Desenvolvimento de Software na Ad Infinitum Business

Márcio José Chaves Oliveira

(Licenciado)

Tese Submetida à Universidade da Madeira para a
Obtenção do Grau de Mestre em Engenharia Informática

Funchal – Portugal

Junho 2014

Orientador Interno:

Eduardo Leopoldo Fermé, PhD
Professor Associado com Agregação do Centro de Competências de Ciências Exatas e da Engenharia da Universidade da Madeira

Orientador Externo:

Hugo Alexandre Teixeira Marques, MsC
Ex. Diretor de Produção na empresa Ad Infinitum Business, S.A.

ABSTRACT

The main goal of this report is to present and describe some of the projects acomplished, methodologies used and the experience I have acquired during my internship in order to conclude the Masters Course of Engenharia Informática from University of Madeira.

The aim of this internship was to experience a corporate environment in order to prepare better for the job market, earning this way a valuable experience that is required by all companies, and allowing myself to better decide as what would be my next step after concluding my course.

In this report there are reference of some responsabilities that were assigned to me, like the leadership of a team and the presentation of soluctions that would solve some technical problems as well as the implementation of this solutions.

Also in this report I will characterize and criticize some decisions, procedures and metodologies used in the company that in certain way affected the software development from a general point of view as well as the success of the company.

This report can serve as example, not only for soluctions to solve similar technical problems as presented in the report, but also to demonstrate the result of some procedures and methodologies used in the company and in the devolpment of software.

KEYWORDS

Mobile applications

Web Services

Software Engineering

Software Development

Agile Methods

RESUMO

Este relatório tem como objetivo apresentar e descrever alguns projetos realizados, metodologias utilizadas e toda a experiência que adquiri durante o meu estágio curricular para finalizar o curso de Mestrado em Engenharia Informática da Universidade da Madeira.

O principal objetivo deste estágio foi experienciar um ambiente empresarial e ganhar alguma experiência profissional, algo que é bastante requisitado no mercado de trabalho nos dias de hoje.

Neste relatório estão referenciadas algumas das responsabilidades que me foram atribuídas, como a liderança de uma equipa e a apresentação de algumas soluções para certos problemas técnicos, bem como a implementação das mesmas.

Serão também caracterizadas e criticadas algumas decisões, procedimentos e metodologias utilizadas pela empresa que afetaram de certa forma o desenvolvimento de software de um ponto de vista geral e o próprio sucesso da empresa.

Este relatório poderá servir de exemplo, não só para soluções para um problema técnico semelhante aos que aqui apresentei, mas também demonstrar o resultado de certos procedimentos e metodologias utilizadas na empresa no desenvolvimento de software.

PALAVRAS-CHAVE

Aplicações Móveis

Serviços Web

Engenharia de Software

Desenvolvimento de Software

Métodos Ágeis

AGRADECIMENTOS

Esta secção foi reservada para agradecer a todas as pessoas que contribuíram significativamente para a realização do meu estágio bem como a concretização do seu relatório.

Queria começar por agradecer ao diretor de produção na empresa e também o meu orientador externo, o Hugo Marques, por me orientar na minha primeira experiência de trabalho, por tentar o impossível e mediar ao máximo toda a turbulência que atingia o departamento de produção, por defender os colaboradores do departamento, e por ser um grande exemplo.

Queria agradecer também a todos os meus amigos pelos concelhos e apoio.

A todos os colaboradores da empresa pois com eles aprendi muito, em especial aos elementos da minha equipa que apesar das circunstâncias e condições da empresa, sempre estiveram dispostos a colaborar comigo.

Aos meus pais que sempre me apoiaram incondicionalmente.

Queria agradecer também a todos os professores que me acompanharam e formaram durante todo o meu período académico na Universidade da Madeira.

Um agradecimento em especial ao meu orientador, o Professor Doutor Eduardo Fermé pela sua total disponibilidade e apoio, e claro, pela sua orientação, notáveis e fulcrais para a conclusão desta tão importante etapa.

ÍNDICE

LISTA DE FIGURAS

LISTA DE TABELAS

ACRÓNIMOS

3G –Third Generation

API – Application Programming Interface

CEO - Chief Executive Officer

CIO – Chief Information Officer

CMS – Content Management System

CRUD – Create, Read, Update and Delete

CSS - Cascading Style Sheets

GPS - Global Positioning System

HTML - HyperText Markup Language

HTTP - Hypertext Transfer Protocol

JSON - JavaScript Object Notation

MVC – Model View Controller

PHP - Hypertext Preprocessor

REST - Representational state transfer

SDK – Software Development Kit

SoC – Separation of Concerns

SSO – Single Sign On

URL - Uniform Resource Locator

URI – Uniform Resource Identifier

W3C – World Wide Web Consortium

XML – eXtensible Markup Language

I. INTRODUÇÃO

"Do not go where the path may lead, go instead where there is no path and leave a trail."

- Ralph Waldo Emerson

I.1. APRESENTAÇÃO

Um estágio curricular no Mestrado em Engenharia Informática numa empresa é uma boa oportunidade para pôr em prática todas as ferramentas aprendidas e todo o conhecimento adquirido ao longo do primeiro e segundo ciclo de Engenharia Informática num mundo empresarial, experienciando também este ambiente que é muito diferente do ambiente académico, razão pela qual foi optado por finalizar este segundo ciclo de Engenharia Informática com um estágio curricular.

Este relatório pretende assim descrever ao máximo toda a experiência adquirida e vivida ao longo de 7 meses de estágio na empresa Ad Infinitum Business. Uma empresa jovem e com métodos de funcionamento e desenvolvimento atípicos a outras empresas da sua área de desenvolvimento para a web.

Durante estes 7 meses tive a oportunidade de pesquisar soluções técnicas, planear, desenvolver e até liderar projetos de aplicações informáticas num ambiente empresarial.

I.2. CONTEXTO DO ESTÁGIO

Após o fim do primeiro ano do 2º ciclo de Engenharia Informática, e com todas as unidades curriculares completadas para tal, segundo o Despacho n.º 14100/2010, publicado no Diário da República, 2.ª série – N.º 175, de 8 de Setembro de 2010, foi necessário optar entre uma dissertação, trabalho de projeto, ou estágio.

A minha decisão recaiu então sobre o estágio curricular. Foram contactadas algumas empresas a nível regional e internacional para a realização deste estágio, no entanto apenas a empresa Ad Infinitum Business acordou com a realização deste estágio curricular.

O primeiro contacto que tive com a empresa Ad Infinitum Business foi com o CEO e o Diretor de Produção. Após responder a um teste maioritariamente sobre PHP (do qual achei trivial), ficou combinado por e-mail com o Diretor de Produção uma reunião onde estariam presentes o Diretor de Produção, o CEO, e outro colega também finalista do curso de mestrado em engenharia informática e também interessado em realizar o estágio curricular de mestrado nesta empresa.

Foi-nos dado uma introdução sobre o que era a empresa, e quais os principais projetos que estavam a ser desenvolvidos. Dentro destes projetos foram-nos apresentados 3 com necessidade de serem desenvolvidos, nomeadamente um gestor de publicidade, uma aplicação móvel e o desenvolvimento de componentes Joomla e deram-nos a liberdade de escolher um destes para desenvolvermos durante o período de estágio. A minha decisão recaiu sobre o desenvolvimento da aplicação móvel visto ser uma área em grande crescimento [1] e também uma área que me desperta grande interesse.

Foi-nos dito também que à medida que entrassem novos programadores na empresa, e houvesse necessidade, alguns destes poderiam ser integrados nos nossos projetos formando assim equipas, nas quais nós seriamos *team-leaders*.

Seria assinado um acordo de intenções com um prazo de 6 meses entre cada um de nós e o CEO da empresa que discriminava que ao fim de 6 meses seriam auferidos 9,40 euros por hora de trabalho em ações na empresa. Dadas estas condições, a empresa não obrigava a nossa presença nos escritórios (à exceção de reuniões com *stakeholders* do projeto que iríamos desenvolver). O objetivo era que ao fim dos 6 meses recebêssemos as ações correspondentes às horas de trabalho e passássemos a membros efetivos, trabalhando a tempo inteiro recebendo um salário de 1500 euros mensais líquidos. O objetivo desta decisão por parte da empresa seria minimizar os custos iniciais, avaliando os colaboradores recrutados durante 6 meses e admitindo-os a membros efetivos caso possuíssem perfil para tal. Todos os colaboradores da empresa no departamento de produção estavam ao abrigo destas condições, para motivar a presença destes e ajudar a redução

de custos aos colaboradores a empresa iria oferecer-lhes o almoço no restaurante Espaço Funchal para aqueles que trabalhassem mais de 5 horas nos escritórios.

Apesar dos conselhos que tive dos meus colegas, professores e familiares de não realizar o estágio devido às condições acima referidas, decidi continuar com a mesma decisão, pois apesar do funcionamento atípico da empresa, este era o único estágio que tinha disponível para efetuar e queria mesmo realizar um estágio, o meu objetivo seria diversificar a minha experiência profissional além do ambiente académico passando para um ambiente empresarial, e, neste caso em particular, poderia chegar até a liderar uma equipa, uma experiência que não teria acesso em qualquer uma das outras opções que tinha disponível.

O objetivo inicial do decorrer do estágio seria o desenvolvimento de uma aplicação móvel, no entanto foram-me atribuídas responsabilidades adicionais, destacando-se o desenvolvimento de um sistema *single sign on* (SSO), um sistema de autenticação de dois fatores (TFA) e um chat Web. Alguns destes projetos foram desenvolvidos em equipa, onde fiquei responsável pela coordenação destas.

I.3. CONTEXTO INSTITUCIONAL

I.3.1.Descrição da empresa

A empresa Ad Infinitum Business admite as seguintes funções e atividades no mercado [2]:

- Criação, gestão, manutenção e exploração de portais web, publicitários, software de gestão e exportação web;
- Atividades de processamento de dados, domiciliação de informação e atividades relacionadas;
- Publicidade, estudos de mercado e sondagens de opinião;
- Atividades de consultoria e programação informática;
- Gestão e exploração de equipamentos informáticos;
- Exercício de quaisquer atividades que sejam complementares, subsidiárias ou acessórias das referidas anteriormente, bem como de comercialização de bens ou de prestação de serviços por conta própria ou de terceiros, designadamente na sociedade de informação, redes e serviços de comunicações eletrónicas, incluindo recursos e serviços conexos.

Figura 1: Logo da empresa

I.3.2.Localização

A empresa está sediada na rua da Carreira, N. 142 9000-042 Funchal dispondo também de outros escritórios. As fotos da localização da empresa e as suas instalações encontram-se em anexo.

I.3.3.Estrutura interna e Organigrama

A estrutura interna da Ad Infinitum Business compreende as seguintes direções:

- Direção Administrativa e Financeira;
- Direção de Produção;
- Direção de Expansão;
- Direção de *Marketing*;
- Direção de Desenvolvimento de Produtos;

Cabe à Direção Administrativa e Financeira a responsabilidade de planeamento e estratégia da *ad8biz*. A Direção de Desenvolvimento de Produtos trabalha na definição e especificação do conceito e, em paralelo com a Direção de Produção, desenvolvem um produto ou sistema. A Direção de Expansão, conjunta com a Direção de *Marketing* têm como objetivos a aquisição de contactos, a relação com o cliente e estratégia de expansão da empresa.

Cada Direção suporta determinados departamentos e são lideradas por um diretor, cada departamento é ainda dividido em pequenas equipas lideradas pelos seus *team-leaders*. Toda a comunicação entre equipas e departamentos é da responsabilidade dos respetivos líderes, evitando assim que sejam adotadas medidas sem o conhecimento de um responsável.

No início do estágio, incluindo o Diretor de Produção, a área de produção da empresa possuía 5 elementos, no entanto houve um grande crescimento em relação ao o número de colaboradores na empresa durante os meus 7 meses de estágio. Na data em que decidi sair da empresa a empresa admitia possuir 150 colaboradores presentes na Madeira e Lisboa e cerca de 35 colaboradores no departamento de produção, todos estes presentes na Madeira. No entanto era complicado verificar estes números já que ninguém era obrigado a estar presente diariamente. Apesar destas condições, era possível contabilizar uma presença média diária de 15 colaboradores no departamento de produção.

Visto que os colaboradores do departamento de produção não eram remunerados e que cerca de 70% eram ainda estudantes, a presença destes nunca foi obrigatória nos escritórios. No entanto cada colaborador era responsável por contabilizar as suas horas e comunicá-las ao Diretor de Produção.

A estrutura geral da empresa pode ser visualizada a partir do organigrama representado na Figura 2. Esta estrutura a uma nível mais detalhado pode ser encontrada no Anexo B.

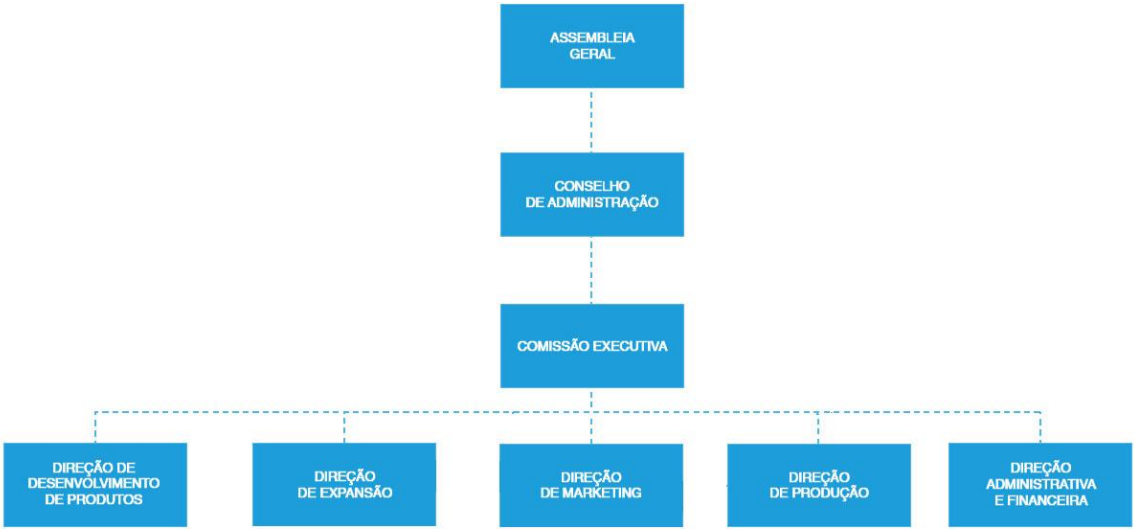


Figura 2- Organigrama da estrutura da Ad Infinitum Business

I.4. MÉTODOS DE TRABALHO E FERRAMENTAS DE APOIO

I.4.1.Reuniões de Controlo

Inicialmente estas reuniões eram entre o CEO, o Diretor de Produção e todos os elementos de do departamento de produção. Estas reuniões eram organizadas ocasionalmente para comunicar decisões, consultar os elementos de produção para a tomada de decisões e agendar tarefas. O objetivo era que todos pudessem intervir e dar a sua opinião em todos os projetos a serem desenvolvidos no departamento, aumentando assim a massa crítica reunida para a tomada de decisões. Com o aumento de colaboradores na área de produção e com a introdução do ciclo de vida baseado no *Scrum*, estas reuniões passaram a ser diárias, curtas e compostas apenas pelo CEO, o diretor de produção e os *team-leaders*.

Os elementos que compunham estas reuniões eram os seguintes:

- CEO.
- Director de produção (CIO).
- t*eam-leader* no Weshha.
- *team-leader* no gestor de publicidade.
- *team-leader* na aplicação 8xbiz mobile e *webservices*. (eu)
- Administrador de sistemas.
- *team-leader* da equipa de web design.
- *team-leader* da equipa Biz Concept e S.E.O.

I.4.2.Stakeholders

De todos os projetos que desenvolvi os *stakeholders* principais eram o CEO e o diretor de produção. No entanto para a aplicação móvel destacava-se o dono de um restaurante como *stackeholder* e que representava o dono de um estabelecimento classificado no 8xbiz, sendo este estabelecimento visualizável também na aplicação móvel.

I.4.3.Métodos Ágeis

Os métodos ágeis são normalmente caracterizadas pelo oposto aos métodos tradicionais que são orientados ao planeamento. Cada um destes possui um conjunto de características que os definem e distinguem [3]:

Principais objetivos – Os principais objetivos dos **métodos ágeis** são a rápida criação de valor num projeto a capacidade de resposta à mudança. Projetos que usam os métodos ágeis normalmente não efetuam análises de retorno de investimento para determinar a ótima distribuição dos recursos, em vez disso o software é desenvolvido o mais rápido possível e através da experiência é decidido que funcionalidades serão adicionadas, isto permite a

prevenção da perda de investimento usado em falsas assunções. A capacidade de resposta rápida é uma postura reativa ao invés de proactiva, num mundo caracterizado por rápidas mudanças no mercado de aplicações, tecnologia e ambiente de desenvolvimento tal postura trará vantagens significativas em vez de uma postura fechada num plano obsoleto. Os principais objetivos dos **métodos orientados ao planeamento** são previsibilidade, estabilidade e confiança.

Dimensão – Os métodos ágeis costumam ser mais apropriados para pequenas e médias equipas de pessoas (não excedendo as 10) trabalhando em aplicações relativamente pequenas enquanto os métodos orientados ao planeamento enquadram-se melhor em projetos de maior dimensão.

Ambiente – Os métodos ágeis são mais aplicáveis em ambientes turbulentos e em grande mudança. É adotada uma visão de que organizações são sistemas complexos e em constante mudança nas quais os requisitos são emergentes ao invés de pré-especificados, no entanto a mudança de requisitos num estágio final de desenvolvimento pode ser mal aplicada e com resultados desastrosos. Os métodos orientados ao planeamento funcionam melhor quando grande parte dos requisitos são determinados antecipadamente. Não são aceitáveis mudanças superiores a 1% por mês.

Relações com os clientes – Os métodos ágeis dependem de clientes (ou representante(s) destes) presentes e dedicados (conhecidos como CRACK *costumers*[1]) de maneira a manter o projeto focado na adição constante de valor à organização, enquanto que os métodos orientados ao planeamento dependem de contractos e especificações com os clientes e estes não necessitam estar presentes.

Planeamento e controlo – Para os métodos ágeis o planeamento é visto como um meio para atingir um fim enquanto que os métodos orientados ao planeamento usam-no para comunicar e coordenar. Os métodos orientados ao planeamento apoiam-se muito na documentação e planeamento de processos (horários, *milestones*, procedimentos) e planeamento de produtos (requisitos, arquitetura, *standards*) de forma a manter todos os intervenientes coordenados.

Comunicação no projeto – Os métodos ágeis apoiam-se muito do conhecimento tácito (conhecimento baseado na partilha de experiencias) e na comunicação muito frequente de pessoa para pessoa, é pouca a comunicação unidirecional dando preferência à colaboração e são usadas técnicas como *standup meetings* e *pair programming,* estas técnicas promovem também o conhecimento tácito. Os métodos orientados ao planeamento apoiam-se na documentação explícita do conhecimento e a comunicação tende a ser unidirecional. Os métodos ágeis usufruem

[1] CRACK (Collaborative, Representative, Authorized, Committed, Knowledgeable) é um acrónimo que significa colaborativo, representativo, autorizado, dedicado e bem informado. Um CRACK *costumer* é um cliente que possui todos estes atributos [3].

da documentação apenas quando necessária e enfatizando a prática, enquanto que os métodos orientados ao planeamento subtraem da documentação algo que já não é necessário.

Requisitos – Os métodos ágeis tomam os requisitos como uma ajustável história de utilizador (*user story*). Os métodos ágeis contam com a rápida iteração de ciclos para determinar mudanças necessárias e resolvê-las na próxima iteração. Os métodos orientados ao planeamento preferem requisitos formais, completos, consistentes, rasteáveis e testáveis.

Desenvolvimento – A diferença principal entre os métodos ágeis e os métodos orientados ao planeamento está nas práticas em relação ao desenho e arquitetura do *software*. Enquanto que os métodos orientados ao planeamento defendem um planeamento e um desenho baseado em arquiteturas, os métodos ágeis dão preferência ao desenho mais simples (se um desenho tem capacidades para além dos requisitos atuais ou antecipa novas funcionalidades, estas deverão ser removidas).

Testing – Os métodos ágeis efetuam testes a cada iteração, e usufruem de *pair programming* ou outras técnicas de revisão para remover defeitos no código à medida que este está sendo gerado. Estes também desenvolvem testes executáveis para permitir o teste prematuro e contínuo. Os métodos orientados ao planeamento desenvolvem verificações de consistência dos requisitos e das especificações arquiteturais num estágio prematuro do desenvolvimento de maneira a evitar resoluções num estágio tardio, o que as tornam muito mais caras.

Desenvolvedores – Os métodos ágeis requerem desenvolvedores com outras competências além das competências técnicas como cordialidade, talento, perícia, comunicação e a mistura de habilidades. Os métodos orientados ao planeamento apostam em qualidades técnicas e específicas e não dependem tanto do talento pois ao planear um projeto e a sua arquitetura conseguem com que desenvolvedores menos capazes consigam contribuir com menos risco.

Cultura – Utilizando os métodos ágeis, as pessoas devem sentir-se confortáveis quando há um nível de liberdade significativo que as permita definir e trabalhar certos problemas, é esperado que qualquer pessoa possa efetuar qualquer trabalho que seja necessário para o sucesso da empresa. Numa cultura onde são praticados os métodos orientados ao planeamento, as pessoas sentem-se confortáveis quando há políticas e procedimentos claros que definem o seu papel na organização, o objetivo é que estas pessoas realizem as tarefas especificadas de maneira a que os seus produtos facilmente se integrem com outros com um conhecimento limitado sobre o que os outros estão a fazer. Uma vez estabelecida uma cultura, é difícil e demorado alterá-la.

Dito isto, é possível diferenciar ou escolher qual a melhor metodologia tendo em conta os seguintes fatores representados na Figura 3.

Análise dos métodos utilizados na empresa

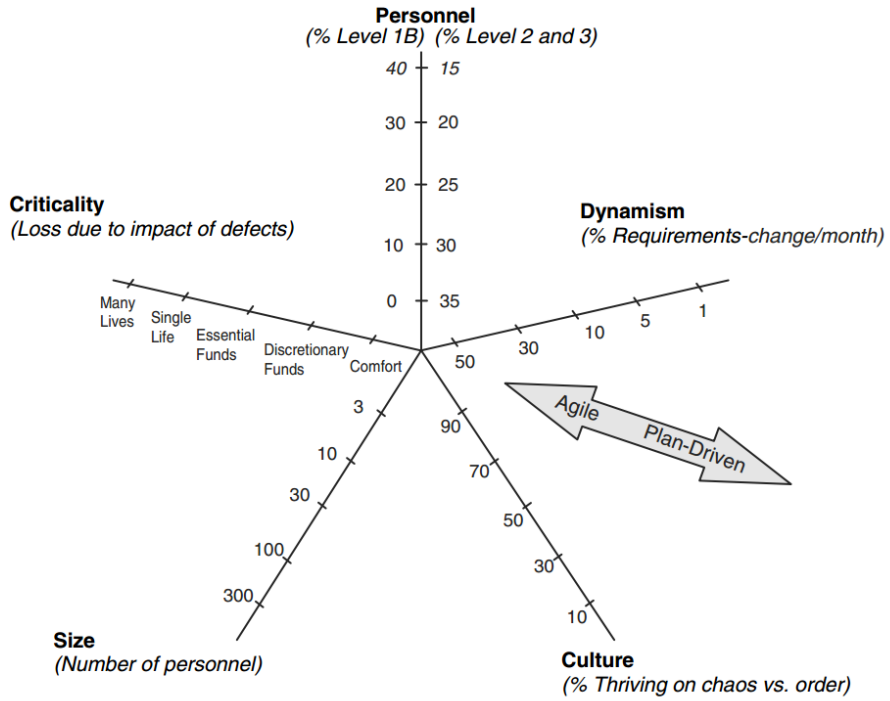


Figura 3 - Dimensões que afectam a selecção de um método de desenvolvimento [3]

Partindo das dimensões apresentadas na Figura 3, irei passar a classificar cada uma destas em relação aos métodos utilizados na Ad Infinitum Business:

Pessoal – A empresa adotava um método de recrutamento em que todos aqueles que mostrassem alguma competência na área de desenvolvimento de software eram admitidos. Pude acompanhar o processo de recrutamento e verificar que não houve qualquer candidatura recusada, e era frequentemente incentivado (por parte do CEO) aos membros existentes que trouxessem membros novos (mesmo não sendo necessário). Este fator estaria mais associado aos métodos ágeis, já que os métodos ágeis defendem uma grande mistura de competências, e que aqueles com mais competência para uma determinada tarefa possam colaborar e treinar aqueles com menos competência (usando por exemplo *pair-programming*). No entanto eu não incluo esta abordagem em nenhum dos métodos já que este método de recrutamento adotado pela empresa é de tal forma arriscado e imprudente que não encaixa em qualquer perfil para um bom método de desenvolvimento de software. Segundo a Figura 3 classificaria o Pessoal como 40% nível 1B e 15% nível 2 e 3[2].

Criticidade – A criticidade era máxima. Era dada pouca ou nenhuma importância à documentação e não eram efetuados testes suficientes à medida que o software era desenvolvido. Normalmente falhas no sistema só eram descobertas depois do *deploy* no servidor. Esta é uma

[2] O nível 1B, 2 e 3 dizem respeito à classificação do pessoal de uma empresa segundo Alistair Cockburn e que vai deste o nível -1 (pouca competência) ao nível 3 (máxima competência).

característica mais associada aos métodos ágeis. Segundo a Figura 3 classificaria a criticidade como muitas vidas.

Dimensão – A empresa possuía um grande número de colaboradores (chegando até aos 150) e inúmeros projetos a serem desenvolvidos, todos estes num estágio inicial. Cada um destes projetos poderia ser desenvolvido individualmente ou em equipa, equipas estas que não teriam mais que 10 elementos. À partida, esta abordagem poderia se incluir nos métodos ágeis que defendem o desenvolvimento de aplicações por pequenas equipas, no entanto as aplicações a ser desenvolvidas não eram independentes, todas elas integravam-se entre si pelo que podemos observar todas estas aplicações como uma só aplicação de grande dimensão. Segundo a Figura 3 classificaria a dimensão como 150.

Cultura – Havia demasiada liberdade na empresa, o que gerava um caos quase impossível de prosperar, verificava por exemplo gestores de produto a dar ordens diretamente a programadores de uma equipa de produção, sem que estas passassem primeiro pelo diretor de produção ou até mesmo pelo *team-leader*. Segundo a Figura 3 classificaria a cultura com 90% de necessidade de prosperar no caos.

Dinâmica – A documentação e o desenho do software era incentivado ao mínimo possível e era adotado um *refactoring* contínuo, isto porque havia muitas falhas para resolver, código difícil de interpretar e era muito frequente a mudança de requisitos. Segundo a Figura 3 classificaria a dinâmica como 50% de mudança de requisitos por mês.

Representando as classificações das 5 dimensões no gráfico presente na Figura 4 é possível confirmar que os métodos praticados na Ad Infinitum Business para o desenvolvimento de software estariam mais associados aos métodos ágeis que aos métodos tradicionais orientados ao planeamento.

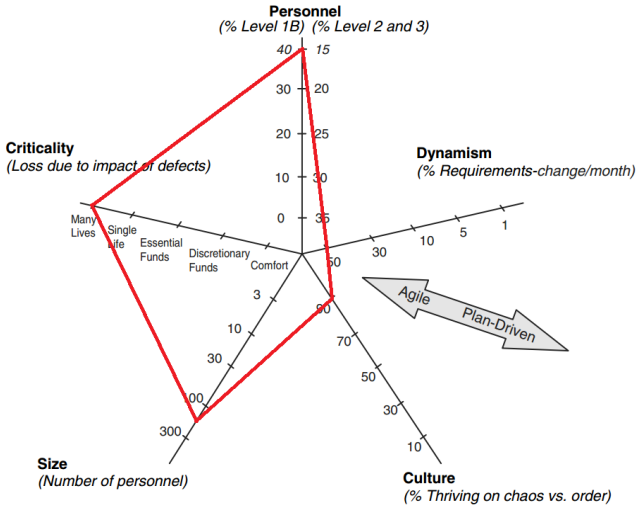


Figura 4 - Classificação das 5 dimensões em relação aos métodos praticados na Ad Infinitum Business

I.4.4.Ciclos de vida e técnicas de desenvolvimento de software

O ciclo de vida do desenvolvimento de software é uma estrutura imposta no desenvolvimento de um produto de software. Existe um conjunto de modelos adotados para vários processos de desenvolvimento de software, cada um deles descrevendo abordagens para uma variedade de tarefas ou atividades que ocorrem durante cada um destes processos. Irei de seguida referir alguns destes modelos adotados na Ad Infinitum Business [4].

Big Bang Model

O Modelo *Big Bang* é um modelo em que é amontoado um grande número de pessoas ou capital, é gasta muita energia e disto resulta o produto perfeito, ou não resulta nada. Existe muito pouco planeamento, pouco agendamento e pouca formalidade no processo. Todos os esforços são gastos desenvolvendo o software e escrevendo código. Este processo é ideal quando os requisitos são pouco explícitos ou mal percebidos e a data de lançamento é flexível [4]. O modelo Big Bang é no entanto um modelo arriscado já que a probabilidade de falha é muito alta [5].

Este modelo foi usado inicialmente pela Ad Infinitum Business como processo de desenvolvimento de software a nível geral.

Code and Fix

No modelo *Code and Fix* também existe muito pouco planeamento inicial, é posta uma certa pressão através de agendamentos curtos de entrega e os programadores começam imediatamente a desenvolver código e a resolver os problemas à medida que estes ocorrem até o projeto estar completo. Uma desvantagem de usar este modelo é que caso hajam problemas arquiteturais num estágio final do projeto é necessária a reescrita de partes significativas da aplicação, o *code and fix* é apropriado apenas para pequenos projetos e sem a intenção que estes sirvam de base para um desenvolvimento futuro [6].

Passados 2 meses de estágio a empresa começou a verificar que o modelo adoptado inicialmente não era o mais indicado, pois não conseguia verificar resultados significativos e por isso começou a ser adoptado o modelo Code and Fix. Começou a ser imposta uma certa pressão e agendamento nos programadores (através de reuniões de controlo de produção ocasionais) para que estes apresentassem resultados executáveis dentro de certas *deadlines*. Os programadores eram incentivados então a desenvolver o código imediatamente e resolviam os problemas à medida que estes apareciam. A versão executável era testada ao entregar ou antes de entregar.

Scrum

O *Scrum* é um processo de desenvolvimento de software que facilita o foco de uma equipa e que segue a mesma filosofia dos métodos ágeis. É uma metodologia de ciclo de vida para pequenas

equipas desenvolverem software incrementalmente em ambientes complexos. O *Scrum* é mais apropriado para projetos onde os requisitos não podem ser definidos e condições caóticas são antecipadas. O *Scrum* divide um projeto em sprints (iterações) de 30 dias em que as funcionalidades a ser implementadas são definidas antes do sprint começar [4].

Passados 3 meses de estágio, com a entrada de novos colaboradores na empresa, com o aumento da complexidade dos projetos, o aumento constante de requisitos e a necessidade de integração entre aplicações de diferentes projetos o modelo *Code and Fix* deixou de ser viável e foi necessário adotar um ciclo de vida que conseguisse prosperar no caos, ficou decidido então em reunião de controlo que seria adotado um modelo semelhante ao *Scrum*. Este ciclo de vida pode ser descrito com o auxílio da Figura 25 (É possível visualizar com melhor detalhe esta mesma figura no Anexo C.

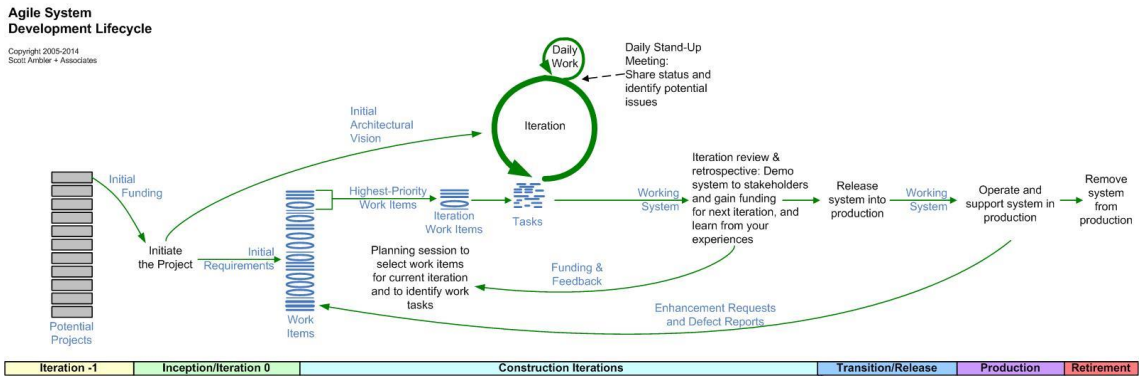


Figura 5 - Ciclo de vida ágil [7]

Os projetos eram normalmente iniciados pelo CEO ou pelo diretor de produção e atribuídos a um *team-leader* durante uma reunião de controlo. Durante a mesma reunião de controlo, seria também efetuado algum *brainstorm* da arquitetura inicial pelos *team-leaders* e eram levantados alguns requisitos que eram imediatamente traduzidos para elementos de trabalho (*work items*).

Eram efetuadas iterações semanais (e não mensais como recomenda o *Scrum*) e *stand-up meetings* diárias. No inicio/fim de cada iteração (segunda-feira) eram levantados novos elementos de trabalho prioritários pelos *team-leaders* e agendadas tarefas para a realização destes elementos de trabalho.

Um elemento de trabalho poderia ser um estudo técnico, a adição de uma funcionalidade, o *refactoring* de uma parte de uma aplicação ou a resolução de uma falha (*bug*) numa aplicação. Cada *team-leader* responsável por uma equipa de programadores e seria responsável pela execução dos elementos de trabalho que lhe eram atribuídos.

As *stand-up meetings* diárias eram realizadas para partilhar o estado da implementação dos elementos de trabalho, identificar falhas e se necessário atribuir novas tarefas.

Os projetos eram trabalhados e testados num ambiente local (o computador pessoal), todas as quintas-feiras era feito o *deploy* de versões executáveis num servidor de testes. As sextas-feiras eram dedicadas para testar as aplicações e a integração com outras aplicações, o resultado dos testes era apresentado no início da próxima iteração e a partir destes resultados eram produzidos novos elementos de trabalho. As versões executáveis no servidor de testes eram por vezes usadas como demonstração e apresentadas aos *stakeholders*, gestores de produto e *marketeers* associados ao projeto ou projetos correspondentes.

Uma vez aprovadas certas versões executáveis no servidor de testes, era feito o *deploy* das mesmas no servidor de produção e acessíveis ao publico em geral.

Este foi o ciclo de vida de desenvolvimento de software predominante na Ad Infinitum Business.

Espiral

O modelo em espiral possui quatro fases: planeamento, análise de riscos, desenvolvimento e avaliação. O processo do desenvolvimento de software utilizando este modelo passa repetidamente por estas quatro fazes através de iterações. Este modelo foca-se no adereçamento de riscos e tenta minimizar os riscos de um projeto dividindo-o em pequenos segmentos, oferecendo facilidade de mudança durante o processo de desenvolvimento e a oportunidade de avaliar riscos durante o seu ciclo de vida [6].

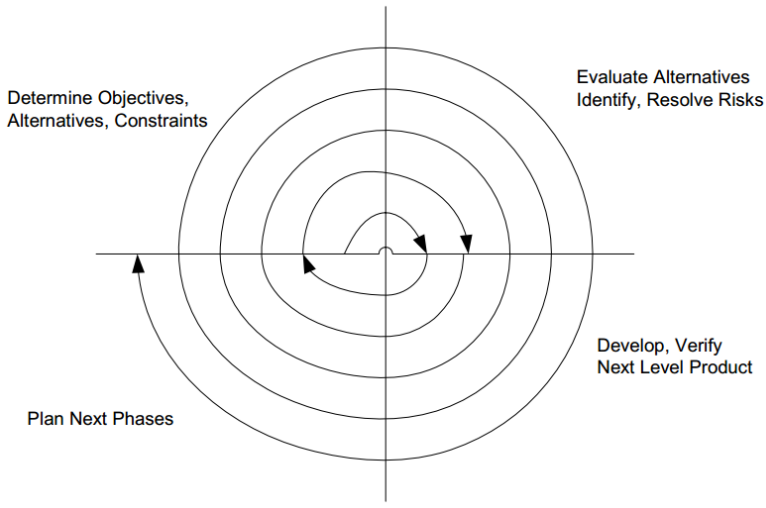


Figura 6 - Representação do modelo em espiral

O modelo em espiral, foi o modelo que utilizei para o desenvolvimento da aplicação 8xbiz mobile em específico. Foi escolhido por ser iterativo, incremental, os requisitos iniciais da aplicação não serem bem definidos e entendidos e por ser compatível com os métodos ágeis praticados pela empresa. Senti a necessidade de utilizar este método individualmente, já que no início do meu estágio não havia nenhum método específico adotado pela empresa.

Prototipagem

A prototipagem não é um ciclo de vida independente, mas sim uma abordagem para manipular porções de uma certa metodologia de desenvolvimento (ex. modelo incremental, modelo em espiral ou o *rapid application development*). A prototipagem tenta reduzir o risco do projeto dividindo o projeto em pequenos segmentos e oferecendo uma maior facilidade de mudança durante o processo de desenvolvimento. O utilizador é envolvido durante este processo, o que aumenta a probabilidade de aprovação da implementação final. Mockups do sistema são desenvolvidos sofrendo uma modificação iterativa até o protótipo evoluir o suficiente para reunir os requisitos estabelecidos [4].

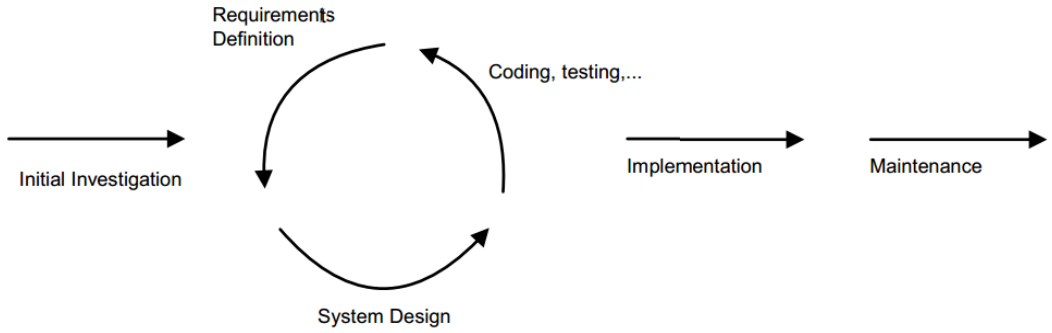


Figura 7 - Representação da protótipagem

A prototipagem foi usada em conjunto com o modelo em espiral para desenvolver a aplicação 8xbiz mobile e foram usados dois tipos de prototipagem:

Throw-away prototyping – uma pequena parte do sistema é desenvolvida e oferecida ao utilizador final para experimentar e avaliar. O utilizador fornece algum feedback que será incorporado no desenvolvimento do sistema principal e o protótipo é então descartado. O objetivo deste tipo de protótipo é assegurar que os requisitos são validados e bem entendidos [8].

Evolutionary prototyping – é apresentado um protótipo inicial ao utilizador, e à medida que o utilizador fornece feedback acerca deste são efetuadas melhorias. Este processo é repetido e o protótipo evolui até atingir o produto final [9].

Pair Programming

O *pair programming* não é um ciclo de vida só por si, mas uma técnica de desenvolvimento de software que pode ser aplicada a qualquer ciclo de vida de desenvolvimento de software, normalmente esta técnica é aplicada em métodos ágeis (ex. *Extreme Programming*).

Esta técnica consiste em que dois desenvolvedores trabalhem lado a lado no mesmo computador, colaborando no mesmo desenho, planeamento, algoritmo, código ou teste, superando assim um desenvolvedor que trabalha individualmente. Um dos desenvolvedores tem o controlo do

rato/teclado e ativamente implementa o programa. O outro observa continuamente e acompanha o trabalho do primeiro desenvolvedor de maneira a identificar defeitos no código ao mesmo tempo que pensa estrategicamente sobre a direção do trabalho. A qualquer momento ambos os desenvolvedores podem parar para debater algum problema que apareça. Ao fim de um curto período de tempo (cerca de 15 minutos) os desenvolvedores trocam os papeis, desta maneira ambos trabalham como iguais no desenvolvimento de software [10].

A prática desta técnica não é algo novo, em 1995 Larry Constantine relata a sua observação desta técnica como produzindo código mais rápido e com menos falhas como nunca antes visto [11].

O *pair programming* oferece também mais satisfação e confiança aos desenvolvedores, torna-os mais bem equipados para resolver problemas complexos e oferece um modo de dupla aprendizagem, permitindo assim ambos os desenvolvedores aprender um com o outro [10].

Decidi usar esta técnica *pair programming* juntamente com o João (nome fictício) no processo de desenvolvimento do segundo projeto descrito neste relatório, nomeadamente o serviço de SSO. A razão porque escolhi esta técnica foi devido à complexidade de desenho e implementação do sistema e principalmente porque o serviço que iriamos desenvolver deveria conter o mínimo de falhas possível.

I.4.5.Ferramentas de auxílio na gestão de projetos

A mudança de requisitos e de tecnologias são forças que direcionam a evolução do software, isto requer por vezes uma adaptação ou o redesenho de um sistema de software e a sua arquitetura, torna-se então indispensável o uso de ferramentas que acompanham a sua mudança. Dependendo da dimensão de um projeto podem ser usadas ferramentas de controlo de versões e ferramentas de relatório e acompanhamento de erros [12].

Quadros Brancos e Post-it

Equipas de desenvolvimento de software que usam os métodos ágeis normalmente usam métodos manuais e de baixa tecnologia, podem ser usados post-its num quadro branco, ou podem ser desenhados gráficos, esboços, arquiteturas e outros diagramas que permitem à equipa acompanhar o trabalho e para que todos possam ver e alterar o conteúdo representado no quadro.

Um quadro branco é muito flexível e é possível pôr qualquer coisa nele, em qualquer posição, de qualquer tamanho e forma sem quaisquer restrições (ao contrário de um sistema eletrónico). É rápido de desenhar ou efetuar alguma alteração em segundos e muito simples e eficiente reorganizar um conjunto de post-its. A informação importante pode ser destacada no quadro branco, a sua natureza visual incita as pessoas a lembrar-se da informação quando olham para o quadro em vez de irem à procura quando esta informação está fora de vista, e permite que tanto

a equipa como todas as pessoas que passam e reparam no quadro possam acompanhar e verificar o trabalho a ser desenvolvido pela equipa, bem como o espírito e carácter da equipa. O mais importante de tudo é que o quadro branco permite a colaboração, todas as discussões de uma equipa acontecem à volta do quadro branco sejam estas de progresso, falhas ou desenho. O quadro branco torna-se um ponto central de informação para a comunicação e colaboração de uma equipa [13].

A partir do momento em que foi adotado o ciclo de vida ágil foram fornecidos ao departamento de produção vários quadros brancos e post-its. Cada *team-leader* e a sua equipa tinha direito a pelo menos um quadro branco. O quadro branco da minha equipa era usado para visualizar prioridades de tarefas a partir dos post-its, efetuar brainstorming e desenhar diagramas e esboços relativos aos projetos desenvolvidos pela equipa, quando era necessário usar o quadro e não havia espaço, o conteúdo a ser apagado era passado para papel (caso ainda fosse necessário).

Estava também presente um quadro branco na sala onde eram efetuadas as *stand-up meetings* diárias. O quadro era usado principalmente para acompanhar as tarefas principais de cada equipa.

Figura 8 - Quadro branco e post-its

JIRA

O JIRA é uma ferramenta desenvolvida pela Atlassian que oferece ferramentas de relatório e acompanhamento de erros e incidentes, bem como algumas funcionalidades para a gestão de projetos [14].

O JIRA foi introduzido na Ad Infinitum Business já no final do estágio como ferramenta para comunicar e acompanhar erros durante a fase de testes depois do *deploy* das versões executáveis de cada projeto no servidor local. No entanto como havia *stand-up meetings* diárias, normalmente os erros eram comunicados nessa altura e com a ajuda dos quadros brancos e post-its, pelo que o JIRA não teve muita adesão na empresa. Esta ferramenta seria mais útil num ambiente onde a

comunicação é unidirecional e não há constantemente o contacto entre aqueles que efetuam os testes, e os que desenvolvem o software.

I.4.6.Sistema de controlo de versões

Um sistema de controlo de versões permite aos desenvolvedores manter a versão atual bem como versões prévias de ficheiros que podem ser de código fonte, páginas Web ou documentação. Todas as alterações feitas nestes ficheiros são possíveis de rastrear a partir deste sistema e é possível reverter de uma versão atual para uma versão anterior de um ficheiro ou projeto caso necessário [15].

Apache Subversion

Dentro do conjunto de sistemas de controlo de versões disponíveis no mercado, foi adotado pela empresa o Apache Subversion (svn), um software distribuído gratuitamente sob a licença da Apache.

Cada projeto teria uma conta de svn separada e era usada pelos programadores atribuídos ao projeto e um programador poderia sincronizar o código do qual estava a trabalhar com a versão mais recente no svn a qualquer altura. Quando necessário, a versão mais estável no svn era exportada para um servidor local como versão de demonstração para ser testada pelos *stakeholders*, gestores de produto e *marketeers*.

Com a introdução do *Scrum* na empresa, era incentivado a exportação da versão mais estável todas as quintas feiras.

II. PROJETOS DESENVOLVIDOS

II.1. 8XBIZ MOBILE

II.1.1.Problemática contextual

8xbiz

8xbiz é um dos principais projetos a serem desenvolvidos na empresa Ad Infinitum Business que consiste num portal de classificados e diretório de produtos e serviços das mais diversas áreas.

O nome 8xbiz significa negócios infinitos e tenta que, tudo o que possa ser vendido possa ser classificado e em qualquer região no mundo.

Este portal está alojado com o domínio: "*8xbiz.com*", foi desenvolvido em Joomla[3], contendo 20 áreas de negócio, cada uma destas possuindo a sua própria instalação Joomla, a sua base de dados e um subdomínio do portal.

A *homepage* é uma instalação Joomla que contem hiperligações para cada uma das áreas deste portal. No entanto apesar do portal ser constituído por várias aplicações diferentes, o utilizador perceciona o 8xbiz como uma única aplicação.

É de extrema importância referir este portal, pois o meu primeiro objetivo de estágio foi desenvolver uma aplicação móvel que iria replicar algumas das funcionalidades de uma das áreas deste, nomeadamente a área Go.

Figura 9: Logo do portal 8xbiz

[3] Joomla é um sistema de gestão de conteúdo (CMS) aberto e gratuito construído usando a arquitectura MVC, escrito em PHP e que usa o MySQL como sistema de gestão de base de dados [60].

Figura 10: *Homepage* do portal 8xbiz (www.8xbiz.com)

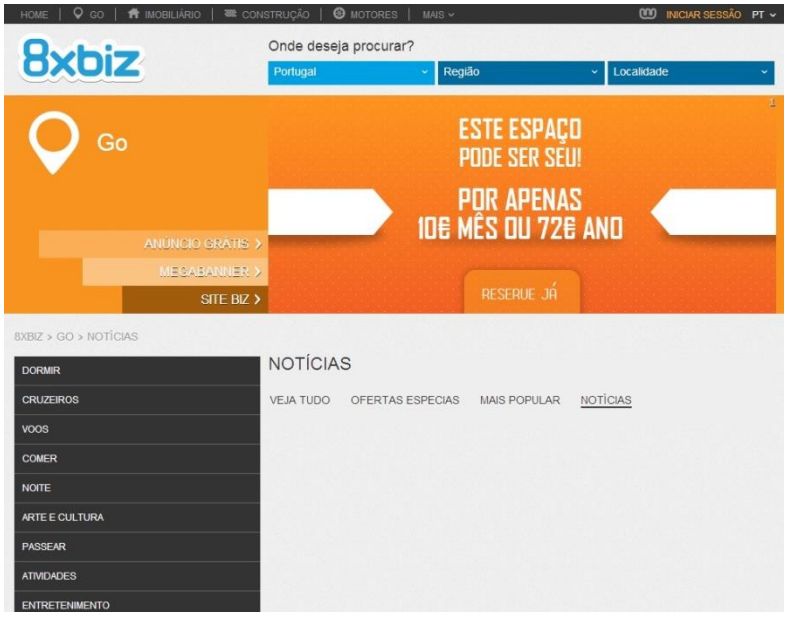


Figura 11: Página inicial da area go do portal 8xbiz (go.8xbiz.com)

8xbiz mobile

8xbiz mobile foi o nome dado à aplicação que iria replicar algumas funcionalidades da área Go do portal 8xbiz (*go.8xbiz.com*) numa versão mobile e seria desenvolvida por mim durante o estágio na empresa Ad Infinitum Business.

O Go é uma área do portal 8xbiz dedicada ao turismo. O utilizador poderia encontrar locais onde dormir, onde comer, atividades de lazer, entretenimento, arte e cultura, aluguer de veículos e até encontrar eventos a decorrer no local desejado.

A empresa sentiu então uma necessidade de portar algumas destas funcionalidades para uma aplicação numa versão mobile e adicionar ainda outras que exploram mais a fundo as capacidades de um *smartphone* (ex. o utilizador poder obter direções desde a sua localização ao local desejado, receber notificação quando está perto de um local de que "gosta", ou até efetuar uma chamada telefónica apenas clicando num botão dentro da aplicação quando um contacto está disponível, suportando também o próprio Skype).

II.1.2.Requisitos

Requisitos Funcionais

RF1. A aplicação deverá ser capaz de listar todos os classificados da área Go através das suas categorias e subcategorias.

RF2. Ao iniciar a aplicação, o utilizador pode definir a localização onde deseja encontrar os classificados.

RF3. Em paralelo ao requisito RF2, o utilizador pode usar a sua posição geográfica para encontrar locais que estão perto de si.

RF4. A aplicação deverá ser capaz de detetar a localização geográfica automaticamente e, se necessário, incitar o utilizador a ativar alguma opção do *smartphone* que desbloqueie o acesso a esta.

RF5. Deverá ser possível para o utilizador visualizar a seguinte informação de cada classificado caso esta esteja disponível em base de dados: texto descritivo, galeria de fotos, localização no mapa e contactos.

RF6. A aplicação deverá ser capaz de notificar o utilizador quando este se encontrar perto de um item classificado.

RF7. Ao visualizar a localização de um classificado no mapa, o utilizador pode também opcionalmente visualizar o trajeto entre a sua posição e a posição geográfica do classificado.

RF8. Ao visualizar os contactos de um classificado, caso este inclua o número telefónico, deverá ser possível ao utilizador realizar uma chamada telefónica apenas com o clique de um botão.

RF9. Caso o classificado inclua também um contacto Skype, deverá ser possível realizar também uma chamada Skype para este contacto.

RF10. No caso de o utilizador tentar uma chamada Skype e o seu *smartphone* não possuir o software compatível para tal, a aplicação deverá detetar e incitar o utilizador a instalar o software apropriado e, se possível, providenciar o link para tal.

RF11. Devido ao **RNF4**, para facilitar o desenvolvimento da aplicação foi aceite o seguinte: Cada classificado pertencia a uma categoria e uma subcategoria e continha pelo menos a seguinte informação:

- Id da cidade (presente na base de dados de países, regiões e cidades)
- Nome da categoria e sub-categoria a que pertence
- Nome do classificado
- Descrição
- Lista de imagens
- Coordenadas da posição geográfica
- Numero de telemóvel/telefone
- Email
- Morada
- Contacto skype(opcional)
- Tipo de produtos ou serviços prestados (opcional)
- Lista de produtos ou serviços prestados (opcional):
  - Nome do produto
  - Imagem
  - Preço

Requisitos Não Funcionais

RNF1. A aplicação deverá estar disponível nos sistemas operativos Android e iOS, dando prioridade ao Android.

RNF2. Os dados dos classificados da área Go a ser representados na aplicação móvel estariam presentes numa base de dados externa em MySQL.

RNF3. Os dados dos países e as suas regiões e cidades estariam também presentes numa base de dados em MySQL.

RNF4. O desenvolvimento da aplicação móvel deveria começar sem o desenho ou implementação das bases de dados.

A lista de requisitos aqui apresentada é uma versão final, pois esta esteve em constante modificação durante todo o desenvolvimento da aplicação.

II.1.3.Pesquisa e análise do problema

Os requisitos não implicavam qual o tipo de aplicação seria (nativa ou *Web*), o que me deu a liberdade de estudar qual seria a melhor escolha. Segue-se então uma descrição destes dois tipos de aplicação.

Aplicação nativa

Uma aplicação nativa, é uma aplicação desenvolvida usando uma linguagem de programação específica (ex. Objective C para iOS ou Java para Android) para ser utilizada num aparelho ou plataforma específicos e usando uma linguagem de programação e *Framework* específicas para cada uma dessas plataformas [16]. A aplicação pode ser previamente compilada para código máquina para um *hardware* específico na qual ela irá correr, ou compilada em *runtime* como é o caso do Android [17].

Aplicação Web

A aplicação Web é uma aplicação desenvolvida em HTML[4], JavaScript[5] e CSS[6], normalmente a aplicação Web está alojada num servidor Web e qualquer dispositivo com um browser que consiga interpretar esta tecnologia poderá correr a aplicação [18, 19]. Atualmente com a tecnologia HTML5, CSS3 e Web kit[7], e com a evolução do hardware dos dispositivos moveis já é possível desenvolver uma aplicação Web com uma interface quase tão natural como a da aplicação nativa, e com a utilização do WebStorage[8] e WebSQL[9] é possível ainda diminuir significativamente a quantidade de pedidos ao servidor e o tempo de download da aplicação [20, 21, 22]. No entanto a performance da aplicação Web em relação à aplicação nativa será sempre inferior visto o processo adicional de interpretação do código HTML, CSS ou JavaScript em *runtime* [23, 24, 25].

[4] HTML é uma linguagem de marcação usada para descrever a estrutura de uma página Web [76].

[5] JavaScript é uma linguagem de programação normalmente usada para desenvolver scripts *client-side* que são posteriormente interpretados por um browser [75].

[6] CSS é uma linguagem de estilo usada para descrever a apresentação e formatação de uma página Web [76].

[7] Web kit é um motor de renderização de páginas Web para browsers [77].

[8] Web Storage é um conjunto de métodos e protocolos usados para armazenamento de dados no *browser* [21].

[9] Web SQL é uma API que pode ser usada por aplicações Web para armazenar dados em bases de dados (no lado do cliente) que podem ser consultadas usando uma variante do SQL.

Baseando-me nos requisitos para o desenvolvimento da aplicação móvel, na Tabela 1 segue-se então uma comparação entre os dois tipos de aplicação acima descritos distinguindo vantagens e desvantagens da sua utilização [18, 19, 20, 23, 24, 25]:

	Aplicação Web		**Aplicação nativa**	
	Vantagens	**Desvantagens**	**Vantagens**	**Desvantagens**
1	A mesma aplicação corre em várias plataformas e sistemas operativos;	Algumas funcionalidades são ainda muito limitadas (acesso à câmara, GPS, contactos, etc..);	Mais fácil de trabalhar, já que usa as funcionalidades nativas e um SDK[10] específico para o aparelho em questão;	Consome muitos recursos desenvolver e manter uma aplicação nativa para várias plataformas.
2	Consome menos recursos de desenvolvimento e manutenção;	É necessário digitar o endereço da aplicação no browser.	Melhor desempenho;	É necessário o download e instalação para correr a aplicação, e downloads frequentes para a manter atualizada.
3	Não necessita de estar presente numa *app store*, nem de aprovação desta;	Poderá ser difícil encontrar a aplicação já que não está listada numa *app store*.	Interface mais natural e completa;	Dependendo da plataforma em questão, o processo de aprovação de uma aplicação numa *app store* pode ser longo e tedioso, e nem sempre poderá ser sucedido.
4	Está sempre atualizada;		Possibilidade de estar numa *app store* o que a torna mais fácil de encontrar;	Os utilizadores da aplicação podem nem sempre usar a ultima versão desta, o que torna a sua manutenção e suporte muito mais difíceis.

Tabela 1 – Aplicação Web vs Aplicação nativa

[10] O SDK é um conjunto de ferramentas de desenvolvimento de software que permite a criação de aplicações numa certa plataforma.

JSON

O JSON é um formato de texto que facilita a troca de dados estruturados entre todas as linguagens de programação. É de fácil leitura e escrita para os humanos, ao mesmo tempo que permite que uma máquina possa efetuar o *parsing* (analisar) e gerar texto usando este mesmo formato [26].

O JSON pode ser estruturado através de um conjunto de pares de nome/valor, ou de uma lista ordenada de valores, estas estruturas de dados são universais e todas as linguagens de programação modernas são capazes de as suportar [27]. No JSON estas estruturas podem ter as seguintes formas:

- **Objeto**: Um objeto é um conjunto de pares nome/valor. A descrição de um objeto começa com "{" e acaba com "}", cada nome é seguido de ":" e cada par é seguido de ",".

- **Array**: Um *array* é uma lista ordenada de valores. Um *array* começa com "[" e acaba com "]" e os seus valores são separados por ",".

- **Valor**: Um valor pode ter as seguintes formas:
  - String
  - Numero
  - Objecto
  - Array
  - True
  - False
  - Null

O seguinte exemplo mostra uma possível representação de uma pessoa através de um objeto JSON:

```
{
    "firstName": "John",
    "lastName": "Smith",
    "isAlive": true,
    "age": 25,
    "height_cm": 167.64,
    "address": {
        "streetAddress": "21 2nd Street",
        "city": "New York",
        "state": "NY",
        "postalCode": "10021-3100"
    },
    "phoneNumbers": [
        { "type": "home", "number": "212 555-1234" },
        { "type": "office",  "number": "646 555-4567" }
    ]
}
```

REST

Os requisitos da aplicação móvel implicavam também um acesso da aplicação a dados presentes em repositórios externos (ex: países, classificados), foi estudado então qual a melhor abordagem para o acesso a estes dados.

Representational state transfer (REST) é um estilo de arquitetura de software[11] que consiste num conjunto coordenado de restrições arquiteturais aplicadas a componentes[12], conectores[13] e elementos de dados[14] dentro de um sistema distribuído de hipermédia[15]. O REST ignora os detalhes da implementação do componente e a sintaxe do protocolo de maneira a se concentrar nos papéis dos componentes, as restrições em relação à sua interação com outros componentes e a sua interpretação de elementos de dados significantes [28, 29].

As restrições arquiteturais presentes no estilo arquitetural REST são as seguintes [28, 30]:

- **Cliente-Servidor –** Uma interface uniforme separa os clientes dos servidores. Esta separação de conceitos[16] implica que os clientes não se preocupem com o armazenamento de dados, deixando esta preocupação interna a cada servidor. Esta particularidade permite aperfeiçoar a portabilidade do código do cliente. Da mesma maneira os servidores não se preocupam com a interface do utilizador ou o estado do utilizador, o que torna o desenvolvimento e implementação dos servidores mais simples e escalável. Tanto os servidores como os clientes poderão ser substituídos e desenvolvidos independentemente desde que a interface entre eles se mantenha inalterada.

- **Sem estado** (*stateless*) – Cada pedido efetuado por qualquer cliente contém toda a informação necessária para servir o pedido e o estado da sessão[17] é guardado no cliente. No

[11] A arquitectura de software é definida por uma configuração de componentes, conectores e dados, todos estes limitados nas suas relações de maneira a alcançar um conjunto desejado de propriedades arquitecturais [63].

[12] O componente é uma unidade abstracta de instruções de software e de estados internos que transformam os dados através da sua interface [63]. Segundo Garlan e Shaw, os componentes são simplesmente os elementos que efectuam a computação [64].

[13] O conector é um mecanismo abstracto que efectua a mediação da comunicação, coordenação e cooperação entre componentes [63].

[14] Os dados são elementos de informação que são transferidos de um componente, ou recebidos por um componente através de um conector [63].

[15] Hipermédia é um meio de comunicação não linear que pode conter gráficos, áudio, vídeo texto ou hiperlinks [74].

[16] Em informática, a separação de conceitos (*separation of concerns*, SoC) é um princípio que defende a separação de um programa informático em secções distintas, de maneira que a cada secção lhe compete um conceito. Um programa que incorpore o SoC é chamado de modular [65].

[17] A sessão é o registo semi-permanente de um intercâmbio interactivo de informação, também conhecido como um diálogo entre dois ou mais dispositivos de comunicação, ou entre um computador e um utilizador [67].

entanto o estado da sessão pode ser transferido pelo servidor para outro serviço como uma base de dados de maneira a manter um estado persistente por um certo período de tempo e permitir autenticação.

- ***Cache*** – Os clientes podem guardar as respostas em cache[18]. Na resposta deve ser incluído, implícita ou explicitamente, se esta pode ser guardada em cache ou não. A boa gestão de *cache* elimina completamente ou parcialmente algumas interações entre o cliente e o servidor, aprimorando assim a escalabilidade e performance.

- **Interface uniforme** – Uma interface uniforme simplifica e desassocia a arquitetura, o que permite que cada parte possa evoluir independentemente. Esta interface é definida pelos seguintes princípios:
  - Identificação/representação dos recursos quando é efetuado um pedido (ex. XML[19], JSON);
  - Tendo a representação de um recurso um cliente tem informação suficiente para o eliminar ou modificar.
  - Cada mensagem deverá incluir informação suficiente que descreva como esta deverá ser processada.
  - Um cliente não assume uma ação em particular a um recurso sem que esta esteja descrita na representação do recurso (excetuando pontos fixos de entrada para a aplicação). Este princípio é chamado de *Hypermedia as the engine of application state*.

- **Sistema em camadas** – Um cliente não consegue identificar se está conectado ao servidor final (*end server*) ou a um intermediário. O uso de servidores intermediários pode aumentar a escalabilidade do sistema pois permite implementar um balanceamento de carga (*load-balancing*) e partilhar caches. Estes servidores intermediários podem servir também para aplicar políticas de segurança.

A performance, escalabilidade, simplicidade, modificabilidade, visibilidade, portabilidade e segurança são algumas das propriedades que as restrições arquiteturais previamente referidas pretendem induzir no estilo arquitetural REST [31].

RESTful Web service

Web service é um sistema de software desenhado para suportar interação e interoperabilidade[20] de uma máquina para outra máquina através de uma rede [32].

[18] A *Web cache* é um mecanismo de armazenamento temporário de documentos Web (ex: páginas HTML, imagens, ficheiros JavaScript ou CSS). A *Web cache* permite reduzir o uso da largura de banda, carga no servidor e alguma latência perceptível ao utilizador [66].

[19] XML é uma linguagem de marcação que define um conjunto de regras para a escrita de documentos num formato que seja legível tanto para o humano como para a máquina [71].

[20] Interoperabilidade em computação é a capacidade da troca de informação entre dois sistemas ou componentes [72].

RESTful Web service é a categorização de um *Web service* que implementa as restrições do estilo arquitetural REST. Uma implementação de um *RESTful Web service* segue os seguintes princípios [33]:

- ***Stateless*** – Tal como referido anteriormente, de maneira a permitir escalabilidade e performance, um servidor não deve guardar estados. Um pedido HTTP de um cliente a um servidor deve conter nos seus *headers* e no *body* toda a informação necessária para que o componente do lado do servidor possa gerar a resposta.

- **URIs[21] bem estruturados** – A estrutura dos URIs deve ser simples, previsível e fácil de entender.

- **Representação em XML, JSON ou ambos** – A representação de um recurso numa resposta de um servidor ou num pedido ao servidor deve ser feita através de XML, JSON ou ambos.

- **Uso explícito dos métodos HTTP**[22] – Este princípio estabelece um mapeamento de um para um entre as operações CRUD[23] e os métodos HTTP.

URI	GET	PUT	POST	DELETE
Exemplo1: O recurso é uma coleção de elementos. (http://exemplo.com/recursos)	**Lista** os URIs dos elementos da coleção.	**Substitui** a coleção por uma nova coleção.	**Cria** um novo elemento na coleção (devolvendo o URI gerado automaticamente para o elemento).	**Elimina** a coleção completamente.
Exemplo2: O recurso é um elemento específico. (http://exemplo.com/recursos/item3)	**Devolve** a representação do elemento em questão.	**Modifica** o elemento.	(Normalmente não é utilizado para elementos de uma coleção).	**Elimina** o elemento da coleção.

Tabela 2 - Utilização dos métodos HTTP na API de um *RESTful Web service*

É de notar também que não há nenhum standard oficial para a implementação um *Web service* RESTful, já que este segue apenas um estilo arquitetural e não um protocolo.

[21] O URI é uma string de caracteres que identifica um recurso (seja este físico ou abstracto). Comparado com o URL, o URI não necessariamente indica a localização do recurso [70].

[22] Os métodos do protocolo HTTP são usados pelo cliente para indicar uma acção a ser efectuada pelo servidor num dado recurso [68].

[23] As operações CRUD (*create, read, update and delete*) são as quarto operações básicas de armazenamento pressistente (em português significa criar, ler, modificar e eleminar) [69].

MVC – Model View Controller (Modelo Vista Controlador)

O MVC é um padrão arquitetural. Em contraste com um estilo arquitetural (como o REST) que define restrições arquiteturais, um padrão arquitetural define a estrutura fundamental de um sistema de software. Ele oferece um conjunto de subsistemas predefinidos, especifica as suas responsabilidades e inclui regras e diretrizes de como são organizadas as relações entre eles [34].

O MVC divide uma aplicação em 3 componentes (modelo, vista e controlador) [35]:

- **Modelo:** O modelo é independente da representação do *output* (a maneira como os dados são apresentados para fora da aplicação) ou o comportamento do *input* (como os dados entram na aplicação). Os componentes do tipo modelo contêm o núcleo funcional da aplicação, estes encapsulam os dados e as funcionalidades principais da aplicação e exportam métodos que efetuam um processamento específico na aplicação (ex: incrementar um valor num campo de um modelo), os controladores acedem a estes métodos em nome do utilizador. O modelo também exporta métodos que permitem o acesso aos seus dados, estes métodos são usados pelas vistas para adquirir dados a serem exibidos ao utilizador.

- **Vista:** Estes componentes disponibilizam a informação ao utilizador da aplicação (*output*). As vistas obtêm os dados do modelo e é possível haver várias vistas associadas a um modelo. Cada vista possui também um controlador associado a esta.

- **Controlador:** Os controladores recebem o *input* da aplicação. O controlador traduz estes *inputs* e efetua pedidos a um modelo ou a uma vista. Um utilizador interage com a aplicação apenas através dos controladores.

Os modelos contêm o núcleo da aplicação, os controladores manipulam o *input* e as vistas disponibilizam o *output*. Ambos os controladores e as vistas compreendem a interface do utilizador de uma aplicação [35].

Abordagem Top Down

A abordagem **top down**, é uma abordagem que pode ser usada para o desenvolvimento de software e que se foca no planeamento. Pretende fragmentar um sistema para permitir a compreensão dos seus subsistemas. Nesta abordagem a formulação da visão geral do sistema parte de uma instancia final para a inicial, sendo que cada nível será detalhado do mais alto para o mais baixo. A abordagem **top down** contrasta com a abordagem **bottom up** em que a formulação do sistema começa pela instancia inicial até à final, são formulados sistemas em que passo a passo, estes se tornam subsistemas para dar lugar a um sistema final maior [36].

II.1.4.Solução técnica

A seguinte solução técnica foi apresentada ao Diretor de Produção e ao CEO da empresa (os *stakeholders* principais nesta aplicação) através de uma apresentação em PowerPoint.

Tendo em conta as vantagens e desvantagens descritas na Tabela 1 e os requisitos para desenvolver a aplicação, a minha proposta seria desenvolver uma aplicação Web mobile em HTML5 que poderia ser acedida a partir de um browser que iria satisfazer os requisitos RF1, RF2, RF3, RF4, RF5 e RF7. Para satisfazer os requisitos do RF6, RF7, RF8, RF9, RF10 seria desenvolvida também uma aplicação nativa que funcionava em conjunto com a aplicação Web. A este tipo de aplicações dá-se o nome de aplicação híbrida (entre Web e nativa) [19]. Desta forma seria possível aceder à aplicação Web a partir um browser comum no entanto para tirar partido de todas as funcionalidades, o utilizador poderia instalar a aplicação nativa, esta simulava um browser disponibilizando a aplicação Web e estaria também à escuta de certas ações do utilizador na aplicação Web, podendo assim efetuar chamadas e usufruir do GPS para determinar a posição exata do utilizador. A Figura 12 descreve a arquitetura deste sistema através do diagrama de componentes.

O desenvolvimento da aplicação Web implicaria o desenvolvimento de uma aplicação em HTML5 e CSS3, algo em que ainda não tinha experiência. No entanto o requisito RNF1 implicava o desenvolvimento da aplicação móvel para Android e iOS e apesar de já ter alguma experiência com o desenvolvimento para Android, não tinha qualquer experiência a desenvolver para iOS. Como já tinha alguma experiência com o JavaScript, que é o mais essencial no desenvolvimento de uma aplicação Web em HTML5, a curva de aprendizagem para o desenvolvimento da aplicação Web seria muito menor que a curva de aprendizagem para o desenvolvimento de uma aplicação para iOS. Além disso não havia nenhum colaborador na empresa com experiência a desenvolver para iOS enquanto que havia alguns que possuíam alguma experiência com HTML5 os quais poderia pedir ajuda eventualmente. Concluí então que o custo de desenvolvimento de duas aplicações para iOS e Android seria muito maior que o custo de desenvolvimento de uma só aplicação Web, além de que a aplicação Web poderia futuramente funcionar em outros dispositivos que não apenas Android e iOS.

Abordagem Top Down

A abordagem *top down* para desenvolver a aplicação móvel poderia ser arriscada, já que a empresa adotava métodos ágeis, a probabilidade da mudança de requisitos era muito alta, tinha pouca experiência em desenvolvimento de software no âmbito empresarial e seria complicado efetuar testes no estágio inicial do desenvolvimento do software. No entanto acabei por escolher a abordagem *top down* para desenvolver a aplicação móvel, pois segundo o requisito RNF4, era a

única que permitia começar o desenvolvimento imediato da aplicação e não necessitava esperar pelo desenvolvimento das bases de dados, das quais a aplicação móvel estaria dependente.

A minha solução proposta seria então planear a arquitetura do sistema e dividir este em componentes que iriam comunicar através de interfaces. As interfaces entre os componentes seriam descritas mediante os requisitos. O desenvolvimento do software começaria pelos componentes da arquitetura de mais alto nível até aos componentes de mais baixo nível usando para o seu desenvolvimento, tal como referi no primeiro capítulo, um modelo em espiral.

Apesar de usar uma abordagem *top down* para desenvolver todo o sistema que implementa a aplicação móvel, cada um dos componentes da arquitetura poderia ser desenvolvido usando uma abordagem *bottom up*, o que iria facilitar os testes no estágio inicial do desenvolvimento de cada componente.

Arquitectura do sistema

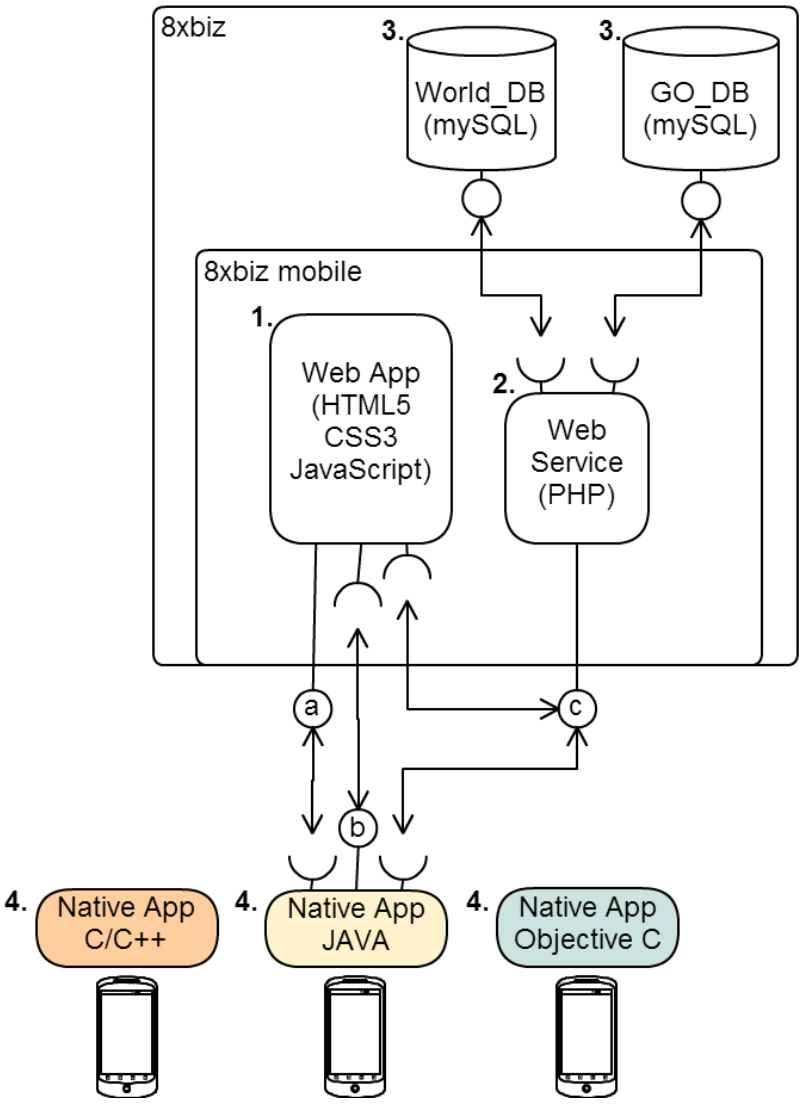


Figura 12 - Diagrama de componentes da arquitetura da aplicação móvel 8xbiz híbrida

Descrição dos Componentes

1.- **Web App** – A Web App é a interface do utilizador, é uma aplicação Web desenvolvida em JavaScript, HTML5 e CSS3. Esta aplicação usufrui da API[24] de um *Web service* (interface c) para obter dados de países, regiões, cidades e classificados. A Web App oferece também uma interface (interface a) que permite a qualquer aplicação com acesso à internet e um motor de renderização compatível com HTML5, CSS3 e JavaScript interpretar o seu conteúdo. Caso a aplicação Web seja acedida por uma aplicação nativa, esta poderá usufruir da sua interface (interface b) para enviar comandos que executam funcionalidades nativas do dispositivo móvel.

2.- ***RESTful Web service*** – Este Web service estaria conectado às bases de dados dos países e dos classificados do Go e oferecia uma API (interface c) compatível com os princípios de um *RESTful Web service*. A implementação do *Web Service* trazia todas as vantagens da implementação de um estilo arquitetural REST, sendo que neste caso em particular a modificabilidade e a portabilidade eram as mais importantes, pois sem uma base de dados ainda definida, permitia o desenvolvimento separado da aplicação Web e da aplicação nativa desde que fossem mantidas interfaces consistentes.

3.- **Bases de dados** – Estas bases de dados iriam conter todos os dados dos classificados e dos países, regiões e cidades. O único componente nesta arquitetura que iria aceder à base de dados seria o *Web service*. Outras aplicações não relevantes a esta arquitetura também iriam aceder e popular estas bases de dados.

4.- **Native App –** Esta aplicação iria implementar as funcionalidades descritas pelos requisitos RF6, RF7, RF8, RF9, RF10, para isso a aplicação irá estar à escuta de comandos especiais através da sua interface (interface b), executados a partir da aplicação Web e acionando assim funcionalidades especiais pela aplicação nativa como aceder ao GPS, aceder e guardar contactos no *smartphone*, efetuar chamadas e comunicar com outras aplicações existentes no *smarphone* como o Skype ou o Android Navigator. Para satisfazer o requisito RF6, esta aplicação estaria à escuta da posição GPS do dispositivo e num certo intervalo de tempo enviava as suas coordenadas através da API do *web service* (interface c), caso o dispositivo se encontrasse a uma certa distancia de um classificado, seria enviada uma notificação ao dispositivo para o utilizador visualizar o classificado.

Descrição das Interfaces

a.- **Web App –** A interface da aplicação Web é descrita da seguinte maneira:

- **Pedido GET** por HTTP ao URL http://mobile.8xbiz.com – É devolvido um ficheiro HTML que descreve toda a estrutura da aplicação.

[24] API (Aplication Programming Interface) é nada mais que uma especificação de como dois programas de software deverão interagir entre si.

- O mesmo pedido com a seguinte ***query string***[25] http://mobile.8xbiz.com**?native=true** – Permite à aplicação Web saber que a aplicação pode efetuar funcionalidades nativas de uma dispositivo móvel, e quando necessário, efetuar um pedido específico à interface da aplicação nativa.

b.- **Native App –** A aplicação nativa iria observar todos os pedidos de URL efetuados pela aplicação Web. Quando estes pedidos eram efetuados sob o protocolo http, não era efetuada qualquer ação, no entanto quando eram detetados os seguintes pedidos, a aplicação nativa iria acionar funcionalidades correspondentes a estes:

- **tel:000000000** – a aplicação nativa inicia o serviço de chamadas do android e liga para o número: "000000000";
- **skpye:username** – a aplicação nativa inicia o skype e envia o pedido para ligar ao "username" (caso a aplicação não esteja instalada, incita o utilizador a instalar)
- **save:number=000000000&name=name&address=address&...** – a aplicação nativa inicia o serviço de guardar contactos do Android.
- **goto:lat;lang** – a aplicação nativa iniciava qualquer aplicação GPS instalada no Android (como o *navigator*) e comunica a esta como destino de rota: "lat;lang";

No caso do Android, a aplicação nativa iria usufruir do objeto "WebView" [37] do Android SDK que contém um interpretador de HTML5, JavaScript e CSS, este objeto seria usado para carregar a aplicação web. A "WebView" contém também uma propriedade (nomeadamente o "shouldOverrideUrlLoading") que pode ser usada para escutar todos os pedidos de URL. Esta propriedade permite então implementar a interface da aplicação nativa que recebe comandos (sob pedidos de URL) a partir da aplicação Web.

c.- **Web service API –** Esta API descreve como deve ser efetuado o acesso aos dados do *RESTful web service*. Os recursos retornados por esta API seriam descritos através de objetos JSON, escolhi o JSON por me sentir mais confortável a trabalhar e ser compatível com qualquer linguagem de programação, o que permite uma futura integração de qualquer aplicação com esta API. Já que o web service apenas necessário para acesso a dados (excluindo assim qualquer pedido de criação, alteração ou eliminação de dados), então apenas seria necessário o método GET do http para satisfazer os pedidos da API. Apresenta-se de seguida uma descrição da API. Não segui as especificações do JSON Schema para a representação da estrutura dos objetos JSON, de maneira a simplificar esta.

- **GET: http://mobile.8xbiz.com/service/countries** - Devolve todos os países disponíveis;

```
{
   Results:[
      {
         Id: (String),
         Name: (String)
      }
   ]
   Pages: (number)
}
```

[25] A *query string* é um parâmetro de um URL ou URI que contém dados para serem interpretados por uma aplicação Web ou pelo recurso identificado pelo URL.

- **GET: http://mobile.8xbiz.com/service/regions[?country=Portugal]** - Devolve todas as regiões disponíveis em Portugal:

```
{
   Results:[
      {
         Id: (String),
         Name: (String),
         Country: {
            Id: (String),
            Name: (String)
         }
      }
   ]
   Pages: (number)
]
```

- **GET: http://mobile.8xbiz.com/service/cities[?country=Portugal|&region=Madeira]** - Devolve todas as cidades disponíveis, ou todas as cidades disponíveis na Madeira:

```
{
   Results:[
      {
         Id: (String),
         Name: (String),
         Region: {
            Id: (String),
            Name: (String)
         }
      }
   ]
   Pages: (number)
]
```

- **GET: http://mobile.8xbiz.com/service/sites?city=Funchal[&country=Portugal|&region=Madeira]** - Devolve todos os classificados disponíveis na Madeira:

```
{
   Results:[
      {
         Id: (String),
         URI: (String),
         Name: (String),
         Description: (String),
         Gallery: [
            {
               Name: (String),
               URI: (String)
            }
         ]
         Contacts: {
            Phone: (String),
            Email: (String),
            Address: (String),
            Skype: (String)
         }
         Coordinates: {
            Latitude: (String),
            Longitude: (String)
         },
         City: {
            Id: (String),
            Name: (String)
         },
         Category: {
            Id: (String),
            Name: (String)
         }
         Subcategory: {
            Id: (String),
            Name: (String)
         }
      }
   ]
   Pages: (number)
}
```

- **GET: http://mobile.8xbiz.com/service/sites/SiteID** - Devolve a descrição do classificado com o Id SiteID:

```
{
   Id: (String),
   URI: (String),
   Name: (String),
   Description: (String),
   Gallery: [
      {
         Name: (String),
         URI: (String)
      }
   ]
   Contacts: {
      Phone: (String),
      Email: (String),
      Address: (String),
      Skype: (String)
   }
   Coordinates: {
      Latitude: (String),
      Longitude: (String)
   },
   City: {
      Id: (String),
      Name: (String)
   },
   Category: {
      Id: (String),
      Name: (String)
   }
   Subcategory: {
      Id: (String),
      Name: (String)
   }
}
```

- A adição do parâmetro opcional "**lang**" numa *query string*, especifica em que língua os resultados devem ser retornados (ex. http://mobile.8xbiz.com/service/resources?**lang**=en). Caso este parâmetro não seja especificado ou seja inválido, os resultados serão apresentados na língua do browser ou em inglês.

Casos de utilização

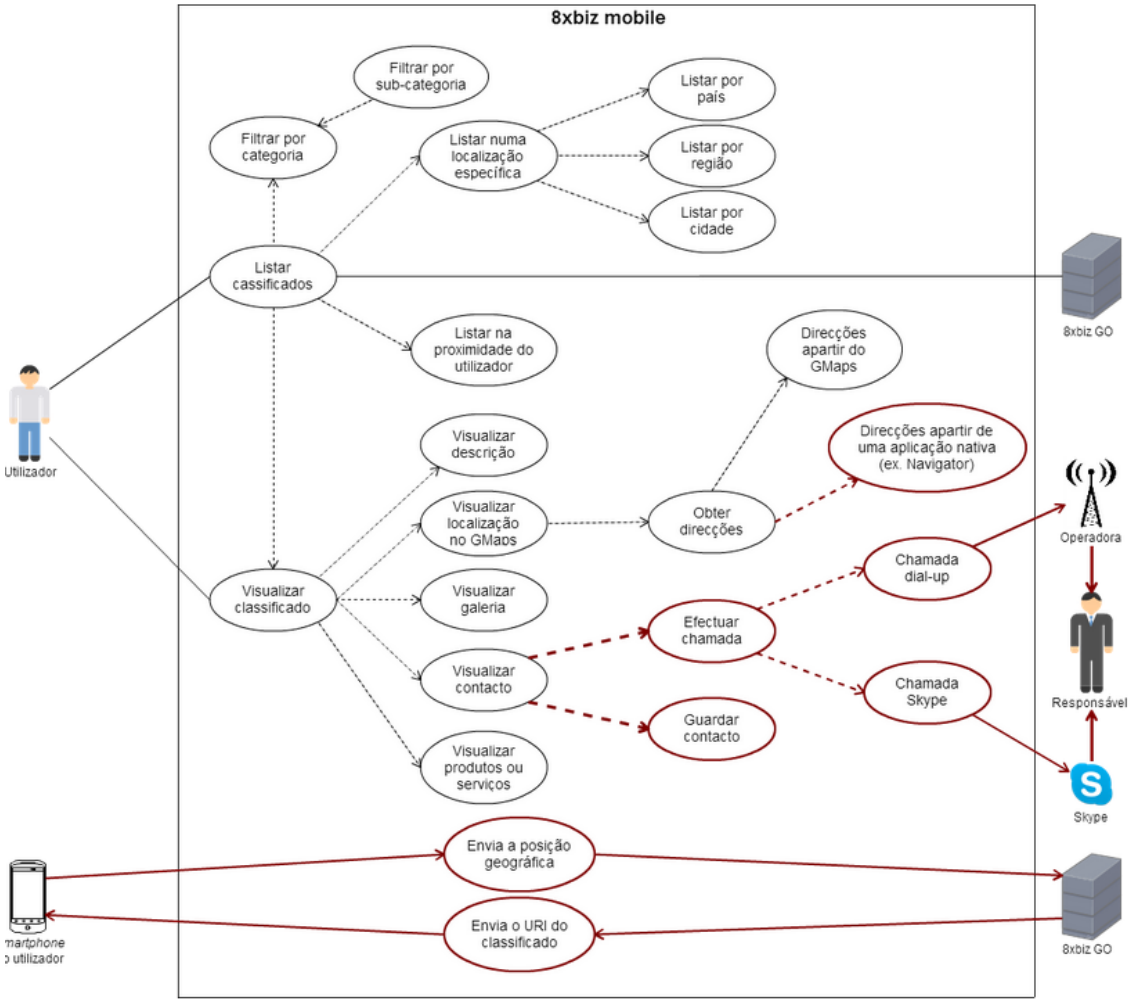


Figura 13 - Casos de utilização da aplicação 8xbiz mobile

Os casos de utilização representados a vermelho apenas têm efeito quando é usada a aplicação nativa.

Apesar dos elementos "8xbiz Go", "operadora" e "Skype" não se incluírem nas diretrizes dos casos de utilização, decidi inclui-los neste caso em especial, de maneira a oferecer uma melhor interpretação e também representar apenas o que seria desenvolvido na aplicação móvel.

Mockups


0
ABC 07:04 PM
Restaurante
Voçê encontra-se perto do Espaço Funchal
Cancelar
Abrir
1
1
Menu
Bacalhau na brasa
5.00€
Bife à portuguesa
5.50€
Bife com cogumelos
5.00€
Bife de atum
5.50€
Menu Descrição Galeria Localização Contacto
2
Descrição
Espaço Funchal
Com uma localização priveligiada convidamo-lo a sentir a história de uma das ruas mais antigas do centro do funchal, onde residiam as personalidades mais ilustres da altura.
A Ementa é inspirada na cozinha regional e Portuguesa tendo também opção de cozinha internacional, sempre com uma apresentação sofisticada e excelência na qualidade do serviço.
Menu Descrição Galeria Localização Contacto
3
Galeria
Menu Descrição Galeria Localização Contacto
4
Localização
Direcções
Menu Descrição Galeria Localização Contacto
5
Contacto
Nome: Espaço Funchal
Telefone: +351291000000
Skype: espaço.funchal
Email: espacofunchal@email.com
Address: Rua da Carreira
nº 9000-042 Funchal
Chamar
Chamar
Guardar contacto
Menu Descrição Galeria Localização Contacto


6
Back
9
8xbiz mobile
Espaço Funchal
Restaurante
1
Bar A
Bar
Bar B
Bar
Restaurante C
Restaurante
Bar D
Bar
Bar E
Bar
Restaurante F
Restaurante
7
8xbiz mobile
Perto de mim
9
Localização específica
8
8
8xbiz mobile
Perto de mim
9
Localização específica
País
Região
Cidade
Go
9
9
7
8xbiz mobile
Comer
6
10
Dormir
Entretenimento
Shopping
10
Categorias
9
Restaurantes
Bares
Fast food
Take away

A interface da aplicação pode ter início a partir do *mockup* 0 ou do *mockup* 7. O ultimo *mockup* representa a interface da aplicação usando o *tablet*, onde é possível facilitar mais informação ao utilizador, ou seja, poderão ser apresentados os *mockups* para *smartphone* do 1-5 (à direita) em conjunto com os *mockups* 6, 9 e 10 (à esquerda).

II.1.5.Ferramentas usadas

Sencha Touch

O Sencha Touch é uma Framework em JavaScript que implementa o padrão arquitetural MVC e é usada para desenvolver aplicações Web para dispositivos móveis com ecrã *touch*. Esta Framework contém uma rica gama de componentes de interface para o utilizador, é conhecida devido à sua versatilidade e modularidade, à sua alta performance na interface de utilizador deixando uma latência na sua interface quase impercetível ao utilizador e possui alguma relevância na comunidade web [38].

Na altura da execução do projeto foi necessário escolher uma Framework para o desenvolvimento da aplicação Web. Esta decisão recaiu entre o Sencha Touch, o JQuery Mobile e o MGWT, as 3 principais *Frameworks* recomendadas pela comunidade Web na altura. Estas 3 *Frameworks* foram testadas num iPhone 4 com o iOS 4 e um Android ZTE v875 com um processador de 800MHz e uma memória RAM de 512MB.

O JQuery Mobile foi excluído logo a partida por possuir uma interface de utilizador com uma latência percetível muito elevada e pobre, pois não foi capaz de simular uma interface tão natural como as outras 2 *Frameworks*.

Tanto o Sencha Touch como MGWT conseguiram simular uma interface quase tão natural como a de uma aplicação nativa, no entanto o MGWT requeria um desenvolvimento em Java enquanto que o Sencha Touch requeria um desenvolvimento em JavaScript no qual eu me sentia mais à vontade para desenvolver, de modo que dei prioridade à escolha do Sencha Touch.

Mediante as razões acima referidas, ficou decidido em reunião de controlo com os stakeholders o desenvolvimento da aplicação Web usando a Framework Sencha Touch.

MGWT (Mobile Google Web Toolkit)

Durante o desenvolvimento do primeiro protótipo usando a Framework Sencha Touch foi verificado que o tempo de download da aplicação era muito elevado, foram efetuados novos estudos que demonstraram que o MGWT seria a melhor escolha.

Tal como o Sencha Touch, o MGWT é uma Framework que implementa o padrão arquitetural MVC e é usada para desenvolver aplicações Web para dispositivos móveis [39]. No entanto esta Framework tem algumas particularidades que a distinguem do Sencha Touch. A aplicação Web é desenvolvida em Java num IDE, neste caso foi utilizado o Eclipse, e contém um *builder* que gera um ficheiro HTML e um ficheiro Javascript com o código reduzido (*minified*) e que contem apenas o código necessário para correr a aplicação, ou seja, não é necessário o download de toda a Framework no dispositivo móvel, o que reduz o volume de download significativamente. Esta Framework permite também que o dispositivo possa guardar a aplicação em *localstorage* [21], o que faz com que seja apenas necessário o download do código *javascript* e CSS da aplicação no primeiro acesso ou quando é lançada uma nova versão da aplicação.

Android SDK

O Android SDK é um conjunto de ferramentas e bibliotecas que permitem desenvolver, compilar, testar e depurar uma aplicação em Java para Android [40]. O SDK é fornecido pela Google, a mesma empresa responsável pelo desenvolvimento do sistema operativo Android.

CodeIgniter

O CodeIgniter é uma *Framework* usada para o desenvolvimento de aplicações em PHP usando também o padrão arquitetural MVC [41]. O CodeIgniter foi escolhido, não só pela sua simplicidade e performance, mas por já ter utilizado esta ferramenta na disciplina de Aplicações Centradas em Redes e estar bem familiarizado com esta.

Juntamente com esta Framework foi usada a biblioteca codeigniter-restserver, uma biblioteca que torna mais simples a implementação do estilo arquitetural REST usando o CodeIgniter como Framework, esta biblioteca pode ser encontrada no GitHub [42].

II.1.6.Execução do projeto

1ª Iteração – Aplicação nativa para Android.

O desenvolvimento do sistema que implementa a aplicação móvel começou então pelo desenvolvimento do componente da arquitetura: "native app" para Android.

Após o seu desenvolvimento, para testar a aplicação nativa para o Android foi desenvolvida uma página Web com apenas 4 botões que executavam código JavaScript para efetuar os pedidos de URL descritos pela interface b da arquitetura (ex. window.location="tel:000000000";), o que forçava a página Web a efetuar um pedido para ser intercetado pela aplicação nativa.

2ª Iteração – Throw away prototype da interface da aplicação Web.

Objetivos:

Na segunda iteração da espiral foi desenvolvido um protótipo para descartar com o objetivo de avaliar o seguinte:

- Comportamento da framework Sencha Touch;
- A interface da aplicação;
- A integração desta aplicação com a aplicação nativa desenvolvida previamente;

Devido à necessidade que a empresa tinha de possuir versões executáveis de demonstração dos projetos a serem desenvolvidos para motivar e angariar *shareholders*, ficou acordado entre os *stakeholders* o desenvolvimento inicial de uma aplicação Web de demonstração, o conteúdo deste protótipo seria completamente estático e o protótipo seria descartado no fim da iteração. A interface desta aplicação seria composta apenas pelos *mockups* 1-5 e iria descrever como classificado o Restaurante Espaço Funchal, cujo proprietário era também um *stackeholder*.

Resultados:

- A integração com a aplicação nativa funcionou perfeitamente;
- A interface foi aprovada pelos stakeholders;
- Os stakeholders consideraram a aplicação em si como: "lenta a carregar";

Foram realizados também testes de usabilidade com 10 colaboradores da empresa onde era testada também a performance da aplicação e da *framework* nos *smartphones* dos mesmos. A interface provou ser simples e intuitiva, no entanto para o tempo de download da aplicação verificou-se uma média de 12 segundos, podendo chegar até aos 20 segundos, o que demonstrou ser desconfortável para o utilizador.

Como o objetivo deste protótipo seria também demonstrar aos *shareholders* e *stakeholders* o que estaria a ser desenvolvido como aplicação móvel, este protótipo foi lançado para o mercado de aplicações do Android (Google Play).

3ª Iteração – Throw away prototype da interface da aplicação Web.

Objetivos:

Para a terceira iteração da espiral foi desenvolvido um novo protótipo para descartar e com o objetivo de avaliar o seguinte:

- Comportamento da framework MGWT;

Já que o protótipo anterior demonstrou resultados de tempos de download muito altos, foi desenvolvido novamente um protótipo para aplicação Web com a mesma interface do protótipo anterior. No entanto, seria usado como *framework* o MGWT. O objetivo seria comparar o tempo de download da aplicação entre as duas *frameworks* (MGWT e Sencha Touch).

Resultados:

- O tempo de download desceu para uma média de 4 segundos, o que ficou aprovado pelos stakeholders.

4ª Iteração – Evolutionary prototype da aplicação Web.

Na quarta iteração foi desenvolvida a aplicação Web idêntica ao protótipo anterior usando a *framework* MGWT. No entanto, este protótipo teve em conta já a futura integração com a API do *RESTful web service* e a extensibilidade da aplicação Web para todas as funcionalidades definidas nos requisitos e para o resto dos *mockups*.

A partir desta iteração o desenvolvimento da aplicação móvel foi desenvolvido em equipa entre mim e a Mariana (nome fictício) e foi da minha responsabilidade a liderança desta equipa.

5ª Iteração – Interface completa

O objetivo da quinta iteração foi adicionar ao protótipo anterior toda a interface da aplicação Web idealizada a partir dos *mockups*.

6ª Iteração – Desenvolvimento da API

Foi desenvolvido um *RESTful web service* utilizando como *framework* o CodeIgniter juntamente com a biblioteca codeigniter-restserver. Foi implementada a API para a comunicação com a aplicação Web através de pedidos HTTP. Os dados (objetos JSON) devolvidos pelo *web service* foram estáticos e permaneceram estáticos até o desenvolvimento das bases de dados das quais o *web service* iria comunicar estivessem completas.

7ª Iteração – Integração da aplicação Web com o web service

Os dados estáticos da aplicação Web passaram a ser carregados dinamicamente através a API do *web service*. Estes dados foram também traduzidos do português para o inglês, usando a língua do browser para decidir em que linguagem oferecer os dados ao utilizador.

Funcionamento e Interface final da Aplicação

Ao aceder à aplicação, era proposto ao utilizador encontrar classificados perto de si, ou em um local específico onde o utilizador podia selecionar o país, região e cidade a pesquisar.

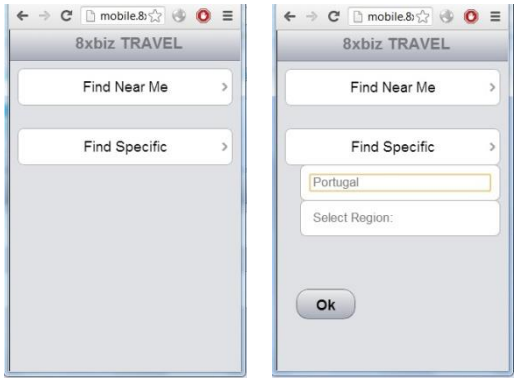


Figura 14 - Vista inicial da aplicação num *smartphone*

Ao clicar OK, eram apresentados ao utilizador todos os classificados na área selecionada por este. (Nota: os seguintes classificados são fictícios, usados apenas para testes)

Figura 15 - Apresentação dos classificados ao utilizador

Uma vez na área de classificados, é possível efetuar a filtragem destes pelas suas categorias.

Figura 16 - Filtragem dos classificados

Ao seleccionar um classificado o utilizador tem acesso à seguinte informação sobre o estabelecimento: Produtos/serviços, descrição, galeria de imagens, geolocalização e contactos.

Figura 17 - Visualização de um classificado

Ao ser visualizada num ecrã de um *tablet* ou desktop, a aplicação adapta-se ao ecrã, podendo mostrar mais opções e funcionalidades.

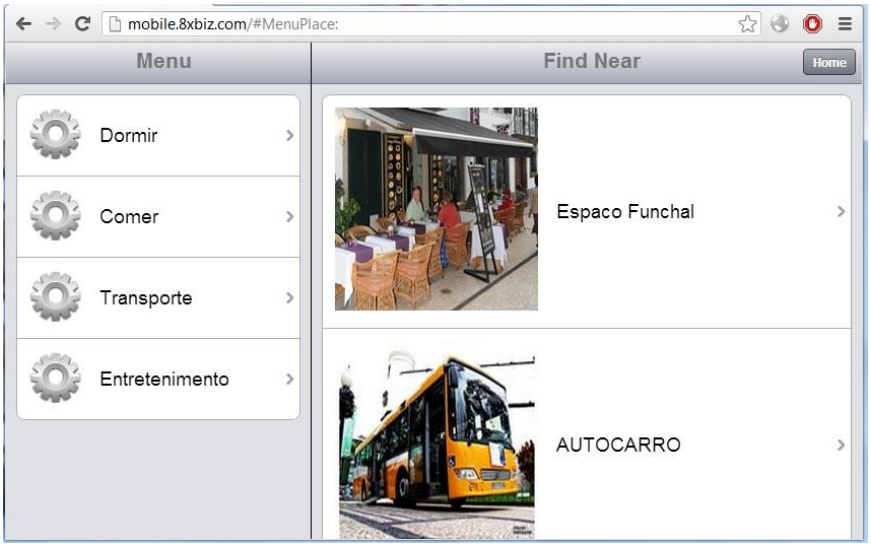


Figura 18 - Visualização da lista de classificados num *tablet*

Figura 19 - Visualização de um classificado num *tablet*

II.1.7.Conclusão

Em relação aos requisitos, faltou apenas desenvolver a pesquisa de classificados em relação à posição geográfica do utilizador (**RF3)**, desenvolver uma aplicação nativa para iOS que replicasse as funcionalidades nativas como no Android, afinar alguns erros nas traduções e adicionar mais línguas. Estas novas tarefas ficaram à responsabilidade da Mariana. Algo mais que poderia ser acrescentado a esta aplicação no futuro poderia ser ordenar os classificados por posição (do mais perto ao mais longe) ou por *ranking*, reduzindo assim o tempo que o utilizador demora a encontrar o classificado de que está à procura e oferecer também aos utilizadores uma opção de avaliar e comentar os classificados.

Era possível então aceder à aplicação com todas as funcionalidades Web em qualquer dispositivo equipado com um browser com suporte para HTML5 e Web-kit através do endereço

http://mobile.8xbiz.com. E com as funcionalidades extra de efetuar chamadas e receber notificações tendo instalada a aplicação nativa para Android.

O resultado deste projeto foi uma aplicação móvel *location aware*, com um design *responsive*, e que agradou bastante aos *stakeholders* não só devido à sua interface bastante simples e intuitiva que permitia a qualquer turista poder saber o que há à sua volta, como bares, restaurantes, alojamento e entretenimento e obter informações sobre estes, como produtos e serviços disponíveis, posição geográfica com direções e contactos e também, permitia à aplicação saber a localização do utilizador e notifica-lo conforme a sua proximidade de certos classificados.

Foi uma pena que as bases de dados das quais a aplicação dependia nunca chegaram a ser desenvolvidas, pelo que a aplicação não passou do estágio de protótipo.

O desenho e implementação de todo o sistema por trás desta aplicação permitiu-me aprender e consolidar conhecimentos adquiridos em várias disciplinas lecionadas durante o meu período académico, principalmente nas áreas de arquiteturas de software e desenho de interfaces e experiência para o utilizador. Apesar de já estar familiarizado com o Android SKD, o desenvolvimento de uma aplicação Web e *responsive* foi algo novo para mim, o que me preparou melhor para o desenvolvimento de novas aplicações Web para o futuro.

II.2. SSO WEB SERVICE

II.2.1.Problemática contextual

Nesta secção irei falar de outras aplicações desenvolvidas na Ad Infinitum Business além do 8xbiz (WESHHA e 8xbiz payments), e qual o problema comum em todas elas, do qual seria a minha responsabilidade apresentar uma solução bem como a sua implementação.

WESHHA

O WESHHA , outro projeto a ser desenvolvido na Ad Infinitum Business, era uma rede social e profissional. O nome dado ao projeto é na verdade um acrónimo que significa: "work, education, sports, health, hobbies, arts" (trabalho, educação, desporto, saúde, passatempos e arte).

Um dos seus objetivos era fornecer uma plataforma ao utilizador capaz de construir o seu currículo online nas áreas acima referidas, e autenticá-lo através de outras instituições qualificadas para tal (como centros de formações, escolas, universidades, etc..). À medida que o utilizador fornece as suas informações e as autentica, este aumenta os seus "pontos weshha" que quantificam o seu currículo. Esta plataforma pretende fornecer também ao utilizador uma ferramenta de pesquisa de trabalho conforme a sua profissão, e este terá mais probabilidade de obter emprego conforme as suas aptidões e pontos weshha. Empresas e outras instituições podem também efetuar pesquisa e encontrar, através de um algoritmo chamado de "matching", os empregados mais qualificados para as suas necessidades.

Para além da pesquisa de trabalho, o Weshha iria permitir também inserir e gerir classificados no portal 8xbiz, a compra de publicidade que apareceria em todos os portais e aplicações desenvolvidas na Ad Infinitum Business, e a compra de micro-sites desenhados conforme a área de negócio desejada, sendo estas as fontes de rendimento e sustentabilidade da empresa.

Esta plataforma está alojada a partir do domínio: "www.weshha.com"

Figura 20: Logo da plataforma weshha

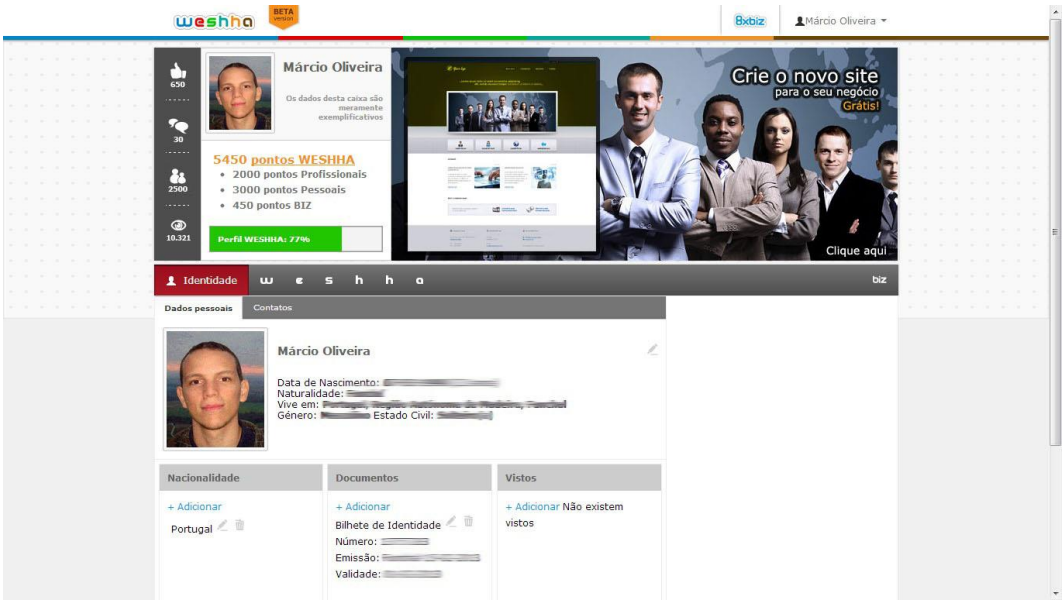


Figura 21 - Página inicial ao entrar no portal weshha

8xbiz payments

8xbiz payments foi o nome dado a uma plataforma que finalizava pagamentos de produtos e serviços disponibilizados pelo 8xbiz, Weshha, ou qualquer eventual aplicação desenvolvida pela Ad Infinitum Business. Cada aplicação era responsável por adicionar itens a um carrinho de compras virtual, em que o *checkout* deste era efetuado a partir da plataforma 8xbiz payments em http://payments.8xbiz.com.

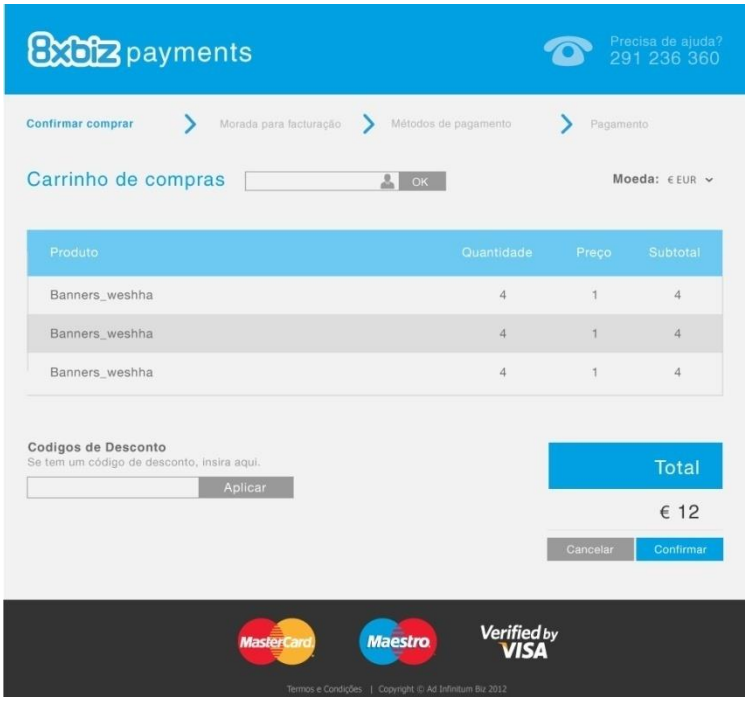


Figura 22 - Portal 8xbiz payments

Problema

A partir do dia 6 de Janeiro foi-me atribuído um novo projeto a ser desenvolvido em paralelo à aplicação 8xbiz mobile, o desenvolvimento de um serviço de Single Sign On (SSO).

O Weshha era até então um portal completamente independente do *8xbiz*, cada um tinha a sua base de dados de utilizadores, o seu sistema de autenticação e um *backoffice* para inserção e edição de conteúdo. No entanto, foi entendido durante o desenvolvimento destas plataformas que a administração de classificados e publicidade do 8xbiz passasse para o Weshha, funcionando assim o Weshha como *backoffice* também do portal 8xbiz. O Weshha iria passar a ser capaz de administrar todas as aplicações desenvolvidas na Ad Infinitum Business e oferecendo ao utilizador um único registo e login.

O portal 8xbiz estava sendo desenvolvido usando o CMS Joomla, o Weshha estava sendo desenvolvido usando a Framework Yii para PHP e outras aplicações como o *8xbiz payments* estavam sendo desenvolvidas usando a Framework CodeIgniter.

Cada um destes portais tinha o seu próprio sistema de sessões e autenticação, incluindo o próprio portal 8xbiz que continha várias instalações Joomla e cada uma destas tinha a sua própria gestão de sessões, ou seja, um utilizador ao navegar entre os portais da Ad Infinitum Business teria que efetuar um login e logout em cada um deles.

A empresa encontrou aqui uma necessidade de implementar um sistema que permitisse ao utilizador ter apenas uma conta e um login comum a todas as aplicações. E ao fazer logout de uma aplicação, fazia logout de todas ao mesmo tempo.

Coube a mim a responsabilidade de propor e implementar a melhor solução técnica para este problema.

II.2.2.Requisitos

Para o desenvolvimento deste serviço não havia uma lista de requisitos funcionais e não funcionais específicos. O Objetivo era propor e implementar uma solução que permitisse ao utilizador em qualquer aplicação efetuar o *login* usando a sua conta Weshha sem necessidade de efetuar o *login* novamente ao mudar de aplicação e aplicar este mesmo conceito também para o *logout*.

II.2.3.Pesquisa e análise do problema

Serviço de Single Sign On

O *Single sign on* é uma propriedade de controlo de acesso com segurança a múltiplos sistemas de software sendo estes relacionados no entanto independentes. Com esta propriedade um utilizador ao conectar-se uma vez ganha acesso a todos os outros sistemas sem ter que fornecer os seus dados de acesso novamente. Da mesma maneira, ao se desconectar a um deles, o acesso a todos os outros sistemas é também terminado [43].

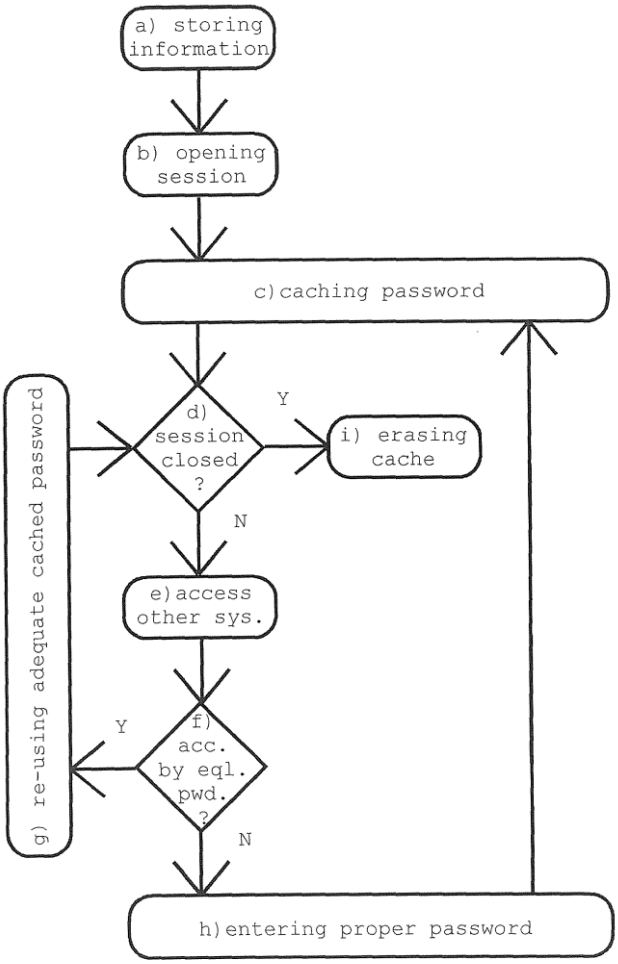


Figura 23 - Fluxograma de exemplo de um sistema de SSO [8]

OpenID

O OpenID é um *standard* aberto que permite a utilizadores serem autenticados por certos sites co operacionais conhecidos como *Relying Parties* usando um serviço para terceiros, eliminando assim a necessidade dos webmasters disponibilizarem o seu próprio sistema ad hoc permitindo assim aos utilizadores consolidar a sua identidade digital [44].

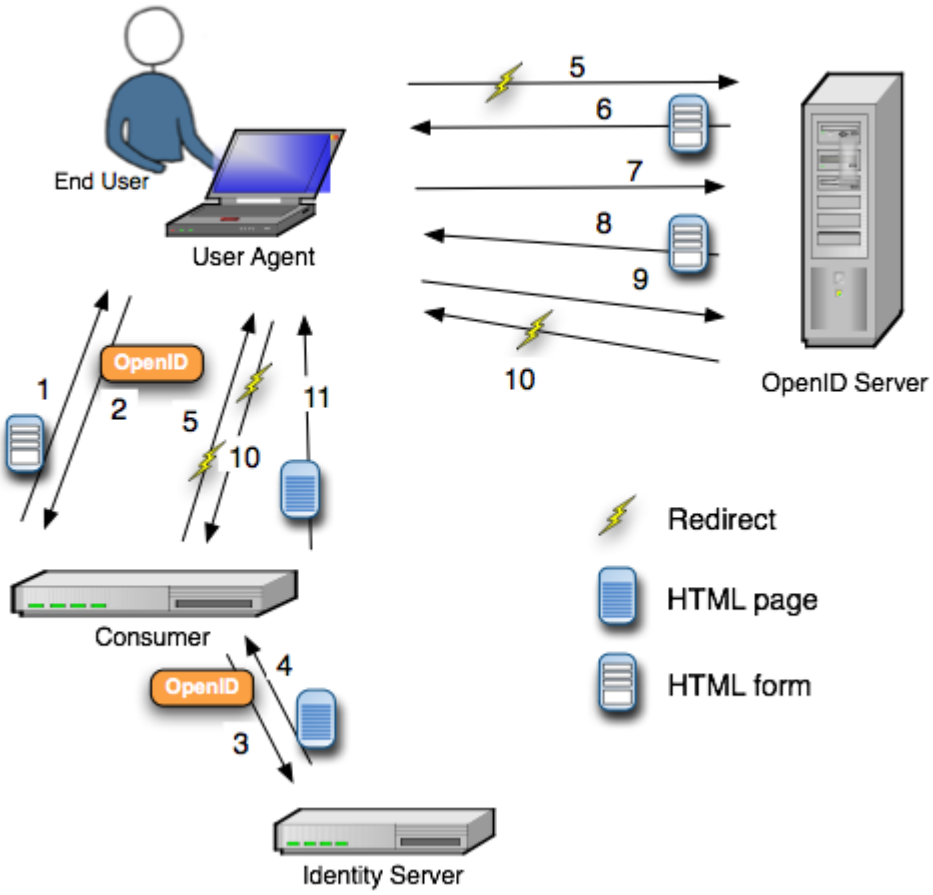


Figura 24 - Diagrama da sequência dos passos utilizados no protocolo OpenID [45]

Este protocolo segue então os seguintes principais passos [45]:

1. O utilizador (*end user*) através de um *browser* (*user agent*) tenta aceder a uma página ou outro recurso protegido num servidor Web (*consumer*), este servidor requer uma autenticação usando o protocolo OpenID e incita o utilizador a inserir o seu OpenID.
2. O *end user* insere e envia o OpenID que pode ter o seguinte formato: "my.openid.com" para o *consumer* através do seu *user agent*.
3. O *consumer* envia um pedido GET por http a outro servidor (*identity server*) utilizando o URL do OpenID (ex.: http://my.openid.com).
4. O *identity server* retorna um documento HTML ao *consumer*, este documento contem informações acerca do servidor Web (*OpenID server*) que irá efectuar a autenticação do utilizador.
5. O *consumer* redirecciona o *user agent* para o endereço Web do *OpenID server* encodificando no URL deste endereço o URL de retorno para o *consumer* em caso de sucesso de autenticação e o URL de retorno em caso de falha de autenticação.
6. Ao ser redireccionado para *OpenID server*, este devolve ao *user agent* um formulário de autenticação.
7. A partir deste formulário o utilizador preenche e envia ao *OpenID server* a sua identificação de *login* e a sua *password*.
8. O *OpenID server* devolve ao *user agent* um novo formulário, perguntando ao *end user* se este confia no *consumer* identificando o URL deste, e poderá indicar também quais os recursos que o *consumer* terá acesso através do *OpendID server* (ex.: email, nome, morada).
9. O *user agent* envia o POST ao *OpendID server* confirmando ou não confirmando a sua confiança no *consumer*.
10. Dependendo do sucesso de autenticação do utilizador no *OpendID server*, este envia um pedido de redireccionamento ao *user agent* para o URL de sucesso ou o URL de falha do *consumer*. Estes URLs foram enviados pelo *consumer* no passo 5.

11. Em caso de sucesso, o *consumer* poderá disponibilizar então os recursos pretendidos pelo utilizador no passo 1.

OAuth 2.0

O OAuth é um standard aberto de autenticação e autorização que, no contexto de uma arquitetura cliente-servidor, garante um acesso seguro de aplicações clientes a um recurso protegido num servidor (*resource server*) em nome do proprietário do recurso (*resource owner*). O OAuth especifica um processo que permite aos proprietários de um recurso autorizar o acesso a este sem partilhar as suas credenciais, ele permite que sejam gerados códigos de acesso (*access tokens*) por um servidor de autorização (*authorization server*) para aplicações clientes de terceiros com a aprovação do utilizador proprietário do recurso (*resource owner*). Depois de gerados os *access tokens* as aplicações podem usa-los para aceder a recursos alojados no servidor que contem estes recursos (*resource server*) [46]. O OAuth 2.0 foi desenhado para trabalhar especificamente com o http e substitui a sua versão anterior OAuth 1.0 já obsoleta [47].

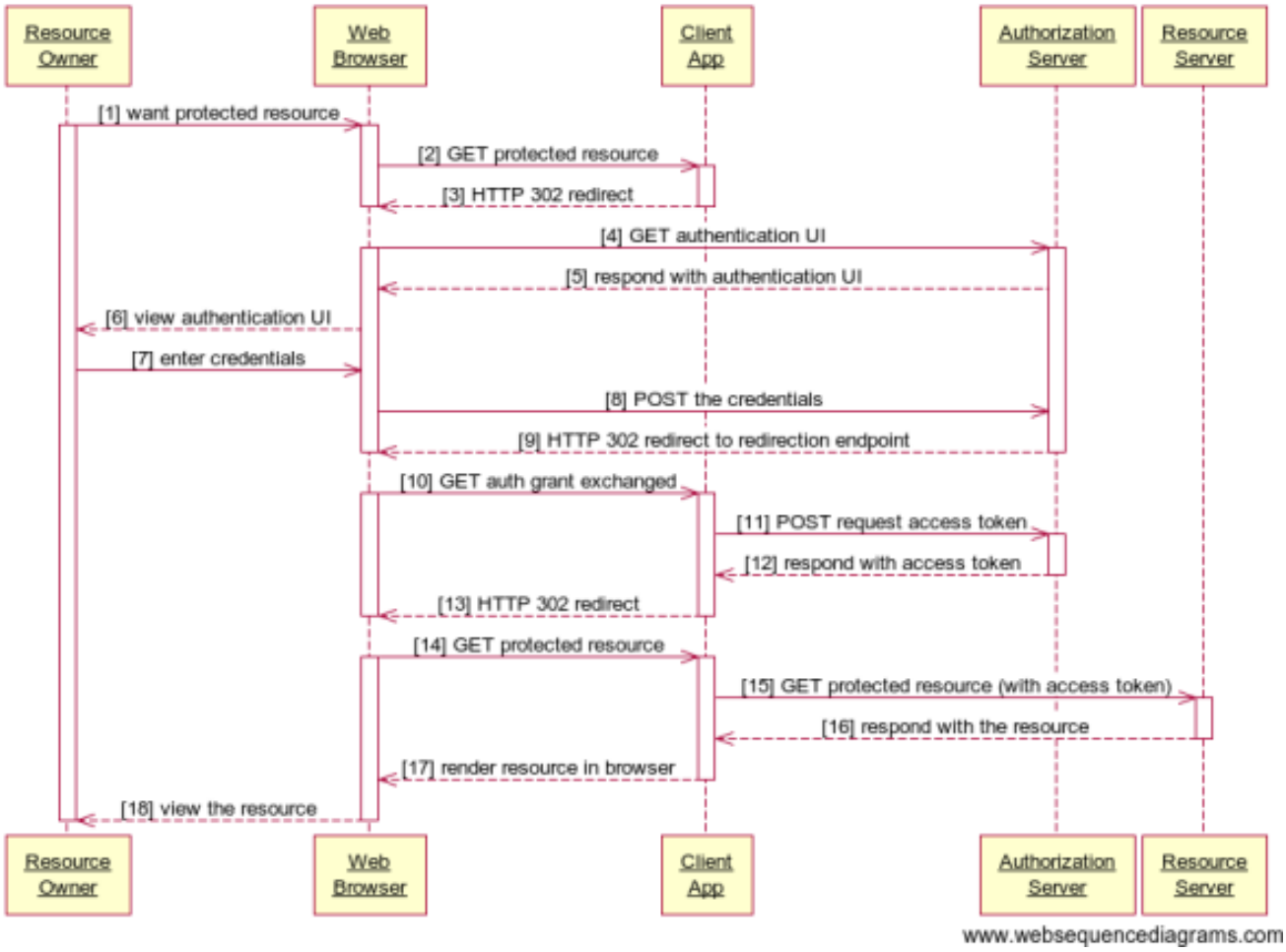


Figura 25 - Diagrama de sequência do protocolo OAuth 2.0 [48]

A nomenclatura do papel de cada actor no protocolo OAuth 2.0 é diferente do OpenID, no OAuth 2.0 o *end user* será o *resource owner*, o *user agent* será o Web browser (ou simplesmente *browser*) e o consumer será o *client app* (ou simplesmente *client*).

Segue-se então os seguintes passos que caracterizam a arquitectura do protocolo OAuth 2.0:

1. O *resource owner* deseja aceder a um certo recurso alojado no *resource server* através do *client* a partir de um *browser.*

2. O *browser* envia um pedido GET por http ao *client*.

3. O *client* verifica que este recurso necessita de autenticação para ser acedido. A autenticação é feita através do a*uthorization server* e por isso o *client* envia um pedido de redireccionamento ao *browser* para o *authorization server* codificando no URL a sua identificação e o URL de retorno depois do sucesso ou falha na autenticação.

4. O *browser* envia o pedido GET ao *authorization server*.

5. O *authorization server* devolve ao *browser* um formulário de autenticação. Caso o *resource owner* já esteja previamente autenticado no *authorization server* este passo poderá ser excluído, bem como os passos 6, 7 e 8.

6. O *resource owner* visualiza o formulário de autenticação, bem como os possíveis recursos que o *client* terá acesso.

7. O *resource owner* preenche os campos do formulário e confirma a sua confiança no *client*.

8. O browser envia as credenciais para o *authorization server*.

9. Em caso de falha de autenticação, o *authorization server* envia um pedido de redireccionamento ao browser para o URL de falha de autenticação do *client*. Em caso de sucesso, o *authorization server* gera um código chamado de *authorization grant* e codifica este código no URL de redireccionamento de sucesso para o *client*.

10. O *browser* envia um pedido GET que contém o *authorization grant* ao *client*.

11. O cliente envia o *authorization grant* ao *authorization server*.

12. *authorization server* verifica o *authorization grant*. A partir deste *authorization grant* o *authorization server* gera um *access token* e devolve-o ao *client* que o permite (enquanto o *access token* for válido) aceder aos recursos alojados no *resource server* em nome do *resource owner*.

13. O *client* poderá redireccionar então o *browser* de volta ao recurso do qual o *resource owner* tentou aceder inicialmente.

14. O *browser* envia novamente o pedido do recurso ao *client*.

15. O *client* verifica que que existe um *access token* para este *browser* de acesso ao *resource server* e envia um pedido de acesso ao recurso ao *resource server* juntamente com o *access token*.

16. O *resource server* valida o pedido do *client* através do *access token* e devolve o recurso.

17. O *client* devolve o recurso ao *browser*.

18. O *resource owner* visualiza o recurso.

Depois da autenticação efetuada, para aceder ao mesmo recurso ou outros recursos protegidos, serão apenas necessários os passos 1, 14, 15, 16, 17, 18. Se eventualmente o *resource owner* terminar a sua sessão no *authorization server*, este poderá invalidar todos os *access tokens* associados ao *resource owner* fazendo com que nenhum *client* tenha mais acesso aos recursos protegidos deste.

É muito importante também referir que as seguintes empresas notáveis no mercado das aplicações Web usam o OAuth 2.0 para oferecer autorização aos recursos dos seus utilizadores a múltiplas aplicações Web, sejam estas pertencentes à mesma empresa e/ou aplicações de terceiros:

- Amazon [49]
- Dropbox [50]
- Facebook [51]
- Google [52]
- LinkedIn [53]
- Microsoft [54]
- Paypal [55]
- Twitter [56]

Existem várias bibliotecas que tornam mais simples a implementação do *client* e do *authorization server*/*resource server* para a implementação do standard OAuth 2.0, como é o exemplo do OAuth 2.0 Server PHP [57, 58].

II.2.4.Soluções técnicas

Foram apresentadas 2 soluções técnicas para resolver o problema em questão. A primeira seria desenvolver um serviço de autenticação usando o standard OAuth 2.0 em conjunto com um *RESTful web service*, que fornecia os recursos mediante a sua autorização, e a segunda solução foi o desenvolvimento de uma biblioteca que implementava a sua própria API para gerar as cookies e sessões de todas as aplicações e *frameworks* a serem desenvolvidas na empresa. Decidi apresentar estas duas soluções visto que considerava a primeira solução como a melhor solução devido à sua escalabilidade e estabilidade, no entanto esta solução iria acarretar muito mais custos de implementação que a segunda solução, bem como implicaria muito mais perícia e especialidade dos desenvolvedores de todas as aplicações que necessitavam de autenticação. Por estas mesmas razões, os *stakeholders* preferiram a segunda solução técnica.

Estas soluções técnicas foram idealizadas por mim e pelo João (nome fictício). Ambos iríamos constituir a equipa de desenvolvimento que iria implementar este serviço de SSO e da qual eu seria responsável por liderar.

II.2.4.1.Solução 1: OAuth 2.0 RESTful Web Service

OAuth ou OpenID

Para esta solução foi escolhido inicialmente um dos seguintes métodos mais utilizados para a implementação de um serviço SSO: o OpenID ou o OAuth 2.0. A escolha recaiu pois na implementação do serviço utilizando o OAuth 2.0 visto ser mais simples para o utilizador (este não tem de decorar e inserir o seu OpenID) e também porque o OpenID não resolve o problema do "single sign off", ou seja, permite que as aplicações de terceiros continuem com a sessão ativa uma vez que o utilizador efetuou o logout. Possivelmente estas serão também algumas das razões pelas quais as grandes empresas no mercado das aplicações Web que fornecem serviços de SSO dão preferência ao OAuth 2.0 em relação ao OpenID.

Arquitectura do sistema

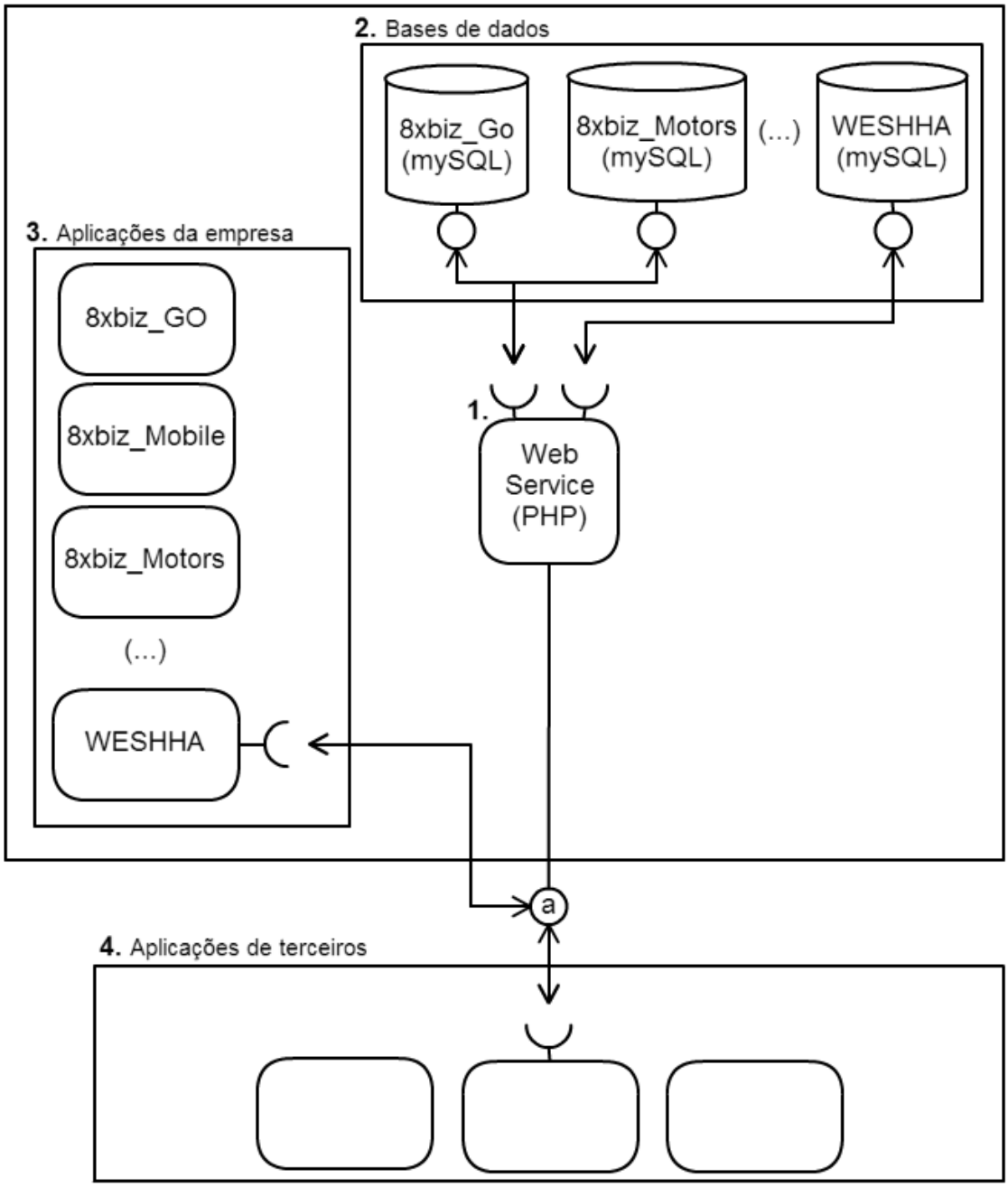


Figura 26 - Diagrama de componentes da arquitetura do serviço de autenticação

Descrição dos componentes

1.- Web Service – Tal como para a aplicação móvel, este *web service* também iria implementar um estilo arquitetural REST para o acesso e manipulação de dados. Como haveriam dados protegidos que iriam necessitar de autenticação, este web service iria também implementar uma interface compatível com o standard OAuth 2.0 (interface a). Uma vez escolhida esta arquitetura como solução técnica, este componente poderia vir a substituir o *web service* da aplicação móvel.

2.- Bases de dados – Bases de dados que continham todos os dados de todas as aplicações que dependiam da autenticação deste serviço.

3.- Aplicações – Estas seriam todas as aplicações a ser desenvolvidas na empresa que necessitavam de autenticação e acesso a recursos protegidos.

4.- Aplicações de terceiros – Tal como o Facebook, o Google ou outras empresas que fornecem a sua API de autenticação OAuth, também seria possível fornecer a API do *web service* a aplicações de terceiros para que estas pudessem efetuar autenticação com uma conta de utilizador do WESHHA e efetuar o acesso a alguns recursos. Apesar de isto não ser um requisito da empresa, a implementação desta arquitetura deixaria em aberto esta oportunidade.

Descrição da interface

A interface do *web service* (interface a) iria implementar o standard OAuth 2.0 e mais uma vez seria usado o JSON para descrever os recursos devolvidos pelo *web service*. Para implementar esta interface seria usada a biblioteca "CodeIgniter-OAuth-2.0-Server" (que neste momento já está descontinuada) [59].

II.2.4.2.Solução 2: Manipulação de cookies

Em relação à solução anterior, esta solução não implicaria grandes mudanças nas arquiteturas atuais das aplicações já existentes.

Na Figura 27 está representada a arquitetura do sistema, onde estão salientados a cor os componentes adicionais à arquitetura atual de maneira a tornar possível a implementação do serviço de SSO.

Arquitetura do sistema

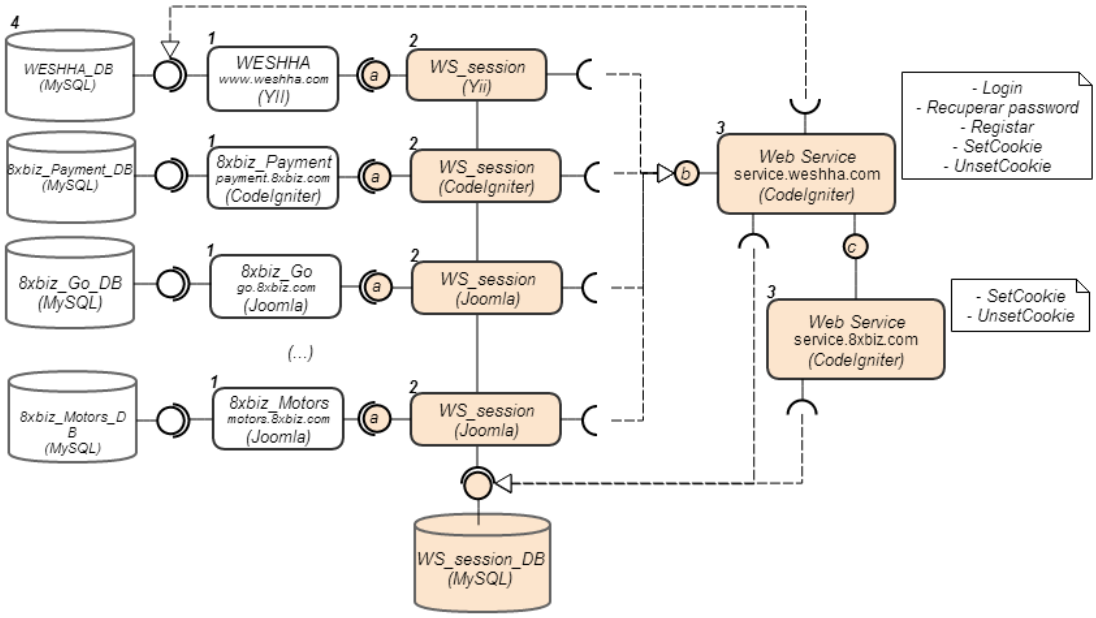


Figura 27 - Diagrama de componentes da solução alternativa

Descrição dos componentes

1.- Aplicações Web – Estas eram todas as aplicações que estariam a ser desenvolvidas na empresa. Todas as aplicações pertenciam ao domínio "8xbiz.com" à exceção do WESHHA que pertencia ao domínio "weshha.com". Ao contrário da solução anterior, com esta arquitetura, todas as aplicações mantinham as ligações às suas bases de dados. A gestão de sessões destas aplicações passaria a ser efetuada, não pela biblioteca fornecida pela Framework, mas pelo WS_session.

2.- WS_session – Esta era uma biblioteca desenvolvida em PHP. A biblioteca WS_Session foi desenvolvida a partir da biblioteca "Session" da Framework CodeIgniter de maneira a satisfazer as necessidades da arquitetura. Uma vez desenvolvida para CodeIgniter, foi desenvolvida também uma versão desta biblioteca para cada uma das Frameworks (Joomla e Yii) respeitando as diretrizes destas. O WS_Session passaria então a gerir as sessões das aplicações, fornecendo a estas uma interface para tal (interface a).

3.- Web Service – O *web service* era composto por duas instancias. A primeira instância estava presente no domínio "wesha.com", a função desta instancia do *web service* era não só efetuar o registo do utilizador e o login (funcionalidades que passariam da aplicação WESHHA para o *web service*) como também manipular o registo da sessão fornecendo uma interface para tal (interface b). A segunda instancia estava presente no domínio "8xbiz.com" e a sua função era manipular o registo de sessão para o domínio "8xbiz.com". Foi tomado o "accounts.google.com" como exemplo de maneira a definir as funcionalidades deste *web service*.

4.- WESHHA_DB – Esta é a base de dados que contém os utilizadores do WESHHA, é administrada pelo WESHHA no entanto o *web service* acede a esta base de dados para inserir novos utilizadores no ato de registo e validá-los no ato de login.

Descrição das interfaces

a.- WS_Session – Esta biblioteca oferece a seguinte interface a uma aplicação:

- **has_session()**: verifica as cookies e o registo de sessão no WS_session_DB para retornar true ou false se o utilizador está em sessão;

- **login()**: comunica com o *web service* através da interface b para este efetuar o login e registo da sessão.

- **get_user()**: acede ao WS_session_DB para devolver um objeto com os dados do utilizador guardados em sessão:
  - **W_id**; (id de utilizador do wesha)
  - **session_id**;
  - **email**;
  - **Nome**;
  - …

- **logout()**: comunica com o *web service* através da interface b para que este elimine os cookies e o registo de sessão do utilizador.

b.- Web service (weshha) – Esta interface funciona à base do redireccionamento de pedidos por HTTP:

- http://service.weshha.com – O redireccionamento do browser para o *web service* com este URL permite registar ou autenticar o utilizador (caso este ainda não o tenha feito). Uma vez concluído esse processo, o *web service* efetua um pedido de redireccionamento para "http://www.weshha.com".

- http://service.weshha.com/?return_url=url.de.retorno – Ao adicionar o parâmetro "redirect_url" na *query string*, o pedido de redireccionamento é efetuado para "http://url.de.retorno" em vez de "http://www.weshha.com".

- http://service.weshha.com/logout?return_url=url.de.retorno – O *web service* elemina o registo da sessão do utilizador e efetua o redireccionamento para o "http://url.de.retorno", caso este parâmetro não esteja presente na *query string*, o browser mantêm-se no "http://service.weshha.com".

c.- Web service (weshha) – Esta interface é idêntica à interface c funcionando também através de pedidos http:

- http://service.8xbiz.com?return_url=url.de.retorno – O redireccionamento do browser para esta instância do *web service* permite guardar o registo da sessão do utilizador na cookie para o domínio "8xbiz.com". Uma vez concluído esse processo, é efetuado um pedido de redireccionamento para o "http://url.de.retorno" ou, caso este parâmetro não esteja presente na *query string* o pedido de redireccionamento é efetuado para o "http://service.weshha.com".

Diagramas de sequencia

Os diagramas de sequencia representados na Figura 28 e na Figura 29 pretendem demonstrar a sequencia do processo de autenticação das aplicações. Na Figura 28 está representado o processo de login e na Figura 29 está representado o processo de logout.

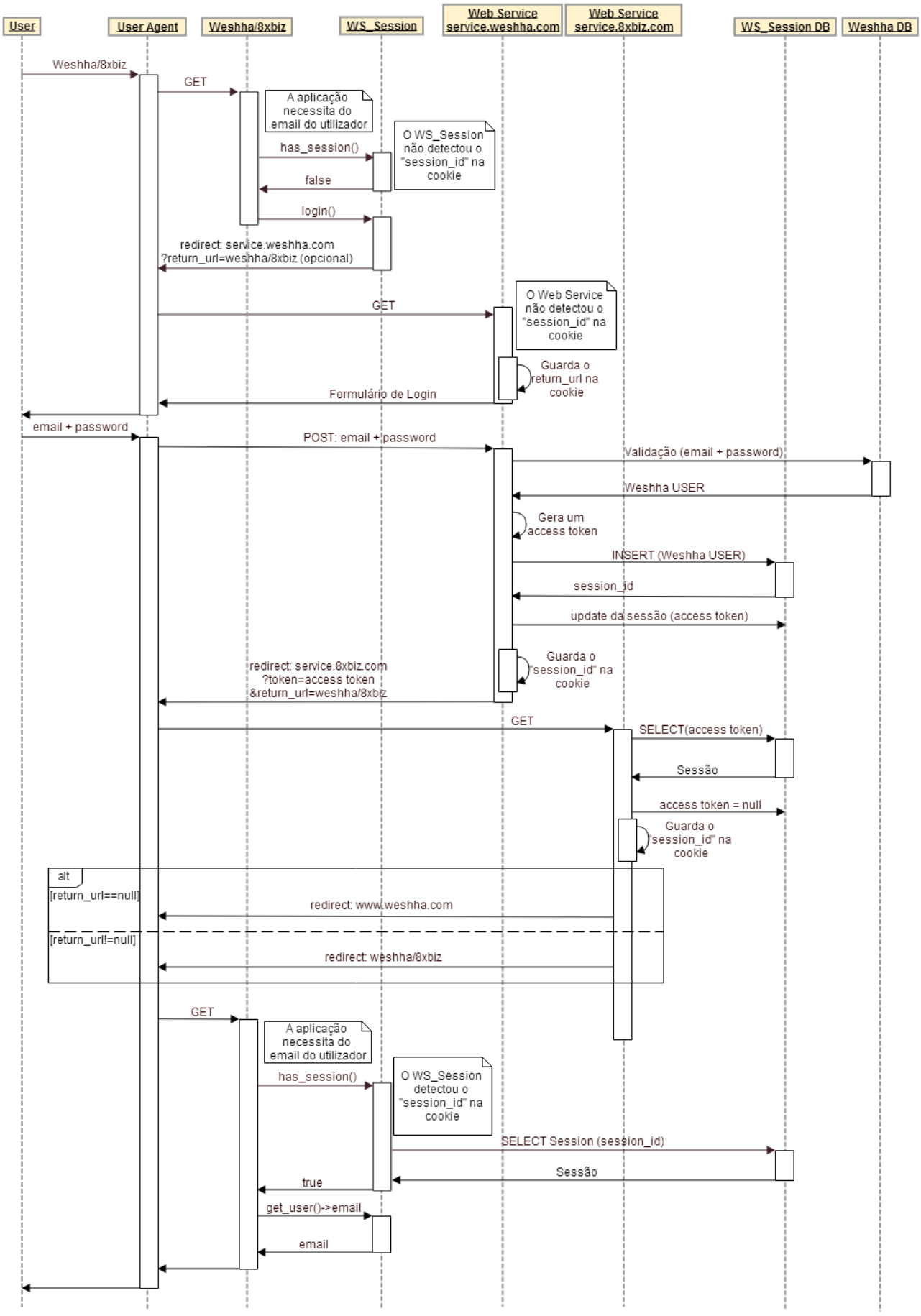


Figura 28 – Diagrama de sequencia para login e autenticação

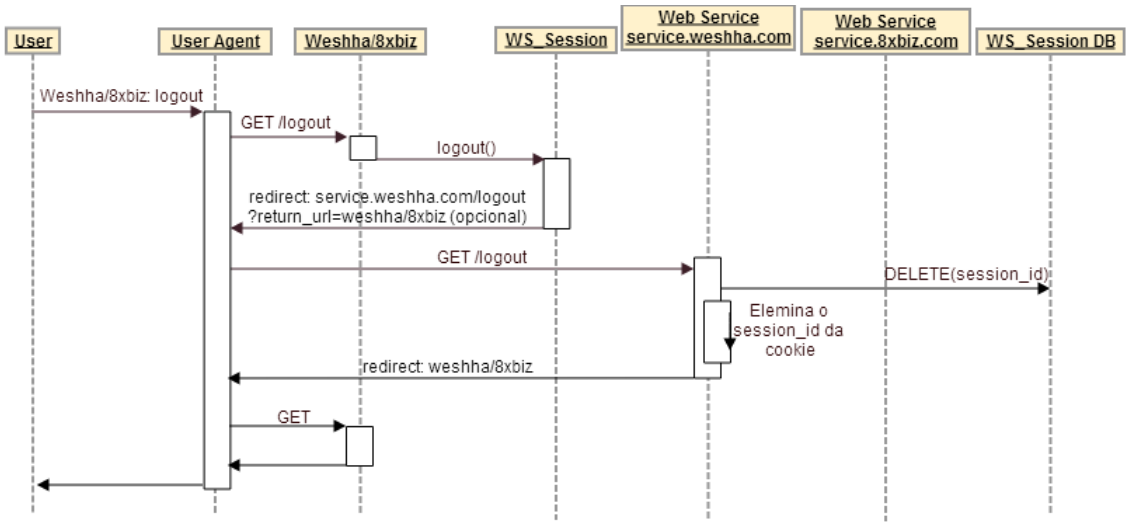


Figura 29 - Diagrama de sequência para o logout

II.2.5.Conclusão

Este foi o único projeto que vi concluído durante o meu estágio na empresa. Provavelmente porque apesar da complexidade do sistema, os seus requisitos eram bastante simples e nunca mudaram, e também porque havia um número bastante limitado de pessoal envolvido neste projeto.

Este foi o projeto que me deu mais satisfação desenvolver, não só por ser o único que ficou concluído, mas porque foi a solução que implementei mais "fora da caixa", ou seja, uma solução que foge aos padrões mas que no entanto conseguiu resolver o problema tal como era pretendido, e também, porque penso que tenha sido aquele projeto em que aprendi mais.

III. CONCLUSÕES E TRABALHO FUTURO

"What you get by achieving your goals is not as important as what you become by achieving your goals."

- Henry David Thoreau

III.1. VISÃO CRÍTICA DO ESTÁGIO

Nesta secção irei criticar de um ponto de vista geral o ambiente da empresa, a sua gestão, metodologias e procedimentos utilizados. Apesar das minhas críticas serem bastante negativas, esta visão crítica não pretende de maneira alguma promover quaisquer ressentimentos ou hostilidades, mas sim demonstrar o resultado de tais decisões que afetaram de certa forma o desenvolvimento do software e o próprio sucesso da empresa. Através desta análise é possível perceber em vários aspetos onde a empresa falhou, e prevenir de certa forma que tais erros voltem a ser cometidos.

Apesar de tudo, tal como irei desenvolver no subcapítulo III.2, o estágio na empresa permitiu-me angariar um vasto leque de competências, experiencias e conhecimentos, pelo que considerei este estágio como algo muito positivo para a minha carreira, e também para aqueles colaboradores que apesar de acabarem por não ser recompensados monetariamente, souberam aproveitar o seu trabalho na empresa.

III.1.1.Ambiente e cultura da empresa

Houve um conjunto de fatores que definiram o ambiente da empresa:

- Foram contratados um *call center*, comerciais, gestores de produto, e *marketeers* sem haver qualquer produto para vender. Isto impôs uma enorme pressão no departamento de produção para desenvolver um produto sob o qual estes pudessem trabalhar.

- O CEO iniciava e descartava projetos com muita frequência e mudava também com muita frequência colaboradores de um projeto para outro, o que atrasava ainda mais o lançamento de um produto, agravava a pressão e desmotivava aqueles que viam o seu trabalho perdido.

- Os *shareholders* que eram quase todos os colaboradores da empresa que investiram nesta o seu tempo de trabalho e também aqueles que investiram capital, queriam ver retornado o seu investimento.

- O CEO e os gestores de produto mudavam conceitos todas as semanas, o que implicava a mudança de requisitos, a mudança de decisões arquiteturais e impossibilitava o planeamento.

- Os gestores de produto tomavam a liberdade de comunicar tarefas diretamente aos programadores, sem consultar o team-leader ou o diretor de produção. Isto levava não só à frustração do diretor de produção, como deixava o programador confuso e pressionado. O programador que acarretava tais tarefas, poderia ter que as desfazer já que não teriam sido aprovadas, o que tornava o desenvolvimento do software ainda mais tardio.

- O CEO adotava um estilo de microgestão, tentava controlar tudo e todos não parecendo confiar nas competências daqueles contratados por ele.

- Foi implícita a promoção de uma certa competitividade entre os projetos que eram desenvolvidos, como "8xbiz vs weshha" ou qual era o gestor de produto que conceituava o produto mais complexo. Um clima competitivo vai contra os princípios de desenvolvimento de software especialmente quando são usados métodos ágeis onde é recomendado um ambiente colaborativo.

Estes foram alguns dos fatores que contribuíram para o ambiente presente na empresa caracterizado sobretudo pelo caos, turbulência, desorganização, desmotivação, confusão e indecisão, competição e muita pressão. Este ambiente gerado na empresa provou ser nocivo ao desenvolvimento de software.

III.1.2.Métodos de desenvolvimento de software

O departamento de produção tentou adaptar os seus métodos de desenvolvimento de software em relação às condições acima referidas impostas pela empresa. Ao introduzir um ciclo de vida semelhante ao *scrum* alguns *team-leaders* começaram a ver "uma luz ao fundo do túnel". No entanto não foi suficiente para prosperar no caos, os gestores de produto continuavam a dar ordens aos programadores sem consultar primeiro o diretor de produção, os *marketeers* tomavam decisões de design, propostas do departamento de marketing eram recusadas pelo departamento de produção e vice-versa. Não haviam entendimentos, e não se conseguia perceber bem quem decidia o quê. A consequência disso era que o software era desenvolvido, alterado, re-desenvolvido, dava "voltas e voltas", mas nunca era acabado.

Bem, a verdade é que os métodos ágeis não eram levados a sério na empresa, nem nunca nos foi comunicado que a empresa iria adotar métodos ágeis para a gestão de projetos, mas sim foi-nos comunicado que seriam usados métodos "atípicos". O que eu tentei fazer na introdução deste relatório foi apenas caracterizar e tentar classificar tais métodos.

Os métodos ágeis permitem prosperar no caos, no entanto estes apoiam-se muito no conhecimento tácito, conhecimento interpessoal, na compreensão, colaboração, na boa vontade da partilha de experiencias e conhecimentos e na comunicação de pessoa para pessoa e estas características na maior parte das vezes, de um ponto de vista geral, não foram verificadas.

III.1.3.Método de recrutamento

Um grande fator que dificultava o desenvolvimento do software era o método de recrutamento usado pela empresa. Como já referi no primeiro capítulo, era efetuado um método de recrutamento "ad infinitum", sem limites, qualquer pessoa com qualificações em informática era

admitida e só depois era decidido qual o projeto em que iria trabalhar. Alguns destes colaboradores eram atribuídos a projetos com pouca ou nenhuma competência para estes, e mesmo aqueles que possuíam competência ainda levavam algum tempo para se tornar produtivos ao mesmo tempo que diminuíam a produção de colaboradores já produtivos, pois, é sempre necessário o treino e acompanhamento inicial de um colaborador para integrá-lo num projeto. Além disso, com o aumento de pessoas num projeto, aumenta também o número de canais de comunicação. Todos os colaboradores a trabalhar numa tarefa devem estar em sintonia (especialmente quando são utilizados métodos ágeis) por isso quando são adicionados mais colaboradores é gasto mais tempo a manter todos estes em sintonia. Esta situação levou-me a confirmar experimentalmente a lei de Brook que diz "a adição de pessoal num estágio tardio de um projeto de software torna-o ainda mais tardio".

Além do facto de o aumento de colaboradores num projeto diminuir a produtividade deste, quando eram adicionados colaboradores com pouca competência e onde se verificava uma diferença significativa de competências, isto diminuía a produtividade dos colaboradores já existentes, não só porque os recém adicionados necessitavam de um treino adicional, mas porque a sua falta de competência desmotivava os colaboradores mais competentes.

O facto de os colaboradores não serem obrigados a estarem presentes, o facto de não serem remunerados e o facto da grande maioria destes serem ainda estudantes universitários, tornava a presença destes e a sua quantidade de trabalho muito imprevisível na empresa, pois se não eram remunerados e acabar o curso na universidade era a sua prioridade, o trabalho na empresa ficaria para segundo plano. Era, pois, quase impossível efetuar um planeamento sem saber a disponibilidade dos colaboradores na empresa, daí a razão pela qual o departamento de produção não pôde adotar nenhum método orientado ao planeamento.

Alguns colaboradores entretanto decidiam deixar a empresa, ou porque não tinham possibilidades de continuar sem remuneração, ou porque tinham outras prioridades como acabar o seu curso universitário, ou então, simplesmente porque deixavam de acreditar que a empresa fosse ter algum futuro. Alguns programadores eram de tal forma pressionados pelos *stakeholders* a mostrar trabalho feito, algo que pudessem ver e experimentar, que o código usado era muitas vezes entregue em "esparguete" e mal documentado. Ora quando estes programadores saíam da empresa, era quase impossível continuar o desenvolvimento dos projetos destes. Muitas vezes o software do qual estes estariam a trabalhar teria que ser descontinuado ou reescrito.

III.1.4.Testes de software e de interface

Não eram realizados testes suficientes para descobrir falhas num estágio precoce do sistema, isto porque a empresa dava muita importância à quantidade e ao tempo de desenvolvimento de

software do que à sua qualidade. Disto resultavam falhas no software e muitas delas só eram descobertas quando o software era lançado ao público.

Quando se desenvolve software para um utilizador comum, um fator importante que é necessário ter em atenção é a interação deste com a aplicação. Não eram efetuados testes de interação, isto em projetos complexos e com muitas funcionalidades pode resultar num software com uma interface para o utilizador muito complicada e difícil de manusear. Alguns gestores de produto responsáveis pela conceção do produto até tinham algumas qualificações na sua área de classificados (no caso do 8xbiz), mas tinham muito pouco ou nenhum conhecimento na área de interação humano-computador e sem testar a interface poderiam acabar por conceituar um produto muito complexo para o utilizador comum.

III.1.5.O CEO

A prioridade do CEO não aparentava ser o desenvolvimento de um sistema simples e estável que pudesse ser lançado para o mercado assim que possível, mas sim aumentar o numero ilimitado de colaboradores na empresa, a adição constante de projetos a serem desenvolvidos e a adição constante de novas funcionalidades nesses projetos. O objetivo do CEO aparentava ser: mostrar trabalho feito visualmente e um aparente crescimento exponencial da empresa de modo a angariar novos *shareholders* e aumentar o investimento dos *shareholders* atuais na empresa. Disto resultavam projetos inacabados e defeituosos sem qualquer possibilidade de lançamento no mercado, impossibilitando assim qualquer retorno de investimento por parte dos *shareholders*.

Tal como eu, alguns colaboradores tentaram abordar a empresa e criticar alguns métodos, procedimentos e requisitos impostos no desenvolvimento do software como não sendo eficazes e até haver estudos que os provam, no entanto a resposta obtida (normalmente pelo CEO) era "não se preocupe que já temos tudo estudado, você só tem que desenvolver o software". O resultado principal do CEO não dar atenção ao pessoal mais competente do departamento de produção resultou que no futuro, todos os colaboradores deste departamento decidiram sair da empresa (inclusive o próprio diretor de produção).

III.2. DESENVOLVIMENTO PESSOAL

III.2.1.Consultoria e Formação

O facto de ser consultado para oferecer soluções técnicas, contribuir para o questionário de testes de aptidão para a entrada de novos colaboradores no departamento de produção e também por dar alguma formação a certos colaboradores da empresa, ajudou-me imenso a consolidar conhecimentos nas áreas das quais fui consultado ou em áreas das quais instruí.

III.2.2.Ferramentas e tecnologias

A responsabilidade de liderar projetos, bem como oferecer soluções técnicas para certos projetos desenvolvidos pela minha equipa ou outras equipas, não só fez com que eu evoluísse o meu conhecimento em certas tecnologias já aprendidas durante o meu período académico, como também permitiu-me aprender novas tecnologias, ferramentas e standards. Como o estilo arquitetural REST, o standard OAuth, o SVN, HTML5, Node.js, websockets, webRTC e mongoDB.

Para oferecer uma solução técnica não quis apenas oferecer uma solução que funcionasse, mas oferecer a melhor solução, e para isso foi necessário conhecer quais as tecnologias que iriam desempenhar o melhor papel para uma determinada solução. Durante o meu período académico pude aprender e angariar bases suficientes em engenharia informática que me forneceram competências e capacidades para facilmente me adaptar a cada uma destas tecnologias, percebê-las e integrá-las para desenvolver uma aplicação ou apresentar uma solução para um problema técnico.

III.2.3.Gestão de projectos e métodos de desenvolvimento

Os métodos praticados na empresa foram muito diferentes daqueles lecionados e praticados durante o meu período académico. Isto permitiu aumentar a minha capacidade de adaptação para um ambiente turbulento e caótico, onde o planeamento e a documentação tinham a menor importância e os requisitos mudavam constantemente.

A minha inexperiência levou-me a cometer vários erros durante todo o meu estágio. Pude observar muitos erros cometidos por outros também. Ciente de certas metodologias e decisões tomadas por mim ou por outros fez-me perceber e experienciar algumas situações que tornavam bastante difícil o desenvolvimento de software, e o que poderia ter sido feito para tornar o ambiente estabelecido para o desenvolvimento de software mais propício a este.

III.2.4.Liderança como exemplo

Sob a minha responsabilidade estavam a Mariana e o João. Como já referi, tal como eu, estes colaboradores não eram remunerados. Aprendi que para obter o máximo de dedicação dos elementos da minha equipa foi necessário dar o meu máximo de dedicação também. Além de trabalhar nas minhas próprias tarefas, tentei sempre acompanhar as tarefas que propunha à minha equipa e manter-me sempre informado de maneira a poder resolver qualquer problema ou a saber responder a quaisquer dúvidas quer da parte da minha equipa, quer da parte do diretor de produção. Tentava sempre também ser o primeiro a chegar e o ultimo a sair da minha equipa, o que penso que acabou por influenciar na quantidade de horas de trabalho por parte dos membros da minha equipa, visto que a minha equipa efetuava horas de trabalho acima da média.

Para aquele team-leader que praticava exatamente o contrário desta prática de liderança como exemplo, verifiquei que o resultado era a perda de entusiasmo e de boa vontade por parte dos membros da sua equipa, o que acabou também por se refletir na dedicação e produtividade dos mesmos.

A utilização da técnica de *pair programming* com o João permitiu que ambos aprendêssemos muito um com o outro, não só a programar mas também a planear, partilhando assim os nossos conhecimentos. Na utilização desta técnica resultaram também menos falhas no software, visto que eram duas pessoas a monitorizar o seu desenvolvimento, o que era muito importante no projeto que estávamos a desenvolver, simplesmente não poderiam haver falhas aqui, pois uma falha poderia implicar também a falha em todas as aplicações que usufruíam deste serviço. Senti também que esta técnica permitiu de certa forma o desenvolvimento mais rápido do software, pois a cada 15 minutos enquanto que um de nós começava a ficar cansado de teclar, o outro estava já "desejando de programar", além disso o *pair programming* tornou também a programação algo mais social, o que acaba por ser também mais motivador.

III.3. PERSPETIVAS PARA O FUTURO

III.3.1.Empresa

Em relação à empresa, no momento em que a decidi deixar, não tinha quaisquer perspetivas de futuro para esta. Deixei então a responsabilidade da continuação do desenvolvimento da aplicação móvel à Mariana, e a continuação do desenvolvimento do serviço de SSO ao João.

III.3.2.Pessoais

Além de aumentar o meu leque de conhecimentos, competências e capacidades, a experiência que adquiri durante o estágio na Ad Infinitum Business motivou-me a começar o meu próprio projeto, participando numa aceleração de *startups* com o objetivo de eventualmente fundar a minha própria empresa. Acreditei que com um número muito mais reduzido de pessoas, mas com as competências certas para tal e com uma ideia inovadora, conseguiria, num tempo de desenvolvimento muito mais reduzido, desenvolver um produto de software com muito mais utilidade em relação aqueles que estavam sendo desenvolvidos na Ad Infinitum Business.

III.3.3.Universidade

A oportunidade de estagiar numa empresa é uma mais valia para o estudante finalista do curso de mestrado. Ao acabar o mestrado em engenharia informática muitos estudantes deparam-se com a indecisão sobre o que fazer depois: continuar a vida académica (ex. tirar doutoramento) ou trabalhar numa empresa. Ao desenvolver um projeto de mestrado ou dissertação, estes estudantes normalmente ficam só com a experiência de trabalhar num ambiente académico. Tendo uma oportunidade de realizar um estágio, estes estudantes podem basear-se na experiência de um ambiente empresarial para a tomada desta importante decisão sobre o que fazer depois. No meu caso, esta experiência motivou-me a começar o meu próprio projeto, noutros casos poderá revelar ao estudante que de facto trabalhar numa empresa é a carreira que este pretende seguir, ou o estudante poderá verificar que se sente mais apto para seguir uma carreira num âmbito académico.

O estágio numa empresa no estrangeiro poderá ser também uma mais valia para o estudante finalista. Muitos engenheiros quando acabam o curso consideram a opção de trabalhar no estrangeiro, não só pela falta de emprego que existe em Portugal, como também porque muitos países, que valorizam os engenheiros portugueses, oferecem um salário maior. Ir para o estrangeiro acarreta certas consequências como uma maior independência, a mudança de cultura, a mudança de língua, etc.. Muitos estudantes poderão acabar por não trabalhar no estrangeiro devido a aversão de certas consequências e ao desconhecido. Aqueles que decidem ir, poderão ir pouco preparados ou até verificar que emigrar não era a melhor opção. Um estágio curricular de

mestrado numa empresa internacional poderia desempenhar um papel muito importante para a preparação e para a tomada de decisão do estudante para migrar para o estrangeiro.

Encontrar uma empresa, tanto na região como no estrangeiro, que estivesse disposta a aceitar-me como estagiário foi muito complicado, visto que cheguei até Novembro conseguindo apenas uma opção: a Ad Infinitum Business. Dito isto, e visto que a opção de estágio poderá ser uma mais valia para o estudante finalista do curso de Mestrado, gostaria de sugerir que fossem estabelecidas mais ligações entre a Universidade e empresas que estejam dispostas a aceitar estagiários, não só a nível regional ou nacional como também a nível internacional, onde poderiam ser utilizadas ferramentas (como o programa ERASMUS) que permitem amenizar custos tanto para as empresas como para os estagiários.

IV. REFERÊNCIAS

[1] "Tablet Shipments Forecast to Top Total PC Shipments in the Fourth Quarter of 2013 and Annually by 2015, According to IDC," IDC Corporate USA, 11 Setembro 2013. [Online]. Available: http://www.idc.com/getdoc.jsp?containerId=prUS24314413. [Accessed 11 Abril 2014].

[2] "Ad Infinitum Business, S.A," 21 Dezembro 2012. [Online]. Available: https://www.racius.com/ad-infinitum-business-s-a/. [Acedido em 10 Agosto 2013].

[3] B. Boehm and T. Richard, Balancing Agility and Discipline: A Guide for the Perplexed, Crawfordsville, Indiana: Addison-Wesley, 2009.

[4] B. T and P. S, "A Survey on Software Development Life Cycle Models," *International Journal of Computer Science and Mobile Computing,* vol. 2, no. 5, pp. 262 - 267, Maio 2013.

[5] "Big Bang Disruption: The innovator's disaster," Accenture, Junho 2013. [Online]. Available: http://www.accenture.com/us-en/outlook/Pages/outlook-journal-2013-big-bang-disruption-innovators-disaster.aspx. [Accessed 16 Maio 2014].

[6] "Development Life Cycle Models," National Instruments Corporation, Junho 2012. [Online]. Available: http://zone.ni.com/reference/en-XX/help/371361J-01/lvdevconcepts/lifecycle_models/. [Accessed 13 Maio 2014].

[7] S. Ambler, "The Agile System Development Life Cycle (SDLC)," Ambysoft Inc., [Online]. Available: http://www.ambysoft.com/essays/agileLifecycle.html. [Accessed 14 Maio 2014].

[8] C. Vennapoosa, "Throwaway Prototyping Model," 14 Janeiro 2013. [Online]. Available: http://www.exforsys.com/career-center/project-management-life-cycle/throwaway-prototyping-model.html. [Accessed 25 Junho 2014].

[9] C. Vennapoosa , "The Evolutionary Prototyping Model," 5 Janeiro 2013. [Online]. Available: http://www.exforsys.com/career-center/project-management-life-cycle/the-evolutionary-prototyping-model.html. [Accessed 25 Junho 2014].

[10] L. Williams, "Integrating Pair Programing into a Software Development Process," IEEE, Raleigh, 2001.

[11] L. Constantine, Constantine on Peopleware, Englewood Cliffs, NJ: Yourdon Press, 1995.

[12] M. Fischer, M. Pinzger and H. Gall, "Analyzing and Relating Bug Report Data for Feature Tracking," 16 Junho 2003.

[13] K. Waters, "The Power of a Whiteboard," 9 Março 2009. [Online]. Available: http://www.allaboutagile.com/the-power-of-a-whiteboard/. [Accessed 5 Maio 2014].

[14] "JIRA," Atallassian, [Online]. Available: https://www.atlassian.com/software/jira. [Accessed 20 Maio 2014].

[15] "Subversion," Free Software Foundation, [Online]. Available: http://directory.fsf.org/wiki/Subversion. [Accessed 20 Maio 2014].

[16] C. Janssen, "What is a Native Mobile App? - Definition from Techopedia," Janalta Interactive Inc., [Online]. Available: http://www.techopedia.com/definition/27568/native-mobile-app. [Accessed 02 Maio 2014].

[17] "native application Definition from PC Magazine Encyclopedia," Ziff Davis Inc., [Online]. Available: http://www.pcmag.com/encyclopedia/term/47651/native-application. [Accessed 02 Maio 2014].

[18] D. Nations, "What is a Web Application?," [Online]. Available: http://webtrends.about.com/od/webapplications/a/web_application.htm. [Accessed 11 Abril 2014].

[19] A. Kosner, "The Appification Of Everything Will Transform The World's 360 Million Web Sites," 16 Dezembro 2012. [Online]. Available: http://www.forbes.com/sites/anthonykosner/2012/12/16/forecast-2013-the-appification-of-everything-will-turn-the-web-into-an-app-o-verse/. [Accessed 11 Abril 2014].

[20] J. Mickens, "Exploiting JavaScript and DOM Storage for Faster Page Loads".*Microsoft Research.*

[21] "Web Storage," 17 Janeiro 2014. [Online]. Available: http://dev.w3.org/html5/webstorage/#introduction. [Accessed 11 Abril 2014].

[22] "Web SQL Database," 18 Novembro 2010. [Online]. Available: http://www.w3.org/TR/webdatabase/. [Accessed 11 Abril 2014].

[23] A. Charland e B. Leroux, "Mobile Aplication Development: Web vs. Native," *Comunications of the ACM,* vol. 54, nº 5, Maio 2011.

[24] "The Future Of Mobile Development: HTML5 Vs. Native Apps," Business Insider, Inc., 23 Abril 2013. [Online]. Available: http://www.businessinsider.com/html5-vs-native-apps-for-mobile-2013-4?op=1. [Accessed 02 Setembro 2013].

[25] P. Viswanathan, "The Pros and Cons of Native Apps and Mobile Web Apps," [Online]. Available: http://mobiledevices.about.com/od/additionalresources/qt/The-Pros-And-Cons-Of-Native-Apps-And-Mobile-Web-Apps.htm. [Accessed 2014 Abril 14].

[26] Standard ECMA-404: The JSON Data Interchange Format, 1 ed., ecma international, 2013.

[27] "JSON," [Online]. Available: http://json.org/. [Accessed 10 Junho 2014].

[28] R. Fielding, "CHAPTER 5: Representational State Transfer (REST)," in *Architectural Styles and the Design of Network-based Software Architectures*, University of California Irvine, 2000.

[29] R. Fielding and R. Taylor, "Principled Design of the Modern Web Architecture," *ACM Transactions on Internet Technology,* vol. 2, pp. 115-150, Maio 2002.

[30] T. Erl, B. Carlyle, C. Pautasso and B. Raj, "SOA with REST," in *Principles, Patterns & Constraints for Building Enterprise Solutions with REST*, Prentice Hall, 2012.

[31] R. Fielding, "CHAPTER 2: Network-based Application Architectures," in *Architectural Styles and the Design of Network-based Software Architectures*, University of California Irvine, 2000.

[32] "Web Services Glossary," 11 Fevereiro 2004. [Online]. Available: http://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/#webservice. [Accessed 30 Abril 2014].

[33] A. Rodriguez, "RESTful Web services: The basics," International Business Machines Corporation, 06 Novembro 2008. [Online]. Available: https://www.ibm.com/developerworks/webservices/library/ws-restful/. [Accessed 29 Abril 2014].

[34] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad and M. Stal, "Architectural Patterns," in *PATTERN-ORIENTED SOFTWARE ARCHITECTURE: A System of Patterns*, vol. 1, Bans Lane, Chichester: John'Wiley & Sons Ltd, 1996, pp. 25-193.

[35] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad and M. Stal, "Model-View-Controller," in *PATTERN-ORIENTED SOFTWARE ARCHITECTURE: A System of Patterns*, vol. 1, Bans Lane, Chichester: John'Wiley & Sons Ltd, 1996, pp. 125-143.

[36] M. Pizka and A. Bauer, "A Brief Top-Down and Bottom-Up Philosophy on Software Evolution".

[37] "Android Developers," google, 30 Maio 2014. [Online]. Available: http://developer.android.com/reference/android/webkit/WebView.html. [Accessed 04 Junho 2014].

[38] G. Grisogono, "5 Best Mobile Web App Frameworks: Sencha Touch," 03 Março 2014. [Online]. Available: http://moduscreate.com/5-best-mobile-web-app-frameworks-sencha-touch/. [Acedido em 14 Abril 2014].

[39] "mgwt - Making GWT work with mobile," [Online]. Available: http://www.m-gwt.com/. [Accessed 14 Abril 2014].

[40] "Android SDK," [Online]. Available: http://developer.android.com/sdk/index.html. [Accessed 14 Abril 2014].

[41] "CodeIgniter User Guide," EllisLab, Inc., [Online]. Available: http://ellislab.com/codeigniter/user-guide/. [Accessed 14 Abril 2014].

[42] "philsturgeon/codeigniter-restserver," [Online]. Available: https://github.com/philsturgeon/codeigniter-restserver. [Accessed 06 Maio 2014].

[43] T. Graser, B. Jostmeyer, N. Lenz, A. Schauberer and W. Schaeberle, "Single Sign On". EUA Patent US 20080276308 A1, 6 Novembro 2008.

[44] E. Eldon, "Single sign-on service OpenID getting more usage," 14 Abril 2009. [Online]. Available: http://venturebeat.com/2009/04/14/single-sign-on-service-openid-getting-more-usage/. [Accessed 14 Abril 2014].

[45] P. Windley, "How Does OpenID Work?," 25 Abril 2006. [Online]. Available: http://www.windley.com/archives/2006/04/how_does_openid.shtml. [Accessed 15 Abril 2014].

[46] "The OAuth 2.0 Authorization Framework," Internet Engineering Task Force, Outubro 2012. [Online]. Available: http://tools.ietf.org/html/rfc6749. [Accessed 21 Abril 2014].

[47] "The OAuth 1.0 Protocol," Internet Engineering Task Force, Abril 2010. [Online]. Available: http://tools.ietf.org/html/rfc5849. [Accessed 22 Abril 2014].

[48] "Web service security principles – OAuth 2.0 example," 30 Outubro 2011. [Online]. Available: http://petermcintyre.com/2011/10/30/web-svc-security-principles-oauth/. [Accessed 14 Abril 2014].

[49] "Home – Login with Amazon Developer Center," [Online]. Available: http://login.amazon.com/. [Accessed 23 Abril 2014].

[50] "Core API - Endpoint reference," [Online]. Available: https://www.dropbox.com/developers/core/docs#oauth2-methods. [Accessed 23 Abril 2014].

[51] E. Hammer, "OAuth 2.0 and the Road to Hell," 26 Julho 2012. [Online]. Available: http://hueniverse.com/2012/07/26/oauth-2-0-and-the-road-to-hell/. [Acedido em 23 Abril 2014].

[52] "Using OAuth 2.0 to Access Google APIs," Google Inc., [Online]. Available: https://developers.google.com/accounts/docs/OAuth2. [Accessed 23 Abril 2014].

[53] "Authentication | LinkedIn Developer Network," LinkedIn Corporation, [Online]. Available: https://developer.linkedin.com/documents/authentication. [Accessed 23 Abril 2014].

[54] "OAuth 2.0," Microsoft Corporation, [Online]. Available: http://msdn.microsoft.com/en-us/library/live/hh243647.aspx. [Accessed 23 Abril 2014].

[55] "How PayPal uses OAuth 2.0," Paypal Inc., [Online]. Available: https://developer.paypal.com/docs/integration/direct/paypal-oauth2/. [Accessed 23 Abril 2014].

[56] "POST oauth2/token," Twitter, Inc., 02 Outubro 2014. [Online]. Available: https://dev.twitter.com/docs/api/1.1/post/oauth2/token. [Accessed 23 Abril 2014].

[57] "OAuth 2.0," [Online]. Available: http://oauth.net/2/. [Acedido em 2014 Abril 24].

[58] "OAuth 2.0 Server PHP," [Online]. Available: http://bshaffer.github.io/oauth2-server-php-docs/. [Accessed 26 Abril 2014].

[59] A. Bilbie, "CodeIgniter-OAuth-2.0-Server," Github, 17 Maio 2013. [Online]. Available: https://github.com/alexbilbie/CodeIgniter-OAuth-2.0-Server. [Accessed 11 Junho 2014].

[60] "What is Joomla?," Open Source Matters, Inc., [Online]. Available: http://www.joomla.org/about-joomla.html. [Acedido em 02 Setembro 2013].

[61] "Kitchen Sink," Sencha Inc., [Online]. Available: http://dev.sencha.com/deploy/touch/examples/production/kitchensink/. [Accessed 14 Abril 2014].

[62] "Definition of: prototyping," LLC. PCMag Digital Group, [Online]. Available: http://www.pcmag.com/encyclopedia/term/49886/prototyping. [Accessed 11 Abril 2014].

[63] R. Fielding, "CHAPTER 1: Software Architecture," in *Architectural Styles and the Design of Network-based Software Architectures*, University of California, 2000.

[64] D. Garlan and M. Shaw, "An introduction to software architecture," in *Advances in Software Engineering & Knowledge Engineering*, vol. 2, A. Vincenzo and G. Tortora, Eds., Singapore, World Scientific Pub Co., 1993, pp. 1-39.

[65] P. Laplante, "What Every Engineer Should Know About Software Engineering," CRC Press, 2007.

[66] G. Huston, "Web Caching," *The Internet Protocol Journal,* Vols. 2, No. 3.

[67] G. Eggleston and M. Hansen, "System having virtual session manager used sessionless-oriented protocol to communicate with user device via wireless channel and session-oriented protocol to communicate with host server". Estados Unidos da América Patent US5771353 A, 23 Junho 1998.

[68] "RFC 1945: Hypertext Transfer Protocol – HTTP/1.0," [Online]. Available: http://tools.ietf.org/html/rfc1945. [Accessed 29 Abril 2014].

[69] M. Heller, "REST and CRUD: the Impedance Mismatch," 29 Janeiro 2007. [Online]. Available: http://www.infoworld.com/d/developer-world/rest-and-crud-impedance-mismatch-927. [Accessed 29 Abril 2014].

[70] T. Berners, "Uniform Resource Identifiers (URI): Generic Syntax," Agosto 1998. [Online]. Available: http://www.ietf.org/rfc/rfc2396.txt. [Accessed 30 Abril 2014].

[71] "Extensible Markup Language (XML) 1.0 (Fifth Edition)," W3C, 26 Novembro 2008. [Online]. Available: http://www.w3.org/TR/REC-xml/. [Accessed 30 Abril 2014].

[72] IEEE Standard Glossary of Software Engineering Terminology, New York: IEEE Standards Board, 1990.

[73] "ABOUT W3C," [Online]. Available: http://www.w3.org/Consortium/. [Accessed 30 Abril 2014].

[74] T. Nelson, "A File Structure for the Complex, the Changing and the Indeterminate," *Proceedings of the 20th National Conference,* p. 84–100, 1965.

[75] D. Flanagan, JavaScript - The Definitive Guide, 5th Edition, O'Reilly, 2006.

[76] "HTML & CSS - W3C," [Online]. Available: http://www.w3.org/standards/webdesign/htmlcss. [Accessed 02 Maio 2014].

[77] "The WebKit Open Source Project," [Online]. Available: http://www.webkit.org/. [Accessed 02 Maio 2014].

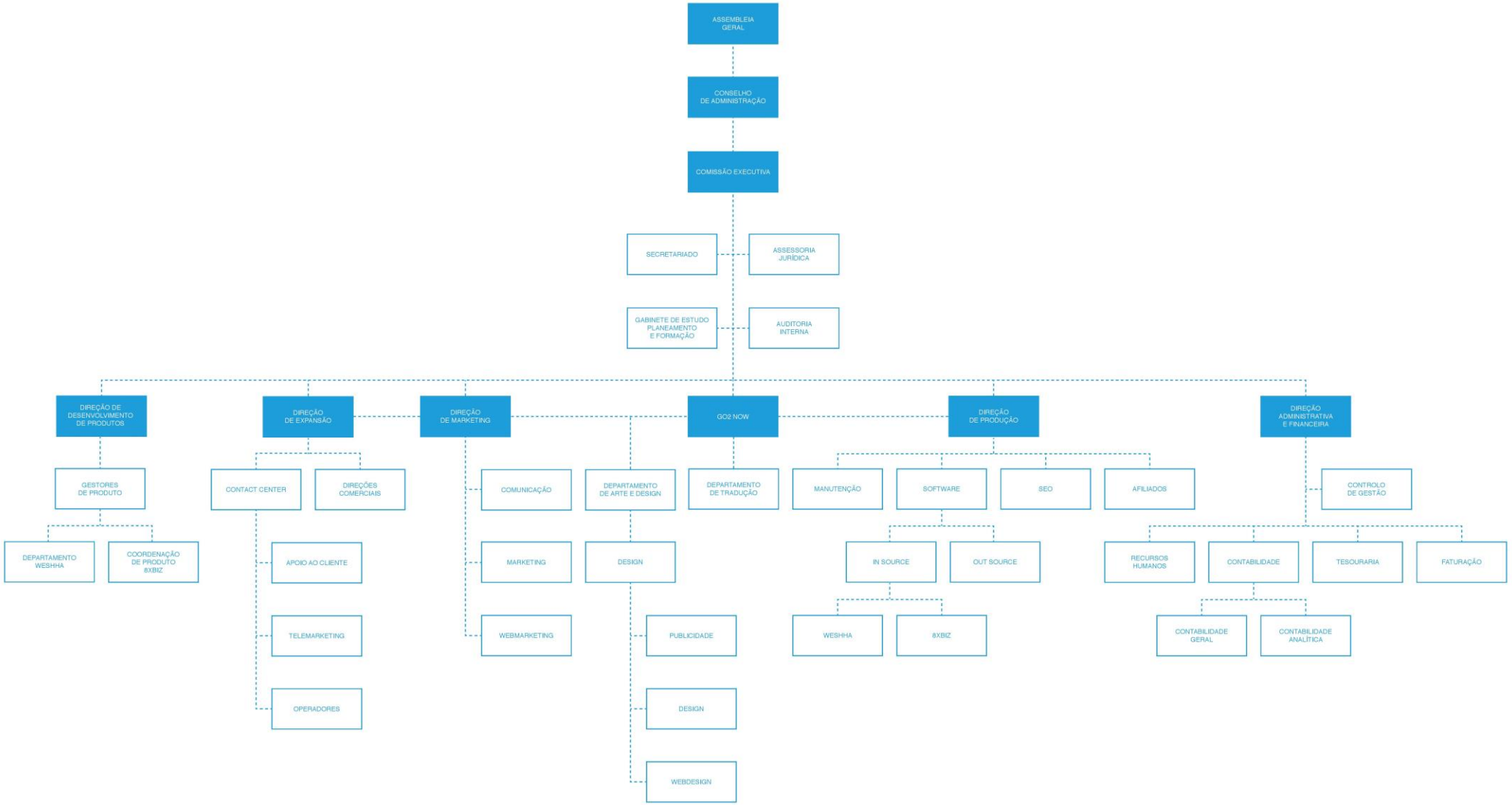
[78] "What is MySQL?," Oracle Corporation, [Online]. Available: http://dev.mysql.com/doc/refman/4.1/en/what-is-mysql.html. [Accessed 05 Maio 2014].

[79] "SELECTING A DEVELOPMENT APPROACH," 27 Março 2008.

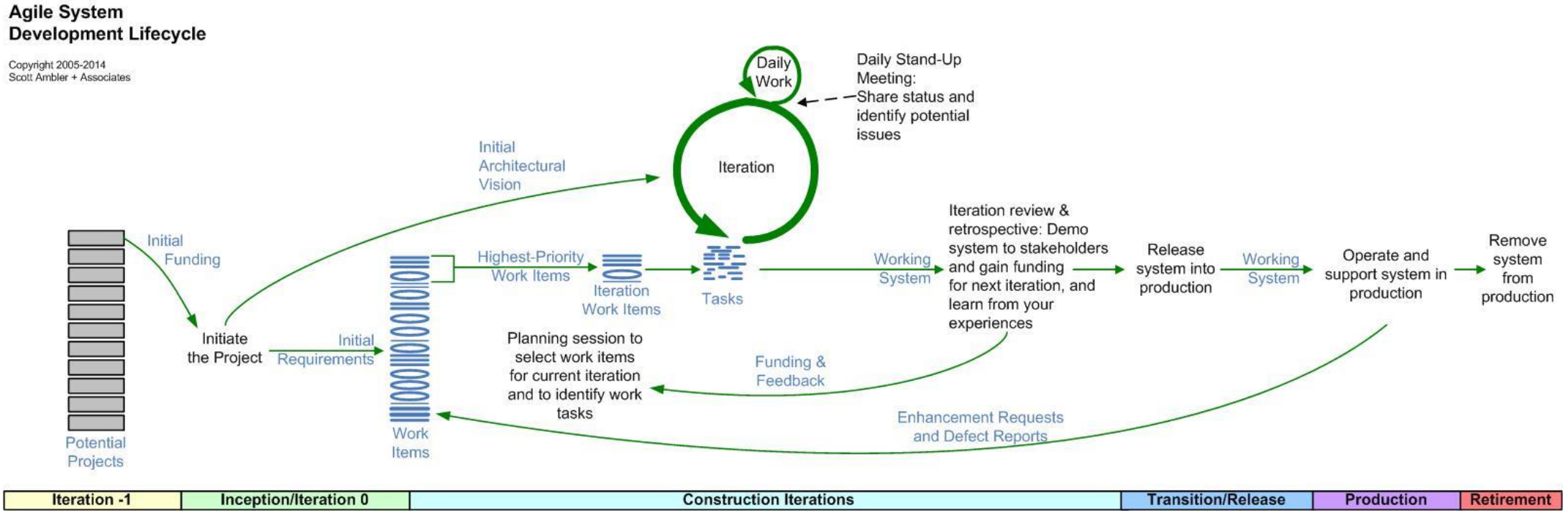
[80] "Prototyping," [Online]. Available: http://www.teach-ict.com/as_a2_ict_new/ocr/A2_G063/331_systems_cycle/prototyping_RAD/miniweb/pg2.htm. [Accessed 15 Maio 2014].

ANEXOS

ANEXO B – ORGANIGRAMA DETALHADO DA EMPRESA



ANEXO C – EXPANSÃO DA FIGURA 5



ANEXO D – PARECER

Por Hugo Marques – Chefe de Produção da Ad Infinitum Business e orientador deste estágio na entidade patronal.

Ponto 1 – Contacto e avaliação pessoal.

O Márcio Oliveira foi um dos primeiros programadores que recebemos na empresa. Na altura da sua integração na equipa de trabalho, havia vários projectos a serem desenvolvidos e foi-lhe dada a escolha sobre qual o projecto que ele se sentiria mais interessado em trabalhar. A sua escolha recaiu sobre a versão mobile do portal web 8xbiz.
Desde logo, o Márcio destacou-se pela sua clareza de ideias e objectividade no trabalho apresentado. O seu nível técnico era elevado, mas sobressaia sobretudo a sua maturidade e avontade em abordar novos desafios assim como a sua capacidade de liderança e organização.
Por isso mesmo, acabou por lhe ser atribuida mais responsabilidades, passando a "team leader", tendo sobre a sua alçada outro projectos tais como todo o sitema de "Web Services" do qual se destacou um sitema global de auticação nas diversas plataformas disponível nas aplicações 8xbiz.

Ponto 2 – Natureza do trabalho

O trabalho proposto como objecto do estágio a ser efectuado, foi uma aplicação móvel que replicasse o portal 8xbiz, nos diferentes equipamentos moveis disponíveis.
Toda a especificação e planeamento foi desenvolvido pelo Márcio, que desde os primeiros dias na empresa, sempre demonstrou uma impressionante autonomia ao encontrar e apresentar diversas soluções técnicas válidas, para os diferentes desafios que este trabalho lhe apresentou. Destaco a sua capacidade de fundamentar e argumentar cada solução, tornando fácil a tarefa de orientação e escolha de rumo a seguir.
Mesmo quando assumiu outros projectos, nomeadamente o web service, não se descuidou da sua tarefa inicial, conseguindo ser produtivo no desenvolvimento, assim como eficaz na orientação das pessoas que entretanto teve à sua responsabilidade.

Ponto 3 – Metodologia do trabalho

Tendo em conta a dimensão "atípica" de todo o projecto que englobava a empresa ad8biz, um dos maiores problemas sempre foi a coordenação e acompanhamento de todo o trabalho a ser desenvolvido. Neste aspecto o contributo do Márcio, foi fundamental coloborando na tentativa de organização e definição da metodologia de trabalho, e sequencia de desenvolvimento do mesmo.
Tal capacidade de organização refletiu-se no concluir de todas as tarefas a que o Márcio se propuz a desenvolver, dentro dos prazos que foram inicialmente propostos. Devo afirmar que das pessoas com quem tive o privilégio de trabalhar, o Márcio é sem duvida dos que mais assertivos conseguem ser em termos de previsão, desenvolvimento e apresentação dentro dos timming inicialmente estabelecidos.

Ponto 4 – Alcance dos Objectivos

O projecto inicialmente proposto ao Márcio, como trabalho de estágio, não foi inicialmente considerado prioritário dentro da estrutura administrativa da empresa. Mas tendo em conta o sucesso da sua implementação, e pronta disponibilidade de demonstração perante sócios e investidóres, passou a ser uma das melhores apresentações em forma de demonstração que foi sendo utilizada como capacidade de toda a empresa, contribuido na altura para a angariação de novos investidores.
É de lamentar, que deu-se mais importância a esse factor (valorização de entrada de investidores) sobre a real qualidade e eficácia do trabalho desenvolvido, uma vez que acabou-se por não potenciar o trabalho desenvolvido (neste caso em particular muito bem desenvolvido) para valorizar a apresentação de mais

aplicações em áreas de actuação cada vez mais vastas. De forma, que parece-me que o trabalho desenvolvido no âmbito deste estágio, atingiu todos os objectivos propostos, faltando apenas a real aposta na integração e promoção do mesmo em vez da desproporcionada ambição da direcção executiva do ir à procura sempre de mais, em vez de apostar no que realmente já tinha sido produzido.
Tendo em conta o que inicialmente previ para este projecto, todos os presupostos de implementação foram tidos em conta, e o trabalho obtido foi até mais interessante do que aquilio que foi primeiramente espectável.

Conclusão

O meu parecer perante este projecto desenvolvido pelo Márcio, só pode ser extremamente positivo. Todo o trabalho foi de encontro ao que foi inicialmente proposto, e todas as dificuldades que naturalmente foram surgindo, não foram de forma alguma um entrave para o Márcio que sempre soube ser inovador apresentando desde logo soluções práticas e eficazes para todas as contrariedades que lhes foram sendo apresentadas.
Pessoalmente lamento, que a globalidade do projecto ad8biz, não conseguiu ser o sucesso que todo este trabalho desenvolvido merecia. Reconheço que o não sucesso da empresa, não se deve à falta de qualidade, sobretudo do trabalho desenvolvido no âmbito deste estágio, mas sim à falta de sensibilidade para potenciar e divulgar o mesmo.