Sistema de Gestão e Auditoria Fiscal na Nuvem

Luís Carlos Pereira Rolo

Orientadores

Professor Doutor Alexandre José Pereira Duro da Fonte

Professor Doutor Eurico Ribeiro Lopes

Dissertação apresentada à Escola Superior de Tecnologia do Instituto Politécnico de Castelo Branco para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Desenvolvimento de Software e Sistemas Interactivos, realizada sob a orientação científica do Professor Adjunto, Doutor Alexandre José Pereira Duro da Fonte, e do Professor Coordenador, Doutor Eurico Ribeiro Lopes, do Instituto Politécnico de Castelo Branco.

Dezembro de 2015

Composição do júri

Presidente do júri

Doutor, Fernando Reinaldo da Silva Garcia

Professor Adjunto na ESTCB

Vogais

Doutor, Francisco Ferreira Francisco

Professor Adjunto na ESTGV

Doutor, Filipe Miguel Bispo Fidalgo

Professor Adjunto na ESTCB

Doutor, Alexandre José Pereira Duro da Fonte

Professor Adjunto na ESTCB

Agradecimentos

O autor desta dissertação pretende efetuar um agradecimento especial aos dois orientadores desta dissertação, o Professor Doutor Alexandre Fonte e o Professor Doutor Eurico Lopes pela elevada disponibilidade para esclarecimento de dúvidas, propostas de melhoria para uma maior credibilidade do trabalho realizado.

O autor pretende agradecer ainda ao contabilista certificado Manuel Salvado Rolo, por facultar dados importantes e confidenciais dos seus clientes que permitiram que o projeto se realizasse com sucesso.

Resumo

A evolução da computação na nuvem tem crescido nos últimos anos e por conseguinte, tem também atraído muitos utilizadores e empresas. De tal forma que, existem cada vez mais aplicações na nuvem e serviços disponibilizados por este tipo de infraestruturas, a custos razoáveis, sobretudo baseados no custo de utilização dos recursos elásticos da nuvem consoante a sua necessidade e quantidade. Isto advém devido estes serem aprovisionados mediante solicitação. A grande capacidade de armazenamento, e o grande poder de processamento, tem potenciado a que arquiteturas como *Data Warehouses* sejam hospedadas na nuvem, originando o surgimento de novas aplicações analíticas. A necessidade de envio de informação sobre a atividade fiscal dos contribuintes ou empresas para os portais do governo de forma automática e integrada, bem como, a ausência de um local centralizado para a sua consulta tem-se revelado um problema dos *softwares* de gestão, bem como, para os contabilistas e técnicos oficiais de contas, recentemente apelidados de contabilistas certificados. Para colmatar estas lacunas, este projeto tem como objetivo a apresentação de uma proposta de arquitetura para um sistema de gestão e auditoria fiscal na nuvem, que permite o armazenamento contínuo da informação fiscal num *Data Warehouse* para posterior consulta numa aplicação *web,* como também o envio automático da mesma para os portais das entidades governativas nacionais via *web services*.

Palavras chave

Auditoria, Data Warehouse, fiscal, nuvem, sistema analítico.

Abstract

The evolution of cloud computing has grown in the last years and therefore has also attracted many users and enterprises. Due to the increasing of cloud applications and services offered by this kind of infrastructure, at a reasonable cost, especially based on pay per use model of cloud elastic resources, and quantity provisioned. The large storage capacity and high processing power, have boosted architectures as data warehouses to be hosted on the cloud, that has led to the emergence of new analytical applications. The need for sending fiscal information of taxpayers and enterprise busineness to the government's portals automatically and the absence of a unique and centralized location led to an issue to the software managements and for the accountants as well. To fill these gaps, this project has as main goal the presenting of an architecture for a management and tax audit system on the cloud that enables the continuous storage of fiscal information in a data warehouse, allowing the access of such information in a web cloud application and submission of tax data to the government's portals automatically through web services.

Keywords

Audit, Data Warehouse, tax, cloud, analytical system.

Índice geral

Índice de figuras

Lista de tabelas

Lista de abreviaturas, siglas e acrónimos

AJAX – Asynchronous Javascript and XML

AT – Autoridade Tributária

AWS – Amazon Web Services

BI – Business Intelligence

CAPSI – Conferência – Associação Portuguesa de Sistemas de Informação

CSV – Comma-Separated Values

CRM – Customer Relationship Management

DNS – Domain Name System

DM – Data Mart

DW – Data Warehouse

ETL – Extract, Transform and Load

HTTP – Hypertext Tranfer Protocol

IaaS – Infrastructure as a Service

IRC – Imposto sobre o Rendimento das Pessoas Coletivas

IRS – Imposto sobre o Rendimento das Pessoas Singulares

IVA – Imposto sobre o Valor Acrescentado

MDX – Multidimensional Expressions

OLAP – Online Analytical Processing

PaaS – Platform as a Service

SaaS – Software as a Service

SAFT-PT – Standard Audit File for Tax Purposes – Versão Portuguesa

SLA – Service Level Agreement

SOAP – Simple Object Access Protocol

SQL – Structure Query Language

SSL – Secure Socket Layer

TIC – Tecnologias de Informação e Comunicação

TWDI – The Data Warehousing Institute

XML – eXtensible Markup Language

WAN – Wide Area Network

WSDL – Web Services Description Language

1. Introdução

A computação na nuvem tem vindo a ganhar posição, na Internet, com a oferta de vários serviços e recursos de computação, para indivíduos e empresas, colmatando uma lacuna existente nos sistemas de informação ao nível do aumento e adição de capacidades e recursos de computação sob solicitação sem a necessidade de investimento numa nova infraestrutura (Eric Knorr, 2008).

A potencial diversidade de oferta de serviços é outra característica intrínseca à computação na nuvem, a qual tem permitido o surgimento de novas aplicações baseadas neste paradigma computacional, originando novos modelos de negócios (Sanjay Rishi, 2015). Daí que faça sentido criar e desenvolver novos projetos na nuvem como sistemas analíticos que possibilitem tirar partido da grande capacidade de recursos elásticos destas infraestruturas a custos reduzidos, como será demonstrado nesta dissertação.

A computação na nuvem é um sistema bastante automatizado, o que permite solucionar problemas de comunicação entre aplicações (Hélder Borges et al, 2012). Por exemplo, permite estabelecer uma ponte de ligação entre aplicações de *software* de gestão e portais *web* das instituições governamentais competentes, pois atualmente o processo de submissão da documentação fiscal gerada por estes programas para estes portais é realizada periodicamente de forma manual, e não integrada. Soma-se a impossibilidade de se poder efetuar o rastreamento da informação fiscal, porque após a sua submissão não se tem conhecimento ao certo para que efeito esta servirá no futuro, bem como, por quanto tempo é mantida nestas organizações no final da sua utilização.

Nesse contexto, o objetivo deste projeto prende-se na apresentação de uma proposta arquitetónica que visa criar um mecanismo automático de submissão de documentação fiscal para as entidades correspondentes, bem como, guardar essa informação num repositório central de dados, tornando-a mais útil por forma a auxiliar na tomada de decisões mais inteligentes.

As tecnologias adotadas na implementação do sistema são a base de dados *MySQL* para o armazenamento do *Data Warehouse (DW)*, e o *Talend Open Studio* (Talend, 2015), esta última uma ferramenta de integração de dados para realizar o processo de extração, transformação e carga. Por outro lado, o *Google App Engine* (Google App Engine, 2015) é usado para efetuar a integração do *DW* com a nuvem, bem como para criar a aplicação de gestão e auditoria fiscal, no sentido de mostrar a informação do *DW* em cubos ou num *Dashboard* e ainda possibilitar o envio de dados fiscais para entidades governamentais via *web services*.

1.1 Definição do tema

A presente dissertação permitiu desenvolver um projeto direcionado para a melhoria dos sistemas fiscais do governo, centralizando a informação num único local através da criação de uma aplicação analítica na nuvem, a qual guarda a informação fiscal dos contribuintes num repositório central e envia automaticamente os correspondentes dados para os portais do governo de forma periódica ou programada.

Apesar de terem sido encontrados vários obstáculos à concretização do presente trabalho, designadamente a instalação dos certificados da Autoridade Tributária, a falta de informação atualizada por parte deste organismo e a integração do repositório central de dados com o *Google App Engine*, foi possível ultrapassar todos estes problemas e concluir o projeto com sucesso, tendo sido obtidos resultados bastante satisfatórios.

1.2 Contribuição do trabalho

Este projeto pretende contribuir para a integração de vários serviços de contabilidade e gestão, nomeadamente com a criação de uma aplicação analítica na nuvem para guardar informação fiscal dos contribuintes e com a automatização do processo de submissão dos dados para os portais do governo, estabelecendo um elo de ligação e comunicação entre serviços.

Este mecanismo pretende ainda reduzir o tempo e o esforço do trabalho diário das autoridades fiscais em tarefas de agregação de informação para efeitos fiscais, análise e consulta, auxiliando igualmente no combate à evasão fiscal.

Por isso, como principais beneficiários do sistema analítico desenvolvido, destacam-se os seguintes:

- **Entidades do governo –** recebem a informação periodicamente para efeitos fiscais no portal da Autoridade Tributária de forma automatizada;
- **Contabilistas, técnicos oficiais de contas e contribuintes em geral** – estes utilizadores podem analisar, consultar e navegar sobre a informação fiscal pretendida de forma fácil e rápida, o que os ajuda no processo de tomadas de decisão mais inteligentes.

De salientar que, o acesso à plataforma por parte dos contribuintes e contabilistas certificados poderá ser realizado por diferentes dispositivos com ligação à Internet, inclusivamente por dispositivos móveis.

Foi também apresentada uma comunicação na 15ª edição da CAPSI (CAPSI, 2015), no dia 2 de Outubro de 2015, no ISCTE-IUL em Lisboa, com o título Sistema de gestão e auditoria fiscal, em que se procurou demonstrar publicamente a contribuição deste trabalho com a proposta de solução arquitetónica exposta, no sentido de com isso resolver o problema atual de submissão de dados fiscais de forma eletrónica ou a sua consulta.

1.3 Cronograma

O desenvolvimento do presente trabalho teve a duração de aproximadamente de um ano letivo, tendo sido previamente realizado o devido planeamento das várias fases necessárias à sua concretização, as quais são apresentadas de forma cronológica, na Figura 1.

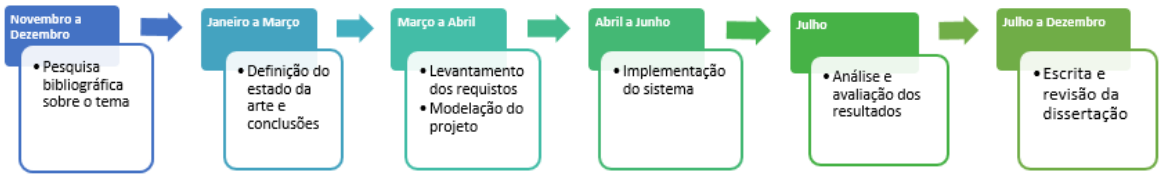

Figura 1 - Cronograma do trabalho

Conforme é possível verificar na Figura 1, o projeto foi dividido em seis fases. A primeira fase consiste na pesquisa bibliográfica inicial sobre o tema a desenvolver para o projeto, nomeadamente sobre a computação na nuvem.

A segunda fase tem como objetivo realizar um levantamento do estado da arte identificando trabalhos de outros autores relacionados com o presente trabalho, e com base no estudo realizado, tirar um conjunto de conclusões sobre o mesmo.

A terceira fase consiste no levantamento e compreensão dos requisitos do sistema analítico, os quais posteriormente darão origem à modelação do projeto, incluindo a especificação dos requisitos funcionais e não funcionais do sistema.

Na quarta fase dá-se início à implementação do sistema, em que se procurou concretizar os requisitos definidos e identificados na fase anterior.

Já a quinta fase é a etapa onde é feita a avaliação dos resultados ao projeto desenvolvido, tendo sido tiradas ilações ao sistema analítico com base na informação obtida.

Por último, a sexta fase consiste na escrita e revisão da dissertação.

1.4 Estrutura do Relatório

O presente relatório está dividido em sete capítulos.

O primeiro capítulo faz uma breve abordagem sobre a computação na nuvem com o objetivo de dar a conhecer o âmbito deste trabalho.

No segundo capítulo é realizado um enquadramento teórico com o projeto realizado, identificando e descrevendo os conceitos da computação na nuvem, bem como o seu surgimento no mundo dos sistemas de informação. Poderão ser vistos ainda neste capítulo os conceitos de *Data Warehouse*, das mais valias na sua migração para nuvem, sendo no fim realizada uma comparação das principais ferramentas *PaaS* e uma descrição da ferramenta escolhida para a concretização deste trabalho.

O terceiro capítulo apresenta o caso de estudo sobre o qual este projeto se baseou, com a descrição do problema encontrado e com a apresentação de uma solução ao mesmo, designadamente, a criação de um sistema de gestão e auditoria fiscal na nuvem.

O quarto capítulo apresenta a modelação do sistema tendo em conta a proposta de solução identificada no capítulo anterior.

No quinto capítulo é apresentado a implementação do sistema, revelando os principais aspetos da criação do sistema de gestão e auditoria fiscal.

O sexto capítulo demonstra os resultados alcançados após a concretização do sistema, bem como as principais dificuldades encontradas durante a sua implementação.

Por último, o sétimo capítulo pretende apresentar uma análise critica sobre o projeto desenvolvido, acompanhada das principais conclusões, e dos aspetos fortes e fracos da solução. No final deste capítulo é ainda apresentado uma proposta de melhoria futura ao sistema analítico desenvolvido no decurso do projeto.

2. Enquadramento teórico

2.1 Introdução

Neste capítulo são abordados os principais conceitos teóricos que dão suporte e fundamentação à realização do presente trabalho de desenvolvimento de um sistema analítico na nuvem. Daí que, em primeiro lugar seja apresentada uma breve resenha histórica sobre a computação na nuvem e a sua evolução ao longo do tempo.

Em seguida, são definidos os conceitos de computação na nuvem, a sua arquitetura, as suas características, os papeis que podem ser desempenhados na nuvem, os seus modelos de serviços, bem como os tipos de nuvens que existem, indicando suas vantagens e desvantagens desta infraestrutura.

Depois é efetuada uma abordagem sobre o conceito de *Data Warehouse,* evidenciando as suas principais características no sentido de avaliar se esta arquitetura poderá ser usada e disponibilizada na nuvem.

Seguidamente, são apresentadas as principais ferramentas de *Platform as a Service* (PaaS) disponíveis no mercado atual, e que foram alvo de análise, no sentido de se identificar os tipos e quantidade de serviços que fornecem e oferecem aos seus consumidores, bem como estabelecer uma comparação entre ambas identificando as suas semelhanças e diferenças.

No final é efetuada uma análise crítica, retirando algumas sínteses e conclusões sobre os temas referidos ao longo do capítulo.

2.2 Evolução da computação na nuvem

Na última década, muito se tem falado em computação na nuvem e do seu elevado crescimento, mas o que isto significa? Na realidade este conceito não é novo. A sua evolução começou nos anos 50 com a introdução da computação de *mainframe* em que múltiplos utilizadores eram capazes de aceder a um computador central através de terminais *dumb,* isto é, através de terminais sem capacidade de processamento (Maximiliano Neto, 2014).

Aproximadamente nos anos 70, o conceito de máquinas virtuais foi criado. Tornou-se possível executar um ou mais sistemas operativos em simultâneo num ambiente isolado. Com isso, a partilha de *mainframes* permitiu a múltiplos ambientes computacionais residirem no mesmo ambiente físico (Maximiliano Neto, 2014).

Até aos anos 90, era comum as empresas de telecomunicações, apenas oferecerem ligações de ponto-a-ponto de acesso a dados. Contudo, nos anos 90, passaram a oferecer ligações de rede privadas com qualidade de serviço idêntica à dos serviços dedicados oferecidos até então, mas a um custo mais reduzido. Em vez de se construírem mais infraestruturas físicas dedicadas para permitirem acomodar mais utilizadores e mais ligações, as empresas de telecomunicações adotaram uma abordagem em que os utilizadores obtêm um acesso partilhado à mesma infraestrutura física (Maximiliano Neto, 2014).

Depois, nos anos 2000, com o crescimento comercial das redes e em particular da Internet, as empresas tiveram que começar a repensar os seus modelos de negócio, pois por mais que este se rentabilizasse, as empresas ainda necessitavam de ter um plano para o negócio sobreviver a longo prazo. Então, verificaram que poderiam providenciar um modelo de serviço de entrega de soluções *web* e recursos usáveis. A *Salesforce* foi pioneira no fornecimento de aplicações para empresas num único portal *web* (Matt Smith, 2014).

Em 2002, a Amazon disponibilizou a *Amazon Web Services*, fornecendo aos utilizadores a capacidade de aceder a armazenamento, computação e a algumas aplicações através da Internet, que os levou em 2006 a lançar a *Elastic Compute Cloud* (EC2), que dá a possibilidade aos programadores de alugar um determinado espaço nos seus servidores, no sentido de estes poderem armazenar e executar as suas aplicações (Matt Smith, 2014).

Em 2009, a *Microsoft* e a *Google*, disponibilizaram aplicações e interfaces de programação (*APIs*) para os seus consumidores e outras empresas sob a forma de serviços simples e acessíveis (Matt Smith, 2014), isto é, começaram a oferecer serviços de computação, serviços de rede, gestão, armazenamento aos seus clientes bem como, possibilitaram a integração com algumas das suas aplicações nativas.

2.3 Computação na nuvem

Na generalidade, a computação na nuvem pode ser vista como um tipo de computação que reside essencialmente na partilha de recursos de *hardware* e *software* onde diferentes serviços como servidores, armazenamento de informação ou aplicações são disponibilizados através da Internet (Beal Vangie, 2015).

A computação na nuvem oferece tecnologia, serviços, e aplicações que são semelhantes aos da Internet e transforma-os num utilitário *self-service* (Sosinsky, 2011). Reside em dois conceitos fundamentais:

Abstração – encapsula o sistema de implementação dos utilizadores e programadores. As aplicações são executadas em sistemas físicos não especificados,

sendo a informação armazenada em locais desconhecidos, e a sua administração terceirizada por administradores de sistemas. O acesso à nuvem é realizado pelos utilizadores de forma ubíqua (Sosinsky, 2011);

Virtualização – os sistemas na nuvem são virtualizados através de agrupamento e partilha de recursos, sendo fornecidos e escalados consoante a necessidade (Sosinsky, 2011).

2.4 Arquitetura

A arquitetura da computação na nuvem é divida em três tipos de modelos, como pode ser visto na Figura 2:

- Modelos de implantação;
- Modelos de oferta de serviços;
- Atributos de serviços.

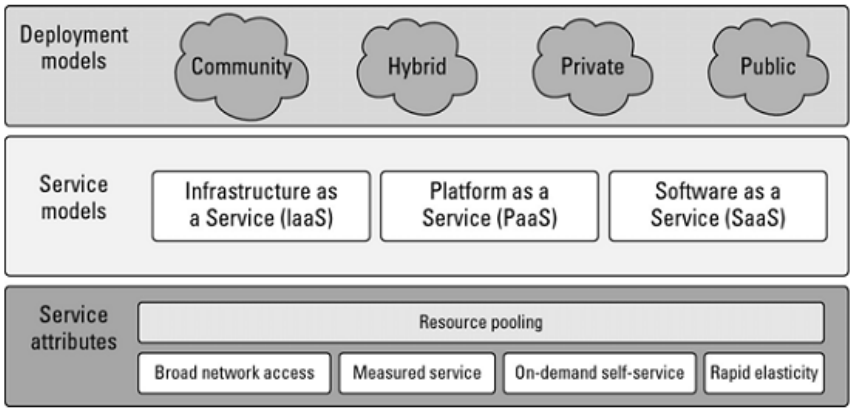

Figura 2 - Arquitetura da computação na nuvem (Fonte: Sosinsky, 2011)

2.4.1 Modelos de implantação

Os modelos de implantação da computação na nuvem definem um propósito na natureza para que a nuvem, na literatura inglesa a *cloud,* foi construída. Assim sendo, os principais modelos de implantação disponibilizados pelos grandes fornecedores deste género de infraestruturas são:

- **Nuvem Pública (*Public Cloud*) –** é propriedade e é operada por várias empresas que oferecem acesso rápido e recursos de computação a outras organizações ou indivíduos, automaticamente e em demanda, tais como, aplicações, armazenamento e outros serviços (IBM, 2015). É baseada no modelo pagamento consoante o uso, isto é, o pagamento pelos serviços disponibilizados por este modelo é cobrado consoante a sua utilização e o seu custo aos seus utilizadores (Parsi et al, 2013);
- **Nuvem Privada (*Private Cloud*) –** é propriedade e é operada por uma única empresa que controla os recursos virtualizados e outros serviços automáticos que são personalizados e utilizados por vários grupos empresariais (IBM, 2015). Pode ser interna *(on-premise),* alojada no próprio centro de dados das organizações que requerem este género de serviços, garantindo uma maior

proteção e segurança ou pode ser externa *(off-premise)* alojada num fornecedor do serviço, assegurando uma maior privacidade em relação às nuvens públicas devido aos riscos associados de partilha de recursos físicos (Parsi et al, 2013);

- **Nuvem Comunidade (Community Cloud) –** é uma nuvem com o objetivo de atender um determinado propósito ou função (Sosinsky, 2011). É partilhada entre as organizações que detêm interesses e requisitos semelhantes, assim como os seus custos. Pode ser alojada e gerida interna, externamente, ou mesmo por fornecedores *third-party* (Parsi et al, 2013), e tem a vantagem de tirar partido de politicas de segurança comuns e de escalabilidade superior em relação a nuvens privadas (Ken, 2012);
- **Nuvem Híbrida (*Hybrid Cloud*) –** pode combinar múltiplas nuvens, privada, comunidade, ou pública, mantendo as suas identidades originais. No entanto, estão unidas como uma unidade (Sosinsky, 2011), originando um único serviço, podendo oferecer flexibilidade, controlo e segurança combinada dos vários modelos de implantação.

A Figura 3 indica a localização das diferentes implantações consoante os tipos de nuvem referidas anteriormente.

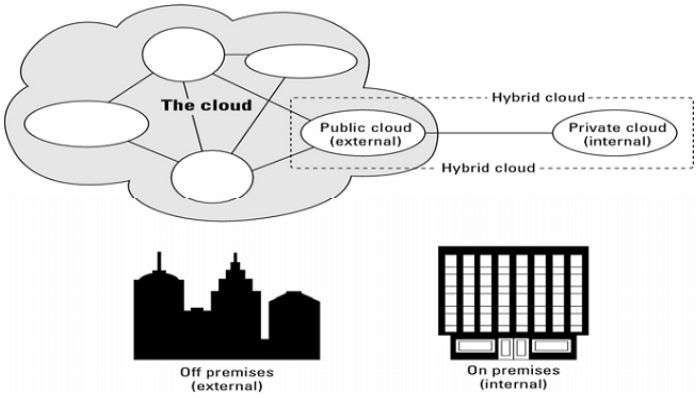

Figura 3 - Diferentes tipos de implantação de nuvens (Fonte: Sosinsky, 2011)

2.4.2 Modelos de oferta de serviços

Também conhecido como *cloud stack*, os principais modelos para oferta de serviços em nuvem que surgem na literatura são:

***Software as a Serv*ice (SaaS) -** camada onde os utilizadores simplesmente fazem uso do *web browser* para aceder a software desenvolvido por outros, mantendo a oferta como um serviço na web. Neste nível, os utilizadores não têm controlo ou acesso à camada inferior da plataforma ou da infraestrutura, que está a ser usada para alojar o *software*. O sistema de gestão de relacionamento com o cliente, *Customer Relationship Management* (CRM), da *Salesforce,* e o *Gmail* do Google são exemplos de adoção do modelo *SaaS* (Armbrust et al., 2009);

***Platform as a Service* (PaaS) –** é a camada onde as aplicações são desenvolvidas usando um conjunto de linguagens de programação e ferramentas que são

suportadas e fornecidas pelo fornecedor de *PaaS*. Fornece aos programadores um elevado nível de abstração que permite que se foquem exclusivamente no desenvolvimento de aplicações. Permite ainda aos programadores fornecer uma aplicação personalizada sem ter que definir e manter a infraestrutura subjacente. Isto é, tal como o modelo *SaaS*, os utilizadores não têm controlo ou acesso à camada de infraestrutura subjacente usada para alojar as aplicações ao nível de PaaS. A *Google App Engine* e a *Microsoft Azure* são exemplos proeminentes da adoção do modelo de serviço *PaaS* (Boniface, et al., 2009);

Infrastructure as a Service **(IaaS) -** camada mais baixa onde os utilizadores podem usar recursos de computação como bases de dados, poder de processamento, memória, e armazenamento de um determinado fornecedor *IaaS* no sentido de poderem implementar e executar as suas aplicações. Ao contrário do modelo de oferta de serviço *PaaS*, o *IaaS* permite aos utilizadores dar acesso à infraestrutura subjacente através do uso de máquinas virtuais que automaticamente são escaladas para cima ou para baixo, ou seja, a possibilidade de aumentar a capacidade de *hardware* ou *software* através da adição ou remoção de recursos computacionais (David Beaumont, 2014). O *IaaS* dá mais flexibilidade do que o *PaaS*, e permite que o utilizador possa implementar qualquer tipo de *software* no sistema operativo. No entanto, a flexibilidade tem custos acrescidos, pois os utilizadores desta camada são responsáveis por atualizar o sistema operativo deste nível. *Amazon Web Services EC2* e *S3* são exemplos de *IaaS* (Abraham et al, 2009).

De forma a compreender melhor como este modelo de camadas funciona na prática, na Figura 4 está representado um esquema em forma de hierarquia indicando as suas principais características e a identificação dos diferentes tipos de utilizadores de cada camada.

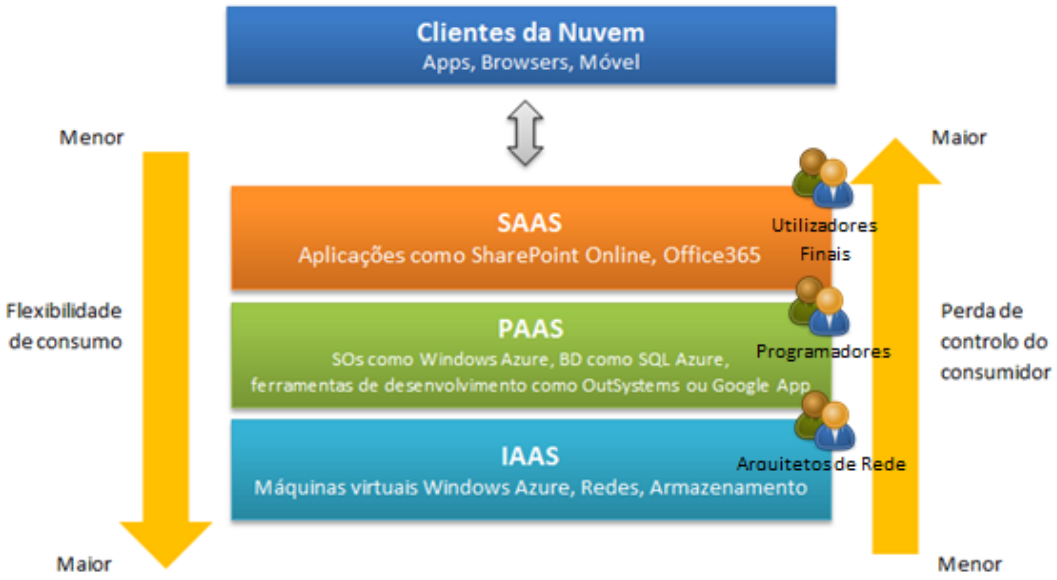

Figura 4 - Modelos de serviços

(Adaptado de: Amit Kumawat, 2013 e Jan Clercq, 2011)

Note-se e de acordo com a Figura 4, que quanto mais elevada for a camada de modelos de oferta de serviços, menor é a flexibilidade de consumo de recursos e maior é a perda de controlo dos mesmos por parte dos consumidores, e vice-versa.

Uma outra forma mais pormenorizada de distinguir as diferentes camadas é através de uma grelha, pois permite evidenciar os principais fatores diferenciadores entre cada um dos modelos descritivamente como por exemplo, as características mais importantes, as suas grandes vantagens, mas sobretudo esclarecer sobre quando se deve usar um determinado modelo em detrimento de outro. Daí que, na Tabela 1 abaixo, esteja representada uma comparação entre os vários tipos de modelos de oferta de serviços da nuvem.

Tabela 1 - Comparação dos modelos de oferta de serviços cloud (Fonte: Khurana e Verma, 2013)

M. Serviço	Paradigma	Características	Termos Chave	Vantagens	Quando não usar
IaaS	Infraestrutura como um ativo	Independente de plataforma Custos da infraestrutura são partilhados Acordos de *SLAs* Pagamento pelo que se usa; Auto escalável	Computação em grelha; Instância de computação *Cloudbursting* *Multi-tenant* Computação *Pooling* de recursos	Eficiência de negócio e produtividade depende largamente da capacidade do vendedor Potencialmente maior a longo-prazo Requer novas medidas de segurança	Quando o orçamento de capital é maior do que o orçamento operacional
PaaS	Aquisição de licença	Consome infraestrutura da nuvem; Atende a métodos de gestão ágil de projetos	Solução stack	Centralização Requer novas medidas de segurança	N/A
SaaS	*Software* como um ativo (empresas e consumidores)	*SLAs*; UI alimentado por aplicações leves Componentes da nuvem Comunicação via APIs Modular Interoperabilidade Sem estado	*Thin-client* Aplicação cliente-servidor	Centralização dos dados Requer novas medidas de segurança	N/A

Conforme se observa na Tabela 1, existe grande diferença nas características de cada um dos modelos, o que é normal, visto que estes têm diferentes propósitos conforme foi descrito anteriormente. Por exemplo, o modelo PaaS foi desenhado para funcionar como um *middleware* que tira partido dos recursos do IaaS, derivado à sua capacidade elástica auto escalável necessária para desenvolver as aplicações que depois serão disponibilizadas na camada superior, ou seja, no SaaS de forma modular.

2.4.3 Atributos de serviço

Os atributos de serviço pertencem à camada mais baixa da arquitetura da nuvem e correspondem a um conjunto agrupado de recursos elásticos de hardware e software, prontos a usar, mediante solicitação. Estes podem ser escalados para cima ou para baixo consoante a necessidade dos utilizadores ou de forma automática, o que obriga a que estes sejam medidos com base no seu uso. O processo *self-service* disponibilizado pelos fornecedores deste género de serviços deve ser simples e amigável (Basant Singh, 2010).

Verifica-se, portanto, que em termos de arquitetura, a computação na nuvem dispõe de três camadas principais que em conjunto fornecem um modelo completo, integrado e centralizado de soluções e serviços aos seus clientes, possibilitando a terceirização de serviços de TI, o que faz com que empresas tecnológicas, como empresas de *outsourcing* se sintam cada vez mais atraídas por este tipo de paradigma, e vejam na nuvem potencial para projetar e escalar os seus negócios pelo mundo de forma fácil, segura e a preços mais competitivos (TI Inside, 2015).

2.5 Papéis na computação na nuvem

Os papéis desempenhados na nuvem são vários. Existem tipos de utilizadores que interagem com a computação na nuvem de forma distinta, conforme pode ser observado na Figura 5.

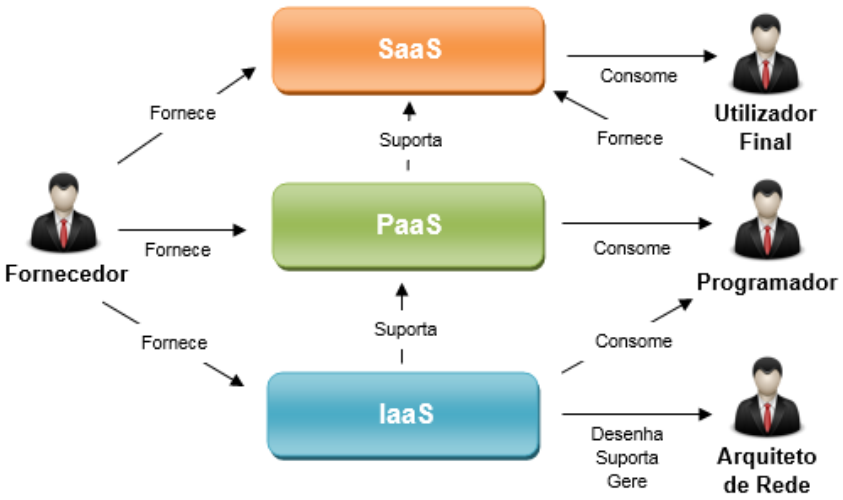

Figura 5 - Papéis na Computação na nuvem

(Adaptado de: Flavio, Leonardo e Javam, 2010)

Na Figura 5, o fornecedor é responsável por disponibilizar, gerir e monitorizar toda a estrutura da nuvem, deixando os programadores, arquitetos de rede e utilizadores finais sem esse tipo de responsabilidades. Para isso, o fornecedor fornece os três modelos de serviços. Os arquitetos de rede desenham, gerem e dão suporte à infraestrutura aos programadores. Os programadores utilizam os recursos disponibilizados e fornecem serviços aos utilizadores finais.

A organização em diferentes papéis ajuda, assim a definir as funções e responsabilidades que cada utilizador tem consoante o seu papel na nuvem. De referir ainda que alguns utilizadores podem assumir vários papéis, podendo partilhar ou aumentar o número de funções e responsabilidades permitidas na nuvem.

2.6 Características da Computação na Nuvem

Existem cinco características principais da computação na nuvem, a saber:

- **Serviço *On-demand*** – um consumidor poderá adquirir recursos de computação consoante a sua necessidade sem ter que requerer interação com um humano para fornecer o serviço (Third Nature, 2012);
- **Acessibilidades em rede** – as capacidades fornecidas poderão estar disponíveis na rede usando *software* padrão do cliente independente do *hardware* (Third Nature, 2012);
- ***Pooling* de recursos** – os recursos de computação são alocados de um pool partilhado de uma forma transparente para os consumidores do serviço. Os recursos podem ser dinamicamente resignados *on-demand* e não tem dependências de localização física (Third Nature, 2012);
- **Elasticidade** – a capacidade deve ser dinamicamente aprovisionada para que possa ser aumentada ou reduzida *on-demand*, e devem aparecer como se a pool de recursos fosse ilimitada (Third Nature, 2012);
- **Serviço medido** – os recursos devem ser entregues num modelo *pay as you go* onde os consumidores são cobrados financeiramente consoante o uso e consumo dos recursos aprovisionados. Os consumidores têm a capacidade de monitorizar e controlar o uso dos recursos, tornando o processo de pagamento transparente (Third Nature, 2012).

De fato, a computação na nuvem tem um conjunto de características e funcionalidades que permitem apresentar e oferecer um serviço de qualidade aos seus clientes. A elevada versatilidade e elasticidade dos seus recursos permite que cada vez mais as organizações se sintam atraídas pelos seus serviços, confiando o seu negócio na nuvem, disponibilizando – o para o mundo, no sentido de poderem crescer mais rapidamente.

2.7 Vantagens e Desvantagens

Em termos de vantagens na adoção da computação na nuvem por parte das empresas ou a título individual, os motivos são inúmeros e bastante atrativos, pois a sua confiabilidade em planos complexos para recuperação de desastres, deixam de ser preocupação para as empresas que usam este tipo serviço (Salesforce, 2015).

Os fornecedores de serviços da computação na nuvem disponibilizam constantemente atualizações automáticas de *software* e os recursos podem ser rapidamente fornecidos, a qualquer altura e em qualquer quantidade de forma agrupada e num sistema *multi-tenant*, sendo dinamicamente alocados ou realocados consoante necessidade. O sistema de medições permite que estes sejam medidos, auditados, e reportados para o cliente (Sosinsky, 2011).

Os negócios baseados na computação na nuvem, usam apenas o espaço de servidor que necessitam, o que potencia a diminuição da pegada de carbono e consumo de energia (Salesforce, 2015).

Outra grande vantagem é o custo reduzido das redes na nuvem, pois estas operam com maior eficiência, o que diminui bastante os custos, para além de fornecerem escalabilidade, balanceamento de carga e *failover*, tornando-a altamente confiável diminuindo drasticamente a necessidade de investimento em infraestruturas e recursos como armazenamento e processamento (Sosinsky, 2011).

Por outro lado, existem também algumas desvantagens da computação na nuvem. Por exemplo, e um pouco contraditório, são os custos elevados de computação associados à sua utilização a médio/longo prazo, sobretudo se estes não forem aprovisionados eficientemente. Especialistas dizem para ter sempre o controlo dos custos em mente, porque os erros mais comuns que costumam ocorrer são pedidos de poder de computação excessivos e desnecessários, ou a não desativação de recursos quando estes não estão a ser usados, ou a não utilização de ferramentas de monitorização dos ciclos de computação desperdiçados, ou ainda fazer pensar aos programadores que esses ciclos são gratuitos (Boulton, 2015).

Outro problema é o fato de as aplicações disponibilizadas na nuvem sofrerem com a latência inerente das ligações *WAN*, e se houver necessidade de transferir grande quantidade de informação, a computação na nuvem pode não ser a melhor opção (Sosinsky, 2011).

Além disso, a computação na nuvem é um sistema sem estado, como a Internet, e solicitações do tipo *HTTP: PUTs,* ou *GET*, podem perder-se se houver uma desconexão arquitetónica entre as mensagens de pedido-resposta (Sosinsky, 2011), porque a computação na nuvem é baseada no paradigma cliente-servidor, tirando partido das comunicações pedido-resposta entre clientes e servidores sem estado. Isto é, não necessita de um cliente para estabelecer a ligação, ao invés disso, verifica o seu pedido como sendo uma transação independente, respondendo a este. Assim, o cliente não é afetado se o servidor for abaixo e voltar ao seu estado operacional, nem

tem que se preocupar com o estado do sistema, o que faz com que este seja simples, mais robusto e escalável (Marinescu, 2013).

Contudo, a maior preocupação é sem dúvida a privacidade e a segurança da informação, uma vez que os dados navegam e são armazenados em sistemas fora do controlo do utilizador, o que aumenta o risco de interceção e prevaricação por parte de terceiros (Sosinsky, 2011). Isto embora, seja usado como contra-argumento, o fato da gestão dos serviços e da infraestrutura na nuvem seja realizada por equipas especializadas, bem treinadas, e sejam adotados nos centros de dados protocolos de segurança bastante rígidos.

Goste–se ou não, a verdade é que a computação na nuvem tem vindo a crescer nos últimos tempos e a ganhar novos adeptos, quer para criar novos modelos de negócio, quer para gerar novo tipo de aplicações e com isso atingir rapidamente o *time-to-market*, muito por culpa das grandes vantagens que apresentam, o que permite reduzir drasticamente o custo de desenvolvimento dos projetos e os custos inerentes com as infraestruturas que os suportam, o que indica claramente que este é e será o futuro das *TIC* nos próximos anos.

2.8 Data Warehouse

Um dos serviços que poderá tirar partido dos recursos de computação, poder de processamento e armazenamento da nuvem é sem dúvida o *Data Warehouse*, um repositório central de dados, utilizado para armazenar informações relativas às atividades de uma determinada organização de forma histórica e consolidada. É visto como um sistema de dados orientado por assuntos, integrados, não voláteis e variantes no tempo (Bill Inmon, 2009).

Tem por objetivo integrar dados de diferentes fontes e formatos, não sendo construído para suportar o processo funcional ou operacional, de uma empresa, mas sim fornecer um meio para facilitar o uso da informação (Ralph Kimball et al, 2002).

2.8.1 Arquitetura

Tradicionalmente, os *Data Warehouses* convencionais analisam dados estruturados e transacionais, que estão contidos em bases de dados relacionais, conforme se pode verificar na Figura 6, ilustrada abaixo.

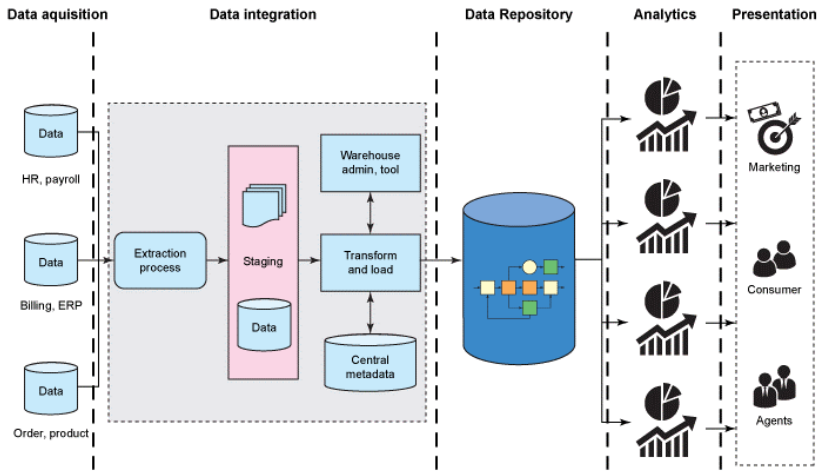

Figura 6 - Arquitetura Data Warehouse (Fonte: Sandip Chowdhury, 2014)

De acordo a Figura 6, cada camada tem uma função particular:

- **Camada de aquisição -** componente que consiste na busca de várias fontes de informação em vários sistemas, como recursos humanos, finanças e faturação (Sandip Chowdhury, 2014).
- **Camada de integração -** componente que consiste na integração e transformação dos dados (*ETL*) provenientes das fontes de informação para o repositório central da arquitetura, o *Data Warehouse* (Sandip Chowdhury, 2014).
- **Camada de repositório -** armazena a informação numa modelo multidimensional no sentido de aumentar o desempenho e extensibilidade (Sandip Chowdhury, 2014).
- **Camada de análise –** vistas com dados no formato de um cubo para tornar as questões de análise dos utilizadores mais fácil de responder (Sandip Chowdhury, 2014).
- **Camada de apresentação –** aplicações ou portais que dão acesso a um conjunto de utilizadores. As aplicações consomem os dados através de páginas web ou móveis que são definidos por ferramentas de relatórios ou serviços web (Sandip Chowdhury, 2014).

Como o processo de *Data Warehousing* é um processo longo e contínuo, a migração para esta arquitetura apresenta-se com o cenário ideal, pois assim poderá tirar-se partido dos recursos elásticos da nuvem, como já foi referido, para além de garantir a segurança e replicação da informação.

Além disso, é sabido que o custo de desenvolvimento deste tipo de soluções é bastante elevado, bem como a sua manutenção, pelo que se forem disponibilizados na nuvem os custos com este género de serviços tende a baixar drasticamente, sobretudo na manutenção dos mesmos.

Depois, a informação consolidada, guardada na nuvem poderá ser usada por aplicações analíticas desenvolvidas nesta plataforma, possibilitando o seu acesso em

qualquer lugar, o que ajudaria nas tomadas de decisão das organizações, dando origem ao designado *business intelligence* na *cloud*.

2.9 *Cloud Business Intelligence*

Nos últimos tempos, o *cloud Business Intelligence (BI),* tem sido posicionado como a próxima evolução do *Business Intelligence*, porque a nuvem fornece armazenamento escalável, poder de computação, e recursos elásticos. A sua adoção ainda tem sido algo lenta, mas o seu interesse tem vindo a aumentar bastante. Por exemplo, recentemente, num inquérito feito pelo *The Data Warehousing Institute* (TDWI), a maioria dos inquiridos que já usam a nuvem ou estão a pensar em usá-la para fins analíticos, usam-na numa combinação de implementação pública, privada e hibrida (Fern Halper, 2014).

A combinação de maior velocidade, menor custo e da natureza elástica da nuvem potencia o desenvolvimento ágil, pois atualmente, projetos pontuais ou aqueles que têm um benefício claro são difíceis de justificar, devido ao processo de orçamento de capital, tempo e esforço para fornecer recursos. Mas, num ambiente em nuvem estes projetos podem ser construídos a um custo menor, mesmo que não aconteça o benefício esperado, porque a acontecer, os projetos BI, podem ser rapidamente desativados sem custos afundados em matéria de licenças de *hardware* ou *software*.

Além disso, o desenvolvimento ágil permite criar novos *workloads* de produção com um maior controlo sobre os custos e a definição de prioridades.

No fundo, a nuvem é uma grande parte do futuro do *Business Intelligence*, pois esta oferece várias vantagens em termos de custo, flexibilidade de implementação, disponibilidade e rapidez de execução. É muito relevante para implementações de projetos BI típicos, que exijam elevados requisitos de infraestrutura, volumes de carga e dados imprevisíveis e envolvam alto investimento inicial, custos com desenvolvimento e manutenção.

2.10 Ferramentas PaaS

Conforme referido anteriormente, as ferramentas *Platform as a Service* (PaaS), permitem desenvolver aplicações na nuvem, que depois serão disponibilizadas na camada superior para os utilizadores finais. Assim sendo, esta seção pretende ilustrar as ferramentas *PaaS* fornecidas pelas principais empresas com oferta de serviços na nuvem.

Convém salientar que todas estas ferramentas diferem no tipo e quantidade de serviços que oferecem, pelo que não se vislumbra fácil escolher entre uma em detrimento de outra, pois todas elas têm vantagens e desvantagens.

Contudo, abaixo poderá ser vista uma análise exaustiva sobre as ferramentas *PaaS*, com a descrição das suas principais características e objetivos.

2.10.1 OutSystems

É uma plataforma *open-source* e de alta produtividade que torna fácil a criação, implementação e gestão de aplicações *web* e móveis, de forma responsiva auxiliando o IT em entregar soluções de negócio de forma rápida (OutSystems, 2015).

A sua arquitetura foi desenhada com o intuito de ser uma infraestrutura com elevado desempenho, escalabilidade, e disponibilidade, tal como indica a Figura 7.

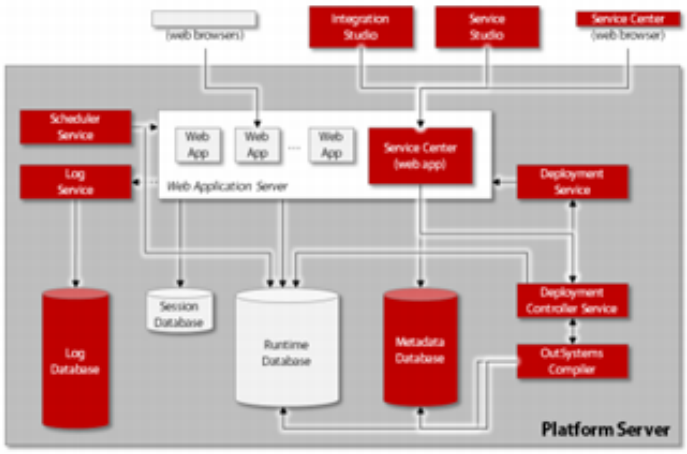

Figura 7 - Arquitetura da plataforma OutSystems (Fonte: OutSystems, 2014)

As ferramentas que interagem com o servidor aplicacional são o *Service Studio*, ambiente de desenvolvimento de aplicações web e móveis, no modelo *RAD* (*Rapid Software Development*) e o *Integration Studio*, que possibilita a integração de sistemas desenvolvidos nas linguagens *.NET* ou *Java*. Permite ainda a integração com vários tipos de bases de dados como *MySQL, SQLServer, Oracle ou DB2* (OutSystems, 2014).

As principais características deste PaaS são:

- Desenvolvimento rápido de aplicações;
- Integração com vários tipos de sistemas existentes;
- Programação única para todos os dispositivos.

Apesar dos preços não serem disponibilizados publicamente, sabe-se que a versão gratuita apenas contempla o ambiente de desenvolvimento das aplicações, na modalidade *Community*, para uso pessoal ou experimental, e na modalidade *Enterprise*, com a subscrição de um *trial* de 30 dias gratuitos, no sentido de fomentar a aprendizagem e a interação com a plataforma. No entanto, após esse período o custo de licenciamento pode ser algo considerável para empresas de pequena ou média dimensão (OutSystems, 2014).

2.10.2 AWS Elastic Beanstalk

O *AWS Elastic Beanstalk* é um serviço de fácil utilização para implementação e escalabilidade de aplicações ou serviços *web* desenvolvidos em *Java, .NET, PHP, Node.js, Python, Ruby e Docker*, em servidores aplicacionais como *Apache, Nginx, Passenger e IIS*. Gere automaticamente os detalhes de implementação do aprovisionamento da capacidade, balanceamento de carga. A escalabilidade é

automática e fornece a monitorização do estado de saúde das aplicações (Amazon Web Services, 2015).

A Figura 8 apresenta um exemplo da arquitetura do *AWS Elastic BeanTalk*, bem como, os componentes que interagem com o ambiente.

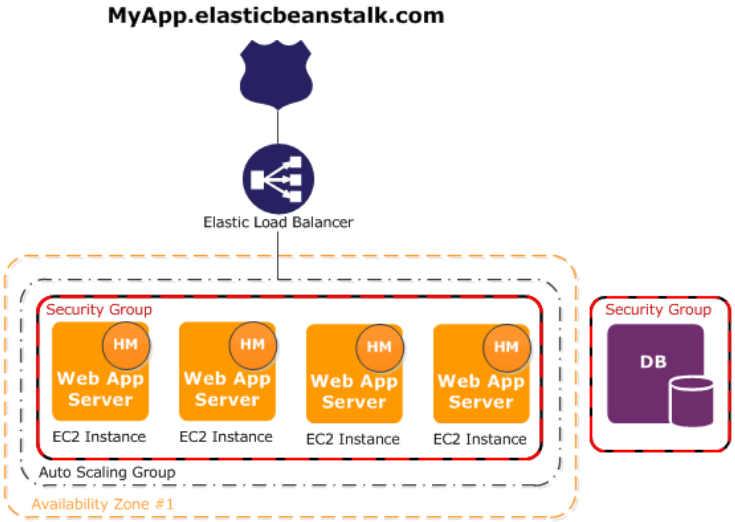

Figura 8 - Arquitetura AWS E. Beanstalk (Fonte: AWS E. Beanstalk, 2010)

As suas principais características são:

- Escalabilidade automática;
- Balanceamento de carga;
- Tolerância a falhas;
- Segurança.

No entanto, esta ferramenta tem a limitação do escalonamento das aplicações ser baseado em métricas. Os serviços disponibilizados pela Amazon podem também exigir uma maior gestão da sobrecarga do que outras opções *PaaS*.

Em termos de preçário, este serviço é gratuito nos primeiros 12 meses, para novas contas na *Amazon Web Services*, passando para um custo mensal de cerca de 30€/mês após esse período nos pacotes mais acessíveis.

2.10.3 Microsoft Windows Azure

É uma junção de *PaaS* e *IaaS* completa, em que apresenta uma oferta abrangente na nuvem onde as empresas podem trazer a sua própria infraestrutura e usá-la para alojar aplicações e VMs. Ao contrário de outros, o *Windows Azure* fornece uma nuvem híbrida, tornando os dados mais seguros das empresas (Regalix, 2014).

Para além disso, o *Azure* é aberto e flexível, isto é, suporta qualquer sistema operativo, linguagem, ferramenta ou *framework*, desde *Windows* a *Linux, SQL Server* a *Oracle,* e linguagens de programação como *C#* a *Java* (Microsoft Azure, 2015).

Dentro da sua estrutura é ainda possível verificar uma plataforma de integração de serviços, designada de *Azure Platform Services*, ilustrado na Figura 9.

Figura 9 - Arquitetura Windows Azure (Fonte: Microsoft Azure, 2015)

Este serviço dispõe de quatro componentes essenciais:

- ***Windows Azure*** **–** fornece computação e armazenamento escalável para aplicações de utilizadores ou outros serviços da plataforma Azure;
- **Serviços *.NET* –** coordena diferentes tipos de autenticação perante diferentes esquemas de segurança e oferece uma infraestrutura de serviços distribuída para aplicações baseadas na *cloud* ou locais;
- **Serviços *SQL* –** providência armazenamento da informação para as aplicações executadas na nuvem, e em centros de dados;
- **Serviços *LIVE* -** permite aos utilizadores coordenarem e partilharem informação entre os seus dispositivos.

As suas principais características são:

- Serviço *PaaS* e *IaaS* numa única plataforma;
- Suporte a múltiplas linguagens, bases de dados e *frameworks;*

Contudo, apesar de ser uma plataforma bastante completa, tem a desvantagem de ter um portal de administração demasiado minimalista (Dan Sullivan, 2014).

Em termos de custos, o *Windows Azure* disponibiliza uma versão *trial* válida por 30 dias para os utilizadores poderem experimentar, sendo o custo mensal calculado consoante o número e tipo de recursos desejado para o negócio após esse período.

2.10.4 Salesforce (force.com e heroku)

O *force.com* é uma das plataformas *PaaS* do Salesforce. Trata-se de uma *framework* de desenvolvimento de aplicações. Facilita o trabalho dos programadores na implementação de aplicações *multi-tenant* que possam ser alojadas e integradas no Salesforce (Chuck Schaeffer, 2014).

O desenvolvimento no *force.com* é realizado usando uma linguagem de programação não padronizada, designada *Apex*. É uma linguagem do estilo C, sendo uma *pseudo* combinação que se assemelha a *Java* e *SQL*. As ferramentas especializadas desta plataforma são projetadas para a camada de apresentação, camada de aplicação e modelo de dados (Chuck Schaeffer, 2014). É diferente das outras soluções *PaaS,*

pois o seu foco é exclusivamente para aplicações de negócio (Chuck Schaeffer, 2014), como por exemplo, aplicações *CRM.*

As suas principais características são:

- **Segurança de classe mundial –** presta um serviço de segurança de classe mundial em todos os níveis (Avandel, 2015);
- **Confiança e transparência –** fornece um serviço transparente, em tempo-real (Avandel, 2015);
- **Verdadeiro *multi-tenancy* –** máxima escalabilidade e desempenho consistente para os seus clientes com uma verdadeira arquitetura *multi-tenancy* (Avandel, 2015)*;*
- **Recuperação completa de desastres –** protege a informação dos clientes executando um serviço em múltiplos e dispersos centros de dados com extensivas cópias de segurança, arquivos e capacidades de *failover* (Avandel, 2015);
- **Alta disponibilidade –** tem uma infraestrutura de elevada disponibilidade, comprovada e aplicativos de *software* (Avandel, 2015).

Contudo, e apesar de ter bastante potencial, esta plataforma *PaaS* apresenta bastantes desvantagens, como por exemplo, a ausência de soluções *Business Intelligence*. Os clientes que procurem armazenamento de dados, mineração, processamento analítico *online* ou análise preditiva, necessitam de adquirir soluções de terceiros, pois o *core business* do Salesforce é ainda a comercialização de soluções *CRM*. Também o fato da empresa ter o seu próprio plano de manutenção dos servidores, costuma causar problemas de acessibilidade das aplicações durante esse período (Chuck Schaeffer, 2014).

Mas, o Salesforce adquiriu recentemente outro *PaaS* que suporta o desenvolvimento de aplicações em *Ruby, Python, Java, Scala, Cloture e Node.js, o* Heroku. Esta plataforma fornece ambientes de computação abstrata chamados *dynos*, que são recipientes virtualizados do estilo Unix e executam processos em ambientes isolados. Os *dynos* respondem a solicitações *HTTP*, designados *dynos web* ou respondem a pedidos por tarefa numa fila, chamados *worker dynos* (Dan Sullivan, 2014).

O Heroku funciona melhor com aplicações que se encaixam na metodologia *Twelve Fator App,* metodologia de uso de formatos declarativos para automação e configuração, no sentido de minimizar o tempo e os custos de desenvolvimento de projetos aos programadores. É possível integrar com aplicações de terceiros, normalmente designadas de *addons*, que também estão disponíveis como serviços dentro desta ferramenta (Dan Sullivan, 2014).

As suas principais características são:

- É ideal para implementações rápidas;
- Encaixa-se numa ampla gama de aplicações distribuídas.

No entanto, tem a limitação dos custos dos próprios *addons* variarem consoante o número ou carga entre as aplicações (Dan Sullivan, 2014).

Em termos de preços, o plano do force.com mais básico custa cerca de 22€ por mês. Já em relação ao outro serviço *PaaS*, o Heroku pode-se verificar que apenas é livre de custos para o primeiro *dyno*, pois a adição de mais *dynos/workers* custa cerca 31€/mês (Reed Law, 2015).

2.10.5 Red Hat OpenShift

O *Red Hat OpenShift* é o serviço *PaaS* do *Red Hat*, que permite aos programadores desenvolver, alojar, e escalar aplicações na nuvem, como se pode verificar na Figura 10. Com o *Open Shift* é possível escolher várias ofertas, incluindo opções *online*, *on premise*, e ainda projetos *open source* (Open Shift, 2015).

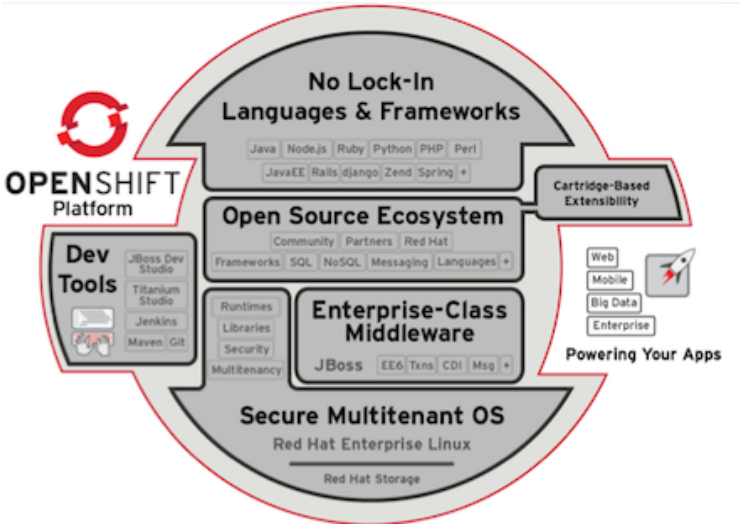

Figura 10 - Arquitetura OpenShift (Fonte: Chris Mayer, 2012)

O *Red Hat OpenShift* é baseado em aplicações *open source* e oferece uma ampla variedade de linguagens de programação, bases de dados e componentes (Dan Sullivan, 2014). Este *PaaS* é altamente personalizável e é oferecido em três formas:

- ***OpenShift Online*** - serviço de alojamento baseado na nuvem;
- ***OpenShift Enterprise*** - *PaaS* privado executado num centro de dados;
- ***OpenShift Origin*** - plataforma aplicacional de alojamento aberto.

Esta ferramenta automatiza tarefas de administração de sistemas, tais como, aprovisionamento de servidores virtuais, configuração, dimensionamento e suporte de repositórios Git para a gestão do código fonte (Dan Sullivan, 2014).

As suas principais características distinguem-se como sendo:

- Um grande número de opções de componentes que medem a pilha de aplicações *front-end* e *back-end* (Dan Sullivan, 2014);
- Os programadores podem interagir com *OpenShift* através de uma consola web, linha de comandos ou através de um ambiente de desenvolvimento integrado (Dan Sullivan, 2014).

No entanto, aplicações não Git podem exigir passos adicionais à sua implementação (Dan Sullivan, 2014), o que se revela uma limitação.

Em termos de preçário, este *PaaS* torna-se muito atrativo, pois tem dois planos grátis (*Free* e *Bronze)*, mas com algumas limitações. Contudo são úteis para pequenas ou médias empresas ou mesmo para quem esteja a iniciar a usar esta ferramenta.

2.10.6 Cloud Foundry

Cloud Foundry é uma ferramenta *open source* que oferece uma escolha de serviço na nuvem, *frameworks* de desenvolvimento, e serviços de aplicativo. Torna o processo de construção, implementação e testes de aplicações mais rápido e fácil. Como é um projeto de código aberto, disponibiliza uma variedade de serviços na nuvem privada e instâncias na nuvem pública (Pivotal, 2015).

Aos componentes do *Cloud Foundry*, representados na Figura 11, é incluído um mecanismo de execução de aplicações em modo *self-service,* isto é, um mecanismo automático de gestão do ciclo de vida das aplicações *web*. Além disso, é disponibilizada ainda uma *interface* de linha de comandos (CLI) programável, e um serviço de integração com ferramentas de desenvolvimento para facilitar o processo de implementação das aplicações (Pivotal, 2015).

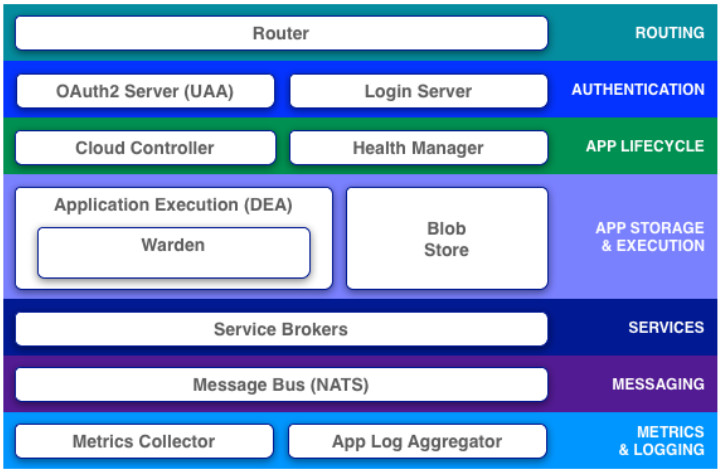

Figura 11 - Arquitetura Cloud Foundry (Fonte: Pivotal, 2015)

De acordo com a arquitetura do *Cloud Foundry*, as principais características desta ferramenta são:

- ***Router*** **–** encaminham o tráfego para o componente apropriado, normalmente o *Cloud Controller* ou uma aplicação em execução num modo de execução DEA (Pivotal, 2015);
- **Autenticação –** o servidor *OAuth2* e o *Login Server* trabalham em conjunto no sentido de fornecerem a gestão de identidades (Pivotal, 2015);
- ***Cloud Controller*** **–** responsável por gerir o ciclo de vida das aplicações (Pivotal, 2015);

- **HM9000** – monitoriza as aplicações para determinar o seu estado, versão e número de instâncias. É essencial para assegurar que as aplicações estão disponíveis (Pivotal, 2015);
- **Execução da Aplicação (DEA) -** o agente *droplet* de execução gere as instâncias das aplicações, rastreia as instâncias iniciadas, e transmite mensagens de estado (Pivotal, 2015);
- **Armazenamento *Blob* –** armazena o código das aplicações, *buildpacks* e *droplets* (Pivotal, 2015);
- ***Brokers* de serviço –** fornece a instância do serviço (Pivotal, 2015);
- **Mensagem de Barramento –** sistema de fila de mensagens leve para comunicação interna entre componentes (Pivotal, 2015);
- ***Logging* e Estatísticas –** o colecionador de métricas reúne métricas dos componentes. Os operadores podem usar esta informação para monitorizar a instância do *Cloud Foundry* (Pivotal, 2015).

No entanto, esta ferramenta tem a limitação da falta de suporte à sua *framework* (Eberhard Wolff, 2011), o que torna difícil a sua adoção.

Em termos de preços, este *PaaS* é bastante interessante, pois trata-se de um serviço de código aberto e livre.

2.10.7 Google App Engine

É uma oferta *PaaS* que permite implementar e executar aplicações na infraestrutura da Google. As aplicações são fáceis de usar, manter, e escalar consoante o tráfico e o armazenamento necessitar de alterações (Google, 2015).

É um ambiente de desenvolvimento robusto para aplicações desenvolvidas em *Java, Python, PHP* e *Go* e suporta múltiplas versões da mesma aplicação (Google, 2015), para além de ser possível integrar as aplicações com várias APIs *open source* da extensa biblioteca de *APIs* da Google designada *Google API Client Library*.

Os programadores usam o *Google App Engine* para simplificar o desenvolvimento e implementação das aplicações *web*. Estas usam o poder de auto escalonamento de computação do *App Engine,* bem como, funcionalidades integradas como memória *cache* distribuída, filas de tarefas, e armazenamento de informação, no sentido de conseguirem criar aplicações robustas, de forma fácil e rápida (Google, 2015).

É um *PaaS* muito bom em termos de custo/benefício, ideal para pequenas ou médias empresas, que necessitem de usar múltiplos serviços integrados da Google. A sua arquitetura pode ser vista na Figura 12.

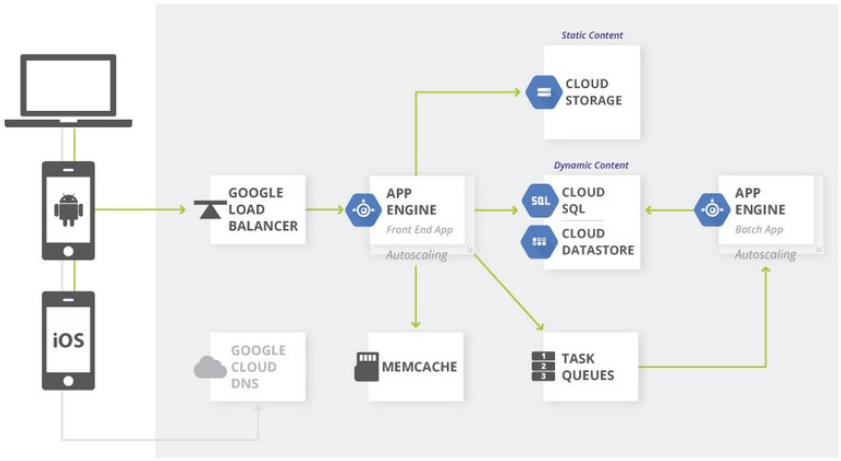

Figura 12 - Arquitetura Google App Engine (Fonte: Google, 2015)

As principais caraterísticas desta ferramenta *PaaS* são:

- ***Memcache* –** memória *cache* interna partilhada por várias instâncias, o que providencia um acesso de grande velocidade à informação (Google, 2015);
- ***Task queue* –** mecanismo que descarrega tarefas de execução longas para os servidores de *backend*, libertando os servidores de *front-end* para servir pedidos de utilizadores (Google, 2015);
- ***Load* Balancer –** fornecimento transparente de balanceamento de carga das camadas 3 e 7 para as aplicações (Google, 2015)**;**
- ***Cloud DNS* –** serviço usado para gerir zonas *DNS* (Google, 2015);
- ***Cloud* SQL –** Armazenamento e gestão de informação usando uma base de dados *MySQL* relacional. O serviço é replicado, gerido e atualizado pela Google para garantir disponibilidade e desempenho (Google, 2015);
- ***Cloud DataStore* –** base de dados *NoSQL* gerido e sem *schema* para armazenar dados não relacionais. Dimensionado à medida e suporta transações e consultas robustas, semelhantes a *SQL* (Google, 2015);
- ***Cloud Storage* –** serviço de armazenamento de objetos durável e altamente disponível. Os dados são protegidos por meio de armazenamento redundante em vários locais físicos (Google, 2015).

Em termos de desvantagens, esta ferramenta *PaaS* tem a limitação do fato de os programadores não poderem aceder diretamente ao sistema de ficheiros do *App Engine*, ou à maioria das aplicações *web* existentes não poderem ser executadas no *App Engine* sem serem modificadas (Kumar, 2014).

O preço do *Google App Engine* é baseado no uso, isto é, tem uma quota livre por dia, sendo o tráfego taxado se o limite estabelecido na quota for excedido.

2.10.8 Engine Yard

O *Engine Yard* foi desenhado para programadores *web* que usam linguagens *Ruby on Rails*, *PHP* e *Node.js,* podendo tirar partido da computação na nuvem, sem a necessidade de gerir operações, pois essas tarefas ficam ao cuidado deste serviço

PaaS. Esta ferramenta é executada na nuvem da *Amazon AWS*, sendo um serviço mais de orquestração e gestão do que propriamente de componentes de *software*. As principais tarefas deste serviço são a execução de cópias de segurança, gestão de *snapshots*, gestão de *clusters*, administração de bases de dados e balanceamento de carga (Dan Sullivan, 2014).

As suas principais características são:

- As instâncias são dedicadas com *multi-tenancy* ao nível da máquina virtual (Dan Sullivan, 2014);
- Apresenta um controlo maior sobre as instâncias da máquina virtual que outros fornecedores de *PaaS* (Dan Sullivan, 2014);
- Tem integração com repositórios *Git* públicos ou privados (Dan Sullivan, 2014).

No entanto é limitado às linguagens de programação *Ruby*, *JRuby*, *REE*, *Rubiniu*, Node.js e *PHP* (Dan Sullivan, 2014).

O seu custo é baseado no uso e no tipo de configuração pretendida com o plano mais básico a rondar os 32€/mês.

2.11 Análise Crítica

Conforme descrito anteriormente, o modelo de computação na nuvem tem vindo a crescer nos últimos tempos e a ganhar novos adeptos, quer para criar novos modelos de negócio, quer para gerar novo tipo de aplicações e com isso atingir rapidamente o *time-to-market*, reduzindo drasticamente o custo de desenvolvimento dos projetos e os custos inerentes com as infraestruturas que os suportam, o que indica claramente que este é e será o futuro das *TIC* nos próximos anos.

Apesar de existirem grandes preocupações como a privacidade da informação e a segurança, a robustez, replicação de informação, recuperação de desastres e a grande variedade de modelos de serviços disponibilizados pela computação na nuvem tem vindo a ser uma forte aposta dos fornecedores que providenciam soluções desta natureza, a fim de ganharem a confiança dos seus consumidores, que podem assim chegar a milhões de utilizadores através da elevada escalabilidade e da capacidade de fornecer recursos elásticos.

Daí que soluções como *Data Warehouses* façam todo o sentido em migrarem para a nuvem e a estarem presentes neste domínio, no sentido de poderem tirar partido de todos os seus recursos e funcionalidades. A sua adoção permitirá que surjam novas aplicações na nuvem, como aplicações analíticas, de modo a que os decisores das organizações possam ter acesso a informação relevante e consolidada do seu negócio em qualquer lugar, e com elevada disponibilidade o que serviria como um serviço de apoio às tomadas de decisão, dando origem ao *Business Intelligence* na *cloud*.

Em relação às ferramentas *PaaS*, a escolha da melhor ferramenta não se afigura fácil, apesar de terem sido descritas anteriormente as principais características e

potencialidades de cada uma delas. Com o objetivo de simplificar a análise, na Tabela 2 pode ser vista uma comparação entre as ferramentas, pretendendo–se ainda identificar as suas principais diferenças.

Tabela 2 - Comparação das ferramentas PaaS

Ferramenta	Tipo	Características	L. Programação e *frameworks*	Vantagens	Desvantagens
OutSystems	PaaS	Modelo RAD Integração com sistemas existentes Programação única para todos os dispositivos	DevOps, Java, .Net, HTML5, Javascript, CSS	Desenvolvimento rápido de aplicações Alta capacidade de integração Suporte (fóruns, vídeos, comunidade)	Custo bastante elevado para pequenas ou médias empresas Apenas um ambiente de desenvolvimento gratuito
AWS E. Beanstalk	PaaS	Auto escalonamento Balanceamento de carga Tolerância a falhas Segurança	Java, .Net, PHP, Node.js, Python, Ruby, Docker	Fácil utilização Escalabilidade automática Gratuito nos primeiros 12 meses Monitorização das aplicações	Escalonamento das aplicações baseado em métricas
Microsoft Azure	PaaS + IaaS	Serviço *PaaS* e *IaaS* como um único serviço Suporte a múltiplas linguagens de programação	.Net, Java, Node.js, PHP, entre outras	Plataforma completa Suporta múltiplas linguagens e sistemas operativos, bases de dados e *frameworks*	Portal de administração minimalista Após período gratuito, o preço mensal é calculado mediante a quantidade e tipo de recursos usado
Salesforce **(force.com e heroku)**	PaaS	Force.com Segurança Confiança e transparência Multi-tenancy Alta performance Recuperação de desastres Alta disponibilidade	Force.com Apex Heroku Ruby, Python, Java, Scala, Cloture, Node.js	Force.com Ideal para implementação de aplicações de negócio Heroku Desenvolvimento rápido de aplicações Integração com *addons*	Force.com Ausência de soluções *BI* Problemas de acessibilidade durante os planos de manutenção Heroku Custos dos *addons* elevados

		Heroku Implementações rápidas Gama de aplicações distribuídas			
Red Hat Open Shift	PaaS	PaaS privado Medição das aplicações *front-end* e *back-end* Consola *web* Ambiente de desenvolvimento Interação com plataforma por linha de comandos	Java, Node.js, Ruby, Python, PHP, Perl, Java EE, Rails, Django, Zend, Spring	Integração com Git Serviço de alojamento Ferramenta aplicacional de alojamento aberto Sem restrição a linguagens de programação	Integração de aplicações não Git
Cloud Foundry	PaaS	Autenticação e segurança Cloud Controller DEA e *brokers* de serviço *Logging* e Estatísticas Armazenamento blob	Java, Ruby, JavaScript e Python	Open source e livre Mecanismo automático para implementar aplicações Segurança Livre de custos de utilização	Suporte Documentação
Google App Engine	PaaS	Memória cache Fila de tarefas Balanceamento de carga Cloud DNS Cloud SQL Cloud Datastore Cloud Storage	Java, Python, PHP, Go	Integração com biblioteca de APIs Múltiplas versões das aplicações Integração com ferramentas *third party* Auto escalonamento	Custo baseado no uso
Engine Yard	PaaS	Suportado na Amazon AWS Multi-tenancy Integração com GIT	Ruby on Rails, PHP, Node.js	Auto gerenciamento Orquestração e gestão completa de	Custo inicial elevado Dependente da AWS L. Programação

				serviços	

Aparentemente, todas as ferramentas estudadas apresentam recursos e serviços bastante atrativos, por isso a escolha de entre uma delas dependerá sempre do grau de necessidade e quantidade de recursos exigidos para implementar uma determinada solução ou do tipo de negócio se pretende desenvolver na nuvem.

No entanto, os melhores *PaaS* em termos custo/benefício são sem dúvida o *PaaS* da Google, o da *Open Shift* do *Red Hat* e o da *Microsoft Azure*. O *Open Shift* tem como grande vantagem o custo, a não restrição de linguagens de programação, e de ter integração com Git para questões de versionamento e dependências de *software*. Mas, peca por exemplo na integração com aplicações não Git, sendo difícil de desenvolver as aplicações na plataforma de outra forma. Já o Microsoft Azure é bastante completo, pois oferece dois modelos de serviços num só. Contudo, falha em termos de custos dos serviços. O seu cálculo aparenta não ser muito transparente e claro para os utilizadores. Por fim, o *Google App Engine* tem a vantagem do seu custo, variedade de linguagens de programação que suporta, possibilidade de usar bases de dados *SQL* e *NoSQL*, bem como, a grande mais valia de ser possível desenvolver as aplicações e integrá-las com a extensa biblioteca de aplicações da Google, o que torna este serviço altamente escalável, ideal para pequenas ou médias empresas.

Face ao exposto, em termos globais o *Google App Engine* aparenta ter uma ligeira vantagem em relação aos anteriores, motivo pelo qual será a ferramenta PaaS adotada para a concretização deste projeto, designadamente para o desenvolvimento da aplicação analítica na nuvem, apesar do processo de *Data Warehousing* ser realizado com uma ferramenta de integração de dados, o *Talend Open Studio*. A integração com a nuvem da Google é efetuada com o *App Engine*, um *plugin* que é adicionado ao ambiente de desenvolvimento Eclipse. Depois, com um simples botão uma versão da aplicação é enviada para a infraestrutura da Google, ficando disponível para os utilizadores finais.

3. Caso de estudo

3.1 Introdução

Em primeiro lugar, o presente capítulo pretende identificar o problema atual das diversas empresas de desenvolvimento de *software* de gestão, que experimentam na submissão automática de informação para as entidades governamentais, bem como a sua real utilidade.

Derivado a esta causa, seguidamente neste capítulo é proposta uma arquitetura de um sistema analítico na nuvem que visa apresentar-se como uma solução viável para resolver este tipo de problema.

Além disso, é também descrita a metodologia adotada para a concretização eficaz e funcional do projeto, referindo as suas principais características e mais valias.

Por fim, neste capítulo é realizada uma conclusão sobre o caso de estudo em curso.

3.2 Descrição do problema

Atualmente existem diversas aplicações de faturação e contabilidade, como Primavera, Filosoft, F3M, entre outras que geram determinada informação em formato digital, de forma padronizada, em ficheiros *eXtensible Markup Language* (*XML)* para diversas entidades governamentais, como a segurança social, ministério das finanças e instituto nacional de estatística, de forma automatizada. Um exemplo de formato padrão para o ficheiro *XML*, e obrigatório para uso por este tipo de aplicações, conforme estabelecido na Portaria n.º 321-A/2007, é o *SAFT-PT* (*Standard Audit File for Tax Purposes*-Versão Portuguesa), o qual permite a recolha dos dados fiscais relevantes referentes aos períodos das declarações fiscais e/ou para análise, em qualquer altura, pela Autoridade Tributária e Aduaneira (AT) (Portaria 321-A/2007, 2017). A estrutura atual do ficheiro SAFT-PT está definida em (SAFT-PT, 2013) onde está representado o modelo completo do ficheiro para as seguintes obrigações fiscais:

- Faturação,
- Documentação de transporte,

- Declarações de IRC,
- Contratos de arrendamento.

Para cada uma destas diferentes situações é construído um ficheiro XML específico, ou seja, não é necessário incluir nós e atributos à estrutura que não sejam usados, o que evita que o ficheiro seja demasiado extenso. Por exemplo, o modelo da estrutura exigida para a faturação dos contribuintes é o representado na Figura 13.

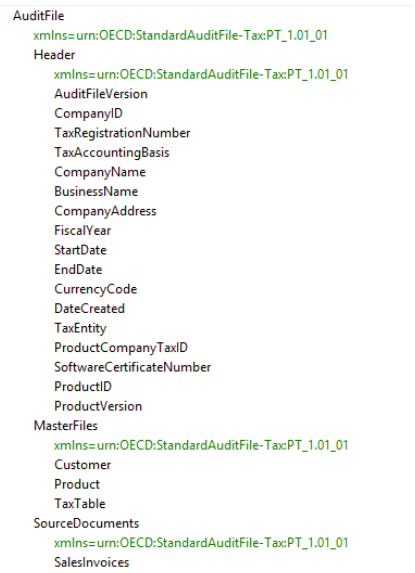

Figura 13 - Estrutura do ficheiro SAFT-PT

De igual modo, na Figura 14 está definido um exemplo do conteúdo deste ficheiro XML, e que foi usado como fonte de informação para este projeto.

TaxRegistrationNumber	AddressDetail	City	Country	FiscalYear	DateCreated	NomeCliente	NifCliente	InvoiceNo	InvoiceDate	InvoiceType	ProductDescription	Quantity	UnitPrice	TaxPercentage	TaxType	TaxPayable	NetTotal	GrossTotal
198938551	TAPADA DO FREIXO - DEVESA	MONFORTE	PT	2013	2013-02-02	Consumidor final	999999990	FB T1/1	2013-01-06	FT	CAFE	1	0.45	23	IVA	0.29	1.26	1.55
198938551	TAPADA DO FREIXO - DEVESA	MONFORTE	PT	2013	2013-02-02	Consumidor final	999999990	FB T1/2	2013-01-06	FT	CAFE	1	0.45	23	IVA	0.35	1.55	1.9
198938551	TAPADA DO FREIXO - DEVESA	MONFORTE	PT	2013	2013-02-02	Consumidor final	999999990	FB T1/3	2013-01-06	FT	CAFE	1	0.45	23	IVA	0.8	3.5	4.3
198938551	TAPADA DO FREIXO - DEVESA	MONFORTE	PT	2013	2013-02-02	Consumidor final	999999990	FB T1/4	2013-01-06	FT	BEBIDAS BRANCAS	1	1.46	23	IVA	0.34	1.46	1.8
198938551	TAPADA DO FREIXO - DEVESA	MONFORTE	PT	2013	2013-02-02	Consumidor final	999999990	FB T1/5	2013-01-06	FT	CAFE	1	0.45	23	IVA	0.2	0.9	1.1
198938551	TAPADA DO FREIXO - DEVESA	MONFORTE	PT	2013	2013-02-02	Consumidor final	999999990	FB T1/6	2013-01-06	FT	CAFE	1	0.45	23	IVA	0.1	0.45	0.55
198938551	TAPADA DO FREIXO - DEVESA	MONFORTE	PT	2013	2013-02-02	Consumidor final	999999990	FB T1/7	2013-01-06	FT	CAFE	1	0.45	23	IVA	0.25	1.1	1.35
198938551	TAPADA DO FREIXO - DEVESA	MONFORTE	PT	2013	2013-02-02	Consumidor final	999999990	FB T1/8	2013-01-06	FT	CAFE	1	0.45	23	IVA	0.1	0.45	0.55
198938551	TAPADA DO FREIXO - DEVESA	MONFORTE	PT	2013	2013-02-02	Consumidor final	999999990	FB T1/9	2013-01-06	FT	CAFE	1	0.45	23	IVA	0.1	0.45	0.55
198938551	TAPADA DO FREIXO - DEVESA	MONFORTE	PT	2013	2013-02-02	Consumidor final	999999990	FB T1/10	2013-01-06	FT	CAFE	1	0.45	23	IVA	0.36	1.59	1.95
198938551	TAPADA DO FREIXO - DEVESA	MONFORTE	PT	2013	2013-02-02	Consumidor final	999999990	FB T1/11	2013-01-06	FT	BEBIDAS BRANCAS	1	0	23	IVA	0.26	1.14	1.4
198938551	TAPADA DO FREIXO - DEVESA	MONFORTE	PT	2013	2013-02-02	Consumidor final	999999990	FB T1/12	2013-01-06	FT	AGUA	1	0	23	IVA	0	0	0
198938551	TAPADA DO FREIXO - DEVESA	MONFORTE	PT	2013	2013-02-02	Consumidor final	999999990	FB T1/13	2013-01-06	FT	CAFE	1	0.45	23	IVA	0.1	0.45	0.55
198938551	TAPADA DO FREIXO - DEVESA	MONFORTE	PT	2013	2013-02-02	Consumidor final	999999990	FB T1/14	2013-01-06	FT	CERVEJA	1	0.57	23	IVA	0.5	2.2	2.7
198938551	TAPADA DO FREIXO - DEVESA	MONFORTE	PT	2013	2013-02-02	Consumidor final	999999990	FB T1/15	2013-01-06	FT	BEBIDAS BRANCAS	1	1.3	23	IVA	1.01	4.39	5.4
198938551	TAPADA DO FREIXO - DEVESA	MONFORTE	PT	2013	2013-02-02	Consumidor final	999999990	FB T1/16	2013-01-06	FT	CERVEJA	1	0.49	23	IVA	0.11	0.49	0.6
198938551	TAPADA DO FREIXO - DEVESA	MONFORTE	PT	2013	2013-02-02	Consumidor final	999999990	FB T1/17	2013-01-06	FT	CERVEJA	1	0.49	23	IVA	0.44	1.96	2.4
198938551	TAPADA DO FREIXO - DEVESA	MONFORTE	PT	2013	2013-02-02	Consumidor final	999999990	FB T1/18	2013-01-06	FT	CAFE	1	0.45	23	IVA	0.31	1.39	1.7
198938551	TAPADA DO FREIXO - DEVESA	MONFORTE	PT	2013	2013-02-02	Consumidor final	999999990	FB T1/19	2013-01-06	FT	CAFE	1	0.45	23	IVA	0.6	2.7	3.3
198938551	TAPADA DO FREIXO - DEVESA	MONFORTE	PT	2013	2013-02-02	Consumidor final	999999990	FB T1/20	2013-01-06	FT	CAFE	1	0.45	23	IVA	0.3	1.35	1.65
198938551	TAPADA DO FREIXO - DEVESA	MONFORTE	PT	2013	2013-02-02	Consumidor final	999999990	FB T1/21	2013-01-06	FT	CAFE	1	0.45	23	IVA	0.36	1.59	1.95
198938551	TAPADA DO FREIXO - DEVESA	MONFORTE	PT	2013	2013-02-02	Consumidor final	999999990	FB T1/22	2013-01-06	FT	CERVEJA	1	0.49	23	IVA	0.31	1.39	1.7

Figura 14 - Exemplo do conteúdo de um ficheiro SAFT-PT

No entanto, é importante referir que nem todos os formatos exigidos por estas entidades são em formato *XML*, embora todos obedeçam a um conjunto de características muito especificas consoante o seu fim. Por exemplo, para o envio de eletrónico de dados referentes à declaração mensal de remunerações, quer para a Segurança Social, quer para a Autoridade Tributária, o formato exigido é em código ASCII (Finanças, 2015a). Daí que as finanças disponibilizem no seu portal informação e modelos de ficheiros com uma estrutura especifica para entrega de obrigações declarativas em formato eletrónico via Internet (Finanças, 2015b) de dados

referentes a contribuintes e empresas, para a entidades governamentais como as Finanças, Segurança Social e Instituto Nacional de Estatística.

Contudo, e apesar da geração destes ficheiros ser efetuado de forma automática, o processo de submissão desta informação é realizado de forma manual e não integrada para cada um dos portais web destes organismos institucionais, o que obriga a que a pessoa responsável ou com privilégios para tal tenha que realizar um processo algo moroso e bastante inflexível.

Além disso, o controlo da informação gerada é perdido ao longo do tempo, pois apenas estes organismos a mantêm nas suas bases de dados, impossibilitando a consulta e navegação sobre dados como faturação ou impostos que são contabilizados e declarados ao longo dos anos fiscais por parte dos contribuintes, ou empresas o que a acontecer permitiria ter uma visão sobre a informação fiscal destas entidades ou indivíduos ao longo dos anos.

3.3 Proposta de solução para sistema analítico E-Gov

Perante este cenário, uma proposta de solução para o problema antes mencionado será a criação de um sistema de gestão e auditoria fiscal, na nuvem, e responsivo com auxilio a um *Data Warehouse* que permitirá persistir a informação proveniente de várias fontes de informação estruturadas e não estruturadas, podendo realizar e fornecer relatórios dinâmicos ou personalizados, para além de poder enviar os ficheiros ou dados requeridos por estas entidades de forma automatizada por remessa direta nos portais das autoridades ou de preferência através dos *web services* disponibilizados.

A grande vantagem de um sistema desta natureza será a possibilidade de aceder e navegar sobre dados em qualquer ponto do globo, em qualquer dispositivo móvel, e em qualquer altura, pois esta estará alojada na nuvem, para além de possibilitar o envio da informação para os diferentes organismos governamentais de forma automática (Joachim et al, 2011; Diarmuid & Donal, 2015).

Os autores identificam situações adicionais onde a arquitetura proposta permitirá um sistema de armazenamento e controle cibernético (*DW*), o qual permitirá complementar a extração de conhecimento e alinhar a empresa com as estratégias que esta vier a definir em termos de gestão (Gates & Germain, 2015).

O processo de submissão de ficheiros fiscais de forma automática e a persistência da informação num repositório central como um Data Warehouse (*DW)* para posterior consulta, ilustrado na Figura 15, pode ser visto como um possível modelo de arquitetura para o sistema de gestão e auditoria fiscal (E-Gov).

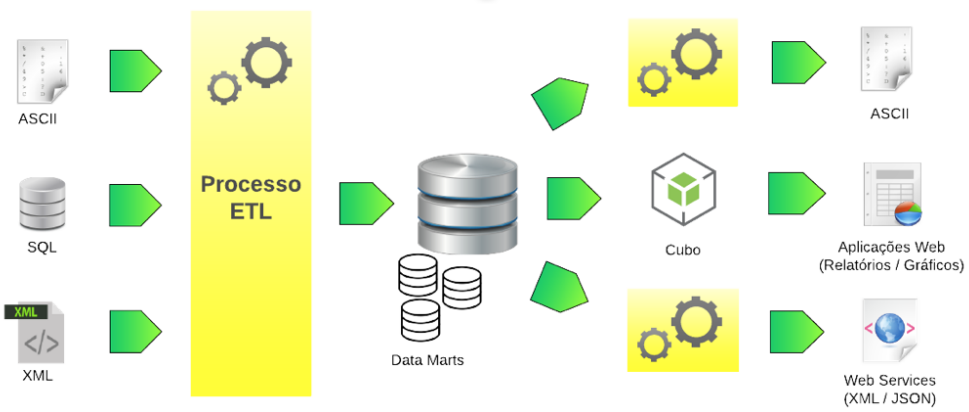

Figura 15 - Arquitetura para sistema analítico E-Gov

De acordo com a Figura 15, numa primeira fase a informação é obtida das diversas fontes de informação, quer esta seja estruturada ou não estruturada. De seguida, os dados são colocados num ambiente de estágio, isto é, o processo *ETL*, passando por um longo processo de transformação e limpeza, no sentido de poder carregar a informação para os modelos multidimensionais definidos, designados por *data marts* (DM), que em conjunto formarão o repositório central de dados, o *Data Warehouse*.

Após o carregamento do DW, a informação poderá ser passada para o cubo *OLAP (Online Analytical Processing)*. O objetivo do cubo *OLAP* é disponibilizar a informação de forma tridimensional com o recurso a *MDX (Multidimensional Expressions) queries*, permitindo navegar sobre a informação e disponibilizá-la no final em aplicações analíticas em relatórios *ad-hoc* ou gráficos, dando origem ao processo convencional de *Data Warehousing*.

No entanto, a grande diferença desta arquitetura em relação ao tradicional processo de *Data Warehousing* reside no fato da informação poder ser novamente alvo de transformação, usando ferramentas de extração, transformação e carregamento de dados. Assim que a informação exista no *DW*, poderão ser agendados novos processos *ETL*, podendo a informação transformada no final ser disponibilizada nas mais diversas formas, como em aplicações *web* analíticas na nuvem, em forma de relatórios ou gráficos, conforme ilustrado na Figura 16, suportadas em infraestruturas de fornecedores deste género de serviços, usando as suas *APIs*.

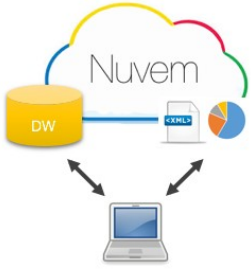

Figura 16 - Serviços integrados na nuvem

Na prática, e conforme indicado na Figura 16, a informação é inicialmente extraída das suas fontes, e transformada localmente com uma ferramenta de integração de dados. De seguida, a informação é persistida no *DW* na nuvem, no sentido de tirar partido do seu poder de computação e capacidade de armazenamento. Depois, os dados são usados diretamente para alimentar as aplicações *web* desenvolvidas na nuvem e que necessitem de usar o *DW* como fonte de informação, para efeitos de análise e navegabilidade sobre a informação, dispondo assim de um repositório centralizado. Seguidamente, e usando a mesma ferramenta, é agendado e realizado um novo processo de transformação à informação, mas desta vez com o *DW* como fonte, para dar origem aos ficheiros padronizados requeridos pelas instituições fiscais. Acredita-se por isso que este modelo arquitetural tornará o processo atual mais flexível e versátil.

Um componente intrínseco ao sistema e com um papel importante na automatização do processo de submissão dos ficheiros ou dados padronizados, é, portanto, o módulo de comunicação com os sistemas de informação destas entidades. Este módulo adotará, consoante o sistema alvo, as formas de submissão disponíveis ou por remessa direta nos portais ou através de *web services*, observando os aspetos legais e técnicos requeridos.

A título de exemplo, a Autoridade Tributária e Aduaneira (AT), disponibiliza vários *web services SOAP* para comunicação da informação fiscal, e.g., declaração IRC (Imposto Rendimento Coletável), faturas ou documentos de transporte (Finanças, 2015b). Neste caso, é requerida a construção dos pedidos SOAP (*Simple Object Access Protocol*) de acordo com a especificação *WSDL* do serviço disponibilizado (e.g., comunicação de faturas em (Finanças, 2015c), e a adoção de mecanismos de segurança. Mais especificamente, o cabeçalho dos pedidos – *SOAP:HEADER* - têm que incluir campos de autenticação do utilizador responsável com cifragem das senhas utilizando chave pública da AT, e a criação de um canal de transporte seguro *HTTPs* utilizando certificado *SSL* previamente submetido e assinado pela AT.

No final, e integrados todos esses serviços, poderia ser desenvolvida uma aplicação *web* que disponibilizaria relatórios analíticos para os seus utilizadores finais, podendo a informação ser acedida através de computadores convencionais ou dispositivos móveis e expondo serviços web, no sentido de permitir que a informação requerida pelas instituições governamentais possa ser consumida por estes o que traria benefícios em termos processuais, pois simplificaria e centralizaria os serviços e informação num único lugar.

3.4 Metodologia de gestão adotada

A metodologia de gestão de *sofware* usada para concretização do sistema analítico foi a metodologia *scrum* derivado ao menor custo de implementação do produto e do tempo de entrega do mesmo, sendo desenvolvido num processo iterativo designados de *sprints*, ilustrados na Figura 17, em que cada um deles apresentam partes funcionais do sistema e do produto, ou simplesmente pequenos protótipos, no

sentido de ir em conta ao que o cliente procura garantindo assim a sua total satisfação.

O processo foi iniciado com um primeiro *sprint*, em que serviu para realizar uma análise aprofundada aos requisitos especificados para o sistema, fazer um levantamento aprofundado das fontes de informação e implementar um pequeno protótipo de um *Data Warehouse,* apenas como prova de conceito, no sentido de verificar se o projeto era possível de ser realizado com sucesso.

Depois no segundo *sprint*, foi definida a modelação de todo o sistema analítico, designadamente, a modelação multidimensional final do DW, os *storyboards* e as *user stories* da aplicação analítica, e realizados alguns ajustes aos *data marts* modelados.

No terceiro *sprint*, foi realizada a implementação do projeto, tendo sido feita a integração com o *Google App Engine*. Além disso, foi adicionada uma nova funcionalidade em termos de segurança à aplicação e à infraestrutura da Google, nomeadamente, um sistema de acesso restrito, permitindo apenas o acesso a utilizadores com conta *Gmail* válida e com um utilizador e senha fornecidos pelos administradores do sistema, dando a ideia de uma validação a dois passos. Depois com base em *feedback* dos utilizadores finais, foram também feitas melhorias aos relatórios e ao sistema analítico em geral.

Por fim, o processo terminou com o quarto *sprint,* que serviu para adicionar novas *user stories* ao sistema, isto é, incorporar novas funcionalidades à aplicação *web*, designadamente, a possibilidade de enviar a faturação de forma automática para a Autoridade Tributária e a capacidade de visualizar os dados provenientes do *DW* de várias formas através dos cubos OLAP.

Os *sprints* tiveram uma média de duração de mês e meio, um pouco anormal tendo em conta a que normalmente, neste género de projetos cada *sprint* costuma variar entre 15 dias a um mês, mas tal sucedeu devido à indefinição clara dos requisitos iniciais.

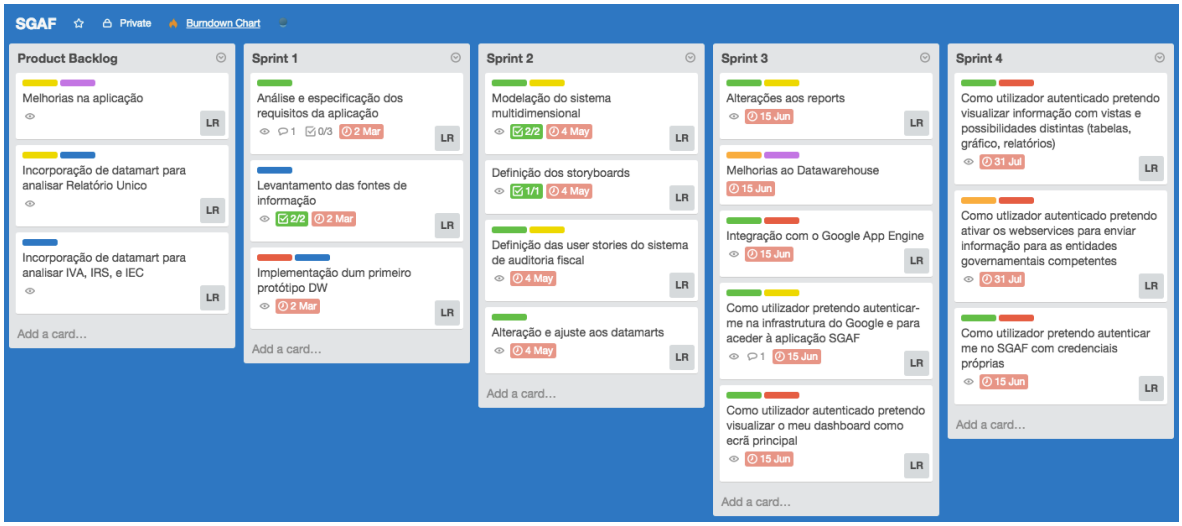

Figura 17 - Quadro scrum

No entanto, nem todas as funcionalidades foram implementadas, pois o *Product Backlog* tem tarefas por concretizar, conforme se pode verificar na Figura 17. A acontecer, estas teriam que entrar num próximo *sprint*, o que iria aumentar o valor do produto final. Assim, desta forma consegue-se ter um melhor controlo dos custos de desenvolvimento do *software*.

Neste género de projetos ágeis é normalmente definido um *Product Owner*, pessoa ou entidade para que o produto é construído. Tem também o dever e responsabilidade em definir os requisitos do *software* desenvolvido. Por isso, neste caso em particular o *Product Owner* deste projeto tratou-se de um contabilista especializado.

O projeto foi desenvolvido por uma equipa constituída por um único elemento, visto se tratar de uma prova de conceito apresentada ao *Product Owner* , não havendo a necessidade de recorrer a ajuda de um *Scrum Master* para desbloqueio de alguma tarefa ou funcionalidade do sistema.

3.5 Conclusão

O presente capítulo pretendeu evidenciar através de um caso de estudo o problema atual da submissão de informação fiscal para as entidades governamentais ser realizado de forma pouco ortodoxa, para além de não ser possível guardar a informação para efeitos de consulta no futuro.

Assim sendo, a solução encontrada para este problema concreto foi a proposta de uma arquitetura de *Data Warehousing*, baseada na criação de um repositório central de dados, juntamente com uma aplicação analítica, que terá a possibilidade de enviar a informação fiscal requerida pelas autoridades governativas de forma automática através dos seus *web services.*

O projeto foi desenvolvido adotando a metodologia *scrum,* a qual permite ter melhor controlo dos custos, bem como garante que o cliente fique totalmente satisfeito com o produto que lhe é entregue.

4. Modelação do sistema

4.1 Introdução

Este capítulo apresenta a modelação do sistema de auditoria e fiscal na nuvem, sendo descritos os requisitos funcionais e não funcionais do sistema, identificados no capítulo anterior, correspondentes aos *sprints* um e dois da metodologia *scrum* adotada. Nesse contexto, numa primeira fase pode ser visto o modelo conceptual do sistema analítico.

Seguidamente, e numa lógica de seguimento são descritos os casos de utilização do sistema analítico desenvolvido, em que se procura apresentar os seus autores, e descrever as suas principais tarefas na aplicação.

Depois, são definidas e identificadas as fontes de informação que o sistema necessita para a criação do repositório central de dados na nuvem. Identificadas as fontes de informação, de seguida, pode ser visto o modelo multidimensional da arquitetura proposta, sendo ilustrado um primeiro *data mart* em formato estrela, e um segundo *data mart* no mesmo formato, mas com a particularidade de apresentar mais uma relação, vulgarmente designado por floco de neve ou constelação. Isto é uma granularidade superior em relação ao modelo em estrela convencional.

Em seguida, pode ser visto o modelo da aplicação *web*, em que são descritos os *storyboards* da aplicação e as suas principais histórias de utilizador.

Por fim, no final do capítulo apresenta-se uma conclusão sobre a modelação adotada para o caso de estudo do presente trabalho.

4.2 Modelo Conceptual

O modelo conceptual do sistema analítico a desenvolver foi pensado para ser baseado num processo iterativo e incremental, fazendo com que as várias fases de desenvolvimento do projeto possam ser executadas e repetidas ao longo do tempo, formando um ciclo. Isto é, após serem executadas todas as fases do projeto, este poderá ser alvo de melhorias ou adição de novas funcionalidades, o que a acontecer o

sistema terá que passar novamente por todas as fases de desenvolvimento, dando a ideia de se tratar de um ciclo.

O ciclo começa com a fase inicial de recolha das fontes de informação, que depois são sujeitas ao processo de extração, transformação e carga. Seguidamente, os dados são persistidos no *Data Warehouse*, e é gerida a manutenção do repositório central, para que na fase seguinte a informação possa ser usada pelas aplicações analíticas ou possa ser gerada para os cubos *OLAP (Online Analytical Processing)*. Para que a comunicação com *web services* da Autoridade Tributária surta efeito, a informação é novamente transformada, passando por um novo processo *ETL (Extract, Transform and Load)*, no sentido de ser disponibilizada no formato de ficheiro requerido. Como o DW serve para guardar a informação de forma consolidada ao longo do tempo, o processo deverá continuar com futuros agendamentos de processo ETL, o que faz com que o processo se inicie novamente, dando a ideia de se tratar dum ciclo iterativo e incremental, conforme se pode observar na Figura 18.

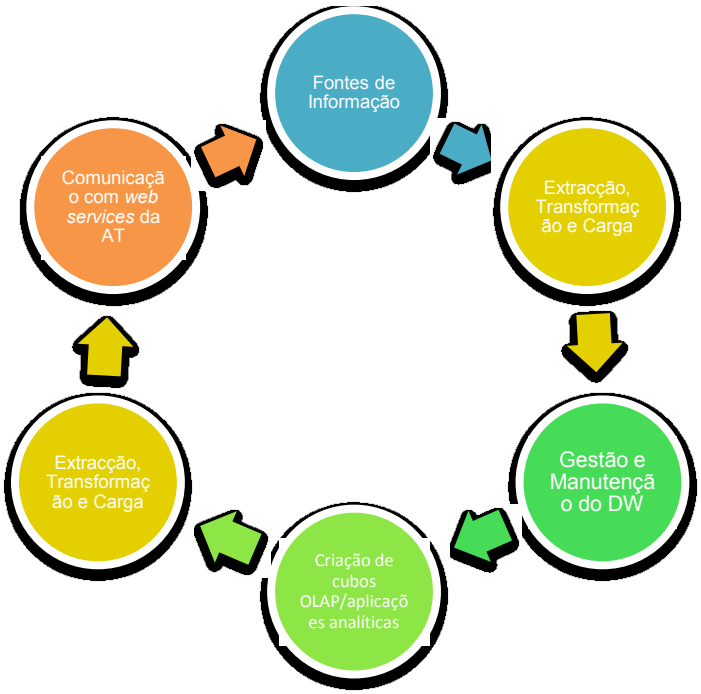

Figura 18 - Modelo conceptual do sistema analítico

Além disso, como a nuvem tem um elevado poder de computação e de armazenamento não existe o perigo efetivo de se esgotar a capacidade do servidor, pois mesmo que a quota aprovisionada pelos clientes seja atingida, o utilizador poderá aprovisionar mais capacidade de armazenamento em demanda, o que demonstra a flexibilidade desta solução.

4.3 Casos de uso

Numa primeira fase, tentou-se perceber de uma forma geral as necessidades que este sistema poderia eventualmente resolver em termos de acessos e disponibilização de informação pretendida, sendo para isso identificados os seus atores, bem como

definidas as suas principais atividades e tarefas a realizar neste sistema (da forma que se pode observar na Figura 19).

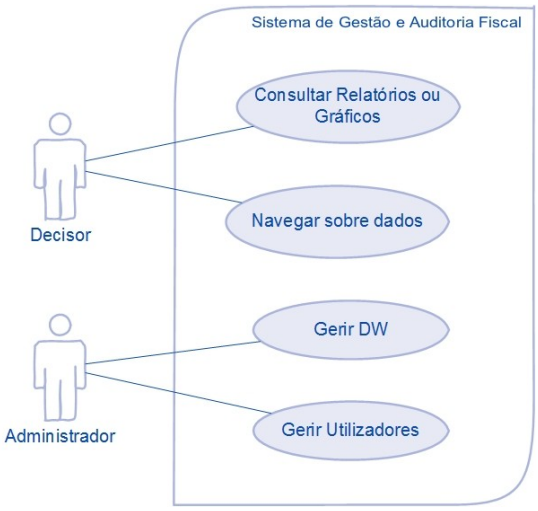

Figura 19 - Casos de uso do sistema analítico

Como se pode verificar na Figura 19, foram identificados dois atores principais para o sistema analítico e fiscal. O primeiro diz respeito ao utilizador do sistema, designado Decisor, que tem como principais responsabilidades consultar os relatórios e gráficos disponibilizados pela aplicação, ou ainda a possibilidade de navegar sobre a informação pretendida.

Por outro lado, o segundo ator, designado de Administrador tem como principais responsabilidades a gestão do sistema analítico e do *DW*, bem como, a gestão de acessos ao mesmo na nuvem.

4.4 Descrição dos casos de uso

4.4.1 Decisor

Consultar Relatórios e Gráficos

Pré-Requisitos: ser um utilizador autenticado no sistema.

Passos necessários

1. O caso de uso começa quando o decisor pretende consultar relatório(s) ou gráficos:
   1.1 É selecionado o link *Dashboard.*
   1.2 O caso de uso termina quando são apresentados os resultados do *Dashboard*.

Pós condições: é disponibilizado o *Dashboard* com a informação pretendida.

Navegar sobre dados

Pré-Requisitos: ser um utilizador autenticado no sistema.

Passos necessários

1. O caso de uso começa quando o decisor quer consultar dados, ver previsões, analisar KPI's, etc.:
    1.1 É selecionado o link *Cubo OLAP*
    1.2 O utilizador escolhe o ficheiro do cubo que pretende visualizar
    1.3 O utilizador arrasta e larga os atributos escolhidos nas linhas ou colunas da tabela *pivot*.
    1.4 O caso de uso termina quando são apresentados os resultados pretendidos.

Pós condições: são visualizados os dados presentes no cubo.

4.4.2 Administrador

Gerir DW

Pré-Requisitos: ser um utilizador autenticado no sistema.

Passos necessários

1. O caso de uso começa quando o utilizador pretende realizar *backups*, alterar, modificar ou eliminar requisitos do *Data Warehouse*.
2. O caso de uso termina quando é realizada a manutenção do *DW*.

Pós condições: é feita a gestão do *DW*.

Gerir Utilizadores

Pré-Requisitos: ser um utilizador autenticado no sistema.

Passos necessários

1. O caso de uso começa quando o decisor pretende criar, apagar ou editar novos utilizador no sistema.
2. O caso de uso termina quando é realizada a manutenção dos utilizadores.

Pós condições: é feita a gestão dos utilizadores no sistema.

4.5 Modelo da aplicação de gestão e auditoria fiscal

Para comprovar e validar a arquitetura proposta para o desenvolvimento do sistema analítico e demonstrar a sua utilidade e mais valia foi criada uma aplicação na nuvem, integrado com o *Google App Engine*, pelo que foi necessário proceder à modelação da aplicação em termos de *layout* numa lógica de seguimento da definição dos casos de uso, e, por conseguinte, da respetiva base de dados.

4.5.1 Storyboards

As histórias de utilizador definidas para a aplicação *web* estão representadas em três ecrãs, representados nas Figuras 20, 21, e 22. O primeiro ecrã diz respeito a um *Dashboard*, o segundo apresenta os cubos *OLAP*, podendo ser feita a interação com os cubos nesta seção e o último mostra a informação em pedido *SOAP* para enviar a faturação para a AT. Estas páginas constam de um cenário de autenticação no servidor da Google com uma conta *Gmail*, uma forma de segurança inicial para aceder

à infraestrutura, sendo de seguida o acesso à aplicação feito através de credenciais devidamente autorizadas.

Este processo é definido em três passos. O primeiro passo trata-se do acesso à plataforma da Google via conta *Gmail*, como pode ser visto na Figura 20.

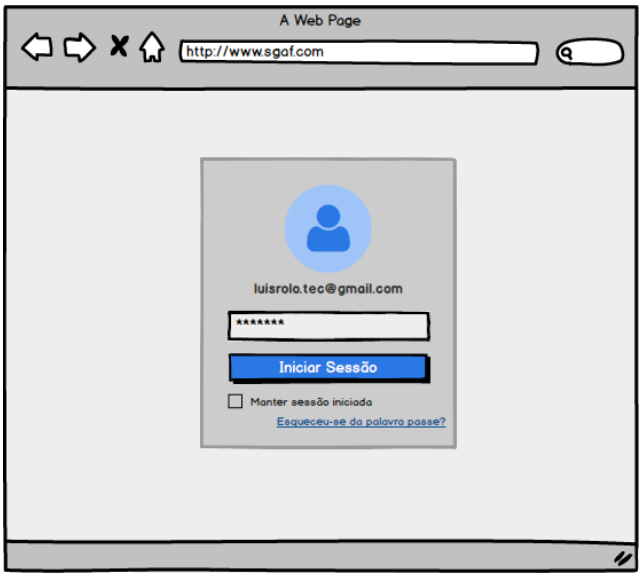

Figura 20 - Acesso à plataforma Google

O segundo passo, ilustrado na Figura 21, funciona como a segunda validação de acesso à aplicação *web*, tornando o sistema mais robusto, confiável e seguro.

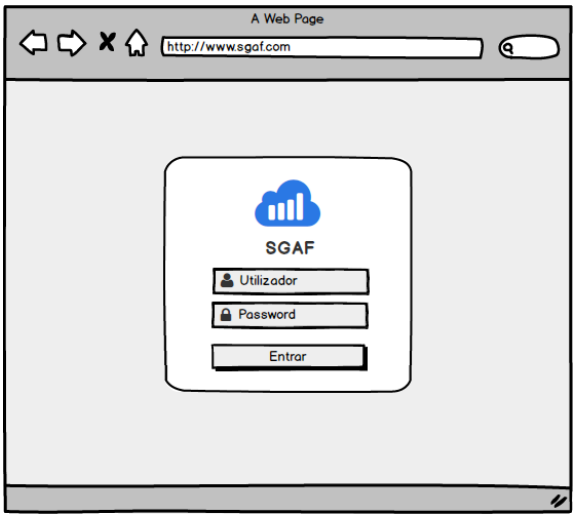

Figura 21 - Autenticação na aplicação web

Após a autenticação na aplicação ser efetuada com sucesso utilizando as credenciais fornecidas pelo administrador do sistema, o utilizador é reencaminhado para a página pré-definida da aplicação, neste caso para o *Dashboard*, como se pode observar na Figura 22.

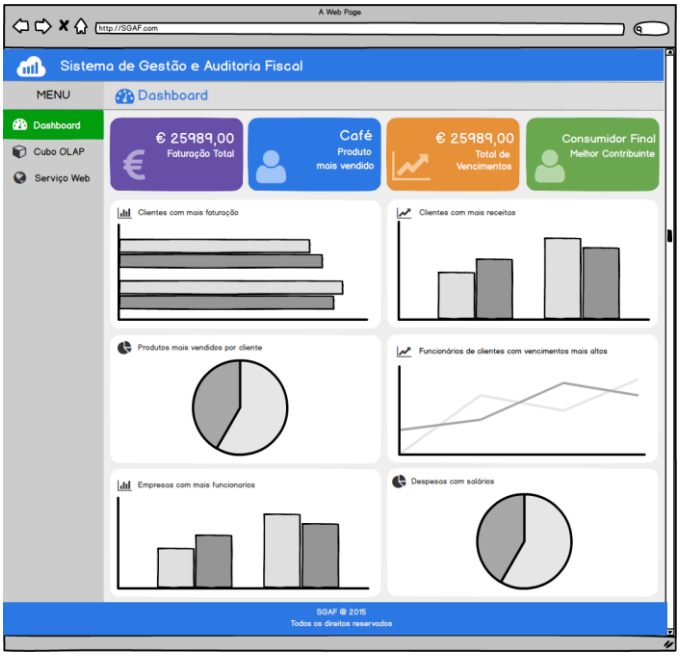

Figura 22 - Dashboard da aplicação

No entanto, o utilizador poderá optar por aceder a um outro ecrã, ilustrado na Figura 23, designado de cubo *OLAP*, no sentido de poder verificar a informação do DW de forma distinta, produzindo relatórios personalizados.

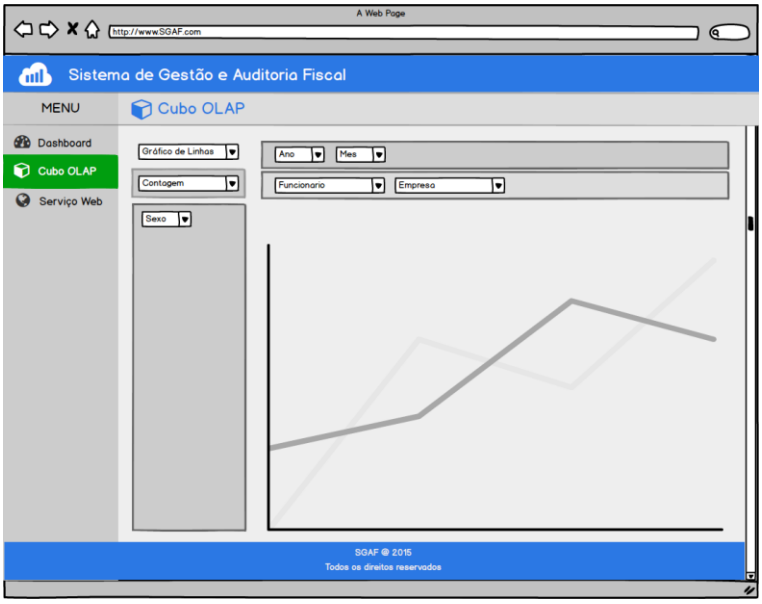

Figura 23 - Ecrã do cubo OLAP

Ou se preferir, o utilizador poderá ainda aceder a uma terceira página, nomeadamente, ao serviço *web*, para enviar a faturação para as entidades governamentais de forma simples. Para isso, bastará clicar no botão Enviar Faturação, conforme indica a Figura 24, pois a informação mais recente a enviar é previamente carregada e gerada uma previsualização no ecrã em formato *XML*, sendo apenas necessário a interação do utilizador através do clique no botão para gerar o pedido *SOAP* para enviar a faturação para o governo.

Figura 24 - Ecrã do serviço web

4.5.2 Modelo da base de dados

O modelo da base de dados para esta aplicação à exceção do modelo do *DW* referido mais à frente na seção 4.5 deste capítulo, trata-se de uma tabela independente do *DW*, apenas para fazer a gestão de utilizadores do sistema, como se pode verificar na Figura 25.

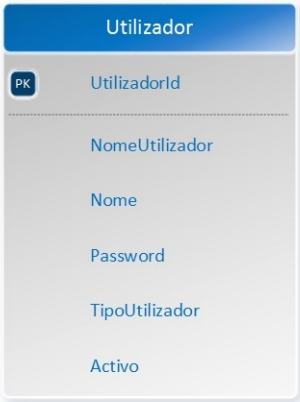

Figura 25 - Tabela utilizadores

Este segundo acesso à aplicação funciona como uma espécie de validação a dois passos muito comum em ambientes na nuvem, o que melhora a segurança da aplicação no que acesso à informação confidencial diz respeito.

4.6 Fontes de Informação

Tal como foi referido no capítulo anterior, após uma análise cuidada das fontes de informação, a grande maioria dos dados são provenientes de aplicações de contabilidade e faturação, tendo sido apenas considerados para este caso em particular ficheiros *XML* respeitantes ao *SAFT-PT (Standard Audit File for Tax Purposes - Versão Portuguesa)* e ficheiros *ASCII*, que dizem respeito à declaração mensal de remunerações de contribuintes, indicado na Figura 26. A ideia é conseguir comprovar que é possível manipular e transformar o conteúdo destes ficheiros com o

auxílio de uma ferramenta de integração de dados, pois esta permite ler e extrair a informação destes ficheiros, bem como usar funções *built-in* para a sua manipulação e transformação, copiando-a para bases de dados relacionais, informação que depois pode ser usada pelas aplicações *web* na nuvem.

Figura 26 - Fontes de informação não estruturada

Para completar os *data marts* com a informação em falta nestes ficheiros, foram usadas tabelas das bases de dados das próprias aplicações de contabilidade e faturação como fontes de informação auxiliares, representadas na Figura 27, com o objetivo de se poder extrair dados mais detalhados sobre funcionários e empresas por forma a assegurar uma melhor qualidade na informação extraída.

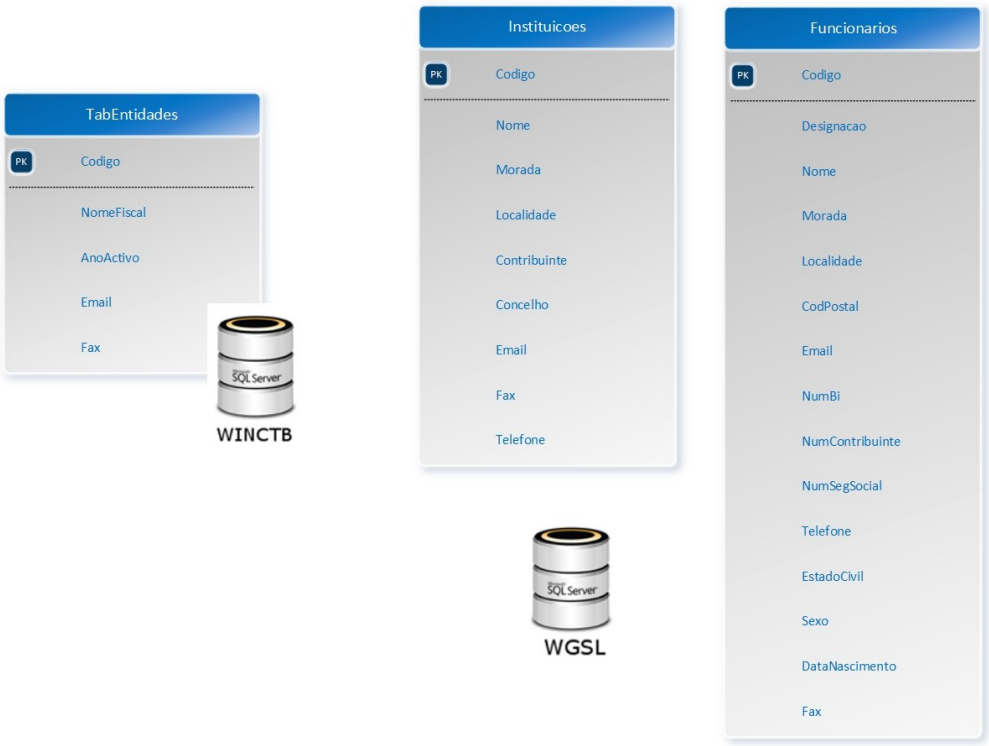

Figura 27 - Fontes de informação estruturada

Por isso, em termos de dados extraídos, foi retirada informação referente à tabela TabEntidades do programa de faturação WINCTB para completar informação para o *data mart* Faturacao e extraídos dados às tabelas Instituicoes e Funcionarios do programa de salários WGSL para perfazer informação em falta (ex. dados pessoais ou contatos de empresas e funcionários) ao *data mart* Vencimentos, conforme indica a Figura 27.

4.7 Modelo Multidimensional

Após análise exaustiva das fontes de informação construiu-se um modelo multidimensional para o sistema *DW*, de acordo com o propósito para que foi definido, isto é, o de armazenar informação de forma definitiva e consolidada para no final poder ser alvo de análise e consulta. Neste caso, foram propostos dois *data marts*, designadamente **Faturacao** e **Vencimento**.

O primeiro *data mart* procura responder à questão da faturação dos contribuintes, tendo sido identificados as seguintes dimensões e tabelas de fatos:

- **Dimensões –** Fatura, Taxa, Produto, Empresa, Calendario
- **Tabela de Fatos –** Faturacao

Por outro lado, o segundo *data mart* procura responder às remunerações dos contribuintes ao longo do tempo, construído com as seguintes dimensões e tabelas de fatos:

- **Dimensões –** Documento, Funcionario, Empresa, Calendario
- **Tabela de Fatos –** Vencimento

No entanto, a análise pormenorizada dos dois modelos com a descrição dos campos das dimensões e tabelas de fatos, bem como a relação entre elas, pode ser vista nas Figuras 28 e 29, respetivamente.

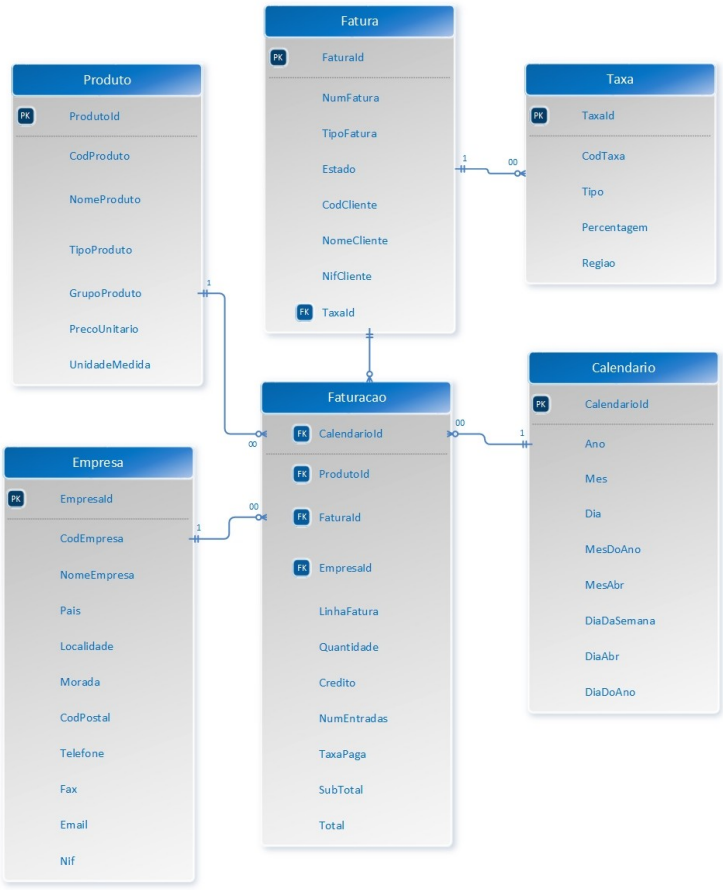

Figura 28 - Data mart Faturacao

Os atributos deste *data mart,* ilustrados na Figura 28, foram escolhidos com base na informação obtida pelos ficheiros *SAFT-PT*. Daí a escolha de dados pessoais relativamente para a dimensão **Empresa** e da descrição dos produtos para a dimensão **Produto**. Para a dimensão **Faturacao**, foram escolhidos dados relativos às faturas e definida uma constelação para aumentar a granularidade da informação das taxas. Finalmente, na tabela de fatos **Faturacao** foram definidos atributos numéricos e quantificáveis que representam as medidas desta tabela, que depois serviram para efeitos de análise e cálculos de forma agregada. Por exemplo, o atributo LinhaFatura permite medir em termos de faturação, quantas linhas de fatura em média uma fatura tem ou medir o seu total temporalmente. No entanto, este atributo poderia ter sido retirado deste modelo, o que poderá ser feito no futuro, à medida em que o modelo vai sendo refinado em novos *sprints* de desenvolvimento do projeto.

Figura 29 - Data mart Vencimento

Por outro lado, os atributos deste *data mart,* representados na Figura 29, foram escolhidos com base na informação proveniente dos ficheiros *ASCII*, relativamente a declarações mensais de remunerações. No entanto, como existia informação com pouco valor nestes ficheiros (apenas dados relativos a vencimentos, nomes de funcionários e empresas), foi extraída informação dos programas de gestão, conforme

referido anteriormente para a recolha da informação extraída ser mais representativa e com melhor qualidade. Daí a escolha dos dados pessoais relativamente para a dimensões **Empresa** e **Funcionario**. Para a dimensão **Documento**, em particular foram escolhidos atributos que permitem adição novos tipos de documentos no futuro, como por exemplo IVA, IRS ou IRC, (neste caso apenas armazenam um único documento) o que permitirá filtrar a informação do DW. Por fim, na tabela de fatos **Vencimentos**, foram escolhidos dados relativos aos vencimentos dos funcionários, atributos numéricos e quantificáveis que permite que estes sejam mensuráveis de forma agregada e no tempo.

De referir ainda que os dois modelos de *data marts* evidenciados têm duas dimensões partilhadas entre si, como é o caso da dimensão **Calendario**, que permite realizar a navegação dos dados temporalmente e a dimensão **Empresa**, que permite consultar informação referente às empresas.

Por isso, a junção dois *data marts* dão origem ao modelo multidimensional final, como pode ser visto na Figura 30.

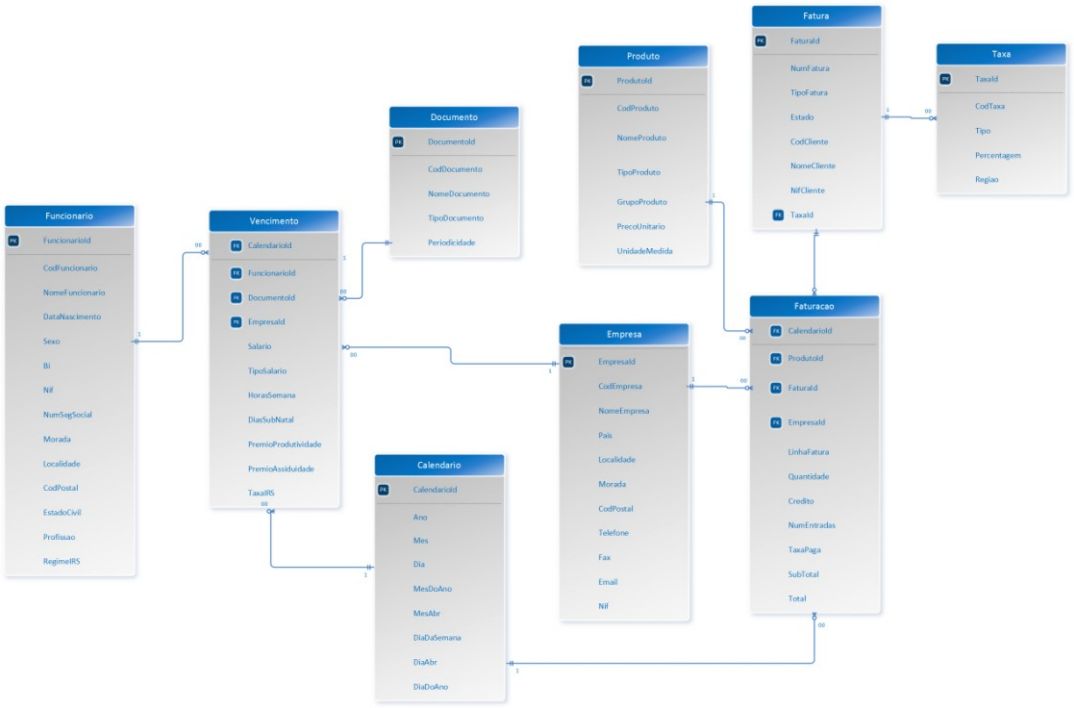

Figura 30 - Modelo multidimensional final

Face ao exposto, pode-se afirmar que este modelo multidimensional apresenta elevada escalabilidade.

4.8 Conclusão

A definição do modelo conceptual, com a identificação das fontes de informação que dão origem aos *data marts* **Faturacao** e **Vencimento**, permitiu criar um modelo multidimensional completo para o repositório central de dados do sistema analítico.

Este modelo, torna evidente a capacidade de filtragem e agregação da informação histórica pretendida por parte dos utilizadores, criando o conceito de navegabilidade da informação de forma temporal, garantido pela dimensão **Calendario**.

Nesta fase, foi ainda possível definir os casos de uso para o sistema, bem como os respetivos *storyboards*, a fim de estabelecer uma modelação completa para a aplicação *web* na nuvem.

De referir que, este modelo conceptual permite que o sistema possa ser sujeito a melhorias no futuro, para inclusão de novas funcionalidades ou tipos de informação sem a necessidade de construir um novo modelo de raiz. Por exemplo, no futuro com este modelo podem ser adicionados novos *data marts*, com informação relativa a *IRS*, *IVA* ou *IRC*, ou com informação mais detalhada dos clientes, o que garante a escalabilidade do sistema de gestão e auditoria na nuvem.

5. Implementação

5.1 Introdução

Neste capítulo é definida a implementação da arquitetura proposta e adotada, tendo em conta a modelação efetuada no capítulo anterior. Trata-se por isso do capítulo correspondente aos *sprints* três e quatro da metodologia *scrum*, a metodologia escolhida para a realização deste projeto.

Daí que num fase inicial se começa por referir como o processo *ETL (Extract, Transform and Load)* para extração, transformação e carregamento foi realizado, especificando todas as dimensões e tabelas de fatos que dão origem à construção do *Data Warehouse (DW)*.

Mais à frente, é mostrado como o cubo *OLAP (Online Analytical Processing)* foi elaborado de forma a poder ser usado como resultado prático pela aplicação *web*. De seguida, pode ser visto como foi implementada a aplicação *web*, bem como foi realizado o processo de criação do *web service*.

No final é realizada uma breve síntese sobre a implementação do projeto.

5.2 Criação do Processo ETL

Após ser definida a modelação do sistema, o primeiro processo de implementação do mesmo começou pela criação do processo *ETL* na ferramenta de integração de dados chamada *Talend Open Studio*, no sentido de gerar as dimensões e tabelas de fatos identificadas no capítulo anterior, possibilitando a definição dos respetivos *data marts*, dos cubos *OLAP* e do *web service*.

5.2.1 Dimensão Calendario

A dimensão **Calendario** poderá ser vista como a dimensão principal do *DW*, visto que esta permite a navegabilidade sobre a informação e é partilhada pelos *data marts*, permitindo a consulta de informação ao longo do tempo. O modelo para a sua criação está definido na Figura 31.

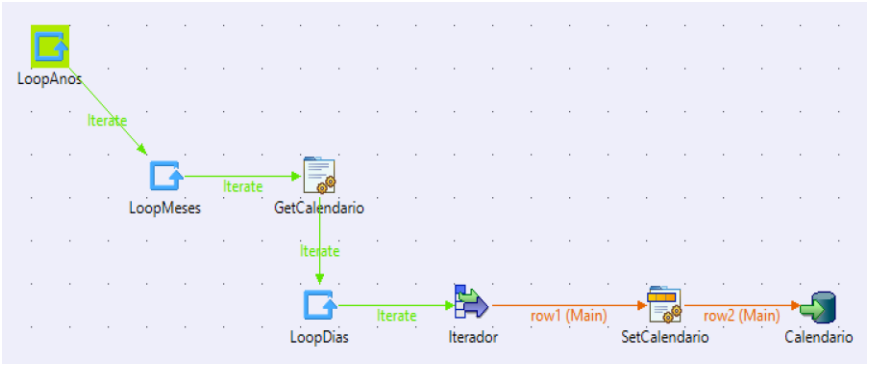

Figura 31 - Fluxo de criação da dimensão Calendario

A Figura 31 mostra três iterações no início para anos, meses e dias, respetivamente, sendo no final usada a biblioteca *Calendar* do *Java* para criar os restantes atributos da dimensão, nomeadamente os meses e dias por extenso, pois a ferramenta de integração, baseada no ambiente de desenvolvimento *Eclipse* permite usar funções e bibliotecas adicionais desta linguagem de programação de forma nativa.

5.2.2 Datamart Faturacao

Após ter sido criada a dimensão **Calendario** com sucesso, procedeu-se à definição do *data mart* **Faturacao**, sendo em primeiro lugar criadas as dimensões **Fatura**, **Taxa Produto** e **Empresa**, com base na informação proveniente dos ficheiros *SAFT-PT*, em *XML*, que depois se relacionam com tabela de fatos, conforme pode observar na Figura 32.

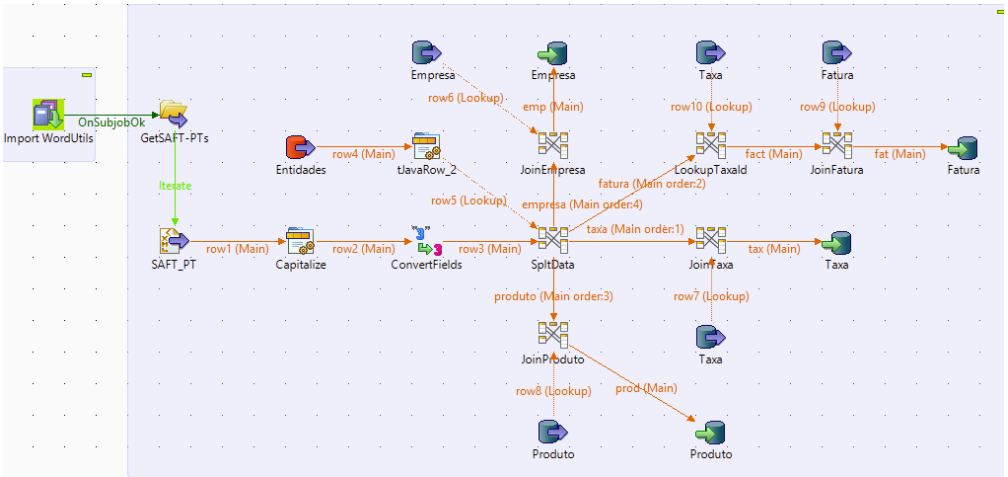

Figura 32 - Criação das dimensões do data mart Faturacao

De notar que com este *data mart* foi possível definir um nível de granularidade mais baixo, com a criação da dimensão **Taxa**, interligada com a dimensão **Fatura**, normalmente designado como floco de neve ou constelação, muito útil para detalhar ainda mais a informação pretendida.

Por outro lado, para a criação da tabela de fatos, para além de buscar e transformar a informação proveniente dos ficheiros *XML*, como se pode verificar na

Figura 33, foi feito um mapeamento entre as dimensões deste modelo de forma a permitir incorporar as *surrogate keys* na tabela de fatos.

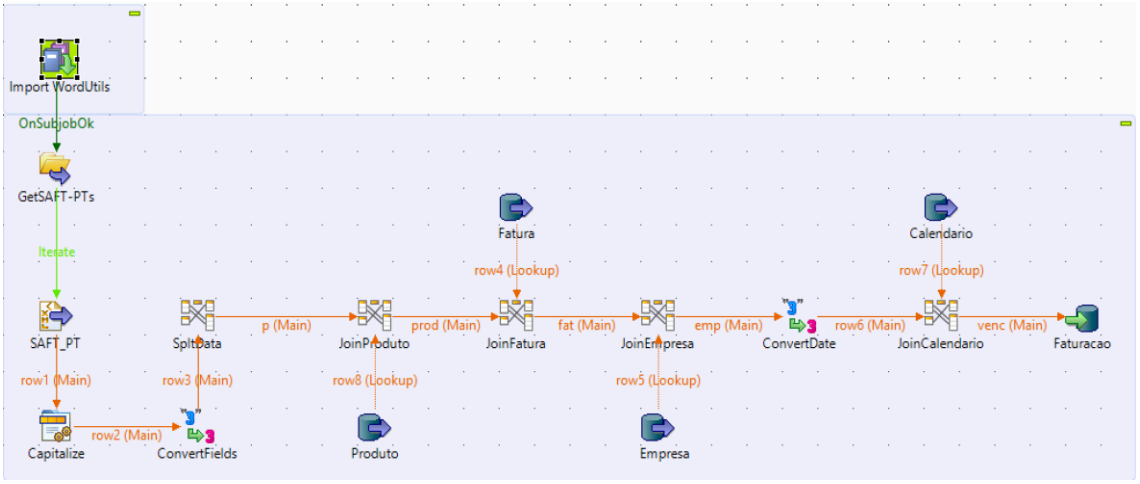

Figura 33 - Criação da tabela de fatos Faturacao

5.2.3 Data mart Vencimento

A implementação do *data mart* **Vencimento** seguiu um processo bastante semelhante ao anterior, sendo apenas desta vez a informação extraída de ficheiros *ASCII* para depois ser transformada e inserida para as dimensões **Empresa**, **Funcionario** e **Documento**.

Contudo, a grande dificuldade encontrada nesta implementação foi a maior necessidade de transformação dos dados, em relação ao *data mart* anterior, o que se revelou algo complexo por vezes. Daí que tenham sido criados dois fluxos em paralelo conforme se pode observar na Figura 34.

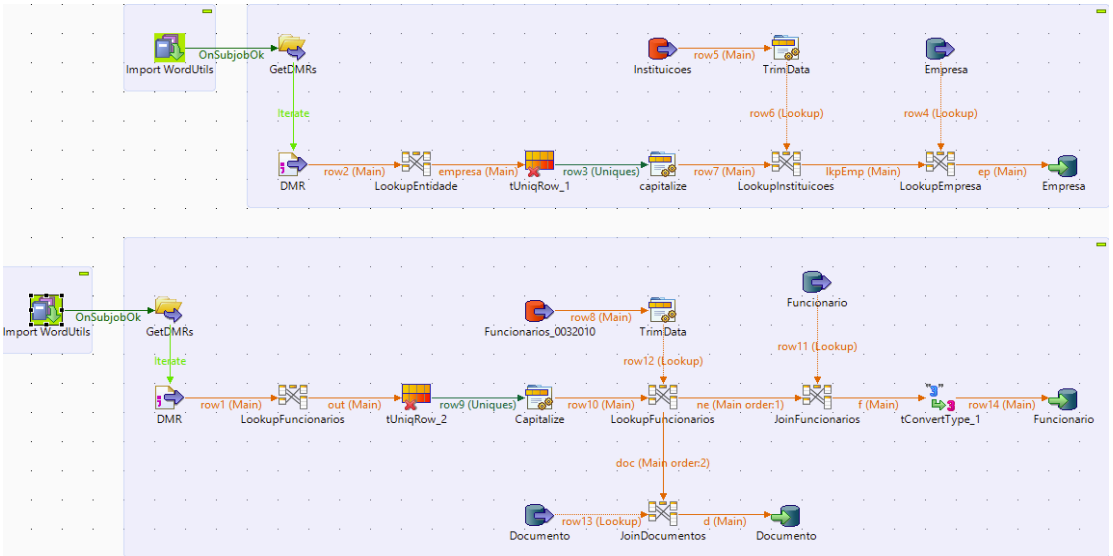

Figura 34 - Criação das dimensões do data mart Vencimento

Também a criação da tabela de fatos **Vencimento** seguiu um fluxo semelhante ao anterior, sendo no final efetuado um mapeamento com as respetivas dimensões podendo ser visto na Figura 35.

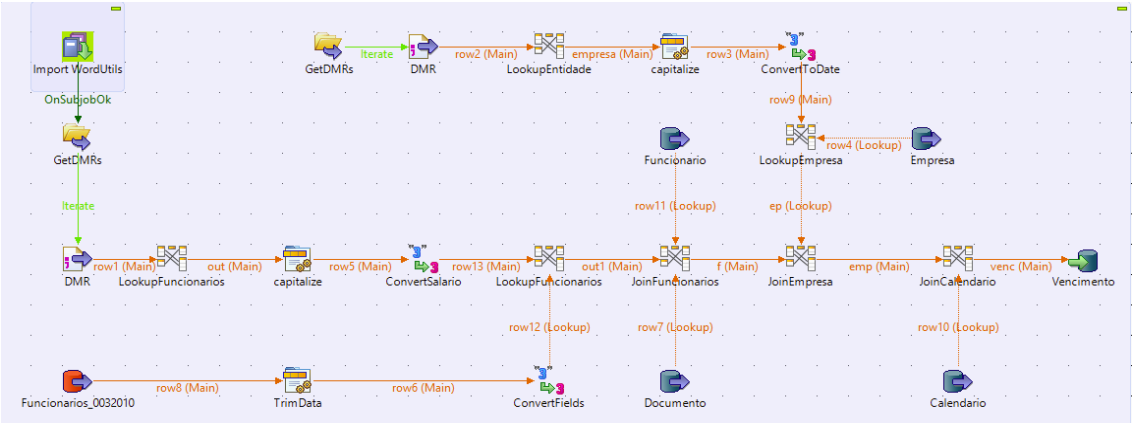

Figura 35 - Criação da tabela de fatos Vencimento

Mas, ao contrário do *data mart* anterior, não foi possível criar uma constelação, pelo que apenas foi possível cumprir com o modelo em estrela, derivado à falta de informação relevante ou valor acrescentado encontrado.

5.2.4 Criação dos cubos OLAP

Os cubos *OLAP* foram criados com base no *DW* como fonte de informação, tendo sido gerada informação de cada um dos seus *data marts*, **Vencimento** e **Faturacao**, respetivamente, conforme indica a Figura 36.

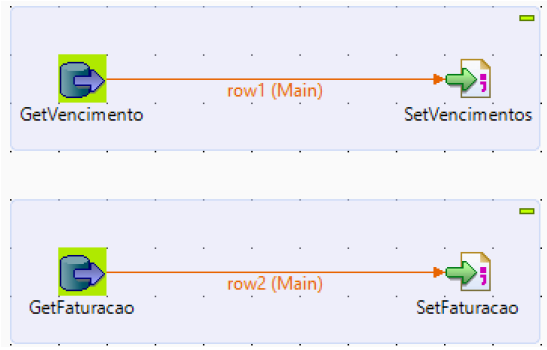

Figura 36 - Criação dos cubos OLAP

Depois, essa informação foi passada para ficheiros de extensão *CSV (Comma-Separated Values)* para depois serem usados pela aplicação *web*, graças a um *plugin Javascript* designado *PivotTable.js* (Pivotable, 2015), que recebe este género de ficheiros, permitindo assim interagir dinamicamente com a informação numa tabela *pivot*, de forma visual e tridimensional. Ou seja, trata-se de uma funcionalidade que permite realizar múltiplas vistas e agregação de informação, à semelhança das funcionalidades encontradas em programas *business intelligence* ou folhas de cálculo, como o *Microsoft Excel*. Por isso, com esta tecnologia, tornou-se possível disponibilizar os dados de forma gráfica e textual.

5.2.5 Comunicação com os *web services* da AT

Inicialmente, tentou-se recorrer ao uso da biblioteca *JAX WS* do *Framework JEE* (JAX-WS, 2015) para construir o pedido *SOAP (Simple Object Access Protocol)* no sentido de conseguir comunicar com os servidores da AT (Autoridade Tributária) e

assim poder enviar a faturação através deste serviço. Infelizmente, visto que o *Google App Engine* não suporta esta biblioteca Java (Google App Engine, 2015), recorreu-se ao uso de outra para contornar esta situação, isto é, foi usado um *plugin JQuery Soap* (JQuery Soap, 2015), que constrói o pedido *SOAP* do lado do cliente, e envia a informação via *AJAX (Asynchronous Javascript and XML).* De forma a demonstrar a sua execução, foi criado um pequeno ecrã, que disponibiliza a faturação presente no *DW*, convertida para formato envelope *SOAP*, em *XML*. Depois, foi implementado um botão para que com um simples clique seja possível enviar a informação gerada automaticamente para a AT.

No entanto, para que a comunicação fosse estabelecida corretamente foi necessário primeiro instalar o certificado da AT no servidor local e gerar uma chave privada com o *OpenSSL (OpenSSL, 2015)*, com o auxílio da chave pública disponibilizada, no sentido de realizar *handshake* das duas chaves para a comunicação em rede, e assim garantir a segurança e integridade do envio da informação. Este mesmo processo é necessário ser realizado no ambiente de produção, ou seja, na nuvem.

5.3 Criação da aplicação analítica na nuvem

A aplicação analítica foi criada localmente, no ambiente de desenvolvimento Eclipse (Eclipse, 2015), codificada na linguagem de programação Java, e usando um *plugin (Google Plugin, 2015)* de integração com o *Google App Engine*, no sentido de no final ser possível enviar a aplicação desenvolvida para a infraestrutura da *Google* com um simples clique num botão. No entanto, antes de ter sido executado este último passo é necessário criar e registar primeiro o projeto no portal Google Developers (Google Developers, 2015), para que este seja reconhecido e gerido pela *Google*.

A integração do *DW* com a plataforma da Google foi realizada de forma simples, pois foi usada uma instância local do sistema de gestão de dados *MySQL (MySQL, 2015)* para criar o *DW*, recomendado pela *Google* para ambientes de desenvolvimento, no sentido de reduzir os custos e *overhead* das transações de rede. Depois a comunicação com o repositório de dados foi realizado com o auxílio da *Java Persistence API* (JPA), pois esta *framework* dispõe de um *entity manager* que permite gerir e persistir as transações efetuadas às bases de dados.

5.4 Conclusão

A presente implementação permitiu ilustrar e descrever como o sistema de gestão e auditoria fiscal foi construído, com base na modelação realizada no capítulo anterior.

Conforme se pode verificar, o processo *ETL* ocupou grande parte do desenvolvimento da solução, assim como a comunicação com os servidores da AT, devido às dificuldades em se assegurar a encriptação da informação para a enviar para as entidades governativas responsáveis. No entanto, estas questões foram ultrapassadas com sucesso.

Por outro lado, a implementação dos cubos *OLAP* revelou-se bastante simples, pois a ferramenta de integração de dados permitiu converter facilmente a informação presente no Data Warehouse, para o formato CSV, exigido pelo *plugin Javascript PivotTable.js* da aplicação.

Por fim, a parte da integração com a nuvem do Google, nomeadamente com o *Google App Engine*, permitiu construir uma aplicação *web* bastante responsiva, integrando todos os serviços como se de um único sistema se tratasse, o que pode concluir-se que o projeto é perfeitamente exequível.

6. Resultados alcançados

6.1 Introdução

O presente capítulo tem como objetivo de mostrar os principais resultados obtidos na implementação deste sistema.

Começa por mostrar os resultados após a criação do processo *ETL (Extract, Transform and Load)*, e dos d*ata marts* construídos que constituem o *Data Warehouse (DW)*.

Depois, segue com uma ilustração da aplicação analítica construída na nuvem, em que se mostra igualmente o resultado da tentativa de comunicação com os *web services* da Autoridade Tributária.

O capítulo termina com uma breve síntese em jeito de conclusão sobre os resultados alcançados na elaboração deste projeto.

6.2 Processo ETL

O processo *ETL* que deu origem ao *DW* possibilitou criar as dimensões e tabelas de fatos que formam os dois *data marts* identificados no capítulo 4, tal como indica a Figura 37.

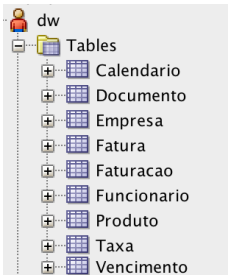

Figura 37 - Dimensões e tabelas de fatos que constituem o DW

Nesta fase, a grande dificuldade encontrada foi conseguir extrair a informação dos ficheiros *ASCII*, respeitante à declaração mensal de remunerações. Foi necessário efetuar previamente uma análise aos documentos no sentido de se encontrar um

padrão, para além de estes conterem pouca informação relevante e que pudesse acrescentar grande valor.

6.2.1 Data mart Faturação

Com uma amostra total de 21222 registos de dados relativos à faturação de três anos foi possível obter alguns resultados significativos, isto é, a quantidade de informação obtida relativamente à faturação dos contribuintes presente nas fontes de informação foi bastante elevada. Daí que, para efeitos de demonstração de resultados seja apresentado uma pequena amostra dos dados presentes no *data mart* **Faturacao**, conforme indica a Figura 38.

Query Result

SQL | Fetched 200 rows in 0,096 seconds

	CalendarioId	ProdutoId	FaturaId	EmpresaId	LinhaFatura	Quantidade	Credito	NumEntradas	TaxaPaga	SubTotal	Total	CalendarioId_1	Ano	Mes	Dia	MesDoAno	MesAbr	DiaDaSemana	DiaAbr	DiaDoAno
1	2013-05-02	7572	7572	15	1	1.0	1076.49	591	0.14	0.61	0.75	2013-05-02	2013	5	2	Maio	Mai	Quinta-feira	Qui	122
2	2013-05-02	7572	7573	15	1	2.0	1076.49	591	0.28	1.22	1.5	2013-05-02	2013	5	2	Maio	Mai	Quinta-feira	Qui	122
3	2013-05-02	7572	7574	15	1	1.0	1076.49	591	0.14	0.61	0.75	2013-05-02	2013	5	2	Maio	Mai	Quinta-feira	Qui	122
4	2013-05-02	7572	7575	15	1	1.0	1076.49	591	0.14	0.61	0.75	2013-05-02	2013	5	2	Maio	Mai	Quinta-feira	Qui	122
5	2013-05-02	7572	7576	15	1	3.0	1076.49	591	0.42	1.83	2.25	2013-05-02	2013	5	2	Maio	Mai	Quinta-feira	Qui	122
6	2013-05-02	7572	7577	15	1	5.0	1076.49	591	0.7	3.05	3.75	2013-05-02	2013	5	2	Maio	Mai	Quinta-feira	Qui	122
7	2013-05-02	7572	7578	15	1	3.0	1076.49	591	0.42	1.83	2.25	2013-05-02	2013	5	2	Maio	Mai	Quinta-feira	Qui	122
8	2013-05-02	7572	7579	15	1	3.0	1076.49	591	0.42	1.83	2.25	2013-05-02	2013	5	2	Maio	Mai	Quinta-feira	Qui	122
9	2013-05-02	7580	7580	15	1	1.0	1076.49	591	0.07	0.33	0.4	2013-05-02	2013	5	2	Maio	Mai	Quinta-feira	Qui	122
10	2013-05-03	7581	7581	15	1	1.0	1076.49	591	0.14	0.61	0.75	2013-05-03	2013	5	3	Maio	Mai	Sexta-feira	Sex	123
11	2013-05-03	7585	7585	15	1	1.0	1076.49	591	0.17	0.73	0.9	2013-05-03	2013	5	3	Maio	Mai	Sexta-feira	Sex	123
12	2013-05-03	7572	7587	15	1	2.0	1076.49	591	0.38	1.67	2.05	2013-05-03	2013	5	3	Maio	Mai	Sexta-feira	Sex	123
13	2013-05-03	7589	7589	15	1	1.0	1076.49	591	0.24	1.06	1.3	2013-05-03	2013	5	3	Maio	Mai	Sexta-feira	Sex	123
14	2013-05-03	7572	7590	15	1	3.0	1076.49	591	0.59	2.56	3.15	2013-05-03	2013	5	3	Maio	Mai	Sexta-feira	Sex	123
15	2013-05-03	7591	7591	15	1	1.0	1076.49	591	0.2	0.85	1.05	2013-05-03	2013	5	3	Maio	Mai	Sexta-feira	Sex	123
16	2013-05-03	7580	7592	15	1	1.0	1076.49	591	0.07	0.33	0.4	2013-05-03	2013	5	3	Maio	Mai	Sexta-feira	Sex	123
17	2013-05-03	7572	7594	15	1	2.0	1076.49	591	0.28	1.22	1.5	2013-05-03	2013	5	3	Maio	Mai	Sexta-feira	Sex	123
18	2013-05-03	7572	7595	15	1	1.0	1076.49	591	0.31	1.34	1.65	2013-05-03	2013	5	3	Maio	Mai	Sexta-feira	Sex	123
19	2013-05-03	7572	7596	15	1	4.0	1076.49	591	0.56	2.44	3.0	2013-05-03	2013	5	3	Maio	Mai	Sexta-feira	Sex	123
20	2013-05-03	7572	7597	15	1	3.0	1076.49	591	0.42	1.83	2.25	2013-05-03	2013	5	3	Maio	Mai	Sexta-feira	Sex	123
21	2013-05-03	7572	7598	15	1	3.0	1076.49	591	0.42	1.83	2.25	2013-05-03	2013	5	3	Maio	Mai	Sexta-feira	Sex	123

Figura 38 - Dados relativos ao data mart Faturacao

Aparentemente, e como mostra a Figura 38, cria a ilusão de as faturas apenas conterem uma única linha de fatura, mas tal perceção deve-se ao fato do ficheiro *XML* por onde é extraída a informação considerar cada registo do produto como uma linha de fatura. Eventualmente, poderia ter-se agregado essa informação à partida no ambiente de estágio onde os dados são transformados, mas tal não sucedeu para garantir a integração da informação no caso da necessidade de enviar esses dados automaticamente para os portais do governo, evitando o excesso de transformações dos dados. Também desta forma o utilizador, com as tabelas pivot poderá facilmente agregar a informação pretendida, o que não condiciona o sistema em termos de funcionalidades e não afeta a qualidade de informação.

6.2.2 Data mart Vencimento

Já em relação ao *data mart* **Vencimento**, não foi possível obter um número semelhante ao anterior, visto que o número total de fatos atingido foi de 350 referente a um único ano, conforme indica a Figura 39, o que se revelou um pouco dececionante em termos de resultados.

Query Result

SQL | Fetched 50 rows in 0,017 seconds

	CalendarioId	FuncionarioId	DocumentoId	EmpresaId	Salario	TipoSalario	HorasSemana	DiasSubNatal	PremioProdutividade	PremioAssiduidade	TaxaIRS	CalendarioId_1	Ano	Mes	Dia	MesDoAno	MesAbr
1	2014-01-01	380	1	1	405.0	P	30.0	65.0	1.11954852E10	42.0	0.0	2014-01-01	2014	1	1	Janeiro	Jan
2	2014-01-01	380	1	1	15.61	X	30.0	65.0	1.11954852E10	42.0	0.0	2014-01-01	2014	1	1	Janeiro	Jan
3	2014-01-01	381	1	1	405.0	P	30.0	65.0	1.11952988E10	42.0	0.0	2014-01-01	2014	1	1	Janeiro	Jan
4	2014-01-01	381	1	1	20.69	X	30.0	65.0	1.11952988E10	42.0	0.0	2014-01-01	2014	1	1	Janeiro	Jan
5	2014-01-01	382	1	1	405.0	P	30.0	65.0	1.11191357E10	42.0	0.0	2014-01-01	2014	1	1	Janeiro	Jan
6	2014-01-01	382	1	1	27.31	X	30.0	65.0	1.11191357E10	42.0	0.0	2014-01-01	2014	1	1	Janeiro	Jan
7	2014-01-01	383	1	1	405.0	P	30.0	65.0	1.11953797E10	42.0	0.0	2014-01-01	2014	1	1	Janeiro	Jan
8	2014-01-01	383	1	1	43.25	X	30.0	65.0	1.11953797E10	42.0	0.0	2014-01-01	2014	1	1	Janeiro	Jan
9	2014-01-01	380	1	1	405.0	P	30.0	65.0	1.11954852E10	42.0	0.0	2014-01-01	2014	1	1	Janeiro	Jan
10	2014-01-01	381	1	1	405.0	P	30.0	65.0	1.11952988E10	42.0	0.0	2014-01-01	2014	1	1	Janeiro	Jan
11	2014-01-01	382	1	1	405.0	P	30.0	65.0	1.11191357E10	42.0	0.0	2014-01-01	2014	1	1	Janeiro	Jan
12	2014-01-01	383	1	1	405.0	P	30.0	65.0	1.11953797E10	42.0	0.0	2014-01-01	2014	1	1	Janeiro	Jan
13	2014-01-01	380	1	1	405.0	P	30.0	65.0	1.11954852E10	42.0	0.0	2014-01-01	2014	1	1	Janeiro	Jan
14	2014-01-01	380	1	1	55.61	X	30.0	65.0	1.11954852E10	42.0	0.0	2014-01-01	2014	1	1	Janeiro	Jan
15	2014-01-01	381	1	1	405.0	P	30.0	65.0	1.11952988E10	42.0	0.0	2014-01-01	2014	1	1	Janeiro	Jan
16	2014-01-01	381	1	1	40.69	X	30.0	65.0	1.11952988E10	42.0	0.0	2014-01-01	2014	1	1	Janeiro	Jan
17	2014-01-01	382	1	1	405.0	P	30.0	65.0	1.11191357E10	42.0	0.0	2014-01-01	2014	1	1	Janeiro	Jan
18	2014-01-01	382	1	1	17.31	X	30.0	65.0	1.11191357E10	42.0	0.0	2014-01-01	2014	1	1	Janeiro	Jan
19	2014-01-01	383	1	1	405.0	P	30.0	65.0	1.11953797E10	42.0	0.0	2014-01-01	2014	1	1	Janeiro	Jan
20	2014-01-01	383	1	1	13.25	X	30.0	65.0	1.11953797E10	42.0	0.0	2014-01-01	2014	1	1	Janeiro	Jan
21	2014-01-01	380	1	1	405.0	P	30.0	65.0	1.11954852E10	42.0	0.0	2014-01-01	2014	1	1	Janeiro	Jan

Figura 39 - Dados relativos ao data mart Vencimento

6.3 Aplicação analítica na nuvem

A aplicação na nuvem permitiu mostrar a informação presente no *DW* de várias formas distintas. A primeira forma foi através de um *Dashboard* com um pequeno somatório da informação gerada e em forma de gráficos provenientes da biblioteca do *Google Charts*, conforme indica a Figura 40.

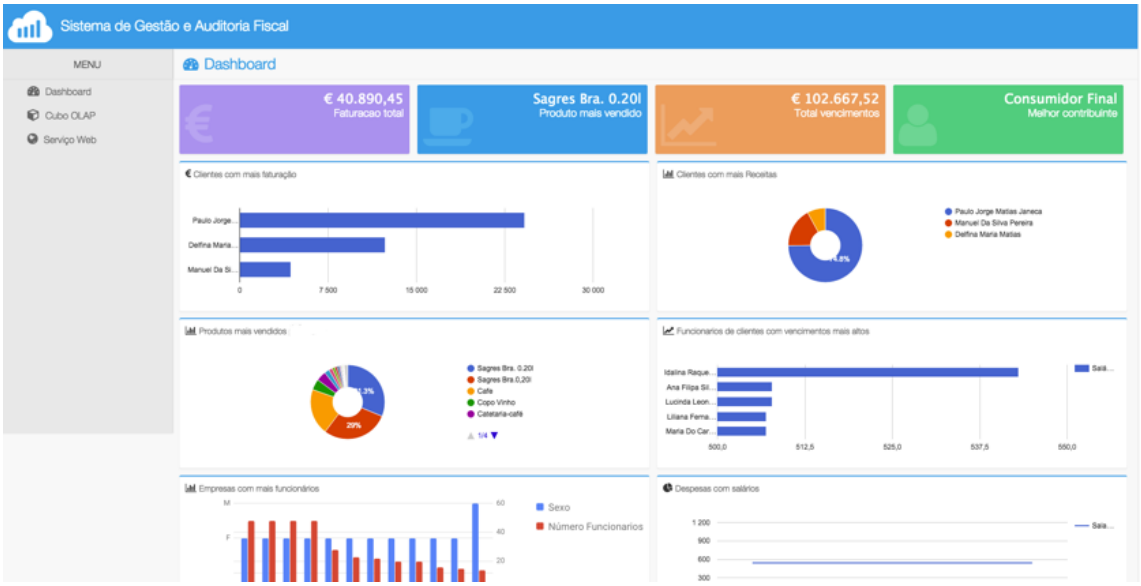

Figura 40 - Dashboard da aplicação

É de salientar que a aplicação é bastante *responsiva*, conforme indica a Figura 41, isto é, o *layout* da aplicação adapta-se facilmente ao tamanho do ecrã da grande maioria dos dispositivos, quer estes sejam dispositivos de secretária, portáteis ou dispositivos móveis, como por exemplo *tablets* ou telemóveis, o que torna a experiência de utilização da aplicação bastante agradável.

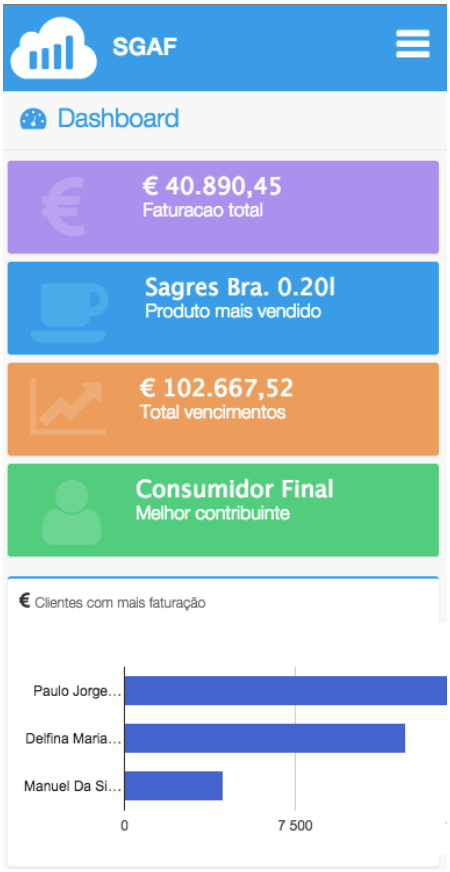

Figura 41 - Exemplo da responsividade da aplicação

Em termos de resultados, e de acordo com a amostra de faturação obtida foi possível verificar que o contribuinte com mais ocorrências se trata do consumidor final. Já em relação aos produtos mais vendidos, verificou-se que os contribuintes costumam vender mais cerveja e café aos seus clientes. Observou-se também que os funcionários em média auferem cerca de 600 euros mensais e que constituem das maiores despesas por parte das empresas.

6.3.1 Cubos OLAP

Uma outra forma de visualizar a informação na aplicação é através da importação de cubos em formato *CSV*. Assim que o ficheiro é carregado para a tabela *pivot* é possível arrastar e largar os campos da tabela nas colunas ou nas linhas dinamicamente, em tempo real, de forma muito simples e fácil. Depois, os dados são agregados e disponibilizados consoante o tipo de agregação escolhida no campo de seleção, conforme indica a Figura 42.

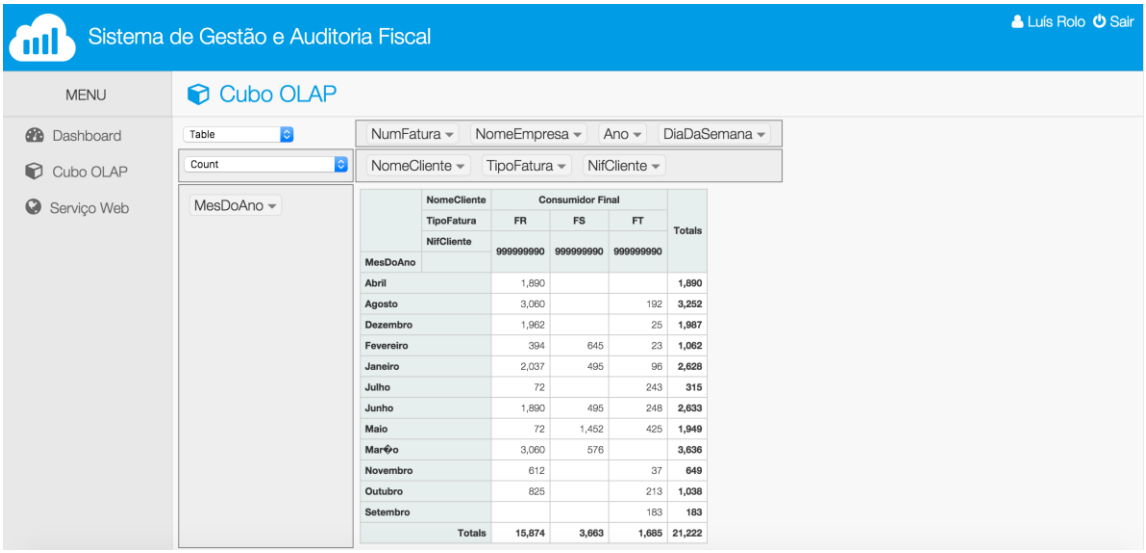

Figura 42 - Cubo OLAP referente à faturação

O *plugin Javascript PivotTable.js* (Pivotable, 2015) tem a vantagem de mostrar a mesma informação sob a forma de um gráfico, conforme ilustra a Figura 43, ou em *heatmap*, o que permite poupar tempo com a elaboração de relatórios.

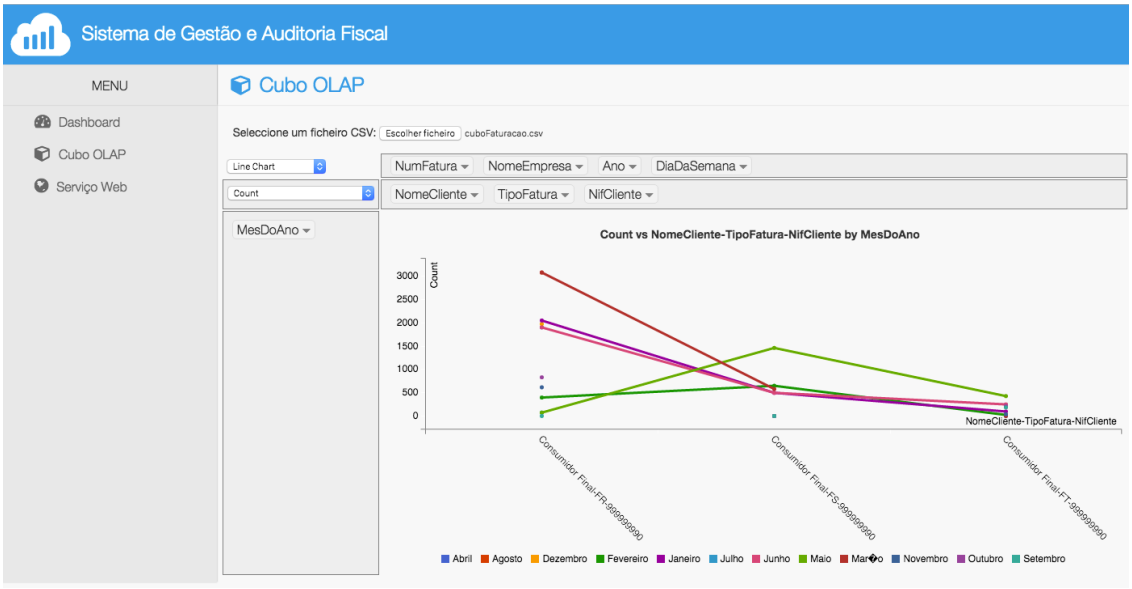

Figura 43 – Exemplo de um gráfico de linhas do cubo

6.3.2 Comunicação com o *web service* da AT

Conforme referido anteriormente no capítulo 5, a comunicação com o *web service* da AT é realizada através de uma pequena interface, designada Serviço Web. Este ecrã permite gerar a informação referente à faturação dos contribuintes, em formato *XML* e enviar esses dados automaticamente com o simples clique num botão, conforme indica a Figura 44.

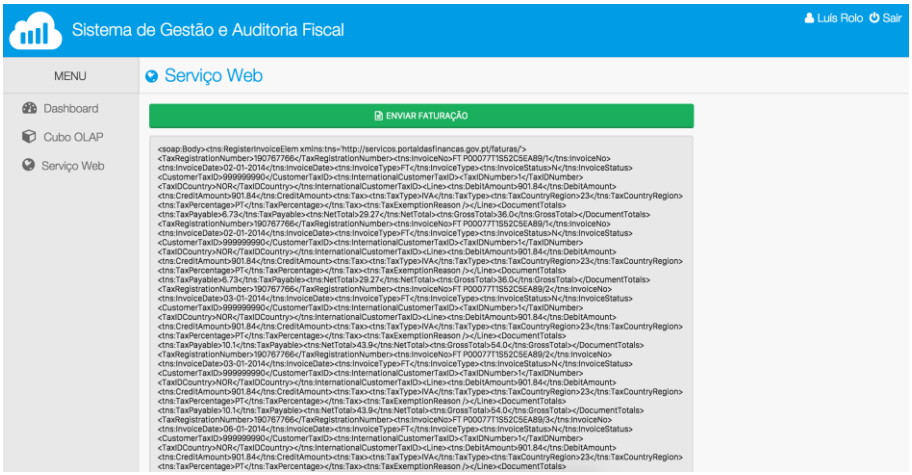

Figura 44 – Envio da faturação via web service através da aplicação

Após clicar no botão Enviar Faturação, foi gerado o pedido *SOAP (Simple Object Access Protocol)* e enviada a informação para os servidores da AT, tendo sido obtida uma mensagem de resposta de sucesso, representada na Figura 45, o que indica que a comunicação foi realizada com êxito, e a faturação foi submetida para os servidores da Autoridade Tributária.

Figura 45 - Mensagem de resposta após submissão da faturação

6.4 Testes ao sistema analítico

A metodologia *scrum* permite que sejam entregues pequenos incrementos funcionais do sistema, de forma a que o cliente seja parte integrante do projeto e ao mesmo tempo valide a proposta de solução apresentada. Por isso, no final de cada *sprint* foram realizados pequenos testes funcionais, e de usabilidade com um contabilista, o utilizador principal para o qual este sistema foi desenvolvido.

Os testes funcionais serviram para verificar se o sistema se comportava conforme o desejado, enquanto que os testes de usabilidade serviram para realizar pequenos ajustes em termos de experiência de utilização, no sentido de serem retificados no *sprint* posterior. Durante estas fases, o utilizador mostrou grande satisfação e surpresa com a acessibilidade da informação apresentada no *dashboard* principal, identificando os dados referentes aos seus clientes neste ecrã, expostos de forma

bastante organizada. Mas, onde a sua surpresa foi maior foi na parte dos cubos *OLAP*, pois com um simples arrastar e soltar de itens conseguiu verificar a faturação e vencimentos dos seus clientes de forma fácil e rápida, em várias formas distintas, funcionalidade que achou bastante útil, equiparando-a por vezes em termos contabilísticos ao balanço anual.

Por fim, no *sprint* final também achou interessante o fato da faturação poder ser enviada automaticamente através da aplicação analítica com um simples clique num botão, o que considerou uma boa integração, que lhe poderia poupar tempo em relação ao processo que efetua atualmente. Contudo, achou que a solução seria ainda mais benéfica se houvesse outro tipo de informação que pudesse ser enviada de forma automática, como por exemplo declarações mensais de remuneração ou relatórios anuais.

Para comprovar e validar a comunicação com os *web services* da AT por parte do sistema analítico, foram realizados testes no portal do *eFatura*, no sentido de verificar se a informação referente à faturação dos contribuintes foi realmente enviada pela aplicação analítica. O teste consistiu na cópia do pedido *SOAP* gerado pela aplicação *web*, diretamente para o portal do *eFatura*, nomeadamente, na seção de testes de conectividade com o *web service* do portal (E-Fatura, 2015). O resultado do teste correu conforme o esperado, como se pode observar na Figura 46.

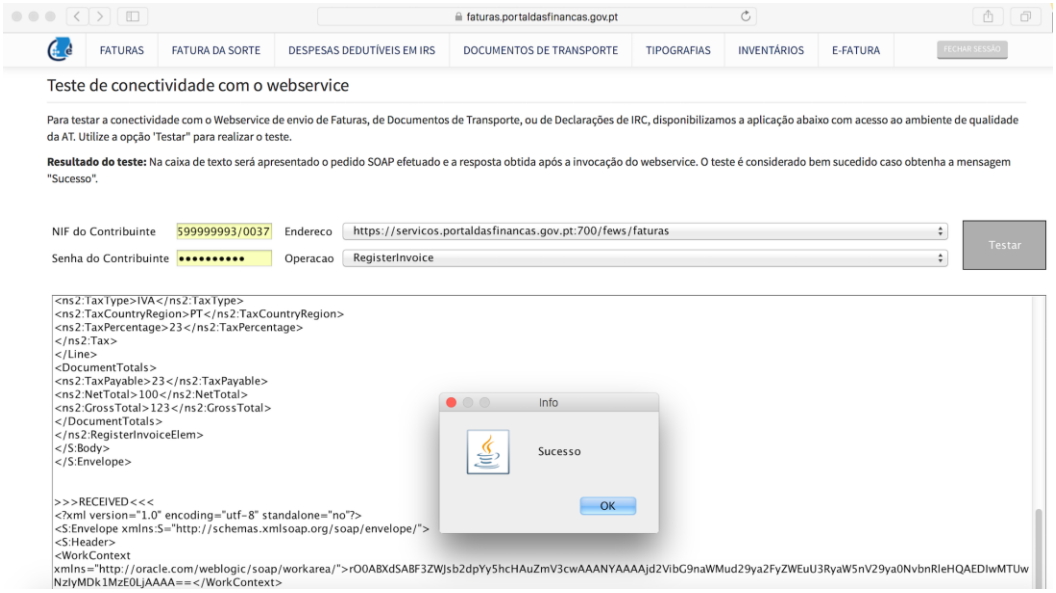

Figura 46 - Teste de conetividade com o web service

De salientar que só foi possível realizar este teste com o navegador de Internet Safari devido a incompatibilidades de execução de *applets* com outros navegadores como o *Chrome* ou o *Firefox* (os outros dois *browsers* testados). No entanto, a AT disponibiliza o código fonte, em Java, caso o utilizador pretenda criar uma aplicação nativa para a realização dos testes.

Finalmente, foram feitos testes de responsividade da aplicação *web* com o *browser* de Internet *Chrome*, da Google, através de um modo de testes que estes disponibilizam para dispositivos móveis, conforme indica a Figura 47.

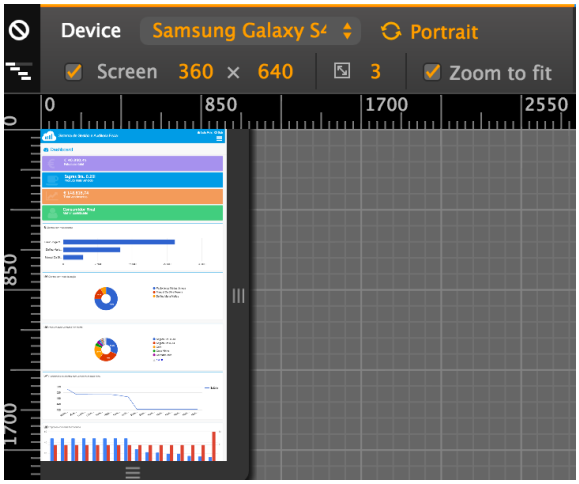

Figura 47 - Simulação de vista da aplicação num dispositivo móvel

6.5 Conclusão

O presente capítulo pretendeu demonstrar os resultados alcançados com a implementação do sistema de gestão e auditoria na nuvem.

Perante a amostra de faturação gerada no *DW*, foi possível concluir que os contribuintes não têm por hábito pedir fatura, e vendem com frequência produtos de café e cerveja.

Já em relação aos vencimentos dos funcionários das empresas conclui-se que os salários são baixos. No entanto, é onde curiosamente as empresas despendem mais dinheiro, ou seja, para pagar salários.

A interface do cubo *OLAP* permite estender as funcionalidades do *Dashboard* criado de forma a ser possível personalizar e visualizar os dados de diferentes formas, dando ênfase ao conceito de navegabilidade sobre a informação.

Os testes de realizados também mostraram a facilidade de utilização da aplicação analítica e que é possível efetuar a integração com os portais da Autoridade Tributária via *web services,* automatizando o processo de submissão e envio de faturação dos contribuintes, o que credibiliza os resultados alcançados no desenvolvimento deste sistema.

Por isso, pode-se concluir que este sistema demonstrou capacidade e fiabilidade, podendo constituir uma mais valia para os técnicos oficiais de contas, contabilistas ou contribuintes, funcionando como um sistema de apoio à tomada de decisão por estes utilizadores.

7.Conclusão

7.1 Introdução

Este capítulo pretende apresentar e evidenciar os aspetos principais do presente projeto com base no resumo do trabalho desenvolvido e de uma reflexão crítica, com o objetivo de salientar as mais valias de um sistema de gestão e auditoria fiscal na nuvem, bem como as suas limitações.

O capítulo conclui apresentando um conjunto de propostas de trabalho futuro, tendo em vista a melhoria do sistema desenvolvido, assim como continuar o trabalho de investigação na área dos sistemas de *e-government*.

7.2 Resumo do Trabalho Desenvolvido

Esta dissertação centrou-se no desenvolvimento dum sistema analítico na nuvem que permitiu armazenar dados fiscais de contribuintes e empresas num *Data Warehouse*, disponibilizando a informação em relatórios e cubos *OLAP* (*Online Analytical Processing)* através duma aplicação *web* na nuvem. O sistema desenvolvido procurou ainda solucionar o problema da submissão desses dados para as entidades do governo para efeitos fiscais, através da comunicação com os *web services* disponibilizados por estas entidades, automatizando-se assim o processo de submissão, conforme ficou demonstrado, validando-se a arquitetura proposta.

Na prática, o processo de desenvolvimento iniciou-se com a identificação das fontes de informação que passaram pelo processo de extração, transformação e carga (ETL). Depois foram definidos dois *data marts,* designadamente, Faturação e Vencimentos, que em conjunto formaram o *Data Warehouse* (DW) e possibilitaram a construção dos cubos *OLAP*. O envio da informação fiscal para as entidades do governo foi realizado com base num novo processo ETL que transformou os dados do DW relativos à faturação dos contribuintes no formato exigido por estas autoridades fiscais que depois os enviou através do consumo do protocolo *SOAP (Simple Object Access Protocol)*.

Em termos de tecnologias, foi utilizada a ferramenta de integração de dados *Talend Open Studio* (Talend, 2015) para integrar a informação com o repositório central de dados. O *Google App Engine* foi usado durante o desenvolvimento da aplicação *web,* em Java EE (Java EE, 2015), o que possibilitou integrar todos estes serviços na infraestrutura da Google e estabelecer a comunicação entre eles. Como o *Google App Engine* não suporta a biblioteca JAVA-WS (JAVA-WS, 2015), recorreu-se à tecnologia *JQuery Soap* (JQuery Soap, 2015) para construir os pedidos SOAP e enviar a faturação para os portais do governo, designadamente para o *eFatura*.

Foi demonstrado que este mecanismo automático contribuiu para a melhoria dos serviços de contabilidade e gestão, reduzindo o tempo e o esforço das entidades governamentais nas tarefas de agregação da informação para efeitos fiscais, e possibilitou o acesso a dados fiscais dos contribuintes e empresas aos principais beneficiários do sistema analítico, podendo ser acedidos em qualquer lugar e por vários dispositivos diferentes. De tal forma que os testes de usabilidade realizados com o utilizador final demonstraram que a aplicação analítica se revelou bastante fácil de usar, e os testes de conetividade com o *web service* no portal da AT, com dados gerados pela aplicação *web* revelaram a capacidade da aplicação em integrar serviços, fiabilizando a solução apresentada.

7.3 Reflexão Crítica

Este projeto resultou da dificuldade com que os contribuintes, contabilistas ou técnicos oficiais de contas se deparam na manutenção da informação pertinente sobre os seus clientes, nomeadamente dos dados relativos à faturação ou dos vencimentos, para posterior consulta e efeitos de análise, no sentido de tornar os processos de tomada de decisão mais simples e eficazes. A submissão desta informação para os organismos governamentais como as finanças e a segurança social, revela-se também um pouco ortodoxa, o que não contribui para a simplificação dos processos contabilísticos, no que a obrigações fiscais para estas entidades diz respeito.

A crescente evolução da computação na nuvem e da Internet permite a que projetos desta natureza sejam capazes de solucionar este tipo de problemas, muito embora tenha que se ter sempre em conta a segurança da informação, e a capacidade para recuperação de desastres para que a sua confiabilidade seja total.

Daí que, a arquitetura proposta para o sistema de gestão fiscal e analítico na nuvem se tenha revelado uma solução viável, de forma a automatizar e otimizar processos de submissão de faturas por exemplo, através da comunicação com os *web services* governamentais ou então simplesmente para apresentar dados aos seus utilizadores de forma estruturada, consolidada e variante no tempo, podendo ser personalizada consoante necessidade em tabelas *pivot*, ou disponibilizada graficamente, conforme foi demonstrado neste projeto.

Contudo, e apesar do projeto integrar várias tecnologias, neste momento só é possível visualizar informação referente à faturação e vencimento dos contribuintes, pelo que pode ser visto como uma limitação deste sistema.

Também, em termos de comunicação com os *web services* o sistema neste momento apenas suporta o envio da faturação para o *eFatura*, pelo que se conclui que as entidades governamentais poderiam eventualmente fornecer o mesmo tipo de API para as restantes obrigações fiscais.

7.4 Trabalho Futuro

Apesar de corresponder às expectativas iniciais criadas, o sistema poderá ser alvo de eventuais melhorias, quer em termos de quantidade de informação, quer em termos de *data marts* a incluir no *DW*. Por exemplo, podem ser adicionados dados referentes a IVA, IRS, IEC ou Relatórios Únicos.

Outra grande mais valia seria a possibilidade de enviar outros tipos de informação que não faturação via *web services*, pois atualmente ainda é só possível enviar documentos de transporte, ou declarações de IRC por transmissão eletrónica (E-Fatura, 2015).

Finalmente, poderão ainda ser adicionados filtros aos gráficos do *Dashboard* principal da aplicação analítica, no sentido de tornar o relatório mais dinâmico e permitir uma maior interação do sistema com o utilizador final.

8. Referências bibliográficas

Abraham, M. et al. 2009. *Autonomic Clouds on the Grid. Journal of Grid Computing*, pp. 1-18.

Amazon Web Services (2015). *Amazon Web Services*. [online] Disponível em: http://aws.amazon.com/pt/ [Acedido a: 13 de Janeiro de 2015].

Amit Kumawat (2013). *CMSWire – Cloud Service Models (IaaS, SaaS, PaaS) + How Microsoft Office 365, Azure Fit In*. [online] Disponível em: http://www.cmswire.com/cms/information-management/cloud-service-models-iaas-saas-paas-how-microsoft-office-365-azure-fit-in-021672.php [Acedido a: 3 de Janeiro de 2015].

Armbrust, M. et al. 2009. *Above the clouds: A berkeley view of cloud computing*. EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28.

Avandel (2015). Order This – Force.com Features [online] Disponível em: http://www.orderthis.net/force-com-features/ [Acedido a: 14 de Janeiro de 2015].

AWS E. Beanstalk (2010). *AWS Elastic Beanstalk - Architectural Overview.* [online] Disponível em: http://docs.aws.amazon.com/elasticbeanstalk/latest/dg/concepts.concepts.architecture.html [Acedido a 13 de Janeiro de 2015].

Basant Singh (2015). *5 Essential Attributes of a Cloud Service.* [online] Disponível em: http://www.techno-pulse.com/2010/12/5-essential-attributes-cloud-service.html [Acedido a: 06/10/2015].

Beal, Vangie (2015). *Webopedia - Cloud Computing (the cloud)*. [online] Disponível em: http://www.webopedia.com/TERM/C/cloud_computing.html, [Acedido a: 02 de Janeiro de 2015].

Bill Inmon (2009). *My DBA World – Blog for Database Concepts*. [online] Disponível em: http://mydbaworld.wordpress.com/2009/07/23/bill-inmon-vs-ralph-kimball/ [Acedido a: 21 de Novembro de 2014].

Bill Inmon (2014). *Beye Network. Big Data Implementation vs Data Warehousing*. [online] Disponível em: http://www.b-eye-network.com/view/17017 [Acedido a: 23 de Novembro de 2014].

Boniface, M. et al. (2009). *Platform-as-a-Service Architecture for Real-time Quality of Service Management in Clouds*.

Boulton (2015). *Wall Street Journal – The Hidden Waste and Expense of Cloud Computing*. [online] http://www.wsj.com/articles/the-hidden-waste-and-expense-of-cloud-computing-1424139032 Disponível em: [07 de Agosto de 2015].

CAPSI (2015). Sistema de gestão e auditoria fiscal na nuvem, em: 15ª Conferência – Associação Portuguesa de Sistemas de Informação. 3-4 de Outubro de 2015. Lisboa, ISCTE-IUL.

Chris Mayer (2012). *Jaxenter – Red Hat outlines OpenShift vision, PaaS to surpass SaaS by 2014*. [online] Disponível em: http://jaxenter.com/red-hat-outlines-openshift-vision-paas-to-surpass-saas-by-2014-104458.html [Acedido a: 17 de Janeiro de 2015].

Chuck Schaeffer (2014). *Crmsearch – Salesforce.com Review – An Independent Assessment*. [online] Disponível em: http://www.crmsearch.com/salesforce-review.php [Acedido a: 15 de Janeiro de 2015].

Dan Sullivan (2014). *Tom´s IT Pro. Real-World Business Technology – PaaS Providers List: Comparison and Guide*. [online] Disponível em: http://www.tomsitpro.com/articles/paas-providers,1-1517.html [Acedido a: 16 de Janeiro de 2015].

David Beaumont (2014). *Thoughts On Cloud – Insights, news, and analysis for the cloud community*. [online] Disponível em: http://www.thoughtsoncloud.com/2014/04/explain-vertical-horizontal-scaling-cloud/ [Acedido a: 01 de Dezembro de 2015].

Diarmuid Ó Coileáin & Donal O'mahony. 2015. *Accounting and Accountability in Content Distribution Architectures: A Survey. ACM Computing Survey* 47, 4, Article 59 (May 2015), 35 pages. DOI=10.1145/2723701.

Eberhard Wolff (2011). *Jaxenter – Cloud Foundry: VMware's PaaS.* [online] Disponível em: http://jaxenter.com/cloud-foundry-vmwares-paas-103117.html [Acedido a: 17 de Janeiro de 2015].

Eclipse (2015). Eclipse IDE Luna (versão 4.4.2). [online] Disponível em: https://www.eclipse.org/downloads/ [Acedido a: 20 de Janeiro de 2015].

E-Fatura (2015). Portal E-Fatura. [online] Disponível em: https://faturas.portaldasfinancas.gov.pt/testarLigacaoWebService.action [Acedido a: 29 de Setembro de 2015].

Eric Knorr (2008). *InfoWorld: What cloud computing really means*. [online] Disponível em: http://www.infoworld.com/article/2683784/cloud-computing/what-cloud-computing-really-means.html [Acedido a: 23 de Janeiro de 2015].

Fern Halper (2014). *TDWI – Demystifying cloud BI – Six Issues to Consider.* [pdf] Disponível em: http://tdwi.org/research/2014/08/checklist-demystifying-cloud-bi [Acedido a: 14 de Janeiro de 2015].

Finanças (2015a). Portal das Finanças. Ajuda – Suporte Informático – Formato de ficheiros. [online] Disponível em: https://www.portaldasfinancas.gov.pt/de/ajuda/DGCI/FAQSI.htm [Acedido a: 07 de Agosto de 2015].

Finanças (2015b). Portal das Finanças. Apoio ao Contribuinte. [online] Disponível em: http://info.portaldasfinancas.gov.pt/pt/apoio_contribuinte/ [Acedido a 15 de Junho de 2015].

Finanças (2015c). Portal das Finanças. Especificação de *Webservice* para comunicação dos dados das faturas (WSDL) [online] Disponível em: http://info.portaldasfinancas.gov.pt/NR/rdonlyres/02357996-29FC-4F11-9F1D-6EA2B9210D60/0/factemiws.wsdl [Acedido a 15 de junho de 2015].

Flavio, Leonardo e Javam (2010). Capítulo 7 - Computação em Nuvem: Conceitos, Tecnologias, Aplicações e Desafios. [pdf] Disponível em: http://www.ufpi.br/subsiteFiles/ercemapi/arquivos/files/minicurso/mc7.pdf [Acedido a: 8 de Janeiro de 2015].

Gates, Stephen & Germain, Christophe (2015). *Designing Complementary Budgeting and Hybrid Measurement Systems that Align with Strategy*, Management Accounting Quarterly. Winter2015, Vol. 16 Issue 2, p1-8. 8p.

Google (2015). Google - *Google Cloud Platform*. [online] Disponível em: https://cloud.google.com [Acedido a: 20 de Janeiro de 2015].

Google App Engine (2015). *Java App Engine should support SOAP via JAX-WS, Axis, etc.* [online] Disponível em: https://code.google.com/p/googleappengine/issues/detail?id=1270 [Acedido a: 12 de Outubro de 2015].

Google Developers (2015). [online] Disponível em: https://console.developers.google.com [Acedido a: 15 de Agosto de 2015].

Google Plugin (2015). Google Plugin for Eclipse (versão 1.9.2) [online] Disponível em: https://developers.google.com/eclipse/docs/appengine [Acedido a: 20 de Janeiro de 2015].

Hélder Borges et al. (2012). Desenvolvimento automático de aplicações e plataformas de trabalho em nuvens computacionais. [pdf] Disponível em: http://livroaberto.ibict.br/bitstream/1/865/2/Desenvolvimento%20autom%C3%A1tico%20de%20aplica%C3%A7%C3%B5es%20e%20plataformas%20de%20trabalho%20em%20nuvens%20computacionais.PDF [Acedido a: 21 de Dezembro de 2015].

IBM (2015). IBM Cloud – *What is cloud? Computing as a service over the Internet.* [online] Disponível em: http://www.ibm.com/cloud-computing/us/en/what-is-cloud-computing.html [Acedido a 6 de Janeiro de 2015].

Jan Clercq (2011). *WindowsITPro – Stay Safe in the Cloud*. [online] Disponível em: http://windowsitpro.com/security/stay-safe-cloud [Acedido a: 3 de Janeiro de 2015].

Java EE (2015). *Oracle – Java EE at a Glance*. [online] Disponível em: http://www.oracle.com/technetwork/pt/java/javaee/overview/index.html [Acedido a: 21 de Dezembro de 2015].

JAX-WS (2015). *The Java EE 6 Tutorial. Building Web Services with JAX-WS*. [online] Disponível em: https://docs.oracle.com/javaee/6/tutorial/doc/bnayl.html [Acedido a 5 de Novembro de 2015].

Joachim Götze, Tino Fleuren, Bernd Reuther, and Paul Müller. 2011. *Extensible and scalable usage accounting in service-oriented infrastructures based on a generic usage record format*. In Proceedings of the 6th International Workshop on Enhanced Web Service.

JQuery Soap (2015). *JQuery plugin for communicating with a web service using SOAP.* [online] Disponível em: https://plugins.jquery.com/soap/ [Acedido a: 05 de Novembro de 2015].

Ken (2015). *Cloud 101: The Four Deployment Models.* [online] Disponível em: https://kensvirtualreality.wordpress.com/2012/10/15/cloud-101-the-four-deployment-models/ [Acedido a: 07 de Agosto de 2015].

Khurana, S. e Verma, A. (2013). *Comparison of Cloud Computing Service Models: SaaS, PaaS, IaaS*. [pdf] Disponível em: http://www.iject.org/vol4/spl3/c0100.pdf [Acedido a: 6 de Janeiro de 2015].

Kumar (2014). *Google Developers*. *Advantages e Disadvantages of Building on the Google App Engine* .[online] Disponível em: http://www.googledevelopers.in/google-blog/advantagesdisadvantagesofbuildinganapplicationonthegoogleappengine [Acedido a 25 de Janeiro de 2015].

Matt Smith (2014). Aero – *The history and Development of Cloud Computing*. [online] Disponível em: https://www.aerofs.com/blog/the-history-and-development-of-cloud-computing/ [Acedido a: 15 de Dezembro de 2014].

Maximiliano Neto (2014). *Thoughts On Cloud – A brief history of cloud computing.* [online] Disponível em: http://thoughtsoncloud.com/2014/03/a-brief-history-of-cloud-computing/ [Acedido a: 15 de Dezembro de 2014].

Marinescu, Dan C. 2013. *Cloud Computing – Theory and Practice*, Morgan Kaufmann, Waltham, USA, 978-0-12404-627-6.

Microsoft Azure (2015). Microsoft Azure – O que é o Microsoft Azure? [online] Disponível em: http://azure.microsoft.com/pt-pt/overview/what-is-azure/ [Acedido a: 13 de Janeiro de 2015].

MySQL (2015). *MySQL - The world's most popular open source database (Version 5.5.42)* [online] Disponível em: https://dev.mysql.com/downloads/mysql/5.5.html [Acedido a: 10 de Janeiro de 2015].

Open Shift (2015). *Open Shift – OpenShift Platform as a Service*. [online] Disponível em: https://www.openshift.com/products [Acedido a: 17 de Janeiro de 2015].

OutSystems (2014). *OutSystems Platform. Arquitecture & Infrastructure*. [pdf] Disponível em: http://www.outsystems.com/home/document-download/178/8/0/0/OutSystems Platform - Architecture and Infrastructure Overview.pdf [Acedido a: 2 de Dezembro de 2014].

OutSystems (2015). *OutSystems - Multi-Channel Application Development.* [online] Disponível em: http://www.outsystems.com/ [Acedido a: 4 de Janeiro de 2015].

OpenSSL (2015). *Open SSL – Cryptography and SSL/TLS Toolkit* (Versão 0.9.8). [online] Disponível em: https://www.openssl.org/source/ [Acedido a: 15 de Abril de 2015].

Parsi, Kalpana et al. (2013). *International Journal of Advanced Research in Computer Science and Software Engineering – A Comparative Study of Different Deployment Models in a Cloud.* [online] Disponível em: http://www.ijarcsse.com/docs/papers/Volume_3/5_May2013/V3I5-0229.pdf [Acedido a: 7 de Agosto de 2015].

Pivotal (2015). *Pivotal Documentation. Cloud Foundry Overview*. [online] Disponível em: http://docs.pivotal.io/pivotalcf/concepts/overview.html [Acedido a: 16 de Janeiro de 2015].

Pivotable (2015). *PivoTable.js Examples*. [online] Disponível em: http://nicolas.kruchten.com/pivottable/examples/ [Acedido a: 5 de Junho de 2015].

Portaria 321-A/2007. 2007. Portaria 321-A/2007 de 26 de Março, Diário da República, 1ª Nº 60- 26 de Março de 2007.

Ralph Kimball et al (2002). *The Data Warehouse Toolk*it. Second Edition. New York, EUA: John Wiley & Sons, Inc.

Reed Law (2015). *SmashingBoxes – Heroku vs Amazon Web Services*. [online] Disponível em: http://smashingboxes.com/ideas/heroku-vs-amazon-web-services [Acedido a 25 de Janeiro de 2015].

Regalix (2014). *Regalix – Market Watch: Top 5 PaaS Companies to Watch in 2014.* [online] Disponível em: http://www.regalix.com/by_regalix/insights/articles/market-watch-top-5-paas-companies-watch-2014/ [Acedido a: 14 de Janeiro de 2015].

SAFT-PT (2013). Estrutura de dados, versão 1.03_01, associada à Portaria n.º 274/2013 de 21 de agosto de 2013 [online] disponível em: http://info.portaldasfinancas.gov.pt/apps/saft-pt03/SAFTPT1.03_01.xsd [Acedido a: 15 junho de 2015].

Sandip Chowdhury (2015). *Data warehouse augmentation, Part 1: Big data and data warehouse augmentation.* [pdf] Disponível em: http://www.ibm.com/developerworks/library/ba-augment-data-warehouse1/ba-augment-data-warehouse1-pdf.pdf [Acedido a 6 da Janeiro de 2015].

Sanjai Rishi (2015). *Creating new business models and revenue streams with cloud.* [online] Disponível em: https://www.ibm.com/cloud/resourcecenter/content/51 [Acedido a: 20 de Dezembro de 2015].

Salesforce (2015). *Salesforce.com – Social Success - Why Move to the Cloud? 10 Benefits of Cloud Computing.* [online] Disponível em: http://www.salesforce.com/uk/socialsuccess/cloud-computing/why-move-to-cloud-10-benefits-cloud-computing.jsp [Acedido a: 6 de Janeiro de 2015].

Sosinsky, Barrie, 2011. *Cloud Computing*. Indianapolis, Indiana: Wiley Publishing, Inc.

Talend (2015). *Talend – Talend Open Studio for ESB (Version 5.6).*[online] Disponível em: https://www.talend.com/download/talend-open-studio#t3 [Acedido a: 21 de Dezembro de 2015].

TI Inside (2015). Serviços de nuvem já afetam negócios de empresas indianas de *outsourcing* de TI [online] Disponível em: http://convergecom.com.br/tiinside/13/07/2015/servicos-de-nuvem-ja-afetam-negocios-de-empresas-indianas-de-outsourcing-de-ti/ [Acedido a: 2 de Dezembro de 2015].

Third Nature (2012). *Cloud Computing Models for Data Warehousing: 2012 Technology White Paper.* [pdf] Disponível em: http://www.enterprisenetworkingplanet.com/ebooks/112275410/95950/6570310/119343 [Acedido a: 28 de Dezembro de 2014].