projetar uma aplicação móvel que pudesse servir de complemento a um dos produtos pertencentes ao portfólio da empresa, o Hospi. Optei por começar a estudar e aprofundar conhecimentos sobre desenvolvimento para aplicações móveis android, ainda antes de iniciar o meu estágio, tendo realizado cursos online e efetuado diversas pesquisas bibliográficas sobre a matéria.

Quando iniciei o processo de estágio, e logo na primeira reunião que tive com o Eng. Luís Pinto, para além de definirmos o plano de estágio, elaboramos um plano para a conceção de uma nova aplicação móvel, que fosse de encontro às necessidades relatadas pelos principais utilizadores dos produtos da Pet Universal, os veterinários. As dificuldades resolvidas através do desenvolvimento de uma nova solução móvel seriam:

- Consulta rápida de tarefas a realizar para cada um dos animais internados;
- Possibilidade de facilmente visualizar os internamentos em clínicas diferentes;
- A criação de alertas que pudessem minimizar as falhas no cumprimento de tarefas;
- Possibilidade de aceder às tarefas a realizar no telefone, sendo a aplicação adaptada a esse tipo de interação.

Para além destas sugestões de melhoria, o tutor pediu-me igualmente que a aplicação obedecesse a alguns requisitos estruturais tais como:

- Um *SplashScreen*;
- Uma página de Login;
- Utilização de um Navigation Drawer;
- Efetua-se pedidos através das APIs existentes;
- Simples, rápida e intuitiva;

Estes requisitos obrigaram-me a pesquisar mais sobre eles, uma vez que a maioria não fez parte da minha formação teórica inicial, o que como foi referido inicialmente neste capítulo, me fez estudar e pesquisar mais aprofundadamente sobre o tema.

Abordando o projeto por partes, e de acordo com os requisitos irei descrever mais detalhadamente o trabalho realizado de modo a cumprir o plano:

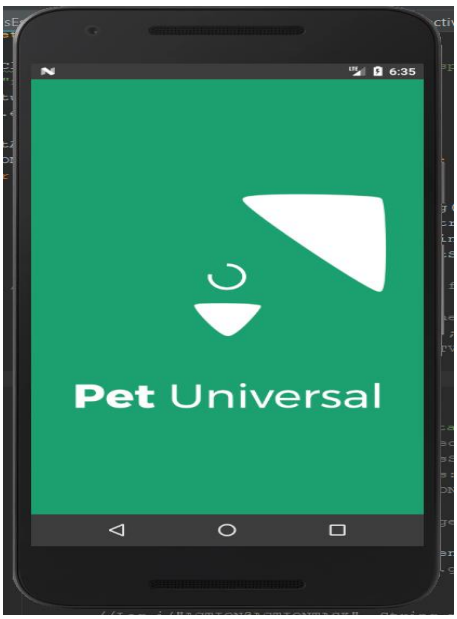
1. Estudo da aplicação existente e da sua API

Antes de poder começar a programar a nova aplicação móvel, fui obrigado a estudar a API do "Hospi", de modo a perceber como poderia fazer os meus pedidos à base de dados. Esta primeira fase levou-me a estudar os pedidos GET e PUT, do protocolo HTTP, de modo a poder obter todos os dados em formato JSON e poder utilizar os mesmos para mostrar informação na aplicação móvel.

Esta aplicação não tem qualquer tipo de ligação a base de dados, sendo que toda a informação é acedida através da api, utilizando o método GET do protocolo HTTP, sendo depois tratada e exibida de acordo com os requisitos propostos.

2. SplashScreen

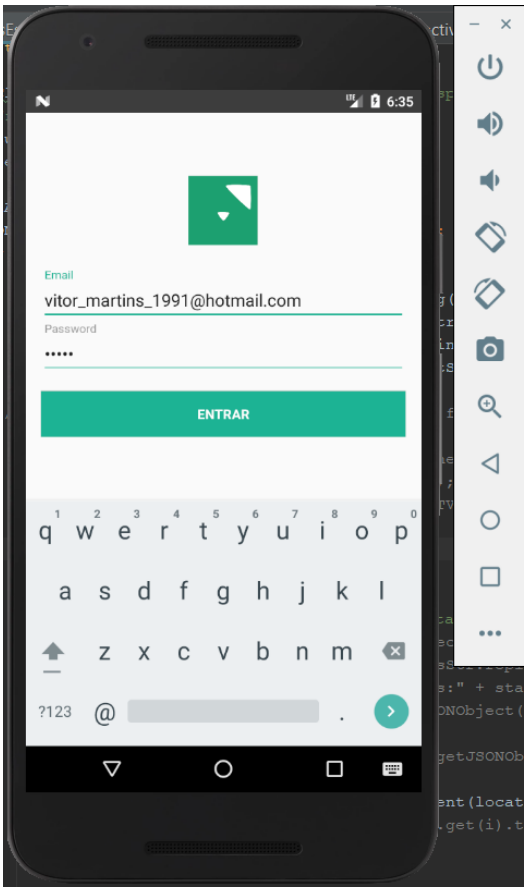
Para criar um *SplashScreen*, ou ecrã inicial, optei por uma atividade do tipo *fullscreen*, em que incluí o logo da Pet Universal e uma *progressBar* de modo a dar a entender ao utilizador que a aplicação já está a correr.



SplashScreen da Aplicação

Optei igualmente por utilizar este momento para testar a ligação à internet, e caso o utilizador não esteja ligado à internet, exibir uma mensagem a alertar precisamente para esse facto.

3. Login



Página de login da Aplicação

A atividade de *Login* representou um grande desafio, uma vez que nunca tinha feito algo parecido e que me ia obrigar a interagir pela primeira vez com a API da Pet Universal.

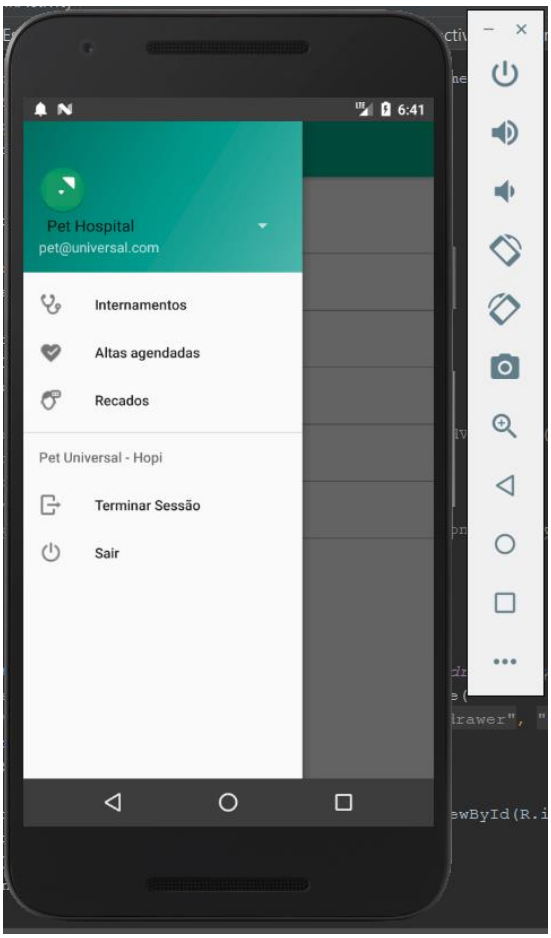
Através de uma AsyncTask, em primeiro lugar fazia POST de um username e de uma password, e depois de executar esta AsyncTask, guardava o token por ela devolvido, assim como o ID do utilizador que acabava de fazer Login.

Estes dados eram posteriormente passados para a atividade principal através de um intent com um input.extra. Neste contexto, o token assume um papel extremamente importante uma vez que o mesmo iria ser necessário para continuar a fazer pedidos à API durante a execução das restantes atividades da aplicação.

4. Naviagation Drawer

Depois de começar a desenvolver os primeiros requisitos para a aplicação móvel, e após reunião com o responsável da empresa pelo design e comunicação, foi-me pedido que alterasse um pouco a interface, mais concretamente o seu design e apresentação para o utilizador.

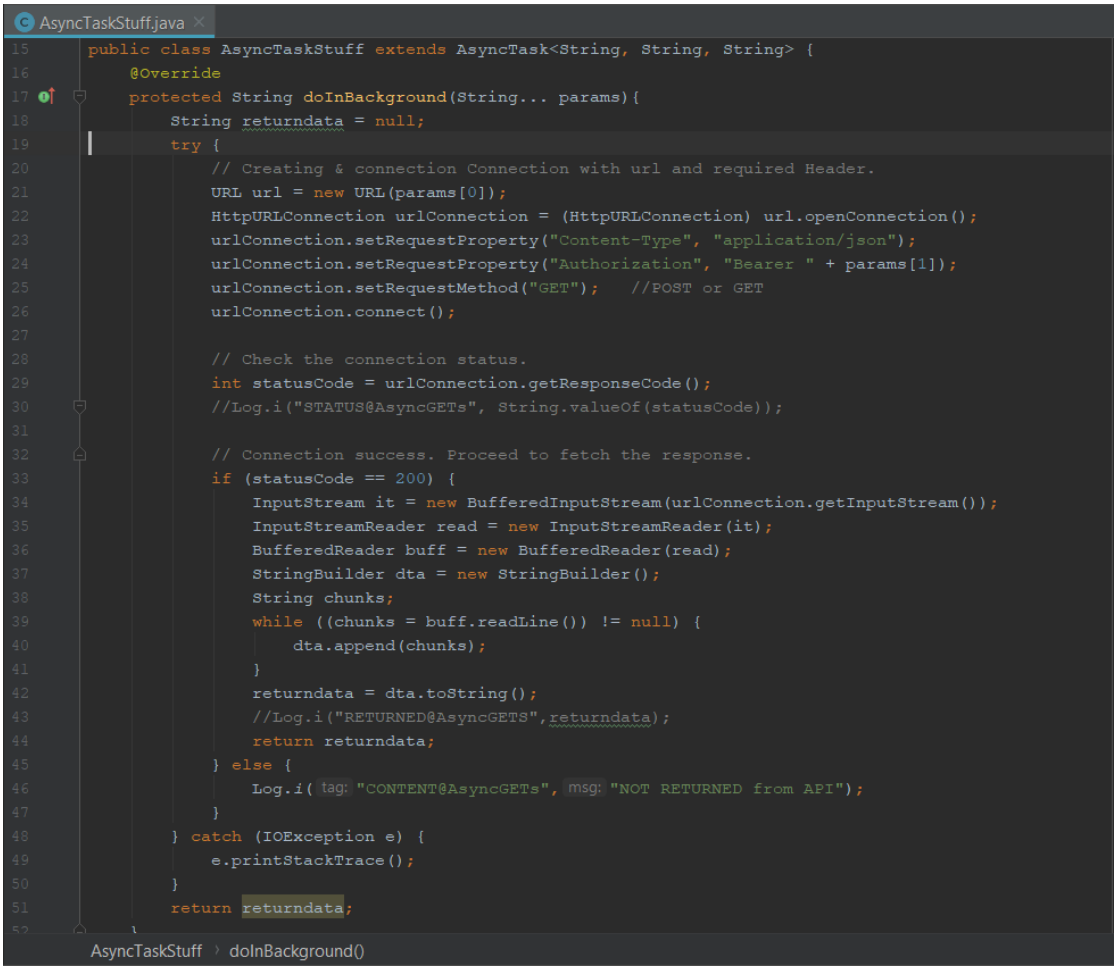
Optámos então por utilizar um *Navigation drawer*, de modo a organizar a informação a que o utilizador poderia ter acesso durante a sua experiência na aplicação. É de salientar que melhorou substancialmente a apresentação da aplicação, assim como a facilidade de interação com a mesma. A organização dos conteúdos num menu lateral interativo facilita bastante o processo de familiarização com a aplicação facilitando a sua utilização.



Navigation Drawer da Aplicação

5. Pedidos através de AsyncTask

Para fazer um pedido optei por criar uma classe chamada AsyncTaskStuff que herda da classe abstrata AsyncTask como é possível ver na seguinte imagem:



Classe AsyncTackStuff

Neste bloco de código, está declarado um método GET, que tem como *headers* um token, e que caso o código de estado do pedido tenha o valor 200, devolve o conteúdo acedido através de um URL específico, que deverá vir em formato JSON.

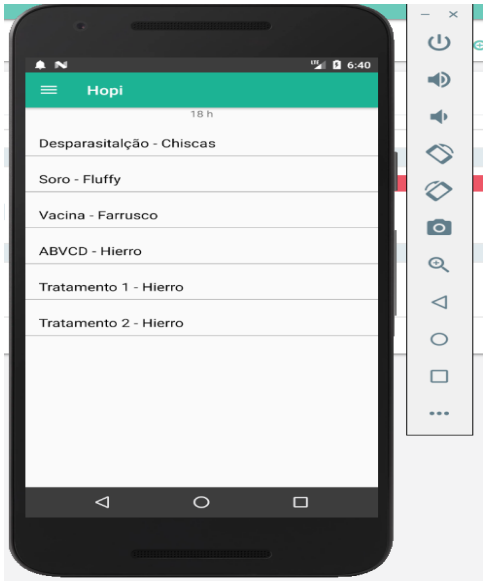
Em baixo vemos mais um exemplo de utilização desta classe AsyncTaskStuff:

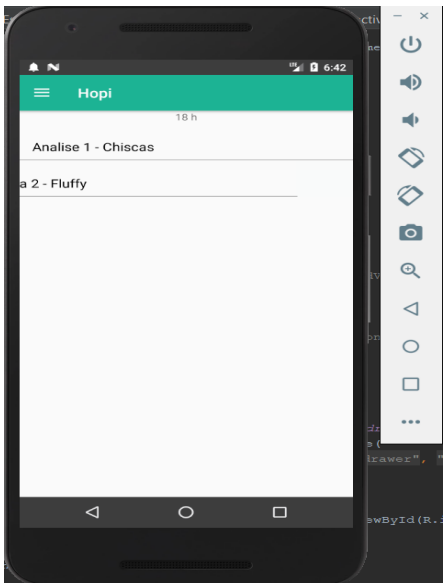
```
MainActivity.java

338
339     private void showInternmentTasks(int item) {
340         nomeTarefas.clear();
341         clinicIDMain = clinicsIDFromJson.get(item);
342
343         Log.i( tag: "DATE@HORA", currentDateAndTime.substring(10, 13));
344         for (int j = 0; j < internmentId.size(); j++) {
345             String getInternActions = "http://qa.petuniversal.com/hospi/api/tasks?internment=" + internmentId.get(j);
346             AsyncTaskStuff getActions = new AsyncTaskStuff();
347             getActions.execute(getInternActions, token);
348             try {
349                 if (getActions.get() != null) {
350                     Log.i( tag: "ACTIONS@HORA", getActions.get());
351                     JSONArray arr = new JSONArray(getActions.get());
352                     Log.i( tag: "ARR", arr.toString());
353                     for (int i = 0; i < arr.length(); i++) {
354                         Log.i( tag: "ARR@LENGHT", String.valueOf(arr.length()));
355                         if(arr.getJSONObject(i).has( name: "started")){
356                         String startedAll = arr.getJSONObject(i).getString( name: "started");
357                         Log.i( tag: "ACTION@STARTED",  msg: "      started:" + startedAll);
358                         Log.i( tag: "DATE@startedALLHORA", startedAll.substring(10, 13));
359                         if (startedAll.substring(10, 13).equals(currentDateAndTime.substring(10, 13))) {
360                             String name = arr.getJSONObject(i).getString( name: "name");
361                             nomeTarefas.add(name);
362                             animalEntrance.add(internmentId.get(j).toString());
363                         }
364                     }
365                     }
366                 } else Log.i( tag: "NULL@HORA",  msg: "Request 1 API is null");
367             } catch (InterruptedException | ExecutionException | JSONException e1) {
368                 e1.printStackTrace();
369             }
370             getActions.cancel( mayInterruptIfRunning: true);
371
372         }
373         animalNames();

MainActivity › showInternmentTasks()
```

Exemplo de uma AsyncTask

6. Lista de tarefas

Para listar as tarefas a realizar por internamento, optei por dispor a informação numa *RecyclerView*, com o intuito de poder implementar o *swap* para apagar as tarefas que, entretanto, vão sendo realizadas como demonstram as seguintes imagens:

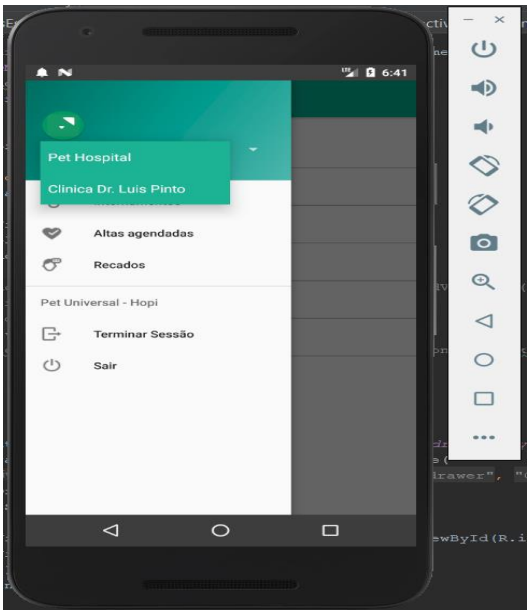




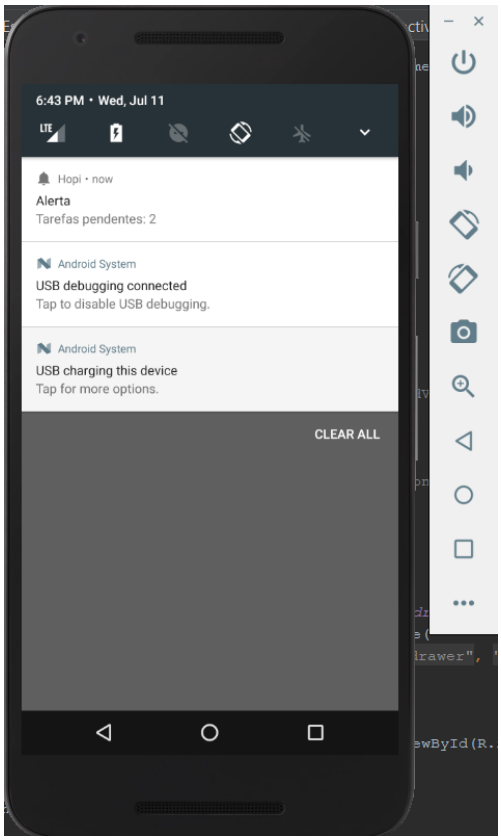
Lista de tarefas

7. Mudança de clínica

Um veterinário ou funcionário pode trabalhar em mais do que uma clínica, e como tal poderá consultar todas as tarefas a realizar relacionadas com todas as clínicas com que colabora.



Lista de clinicas de um utilizador



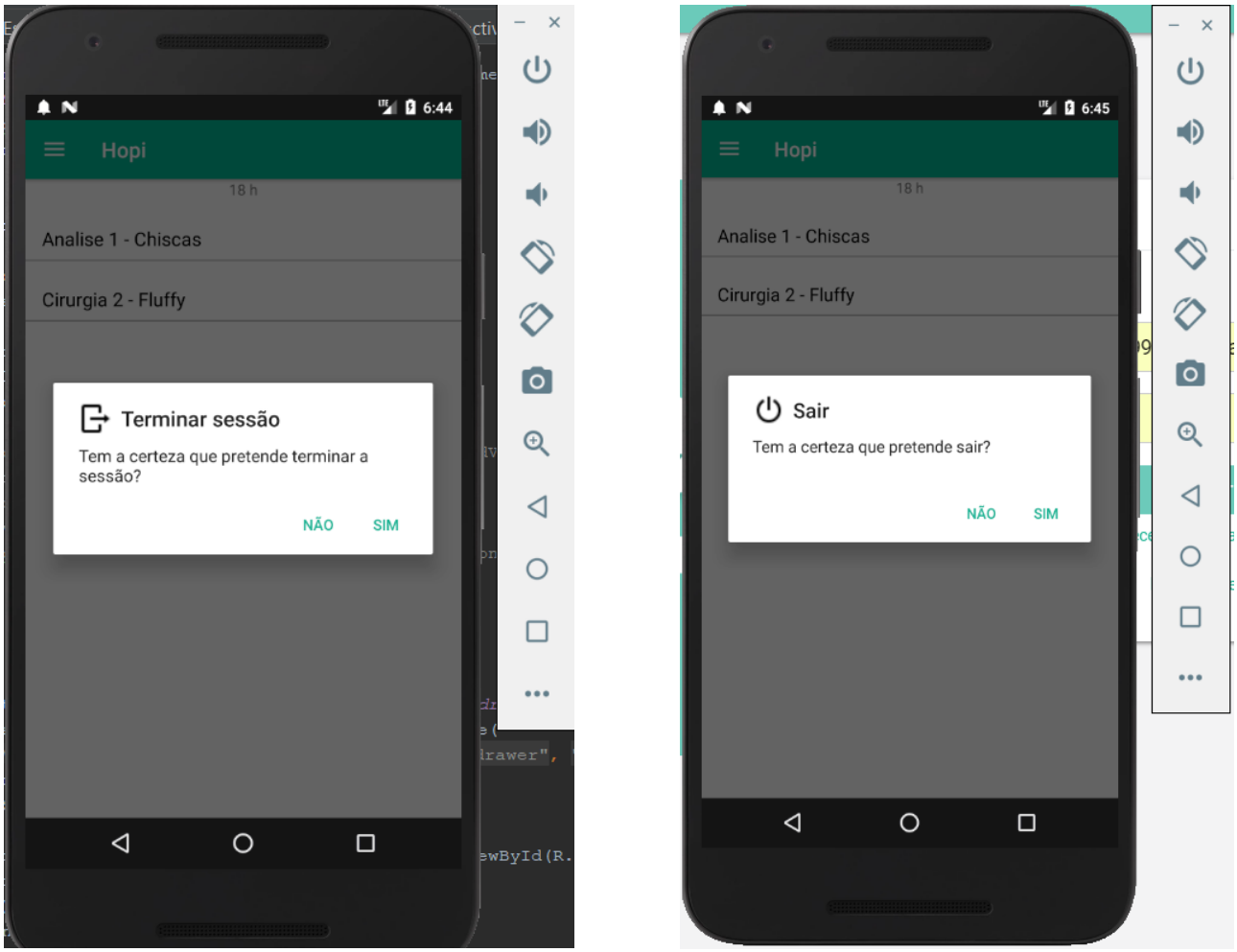
Alertas criados

8. Alertas

Por fim, foram adicionados alertas para que os funcionários não se esqueçam de realizar qualquer tarefa. Sempre que a aplicação abre é emitido um alerta com o número de tarefas a realizar, e cinco minutos antes da hora final para realizar a tarefa é lançado outro alerta.

9. Terminar sessão e sair

Por fim foram ainda tidas em conta as possibilidades do utilizador querer terminar sessão e iniciar a mesma com outra conta, ou simplesmente sair da aplicação, havendo sempre uma caixa de confirmação da ação escolhida.



Validação de terminar sessão e sair

5.2. Pet Universal - Solução modular

A meio do mês de abril, foi-me lançado um novo desafio, o de participar ativamente no desenvolvimento do novo produto da empresa, mais concretamente a trabalhar no seu *backend*. Este novo produto, com o nome da própria empresa, tem um *backend* em Java e um *frontend* em EmberJS, e como uma das linguagens mais abordadas ao longo da parte teórica do curso foi Java, foi possível contribuir para o projeto ao programar exatamente nesta linguagem.

As tarefas desenvolvidas estiveram maioritariamente relacionadas com os seguintes constituintes do *backend* do software:

- Entidades
- Facades
- Controllers
- DTO's
- DTS's
- Services

Quando entrei no projeto, o mesmo já ia numa fase mais avançada, o que fez com que o trabalho inicial de definir a estrutura de dados já tivesse sido realizado, e como tal apenas tive de me adaptar ao estado atual do mesmo. Para todos os constituintes do *backend* anteriormente referidos, foram criadas classes abstratas das quais herdariam outras do mesmo tipo, facilitando assim o trabalho da equipa de desenvolvimento ao poupar linhas de código comum a classes similares, e que estava compilado numa classe abstrata específica desse tipo.

Entidades

Sendo o *backend* considerado como a camada responsável por guardar, aceder e organizar dados de uma aplicação, o mesmo estava organizado de acordo com este pressuposto. As entidades representam as tabelas da base de dados, e os dados que serão persistidos durante a execução da aplicação. No momento da criação de uma nova entidade torna-se necessário para além de definir os seus atributos, mapear as suas relações com as outras já existentes. Para tal foram utilizadas anotações como é possível verificar na imagem seguinte:

```java
@Entity
public class Animal extends AbstractEntity implements Serializable {

    private static final long serialVersionUID = 1L;

    /*
     * Personal
     */
    @OneToMany(mappedBy = "patient", fetch = FetchType.LAZY, orphanRemoval = true, cascade = CascadeType.REMOVE)
    private List<PatientAllergy> allergies;

    @ManyToOne(fetch = FetchType.LAZY)
    private Species species;

    @Enumerated(EnumType.ORDINAL)
    private Sex sex;

    private Boolean sterilized;

    private Integer temperament;

    private String pedigree;

    @Temporal(TemporalType.TIMESTAMP)
    private Calendar deathdate;

    @Temporal(TemporalType.TIMESTAMP)
    private Calendar birthdate;

    @ManyToOne(fetch = FetchType.LAZY)
    private Breed breed;
```

Exemplo dos atributos de uma entidade

No exemplo anterior é possível verificar relações um para muitos e vice-versa, enumerações entre outros. Estas relações são obviamente fulcrais para qualquer base de dados relacional.

De seguida eram criados três construtores diferentes para a entidade, sendo um vazio, um apenas contendo o id, e um terceiro completo com todos os atributos da classe assim como os herdados da seguinte entidade abstrata:

```java
@MappedSuperclass
public abstract class AbstractEntity implements UniverseEntity {

    private static final long serialVersionUID = 1L;

    @Id
    @Column(updatable = false, nullable = false)
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Basic(optional = false)
    @Column(nullable = false)
    @Size(min = 1, message = "Invalid name size")
    private String name;

    @ManyToOne(fetch = FetchType.LAZY)
    private User creator;

    @Temporal(TemporalType.TIMESTAMP)
    @Column(updatable = false, nullable = false)
    private Calendar created;

    @ManyToOne(fetch = FetchType.LAZY)
    private User updater;

    @Temporal(TemporalType.TIMESTAMP)
    @Column(nullable = false, updatable = false, columnDefinition = "TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP")
    private Calendar updated;
```

Atributos da abstract entity

Exemplo de construtores de uma entidade:

```
/*
 * Contructors
 */
public Animal() {...3 lines }

public Animal(Long id) {...3 lines }

public Animal(List<PatientAllergy> allergies,
        Species species,
        Sex sex,
        Boolean sterilized,
        Tail tailType,
        Integer temperament,
        String pedigree,
        Float height,
        String furDescription,
        String furColor,
        String furColor2,
        String furColor3,
        String furLength,
        Long furType,
        String eyes,
        Calendar deathdate,
        Calendar birthdate,
        Breed breed,
        Image avatar,
        Boolean tattoo,
        Long tattooNumber,
        String passport,
        Boolean insurance,
        String insuranceType,
        String census,
        String siraID,
        String chip,
        Customer owner,
        Boolean ownerPrefered,
        Diet diet,
        Environment environmentType,
        String name, User creator, Calendar created, User updater, Calendar updated, Long id) {...34 lines }
```

Exemplo de construtores de uma entidade

Depois eram criados os getters e setters como no exemplo seguinte:

```
/*
 * Getters/Setters
 */
public List<PatientAllergy> getAllergies() {...3 lines }

public void setAllergies(List<PatientAllergy> allergies) {...3 lines }

public Species getSpecies() {...3 lines }

public void setSpecies(Species species) {...3 lines }

public Sex getSex() {...3 lines }

public void setSex(Sex sex) {...3 lines }

public Boolean getSterilized() {...3 lines }

public void setSterilized(Boolean sterilized) {...3 lines }

public Integer getTemperament() {...3 lines }

public void setTemperament(Integer temperament) {...3 lines }

public String getPedigree() {...3 lines }

public void setPedigree(String pedigree) {...3 lines }

public Calendar getDeathdate() {...3 lines }

public void setDeathdate(Calendar deathdate) {...3 lines }

public Calendar getBirthdate() {...3 lines }

public void setBirthdate(Calendar birthdate) {...3 lines }

public Breed getBreed() {...3 lines }

public void setBreed(Breed breed) {...3 lines }
```

Exemplo de uma getters/setters

Facades

De seguida eram criadas facades, que serviriam para interagir com a base de dados do ponto de visto dos pedidos a fazer. Esta camada de interação com a base de dados representava todas os tipos de pedidos que lhe poderiam ser feitos, desde o *create* ao *find*.

```java
public abstract class AbstractFacade<T> implements Facade<T> {

    private final Class<T> entityClass;

    /*
     * Constructors
     */
    public AbstractFacade(final Class<T> entityClass) {...3 lines }

    /*
     * DAO
     */
    @Override
    public List<T> find(final Map<String, Object> options) {...5 lines }

    @Override
    public T retrieve(final Long id) {...7 lines }

    public List<T> findRange(int[] range) {...8 lines }

    public Long count() {...7 lines }

    @Override
    public void create(final T entity) {...3 lines }

    @Override
```

Métodos da abstract facade

Exemplo de uma facade:

```java
@Stateless
public class AnimalFacade extends AbstractFacade<Animal> {

    @PersistenceContext
    private EntityManager em;

    public AnimalFacade() {
        super(Animal.class);
    }

    @Override
    public List<Animal> find(Map<String, Object> options) {
        if (options.containsKey("owner")) {
            Long ownerID = (Long) options.get("owner");
            /*
             * By person all
             */
            return getByOwner(ownerID);
        }

        throw new AssertionError("Cannot list all animals. Please report this!");
    }

    @Override
    protected EntityManager getEntityManager() {
        return em;
    }

    private List<Animal> getByOwner(final Long ownerID) {
        return getEntityManager().createQuery(
                "SELECT a FROM Animal a WHERE a.owner.id = :ownerID", Animal.class)
                .setParameter("ownerID", ownerID)
                .getResultList();
    }
}
```

Exemplo de uma facade

Controller’s

Os *controller’s,* como o próprio nome indica servem para controlar as interações com a base de dados, e impor regras à interação com a base de dados, mais concretamente quanto aos dados inseridos, editados etc.

Na classe abstrata controllerCRUD, existem métodos create e doCreate, entre outros, que servem precisamente para regular a informação que é inserida na base de dados, uma vez que apenas é criado algo após passar pelo método doCreate e não apenas pelo Create, realizando assim uma espécie de controlo sobre a informação.

DTO’s e DTS’s

Os *data transfer objects*, conhecidos vulgarmente por DTO’s, servem como intermediários da informação que é partilhada entre o utilizador e o sistema, ou seja, embora tenham uma estrutura semelhante às entidades, os DTO’s podem e devem ter uma estrutura diferente, de acordo com a informação que se pretende passar ao utilizador. Eles servem igualmente para moldar a estrutura de informação que é recebida vinda do utilizador, antes de a mesma ser transformada numa entidade já existente.

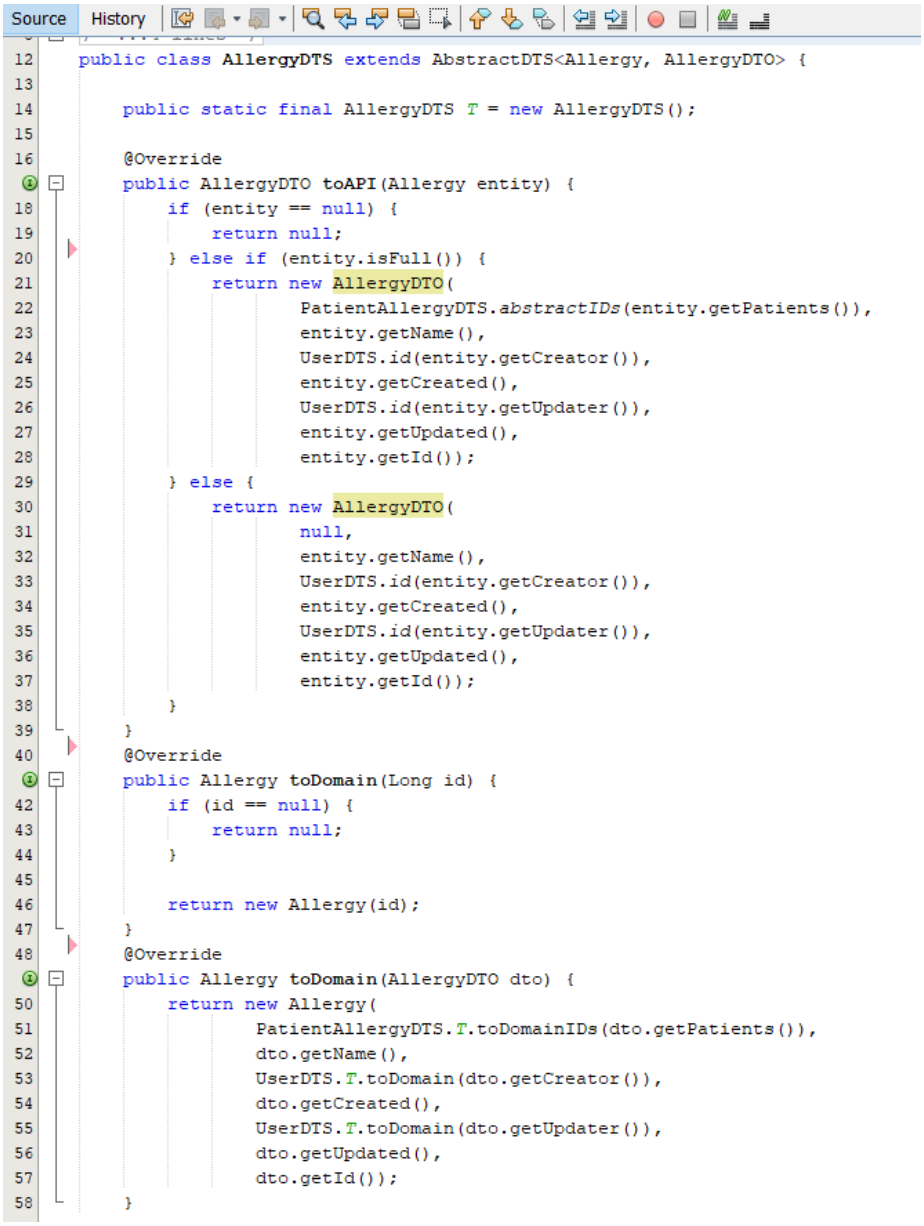
Exemplo de um DTO:

```
Source  History
1  package com.petuniversal.hopi.api.dto;
2
3  import eu.lpinto.universe.api.dto.AbstractDTO;
4  import java.io.Serializable;
5  import java.util.Calendar;
6  import java.util.List;
7
8  /**...4 lines */
12 public class AllergyDTO extends AbstractDTO implements Serializable {
13
14     public static final long serialVersionUID = 1L;
15
16     private List<Long> patients;
17
18     /*
19      * Constructors
20      */
21     public AllergyDTO() {...2 lines }
23
24     public AllergyDTO(Long id) {...3 lines }
27
28     public AllergyDTO(List<Long> patients, String name, Long creator, Calendar created, Long updater, Calendar updated, Long id) {
29         super(name, creator, created, updater, updated, id);
30         this.patients = patients;
31     }
32
33     /*
34      * Getters/Setters
35      */
36     public List<Long> getPatients() {
38         return patients;
39     }
40     public void setPatients(List<Long> patients) {
42         this.patients = patients;
43     }
44 }
45
```

Exemplo de um DTO

E para isso servem os DTS's, *data transfer services*, que fazem exatamente essa transformação de informação de entidade para uma estrutura a apresentar ao utilizador e transformam os dados recebidos numa estrutura com que a entidade se identifique, e com a informação que deve ser armazenada do lado do sistema.

Exemplo de um DTS:



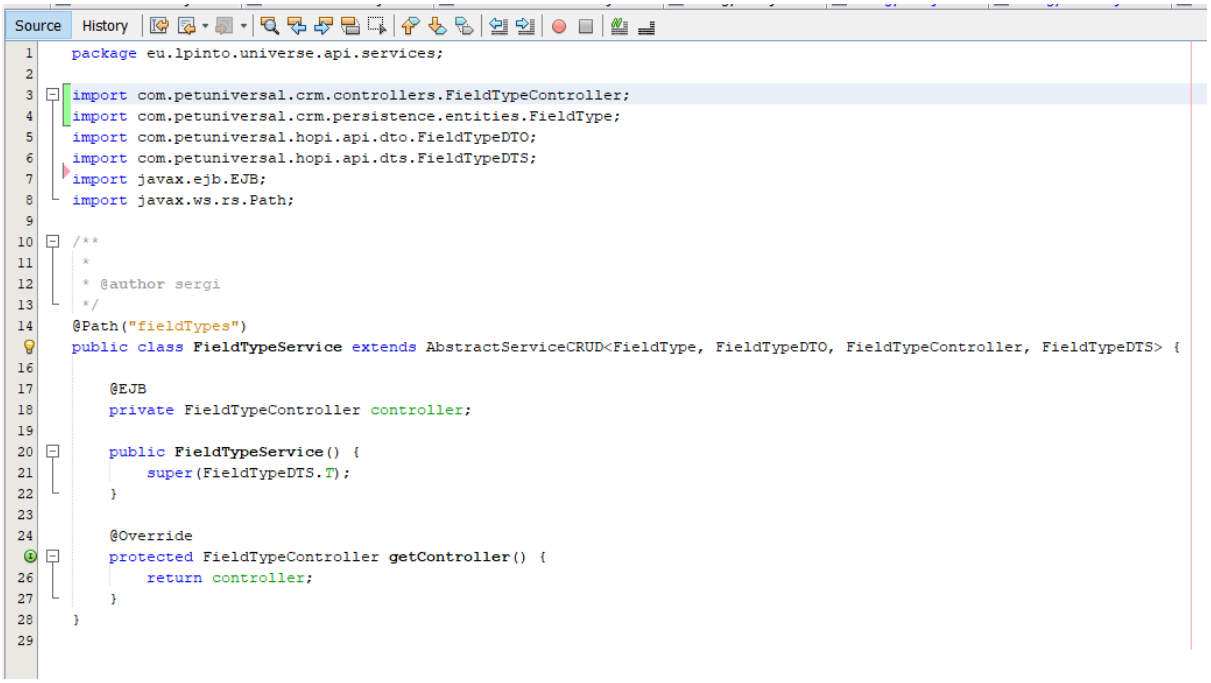
```
Source  History
12  public class AllergyDTS extends AbstractDTS<Allergy, AllergyDTO> {
13
14      public static final AllergyDTS T = new AllergyDTS();
15
16      @Override
17      public AllergyDTO toAPI(Allergy entity) {
18          if (entity == null) {
19              return null;
20          } else if (entity.isFull()) {
21              return new AllergyDTO(
22                      PatientAllergyDTS.abstractIDs(entity.getPatients()),
23                      entity.getName(),
24                      UserDTS.id(entity.getCreator()),
25                      entity.getCreated(),
26                      UserDTS.id(entity.getUpdater()),
27                      entity.getUpdated(),
28                      entity.getId());
29          } else {
30              return new AllergyDTO(
31                      null,
32                      entity.getName(),
33                      UserDTS.id(entity.getCreator()),
34                      entity.getCreated(),
35                      UserDTS.id(entity.getUpdater()),
36                      entity.getUpdated(),
37                      entity.getId());
38          }
39      }
40      @Override
41      public Allergy toDomain(Long id) {
42          if (id == null) {
43              return null;
44          }
45
46          return new Allergy(id);
47      }
48      @Override
49      public Allergy toDomain(AllergyDTO dto) {
50          return new Allergy(
51                  PatientAllergyDTS.T.toDomainIDs(dto.getPatients()),
52                  dto.getName(),
53                  UserDTS.T.toDomain(dto.getCreator()),
54                  dto.getCreated(),
55                  UserDTS.T.toDomain(dto.getUpdater()),
56                  dto.getUpdated(),
57                  dto.getId());
58      }
```

Exemplo de um DTS

Serviços

Os web services utilizados eram do tipo RESTful pelo que a sua associação ao protocolo HTTP foi naturalmente estabelecida, e neste caso utilizando uma notação para definir o caminho (*path*) do serviço, eram utilizados os métodos GET, POST, PUT e DELETE, normalmente denominados por CRUD.

Um exemplo de um service:



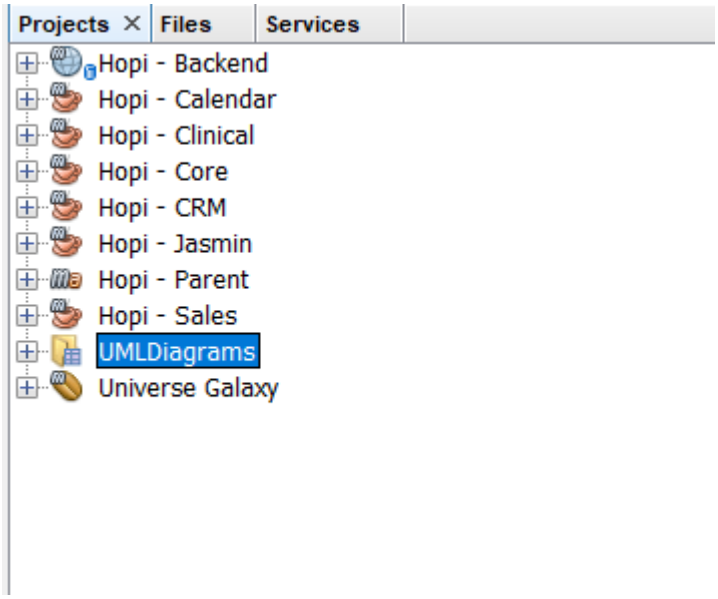
```
Source  History
1  package eu.lpinto.universe.api.services;
2
3  import com.petuniversal.crm.controllers.FieldTypeController;
4  import com.petuniversal.crm.persistence.entities.FieldType;
5  import com.petuniversal.hopi.api.dto.FieldTypeDTO;
6  import com.petuniversal.hopi.api.dts.FieldTypeDTS;
7  import javax.ejb.EJB;
8  import javax.ws.rs.Path;
9
10 /**
11  *
12  * @author sergi
13  */
14 @Path("fieldTypes")
   public class FieldTypeService extends AbstractServiceCRUD<FieldType, FieldTypeDTO, FieldTypeController, FieldTypeDTS> {
16
17     @EJB
18     private FieldTypeController controller;
19
20     public FieldTypeService() {
21         super(FieldTypeDTS.T);
22     }
23
24     @Override
       protected FieldTypeController getController() {
26         return controller;
27     }
28 }
29
```

O desenvolvimento de serviços Web RESTful é barato e portanto, tem uma barreira muito baixa para adoção, sendo possível utilizar ferramentas de desenvolvimento como o NetBeans IDE, para reduzir ainda mais a complexidade do seu desenvolvimento. Neste desenvolvimento voltei a recorrer a uma classe abstrata já existente denominada por AbstractServiceCRUD, como mostro na imagem seguinte:

```java
public abstract class AbstractServiceCRUD<E extends UniverseEntity, D extends UniverseDTO, C extends AbstractControllerCRUD<E>, DTS extends AbstractDTS<E, D>> extends Abstract

    private static final Logger LOGGER = LoggerFactory.getLogger(AbstractServiceCRUD.class);
    private final DTS dts;

    /*
     * Constructors
     */
    public AbstractServiceCRUD(final DTS dts) {...3 lines }

    /*
     * Services
     */
    @GET
    @Asynchronous
    @Produces(value = MediaType.APPLICATION_JSON)
    public void find(@Suspended final AsyncResponse asyncResponse,
                     final @Context UriInfo uriInfo,
                     final @HeaderParam(value = "userID") Long userID) throws PreConditionException {

        /*
         * Setup
         */
        final String requestID = UUID.randomUUID().toString();
        Map<String, Object> options = new HashMap<>(uriInfo.getQueryParameters().size());
        options.put("request", requestID);

        MultivaluedMap<String, String> queryParameters = uriInfo.getQueryParameters();
        for (String key : queryParameters.keySet()) {
            List<String> values = queryParameters.get(key);

            if (values != null && !values.isEmpty() && values.size() == 1) {

                try {
                    Long l = Long.valueOf(values.get(0));
                    options.put(key, l);
                } catch (NumberFormatException ex) {
                    options.put(key, values.get(0));
                }
```

Abstract service

Este software é considerado modular, uma vez que está organizado precisamente por módulos de categorias diferentes, como mostra a seguinte imagem:



Lista de módulos do projeto

Muitas das tarefas desenvolvidas tiveram como base uma integração feita com o "Primavera", um software de faturação já existente, e que nos ajudaria a completar o modulo de faturação, mas que foi denominado por Jasmin.

Como se tratava de um software já existente, com uma estrutura de dados específica, foi necessário estudar a documentação toda sobre a API do Primavera, e através da adequação dos DTO's e DTS's tornou-se possível comunicar através da API do Primavera, enviando os dados do cliente, necessários para por exemplo a emissão de uma fatura, e recebendo da parte do Primavera, os dados referentes à fatura emitida, e que deveriam ser persistidos em base de dados do nosso lado. Este assumiu-se como um grande desafio, uma vez que nunca tinha feito algo parecido, o que me obrigou numa primeira fase a estudar uma API totalmente desconhecida, mas com uma documentação bastante completa e clara, e numa segunda fase perceber todos os passos a realizar para realizar uma integração do género.

6. Conclusões

6.1. Dificuldades Sentidas e Necessidade de Formação

Com o período de estágio terminado, e depois de todas as aprendizagens realizadas ao longo deste ano letivo, permaneço com consciência de que o meu processo de formação como profissional das tecnologias da informação não termina aqui, tendo a perfeita noção de que várias aprendizagens deverão ainda ser levadas a cabo na busca do objetivo final da formação de qualquer bom profissional, que é a busca pela competência. Termino o período de estágio desgastado, mas com a noção de que cumpri com as minhas funções enquanto programador, tendo sido ao mesmo tempo leal com os objetivos iniciais do plano de estágio, com os quais me tinha comprometido.

As principais dificuldades sentidas ao longo do processo de estágio, surgiram da falta de experiência característica de qualquer aluno, que aplica pela primeira vez todos os conteúdos que adquiriu ao longo da formação inicial no contexto prático. A dificuldade inicial em perceber como utilizar os conhecimentos adquiridos numa fase inicial de estudante, em nada contribuem para um início mais simples do estágio pedagógico. Não se deve nunca cair no erro de pensar que o seu processo de formação está terminado, sob consequência de cair num processo de estagnação quanto à evolução das suas competências.

Sendo a "experiência condição base para a competência", penso que as minhas futuras intervenções devem ser orientadas para este facto, definindo como principal objetivo a aquisição de novas competências que apenas a experiência prática na área poderá proporcionar. Em contraste com o ano de estágio, para que seja possível a manutenção de um processo eficaz de aprendizagem, terei de manter um espírito de abertura elevado, devendo recorrer a processos de autocrítica que me permitam refletir sobre o trabalho realizado e retirar conclusões sobre o mesmo.

6.2. Impacto do estágio na realidade e contexto empresarial

O processo de estágio tem um elevado impacto sobre a realidade de um profissional que se encontra a concluir a fase de estudante, marcando o momento da sua primeira experiência em contexto empresarial. Contudo não é apenas o estagiário que sente este impacto em relação ao seu processo de formação, sendo verificável também o impacto que o estágio

desenvolvido tem em relação à realidade e contexto empresarial. O perfil do estagiário por norma caracteriza-se por um maior dinamismo em relação aos restantes colaboradores, muito também, devido à fase da vida em que o estagiário se encontra. Esta capacidade de dinamismo tem um impacto com alguma relevância na empresa em que é realizado o estágio curricular. O estagiário, derivado do facto de ter realizado o seu processo de formação mais recentemente, encontrasse mais atualizado no que toca a abordagens e metodologias específicas, podendo o mesmo servir de ponte de ligação entre a entidade formadora e o contexto profissional, em que a maioria dos colaboradores estão inseridos. É muitas vezes percetível, a influência, por vezes indireta, que a ação dos estagiários tem em relação ao processo de atualização dos restantes colaboradores.

6.3. Experiência Profissional e Pessoal

Através de um processo de constante reflexão sobre as decisões e opções tomadas foi-me possível adquirir experiência profissional e pessoal ao longo deste estágio, através da aplicação da minha formação teórica no contexto prático. Não apenas a reflexão contribuiu para que conseguisse adquirir novas experiências profissionais e pessoais relacionadas com tecnologias da informação, servindo igualmente o processo de observação da ação, quer dos meus colegas, quer do tutor, como método de análise de práticas específicas, possibilitando-me assim planear estratégias de ação com base nas ações desenvolvidas por outros.

Desde o início do processo de estágio que foram realizadas reuniões de estágio de três em três semanas, momento em que o processo de reflexão sobre as ações desenvolvidas, opções tomadas ou a tomar etc., era realizado em conjunto, participando nele o estagiário e o tutor. Estas reuniões, e mais que tudo processos de análise e reflexão, serviram como uma importante ferramenta para a evolução na realização das tarefas seguintes.

Sinto-me atualmente capaz de descrever quer na teoria quer na prática o papel de um programador numa empresa. Esta experiência adquirida não me influenciou apenas a nível profissional, mas também a nível pessoal. Considero-me neste momento uma pessoa diferente, mais sensibilizada para questões relacionadas com a realidade e contexto empresarial. Modifiquei também a minha opinião e "maneira de ver" uma empresa de desenvolvimento de software e mais especificamente uma startup, considerando que a mesma mais do que uma mera empresa de que desenvolve o seu leque de produtos, pode assumir o papel de formar futuros profissionais, capazes de se integrarem mais facilmente num futuro próximo.

Bibliografia

Robehmed, Natalie (16 December 2013). "What Is A Startup?". Forbes. Retrieved 30 April 2016.

[Online]. Available: https://developer.android.com/studio/releases/?hl=pt-br [Acedido em 10 julho 2018].

[Online]. Available: http://www.itpro.co.uk/android/19905/best-android-apps-for-2018 [Acedido em 10 julho 2018].

[Online]. Available: https://spring.io/guides/gs/gradle [Acedido em 10 julho 2018].

[Online]. Available: https://pt.wikipedia.org/wiki/Android. [Acedido em 10 julho 2018].

[Online]. Available: https://developer.android.com/training/implementing-navigation/nav-drawer#DrawerLayout [Acedido em 10 julho 2018].

XML: XML Guia de Consulta Rápida, Autor: Otávio C. Décio, Editora Novatec
Patrick Naughton, Dominando o Java, Guia Autorizado da Sun Microsystems, Editora Makron Books, 1997, Osborne, ISBN 80025-75540

[Online]. Available: https://android-developers.googleblog.com/2013/05/android-studio-ide-built-for-android.html. [Acedido em 10 julho 2018].

[Online]. Available: https://www.caelum.com.br/apostila-java-orientacao-objetos/ [Acedido em 10 julho 2018].

[Online]. Available: https://netbeans.org/about/index.html [Acedido em 10 julho 2018].

[Online]. Available: http://hibernate.org/ [Acedido em 10 julho 2018].

[Online]. Available: https://mariadb.org/ [Acedido em 10 julho 2018].

[Online]. Available: https://netbeans.org/about/index.html [Acedido em 10 julho 2018].

[Online]. Available: https://git-scm.com/ [Acedido em 10 julho 2018].

[Online]. Available: https://visualstudio.microsoft.com/pt-br/team-services/ [Acedido em 10 julho 2018].

[Online]. Available: https://pt.wikipedia.org/wiki/Desenvolvimento_%C3%A1gil_de_software [Acedido em 10 julho 2018].

[Online]. Available: https://slack.com/ [Acedido em 10 julho 2018].

[Online]. Available: https://pt.wikipedia.org/wiki/Hibernate [Acedido em 10 julho 2018].

[Online]. Available: https://mirrors.edge.kernel.org/pub/software/scm/git/docs/user-manual.html#ensuring-reliability [Acedido em 10 julho 2018].

[Online]. Available: https://en.wikipedia.org/wiki/API_testing [Acedido em 10 julho 2018].

[Online]. Available: https://thecuriousdev.org/postman-test-your-rest-api/ [Acedido em 10 julho 2018].

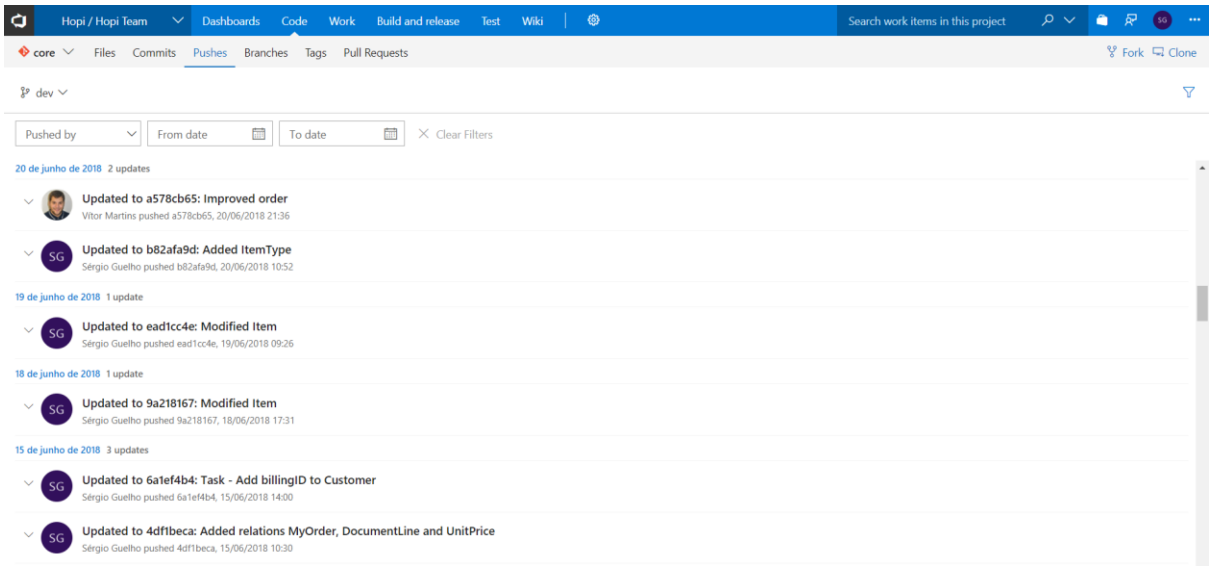
[Online]. Available: https://www.codecademy.com/articles/what-is-rest [Acedido em 10 julho 2018].

[Online]. Available: https://jenkins.io/ [Acedido em 10 julho 2018].

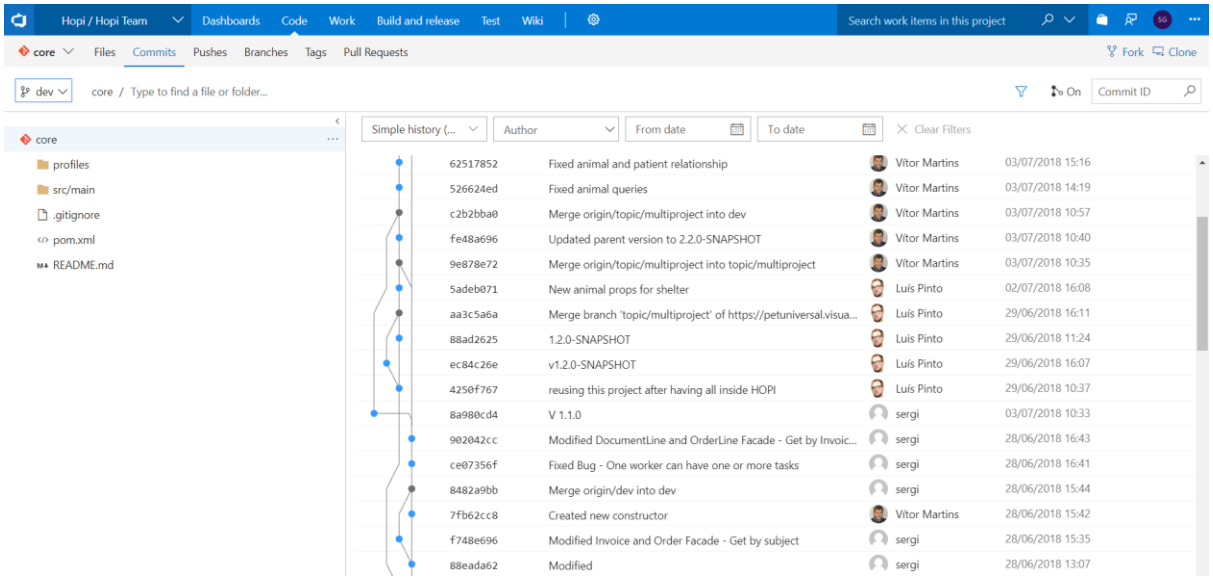
Pautasso, Cesare; Zimmermann, Olaf; Leymann, Frank (April 2008), "RESTful Web Services vs. Big Web Services: Making the Right Architectural Decision", 17th International World Wide Web Conference (WWW2008), Beijing, China

Anexos

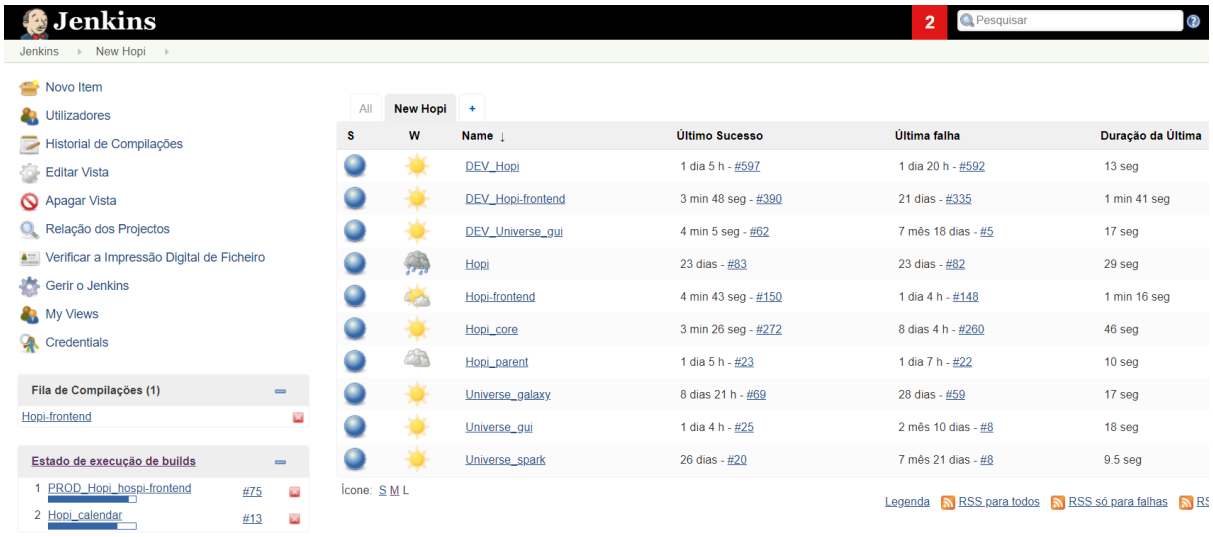
Anexo 1 – Visual Studio Team, lista de pushes



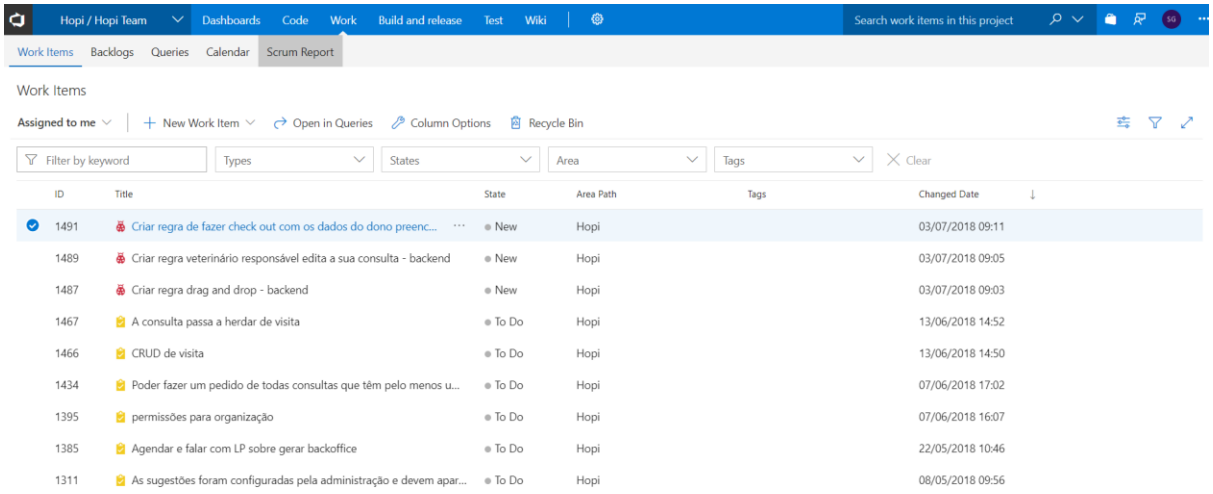
Anexo 2 – Visual Studio Team, lista de commits



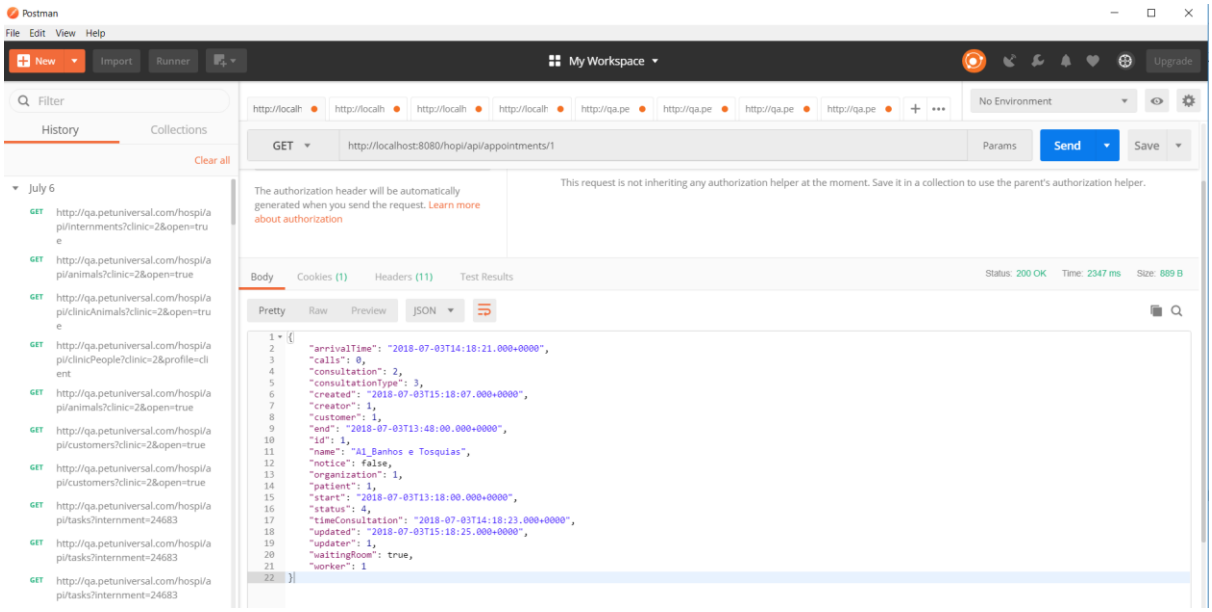
Anexo 3 – Lista de builds do Jenkins



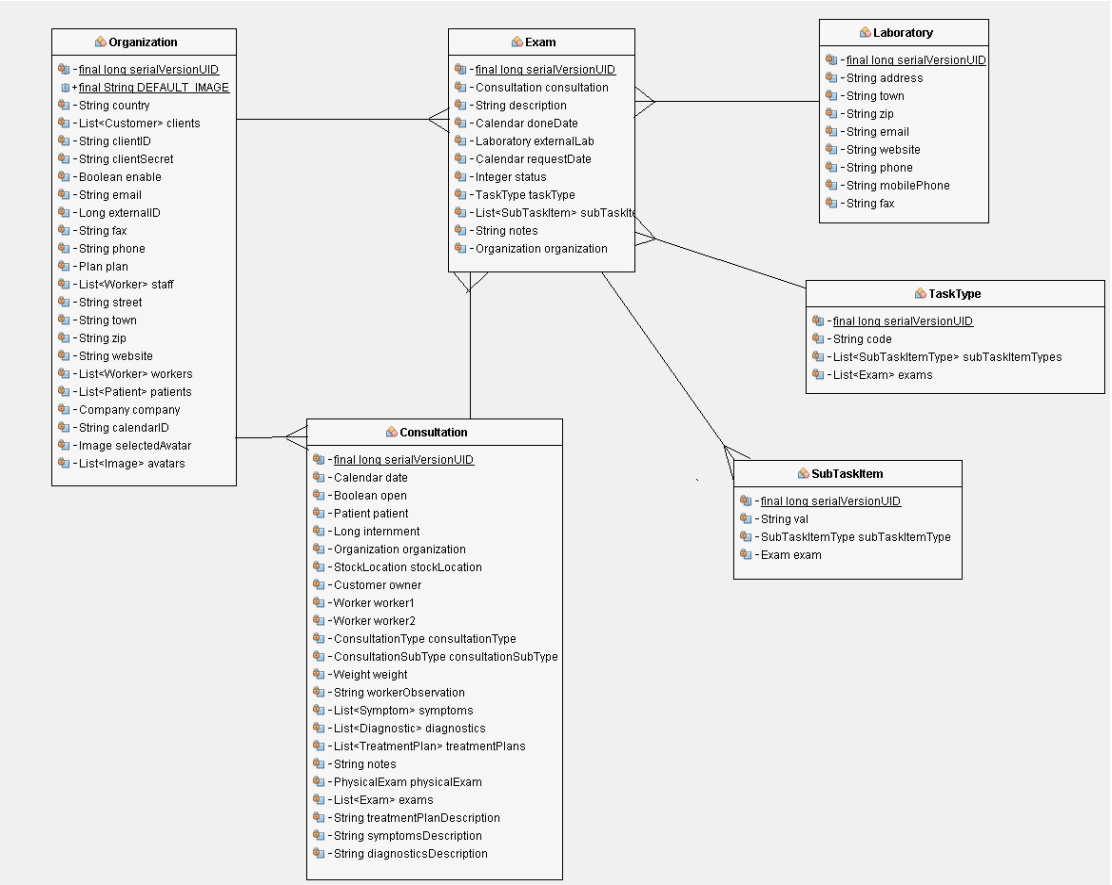
Anexo 4 – Visual Studio Team, work items



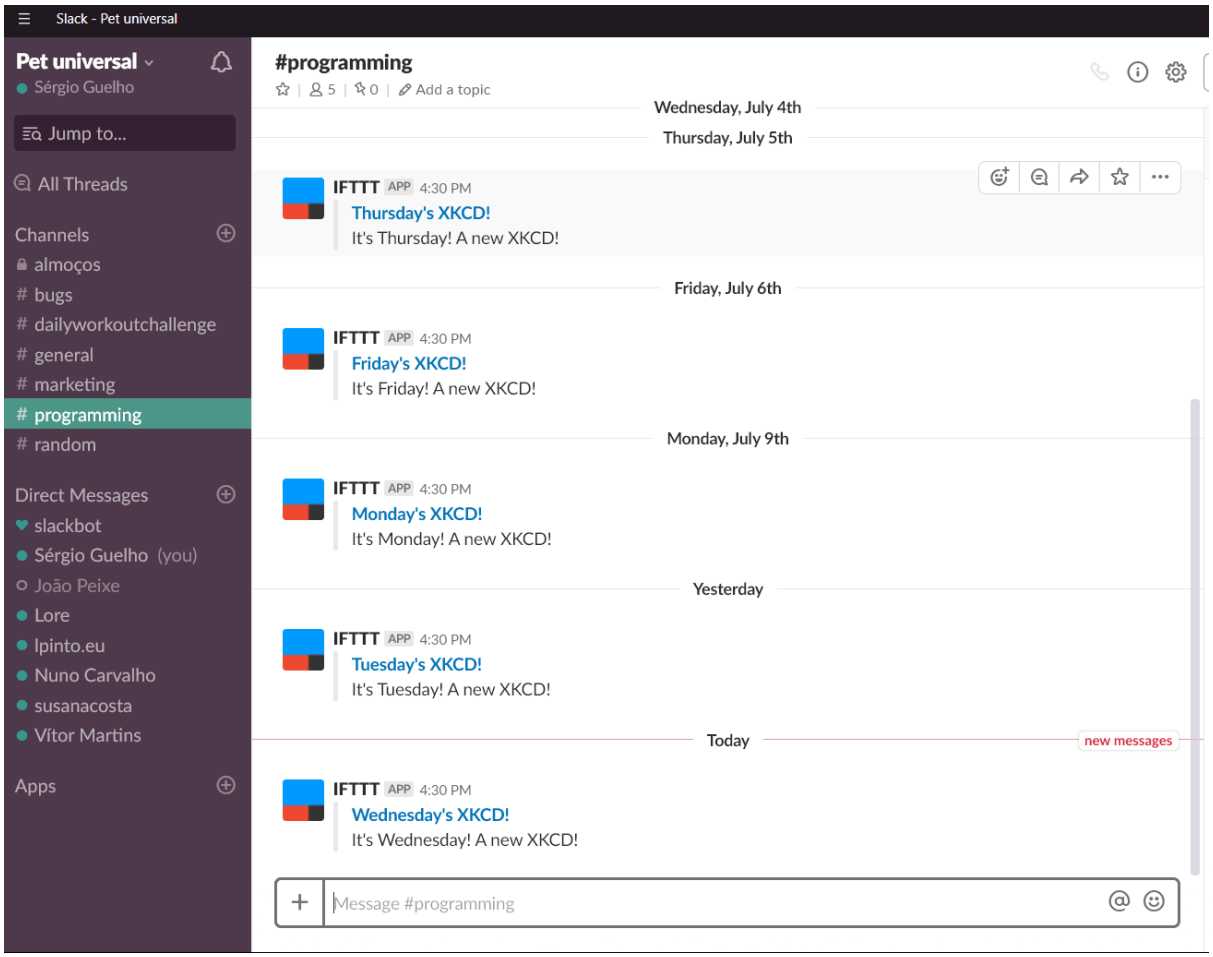
Anexo 5 – Postman, exemplo de um get no computador local



Anexo 6 – Exemplo de um diagrama de classes do Pet Universal – Solução modular



Anexo 7 – Canais do slack utilizados para comunicar



Anexo 8 – Visual Studio Team, lista de backlog's