



Novas Tendências, O Big Data

Deteção de Fraudes e Transações no Comercio Local: SAF-T com Tecnologia Big Data.

Elói Manuel Cardoso Lopes

Orientadores

Professor Doutor Eurico Ribeiro Lopes

Professor Doutor Henrique O'Neill

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 da categoria profissional do orientador Doutor Eurico Ribeiro Lopes, do Instituto Politécnico de Castelo Branco.

Março de 2016

Composição do júri

Presidente do júri

Grau académico, nome do presidente do júri”

Vogais

Prof. Doutor, Eurico Ribeiro Lopes

Professor Coordenador na ESTCB

Prof. Doutor, Henrique José da Rocha O'Neill

Professor Associado no ISCTE - Instituto Universitário de Lisboa

Prof. Doutor Renato Jorge Nunes

Professor Auxiliar no Instituto Superior Técnico de Lisboa

Prof. Doutor Nuno Castela

Professor adjunto na ESTCB

Resumo

A fraude e a evasão fiscal tem sido algo que as autoridades tributárias têm enfrentado arduamente nos últimos anos [49], com a adoção do e-fatura [48] em Portugal foi introduzido algum controlo das transações efetuadas entre o consumidor final e o comerciante, ficando tudo registado num ficheiro denominado SAF-T [26]. O projeto desenvolvido nesta dissertação consistiu no desenvolvimento de um protótipo que pretende detetar fraudes em vendas, *stocks* e controlo de inventários de produtos para comércios locais utilizando tecnologia no domínio do Big Data [5].

Foram utilizadas ferramentas na área do Big Data, em particular o ecossistema do Hadoop, nomeadamente o HDFS e o Hive. Este último permite o armazenamento, a consulta e a gestão dos dados. Com o auxílio destas ferramentas foi desenvolvido um protótipo onde são carregados ficheiros SAF-T gerados pelas máquinas registadoras que estão instaladas nos estabelecimentos, sendo os dados transformados num processo que será descrito na dissertação, para poderem ser manuseados com o Pentaho Report Designer [50].

A aplicação tem relatórios específicos que permite detetar determinados tipos de fraudes em vendas, existências de *stocks* e controlo de inventários de produtos para comércios locais. Destaca-se por exemplo o cruzamento de dados entre cliente (comerciante) e o retalhista para se poder confirmar a consistência entre o que é declarado por ambos, em auditorias se os bens (produtos) que o comerciante adquiriu, corresponde ao que foi faturado, bem como deteta se todos os produtos que são vendidos foram faturados, e que não houve evasão fiscal.

Os dados que foram utilizados nos testes da ferramenta foram cedidos por um comerciante local e a ferramenta desenvolvida num portátil que prova que o Hadoop pode ser executado em máquinas sem ser necessário ter hardware de elevada qualidade.

Palavras chave

Big Data, Hadoop, Business Intelligence; SAF-T, Auditoria, Evasão Fiscal

Abstract

Fraud and tax evasion has been something that the tax authorities have faced hard in recent years [49], with the adoption of *e-fatura* [48] in Portugal was introduced some control of transactions between the consumer and the shopkeeper, getting all recorded in a file named SAF-T [26]. The project developed in this dissertation was the development of a prototype that aims to detect fraud in sales, stocks and control products inventories for local businesses using technology in the area of Big Data [5].

Tools have been used in the field of big data, in particular the ecosystem Hadoop, including HDFS, Map and Reduce hive. The Hive allows the storage of data, queries and management. With the help of these tools has been developed a prototype which are loaded SAF-T files generated by cash register that are installed in shops, data is transformed in a process that will be described in the dissertation, in order to be handled by reporting tool the Pentaho Report Designer [50].

The application have specific reports to detect certain types of fraud in sales, stock inventory and control of product inventory to local businesses. It stands out for example the exchange of data between client (shopkeeper) and retail to be able to confirm the consistency between what is declared by both, in audits if the goods (products) that the shopkeeper has corresponds to what was billed and detect whether all the products that are sold were invoiced for no tax evasion.

The data that were used in the tool tests were provided by a shopkeeper and tool was developed in a laptop that proves Hadoop can run on machines without the need to have high quality hardware.

Keywords

Big Data, Hadoop, Business Intelligence; SAF-T,Audit, Tax Evasion

Índice geral

Composição do júri	iii
Resumo	v
Palavras chave.....	v
Abstract	vii
Keywords	vii
Índice geral.....	ix
Índice de figuras	xiii
Lista de tabelas.....	xv
Lista de abreviaturas, siglas e acrónimos	xvi
1. Introdução.....	1
1.1 Contexto	1
1.2 Estrutura da dissertação	1
2. Estado da Arte	3
2.1 O que é o Big Data?.....	3
2.2 Origem do Big Data	3
2.3 Características do Big Data	5
2.4 Arquitetura Big Data	6
2.5 Big Data nas empresas	8
2.6 Principais Players	9
2.6.1 Hortonworks Data Platform	10
2.6.2 Cloudera	11
2.6.3 MapR	11
2.7 Business Intelligence & Analytics e Big Data.....	12
2.7.1 Business Intelligence & Analytics	12
2.7.2 Componentes de um Sistema de Business Intelligence	13
2.8 Ferramentas	15
2.8.1 Google Big Table.....	15
2.8.2 Hadoop	15
2.8.3 O Ecosistema do Hadoop.....	16
2.8.3.1 Hadoop Distributed File System.....	17
2.8.3.2 NameNode e DataNodes.....	18
2.8.3.4 MapReduce	19
2.8.3.5 Arquitetura do MapReduce.....	19

2.8.3.6	Arquitetura Hadoop Map Reduce	19
2.8.3.7	YARN	21
2.8.3.4	HIVE	22
2.9	Bases de Dados [22]	23
2.9.1	Bases de dados Relacionais e NoSQL	24
2.10	SAF-T.....	24
2.11	Análise final	25
3.	Análise de Requisitos.....	27
3.1	Introdução	27
3.2	Caso de Estudo	27
3.2.1	Arquitetura Hadoop	28
3.3	Plano de ações.....	29
3.3	Conclusão.....	29
4.	Modelo de Dados.....	31
4.1	Introdução	31
4.2	Arquitetura tecnológica	31
4.3	Casos de Utilização	32
4.4	Modelo Conceptual	33
4.5	Fontes de Dados	34
4.6	Modelação de dados.....	35
4.7	Conclusão.....	37
5	Implementação	38
5.1	Introdução	38
5.2	Processo ETL.....	38
5.3	Carregamento dos ficheiros SAF-T	38
5.3.1	Dimensão Cliente	41
5.3.2	Dimensão Produto.....	41
5.3.3	Dimensão Comerciante	41
5.3.4	Dimensão Taxa	42
5.3.4	Dimensão Tempo.....	42
5.3.5	Tabela de Factos Fatura	43
5.3.6	Tabela de factos Stocks	43
5.4	Relatórios.....	44
5.5	Conclusão.....	45
6	Resultados Obtidos.....	46
6.1	Introdução	46

6.2	Processo ETL	46
6.2.1	Dimensão Comerciante	46
6.2.2	Dimensão Cliente	47
6.2.3	Dimensão Produto.....	47
6.2.4	Dimensão Taxa	47
6.2.5	Dimensão Tempo	48
6.2.6	Tabela de Factos FactFactura	48
6.2.7	Tabela de Factos Stocks.....	49
6.3	Relatórios	49
6.4	Conclusão	49
7	Conclusões	51
7.1	Reflexão	51
7.2	Trabalho Futuro	52
	Referências	54

Índice de figuras

Figura 1 - Dados gerados a nível mundial [33]	5
Figura 2 - Arquitetura Big Data [37].....	7
Figura 3 - Big Data Plataformas [40].....	10
Figura 4 - Hortonworks Data Platform.....	10
Figura 5 - Cloudera Distribution for Hadoop [43]	11
Figura 6 - MapR Converged Data Platform [44].....	11
Figura 7 - Diferenças entre MapR, Hortonworks e Cloudera [45]	12
Figura 8 - Arquitetura BI de 3 camadas	14
Figura 9 - Sistema de BI com cinco camadas.....	15
Figura 10 - Ecossistema do Hadoop [14].....	17
Figura 11 - Arquitetura HDFS.....	18
Figura 12 - Arquitetura Master / Slave do MapReduce [36]	20
Figura 13 - Arquitetura YARN.....	21
Figura 14 - Arquitetura do Hive [19]	23
Figura 15 - Estrutura SAF-PT.....	25
Figura 16 – Arquitetura proposta	28
Figura 17 - Arquitetura Hadoop desenhada.....	29
Figura 18 - Arquitetura tecnológica	32
Figura 19 - Diagrama de Casos de Uso	33
Figura 20 - Modelo Conceptual	34
Figura 21 - Estrutura ficheiros	35
Figura 22 - Modelo de dados de Stocks.....	35
Figura 23 -Modelo de dados Faturas.....	36
Figura 24 - Exemplo estrutura campo ComanyID.....	40
Figura 25 - Dimensão Cliente.....	41
Figura 26 - Dimensão Produto	41
Figura 27 - Dimensão Comerciante	42
Figura 28 - Dimensão Taxa	42
Figura 29 - Dimensão Tempo.....	42
Figura 30 - Tabela de factos FactFactura	43
Figura 31 - Tabela de factos Stocks	43
Figura 32 - Construção de relatório	44
Figura 33 - Relatório formatado	45
Figura 34 - Tabelas da Data Warehouse	46
Figura 35 - Registo na Dimensão Comerciante	47
Figura 36 - Registos na Dimensão Cliente.....	47
Figura 37 - Registos na Dimensão Produto	47
Figura 38 - Registo na Dimensão Taxa	48
Figura 39 - Registos na Dimensão Tempo.....	48
Figura 40 - Registos na tabela de factos Fatura	48
Figura 41 - Registo na tabela de factos Stocks	49
Figura 42 - Visualização de relatórios no Report Designer	49

Lista de tabelas

Tabela 1- Plano de ações.....	29
Tabela 2 - Descrição do tipo de taxas.....	42

Lista de abreviaturas, siglas e acrónimos

BI – *Business Intelligence*

CPU – *Central Process Unit*

DAG - *Directed Acyclic Graphs*

DBMS - *Database Management Systems*

DDL - *Data Definition Language*

DW - *Data Warehouse*

EDW - *Enterprise Data Warehouse*

ETL – *Extract transform and load*

FIFO - *First in First Out*

GFS - *Google File System*

GPS - *Global Position System*

HiveQL - *Hive Query Language*

HDFS - *Hadoop Distributed File System*

HDP - *Hortonworks Data Platform*

IBM – *International Business Machine*

I/O – *Input / Output*

JVM - *Java Virtual Machine*

NIF – *Número de Identificação Fiscal*

RDBMS - *Relational Database Management Systems*

ROI – *Return of Investment*

SAF-T - *Standard Audit File for Tax*

SGBD – *Sistemas de Gestão Base de Dados*

SQL - *Structured Query Language*

TCO - *Total Cost of Ownership*

TI - *Tecnologias de Informação*

1. Introdução

1.1 Contexto

A grande capacidade de transformar dados e análise do Big Data são uma mais-valia para as empresas. A combinação de automação de processos, integração de dados e capacidades analíticas inovadoras estão a mudar radicalmente a forma como as empresas estão a trabalhar.

O objetivo desta dissertação de mestrado é construir um demonstrador de conceito para poder responder à falta de uma solução informática sem custos para os pequenos comerciantes, que os ajude a controlar os seus *stocks* e a, analisar as suas vendas.

Esta dissertação tem o objetivo de dar uma visão geral do mercado de Big Data, soluções já existentes e os benefícios para as empresas. A solução deve ser entendida como uma forma como o Big Data pode ser utilizado e a sua ajuda para os mais distintos problemas de tratamentos com grandes volumes de dados.

Os sistemas de informação têm tido um papel relevante na deteção de fraudes e situações de evasão fiscal. O sistema *e-fatura* foi um dos mecanismos que as autoridades tributárias introduziram para ter um controlo rigoroso das transações comerciais efetuadas entre empresas e o consumidor final.

Nos pequenos comércio mesmo com o recurso a *softwares* de gestão o comerciante não os sabe manusear de forma correta, e desta forma existe alguma dificuldade em manter os *stocks* atualizados. Este problema torna-se especialmente grave no final de cada ano, quando é necessário efetuar o inventário anual. Enganos suspeitos, controlo negligente de *stocks* ou falta de organização são algum dos principais problemas que afetam um inventário.

O consumidor final não é obrigado a dar o seu número de contribuinte, mas o comerciante é obrigado a emitir a fatura pelos produtos vendidos, algo que nem sempre acontece. Se juntarmos a este cenário a falta de controlo na receção de mercadorias, muitos documentos em formato de papel o deterioramento de estes, leva a que nunca haja um controlo rigoroso e preciso sobre as entradas e saídas de *stocks*.

Big Data tornou-se uma das tecnologias mais faladas nos últimos anos na área das Tecnologias de Informação (TI). O aparecimento do Big Data teve origem no aumento repentino de dados em algumas empresas, como os dados da web, dados resultantes de simulações científicas ou de negócios ou de outras fontes de dados.

1.2 Estrutura da dissertação

A dissertação está composta por sete capítulos. O primeiro efetua um enquadramento do tema, descrevendo os motivos que levaram à realização desta

dissertação e detalhando o problema que é abordado e definindo os objetivos do trabalho.

O segundo capítulo faz um estudo extenso sobre esta tecnologia que se denomina Big Data, apresentando as suas origens, as principais ferramentas e quais as empresas pioneiras nesta tecnologia bem como o estado atual desta e a forma como a informação é tratada e manipulada. Também aborda os ficheiros SAF-T a sua estrutura e o papel que desempenham nos sistemas financeiros.

No terceiro capítulo são definidos os requisitos para a construção de uma solução que consiga resolver o problema que está a ser estudado, sendo propostos uma arquitetura tecnológica para o desenvolvimento da aplicação bem como as ferramentas que vão ser utilizadas.

O quarto capítulo é descrito o modelo de dados que vai ser a base para dar resposta ao problema estudado nesta dissertação. Um modelo bem construído é essencial em projetos que envolvem ferramentas de Business Intelligence (BI).

No quinto capítulo é apresentada toda a fase de desenvolvimento da solução, como os processos foram construídos com a ferramenta Hadoop.

O sexto capítulo são mostrados todos os resultados alcançados, após a implementação de toda a solução.

No sétimo capítulo e último capítulo é feita uma reflexão da investigação e do trabalho realizado durante a dissertação. É apresentada uma análise crítica sobre os desenvolvimentos (processos criados, relatórios), assim como diversas conclusões e uma breve projeção de trabalho a ser desenvolvido no futuro.

2. Estado da Arte

2.1 O que é o Big Data?

Vivemos num mundo digital em que o crescimento exponencial da tecnologia teve impacto nas nossas vidas e mudou drasticamente o modo como as empresas tratam os dados.

Os dados podem ser gerados das mais variadas formas e fontes desde as bases de dados relacionais, redes sociais (onde cada partilha ou “Like” no Facebook ou “tweets” no Twitter são analisados) mas também a informação recolhida de aparelhos eletrónicos como GPS ou sensores, bem com imagens, vídeos, músicas e o número de clicks em *websites*.

Big Data é um novo termo usado principalmente para descrever os conjuntos de dados que são tão grandes e complexos que requerem tecnologias avançadas exclusivas de armazenamento, gestão, análise e visualização. Chen [32] define Big Data como “Técnicas e tecnologias que tornam a manipulação de dados em escala extrema acessível”. Abordagens Big Data diferem de mineração de dados tradicional. Abordagens Big Data são capazes de seguir o fluxo de informações e análise de dados em tempo real. No entanto, como mencionado em um relatório da Forrester [32], as organizações estão atualmente utilizando fontes de Big Data e a integrar novas abordagens de análise de dados, com o intuito de alcançar uma compreensão mais profunda dos seus clientes e otimização do envolvimento do cliente.

Big data pode ser definido também pela capacidade de analisar um conjunto de dados não estruturados principalmente a partir de fontes tão diversas como web logs, media social, comunicações móveis, sensores e transações financeiras. Ela também requer a capacidade de extrair informações de dados não estruturados, ou seja, dados que carecem de um modelo predefinido (explícita ou implícita). No passado, extrair valor de dados não estruturados era um trabalho intensivo [46].

No mundo empresarial, técnicas de “Big Data” podem ser utilizadas em uma extensa série de operações que vão desde a otimização da cadeia de valor a uma fabricação mais eficiente utilizando o trabalho e explorando o contacto com os consumidores. Isto permite que as empresas exponham a variabilidade que existe que de outra forma ficaria oculta, ficando a possibilidade de realizar provas e adaptar os seus produtos a um baixo custo [47].

2.2 Origem do Big Data

O termo “Big data” parece ter sido usado pela primeira vez, da forma como é atualmente entendido, no final de 1990 [8].

1991 - Nasce a World Wide Web como a conhecemos, O protocolo *Hypertext Transfer Protocol* (HTTP) torna-se padrão para a troca de informações neste novo meio.

1997 - Michael Cox e David Ellsworth do Centro de Pesquisa Ames da NASA publicam um artigo sobre visualização onde se discutem os desafios de trabalhar com conjuntos de dados muito grande para os recursos de computação. "Chamamos a isto o problema de Big Data", escreveram eles. [9]

2001 - O Wikipedia é lançado o que revoluciona a forma como as pessoas consultam informação.

2003 - O primeiro trabalho académico foi apresentado em 2000 e publicado em 2003, por Francis X. Modelos Diebolt- "fator dinâmico Big Data para macroeconómica Medição e Forecasting. [8]

2006 - Uma ferramenta *open source* para o Big Data. Hadoop foi criada em 2006 a partir da necessidade de novos sistemas para lidar com a explosão de dados a partir da web. [10]

2008 - O número de dispositivos conectados à Internet excede a população do mundo.

2011 - O supercomputador Watson da IBM varre e analisa quatro Terabytes (200 milhões de páginas) de dados em segundos para derrotar dois jogadores humanos em "Jeopardy!" [11]

2012 - Segundo a IBM em 2012 foram gerados 2,5 Exabytes - que é 2,5 biliões de gigabytes (GB) - de dados a cada dia em 2012 e cerca de 75% dos dados são não estruturados, provenientes de fontes, tais como texto, voz e vídeo. [1]

2013 - A democratização dos dados começa. Com smartphones, tablets e Wi-Fi, os dados são gerados por todos a taxas prodigiosas. Mais indivíduos podem aceder a grandes volumes de dados públicos e a enviar dados para uso criativo.

Em 2014 por cada minuto geramos os seguintes dados: [2]

- Os utilizadores do Facebook partilham cerca de 2,5 milhões de peças de conteúdo.
- Os utilizadores do Twitter "twitaram" cerca de 300.000 vezes.
- Utilizadores do Instagram postaram quase 220 mil novas fotos.
- Utilizadores do YouTube fizeram o *upload* de 72 horas de novos conteúdos de vídeo.
- Utilizadores da Apple fizeram o *download* cerca de 50.000 apps.
- Utilizadores de e-mail enviaram mais de 200 milhões de mensagens.
- Amazon gera mais de US \$ 80.000 em vendas *online*.

Em 2015 foram gerados por minuto os dados apresentados na figura 1.

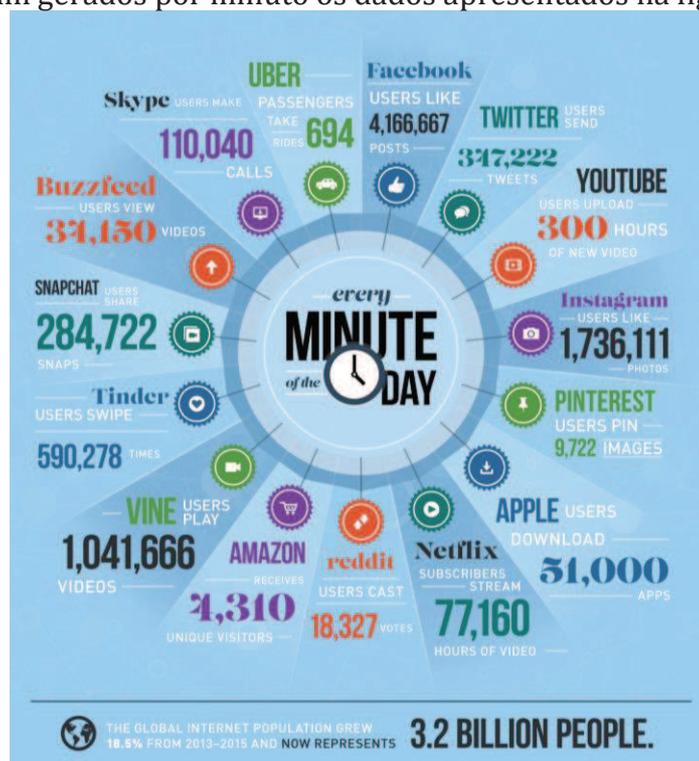


Figura 1 - Dados gerados a nível mundial [33]

2.3 Características do Big Data

O Big Data pode ser definida pelos 4 V's que representa Volume, Velocidade, Variedade e Variabilidade, sendo que existe opiniões diferentes e há quem defenda que o Big Data é definido por 7 V's correspondendo os outros três V's a Veracidade, Visualização e Valor [3].

Volume – Atualmente, vê-se o crescimento exponencial do armazenamento de dados, os dados são agora mais do que dados de texto. Podemos encontrar dados no formato de vídeo, música, imagem, sensores. É muito comum ter *Terabytes* e *Petabytes* de armazenamento para as empresas. Como as base de dados crescem as aplicações e as arquiteturas construídas para suportarem os dados precisam ser reavaliadas com bastante frequência. A rapidez com que a informação é gerada e modificada, posts em redes sociais ou *blogs* tornam-se virais em segundos.

No passado, o excessivo volume de dados foi um problema de armazenamento. Mas, com a diminuição dos custos de armazenamento, outras questões emergem, incluindo como encontrar informação relevante dentro de grandes volumes de dados e como usar o *Google Analytics* para criar valor a partir de dados relevantes [4].

Velocidade - Inicialmente, as empresas analisavam os dados utilizando um processo *batch*. Um processo leva uma quantidade de dados, envia um *Job* para o servidor e aguarda a entrega do resultado. Esse esquema funciona quando a taxa de dados de entrada é mais lenta do que a taxa de processamento do *batch* e quando o

resultado é útil, apesar do atraso. Com as novas fontes de dados, como aplicações sociais e móveis, o processo *batch* é obsoleto. Os dados agora vão diretamente para o servidor, em tempo real, de uma forma contínua e o resultado é apenas útil, se o atraso for muito curto.

Os dados fluem dentro das organizações a uma grande velocidade. A Web e os dispositivos móveis permitiram a geração de um fluxo de dados de volta para os fornecedores. As Compras *online* revolucionaram a interação entre o consumidor e fornecedor. Os Retailistas *online* podem agora manter o registo dos clientes e ter acesso a cada interação destes e podem manter o histórico e utilizar rapidamente esta informação em recomendar produtos e colocar a organização sempre atualizada. Organizações de marketing *online* estão a tirar uma enorme vantagem com a capacidade de obter conhecimento instantaneamente. Com a invenção dos *smartphones*, há ainda mais dados gerados baseados em localização e está a ser importante para ser capaz de aproveitar essa enorme quantidade de dados.

Variedade – Todos os dados gerados com os media social e digital raramente são dados estruturado. Documentos não estruturados de texto, vídeos, dados de áudio, imagens, transações financeiras, interações em *sites*, redes sociais são exemplos de dados não estruturados.

As Bases de dados convencionais suportam objetos grandes (LOB's), mas têm suas limitações se não são distribuídas. Esta informação é difícil de encaixar nas estruturas de gestão de base de dados relacionais convencionais e não é de fácil integração e leva a muitas tentativas de adaptação aos dados. E isto conduz a perdas de informações. Se os dados são perdidos, então é uma perda que não pode ser recuperada. O Big Data, por outro lado tende a manter todos os dados, pois é escrito uma vez e lido muitas vezes. Big Data acredita que poderia haver conhecimento em cada bit de dados escondidos.

Variabilidade – Para além do aumento de velocidade e variedade de dados cada vez maiores, os fluxos de dados podem ser altamente inconsistentes com picos periódicos. Diariamente, cargas de dados de pico sazonal pode ser um desafio de gestão. Ainda mais com dados não estruturados envolvidos [5]. Informação altamente inconsistente e com diferentes interpretações podem levar a erros de análise.

2.4 Arquitetura Big Data

De modo a poder ter uma visão geral sobre os componentes que fazem parte de uma arquitetura Big Data, o diagrama na figura 2 ajuda entender a relação entre os diferentes componentes.

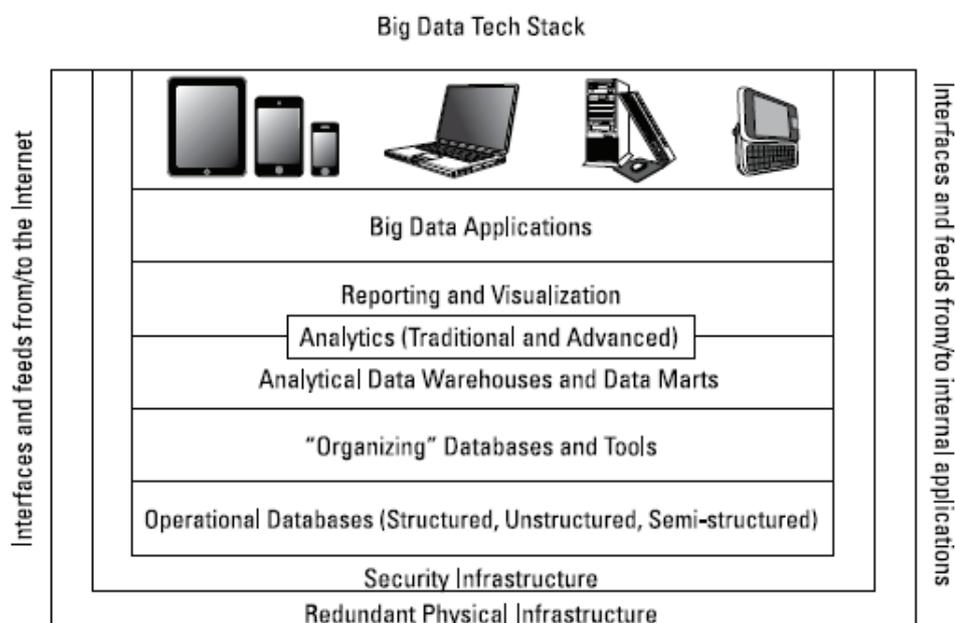


Figura 2- Arquitetura Big Data [37]

Interfaces and feed from / to the internet

As interfaces e *feed* (alimentação) ambas gerem internamente dados e alimentam dados de fontes externas. Para entender como o Big Data funciona no mundo real, é importante entender esta necessidade. As interfaces programáveis (API's) irão estar em todas as partes de uma arquitetura de Big Data. Estas interfaces irão estar presentes em cada nível da arquitetura, sem a integração de serviços o Big Data não existiria. [37]

Redundant Physical Infrastructure

A infraestrutura física é a base fundamental para qualquer Sistema de informação sem uma infraestrutura adequada, a parte dos sistemas que é executada sobre esta fica comprometida. A infraestrutura de uma arquitetura Big Data é um pouco diferente do que é utilizada para os dados tradicionais. A infraestrutura física é baseada sobre um sistema de computação distribuído, pelo qual os dados podem estar em diferentes partes e podem ser juntos através da rede. [37]

A redundância é importante porque estamos a lidar com muitos dados de diferentes fontes, a redundância pode ser definida como, uma regra de segurança criada no sistema de armazenamento para que se possa ter a mesma informação em diferentes sítios, daí ser redundante, a repetição dessa informação é útil quando acontecem falhas nos sistemas de informação e podem ser utilizadas como uma *backup*.

Infraestrutura de segurança

O mais importante em projetos de Big data para as empresas será a segurança e a privacidade dos dados. Os requisitos de segurança têm de estar estreitamente

alinhados com as necessidades específicas do negócio. Alguns desafios surgem quando o Big Data torna-se parte da estratégia:

Acesso aos dados - O acesso do utilizador aos dados não tratados ou aos dados finais (tratados) têm aproximadamente o mesmo nível de requisitos técnicos como as implementações que não são de Big Data. Os dados devem estar disponíveis apenas para aqueles que têm uma necessidade de negócio legítima para examinar ou interagir com estes. [38]

Acesso à aplicação - O acesso a dados de aplicações também é relativamente simples do ponto de vista técnico. A maioria das programming interfaces (API's) oferecem proteção contra o uso ou acesso não autorizado. Esse nível de proteção é provavelmente adequado para a maioria das implementações de Big Data. [38]

Encriptação de dados - A encriptação de dados é um dos aspetos mais desafiantes a nível de segurança em um ambiente de Big Data. Em ambientes tradicionais, encriptar e desencriptar dados realmente stressa os recursos dos sistemas. Este problema é agravado com o Big data. A abordagem mais simples é aumentar a capacidade computacional para que seja mais rápida. [38]

Fontes de dados Operacionais

Tradicionalmente, uma fonte de dados operacional consistia em dados altamente estruturados geridos pelo negócio em uma base de dados relacional. Mas, como o mundo muda, é importante compreender que os dados operacionais têm agora de abranger um amplo conjunto de fontes de dados, incluindo as fontes de dados não estruturados, tais como de clientes e dados sociais em todas as suas formas.

2.5 Big Data nas empresas

É cada vez mais imperativo para as organizações minerar os dados para se manterem competitivos. Os dados, quando analisados corretamente levam a uma riqueza de informações que ajuda as empresas a redefinir estratégias. No entanto, o volume atual de conjuntos de Big Data é muito complicado de ser gerido e processado por bases de dados relacionais convencionais e de DW.

A pressão para lidar com a quantidade de dados que cresce na *web*, levou a Google a desenvolver o *Google File System* e MapReduce. Foram feitos esforços para reconstruir essas tecnologias como *software open source*.

Isto resultou no Apache Hadoop e Hadoop File System e lançou as bases para tecnologias como Big Data. *Hardware* de qualidade, arquiteturas em *cloud* e *software open source*, hoje trazem um grande processamento de dados ao alcance daqueles que têm menos recursos. O Processamento de grandes quantidades de dados é eminentemente factível, mesmo para as pequenas *Startups* de garagem, que podem alugar com custos reduzidos um servidor na *cloud*. [27]

Apesar do tamanho, velocidade ou fonte dos dados, Big Data impulsiona a necessidade de fazer sentido a partir da confusão, impulsiona a necessidade de encontrar significado nos dados que estão em constante mudança e para encontrar relações entre os dados criados. Entender estes cruzamentos de dados e ser capaz de recolher a informação escondida em Big Data desbloqueia o valor do Big Data, que só pode ser adquirida por ser capaz de resolver os nossos próprios desafios do Big Data.

Executar análises com o detalhe mais fino possível e enquanto ainda se consegue ter dados suficientes para que os resultados sejam significativos e precisos para tomar decisões com maior precisão e, por sua vez mais lucros para a empresa e os clientes.

2.6 Principais Players

O Hadoop é integrado em uma série de plataformas criadas por distintas empresas, são disponibilizadas dois tipos de versões, a versão livre e a comercial. Na versão comercial é assegurado suporte técnico da plataforma, o Hadoop é personalizado e otimizado de forma a ser uma ferramenta mais fácil de interagir e alcançar melhores resultados.

Segundo a Forrester [39] estima que 100% das grandes empresas irão adotar o Hadoop nos próximos dois anos [40].

Atualmente os principais provedores no mercado do Hadoop são a Cloudera, Hortonworks, MapR e IBM sendo que a empresa Pivotal Software também já é um forte concorrente e uma opção a ter em conta quando falamos de soluções empresariais do Hadoop.

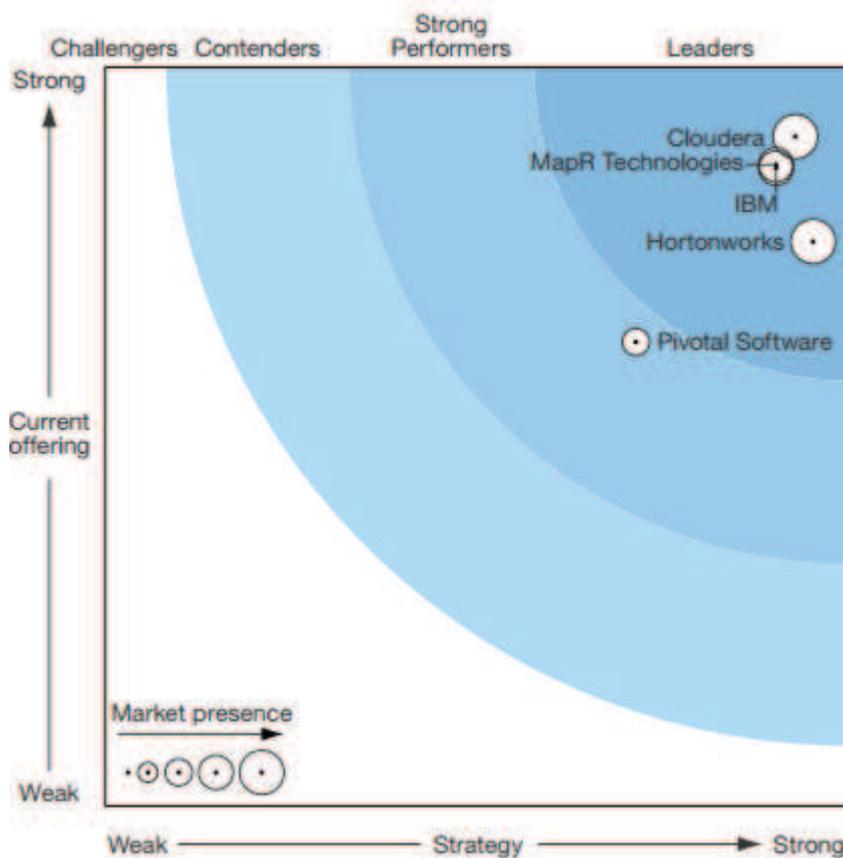


Figura 3- Big Data Plataformas [40]

2.6.1 Hortonworks Data Platform

Hortonworks Data Platform (HDP) é uma plataforma Apache Hadoop 100% Open Source. HDP oferece todos os projetos relacionados com o Apache Hadoop necessários para integrar o Hadoop ao lado de uma Enterprise Data Warehouse (EDW) como parte de uma arquitetura de dados modernos [42].

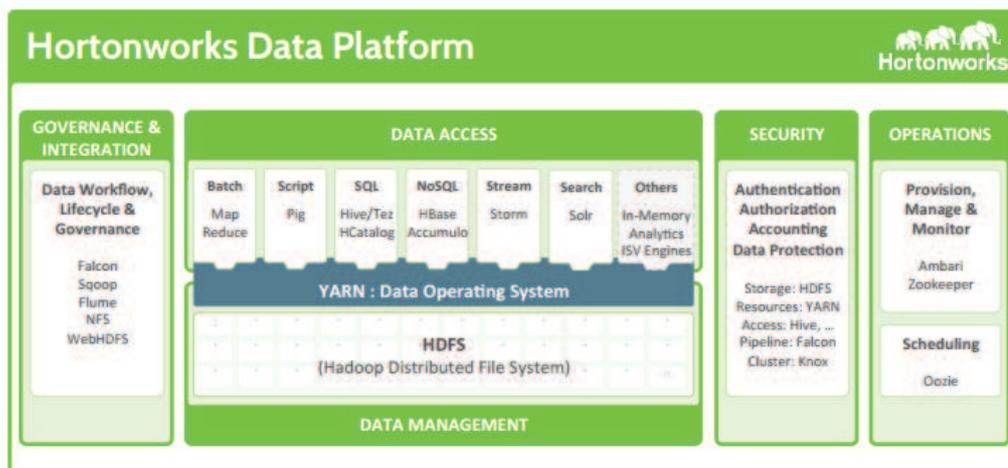


Figura 4 - Hortonworks Data Platform

2.6.2 Cloudera

Cloudera é um dos *players* mais conhecido no que envolve o Hadoop . Com um número de clientes considerável e com uma contribuição ativa para o desenvolvimento do Hadoop , o Cloudera está no top 3 da lista quando se trata de construir ferramentas inovadoras. O Cloudera Manager é fácil de usar e implementar, com uma interface para o utilizador bastante acessível, apresentando todas as informações de forma organizada e limpa. O Cloudera automatiza o processo de instalação e também presta outros serviços avançados para os utilizadores.

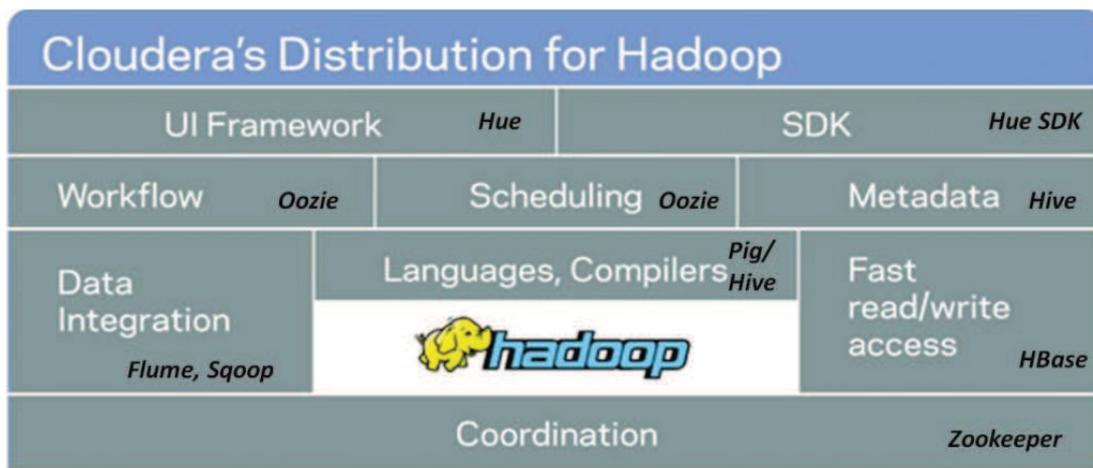


Figura 5 - Cloudera Distribution for Hadoop [43]

2.6.3 MapR

MapR é uma distribuição de nível empresarial para o Apache Hadoop. O MapR Converged Data Platform foi projetado para melhorar a fiabilidade, desempenho e facilidade de uso do Hadoop. A distribuição MapR fornece uma *stack* completa do Hadoop, que inclui o MapR File System (MapR-FS), o sistema de gestão de base de dados MapR-DB NoSQL, MapR Streams, a interface do utilizador MapR Control System (MCS), e o ecossistema completo do Hadoop. O MapR pode ser utilizado com o Apache Hadoop, HDFS e API's MapReduce.

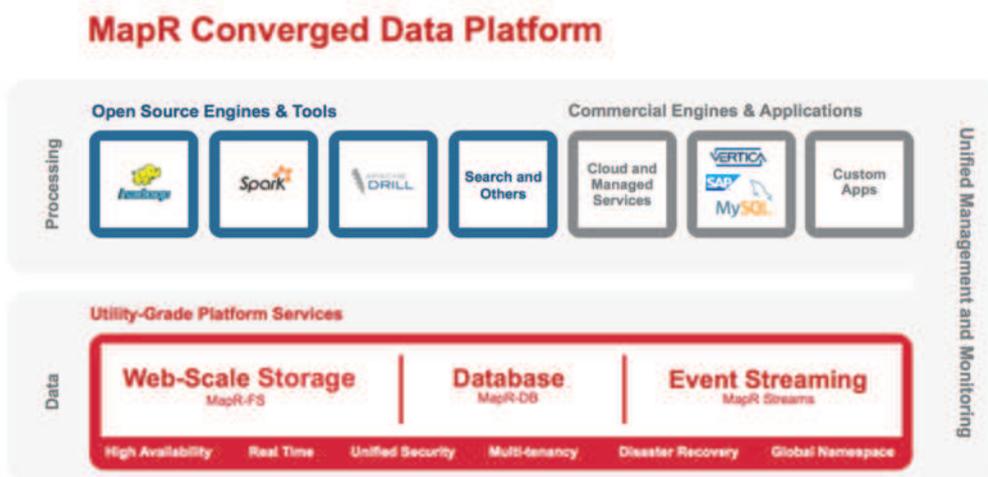


Figura 6- MapR Converged Data Platform [44]

Como diferença principal em relação ao Cloudera e Hortonworks, o MapR tem uma forma distinta para armazenar os metadados no processamento dos nós, pois depende de um sistema de ficheiros diferente que é o MapR-FS e não utiliza a arquitetura *NameNode*.

O tecnólogo Robert D. Schneider [41] resumizou as diferenças entre Cloudera, Hortonworks e MapR.

	Hortonworks	Cloudera	MapR
Performance and Scalability			
Data Ingest	Batch	Batch	Batch and streaming writes
Metadata Architecture	Centralized	Centralized	Distributed
HBase Performance	Latency spikes	Latency spikes	Consistent low latency
NoSQL Applications	Mainly batch applications	Mainly batch applications	Batch and online/real-time applications
Dependability			
High Availability	Single failure recovery	Single failure recovery	Self healing across multiple failures
MapReduce HA	Restart jobs	Restart jobs	Continuous without restart
Upgrading	Planned downtime	Rolling upgrades	Rolling upgrades
Replication	Data	Data	Data + metadata
Snapshots	Consistent only for closed files	Consistent only for closed files	Point-in-time consistency for all files and tables
Disaster Recovery	No	File copy scheduling (BDR)	Mirroring
Manageability			
Management Tools	Ambari	Cloudera Manager	MapR Control System
Volume Support	No	No	Yes
Heat map, Alarms, Alerts	Yes	Yes	Yes
Integration with REST API	Yes	Yes	Yes
Data and Job Placement Control	No	No	Yes
Data Access			
File System Access	HDFS, read-only NFS	HDFS, read-only NFS	HDFS, read/write NFS (POSIX)
File I/O	Append only	Append only	Read/write
Security: ACLs	Yes	Yes	Yes
Wire-level Authentication	Kerberos	Kerberos	Kerberos, Native

Figura 7- Diferenças entre MapR, Hortonworks e Cloudera [45]

2.7 Business Intelligence & Analytics e Big Data

2.7.1 Business Intelligence & Analytics

Business Intelligence (BI) é a recolha, gestão, análise e partilha de informação, com o objetivo de obter dados que podem ser usados para tomar melhores decisões. BI

transforma informação em inteligência, a inteligência em conhecimento e conhecimento em sabedoria de negócio. Combinando técnicas avançadas, tais como armazenamento de dados, mineração de dados e apoio à decisão, soluções BI oferecem a capacidade de transformar a informação em sistemas de gestão de relacionamento com grandes clientes que podem ajudar a criar relacionamentos mais fortes, mais rentáveis, identificar novas oportunidades de negócios e até mesmo antecipar as procuras dos clientes. De uma maneira mais simplificada o BI é um conjunto de *softwares* para analisar dados em bruto de uma organização, para tomar decisões inteligentes para o sucesso do negócio.

O objetivo do BI é fornecer aos gestores, informações sobre o negócio a tempo de permitir-lhes tomar decisões que podem resolver problemas ou aproveitar oportunidades. Não se limita a fornecer aos decisores informações sobre eventos, um sistema de BI permite-lhes explorar os dados subjacentes, com o objetivo de entender melhor o problema. Em outras palavras, o BI permite que os gestores administrem melhor.

Algumas pessoas confundem BI com o “*Analytics*”. O termo “*Analytics*” pode ser definido segundo Davenport and Harris [20] como o uso extensivo de dados, análise estatística e quantitativa, modelos explicativos e preditivos e gestão baseada em factos para orientar decisões e ações.

A análise analítica (*Analytics*) é uma parte importante do apoio à decisão para os decisores. As análises são os resultados da interação entre os modelos e os dados. As análises são fornecidas ao decisor para o ajudar a tomar uma decisão. Pode querer uma análise por causa de sua simplicidade e da sua capacidade de previsão, a análise não é a única saída de um sistema de apoio à decisão.

Os modelos analíticos funcionam melhor para a previsão de curtos períodos para o futuro onde se acredita que podem ser semelhantes ao passado. Decisões de longo alcance e decisões em um ambiente turbulento não são bons candidatos para modelos analíticos e de previsão, modelos qualitativos devem ser utilizados para completar a visão da situação.

2.7.2 Componentes de um Sistema de Business Intelligence

Muitas organizações utilizam hoje um conjunto de ferramentas e aplicações de Business Intelligence (BI), para permitir que peritos recolham informações a partir de uma variedade de fontes, analisem e partilhem com os gestores e funcionários. No entanto, é cada vez maior a dinâmica empresarial e o aumento da concorrência o que significa que as empresas agora exigem um nível muito mais elevado do valor de seus investimentos em BI.

O BI deve agora ajudar a impulsionar o crescimento rentável, mudança, e muitas outras metas de desempenho operacional e financeiro. Não só o BI precisa entregar *Return on Investment* (ROI) mas também precisa ser implementada uma forma que

minimize o *Total Cost of Ownership* (TCO), para tal é necessário ter uma arquitetura adequada ao negócio e estruturada de forma a não falhar.

A arquitetura de um sistema de Business Intelligence genericamente tem três camadas padrão (Figura 2), a camada das fontes de dados, a camada de armazenamento dos dados centralizados e a camada de exploração de dados e é nesta última que o utilizador final navega sobre dados e consulta ou constrói relatórios, conjuntos de indicadores.

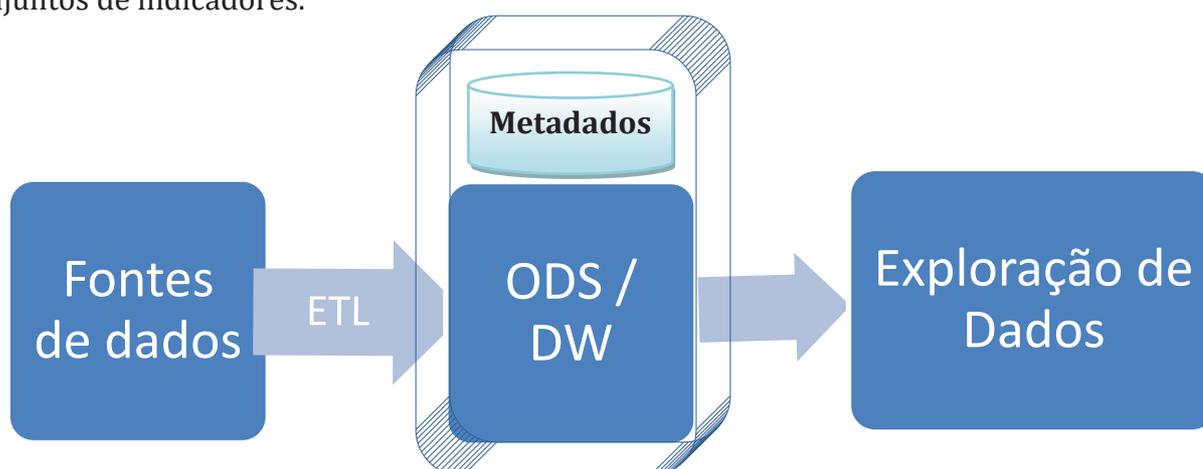


Figura 8 - Arquitetura BI de 3 camadas

Na figura 3 está representada uma abordagem de cinco camadas em um sistema de BI, esta arquitetura proposta por Lih Ong, Hwa Siew, Fan Wong no *paper A Five-Layered Business Intelligence Architecture* [21], defende que o processo ETL e a DW estejam em camadas separadas e adicionam uma camada de Metadados que consiste em Metadados para cada componente dentro do sistema de BI e, assim, separa os dados, definições dos dados e regras de negócio sobre os dados.

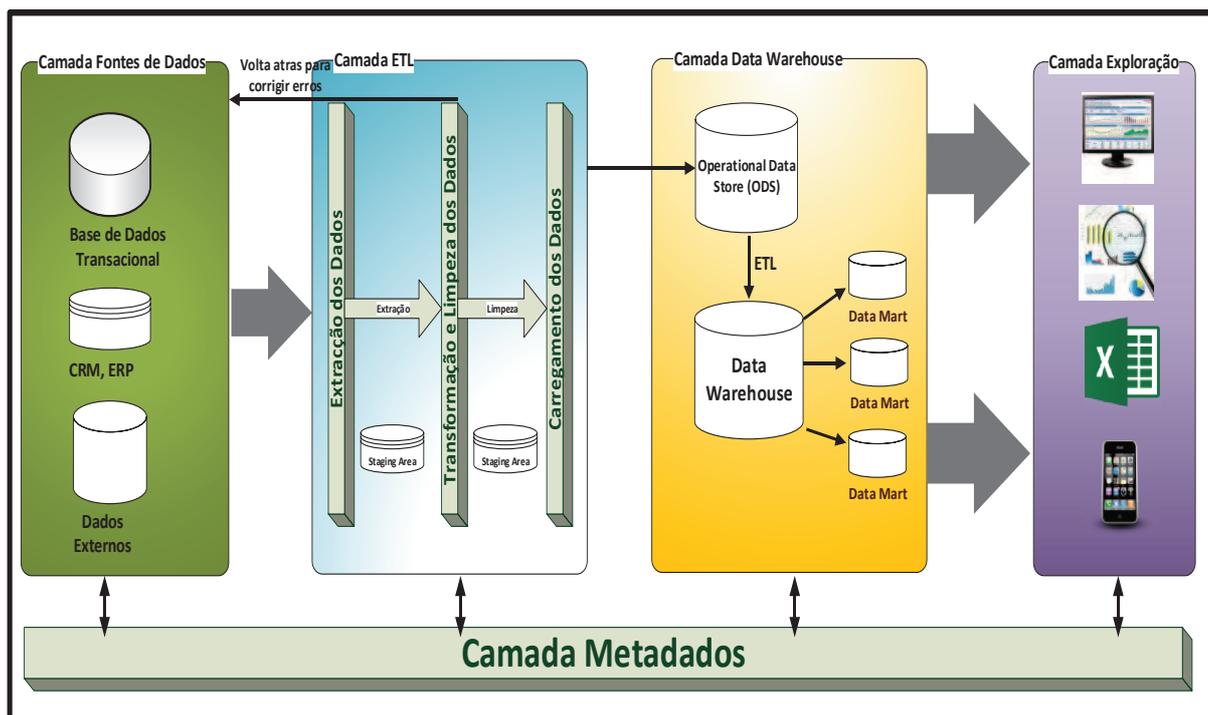


Figura 9- Sistema de BI com cinco camadas.

2.8 Ferramentas

2.8.1 Google Big Table

Google usa como armazenamento de dados um mecanismo chamado Bigtable, é distribuído, de mapa persistente de ordenação multidimensional, armazenamento de dados orientado para a coluna, foi criado pelo Google [16]. Para lidar com grandes quantidades de dados estruturados associados às operações de pesquisa na Internet e de serviços Web da empresa. Google construiu o Bigtable principalmente para aplicações internas, e o disponibilizou externamente como um repositório de dados para os clientes que usam a plataforma *Google App Engine* como uma oferta de serviço.

Bigtable é projetado para escalar de forma confiável *Petabytes* de dados e milhares de máquinas. Bigtable alcançou vários objetivos: aplicabilidade ampla, escalabilidade, alta performance e alta disponibilidade [16]. O Bigtable não suporta o modelo de dados relacional.

2.8.2 Hadoop

A biblioteca de *software* Apache Hadoop é um *framework* que permite o processamento distribuído de grandes conjuntos de dados em *clusters* de computadores que utilizam modelos de programação simples.

Ele é projetado para se expandir a partir de um único servidor para milhares de máquinas, cada uma oferecendo computação e armazenamento local. Em vez de

confiar no *hardware* para proporcionar uma alta disponibilidade, a biblioteca em si é concebida para detetar e tratar falhas na camada de aplicação, de modo a fornecer um serviço altamente disponível no topo de um conjunto de computadores, cada um dos quais pode ser propenso a falhas. [6]

Hadoop usa a *framework* MapReduce introduzida pelo Google, aproveitando o conceito de *mapa(map)* e *reduz(reduce)* funções conhecidas na programação funcional [7]. Hadoop permite aos utilizadores armazenar e processar grandes volumes de dados e analisá-lo de maneiras que não era possível anteriormente com soluções menos escaláveis ou abordagens baseadas em SQL.

Hadoop está a tornar-se uma parte crítica de muitos departamentos modernos de sistemas de informação. Está a ser utilizado como requisito, incluindo no *analytics*, armazenamento de dados, processamento de dados e recursos de computação compartilhados [25].

O crescimento significativo é importante que seja tratado como uma componente da organização de TI, e gerido como tal. O Hadoop não deve ser relegado apenas para projetos de pesquisa, deve ser gerido como uma empresa iria gerir qualquer outro grande componente de sua infraestrutura de TI. [25]

Assim, o *core* do Hadoop é constituído pelo *Hadoop Distributed File System (HDFS)* e Hadoop MapReduce. Normalmente, os dois, HDFS e MapReduce, são implementados juntos em um *cluster*. Os dados de entrada para o MapReduce precisam de ser armazenados em uma instância HDFS no mesmo *cluster* e os *outputs* são escritos lá.

2.8.3 O Ecossistema do Hadoop

A plataforma Hadoop consiste em dois serviços essenciais: um sistema de ficheiros confiável, distribuído chamado HDFS e o mecanismo de processamento de dados em paralelo de alto desempenho chamado Hadoop MapReduce, que vai ser descrito detalhadamente um pouco mais abaixo. Hadoop foi criado por Doug Cutting. Os vendedores que fornecem plataformas baseadas em Hadoop incluem-se *Cloudera*, *Hortonworks*, *MapR*, *IBM*, *Amazon* entre outros.

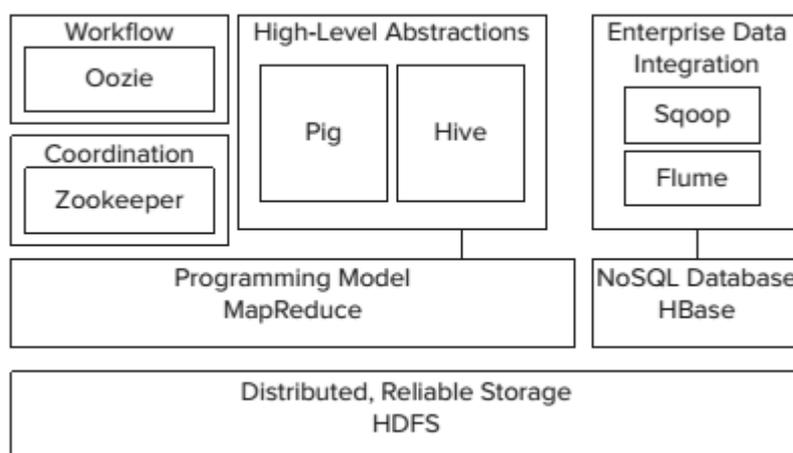


Figura 10 - Ecossistema do Hadoop [14]

2.8.3.1 Hadoop Distributed File System

O armazenamento de ficheiros é chamado HDFS. O HDFS fornece um armazenamento escalável tolerante a falhas e de baixo custo. O HDFS deteta e contorna problemas de *hardware*, incluindo problemas de disco e falhas do servidor.

HDFS é construído em torno da ideia de que o padrão de processamento de dados mais eficiente é escrever uma vez e ler muitas. Um conjunto de dados é normalmente gerado ou copiado da fonte, em seguida, várias análises são realizadas no conjunto de dados ao longo do tempo. Cada análise envolve uma grande parte, senão a totalidade, do conjunto de dados, de modo que o tempo para ler todo o conjunto de dados é mais importante do que a latência na leitura do primeiro registo.

HDFS armazena ficheiros através de uma coleção de servidores em *cluster*. Os ficheiros são decompostos em blocos, e cada bloco é escrito em mais de um (o número é configurável, mas três é comum) dos servidores. Essa replicação fornece tolerância a falhas (perda de um único disco ou servidor não destrói um arquivo de dados) e performance (qualquer bloco pode ser lido a partir de um de vários servidores, melhorando o rendimento do sistema).

HDFS garante a disponibilidade de dados, monitorizando continuamente os servidores em um *cluster* e os blocos que gerem. Blocos individuais incluem *checksums*. Quando um bloco é lido, o *checksum* é verificado, e se o seu bloco foi danificado será restaurado a partir de uma de suas réplicas. Se um servidor ou disco falhar, todos os dados armazenados são replicados para outro nó ou os nós do *cluster*, a partir da coleção de réplicas. Como resultado, o HDFS funciona muito bem em *hardware commodity*.

Tolera, e compensa, as falhas no *cluster*. Como os *Cluster* são grandes servidores tolerantes a falhas, mesmo muito caros tendem a falhar.

2.8.3.2 NameNode e DataNodes

HDFS tem uma arquitetura *master / slave*. Um *cluster* HDFS consiste em um único *NameNode*, um servidor *master* que gere os *Namespaces* do sistema de ficheiros e regula o acesso aos ficheiros pelos clientes. Além disso, há uma série de *DataNodes*, geralmente um por nó no *cluster*, que gerem o armazenamento ligado aos nós em que eles estão a ser executados. O HDFS expõe um *namespace* do sistema de ficheiros e permite que os dados do utilizador sejam armazenados em ficheiros. Internamente, um ficheiro é dividido em um ou mais blocos e esses blocos são armazenados em um conjunto de *DataNodes*. O *NameNode* executa o sistema de ficheiros de operações *namespace* tal como abrir, fechar e renomear arquivos e diretorias. Também determina o mapeamento dos blocos para os *DataNodes*. Os *DataNodes* são responsáveis por servir os pedidos de leitura e escrita a partir de clientes do sistema de ficheiros. Os *DataNodes* também fazem a criação de blocos, exclusão e replicação mediante a instrução do *NameNode*. [12]

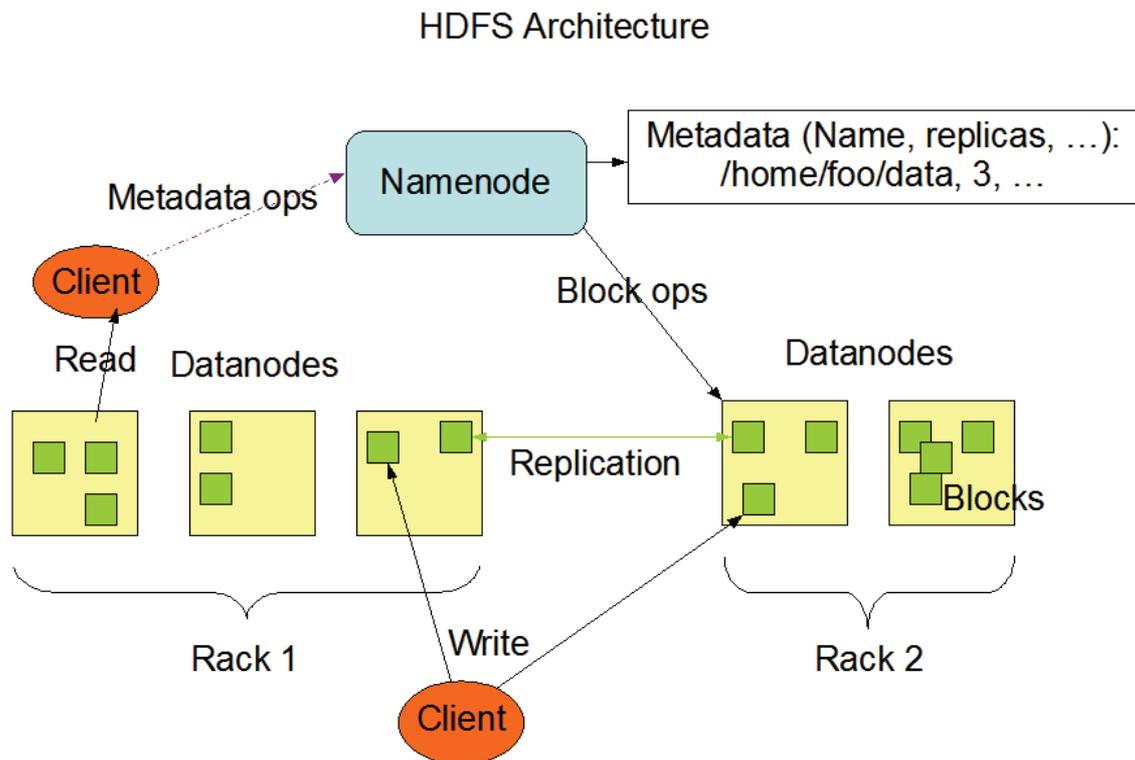


Figura 11 - Arquitetura HDFS

2.8.3.4 MapReduce

HDFS é barato, confiável e disponibiliza armazenamento de ficheiros. Esse serviço sozinho, porém, não seria suficiente para criar nível de interesse, ou conduzir à taxa de adoção, que caracterizam o Hadoop ao longo dos últimos anos. O segundo componente principal do Hadoop é o sistema de processamento de dados em paralelo chamado MapReduce.

Hadoop MapReduce é um *framework* de *software* para escrever facilmente aplicações que processam grandes quantidades de dados em paralelo em grandes *Clusters* (milhares de nós) em *hardware commodity*, de forma confiável e tolerante a falhas [13]. O Hadoop MapReduce é um *framework* de programação modelada após o *paper* MapReduce do Google [15] e, normalmente, implementado sobre uma instância do HDFS.

2.8.3.5 Arquitetura do MapReduce

O modelo do MapReduce baseia-se no conceito de programação paralela. Com base na noção de programação paralela, o processamento é decomposto em n sub entidades que são executadas simultaneamente. As instruções para cada sub entidade são executadas simultaneamente em diferentes *CPUs*. Dependendo da configuração da infraestrutura de TI, os *CPUs* devem estar disponíveis no mesmo sistema de servidores ou em nós remotos que são acedidos através de uma ligação de rede. Um *Job* do MapReduce geralmente divide os conjuntos de dados de entrada em pedaços menores que são processados pela tarefa do mapa de uma forma completamente paralela. Os *Outputs* obtidos a partir de todas as tarefas do mapa são então classificados pela estrutura e disponibilizados como *Inputs* para as tarefas de redução.

2.8.3.6 Arquitetura Hadoop Map Reduce

O Hadoop MapReduce usa uma arquitetura *master-slave*. O *Master* também é conhecido como *JobTracker*, é executado num único nó, enquanto o *slave*, também chamados *TaskTrackers*, são executados nos restantes nós. Antes de o cliente MapReduce poder enviar um *job* para o *JobTracker*, ele precisa armazenar todos os dados de entrada necessária para o sistema de arquivos subjacente (tipicamente o HDFS) e calcular se divide para o *job*. O *JobTracker* representa um programa centralizado que mantém o controlo dos nós escravos, e fornece uma infraestrutura de interface para envio de *jobs*. O *TaskTracker* executa em cada um dos nós escravo em que os dados reais são normalmente armazenados. Em outras palavras, o *JobTracker* reflete o ponto de interação entre os utilizadores e a plataforma do Hadoop. Os utilizadores enviam os *Jobs* de MapReduce para o *JobTracker*, que insere os *jobs* na fila de *jobs* pendentes e executa-os (normalmente) em uma base *First in First Out* (FIFO).

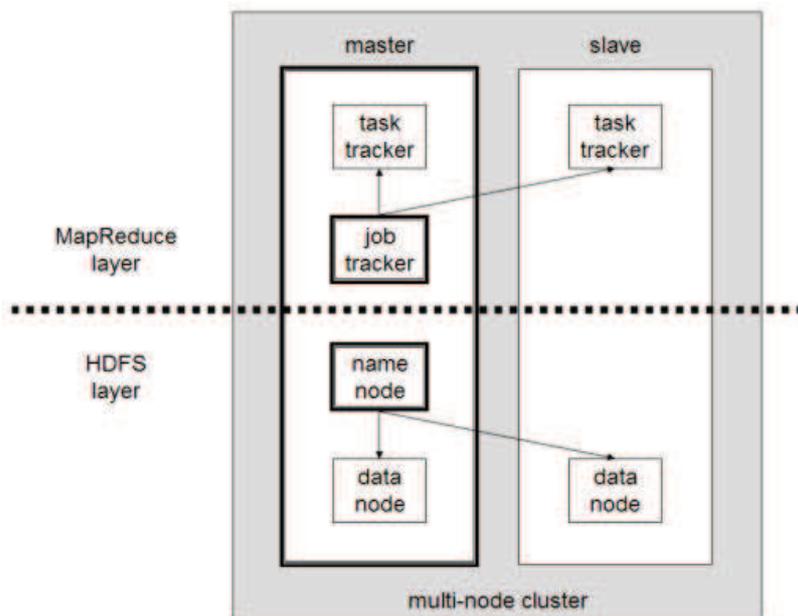


Figura 12- Arquitetura Master / Slave do MapReduce [36]

Vantagens

Simple e fácil de usar - O modelo MapReduce é simples, mas expressivo. Com o MapReduce, um programador define o seu *job* apenas com mapear e reduzir as funções, sem ter que especificar a distribuição física do *job* através de nós.

Flexível - O MapReduce não tem qualquer dependência do modelo de dados ou do esquema. Com o MapReduce um programador pode lidar com dados não estruturados irregulares mais facilmente do que eles fazem com o SGBD.

Independente do armazenamento - MapReduce é, basicamente, independente das camadas de armazenamento subjacentes. Assim, o MapReduce pode trabalhar com camadas de armazenamento diferentes, como o BigTable [16] e outros.

Tolerante a Falhas - MapReduce é altamente tolerante a falhas. Por exemplo, é relatado que MapReduce pode continuar a trabalhar apesar de uma média de 1,2 falhas por trabalho de análise pelo Google [17].

Alta escalabilidade - A melhor vantagem do uso de MapReduce é alta escalabilidade.

Desvantagens

Um problema com o encadeamento das etapas do MapReduce é, que os dados são automaticamente persistidos após cada *task* de redução e precisa ser acessado no HDFS como *input* para o próximo MapReduce. Isso cria sobrecarga de I/O e pode retardar o processamento. Os *Joins* de vários conjuntos de dados são complicados e lentos, e não têm índices. Muitas vezes, um conjunto de dados inteiro é copiado no processo.

2.8.3.7 YARN

MapReduce passou por uma reforma completa no Hadoop-0.23 e agora chama-se MapReduce 2.0 (MRv2) ou YARN.

A ideia fundamental do MRv2 é dividir as duas principais funcionalidades do *JobTracker*, gestão de recursos e a calendarização / monitorização de *jobs*. A ideia é ter um *ResourceManager* (RM) global e *ApplicationMaster* (AM) por aplicativo. Um aplicativo é um único trabalho no sentido clássico MapReduce *jobs* ou um DAG jobs [16].

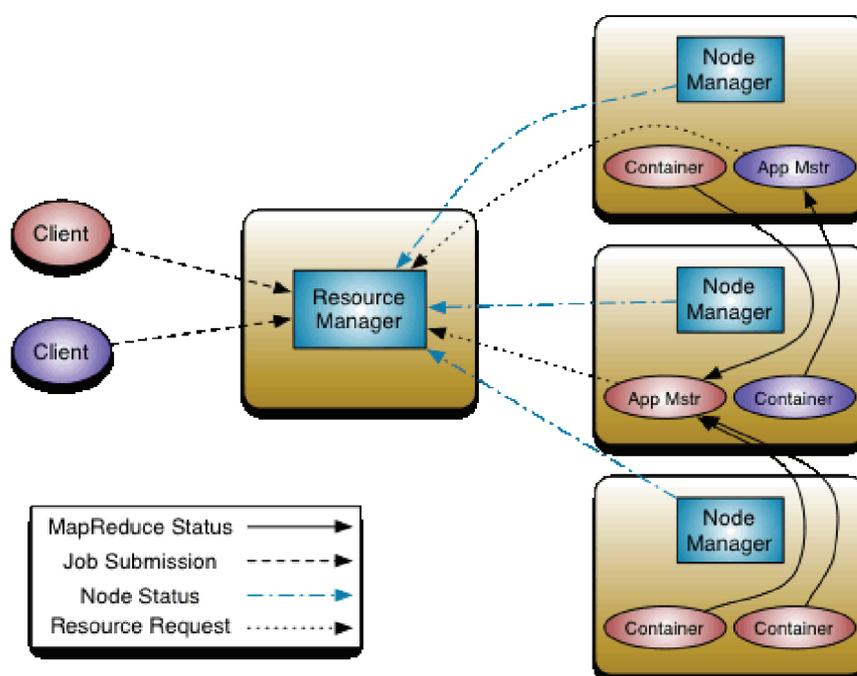


Figura 13- Arquitetura YARN

O *ResourceManager* tem dois componentes principais: *Scheduler* e o *ApplicationsManager*.

O *Scheduler* é um puro programador no sentido de que ele executa qualquer acompanhamento ou monitorização de *status* para os aplicativos. Além disso, ele não oferece garantia sobre como reiniciar tarefas que falharam ou devido a falhas de falha de *hardware* ou de aplicação. O *Scheduler* executa a sua função de programação com base nos requisitos de recursos das aplicações, fá-lo com base na noção abstrata de um *Container* de recursos que incorpora elementos como memória, *CPU*, disco, rede, etc. Na primeira versão, apenas a memória é suportada. [16]

O *ApplicationsManager* é responsável por aceitar pedidos dos *jobs*, a negociação do primeiro *Container* para a execução da aplicação específica do *ApplicationMaster* e fornece o serviço para reiniciar o *Container* do *ApplicationMaster* em caso de falha.

O *NodeManager* é o agente da *framework* por máquina que é responsável pelos *containers*, monitorizando a utilização de recursos (*CPU*, memória, disco, rede) e reportar os mesmos para o *ResourceManager / Scheduler*.

Por aplicação o *ApplicationMaster* tem a responsabilidade de negociar os recursos dos *containers* apropriados para o *Scheduler*, de pesquisar o seu estado e monitorizar o seu progresso.

2.8.3.4 HIVE

O Apache Hive é um *software* de DW que facilita a consulta e gestão de grandes conjuntos de dados que residem em armazenamento distribuído. [18]

O Hive permite escrever instruções de HiveQL (*Hive Query Language*) que é bastante parecida com o SQL que é usado nas bases de dados relacionais, desta forma permite analisar grandes conjuntos de dados armazenados em ficheiros do Hadoop.

O Hive tem três funções principais: a sumarização, a consulta (*query*) e a análise. As instruções HiveQL são partidas pelo serviço Hive em *jobs* MapReduce e executado através de um *cluster* Hadoop.

O SQL contém várias características como sub consultas, *joins – inner, left outer, right outer, outer joins, cartesian products, group bys* e agregações, *union all, create table as select* de funções úteis seleccione e muitas outras funções que tornam o HiveQL muito parecido ao SQL, esta forma qualquer pessoa que já esteja familiarizada pode começar a fazer *queries* [19]. No entanto existem algumas diferenças em relação ao SQL, como por exemplo, só se pode igualar utilizando a sintaxe ANSI do *join* tal como:

```
SELECT t1.a1 as c1, t2.b1 as c2
FROM t1 JOIN t2
ON (t1.a2 = t2.b2);
```

Em vez de sintaxe mais utilizada:

```
SELECT t1.a1 as c1, t2.b1 as c2
FROM t1, t2
WHERE t1.a2 = t2.b2;
```

A arquitetura do Hive é composta pelos seguintes componentes:

- **Metastore** - O componente que armazena o sistema de catálogos e metadados de tabelas, colunas, partições etc.
- **Driver** - O componente que gere o ciclo de vida de um *statement* HiveQL dentro do Hive.
- **Query Compiler** - O componente que compila HiveQL em um DAG de tarefas do *map reduce*.
- **Execution Engine**- O componente que executa as tarefas produzidas pelo compilador pela melhor ordem. O *execution engine* interage com a instância do Hadoop.
- **HiveServer** - O componente que proporciona uma *thrift interface* e um servidor JDBC / ODBC e fornece uma forma de integrar Hive com outras aplicações.

- Componentes cliente como a linha de comandos para o Hive, interface Web e JDBC/ODBC drivers.
- Interfaces de extensibilidade que incluem as interfaces SerDe e ObjectInspector, bem como a UDF (User Defined Function) e UDAF (User Defined Aggregate Function) interfaces que permitem que os utilizadores definam suas próprias funções personalizadas [19].

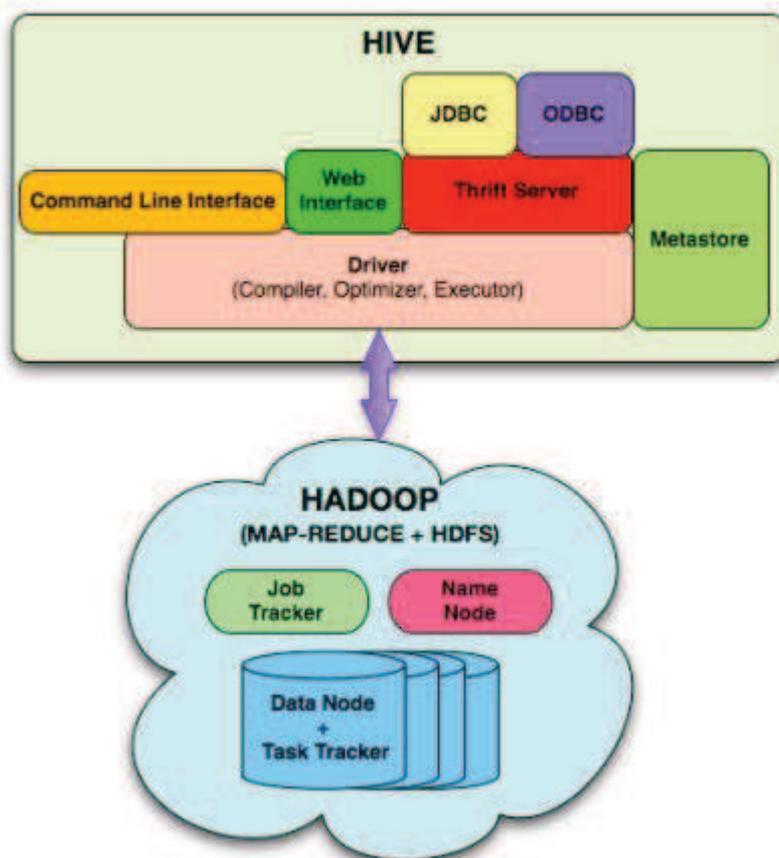


Figura 14 - Arquitetura do Hive [19]

2.9 Bases de Dados [22]

Existem diferentes tipos de produtos de base de dados também conhecidas como SGDB os tipos de base de dados mais conhecidos são a relacional, hierárquica e de rede. As mais utilizadas atualmente são as bases de dados relacionais, em inglês *Relational Database Management Systems* (RDBMS).

Nos primeiros sistemas *Database Management Systems* (DBMS) era necessário que os dados fossem estruturados da melhor forma para serem armazenados e acedidos Os dados eram armazenados em bases de dados que foram ligados a dados relacionais via ponteiros. Embora a velocidade de acesso fosse boa, o acesso aos dados era pouco flexível.

Uma das principais funções de uma DBMS é manter as definições dos dados para cada tabela e colunas na base de dados. Cada pedaço de dados deve ser atribuído um

nome, um tipo de dados e um estatuto obrigatório / opcional. A maioria dos pacotes DBMS irá impor essas regras quando linhas são adicionadas à base de dados.

2.9.1 Bases de dados Relacionais e NoSQL

Bases de dados relacionais têm uma posição de longa data na maioria das organizações, e por boas razões [34]; elas são apoiadas por um extenso ecossistema de ferramentas, e há uma grande reserva de mão-de-obra qualificada para implementação e manutenção desses sistemas. Mas as empresas estão cada vez mais ponderando alternativas para infraestrutura relacional. Em alguns casos, a motivação é técnica, como uma necessidade de escalar ou realizar além das capacidades dos seus sistemas existentes, enquanto em outros casos, as empresas são impulsionadas pelo desejo de identificar alternativas viáveis para o *software* caro. A terceira motivação é a agilidade ou a velocidade de desenvolvimento, como as empresas tentam-se adaptar ao mercado mais rapidamente e abraçar metodologias de desenvolvimento ágil. Esses drivers são aplicáveis tanto para aplicações analíticas e transacionais. As empresas estão a passar o *workload* para o Hadoop *offline*, as *workloads* analíticas, estão a criar aplicações *online*, operativas com uma nova classe de tecnologias de gestão de dados chamado "NoSQL".

NoSQL significa *Not Only SQL*, o que implica que ao conceber uma solução de *software* ou produto, há mais do que um mecanismo de armazenamento que podem ser utilizados de acordo com as necessidades. [23].

Bases de dados NoSQL representam uma evolução recente na arquitetura de aplicações empresariais, continuando a evolução dos últimos vinte anos. Na década de 1990, as aplicações integradas verticalmente deram lugar a arquiteturas cliente-servidor e, mais recentemente, arquiteturas cliente-servidor deu lugar a arquiteturas de aplicações *web* de três camadas. [35]

NoSQL é muitas vezes caracterizado por aquilo que não é, é ou não é apenas um sistema de gestão de base de dado relacional baseado em SQL ou não é simplesmente um RDBMS baseada em SQL. Enquanto essas definições explicar o que NoSQL não é, eles fazem muito pouco para explicar o que é NoSQL. Considere os fundamentos que conduziram à gestão de dados para os últimos 40 anos. Sistemas RDBMS e gestão de dados em larga escala têm sido caracterizados por as propriedades ACID (atomicidade, consistência, isolamento e durabilidade). [35]

2.10 SAF-T

SAF-T é um ficheiro que contém dados de contabilidade confiáveis exportados a partir de um sistema de contabilidade, por um período de tempo específico, de fácil

leitura, em virtude da sua padronização de *layout* e formato, e que está de acordo com a necessidade extensível.

Os auditores estão a enfrentar o crescente desafio de autenticação, onde os avanços na tecnologia, o crescente número de sistemas operativos, formatos de dados, *backup* e opções de retenção de ficheiros propiciam que o seu trabalho adquira uma maior complexidade. O SAF-T foi desenvolvido em resposta a este problema, tanto geral e sistemas de contabilidade modernos que são totalmente eletrónicos.

O *design* do SAF-T atualmente não abrange impostos, como folha de pagamento, em que só as informações resumidas de subsistemas podem ser representadas em SAF-T. Futuras extensões SAF podem incluir maior detalhe para fins de auditoria. [24]

O ficheiro tem como base entradas que podem ser encontradas em *General Ledger Chart of Accounts*, juntamente com o ficheiro de dados mestres para clientes e fornecedores, e detalhes de faturas, pedidos, pagamentos e ajustes. [24]

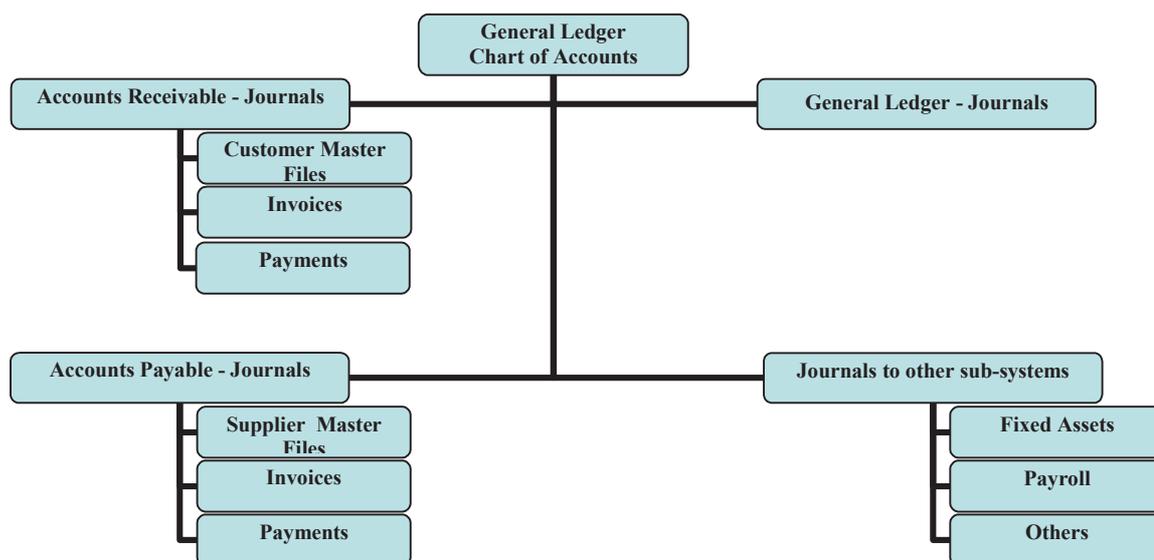


Figura 15 - Estrutura SAF-PT

SAF-T será criado a partir de qualquer entrada de dados armazenados no sistema de contabilidade original no tempo de processamento, ou a partir de uma combinação de dados de entrada e informação atual ficheiro mestre de dados no momento da entrada original.

A SAF-T deve ser criado em um formato de dados que atenda às necessidades do auditor, de preferência uma que é amplamente utilizado em todo o mundo. O fisco reconhece que formatos padrão facilitam processos de automação de auditoria. [24]

2.11 Análise final

A decisão de escolher uma plataforma de Big Data não é a mais fácil. Tendo em conta as características das 3 plataformas estudadas neste capítulo, não existe um claro vencedor. A decisão de escolher uma ou outra vai depender muito das necessidades de cada empresa.

O MapR apresenta uma distribuição Hadoop mas não tem como base o Apache Hadoop. Tendo construído MapRFS, e alcançando um produto final mais rápido do que as restantes plataformas, tem uma versão paga para os seus clientes e também disponibiliza uma versão livre para os restantes utilizadores.

O Hortonworks é a única das 3 plataformas a oferecer um produto completamente *open source* e também com uma versão nativa para Windows sendo que o único custo que pode ser associado é o suporte que disponibilizam, não oferece uma versão “fechada” como os seus concorrentes. A plataforma é bastante rápida, mas no entanto não tem nenhuma versão com *software* desenvolvido por eles próprios de forma a melhorar a utilização da plataforma tal como os seus concorrentes, a interface Ambari que utiliza não tem opções tão úteis como a do MapR e Cloudera, mantém a infraestrutura do Hadoop *in-house*.

O Cloudera é o *player* com mais clientes e a primeira empresa de distribuições Hadoop, tal como Hortonworks também é *open source* e tem como base o Apache Hadoop, fornece como o MapR também uma versão própria para os clientes que pagam licenças e uma versão livre para os restantes utilizadores. Tem o *Cloudera Manager* que é fácil de utilizar e automatiza processos de instalação e inicialização de serviços.

Desta forma a plataforma escolhida para ser utilizada é o Cloudera, porque oferece uma interface de acesso muito intuitiva e *user friendly* evitando ao utilizador final uma série de problemas e contratemplos, também disponibiliza uma versão livre com bastante qualidade, a versão que o MapR disponibiliza podia também ter sido a escolha final mas visto que não utiliza o HDFS de forma nativa em detrimento do MapRFS sendo este ultimo propriedade da empresa ficando o utilizador final dependente do sucesso deste componente, o mesmo não acontece com o HDFS que é *open source* e conta com uma comunidade ativa que ajuda no desenvolvimento deste.

3. Análise de Requisitos

3.1 Introdução

Neste capítulo vão ser definidos os requisitos que foram considerados para que seja possível alcançar uma solução plausível para o problema em questão, com o objetivo de promover uma solução mais adequada e mais completa para os problemas dos utilizadores. Desta forma foi desenvolvido um demonstrador de conceito, para que possa ajudar os pequenos comerciantes a detetar possíveis ilegalidades que estejam a cometer por falta de rigor técnico ou desconhecimento de certas regras a cumprir com o seu negócio.

O capítulo termina com uma breve conclusão sobre o caso de estudo realizado, a proposta de solução encontrado perante os novos requisitos propostos.

3.2 Caso de Estudo

Este projeto surgiu da necessidade recorrente que as empresas têm de manipular dados (estruturados e não estruturados) de forma rápida, esta necessidade aliada ao novo paradigma do Big Data e as ferramentas que são utilizadas nesta área, nomeadamente o ecossistema Hadoop.

A base do caso de estudo são os pequenos comércio onde são gerados dados em ficheiros denominados SAF-T, estes dados podem ser analisados de diversas formas, da perspectiva do pequeno comerciante é uma fonte de informação onde estão armazenadas todas as transações do seu negócio, que podem ser tratadas e “transformadas” em informação valiosa, melhorando a tomada de decisão para uma pequena empresa.

O Comerciante para além de vender os seus produtos, também tem os seus fornecedores (empresas de retalho) onde faz a reposição dos seus stocks. Na compra desses produtos são entregues as respetivas faturas, em formato papel, por esse motivo a informatização dos *stocks* é feita de forma manual em ficheiros Excel, sendo estes ficheiros a segunda fonte de informação.

A falta de informação junta com alguma desorganização leva a que os comerciantes de pequenos negócios locais incumpram normas e leis, entre as quais estão as fraudes, a falta de controlo nos *stocks* e inventários dos produtos.

Como requisito principal para os comerciantes é necessário que tenham acesso a relatórios personalizados em que estes disponham uma comparação entre o que foi vendido e o que saiu de *stocks* por um período de tempo, poder efetuar o inventário do estabelecimento sempre que seja necessário.

A grande vantagem para o comerciante é o controlo detalhado do seu negócio, ajudando-o a controlar a faturação, *stocks* e tendo a liberdade de poder acrescentar

novos relatórios que o possam auxiliar na melhoria de tomadas de decisões, e desta forma evitar qualquer tipo de irregularidade a nível fiscal.

Com este cenário, foi proposto criar a seguinte arquitetura:

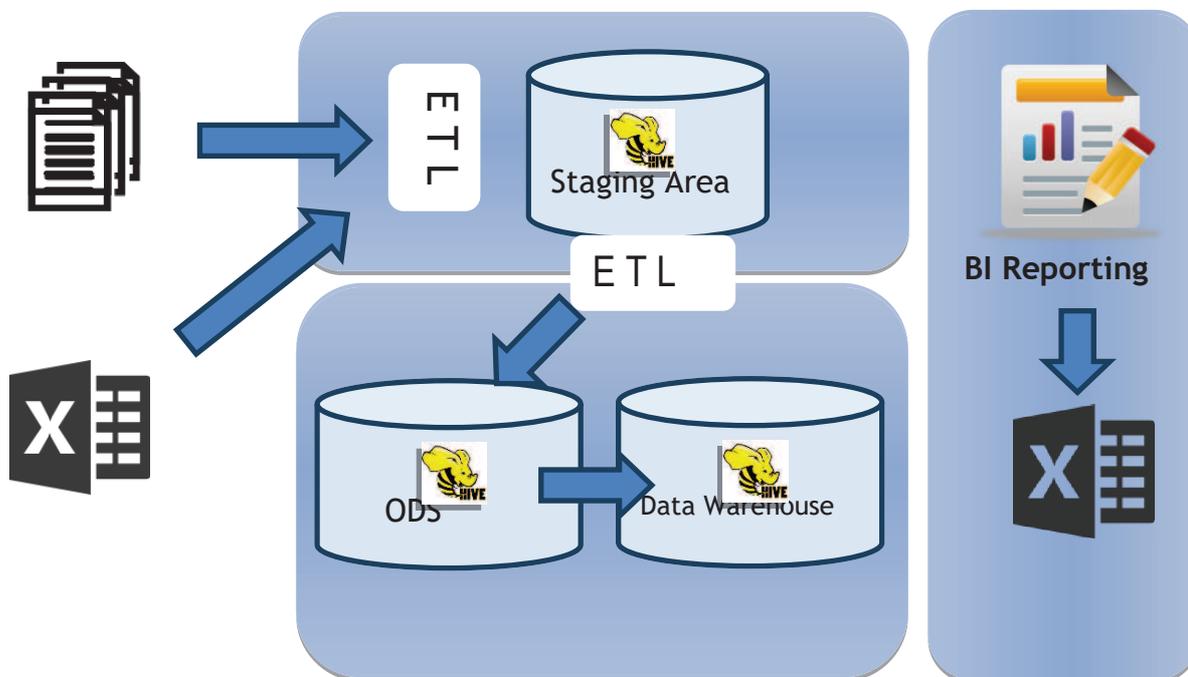


Figura 16 - Arquitetura proposta

A arquitetura passa por ter um repositório de dados centralizados, ou seja uma Data Warehouse, no Hive, com um modelo de dados em estrela, sendo as fontes de dados os ficheiros SAF-T ou outro tipo de ficheiros, e inclusive outras base de dados para que possa enriquecer os dados ao máximo.

Para analisar e visualizar os dados foi também proposto a criação de relatórios que contemplem estes requisitos, com o intuito de responder às necessidades básicas dos comerciantes. De forma a evitar qualquer tipo de custos associados aos programas de *software*, a solução desenhada é de origem cem por cento *open source*. Para tal, foi proposto a utilização de uma das ferramentas do Pentaho, nomeadamente, o Pentaho Report Designer para a visualização dos relatórios construídos, esta ferramenta de reporting é bastante *user friendly*, sendo fácil a aprendizagem no seu uso para os utilizadores finais.

3.2.1 Arquitetura Hadoop

A arquitetura desenhada com o Hadoop como base utiliza o HDFS, Map Reduce e o Hive. Como se pode observar na Figura 17 os ficheiros (XML) são carregados na camada mais baixa do Hadoop, o HDFS, nesta camada os dados gerados pelas vendas e *stocks* são armazenados e ficam disponíveis para serem utilizados pelos restantes componentes do Hadoop, o Hive vai utilizar os ficheiros que estão no HDFS e carregá-los nas suas tabelas, neste processo o Hive internamente utiliza o Map Reduce, depois de carregar os dados, é aberta uma ligação JDBC para que comunique com a ferramenta de *reporting* onde vão ser apresentados os relatórios.

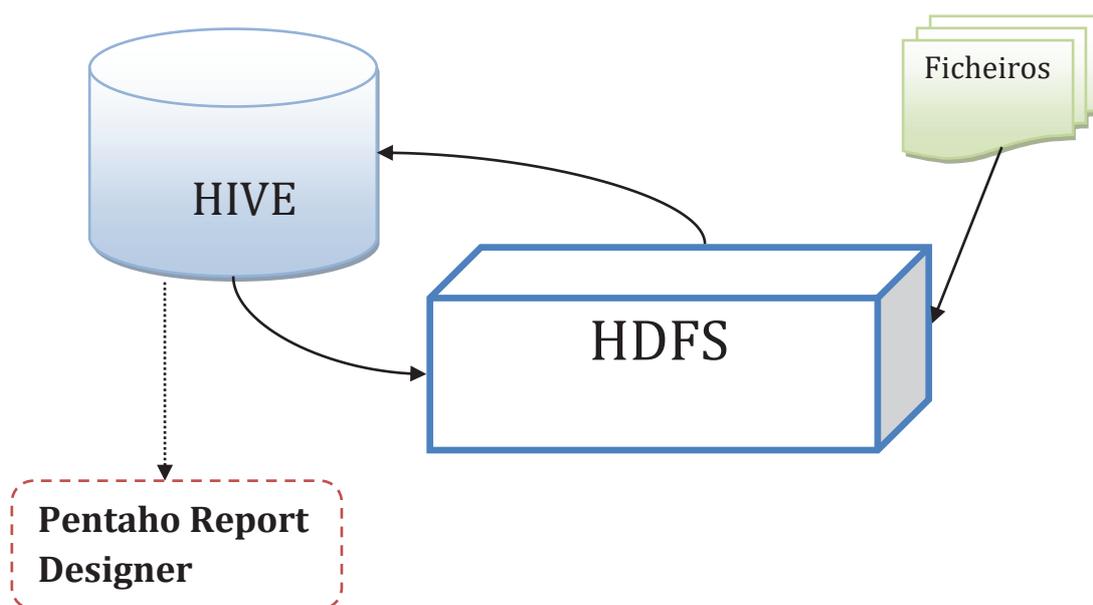


Figura 17- Arquitetura Hadoop desenhada

3.3 Plano de ações

O plano de ações adotado para dividir as diferentes ações em cada parte de desenvolvimento do projeto, as tarefas foram divididas por cada fase.

Fase	Iteração	Objetivos
Kick - Off	M0	- Brainstorming - Estudo de soluções para o problema - Plano de ação
Estado da Arte	M1	- Levantamento de ferramentas Big Data - Análise geral do mercado Big Data
Análise de Requisitos	M2	- Especificação dos requisitos - Definição da arquitetura
Modelo de dados	M3	- Modelação do sistema; - Casos de uso - Verificar se modelo responde ao problema
Implementação	M4	- Configuração do ambiente de desenvolvimento; - Instalação de ferramentas de desenvolvimento - Construção de tabelas, relatórios, funções.

Tabela 1- Plano de ações

3.3 Conclusão

Este capítulo teve como principal objetivo identificar os requisitos com recurso a um pequeno caso de estudo, a quanto a necessidade de criar uma arquitetura que ajude os pequenos comerciantes, nomeadamente a detetar fraudes em vendas, *stocks* e controlo de inventários de produtos para comércio locais.

A solução encontrada foi a criação de um repositório central de dados que é alimentado com ficheiros SAF-T e caso seja necessário outro tipo de ficheiros, por exemplo Excel, para que possa responder a todos os requisitos e que esteja preparado para no futuro atender a novos requisitos, tudo desenvolvido com ferramentas *Open Source*, de forma a evitar custos desnecessários com licenças de *software*.

Assim, a solução está ao alcance de clientes que tenham recursos financeiros reduzidos e ao mesmo tempo poder construir relatórios simples mas ricos em dados.

4. Modelo de Dados

4.1 Introdução

Neste capítulo é apresentado o modelo de dados dos requisitos que foram identificados no capítulo anterior, de modo a apresentar um modelo funcional da solução apresentada. Em primeiro lugar são identificadas as fontes de informação, que servirão como fontes para a construção do esquema no Hive e com recurso ao HiveQL é efetuado o processo ETL. De seguida, é apresentado o modelo de dados para esta solução, para o qual foi escolhido o modelo em estrela, com as respetivas dimensões e tabela de factos, que são a base da Data Warehouse. No fim deste capítulo, é feita uma breve conclusão sobre arquitetura tecnológica definida.

4.2 Arquitetura tecnológica

A arquitetura tecnológica que foi construída foi dividida em duas partes, a parte física e a parte de sistemas.

A parte física consiste em todo o *Hardware* onde o sistema operativo e todos os *softwares* vão ser executados, podendo também ser denominada infraestrutura tecnológica.

O Hardware é um Lenovo G50-80 equipado com um processador Intel® Core i7-5500U com uma velocidade de 2.4GHz com dois núcleos, com 16GB de RAM DDR3L a 1600MHz, com 1TB de armazenamento interno com uma velocidade de 5400 rpm.

Sobre o *hardware* é executada uma máquina virtual com o sistema operativo CentOS *release* 6.5 (Final) com 2 CPUs, 11GB de RAM. Na máquina virtual está instalado o Hadoop 2.5.0 com um único *node*, bem como os restantes componentes do seu ecossistema.

Para a construção e visualização de relatórios foi instalado o Pentaho Report Designer 3.7.

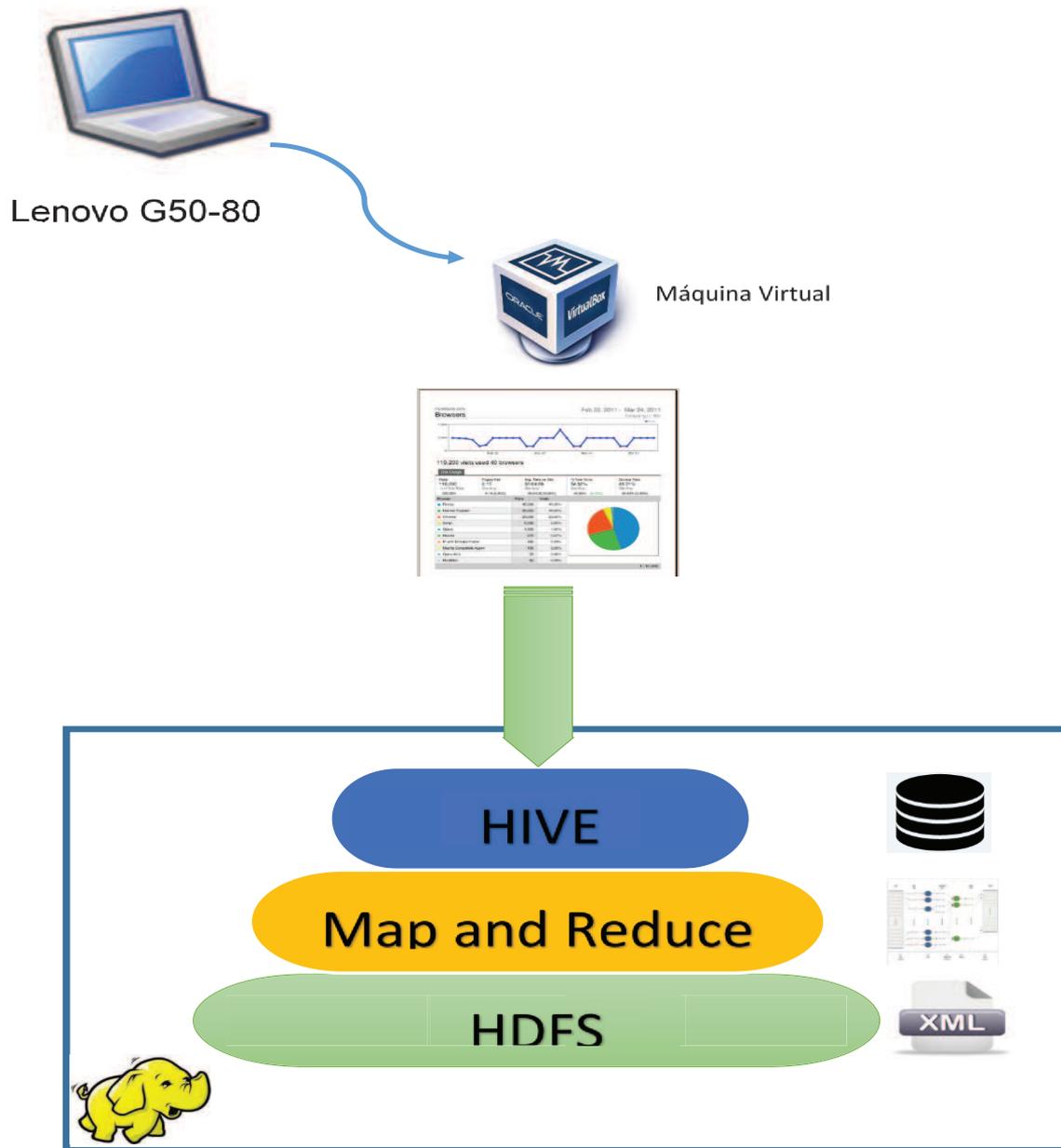


Figura 18- Arquitetura tecnológica

4.3 Casos de Utilização

Na construção de um novo sistema, em primeiro lugar é efetuado um desenho para entender quais são os papéis de cada entidade no sistema. Após se perceber como se iriam implementar este tipo de sistemas, e a utilidade que estes iriam ter, foi efetuado um esquema de casos de utilização onde figuram os principais atores e as atividades de cada um como se pode verificar na figura 19.

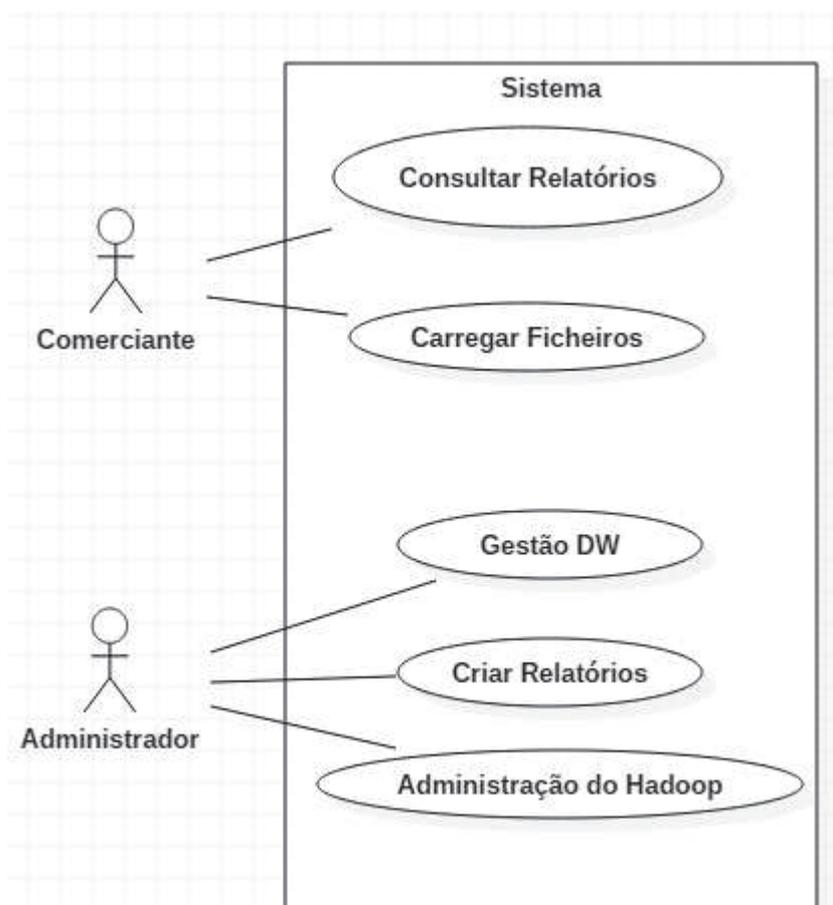


Figura 19- Diagrama de Casos de Uso

Na figura 19 podemos identificar os atores que o sistema vai ter, o primeiro Ator é o Comerciante que tem como principais atividades consultar relatórios e carregar os ficheiros SAF-T no sistema.

O segundo Ator chamado administrador tem como principais responsabilidades a gestão da DW, criar a relatórios personalizados para o Ator anterior, bem como efetuar toda a administração do Hadoop, bem como toda a sua infraestrutura.

4.4 Modelo Conceptual

Para a conceção deste tipo de sistemas, independentemente dos seus requisitos, tem que seguir uma lógica, para que no futuro seja possível implementar novas funcionalidades, ou novos *datamarts*, isto é, o processo de desenvolvimento e manutenção do projeto tem que ser iterativo e cíclico como se pode ver na figura 20.



Figura 20- Modelo Conceptual

Na primeira fase deste processo cíclico é iniciado com o carregamento dos dados que são os ficheiros XML que o comerciante disponibiliza, no seguinte passo é iniciado o processo ETL onde os dados que os ficheiros contêm são passados para tabelas e posteriormente é efetuado todo o tipo de transformações e tratamento dos dados. Na terceira fase do processo são criados os relatórios que são disponibilizados por uma ferramenta de *reporting* após esta fase é necessário que seja feita a administração do sistema.

4.5 Fontes de Dados

As fontes de informação para esta solução foram fornecidas por parte de uma empresa localizada na cidade de Elvas de forma a proteger a identidade de tal, os dados foram anonimizados.

Pode-se observar na figura 21 um exemplo dos dados que os ficheiros que foram fornecidos contêm.

```

- <Product>
  <ProductType>P</ProductType>
  <ProductCode>25060618776084507</ProductCode>
  <ProductGroup>Família</ProductGroup>
  <ProductDescription>OUTROS PRODUTOS</ProductDescription>
  <ProductNumberCode>25060618776084507</ProductNumberCode>
</Product>
- <Product>
  <ProductType>P</ProductType>
  <ProductCode>25060618776084509</ProductCode>
  <ProductGroup>Família</ProductGroup>
  <ProductDescription>OUTROS PRODUTOS</ProductDescription>
  <ProductNumberCode>25060618776084509</ProductNumberCode>
</Product>
- <Product>
  <ProductType>P</ProductType>
  <ProductCode>25060618776084809</ProductCode>
  <ProductGroup>Família</ProductGroup>
  <ProductDescription>MINI CROISSANT</ProductDescription>
  <ProductNumberCode>8413760005901</ProductNumberCode>
</Product>
- <Product>
  <ProductType>P</ProductType>

```

Figura 21- Estrutura ficheiros

4.6 Modelação de dados

Para conceber a solução anteriormente descrita foi construído um modelo de dados em estrela, com o objetivo de responder da melhor forma aos requisitos, foram escolhidos os elementos relevantes dos ficheiros SAF-T para construir as dimensões e a tabela de factos. Nas seguintes figuras estão representados os modelos que servirão de referência para toda a estrutura do Data Warehouse, o modelo é carregado através de um processo ETL no sentido de não existir atributos com informação nula, ou pouco coerente.

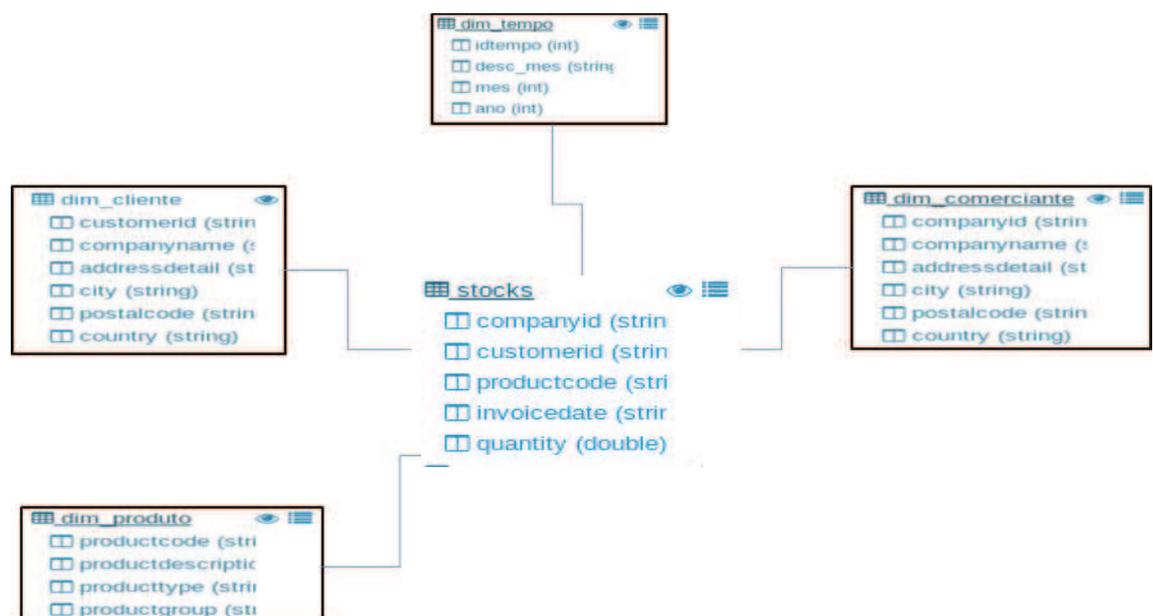


Figura 22- Modelo de dados de Stocks

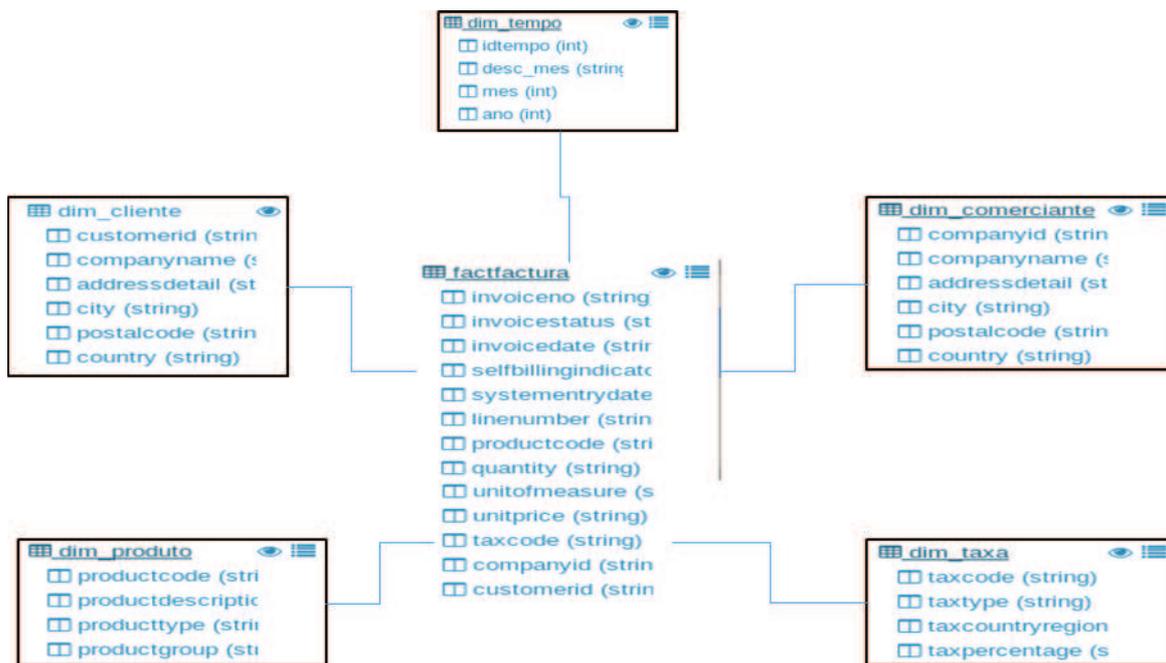


Figura 23-Modelo de dados Faturas

O primeiro modelo representa todos os Stocks que a entidade contém nas suas instalações, enquanto o segundo modelo representa todas as vendas efetuadas pela empresa durante um ano, isto é, todos os produtos que são vendidos diariamente com o detalhe de clientes, representados pelo Número de Identificação Fiscal (NIF), as taxas aplicadas por cada produto, bem como a forma como os produtos são catalogados pelos diferentes tipos existentes.

Neste modelo foram criadas quatro dimensões, a dimensão Cliente que representa os clientes que efetuam compras na empresa, a dimensão Comerciante que representa a empresa, a dimensão Produto que representa todos os produtos comercializados e comprados aos seus fornecedores.

As dimensões utilizadas para a tabela de factos FactFactura são as mesmas que se utilizam, com a exceção da dimensão Taxa que contém todas as taxas que os produtos são sujeitos, que é só utilizada neste último modelo, que contém as vendas dos produtos, as quantidades vendidas.

Com este modelo é possível mostrar todos os produtos que foram vendidos, validar se o que se tinha em *stock* corresponde ao que foi vendido, bem como o inventário de produtos que a empresa tem.

4.7 Conclusão

Neste capítulo foi mostrado o modelo de dados em estrela desenvolvido para os requisitos que foram descritos anteriormente, para tal foi necessário criar quatro dimensões, nomeadamente cliente, comerciante, produto e taxa, bem como uma tabela de factos chamada FactFactura.

5 Implementação

5.1 Introdução

Este capítulo pretende apresentar os resultados obtidos do trabalho desenvolvido, demonstrando as suas funcionalidades, e se este tipo de projeto é de fácil compreensão e implementação.

Pretende ainda demonstrar as melhorias e os benefícios que o sistema poderá trazer para os utilizadores finais.

O capítulo termina com uma breve síntese sobre os resultados alcançados, no sentido de provar se estes cumpriram o objetivo esperado e ainda o que pode ser melhorado em termos futuros.

5.2 Processo ETL

O processo ETL foi desenvolvido no ecossistema Hadoop, nomeadamente utilizando a Linguagem HiveQL que o Hive nos fornece para poder desenvolver o processo de tratamento e extração de dados. O Hive deixa integrar novas funções ou classes Java, Python e dá a liberdade ao utilizador de poder desenvolver à medida das suas necessidades.

O Hive possui bastantes semelhanças com as bases de dados que são utilizadas e por isso partilha diversas funcionalidades. Para este processo o tipo de tabelas utilizadas na camada de *Staging Area*, são *External Tables* mas, pode-se usar o recurso da tabela externa para aceder aos ficheiros externos como se fossem tabelas dentro da base de dados. Quando se cria uma tabela externa, define-se a sua estrutura e localização dentro do Hive. Quando se faz uma *query* à tabela, o Hive lê a tabela externa e retorna os resultados como se os dados fossem armazenados na base de dados.

Mas como os dados estão fora da base de dados, não é necessário preocupar com o processo de carregamento para a base de dados, é potencialmente um dos benefícios significativo para DW. As tabelas externas também têm limitações, não se pode atualizar ou eliminar registos.

5.3 Carregamento dos ficheiros SAF-T

Para o desenvolvimento do carregamento dos ficheiros SAF-PT para as tabelas externas localizadas no Hive, na camada de *Staging Area* da arquitetura definida, foi utilizado o projeto *open source* Hive-XML-SerDe [28].

Em primeiro lugar, o SerDe é curto para ser *Serializer / Deserializer*. O Hive usa a interface SerDe para *input / output* (IO). A interface lida com a serialização e “desserialização” e também a interpretação dos resultados de serialização como campos individuais para o processamento.

A SerDe permite ao Hive ler dados de uma tabela, e escrevê-lo de volta para o HDFS em qualquer formato personalizado. Qualquer pessoa pode escrever seu próprio SerDe para os seus próprios formatos de dados. [29]

O Hive-XML-SerDe consiste num projeto desenvolvido por programadores independentes com o intuito de ajudar quem utilize o Hive e permite que o utilizador faça o mapeamento do esquema XML com os tipos de dados do Hive através do *Data Definition Language* (DDL), de acordo com as seguintes regras e sintaxe.

```
CREATE [EXTERNAL] TABLE <table_name> (<column_specifications>)
ROW FORMAT SERDE 'com.ibm.spss.hive.serde2.xml.XmlSerDe'
WITH SERDEPROPERTIES (
["xml.processor.class"=<xml_processor_class_name>,"]
"column.xpath.<column_name>"=<xpath_query>,"
...
["xml.map.specification.<xml_element_name>"=<map_specification>"
...
]
)
STORED AS
INPUTFORMAT 'com.ibm.spss.hive.serde2.xml.XmlInputFormat'
OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.IgnoreKeyTextOutputFormat'
[LOCATION "<data_location>"]
TBLPROPERTIES (
"xmlinput.start"=<start_tag >,"
"xmlinput.end"=<end_tag>"
);
```

[30]

Os dados são armazenados nas tabelas em formato de *Array*, nesta fase os dados estão todos distribuídos por cada coluna dependendo do que foi configurado no *SQL statement*, para que os dados fiquem com cada atributo por cada linha é necessário efetuar um *Split* ao *Array* e passar todos os dados para a tabela final que está na camada da *DW*.

Para o processo de tratamento de dados que insere a informação na *DW* foi utilizado o *BrickHouse* que é uma coleção de funções definidas pelo utilizador para melhorar a produtividade, escalabilidade e robustez no desenvolvimento de *queries* para o Hive. [31].

Para utilizar o Hive XML SerDe e o Brickhouse é necessário que cada vez que vá ser utilizado seja adicionada as bibliotecas por cada sessão que está aberta, para tal é utilizado as seguintes linhas de comando:

```
add jar /home/cloudera/Downloads/hivexmlserde-1.0.5.3.jar;
add jar /home/cloudera/brickhouse/target/brickhouse-0.6.0.jar;
```

Depois é necessário ajustar o código acima, a primeira parte da sintaxe é a criação da tabela e os campos que esta vai ter, dentro do *serdeproperties*, é definido o que cada coluna vai receber de cada atributo que existe no ficheiro SAF-T, em modo de exemplo, a coluna *CompanyID* para ser preenchida com o *CompanyID* que existe no ficheiro, é necessário indicar todo o caminho do atributo, como se pode verificar na Figura 24.

```
create external table tmp_comerciante
(
  CompanyID ARRAY<STRING>,
  CompanyName ARRAY<STRING>,
  AddressDetail ARRAY<STRING>,
  City ARRAY<STRING>,
  PostalCode ARRAY<STRING>,
  Country ARRAY<STRING>
)
row format serde 'com.ibm.spss.hive.serde2.xml.XmlSerDe'
with serdeproperties (
  'column.xpath.CompanyID' = '/AuditFile/Header/CompanyID/text()',
  'column.xpath.CompanyName' = '/AuditFile/Header/CompanyName/text()',
  'column.xpath.AddressDetail' =
'/AuditFile/Header/CompanyAddress/AddressDetail/text()',
  'column.xpath.City' = '/AuditFile/Header/CompanyAddress/City/text()',
  'column.xpath.PostalCode' =
'/AuditFile/Header/CompanyAddress/PostalCode/text()',
  'column.xpath.Country' = '/AuditFile/Header/CompanyAddress/Country/text()'
)
stored as
inputformat 'com.ibm.spss.hive.serde2.xml.XmlInputFormat'
outputformat 'org.apache.hadoop.hive.ql.io.IgnoreKeyTextOutputFormat'
location '/eloi'
tblproperties (
  'xmlinput.start' = '<AuditFile ',
  'xmlinput.end' = '</AuditFile>'
);
```

```
-<AuditFile
  -<Header>
    <AuditFileVersion>1.03_01</AuditFileVersion>
    <CompanyID>1      283</CompanyID>
```

Figura 24 - Exemplo estrutura campo ComanyID

5.3.1 Dimensão Cliente

A dimensão Cliente representa o cliente que efetua compras nos estabelecimentos comerciais e que facilitam o seu NIF para poder obter a fatura para que os seus gastos sejam registados.

Para a carga da dimensão cliente, primeiro foram selecionados os campos relevantes relativos aos clientes que estão presentes no ficheiro SAF-T, sendo utilizada uma tabela temporária para armazenar os dados em brutos e só depois de serem filtrados foram passados para a tabela final de clientes.



Figura 25 - Dimensão Cliente

5.3.2 Dimensão Produto

A dimensão Produto representa os vários produtos que são vendidos e armazenados na empresa. Para implementar esta dimensão foi necessário filtrar os diversos tipos de produtos, para que não houvesse ambiguidade entre estes. O carregamento da tabela de produtos teve o mesmo tratamento que a dimensão cliente.

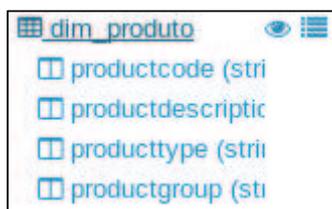


Figura 26- Dimensão Produto

5.3.3 Dimensão Comerciante

A dimensão comerciante representa a empresa que vende os produtos ao público em geral e seus clientes, esta dimensão contém os dados relevantes da empresa, o carregamento dos dados é igual ao descrito nas tabelas anteriores.



Figura 27 - Dimensão Comerciante

5.3.4 Dimensão Taxa

A dimensão Taxa representa as taxas no caso de Portugal o IVA (imposto sobre o valor acrescentado) que é aplicado aos variados produtos. As taxas que podem ser aplicadas aos produtos são as seguintes:

Tipo	Taxa
Isenta	0%
Reduzida	6%
Intermédia	13%
Normal	23%

Tabela 2 - Descrição do tipo de taxas

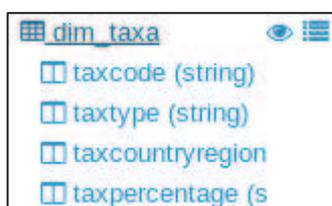


Figura 28- Dimensão Taxa

5.3.4 Dimensão Tempo

A dimensão Tempo contém as datas referentes às faturas que foram registadas.



Figura 29 - Dimensão Tempo

5.3.5 Tabela de Factos Fatura

A tabela de factos FactFactura representa as vendas efetuadas a cada pessoa que efetua as suas compras, contém os produtos que são comprados bem como o seu preço unitário e a quantidade e a hora exata das transações.

Para a tabela de factos foi necessário procedimentos adicionais para efetuar a carga de todas as faturas, porque uma fatura pode conter uma ou mais linhas, e devido a essa condição foi necessário um tratamento diferente do que foi utilizado nas outras tabelas.

Column Name	Data Type
invoiceno	string
invoicestatus	string
invoicedate	string
selfbillingindicatc	string
systementrydate	string
linenummer	string
productcode	string
quantity	string
unitofmeasure	string
unitprice	string
taxcode	string
companyid	string
customerid	string

Figura 30 - Tabela de factos FactFactura

5.3.6 Tabela de factos Stocks

A tabela de factos Stocks representa *stocks* que a empresa tem, denominamos stocks ou existências de uma empresa para todos os materiais e objetos armazenados, tanto os que são necessárias para o processo de produção como os que estão para venda.

A tabela de factos de *stocks* contém o número identificador da empresa, do cliente e do produto, sendo a medida a quantidade.

Column Name	Data Type
companyid	string
customerid	string
productcode	string
invoicedate	string
quantity	double

Figura 31 - Tabela de factos Stocks

5.4 Relatórios

Na criação de relatórios, foi usada a plataforma Report Designer do Pentaho, em que se utilizou o Report Wizard disponível na *suite* Pentaho, conforme se pode verificar na Figura 32.

O *template* escolhido foi o Jade, mas pode-se escolher outro ou criar um de raiz. De seguida, foi necessário configurar a ligação ao Hive. É realçar que o Report Designer é das poucas ferramentas de Reporting que incluem nativamente a opção de configurar a ligação para o Hive. Para isso removeu-se a ligação existente, criou-se uma nova JDBC e editou-se, definindo uma ligação ao Hive, local onde está armazenado o DW. Após isto foi necessário criar uma *query*, através da *query designer*, seleccionando as tabelas e os atributos pertinentes do *DW*.

De seguida, foram seleccionados os itens, que se pretenderam analisar, para ficarem no *layout* da página como se pode ver na Figura 32.

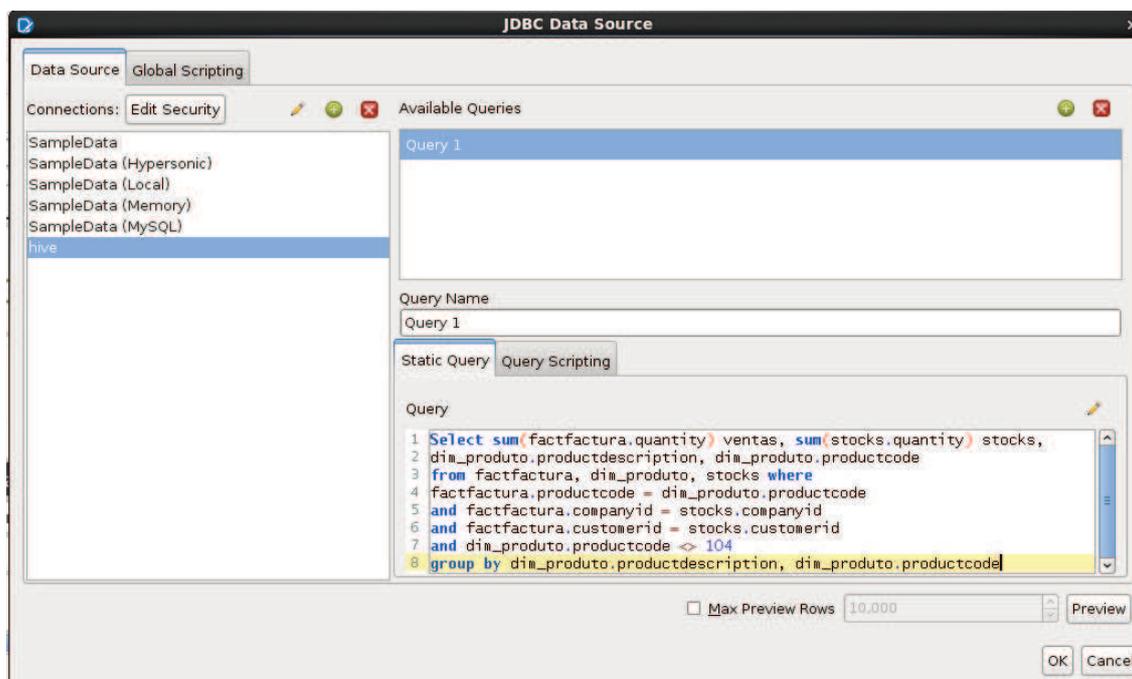
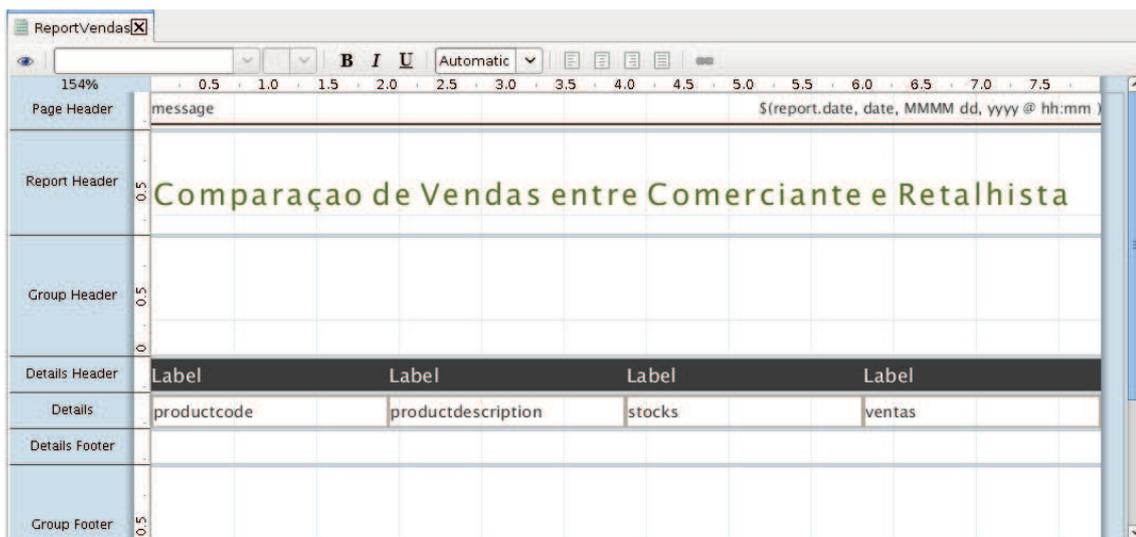


Figura 32- Construção de relatório

Seguidamente, foi necessário formatar os itens seleccionados indicando-lhe uns nomes próprios, alinhando alguns ao centro outros à esquerda, representado na Figura 24. Em seguida, foi seleccionado finish para terminar o *Wizard*, para proceder à personalização do *layout* da página.



The screenshot shows a reporting tool window titled 'ReportVendas'. The interface includes a toolbar with bold, italic, and underline options, and a font size dropdown set to 'Automatic'. The report content is structured as follows:

Section	Content				
Page Header	message \$(report.date, date, MMMM dd, yyyy @ hh:mm)				
Report Header	<h2 style="text-align: center;">Comparação de Vendas entre Comerciante e Retalhista</h2>				
Group Header					
Details Header	<table border="1" style="width: 100%;"><thead><tr><th>Label</th><th>Label</th><th>Label</th><th>Label</th></tr></thead></table>	Label	Label	Label	Label
Label	Label	Label	Label		
Details	<table border="1" style="width: 100%;"><tbody><tr><td>productcode</td><td>productdescription</td><td>stocks</td><td>ventas</td></tr></tbody></table>	productcode	productdescription	stocks	ventas
productcode	productdescription	stocks	ventas		
Details Footer					
Group Footer					

Figura 33 - Relatório formatado

5.5 Conclusão

Neste capítulo foram mostradas as diversas fases de uma solução Big Data com recurso a uma ferramenta de *Reporting*, durante o capítulo foi demonstrado como é possível criar um processo de ETL com as ferramentas atuais de Big Data, sendo inicialmente um pouco difícil conseguir adaptar a esta nova ferramenta que tem um grande potencial para ser utilizada como ferramenta de ETL no que trata a ficheiros XML.

Também foi possível observar como pode ser integrado com uma ferramenta de Reporting e gerar relatórios ricos com informação tratada e perceptível para o utilizador.

6 Resultados Obtidos

6.1 Introdução

Este capítulo tem como objetivo mostrar os resultados obtidos após a implementação da solução que pretende demonstrar a sua escalabilidade, o seu baixo custo, a sua eficiência no processamento de dados bem como a sua fiabilidade.

Na solução construída a visualização dos dados é feita através de uma ferramenta de relatórios, e pretende ainda demonstrar as melhorias e os benefícios que o sistema poderá trazer.

O capítulo é concluído com um breve resumo sobre os resultados obtidos, com o objetivo de provar se estes tiveram o resultado esperado.

6.2 Processo ETL

O desenvolvimento do processo ETL foi concluído com sucesso, a construção de um projeto com duas camadas, uma de *Staging Area* e a outra de DW com a modelação em estrela.



dim_cliente	
dim_comerciante	
dim_produto	
dim_taxa	
dim_tempo	
factfactura	
stocks	
tmp_cliente	
tmp_comerciante	
tmp_factura	
tmp_produto	
tmp_taxa	
tmp_tempo	
wk_tempo	

Figura 34 - Tabelas da Data Warehouse

6.2.1 Dimensão Comerciante

Pode-se observar que foi inserido corretamente o registo referente aos dados do comerciante.

Data sample for dim_comerciante [View in Metastore Browser](#)

companyid	companyname	addressdetail	city	postalcode	country
19	ISSI			258	PT

Figura 35 - Registo na Dimensão Comerciante

6.2.2 Dimensão Cliente

O carregamento dos dados da dimensão cliente foi concluído com sucesso, como se pode confirmar na figura 27.

Data sample for dim_cliente [View in Metastore Browser](#)

customerid	companyname	addressdetail	city	postalcode	country
1	carmen	Desconhecido	Desconhecido	Desconhecido	PT
2	Ass ca adores monforte da beira	Desconhecido	Desconhecido	Desconhecido	PT
999999990	Consumidor final	Desconhecido	Desconhecido	Desconhecido	PT

Figura 36 - Registos na Dimensão Cliente

6.2.3 Dimensão Produto

Tal como aconteceu nas dimensões anteriores os dados também foram carregados corretamente para a dimensão produto.

Data sample for dim_produto

6	ginga	P	1
7	Cha	P	1
8	frize	P	1
9	ginga	P	1
11	cerj min	P	1
12	ginga	P	1
13	whisky	P	1
14	porto	P	1

Figura 37 - Registos na Dimensão Produto

6.2.4 Dimensão Taxa

A figura 29 ilustra os dados carregados para a dimensão taxa, e que os resultados obtidos são positivos.

Data sample for dim_taxa

taxcode	taxtype	taxcountryregion	taxpercentage
INT	IVA	PT	13
ISE	IVA	PT	0
NOR	IVA	PT	23
RED	IVA	PT	6

Figura 38- Registo na Dimensão Taxa

6.2.5 Dimensão Tempo

Para a dimensão Tempo, os dados também foram carregados com sucesso.

Data sample for dim_tempo

[View in Metastore Browser](#)

iddate	year	month	day
2014-04-01	2014	4	1
2014-04-02	2014	4	2
2014-04-03	2014	4	3
2014-04-04	2014	4	4
2014-04-05	2014	4	5
2014-04-06	2014	4	6

Figura 39 - Registos na Dimensão Tempo

6.2.6 Tabela de Factos FactFactura

Na tabela de factos FactFactura pode-se observar na Figura 31 que os dados foram carregados corretamente, com o conteúdo das faturas emitidas pelo comerciante.

Data sample for factfactura

[View in Metastore Browser](#)

invoiceno	invoicestatus	invoicedate	selfbillingindicator	systementrydate	linenumber	productcode	quantity	unitofmeasure	unitprice	taxcode	companyid	cu
FA T1/1	None	2013-10-07	0	2013-10-08T16:29:19	1	104	1	UN	0.45	NOR	198938551	9999
FA T1/2	None	2013-11-03	0	2013-11-03T13:49:53	1	37	1	UN	0.57	NOR	198938551	9999
FB T1/1	N	2013-01-06	0	2013-01-06T06:46:47	1	1	1	UN	0.45	NOR	198938551	9999
FS T1/1472	N	2013-04-01	0	2013-04-01T07:45:20	1	1	1	UN	0.45	NOR	198938551	9999
FS T1/2850	N	2013-06-01	0	2013-06-01T07:27:09	1	104	1	UN	0.45	NOR	198938551	9999
FS T1/3549	N	2013-07-01	0	2013-07-01T07:24:17	1	104	1	UN	0.45	NOR	198938551	9999
FS T1/373	N	2013-02-01	0	2013-02-01T07:14:34	1	1	1	UN	0.45	NOR	198938551	9999

Figura 40 - Registos na tabela de factos Fatura

6.2.7 Tabela de Factos Stocks

Também para a tabela de factos de Stocks os dados foram inseridos corretamente, e foram alcançados os resultados esperados.

Data sample for stocks [View in Metastore Browser](#) ✕

companyid	customerid	productcode	invoicedate	quantity
198938551	999999990	1	2013-01-06	1.0
198938551	999999990	1	2013-02-01	1.0
198938551	999999990	1	2013-03-01	1.0
198938551	999999990	1	2013-04-01	1.0
198938551	999999990	101	2013-12-01	1.0
198938551	999999990	104	2013-06-01	1.0

Figura 41 - Registo na tabela de factos Stocks

6.3 Relatórios

A construção de relatórios facilitou a compreensão e entendimento da informação que é recolhida do *DW*, ajudando a manter a informação organizada e confiável, permitindo ao utilizador fazer análises que tal não poderia acontecer através dos ficheiros SAF-T.

January 22, 2016 @ 12:40

Vendas vs Stocks

productcode	ventas	stocks
1	16	16
101	1	1
104	25	25
37	1	1

Figura 42- Visualização de relatórios no Report Designer

Os relatórios desenvolvidos permitem ao utilizador final (comerciante) controlar os seus *stocks*, efetuar um cruzamento de dados entre o que vendeu e o que tinha em *stock* de modo a poder detetar inconsistências e falhas que podem ter acontecido ao final de um dia de vendas.

Também foi construído um relatório que permite uma entidade como a autoridade tributária, que tendo em sua posse a informação de comerciante e retalhista, faz o cruzamento entre as vendas de ambos e consegue encontrar diferenças entre o que comerciante comprou ao retalhista e posteriormente o que foi vendido.

6.4 Conclusão

As ferramentas existentes de Big Data nomeadamente o Hadoop, atualmente não são de fácil utilização e é necessário conhecimentos técnicos profundos. O capítulo pretendeu demonstrar que é possível implementar soluções de Big Data em conjunto com ferramentas de BI.

Os componentes utilizados do Hadoop neste projeto são uma alternativa credível e com um valor significativo para serem utilizadas como ferramentas de ETL, tendo a vantagem de ser *open source* e podendo ser desenvolvidas funções à medida das necessidades do projeto, sendo que o Hadoop tem uma capacidade de processar e tratar dados que as atuais ferramentas de ETL não conseguem alcançar sem uma infraestrutura tecnológica bastante mais custosa que a que foi apresentada nesta dissertação.

7 Conclusões

7.1 Reflexão

A iniciativa deste projeto teve origem na grande importância que o Big Data está a ter a curto e a médio prazo na estratégia empresarial e inteligência de negócio nas empresas e a vontade de o autor se envolver nesta nova tecnologia que vai ter um papel fulcral no mundo das tecnologias de informação.

Quando se fala de Big Data a primeira ligação que se faz é aos dados (informação), o que é correto, sendo que o Big Data também trouxe para este entorno novas ferramentas com uma alta capacidade de processar dados. Entre as inúmeras vantagens que estão associadas ao Big Data destacam-se a possibilidade a deteção de fraudes, de predizer com base num vasto aglomerado de dados.

Depois de estarem clarificados os objetivos e a motivação para o tema da dissertação, que foi encontrado após um *brainstorming* inicial, isto porque não bastava a exploração desta nova tendência a nível teórico, era necessário encontrar algo que conjugasse os dois fatores, utilizar ferramentas de Big Data e ao mesmo tempo informação que fosse útil para tratar e que servisse de ajuda em novos projetos.

A possibilidade de processar os ficheiros SAF-T que tanto as autoridades tributárias como os comerciantes utilizam, estes em último para registar os movimentos da sua atividade empresarial.

Foi realizado um levantamento do estado da arte (uma pesquisa de fontes de informação) com enfoque no tema principal mas também em tecnologias relacionadas e bastantes vezes associadas, como é o caso do Business Intelligence.

As principais dificuldades encontradas inicialmente no desenvolvimento deste projeto começaram pela configuração e instalação de todo o ambiente de desenvolvimento onde ia ser desenvolvido o projeto, sendo que a determinada altura do projeto houve a necessidade de mudar o ambiente de desenvolvimento para uma versão pré configurada disponibilizada pela empresa Cloudera após uma análise das diferentes ofertas que existem no mercado do Big Data.

Outra dificuldade relevante foi o desenvolvimento do processo ETL a partir de ficheiros XML que são ficheiros altamente expansíveis e que têm uma estrutura definida que tem que ser respeitada, a recolha da informação para as tabelas temporárias e posteriormente para a Data Warehouse foram um dos pontos mais difíceis de ultrapassar.

Para a construção dos relatórios que o utilizador pode consultar, é necessário ter conhecimentos de SQL para que se possa criar as *queries* que vão ser a fonte de informação para os relatórios, o Report Designer do Pentaho executa bem o que lhe é

passado, e é das poucas ferramentas que tem suporte direto para criar ligações ao Hive, sendo essa uma das razões pelo qual foi escolhido. Outros players do mercado de BI com provas dadas atualmente não têm disponível de forma nativa, a possibilidade de conectarmos diretamente ao Hive tal como acontece no Report Designer.

Foi também possível verificar que o modelo multidimensional criado para este projeto revelou-se o mais indicado, visto que todos os resultados obtidos foram os esperados, indo assim ao encontro das expectativas de desenvolvimento de um Data Warehouse.

A solução em termos gerais está bem desenvolvida e contribui para combater o problema que foi relatado ao longo da dissertação, pelo qual se pode concluir que a implementação deste projeto tem inúmeros benefícios para quem se preocupe em controlar o seu pequeno negócio local.

O resultado alcançado com o Hadoop e o Report Designer poderia também ser construído com outro tipo de ferramentas e base de dados, no entanto a longo prazo com o acumular de dados e o aumento da volumetria, o processamento e a gestão dos dados tornar-se-ia algo penoso para o sistema. Com o Hadoop o aumento dos dados não é um problema já que está preparado para tratar grandes volumes de dados.

Um dos pontos fracos que se pode identificar é o ambiente de desenvolvimento só ter um único *Node* perdendo uma das vantagens do Hadoop que é a tolerância a falhas.

Na opinião do autor o facto de o projeto não ter qualquer custo para a entidade que o deseje utilizar é uma enorme vantagem, visto que se pode ter uma ferramenta grátis para ajudar a controlar o negócio.

7.2 Trabalho Futuro

O projeto desenvolvido corresponde às expectativas criadas inicialmente pelo autor e em termos conceptuais foram alcançados os objetivos que se tinham definido, havendo espaço para novas melhorias e novos desenvolvimentos neste projeto embrião.

Uma das melhorias que podiam ser adicionadas em futuros desenvolvimentos seria a escolha de uma ferramenta de BI mais sofisticada e que permitisse efetuar consultas *ad-hoc* como requisito mínimo, desta forma a experiência de usabilidade com a plataforma iria melhorar significativamente. O Report Designer do Pentaho apesar de ter desempenhado bem os relatórios o facto de ser necessário construir *queries* manualmente é uma desvantagem, visto o utilizador final (Comerciante) não ter conhecimentos técnicos para construir *queries* que possam responder às suas necessidades de negócio.

A nível de ambiente de desenvolvimento seria bastante útil ter uma infraestrutura em *node* e ter múltiplos *nodes* para que no caso de existir alguma falha seja possível ter a informação em outro *node*, no caso descrito não era imperativo devido ao volume de dados ser pequeno, mas se o número de utilizadores e volume de dados cresce-se teria de ser adotada esta melhoria.

Por fim poderiam ser incluídos ficheiros SAF-T de tamanho significativo, na ordem dos *Gigas* ou *Terabytes* para que se pudesse efetuar um processamento de dados com ficheiros desse tamanho sendo que tais ficheiros só são gerados por grandes empresas no setor dos hipermercados.

A plataforma não foi testada com ficheiros dessa grandeza, como trabalho futuro seria adquirir ficheiros com um tamanho superior e verificar se a plataforma consegue processar os dados em tempos aceitáveis, no caso dos tempos fossem bons, poderia ser proposto autoridade tributaria testar a plataforma visto que é a única entidade que tem acesso a todos os dados contabilísticos das empresas.

Referências

- [1] - Big Data: Are you ready for blast-off?, [Online] <http://www.bbc.com/news/business-26383058> [Acedido a: 11 de Março de 2016]
- [2] -The Data Explosion in 2014 Minute by Minute – Infographic , [Online] <http://newstex.com/2014/07/12/the-data-explosion-in-2014-minute-by-minute-infographic/> [Acedido a: 11 de Março de 2016]
- [3] - UNDERSTANDING BIG DATA: THE SEVEN V'S [Online] <http://dataconomy.com/seven-vs-big-data/> [Acedido a: 11 de Março de 2016]
- [4] - http://www.sas.com/en_us/insights/big-data/what-is-big-data.html [Acedido a: 11 de Março de 2016]
- [5] -Big Data What it is and why it matters [Online] http://www.sas.com/en_us/insights/big-data/what-is-big-data.html [Acedido a: 11 de Março de 2016]
- [6] -What Is Apache Hadoop? [Online] <http://hadoop.apache.org/> [Acedido a: 11 de Março de 2016]
- [7] - Leveraging Massively Parallel Processing in an Oracle Environment for Big Data Analytics [Online] <http://www.oracle.com/technetwork/database/bi-datawarehousing/twp-hadoop-oracle-194542.pdf> [Acedido a: 11 de Março de 2016]
- [8] – Dean, Jared (2014), Big Data, Data Mining, and Machine Learning, John Wiley Sons Inc
- [9] - Tracking the evolution of big data: A timeline [Online] <http://gcn.com/articles/2013/05/30/gcn30-timeline-big-data.aspx> [Acedido a: 11 de Março de 2016]
- [10] - Big Data Timeline [Online] <http://www.datasciencecentral.com/profiles/blogs/big-data-timeline> [Acedido a: 11 de Março de 2016]
- [11] -What is Watson? [Online] <http://www.ibm.com/smarterplanet/us/en/ibmwatson/what-is-watson.html> [Acedido a: 11 de Março de 2016]
- [12] - Introduction [Online] - http://hadoop.apache.org/docs/r1.1.1/hdfs_design.html#Introduction [Acedido a: 11 de Março de 2016]

- [13] - MapReduce - Purpose [Online] http://hadoop.apache.org/docs/r1.1.1/mapred_tutorial.html#Purpose [Acedido a: 11 de Março de 2016]
- [14] -Boris Lublinsky, Kevin T. Smith, Alexey Yakubovich (2013), Professional Hadoop Solutions, John Wiley & Sons, Inc, Indianapolis, Indiana
- [15] -MapReduce: Simplified Data Processing on Large Clusters [Online] - <http://static.googleusercontent.com/media/research.google.com/pt-PT//archive/mapreduce-osdi04.pdf> [Acedido a: 11 de Março de 2016]
- [16] -Bigtable: A Distributed Storage System for Structured Data [Online] <http://static.googleusercontent.com/media/research.google.com/pt-PT//archive/bigtable-osdi06.pdf> [Acedido a: 11 de Março de 2016]
- [17] - HadoopDB: An Architectural Hybrid of MapReduce and DBMS Technologies for Analytical Workloads [Online] <http://db.cs.yale.edu/hadoopdb/hadoopdb.pdf> [Acedido a: 11 de Março de 2016]
- [18] [Online] <https://hive.apache.org/> [Acedido a: 11 de Março de 2016]
- [19] - Hive - A Petabyte Scale Data Warehouse Using Hadoop [Online] <http://infolab.stanford.edu/~ragho/hive-icde2010.pdf> [Acedido a: 4 de Abril de 2016]
- [20] - Davenport and Harris (2007), Competing on Analytics: The New Science of Winning.
- [21] - A Five-Layered Business Intelligence Architecture - [Online] <http://www.ibimapublishing.com/journals/CIBIMA/2011/695619/695619.pdf> [Acedido a: 11 de Março de 2016]
- [22] - Data Definitions and DBMS -[Online] <http://dbms.ca/concepts/structures.html> [Acedido a: 11 de Março de 2016]
- [23] - NoSQL Databases: An Overview -[Online] <http://www.thoughtworks.com/insights/blog/nosql-databases-overview>[Acedido a: 11 de Março de 2016]
- [24] - OECD(2016) Guidance Note - Guidance for Developers of Business and Accounting Software Concerning Tax Audit Requirements, [online] <http://www.oecd.org/tax/administration/guidancenote-guidancefordevelopersofbusinessandaccountingsoftwareconcerningtaxauditrequirements.htm> [acedido a 11 de Março de 2016]
- [25] - Hadoop in the Enterprise [online] - http://static.cybercommons.org/dell/dell_hadoop/Dell%20White%20Paper%20%20-%20Hadoop%20in%20the%20Enterprise.pdf [acedido a 11 de Março de 2016]

- [26] - SAF-T PT (Standard Audit File for Tax purposes) - Versão Portuguesa – [Online] http://info.portaldasfinancas.gov.pt/pt/apoio_contribuinte/news_saf-t_pt.htm [acedido a 03 de Abril de 2016]
- [27] - [online] <http://www.b9bit.com.br/> [acedido a 11 de Março de 2016]
- [28] - [online] <https://github.com/dvasilen/Hive-XML-SerDe/wiki/XML-data-sources>[acedido a 11 de Março de 2016]
- [29] --SerDe [online] <https://cwiki.apache.org/confluence/display/Hive/SerDe>[acedido a 11 de Março de 2016]
- [30] - -[online] <https://github.com/dvasilen/Hive-XML-SerDe/wiki/XML-data-sources>[acedido a 11 de Março de 2016]
- [31] -[online] <https://github.com/klout/brickhouse>[acedido a 11 de Março de 2016]
- [32] - (Chen et al., 2012). Forrester (2012)
- [33] - Data Never Sleeps 3.0-[online] <https://www.domo.com/blog/2015/08/data-never-sleeps-3-0/>[acedido a 11 de Março de 2016]
- [34]- Top 5 Considerations When Evaluating NoSQL Databases -[online] https://s3.amazonaws.com/info-mongodb-com/10gen_Top_5_NoSQL_Considerations.pdf [acedido a 11 de Março de 2016]
- [35] Oracle NoSQL Database -[online] <http://www.oracle.com/technetwork/database/nosqldb/learnmore/nosql-wp-1436762.pdf> [acedido a 11 de Março de 2016]
- [36] [online] <http://storagegaga.com/2011/12/> [acedido a 11 de Março de 2016]
- [37] - Judith Hurwitz, Alan Nugent, Dr. Fern Halper and Marcia Kaufman, Big Data for Dummies, 18 -19
- [38] -Layer 1 of the Big Data Stack: Security Infrastructure [online] <http://www.dummies.com/how-to/content/layer-1-of-the-big-data-stack-security-infrastruct.html> [acedido a 03 de Abril de 2016]
- [39] – [online] <https://www.forrester.com/home/> [acedido a 11 de Março de 2016]
- [40]- The Forrester Wave™: Big Data Hadoop Distributions,Q12016-[online] <https://www.cloudera.com/content/dam/www/static/documents/analyst-reports/forrester-wave-big-data-hadoop-distributions.pdf> [acedido a 03 de Abril de 2016]
- [41] - [online] <http://rdschneider.com/about/> [acedido a 03 de Abril de 2016]

[42] - The Journey to a Data Lake [online] <http://info.hortonworks.com/rs/h2source/images/Hadoop-Data-Lake-white-paper.pdf%20> [acedido a 03 de Abril de 2016]

[43] -Hadoop Overview from the Austin Cloudera Sessions [online] <http://www.techweekly.com/viewpoints/2013/06/26/hadoop-overview-from-the-austin-cloudera-sessions/> [acedido a 03 de Abril de 2016]

[44] - MapR System Overview - [online] http://maprdocs.mapr.com/51/index.html#MapROverview/c_overview_intro.html [acedido a 03 de Abril de 2016]

[45] - Hadoop Buyer's Guide [online] - http://insights.ubuntu.com/wp-content/uploads/HadoopBuyersGuide_sm.pdf [acedido a 03 de Abril de 2016]

[46] - OECD (2013), "Exploring Data-Driven Innovation as a New Source of Growth: Mapping the Policy Issues Raised by "Big Data"", OECD Digital Economy Papers, No. 222, OECD Publishing

[47] - Reimsbach-Kounatze, C. (2015), "The Proliferation of "Big Data" and Implications for Official Statistics and Statistical Agencies: A Preliminary Analysis", OECD Digital Economy Papers, No. 245, OECD Publishing

[48] - SOBRE O E-FATURA [online] - http://info.portaldasfinancas.gov.pt/pt/faturas/sobre_efatura.html - [acedido a 03 de Abril de 2016]

[49] - Combate à Fraude e Evasão Fiscais e Aduaneiras [online] http://info.portaldasfinancas.gov.pt/NR/rdonlyres/E245BDAE-D856-4186-A950-F0BE649869DF/0/Plano_Estrategico_Combate_Fraude_Fiscal_Aduaneira_2015_2017.pdf [acedido a 03 de Abril de 2016]

[50] - Pentaho Reporting - User Guide for Report Designer [online] <http://wiki.pentaho.com/display/Reporting/Pentaho+Reporting+-+User+Guide+for+Report+Designer> [acedido a 03 de Abril de 2016]