INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Automação de Integração de Dados com Ferramenta de Business Intelligence

MARCO ANTÓNIO DA ROCHA NUNES
Julho de 2020

POLITÉCNICO
DO PORTO

Automação de Integração de Dados com Ferramenta de Business Intelligence

Marco António da Rocha Nunes

Dissertação para obtenção do Grau de Mestre em Engenharia Informática, Área de Especialização em Engenharia de Software

Orientador: Doutor Alexandre Bragança

Júri:
Presidente:

Vogais:

Porto, 5 de Julho de 2020

Dedicatória

Dedico aos meus pais, por toda a força e suporte que me deram ao longo destes anos, permitindo que este objetivo pessoal fosse alcançado.

Resumo

Nas organizações, a monitorização dos seus processos com a definição de indicadores de desempenho específicos para acompanhar o seu estado e evolução revela-se como determinante no auxílio do processo de tomada de decisão.

A forma como a informação é obtida deve ser rápida e automatizada, não despendendo recursos em tarefas repetitivas. Desta forma, a automatização de processos deve ser considerada, levando a que as decisões sejam tomadas dentro dos prazos pretendidos, de modo a que haja mais lucro e produtividade nas organizações.

Atualmente, na empresa *DevScope*, o processo de obtenção e integração de dados de uma das suas aplicações, a aplicação *web PowerBI ScoreCards*, com uma ferramenta de *Business Intelligence*, para a criação de relatórios a serem entregues aos seus clientes é um processo que se apresenta como repetitivo e moroso.

A realização desta dissertação pretende criar uma solução que implemente um processo que contraste com o atual, de forma a que haja uma integração automática dos dados da aplicação *web PowerBI ScoreCards* com uma ferramenta de *Business Intelligence*, caracterizando-se por ser um processo automatizado e agilizado.

Palavras-chave: *Business Intelligence*, Automação, Integração de Dados

Abstract

In the organizations, the monitoring of their processes with the definition of specific key performance indicators to follow its status and evolution is a determining factor in assisting the decision-making process.

The way in which information is obtained must be fast and automated, without spending resources on repetitive tasks. Therefore, the automation of processes must be considered, leading to decisions being made within the intended deadlines, in order to have more profit and productivity in the organizations.

Currently, at DevScope company, the process of obtaining and integrating data from one of its applications, the web application PowerBI ScoreCards, with a Business Intelligence tool, for creating reports to be delivered to its customers is a process that presents itself as repetitive and time-consuming.

This master thesis intends to create a solution that implements a process that contrasts with the current one, in order to have an automatic integration of the data from the web application PowerBI ScoreCards with a Business Intelligence tool, characterized by being an automated and streamlined process.

Keywords: Business Intelligence, Automation, Data Integration

Agradecimentos

Agradeço ao Instituto Superior de Engenharia do Porto e a todos os docentes pelos ensinamentos que me ofereceram ao longo destes anos.

Em seguida, quero agradecer à empresa *DevScope* por me permitir realizar a dissertação de mestrado e a toda a equipa de desenvolvimento com quem eu tive a oportunidade de contactar.

Agradeço também ao professor Alexandre Bragança, por todas as dúvidas esclarecidas e conselhos dados que permitiram a realização desta dissertação.

Um agradecimento a todos os meus amigos que me ajudaram ao longo desta caminhada.

Por último, deixo o agradecimento mais especial, ao qual agradeço aos meus pais. Sem eles nada disto era possível!

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Código

Lista de Acrónimos

AHP	Analytic Hierarchy Process.
API	Application Programming Interface.
Azure AD	Azure Active Directory.
BI	Business Intelligence.
DAX	Data Analysis Expression.
DW	Data Warehouse.
ETL	Extract, Transform, Load.
HTTP	Hypertext Transfer Protocol.
OLAP	Online Analytical Processing.
OLTP	Online Transaction Processing.
REST	Representational State Transfer.
SaaS	Software as a Service.
SQL	Structured Query Language.
UML	Unified Modeling Language.
URL	Uniform Resource Locator.

Capítulo 1

Introdução

Neste capítulo é abordado o contexto onde esta dissertação se insere, a apresentação do problema, os objetivos a atingir, a abordagem a adotar para resolver o problema e, por último, é referida a estrutura do documento.

1.1 Contexto

Hoje em dia, a gestão e análise de dados analíticos de uma organização são pontos fundamentais para o desenvolvimento do negócio, uma vez que as decisões estratégicas são suportadas pela informação adquirida a partir dos mesmos [1]. Face a isto, deve ser efetuado um acompanhamento sistemático das atividades e processos pretendidos, e, para tal, a informação disponível deve estar organizada e sintetizada, com a finalidade de permitir uma correta tomada de decisão e visão estratégica por quem a realiza [2].

Num mercado competitivo, a rapidez com que a informação é disponibilizada sobre o estado atual das tarefas deve incluir processos de obtenção da informação rápidos e automatizados, de modo a melhorar as atividades principais e a criar vantagem competitiva às organizações nos mercados [3]. Deste modo, a informação que as organizações possuem sobre si próprias revela-se um fator importante, e se a mesma for traduzida em conhecimento pode ter um impacto significativo na melhoria do seu negócio [4].

Como forma de acompanhar o estado dos objetivos e metas a atingir por uma organização, a definição de indicadores de desempenho é uma opção a adotar [5]. Para se visualizar o seu estado e desempenho, estes podem ser observados em elementos visuais expostos, por exemplo, num *dashboard*, verificando, assim, se os objetivos estão a ser cumpridos face às metas inicialmente estabelecidas [6].

Neste contexto, ferramentas de Business Intelligence (BI) tem assumido um destaque importante no processo de recolha, preparação e representação dos dados em relatórios e/ou *dashboards*, auxiliando a gestão estratégica das organizações.

Este trabalho desenvolvido surge na empresa *DevScope*, que pretende automatizar o processo de integração de dados de uma das suas aplicações, a aplicação *web PowerBI ScoreCards*, com uma ferramenta de BI, sendo o seu propósito facilitar o trabalho da equipa de BI ao realizar as suas tarefas relativas à representação de dados.

1.2 Problema

A aplicação *web PowerBI ScoreCards* desenvolvida pela empresa *DevScope* é uma aplicação *web* de gestão que permite aos seus clientes estabelecer objetivos individuais e de equipa, planear e definir indicadores de desempenho estratégicos. Com os dados que são introduzidos na aplicação *web* e/ou obtidos automaticamente dos sistemas de base de dados dos clientes, é possível verificar se os vários objetivos estabelecidos estão conforme o que foi definido.

Contudo, face à quantidade de dados, existe a necessidade de oferecer aos clientes um maior detalhe na representação e consulta dos mesmos. Para tal, o processo de criação de relatórios e *dashboards* para acompanhar o estado e evolução dos indicadores de desempenho são opções a tomar para garantir maior qualidade de análise.

Atualmente, este processo de criação de relatórios sobre a informação dos indicadores de desempenho de um cliente é realizado por uma equipa de BI. Para tal, é utilizada uma ferramenta de BI, *Microsoft Power BI* [7], em que se acede aos dados da base de dados da aplicação *web PowerBI ScoreCards* e, seguidamente, são elaborados relatórios com diferentes visualizações gráficas.

Posto isto, este processo nem sempre é possível de se realizar em tempo útil, já que, muitas das vezes, necessita da intervenção de duas equipas: a equipa de desenvolvimento da aplicação *web PowerBI ScoreCards* e da equipa de BI, caracterizando-se por ser demorado e repetitivo, uma vez que, para cada cliente, o fluxo de obter dados do cliente e criar respetivo relatório é repetido. Desta forma, a automatização e uniformização deste processo descrito é algo necessário, de modo a aumentar o próprio valor da aplicação *web PowerBI ScoreCards*, pois os seus dados passam a estar integrados de forma automática com uma ferramenta de BI, assim como, reduzir custos e tempo de implementação das tarefas a realizar, indo ao encontro das necessidades dos clientes.

1.3 Objetivos

O objetivo principal é a integração automática dos dados da aplicação *web PowerBI ScoreCards* com uma ferramenta de BI, de modo a que o processo de criação de relatórios seja mais automatizado. Este relatório será entregue aos clientes para permitir o acompanhamento dos seus processos e tomarem decisões pela visualização de dados.

A solução deve ter a capacidade de aceder aos dados da aplicação *web PowerBI ScoreCards* para permitir a representação da informação nos relatórios que são entregues aos clientes, sendo um processo automatizado e agilizado. Os clientes terão acesso a um relatório personalizado com visualizações gráficas interativas, que suporte diferentes análises baseadas na filtragem de dados.

Assim sendo, os objetivos passam por:

- Desenvolver um mecanismo de integração automático dos dados da aplicação *web PowerBI ScoreCards* com uma ferramenta de BI para agilizar o processo de criação de relatórios.
- Criar um relatório genérico que sirva como modelo (*template*) para representar a informação de qualquer cliente.

1.4 Abordagem

De forma a abordar o problema indicado e dar resposta às suas necessidades, é seguido um conjunto de passos:

- Estudar o conceito de BI e a arquitetura de um sistema de BI.
- Realizar uma pesquisa sobre as ferramentas de BI e analisar as suas funcionalidades.
- Identificar e explicar o processo e arquitetura existente.
- Analisar e desenhar diferentes alternativas/soluções e escolher a melhor solução.
- Implementar a solução desenhada.
- Avaliar a solução desenvolvida.

1.5 Estrutura do Documento

Este documento encontra-se dividido em sete capítulos.

O primeiro capítulo é relativo à introdução, onde é apresentado o contexto, o problema, os objetivos e a abordagem a seguir.

O segundo capítulo é referente ao contexto e estado da arte, sendo abordado o conceito de *Business Intelligence*, a arquitetura de um sistema deste conceito e seus processos constituintes. É efetuado um estudo das ferramentas de BI e sua análise comparativa. É também apresentada a arquitetura do processo existente e, por último, é referida uma visão da solução.

O terceiro capítulo contempla a análise de valor, onde se recorre a diferentes técnicas e métodos para avaliar o processo a ser desenvolvido.

O quarto capítulo expõe a análise e design, sendo apresentada a solução atual, a definição de requisitos do novo processo a desenvolver, a exposição de artefactos para as diferentes soluções e a escolha da solução. Por último, relativamente à solução escolhida, é ilustrado um artefacto de como esta será implantada.

O quinto capítulo explicita a implementação da solução desenhada e detalhes técnicos que a mesma aborda. Apresenta-se o ambiente de desenvolvimento necessário com indicação das ferramentas e tecnologias utilizadas.

O sexto capítulo apresenta a experimentação e avaliação da solução implementada, com a definição dos indicadores de avaliação, as suas hipóteses e a metodologia de avaliação. Conforme os métodos de avaliação aplicados na solução desenvolvida, são avaliados os resultados obtidos.

O sétimo capítulo aborda a conclusão da dissertação, apresentando-se os objetivos alcançados, as limitações apontadas e o trabalho futuro a desenvolver.

Capítulo 2

Contexto e Estado da Arte

Neste capítulo é abordado o conceito de Business Intelligence, que é a área de conhecimento onde se enquadra esta dissertação, sendo referida e explicada a arquitetura de um sistema de *Business Intelligence* e seus processos constituintes.

É efetuado um estudo das ferramentas de Business Intelligence existentes, com apresentação das suas funcionalidades, sendo, seguidamente, realizada uma análise comparativa baseada em vários critérios.

Por último, é apresentada e explicada a arquitetura do processo existente e a descrição da aplicação *web PowerBI ScoreCards*, sendo também referida uma visão da solução para o projeto desta dissertação.

2.1 Business Intelligence

O conceito de BI foi introduzido pela primeira vez por Luhn em 1958 [8], definindo inteligência como "a capacidade de apreender as inter-relações dos factos apresentados, de forma a guiar a ação em direção a uma meta desejada"[1].

Este termo é associado como um conjunto de sistemas e técnicas que visam analisar dados de negócio, suportado pelo uso de tecnologias para permitir visualizar e representar a informação. Desta forma, no âmbito de análise da informação, são importantes os sistemas de BI.

Um sistema de BI tem tarefas relativas à agregação, integração e análise de dados provenientes de várias fontes de informação. Além disto, deve ser capaz de transformar os dados em informação e conhecimento, de modo a promover uma tomada de decisão eficaz, assim como, um pensamento estratégico nas organizações, promovendo uma maior qualidade nas mesmas [4]. Deste modo, este sistema deve suportar a tomada de decisão em todos os níveis de gestão de uma organização, independentemente do seu nível de estruturação [4].

Por último, um sistema de BI consiste, essencialmente, em analisar os dados com ferramentas analíticas, fornecendo informação detalhada aos gestores, diretores e pessoas que analisam a informação, auxiliando de forma mais rápida e eficaz o processo de tomada de decisão [9].

[1] Tradução livre do autor. No original "the ability to apprehend the interrelationships of presented facts in such a way as to guide action towards a desired goal."

2.1.1 Arquitetura de um Sistema Business Intelligence

Segundo Chaudhuri, Dayal e Narasayya [9], a estrutura típica de um sistema de BI consiste em cinco camadas, tal como se encontra ilustrado na Figura 2.1.

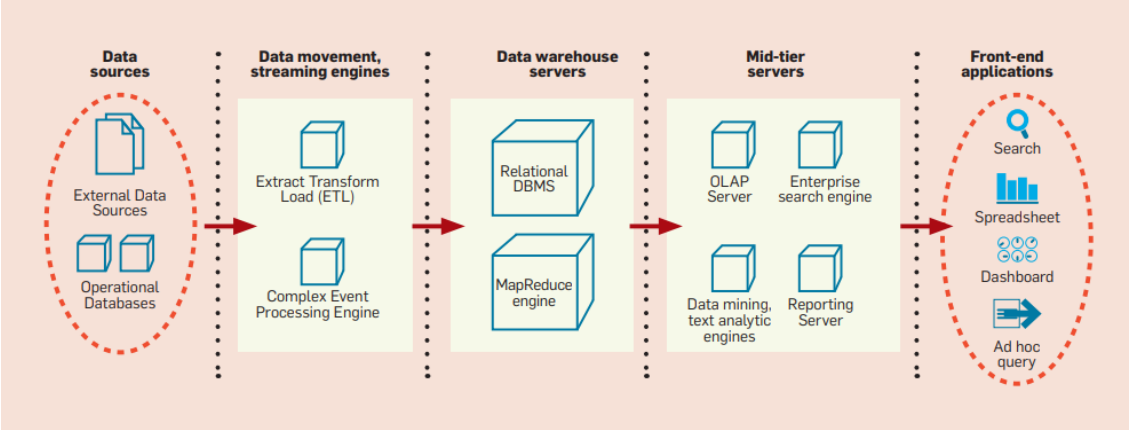


Figura 2.1: Arquitetura de um Sistema de Business Intelligence [9]

Pela análise da Figura 2.1, verifica-se que a primeira camada desta arquitetura é relativa a extrair dados de diferentes fontes de dados, sendo, normalmente, obtidos de várias base de dados operacionais da organização. Uma vez que diferentes fontes contêm os dados em formatos diferentes, é necessário resolver os problema de integração e limpeza dos dados.

Para tal, o processo de Extract, Transform, Load (ETL), apresentado na Secção 2.1.2, resolve os problemas indicados anteriormente, garantindo o correto formato e consistência dos dados. Após isto, os dados são carregados para um repositório destino - Data Warehouse (DW).

Posteriormente, uma das camadas seguintes corresponde ao complemento do DW por um conjunto de servidores *mid-tier*. Para tal, é utilizado sistemas Online Analytical Processing (OLAP) (explicado na Secção 2.1.4) para representar os dados de forma multidimensional e permitir operações comuns, sendo estas, agregação ou filtragem de dados. Contudo, tem surgido outras alternativas, tais como, servidores de relatórios (definição, execução e criação de relatórios); mecanismos de pesquisa (pesquisa por palavras-chave aos dados); mecanismos de *data mining* (capacidade de efetuar modelos preditivos e obter padrões nos dados); mecanismo de análise de texto (analisar um grande volume de texto proveniente, por exemplo, de comentários de clientes, de modo a obter conhecimento que pode ser valioso para a organização)[9].

Na última camada desta arquitetura, existe um conjunto de aplicações *front-end* que são utilizadas para realizar as tarefas de BI. Neste caso, englobam-se a análise de dados em *spreadsheet*, ferramentas para elaborar pesquisas e aplicações de gestão de desempenho, que permite a quem toma decisões acompanhar o estado dos indicadores de desempenho no *dashboard*. Em adição, a área móvel de BI tem vindo a crescer, permitindo a realização de análise de dados e acompanhamento de métricas em diapositivos móveis.

Relatório e Dashboard

A informação a analisar pode ser apresentada por meio de relatórios e/ou *dashboards*. Um relatório é uma apresentação mais detalhada da informação proveniente de uma fonte de dados, podendo ser constituído por várias páginas [10]. Por sua vez, um *dashboard* é um conjunto de visualizações que se encontram organizadas e consolidadas numa única janela, de modo a que a informação mais importante possa ser monitorizada de forma mais rápida e simples [11].

No relatório, a informação apresentada é mais extensa do que no *dashboard*, contendo um número mais elevado de elementos visuais, enquanto que, num *dashboard*, os elementos visuais apresentados pretendem fornecer uma rápida compreensão de forma resumida e concisa à análise que se pretende efetuar [10]. Normalmente, um relatório tende a ser mais complexo, uma vez que, potencialmente, fornece um histórico de toda a informação, ao contrário do *dashboard*, que pretende ser simples e objetivo na comunicação da sua informação [12].

No caso da Figura 2.2, encontra-se um exemplo de uma página de um relatório, que contém vários visuais, promovendo um nível elevado de detalhe dos dados atuais e seu histórico. Este exemplo de página de relatório permite examinar os dados de forma minuciosa, avaliando a disponibilidade em horas tanto de forma geral (por departamento e/ou grupo), bem como, por colaborador (ao longo dos meses de um ano).

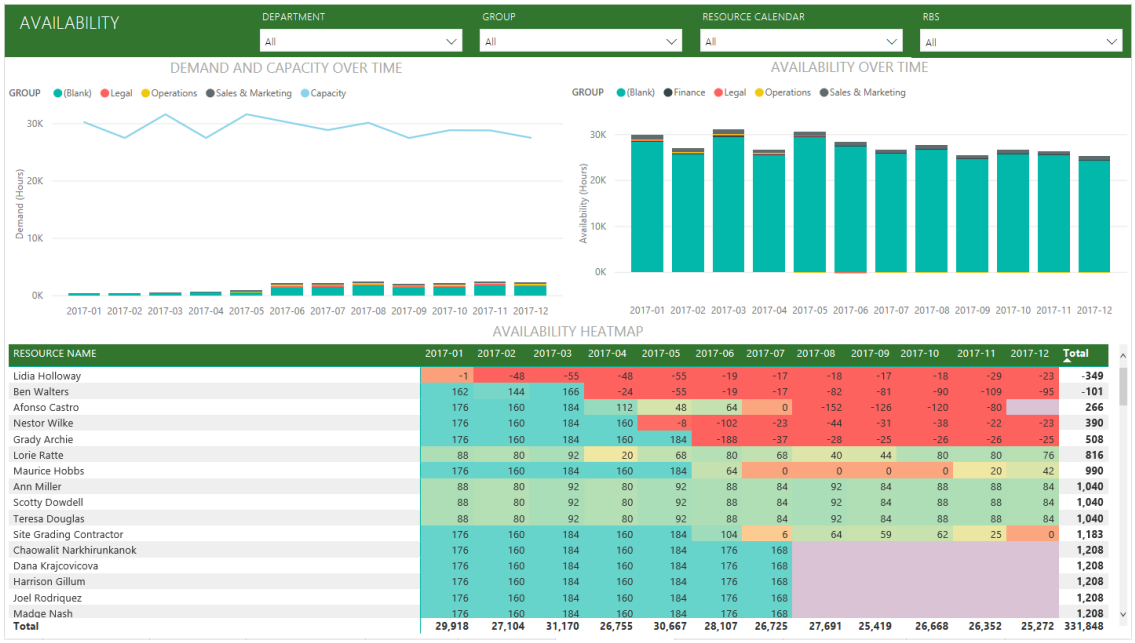
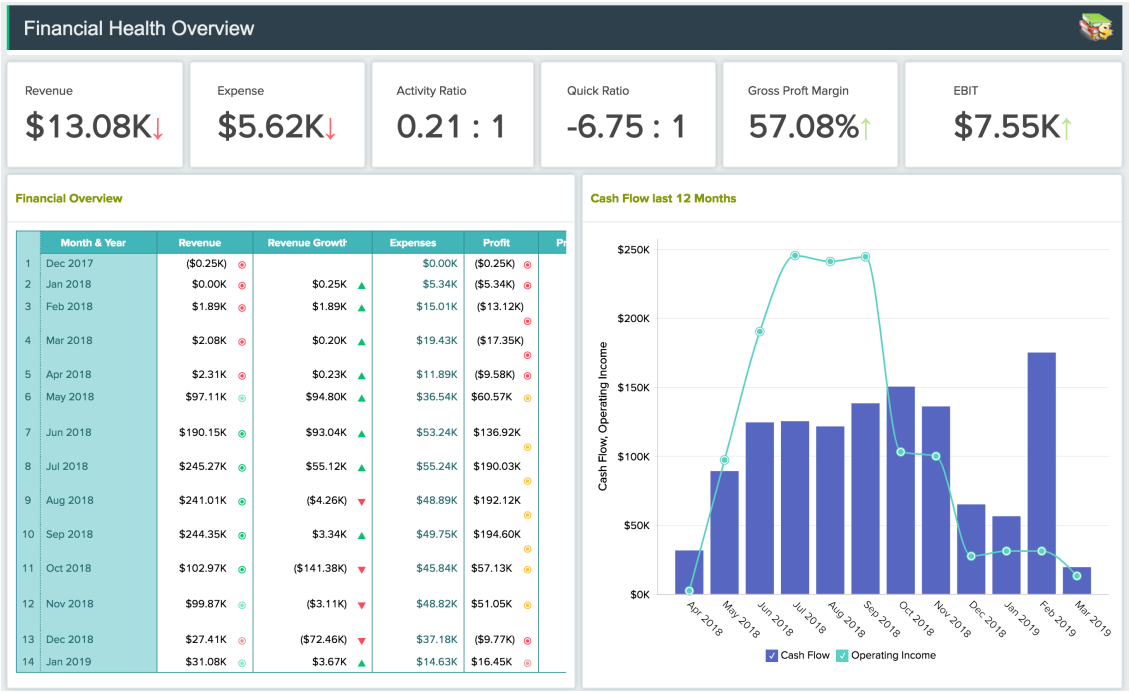


Figura 2.2: Exemplo de um Relatório [13]

A Figura 2.3 ilustra um exemplo de um *dashboard*, que apresenta a informação de forma muito objetiva, fornecendo indicações precisas sobre o valor que se regista em determinado aspeto de negócio. Ao contrário da informação presente no relatório apresentado na Figura 2.2, que tende a ser mais detalhada e elaborada, a informação presente neste *dashboard* é simples e concisa, de maneira a ser rapidamente entendida.



	Month & Year	Revenue	Revenue Growth	Expenses	Profit
1	Dec 2017	($0.25K)		$0.00K	($0.25K)
2	Jan 2018	$0.00K	$0.25K	$5.34K	($5.34K)
3	Feb 2018	$1.89K	$1.89K	$15.01K	($13.12K)
4	Mar 2018	$2.08K	$0.20K	$19.43K	($17.35K)
5	Apr 2018	$2.31K	$0.23K	$11.89K	($9.58K)
6	May 2018	$97.11K	$94.80K	$36.54K	$60.57K
7	Jun 2018	$190.15K	$93.04K	$53.24K	$136.92K
8	Jul 2018	$245.27K	$55.12K	$55.24K	$190.03K
9	Aug 2018	$241.01K	($4.26K)	$48.89K	$192.12K
10	Sep 2018	$244.35K	$3.34K	$49.75K	$194.60K
11	Oct 2018	$102.97K	($141.38K)	$45.84K	$57.13K
12	Nov 2018	$99.87K	($3.11K)	$48.82K	$51.05K
13	Dec 2018	$27.41K	($72.46K)	$37.18K	($9.77K)
14	Jan 2019	$31.08K	$3.67K	$14.63K	$16.45K

Figura 2.3: Exemplo de um Dashboard [14]

2.1.2 Processo de ETL

Como forma de facilitar a gestão de um DW, pode-se fazer uso do processo de ETL. Existem ferramentas ETL que, de uma forma geral, permitem identificar a informação importante da fonte de dados e proceder à sua extração.

Posteriormente, as ferramentas ETL customizam e integram a informação proveniente de uma ou múltiplas fontes de dados no formato correto, limpando e filtrando esses conjunto de dados, conforme as regras de negócio e a base de dados definida. Por último, carregam os dados para o DW respetivo.

O processo ETL é constituído por três fases, tal como ilustrado na Figura 2.4.

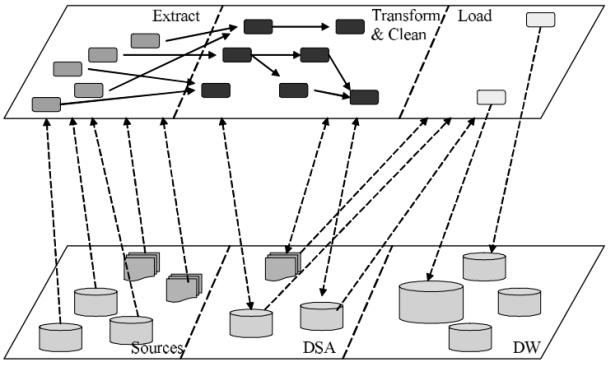


Figura 2.4: Processo ETL [15]

Estas três fases consistem em [15]:

- Extract (Extração) – aceder e extrair os dados de várias fontes de dados.
- Transform (Transformação) – limpar, agregar e tratar os dados, aplicando o formato e estrutura desejada. Esta operação é realizada num *Data Staging Area* (DSA).
- Load (Carregamento) – carregar os dados para um DW.

2.1.3 Data Warehouse

Um DW refere-se a um repositório de dados que está separado da base de dados da organização, mantendo um histórico de dados consolidados com informação que suporta as decisões estratégicas de uma organização [16].

Segundo William H. Immon [17], este apresenta quatro conceitos chave que caraterizam um DW como "*data warehouse is a subject-oriented, integrated, time-variant, and nonvolatile collection of data in support of management's decision making process*".

Seguidamente, apresenta-se uma breve descrição para cada um conceitos apresentados anteriormente.

- *Subject-oriented* – Uma vez que um DW tem como finalidade modelar e preparar dados para análise, estes devem ser relativos ao que se pretende, excluindo dados que não são úteis para o processo de tomada de decisão.
- *Integrated* – Um DW é construído integrando dados de várias fontes de dados heterogéneas e, para tal, devem ser aplicadas técnicas para garantir a consistência dos mesmos.
- *Time-variant* – Os dados são guardados para fornecer informação sobre o seu histórico.
- *Nonvolatile* – Um DW é um armazém de dados que foram transformados e que se encontram separados dos dados da aplicação em ambiente de produção.

Para além disto, um DW permite a utilização de ferramentas OLAP para a análise interativa dos dados, que podem ser visualizados em diferentes perspetivas, fazendo uso de um cubo de dados para fornecer uma visão multidimensional da informação [16].

2.1.4 Operational Database Systems e Data Warehouses

A maior parte das tarefas que são realizadas em *Operational Database Systems* são para executar transações *online*, ao qual se dá o nome de Online Transaction Processing (OLTP). Estes sistemas são muito utilizados nas operações que são realizadas diariamente (compras, contabilidade e registos).

Por outro lado, um DW, tal como referido anteriormente, é utilizado para os utilizadores a quem se destina analisar os dados e tomar decisões. Como forma de apresentar e organizar os dados nos devidos formatos, são utilizados sistemas OLAP.

Posto isto, apresentam-se, de seguida, as diferenças entre OLAP e OLTP nas diversas características [16]:

- Orientação ao sistema e utilizadores – Um sistema OLAP é usado para análise de dados por gestores ou diretores, enquanto que, um sistema OLTP, é usado para transações e é utilizado por clientes.
- Conteúdo dos dados – Um sistema OLAP é utilizado para grande volume de dados e para a sua análise e decisão em diferentes níveis de granularidade. Nos sistemas OLTP, estes utilizam os dados atuais que são demasiados específicos e detalhados para serem utilizados nos processos de decisão.
- Design da base de dados – Um sistema OLAP adota um *design* de base de dados mais orientado ao assunto de análise (*subject-oriented*). Relativamente ao sistema OLTP, o *design* da base de dados é orientado à aplicação (*application-oriented*).
- Vista – Um sistema OLAP trabalha com informação proveniente de várias fontes de dados e armazena dados da evolução da organização. Por outro lado, um sistema OLTP é relativo aos dados atuais da organização.
- Padrões de acesso: No sistema OLAP, as operações a realizar são de leitura ao DW, enquanto que, num sistema OLTP, os acessos consistem em transações atómicas - conjunto de operações que devem ser todas executadas, caso contrário, a transação atómica falha.

2.2 Ferramentas de Business Intelligence

Nesta secção, é realizado um estudo e análise das ferramentas de BI existentes, efetuando um levantamento das funcionalidades que cada uma apresenta.

Como forma de se ter uma visão de quais as melhores ferramentas de BI existentes, utilizou-se o Quadrante Mágico da *Gartner* [18], que resulta numa pesquisa sobre uma área específica, permitindo saber quais as posições que ocupam os vários concorrentes no mercado.

A Figura 2.5 apresenta o quadrante mágico para as ferramentas de BI do ano de 2020, resultante da pesquisa efetuada pela *Gartner* seguindo um conjunto uniforme de critérios de avaliação.

Figura 2.5: Quadrante Mágico das Ferramentas de BI [19]

Pela análise da Figura 2.5, verifica-se que as diferentes ferramentas de BI se encontram classificadas como [19]:

- *Leaders* – Representam uma forte compreensão das suas capacidades e funcionalidades, comprometendo-se com o sucesso do cliente. Na área de BI, ao nível da decisão de compra, os utilizadores pretendem ferramentas que sejam fáceis de utilizar, não necessitando de muito conhecimento técnico.
- *Challengers* – Marcam uma posição bem sucedida no mercado, no entanto, estão limitados a casos de uso específicos, ambientes técnicos ou a certos domínios de aplicação.
- *Visionaries* – Possuem uma visão forte e única naquilo que se comprometem a oferecer, contudo, ainda têm lacunas ao abordar requisitos mais abrangentes, o que tem consequências ao nível do seu crescimento e consistência.
- *Niche Players* – Posicionam-se num segmento específico do mercado, concentrando-se em certos aspectos específicos de BI. Apesar disto, podem ter falhas em funcionalidades mais amplas, podendo apresentar recursos limitados de implementação e suporte, assim como, uma base de clientes baixa.

Seguidamente, serão analisadas as ferramentas de BI que se encontram no quadrante *Leaders*, uma vez que apresentam-se como sendo as melhores em termos das suas capacidades e funcionalidades de utilização. Para tal, serão detalhadas quatro ferramentas: *Microsoft Power BI, Tableau, Qlike* e *ThoughtSpot*.

2.2.1 Microsoft Power BI

O *Microsoft Power BI* [7] consiste numa ferramenta de análise de dados de negócio criada em 2013 pela *Microsoft*. Esta ferramenta baseia a sua interação em *drag and drop*, possibilitando a representação dos dados em vários visuais/gráficos, com o intuito de ajudar os utilizadores a tomarem as melhores decisões.

A ferramenta permite a conexão a várias fontes de informação, fazer uma preparação e transformação de dados, elaborar relatórios e publicá-los no *Power BI Service* [20], que é a plataforma *online* que permite a publicação de relatórios e, através destes, criar *dashboards* personalizados para uma visão mais intuitiva dos dados.

Desta forma, o *Microsoft Power BI* consiste nas seguintes partes:

- *Power Query*, que funciona como um conetor que se liga à fonte de dados pretendida (ficheiros *Excel, JSON*, base de dados do tipo Structured Query Language (SQL), Application Programming Interface (API), entre muitas outras) para obter e preparar os dados conforme o pretendido.
- *Power Pivot*, que é uma técnica de modelação de dados, permitindo criar modelos de dados e estabelecer relações entre estes. Além disto, fornece uma linguagem Data Analysis Expression (DAX) para definição de, por exemplo, métricas, fórmulas e funções.
- *Power View*, que corresponde a uma técnica de visualização de dados para criar gráficos interativos.
- *Power BI Desktop*, que é uma aplicação *desktop* que engloba as três funcionalidades anteriormente explicadas, com o intuito de criar um relatório.
- *Power BI Service*, que é uma aplicação *online* Software as a Service (SaaS), em que os relatórios criados previamente no *Power BI Desktop* podem ser publicados, dando a possibilidade de serem visualizados e partilhados com outros utilizadores via *web browser*. Nesta aplicação, através dos relatórios publicados e selecionando os visuais pretendidos, podem ser criados *dashboards*.
- Aplicação móvel *Power BI* para *Windows*, *iOS* e *Android*, onde podem ser visualizados os relatórios e *dashboards* previamente criados.

Nesta ferramenta, é possível restringir o acesso dos utilizadores aos dados, permitindo uma maior segurança da informação. Assim, na aplicação *Power BI Desktop*, pode-se fazer uso do *row-level security*, que estabelece diferentes papéis (*roles*), definindo quais os utilizadores a terem acesso aos dados ao nível das linhas. Com isto, para cada *role*, os filtros a serem aplicados a cada linha de um conjunto de dados é realizado por via de regras escritas em DAX, filtrando a informação que essa *role* pode visualizar. Contudo, a definição concreta de quais os utilizadores que devem ficar associados a cada *role* é realizada na aplicação *Power BI Service*.

Posto isto, o *Power BI Desktop* permite que sejam adicionados novos conetores aos já existentes, como forma de permitir conexão a outras fontes de dados. Estes conetores são desenvolvidos com a linguagem M, usando o *Power Query SDK* [21], que é uma ferramenta de desenvolvimento de conetores para a ferramenta *Microsoft Power BI*[21], mais concretamente para a aplicação *desktop* (*Power BI Desktop*).

O conetor permite criar novas fontes de dados ou customizar uma fonte de informação existente. Os casos típicos são [22]:

- Ser uma maneira fácil de conexão a API Representational State Transfer (REST).
- Colocar uma vista limitada e filtrada sobre uma fontes de dados para melhorar a usabilidade.
- Definir tipos de autenticação (*OAuth, API Key, Windows Authentication* e *Basic Authentication*) aquando ligação a fontes de dados [23].

Além disto, existe a possibilidade de serem desenvolvidos novos visuais gráficos à medida do que se pretenda (*Power BI Custom Visuals*) [24], sendo adicionados aos visuais já existentes, para depois serem usados nos relatórios e *dashboards*.

A Figura 2.6 ilustra um exemplo de relatório criado no *Power BI Desktop*.

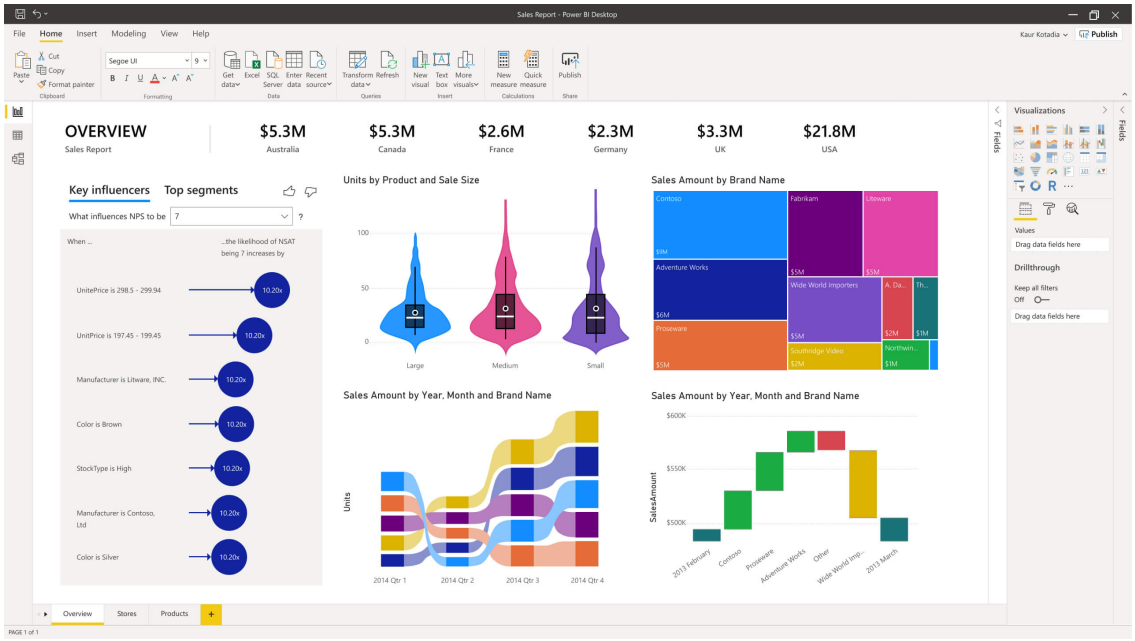


Figura 2.6: Exemplo de Relatório Criado com a Aplicação Power BI Desktop [7]

2.2.2 Tableau

A ferramenta *Tableau* [25], criada em 2003, é uma ferramenta de análise e visualização de dados. As suas funcionalidades permitem aos utilizadores representar os dados sob vários visuais, sendo que a interação que é realizada com a ferramenta é por um sistema *drag and drop*.

Esta ferramenta é constituída por um conjunto de produtos, que podem ser classificados em três categorias, sendo estas [26]:

- *Developer Tools*, onde se encontram os produtos *Tableau Desktop* e *Tableau Public*, que possibilita o desenvolvimento de visuais, relatórios e *dashboards*.

- *Sharing Tools*, que engloba os produtos *Tableau Online*, *Tableau Server* e *Tableau Reader*, e que permite a partilha de visualizações, relatórios e *dashboards* criados.
- *Data Preparation Tools*, que contém o produto *Tableau Prep*, para preparação dos dados a serem utilizados.

Seguidamente, é apresentada uma explicação para cada um dos produtos referidos anteriormente [26].

- *Tableau Desktop*, que é uma aplicação *desktop* paga que permite a conexão a várias fontes de dados, personalizar relatórios com o uso de vários visuais e serem criados *dashboards*.
- *Tableau Public*, que permite que as representações dos dados através de elementos visuais possam ser guardadas na nuvem pública do *Tableau*, significando que podem ser acedidos por qualquer utilizador.
- *Tableau Server*, que é uma aplicação servidora paga que precisa de ser instalada num servidor *Windows* ou *Linux*. É utilizado para partilhar relatórios e publicar *dashboards* a partir do *Tableau Desktop*, sendo partilhados por toda a organização.
- *Tableau Online*, que é uma aplicação servidora paga que, ao contrário do *Tableau Server*, está instalada nos servidores do *Tableau*. Apresenta as mesmas funcionalidades do produto anterior.
- *Tableau Reader*, que é uma aplicação *desktop* grátis que permite visualizar os *dashboards* criados no *Tableau Desktop*, sendo possível realizar filtros no mesmo. Contudo, não é possível editar os visuais.
- *Tableau Prep*, que é responsável pela preparação de dados, fazendo com que haja a limpeza, agregação e conversão dos mesmos para serem analisados, funcionando com uma espécie de ETL.

Além dos conetores existentes, permite que sejam criados novos para se conetarem a fontes de dados pretendidas, usando o *Tableau Connector SDK* [27]. Contudo, não suporta a criação de novos visuais, como forma de estes serem adicionados aos visuais já existentes.

A Figura 2.7 apresenta um exemplo de um relatório criado no *Tableau Desktop*.

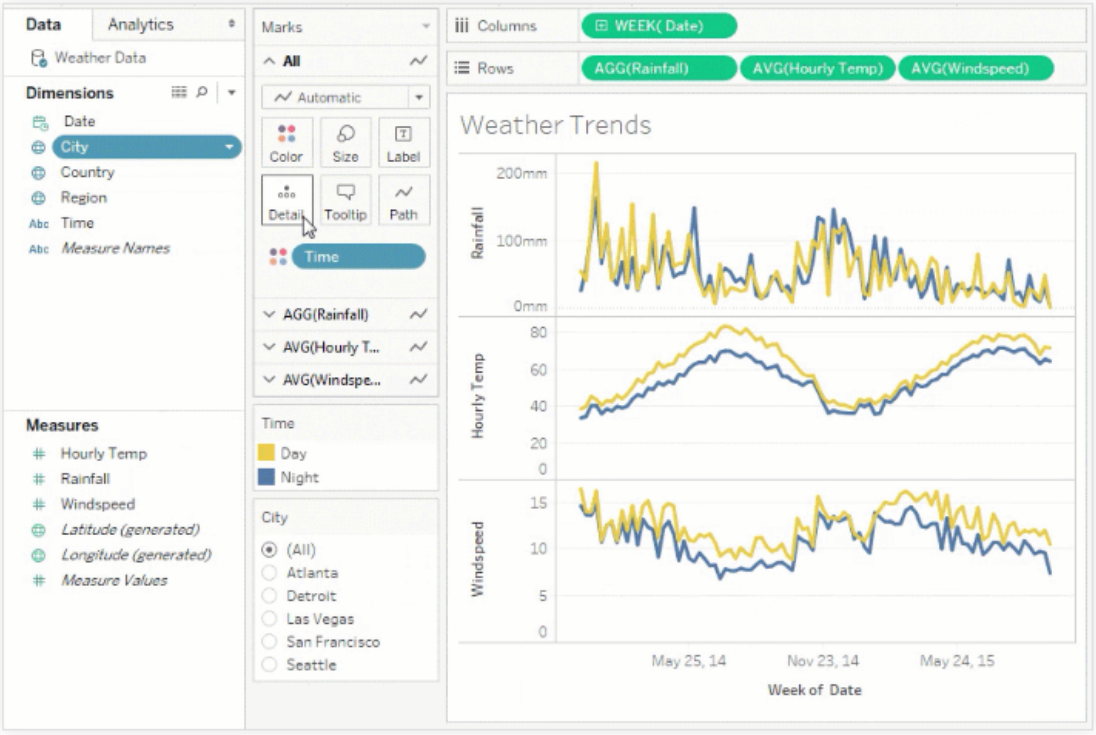


Figura 2.7: Exemplo de Relatório Criado com a Aplicação Tableau Desktop [25]

2.2.3 Qlik

O *Qlik* [28], criado em 1993, tem como seu principal produto o *Qlik Sense* [29], que é uma plataforma analítica que permite personalizar relatórios e elaborar *dashboards* detalhados. A ferramenta permite a conexão a várias fontes de informação, promove a correta associação e relação dos dados para posterior filtragem e representação gráfica e utiliza um sistema de *drag and drop*.

O *Qlik Sense* faz uso das seguintes ferramentas [29]:

- *Qlik Sense Enterprise*, que corresponde à versão completa do *Qlik Sense* e permite todo o processo de explorar dados e representá-los graficamente, funcionando como uma aplicação analítica que dá resposta aos processos de negócio, permitindo que as visualizações dos dados e *dashboards* criados sejam publicados na nuvem do *Qlik Sense*.
- *Qlik Sense Cloud*, onde é possível criar, partilhar e publicar as visualizações criadas, sendo isto realizado através de um *web browser*.
- *Qlik Sense Desktop*, que é uma aplicação *Windows* responsável por se ligar às fontes de dados e pela criação de relatórios e *dashboards*, que podem ser exportados e usados no *Qlik Sense Enterprise* para depois serem publicados na nuvem.
- *Qlik Sense Mobile App*, que é uma aplicação móvel disponível para os sistemas *Android* e *iOS*, que possibilita aos utilizadores visualizar relatórios e *dashboards* a que tem acesso e fazer *download* dos mesmos.

Face aos conetores existentes na ferramenta *Qlik Sense Desktop*, é possível de serem desenvolvidos novos, usando o *QVX SDK* [30]. Apesar disto, novos visuais podem ser criados e adicionados aos existentes, fazendo uso da biblioteca de *widgets* do *Qlik Sense* [31].

A Figura 2.8 ilustra um relatório criado com o *Qlik Sense Desktop*.

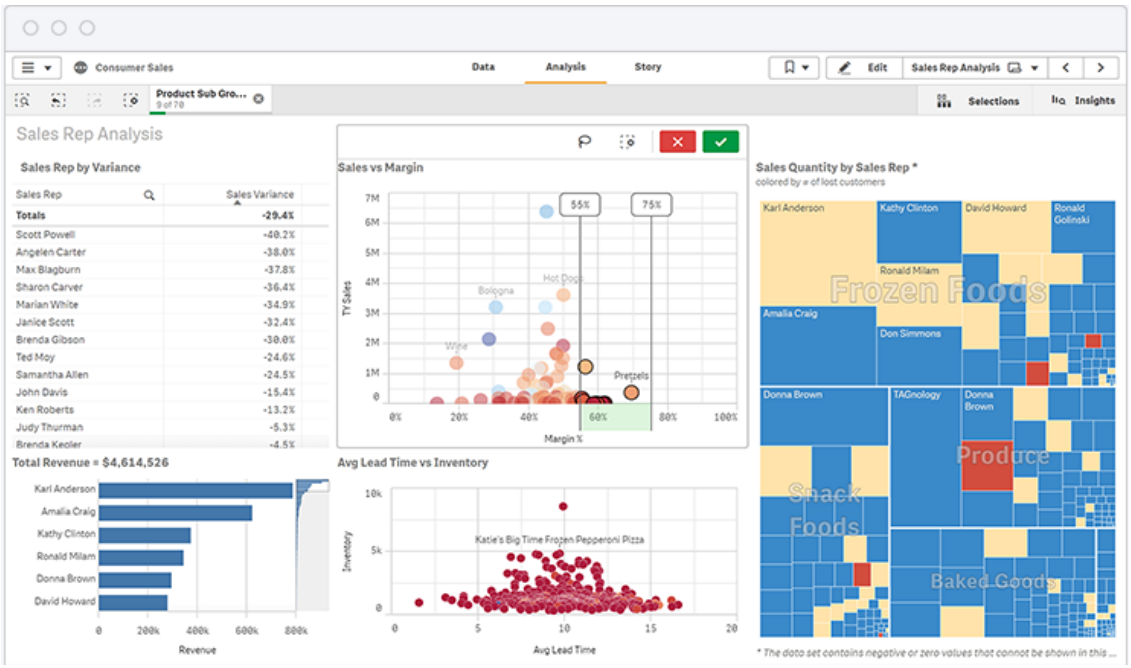


Figura 2.8: Exemplo de Relatório Criado com a Ferramenta Qlik Sense Desktop [28]

2.2.4 ThoughtSpot

O *ThoughtSpot* [32] é uma ferramenta de BI criada em 2012, que permite a sua interação por via de *drag and drop*. Esta ferramenta consiste em analisar dados, criar relatórios e *dashboards*, através da sua capacidade de conetar a diversas fontes de dados e ao seu motor de pesquisa, que permite criar visuais sobre determinado assunto a visualizar por meio de questões que se introduzem no campo de pesquisa.

Esta ferramenta permite partilhar informação com os utilizadores e definir uma granularidade de acesso aos mesmos, ou seja, definir o acesso a determinadas colunas e linhas e outras partes da informação, como forma de controlar a autorização a dados sensíveis.

O *ThoughtSpot* apresenta-se como uma única e simples solução de BI para realizar todas as funcionalidades enunciadas anteriormente, apresentando apenas outra ferramenta, o *ThoughtSpot Mobile* [33], disponível apenas para *i0S*, que permite o acesso à informação em dispositivos móveis.

Esta ferramenta não permite a extensão e desenvolvimento de novos conetores de ligação a fontes de informação e de visuais gráficos para representar os dados.

A Figura 2.9 mostra um relatório criado com o *ThoughtSpot*.

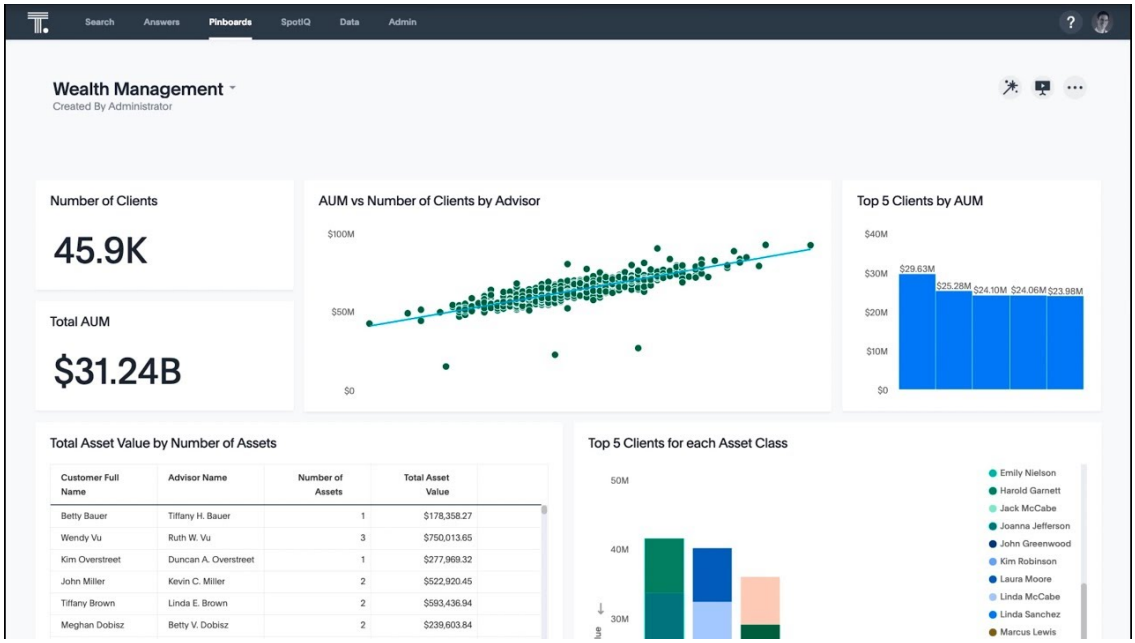


Figura 2.9: Exemplo de Relatório Criado com a Ferramenta ThoughtSpot [32]

2.2.5 Análise Comparativa das Ferramentas de Business Intelligence

Uma vez analisadas as ferramentas de BI conforme apresentado nas secções anteriores, torna-se relevante fazer uma análise comparativa das ferramentas, com o intuito de perceber os aspetos onde se destacam, avaliando qual a melhor ferramenta em determinadas características.

Posto isto, as quatro ferramentas serão comparadas nas características que se apresentam na Tabela 2.1, sendo estas evidenciadas pela *Gartner* como as mais importantes que as ferramentas de BI devem apresentar. As marcações assinaladas (x) indicam se a ferramenta em questão dispõe ou não da característica em questão.

Tabela 2.1: Análise Comparativa de Ferramentas

	Power BI	Tableau	Qlik	ThoughtSpot
Representação de dados em sistema *drag and drop*	x	x	x	x
Interação simples e intuitiva	x	x	x	x
Curva aprendizagem rápida	x	x	x	x
Criação de relatórios e *dashboards*	x	x	x	x
Incorporação de relatórios em aplicações *web*	x	x	x	x
Publicação de relatórios *online*	x	x	x	
Partilha de relatórios e *dashboards*	x	x	x	x
Restrição de utilizadores no acesso a dados	x			x
Conexão a fontes de dados	x	x	x	x
Desenvolvimento de novos conetores de conexão a fontes de dados	x	x	x	
Desenvolvimento de novos visuais	x		x	
Elevada comunidade de clientes e suporte	x			

Através da análise da Tabela 2.1, relativamente à característica de representação de dados, pode-se dizer que as todas as ferramentas apresentam um sistema de *drag and drop* para a representação dos dados, dispondo de vários visuais gráficos à escolha para os representar, sendo uma *interface* fácil e intuitiva para os intervenientes. Face a isto, todas as ferramentas apresentam um curva de aprendizagem rápida e fácil das suas funcionalidades.

No que diz respeito à análise visual de dados, todas as ferramentas promovem a criação de relatórios e, por conseguinte, a criação de *dashboards* personalizados para posterior análise da informação em tempo real, suportando o processo de tomada de decisão. Em todas as ferramentas, os relatórios de análise criados podem ser incorporados/embebidos numa aplicação *web*, o que permite ter uma visão dos dados representados no relatório diretamente na aplicação *web* que se pretenda.

Nesta análise de dados, no que respeita a interações com os visuais, todas as ferramentas permitem uma filtragem de dados à medida das necessidades do utilizador, sendo bastante interativo e intuitivo.

No caso das ferramentas *Microsoft Power BI, Tableau* e *Qlik*, tanto o relatório, bem como, os *dashboards*, podem ser publicados nas suas aplicações *online* e serem partilhados para que outros utilizadores tenham acesso. Por outro lado, a ferramenta *ThoughtSpot* permite apenas a partilha dos relatórios e/ou *dashboards* criados, não existindo a publicação *online*, devido a ser uma única solução de BI.

Ainda relativamente à partilha de acessos de relatórios e/ou *dashboards*, as ferramentas *Microsoft Power BI* e *ThoughtSpot* são as que permitem estabelecer uma maior segurança a dados mais sensíveis de negócio. No caso do *Microsoft Power BI*, é definido um conjunto de *roles*, que possui a informação filtrada à qual os utilizadores associados à *role* devem aceder. Relativamente à ferramenta *ThoughtSpot*, esta define uma granularidade de acesso, por exemplo, ao nível de linhas ou colunas, controlando o acesso dos utilizadores aos dados. As restantes ferramentas apenas definem com que utilizadores desejam partilhar a informação,

não apresentando a funcionalidade de restringir o acesso mais específico de dados por parte dos utilizadores.

No que concerne à característica de conexão a fontes de dados, as quatro ferramentas possibilitam a ligação a várias fontes de dados diferentes, desde ficheiros *Excel, XML e JSON*, até base de dados e API. No caso da ferramenta *Microsoft Power BI*, esta tem a vantagem de se integrar facilmente com produtos *Microsoft* e aplicações parceiras da mesma, tais como, *Oracle, Zendesk, MailChimp*, entre outros.

Em termos de desenvolvimento de novos conetores, como forma de permitir a conexão a fontes de dados que não existem ou customizar ligações a fontes existentes, as três ferramentas (*Microsoft Power BI, Tableau* e *Qlik*) oferecem esse propósito, graças à disponibilização de um *Software Development Kit* (SDK) respetivo. Um conetor permite, por exemplo, aceder a estruturas de dados de uma forma mais rápida e que oferecem proteção, tal como, uma API autenticada.

Desta forma, o desenvolvimento de conetores de ligação à fonte de dados pretendida pode facilitar a obtenção de dados com a ferramenta de BI, garantindo vantagens em termos de automatização. Assim, pode-se obter diretamente os dados pretendidos para a ferramenta de BI, sem necessitar de muito tratamento e limpeza dos mesmos, uma vez que parte disto pode ser realizado no conetor.

Relativamente ao desenvolvimento de novos visuais, para oferecer uma maior representação gráfica da informação, isto apenas é suportado nas ferramentas *Microsoft PowerBI* e *Qlik*.

No que se refere à comunidade de clientes e suporte, o facto da ferramenta *Microsoft Power BI* pertencer à *Microsoft* e poder ser integrada com outros produtos seus, faz com que tenha uma maior comunidade. Contudo, as outras três ferramentas também possuem uma boa comunidade, destacando-se a ferramenta *Tableau*, sendo que, a ferramenta *ThoughtSpot*, é a que apresenta uma menor comunidade, derivado também de ser a mais recente no mercado dessas três.

Posto isto, pode-se dizer que quase todas as ferramentas cumprem na totalidade as características enunciadas. As características onde não existe o cumprimento total de todas as ferramentas são relativas à publicação de relatórios *online*, à restrição de utilizadores no acesso a dados, ao desenvolvimento de novos conetores de conexão a fontes de dados, ao desenvolvimento de novos visuais e à elevada comunidade de clientes e suporte.

Em determinadas características, algumas ferramentas sobressaem-se relativamente a outras, destacando-se, neste aspeto, a ferramenta *Microsoft Power BI*, em termos de integração com várias fontes de dados e na sua comunidade. Para além disto, a ferramenta *Microsoft Power BI*, juntamente com a ferramenta *ThoughtSpot*, permitem partilhar relatórios/*dashboards* com outros utilizadores (tal como as restantes ferramentas), mas são as únicas que possibilitam definir acessos específicos a determinados dados, como forma de proteção da informação importante de negócio.

Por último, é de referir que, a ferramenta *ThoughtSpot*, apresenta-se como uma única solução de BI, sendo as restantes compostas por várias aplicações diferentes, contribuindo para a execução de diferentes tarefas no âmbito da área de BI.

2.3 Processo Atual

Nesta secção, é apresentado o processo atual existente na empresa onde surge o problema, com a representação e explicação da sua arquitetura.

De uma forma geral, a aplicação *web PowerBI ScoreCards*, mais abordada na Secção 2.3.2, pode ser utilizada por qualquer organização em qualquer área de atividade, para acompanhar o estado e evolução dos seus objetivos, permitindo que sejam definidos *scorecards*, que funcionam como uma representação geral dos indicadores de desempenho. Com a definição de um *scorecard* relativo, por exemplo, à Atividade Hospitalar, pode ser definido um conjunto de áreas (por exemplo, Consultas Externas, Urgência e Internamento), com um peso (em percentagem) que cada uma contribui para o *scorecard*. Posteriormente, pode ser definido um conjunto de indicadores de desempenho para cada área.

Para os indicadores de desempenho é estabelecido um peso (percentagem) com o qual este contribui para o valor total da sua área, existindo a possibilidade de definição para serem preenchidos de forma mensal, trimestral, semestral ou anual, definindo-se um valor a atingir (meta) para cada um. Os dados a serem guardados na base de dados da aplicação *web PowerBI ScoreCards* são dados já consolidados, destinados a serem analisados para se avaliar o estado e evolução dos indicadores de desempenho, suportando o processo de tomada de decisão.

Face a isto, estes dados podem ser obtidos por uma ferramenta de BI e representados no relatório a ser criado com a própria ferramenta, no caso atual pelo *Microsoft Power BI*, para representar todos os dados dos *scorecards* do cliente pretendido, permitindo interpretar os dados em diferentes perspetivas visuais. Desta forma, pode-se verificar, por exemplo, o valor que um indicador de desempenho obteve num determinado mês e ano e/ou analisar a área que mais tem mais contribuído/melhor registo no *scorecard*.

A Figura 2.10 é um exemplo de uma página de relatório de um *scorecard* e seus indicadores de desempenho.

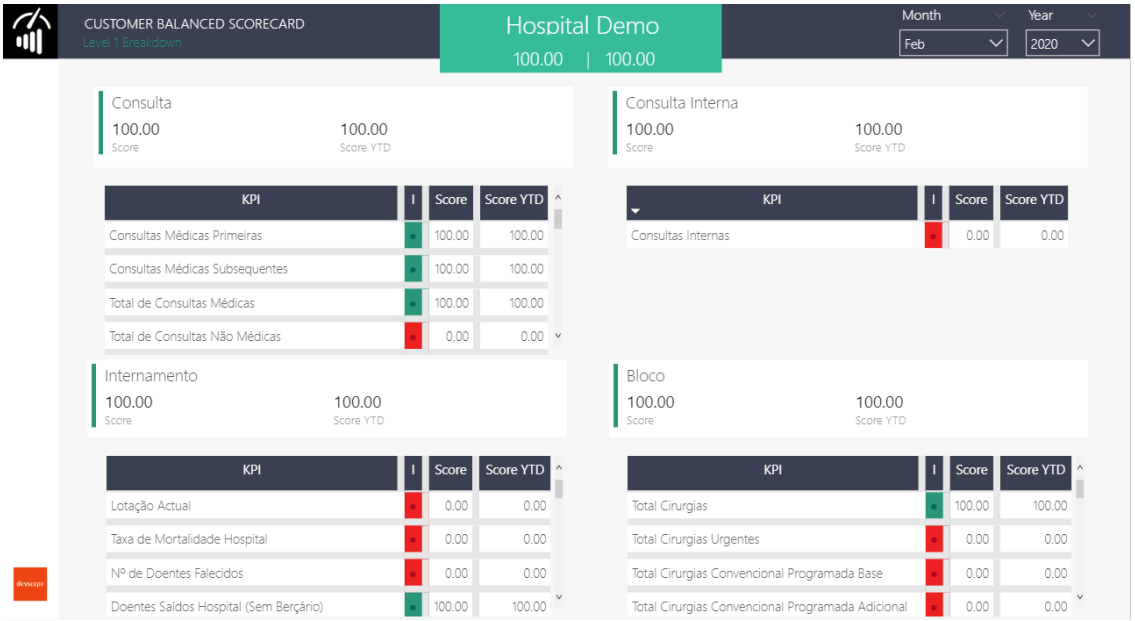


Figura 2.10: Exemplo de um Relatório Representativo de um Scorecard

2.3.1 Arquitetura

A Figura 2.11 representa a arquitetura atualmente existente no processo de obtenção de dados e criação de relatório. Para tal, é apresentado um diagrama de componentes, utilizando a linguagem Unified Modeling Language (UML) para desenvolver este artefacto.

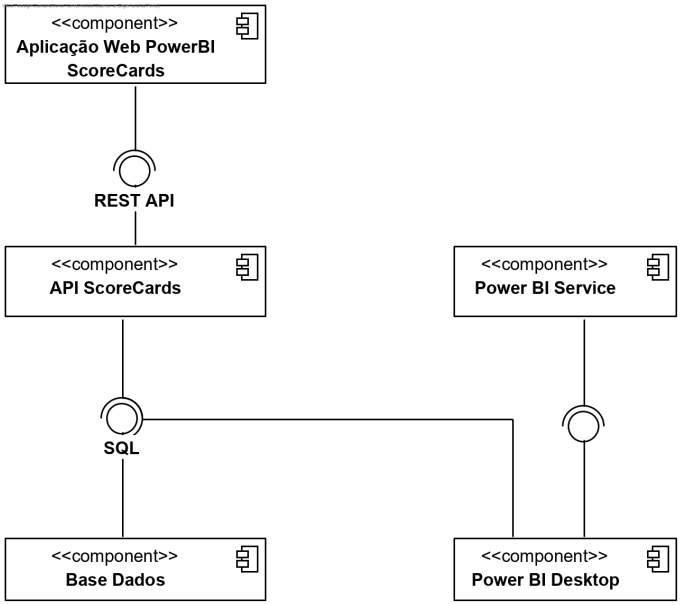


Figura 2.11: Diagrama de Componentes da Solução Atual

Pela apresentação do diagrama da Figura 2.11, identificam-se cinco componentes, sendo estes:

- Aplicação *Web PowerBI ScoreCards* – Representa a aplicação *web* onde são introduzidos dados relativos aos indicadores de desempenho para os *scorecards* definidos. Esta aplicação é abordada com mais detalhe na Secção 2.3.2.
- API *ScoreCards* – Esta API permite a comunicação entre a aplicação *web PowerBI ScoreCards* e a base de dados, permitindo que os dados introduzidos na aplicação anterior sejam persistidos na base de dados, assim como, aceder aos dados da base de dados para os mostrar na aplicação.
- Base de Dados – Armazena a informação introduzida na aplicação *web*, contendo informação sobre os *scorecards* definidos, o estado atual e meta a atingir pelos indicadores de desempenho. Para tal, estabelece uma ligação com a API apresentada anteriormente.
- *Power BI Desktop* – Aplicação *desktop* que acede à base de dados anterior, através de uma conexão SQL, para obter os dados dos clientes e criar o relatório pretendido.
- *Power BI Service* – Aplicação *online*, onde permite que os relatórios criados no *Power BI Desktop* sejam publicados e possam ser definidos *dashboards*. Esta parte é realizada pelo cliente depois de obter o relatório.

2.3.2 Aplicação Web PowerBI ScoreCards

Nesta secção, é apresentada a aplicação *web PowerBI ScoreCards* desenvolvida pela empresa *DevScope*, que assenta na metodologia *Balanced Scorecard*.

O termo *Balanced Scorecard* foi introduzido por Robert S. Kaplan e David Norton em 1992 [34], e é uma metodologia de gestão e planeamento estratégica, que pretende alinhar a visão e a estratégia da organização com as suas atividades, promovendo uma melhoria de funções internas da organização, de forma a melhorar os seus resultados externos. O modelo *Balanced Scorecard* assenta em quatro perspetivas [35]:

- Financeira – Fornece uma linguagem comum para analisar e comparar organizações.
- Cliente – Identifica os clientes e os segmentos de mercado no qual o negócio se irá focar.
- Processos Internos – Identifica os processos chave internos e seus objetivos de forma a alcançar os objetivos do cliente.
- Aprendizagem e Crescimento – Foca-se na capacidade das pessoas, onde se deve promover o seu desenvolvimento.

Esta aplicação é uma aplicação de gestão, onde permite orçamentar, definir objetivos individuais e/ou de equipa aos seus clientes, permitindo o acompanhamento sobre o seu estado e evolução. Com a definição de um *scorecard* na aplicação, é possível definir um conjunto de indicadores de desempenho. Os indicadores de desempenho são uma medida de valor que demonstram como uma organização está a alcançar os seus objetivos principais de negócio, ou seja, se o progresso dos mesmos está em direção ao resultado/meta pretendido [36].

Para além do que foi referido, existe a possibilidade de definir indicadores de desempenho com fórmulas e unidade de medida à escolha dos clientes, cobrindo todos os indicadores de produtividade que estes considerem importantes. Estes indicadores de desempenho podem estar definidos relativamente a colaboradores individuais, bem como, a múltiplos colaboradores ou múltiplas equipas, permitindo que estas trabalhem para objetivos comuns.

2.4 Visão da Solução

Em virtude do que foi apresentado nas secções anteriores, a aplicação *web PowerBI ScoreCards* permite aos seus clientes implementar estratégias e tomar decisões de negócio através de um sistema centralizado, que monitoriza e exibe a produtividade de uma forma fácil de leitura.

Posto isto, face aos dados (indicadores de desempenho) que um cliente insere na aplicação *web PowerBI ScoreCards* e/ou obtidos automaticamente das suas bases de dados, através da utilização de uma ferramenta de BI, *Microsoft Power BI*, são elaborados relatórios que contêm vários visuais representativos da informação. Contudo, este processo de obter os dados e fornecer o relatório ao cliente é realizado pela equipa de BI, em que, na ferramenta de BI, acedem à base de dados da aplicação, efetuando consultas (*queries*) diretas à mesma para obter os dados pretendidos e, após isto, o relatório é criado e entregue ao cliente. A ferramenta acede a uma única base de dados que contém os dados dos clientes.

Com a descrição do fluxo apresentado anteriormente, este vai repetir-se para todos os clientes a quem se irá entregar o relatório desenvolvido com a ferramenta de BI. Assim sendo, surge a necessidade de automatizar este processo, pretendendo a integração automática dos dados da aplicação *web PowerBI ScoreCards* com uma ferramenta de BI e a criação de um relatório genérico, a ser usado por qualquer cliente. Desta forma, para cada cliente, será entregue um relatório previamente preparado, que acede de forma automática à sua informação e mostra o seu conteúdo visualmente.

Face à arquitetura apresentada na Secção 2.1.1 referente a um sistema de BI, o processo atual e o processo a ser desenvolvido não pressupõe a implementação deste tipo de sistema. Isto pode ser explicado porque os dados da aplicação *web PowerBI ScoreCards*, ao qual se pretendem integrar automaticamente com a ferramenta de BI, são dados já consolidados (como se fossem provenientes de um DW), com o intuito de serem representados e analisados para se tomarem decisões de negócio, podendo ser apresentados em gráficos ou relatórios (aplicações *front-end*). Assim, com base na arquitetura de um sistema de BI e ao que é pretendido nesta dissertação, o processo a abordar envolve a parte final da arquitetura, ou seja, a integração e representação de dados consolidados da aplicação *web PowerBI ScoreCards* com uma ferramenta de BI.

Ainda sobre o processo atual, que utiliza a ferramenta *Microsoft Power BI*, depois de se efetuar um estudo das ferramentas existentes e a sua análise comparativa, não se prevê uma mudança de ferramenta de BI na implementação do novo processo.

Posto isto, com a criação de um relatório genérico, a solução a ser implementada (desenvolvimento de um mecanismo de integração de dados) permitirá que este tenha a capacidade de obter a informação de forma automática do cliente respetivo. Deste modo, não é necessário criar, repetidamente, um relatório para cada cliente, já que será entregue um relatório previamente preparado que efetue o processo descrito anteriormente. O cliente receberá o relatório e vai consultar o mesmo na ferramenta de BI adequada para este efeito (*Microsoft Power BI*), sendo um fluxo agilizado e automatizado.

Capítulo 3

Análise de Valor

Este capítulo tem como finalidade representar a análise de valor, recorrendo a diferentes técnicas e métodos para avaliar o processo de criação de produtos/serviços.

No âmbito desta dissertação, a análise de valor é realizada pelo modelo *New Concept Development*, proposta de valor, benefícios e sacrifícios, modelo *Canvas* e, por último, pelo método de análise hierárquica.

3.1 New Concept Development

O processo de inovação pode ser divido em três partes: o *Fuzzy Front End* (FFE), o *New Product Development* (NPD) e a Comercialização.

A primeira parte, o FFE, é considerado como uma das principais oportunidades de melhoria do processo de inovação, correspondendo à fase inicial do processo de desenvolvimento, em que as escolhas e ideias tomadas vão determinar quais as opções de inovação que vão ser consideradas nas fases de desenvolvimento e comercialização.

Contudo, devido à inexistência de uma linguagem ou definição comum para os elementos chaves do FFE, surge o modelo *New Concept Development* (NCD), desenvolvido para fornecer uma linguagem comum.

Segundo Peter A. Koen [37], o modelo NCD consiste em três componentes, tal como se verifica na Figura 3.1:

- A área interna define os cinco elementos principais que constituem o *Front End of Innovation* (FEI).
- O motor que impulsiona os cinco elementos do *front-end* é alimentado pela liderança e cultura da organização.
- Os fatores de influência consistem em: Capacidades Organizacionais, Estratégia de Negócio, Mundo Exterior (por exemplo, canais de distribuição, clientes e concorrentes) e a Ciência Influenciadora que será utilizada.

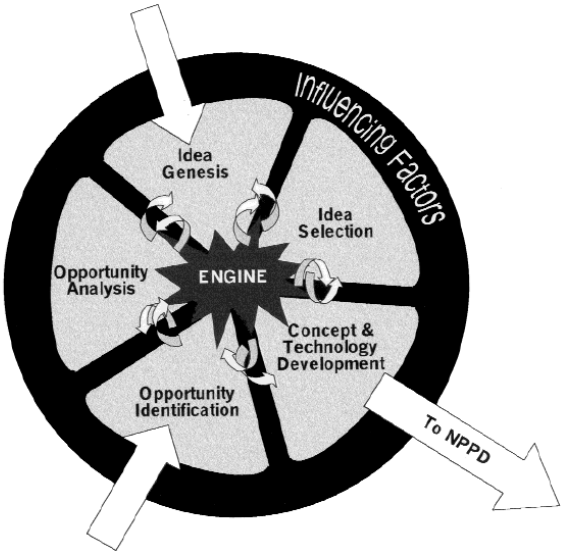


Figura 3.1: Modelo de New Concept Development [37]

Seguindo o modelo apresentado na Figura 3.1, são analisados os cinco elementos do FEI, que são apresentados nas seguintes secções.

3.1.1 Identificação da Oportunidade

Nesta fase, é onde uma organização identifica as oportunidades que pretende realizar, orientadas aos objetivos do negócio [37].

A oportunidade surge com a necessidade de automatizar o processo de obtenção de dados da aplicação *web PowerBI ScoreCards* (Secção 2.3.2) com uma ferramenta de BI e, por conseguinte, a criação de um relatório que ofereça representação dos dados, sendo apresentados de forma simples, detalhados e intuitivos, suportados pela visualização gráfica. Este problema verificou-se devido ao processo descrito no problema (Secção 1.2) ser repetitivo e moroso.

3.1.2 Análise da Oportunidade

De modo a traduzir uma identificação da oportunidade numa concreta oportunidade de negócio, é necessário realizar alguns estudos de mercado ou experiências científicas [37].

Atualmente, a aplicação *web PowerBI ScoreCards* contabiliza um total de dez clientes nacionais nas áreas hospitalares, nas empresas metalúrgicas e municipais que monitorizam as suas tarefas e tomam as suas decisões baseadas na análise dos indicadores de desempenho.

Conforme o supratranscrito, verifica-se que a oportunidade identificada é relevante para empresa, uma vez que permite um aumento de valor à aplicação, uma diminuição de tempo do processo a ser desenvolvido e facilidade de manutenção do mesmo. Além disto, é também vantajoso para os clientes finais, que irão usufruir deste novo processo desenvolvido, uma vez que permite ter os seus relatórios dentro dos prazos pretendidos, ou seja, quando adquirem a aplicação *web PowerBI ScoreCards*.

3.1.3 Génese da Ideia

Esta parte corresponde ao desenvolvimento e maturação de uma oportunidade numa ideia concreta, sendo um processo evolutivo, já que a ideia é estudada, modificada e atualizada ao longo de várias iterações. A génese da ideia pode ser um processo que inclui sessões de *brainstorming* como forma de criar novas ideias ou modificar ideias existentes para a oportunidade identificada [37].

Com a identificação e análise da oportunidade, é efetuado um *brainstorming* de ideias e pesquisa de ferramentas e tecnologias que possam vir a ser usadas, como forma de encontrar uma solução para o cenário identificado, verificando quais as funcionalidades que cada uma oferece.

Face ao que foi debatido e pesquisado, as ideias apresentadas são as seguintes:

- Implementação de um sistema automatizado de BI (uso de *Data Warehouse*, sistemas OLAP), desde a obtenção da informação da fonte de dados até à disponibilização da mesma ao cliente em relatórios.
- Desenvolver um mecanismo de integração automático dos dados da aplicação *web PowerBI ScoreCards* com uma ferramenta de BI, de forma a agilizar o processo de criação de relatórios.

3.1.4 Seleção da Ideia

Esta fase corresponde à escolha da ideia que garante atingir o maior valor de negócio. Caracteriza-se por ser um processo complicado quando existe um número elevado de ideias, pois torna-se difícil o processo de seleção [37].

Face às ideias apresentadas na Secção 3.1.3, a primeira ideia apresentada levaria a um maior tempo de desenvolvimento e a uma maior complexidade para o processo que se pretende, já que não garantia que as necessidades fossem totalmente atingidas, o que levou a que a segunda opção fosse a ideia escolhida. Desta forma, uma ferramenta de BI, com um mecanismo que obtenha de forma automática os dados da aplicação *web PowerBI ScoreCards* e, posteriormente, a criação de relatórios para representar a informação, revela-se como mais eficiente e mais indicada. Assim, esta ideia selecionada é capaz de dar resposta ao que se pretende, facilitando a manutenção e automatização do processo.

3.1.5 Desenvolvimento do Conceito e Tecnologia

Este elemento final do modelo é relativo ao desenvolvimento de um caso de negócio baseado em estimativas de potencial de mercado, necessidades do cliente, requisitos investidos, avaliação da concorrência, tecnologia desconhecida e risco geral do projeto [37].

Em virtude da oportunidade apresentada, considera-se o desenvolvimento de um mecanismo que integra de forma automática os dados do cliente da aplicação *web PowerBI ScoreCards* com a ferramenta de BI, em conjunto com a utilização de um relatório genérico. Este relatório irá conter a informação do cliente exibida por via de visuais. Desta forma, automatizará o processo existente, pois será criado um relatório que será entregue a todos os clientes,

ao invés do processo atual, em que se cria um relatório para cada cliente, sendo o mesmo caracterizado por ser moroso e repetitivo.

Com isto, através de reuniões com o cliente, identificaram-se as necessidades que visam ser atingidas, estabelecendo os requisitos pretendidos, assim como, o tempo e esforço necessário para o desenvolvimento.

3.2 Valor, Valor para o Cliente, Valor Percecionado

De forma a que um produto/serviço apresente valor, torna-se importante abordar conceitos relacionados com o valor para o cliente e valor percecionado, identificando quais os benefícios e sacrifícios que o cliente irá ter.

O valor para o cliente é o valor que se atribui a um produto/serviço, podendo estar sujeito a diferentes interpretações por diferentes clientes. No âmbito deste projeto, o valor para o cliente é respetivo à automatização de um processo que, atualmente, é repetitivo. Por isso, perspetiva-se que, com a implementação deste projeto, a equipa de desenvolvimento sinta uma maior satisfação no processo, reduzindo o tempo despendido no mesmo, aumentando a sua produtividade.

O valor percecionado consiste na diferença entre os benefícios e sacrifícios que o cliente atribui [38]. No caso deste projeto, os benefícios deste projeto são relativos a uma solução automatizada que integrará os dados da aplicação *web PowerBI ScoreCards* com uma ferramenta de BI, agilizando o processo em termos de criação de relatórios, sendo mais fácil a sua implementação e manutenção. Em termos de sacrifícios, existe um tempo na pesquisa de alternativas relacionadas com o problema e implementação do novo processo, implicando, também, que a equipa de desenvolvimento (cliente) dispense algum tempo a perceber o novo paradigma que se vai desenvolver. Na Secção 3.3, são apresentados os benefícios e sacrifícios numa perspetiva longitudinal de valor.

3.3 Benefícios e Sacrifícios

Segundo Woodall [39], o valor para o cliente pode ser introduzido sob uma perspetiva longitudinal, existindo quatro fases temporais em que o valor para o cliente é percecionado, sendo estas as seguintes:

- Pré-compra
- Ponto de negócio ou experiência
- Pós-compra
- Após uso ou experiência

Posto isto, na Tabela 3.1, são expostos os benefícios e sacrifícios para o cliente para cada uma das quatro fases indicadas anteriormente, referentes ao processo a implementar.

Tabela 3.1: Benefícios e Sacrifícios numa Perspetiva Longitudinal

	Benefícios	Sacrifícios
Pré-compra	Solução que se espera que ofereça mais rapidez de processo, fácil manutenção e integração automática de dados com ferramenta de BI	Pesquisa sobre as vantagens da ferramenta, quais as funcionalidades que a mesma oferece e como o processo será implementado
Ponto de negócio ou experiência	Adaptação à solução de forma a perceber as suas qualidades técnicas e funcionais	Aprendizagem do novo paradigma que implica esforço e tempo
Pós-compra	Domínio completo da solução e suas funcionalidades	
Após uso ou experiência	Solução apresentada promove um aumento de qualidade, rapidez, eficácia e robustez no processo de integração automática de dados com ferramenta de BI	

3.4 Proposta de Valor

A proposta de valor assume um papel importante na criação de um produto e/ou serviço, na medida em que são apresentados os motivos pelo qual um cliente deve comprar um produto e/ou serviço, assim como, os benefícios que este traz ao cliente e as dificuldades que são suprimidas.

Como forma das ideias serem organizadas e entendidas na proposta de valor, pode-se fazer uso do *Value Proposition Canvas*, disponibilizado pela *Strategyzer* [40], tal como ilustrado na Figura 3.2.

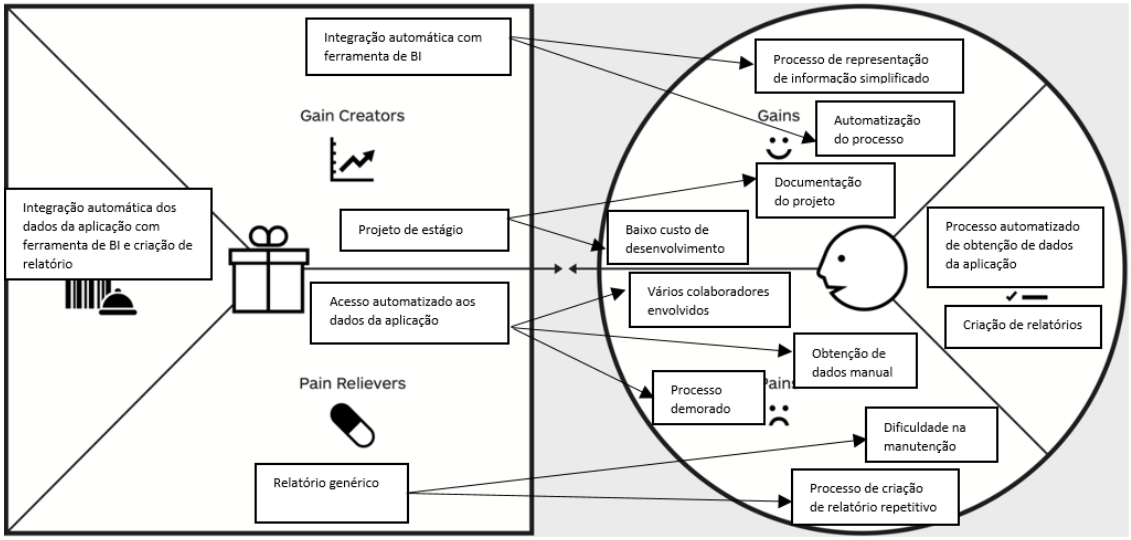


Figura 3.2: Value Proposition Canvas do Projeto

Assim sendo, a proposta de valor no âmbito deste projeto refere-se ao desenvolvimento de um mecanismo de integração automático dos dados da aplicação *web PowerBI ScoreCards* com uma ferramenta de BI, permitindo criar um relatório genérico para representar a informação.

Esta solução irá trazer efeitos benéficos para a empresa, pois automatizará o processo existente, diminuindo assim os recursos humanos envolvidos, o tempo de implementação e a manutenção. Por sua vez, o cliente beneficiará de um produto total (aplicação *web PowerBI ScoreCards* + relatório construído com ferramenta BI para visualização de dados) mais robusto, que integra os dados que estão presentes na aplicação *web PowerBI ScoreCards* com uma ferramenta de BI, onde se cria relatórios para a sua informação, sendo este fluxo automatizado e agilizado.

3.5 Modelo de Canvas

O Modelo de Canvas, *Business Model Canvas*, proposto por Alexander Osterwalder na sua tese de Doutoramento no ano de 2004 [41] e apresentado no livro *Business Model Generation: A Handbook for Visionaries, Game Changers, and Challengers* em 2010 [42], é uma ferramenta que permite entender um modelo de negócio de uma forma direta e estruturada. Segundo Alexander Osterwalder [42], este descreve o modelo de negócio como "*A business model describes the rationale of how an organization creates, delivers, and captures value*".

No contexto deste projeto, é apresentado o *Business Model Canvas*, representado na Figura 3.3, composto por nove bloco diferentes.

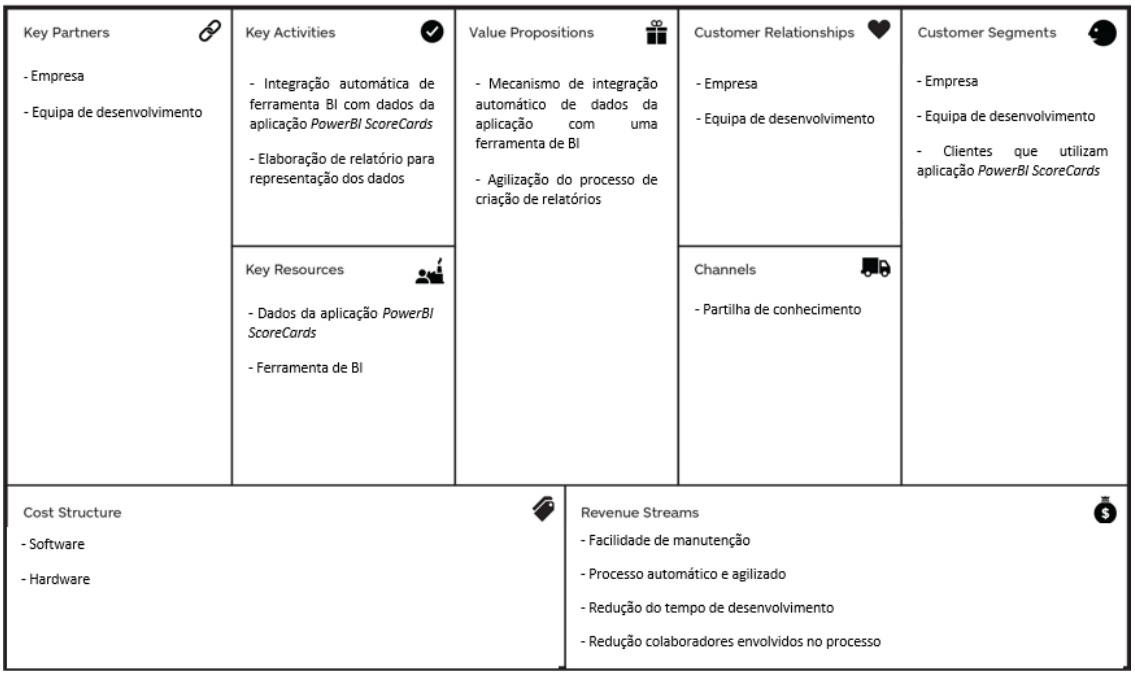


Figura 3.3: Business Model Canvas

Seguidamente, apresenta-se uma explicação para cada bloco do modelo de canvas, representado na Figura 3.3.

Parceiros Chave (Key Partners)

Este bloco descreve a rede de fornecedores e parceiros que fazem com o que o modelo de negócio funcione, com o intuito do otimizar, reduzindo riscos ou adquirindo recursos [42].

No caso deste projeto, os parceiros passam por ser a empresa, pois é o local onde irá ser realizado o processo, assim como, a equipa de desenvolvimento, que irá dar suporte no processo de desenvolvimento do projeto e validar o mesmo.

Atividades Chave (Key Activities)

Este bloco, respeitante às atividades chave, é relativo às tarefas que uma organização deve fazer para que o seu modelo de negócio funcione. Estas atividades são necessárias para criar a proposta de valor, alcançar mercados e manter clientes [42].

No âmbito deste projeto, as atividades chave passam por realizar a integração automática dos dados da aplicação *web Power BI ScoreCards* com a ferramenta de BI, permitindo criar um relatório para representar os dados. Estas duas atividades são essenciais para o projeto atingir toda a sua plenitude.

Recursos Chave (Key Resources)

Este bloco é relativo aos ativos fundamentais para que o modelo de negócio funcione, podendo os recursos chave serem físicos, financeiros, intelectuais ou humanos [42].

No contexto deste projeto, os dois recursos chave são os dados da aplicação *web Power BI ScoreCards*, bem como, a ferramenta de BI, onde se pretende que exista uma integração automática entre ambos.

Proposta de Valor (Value Propositions)

Este bloco descreve o conjunto de produtos ou serviços que criam valor para um específico segmento de clientes. Apresenta-se como a solução para resolver um problema do cliente ou por ir ao encontro das suas necessidades [42].

Neste caso, a proposta de valor é relativa ao desenvolvimento de um mecanismo de integração automático dos dados da aplicação *web PowerBI ScoreCards* com uma ferramenta de BI, permitindo agilizar o processo de criação de um relatório genérico para representar a informação de cada cliente.

Relação com os clientes (Customer Relationships)

Este bloco descreve o tipo de relação que uma organização estabelece com os seus clientes específicos [42].

No âmbito deste projeto, a empresa é o cliente principal, pretendendo que a solução a desenvolver responda às suas necessidades, e a equipa de desenvolvimento, pois esta validará se o processo a ser implementado é melhor que o atual.

Canais (Channels)

Este bloco descreve a forma como uma organização comunica com os seus clientes para entregar a sua proposta de valor [42].

Neste caso, a partilha de conhecimento através de realização de formações é um meio para mostrar como o processo a ser desenvolvido foi implementado.

Segmentos de Clientes (Customer Segments)

Este bloco define os diferentes grupos de pessoas ou organizações que uma empresa pretende alcançar ou servir [42].

No âmbito deste projeto, a empresa é um dos segmentos dos clientes, já que existirá uma adição de valor a uma das suas aplicações, a equipa de desenvolvimento, pois contará com um processo automatizado, contrastando com o processo que existe atualmente e, por fim, os clientes que utilizam a aplicação *web PowerBI ScoreCards*, pois terão a possibilidade de ter os seus dados integrados automaticamente com uma ferramenta de BI, que suporta a criação de relatórios para representar a sua informação para posterior análise.

Estrutura de Custos (Cost Structure)

Este bloco descreve todos os custos envolvidos para que o modelo de negócio seja realizado [42].

No contexto deste projeto, a estrutura de custos passa pelo *hardware* e *software* a ser utilizado para desenvolvimento da solução.

Fontes de Receita (Revenue Streams)

Este bloco representa os lucros que a organização gera a partir de cada segmento de clientes [42].

Neste âmbito, o desenvolvimento deste processo permitirá automatizar e agilizar o processo existente na criação de relatórios, diminuindo a redução de colaboradores necessários e tempo de desenvolvimento para a realização do mesmo, permitindo, também, uma maior facilidade na sua manutenção.

3.6 Método de Análise Hierárquica para Comparação de Tecnologias

O Analytic Hierarchy Process (AHP) criado por Tomas L. Saaty [43] em 1980 é um dos principais métodos de apoio à decisão multicritério. Este método foi aplicado conforme as sete fases definidas pelo mesmo.

O objetivo de aplicar o AHP nesta dissertação é para decidir qual a ferramenta de BI a utilizar, de modo a que os dados da aplicação *web PowerBI ScoreCards* sejam automaticamente integrados com a mesma. As quatro ferramentas a considerar foram descritas anteriormente na Secção 2.2.

A primeira fase, correspondente à construção da árvore hierárquica de decisão, consiste na definição do objetivo de decisão, nos critérios associados e as alternativas disponíveis. A Figura 3.4 ilustra a estrutura hierárquica do AHP, tendo como finalidade de escolher qual a ferramenta de BI a utilizar.

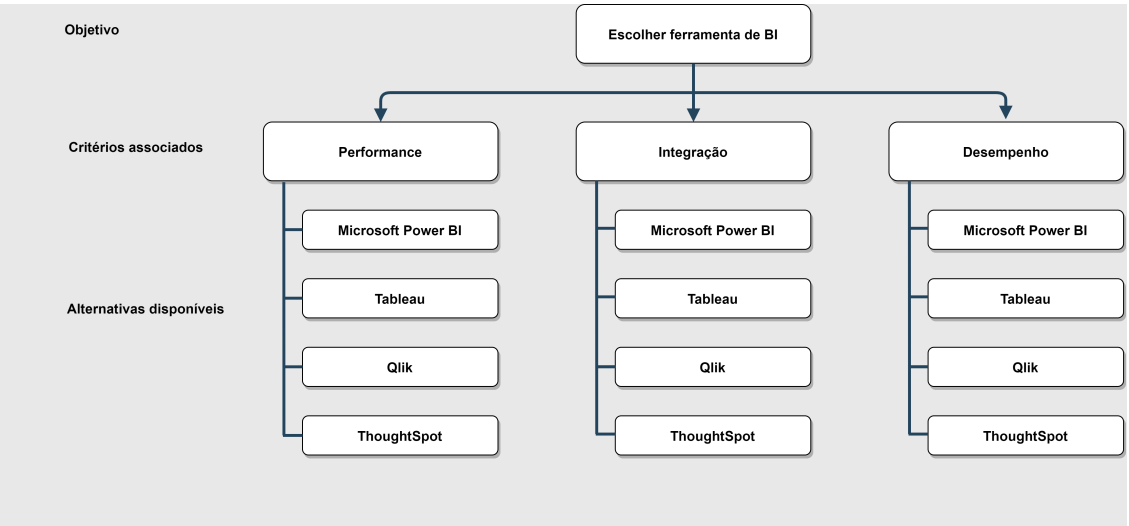


Figura 3.4: Estrutura Hierárquica de Decisão

Como forma de alcançar o objetivo definido, estabeleceram-se os três critérios seguintes:

- Performance – Desempenho na execução de tarefas
- Integração – Integração com várias fontes de dados
- Usabilidade – Facilidade de utilização da ferramenta

A segunda fase do método AHP é relativa à comparação entre os elementos da hierarquia, que se caracteriza por definir as prioridades para os critérios e alternativas apresentadas anteriormente, utilizando a escala de Saaty [44], atribuindo valores de 1 a 9.

Assim sendo, a Tabela 3.2 apresenta a matriz de comparação entre os critérios.

Tabela 3.2: Matriz de Comparação entre Critérios

	Performance	Integração	Usabilidade
Performance	1	3	5
Integração	1/3	1	4
Usabilidade	1/5	1/4	1

Pela análise da Tabela 3.2, constata-se que o critério *Performance* tem uma forte importância quando comparado com o critério Usabilidade e é levemente favorecido em relação ao critério Integração. Relativamente à comparação entre os critérios Integração e Usabilidade, verifica-se que a Integração tem maior importância que a Usabilidade.

Na fase três do método AHP, obteve-se a matriz normalizada dos critérios de segundo nível (tem como finalidade igualar todos os critérios à mesma unidade, onde cada valor da Tabela 3.2 é dividido pelo total da sua respetiva coluna), com a prioridade relativa de cada critério, tal como apresentado na Tabela 3.3.

Tabela 3.3: Matriz Normalizada dos Critérios com Cálculo da Prioridade Relativa

	Performance	Integração	Usabilidade	Prioridade Relativa
Performance	15/23	12/17	1/2	0.62
Integração	5/23	4/17	4/10	0.29
Usabilidade	3/23	1/17	1/10	0.10

Pela análise da Tabela 3.3, a prioridade relativa, calculada através da média dos valores de cada linha da matriz normalizada, indica a ordem de importância de cada critério, o que traduz que a Performance é o critério mais importante, seguido da Integração e, por último, a Usabilidade.

De seguida, a fase quatro pretende avaliar a consistência das prioridades relativas, sendo necessário calcular a Razão de Consistência (RC). Se a RC for inferior a 10%, pode-se concluir que os valores das prioridades relativas são consistentes.

Assim sendo, considera-se que:

$$[Ax = \lambda_{max} x]$$

onde x é o vetor próprio, obtido pela prioridades relativas da Tabela 3.3.

Assim sendo, obtém-se a seguinte matriz.

$$\lambda_{max} \cong \begin{bmatrix} 1.99 \\ 0.90 \\ 0.30 \end{bmatrix}$$

O λ_{max}, valor próprio, é calculado pela média dos valores obtidos da matriz anterior dividido pelos valores da matriz normalizada com o valor da prioridade relativa de cada critério (Tabela 3.3).

$$\lambda_{max} = \frac{1.99/0.62 + 0.90/0.29 + 0.30/0.10}{3} = 3.10$$

Para se calcular o Índice de Consistência (IC), sendo n igual ao número de critérios, aplica-se a seguinte fórmula:

$$IC = \frac{(\lambda_{max} - n)}{(n-1)} = \frac{(3.10 - 3)}{(3-1)} = 0.05$$

Uma vez calculado o IC, para se calcular a RC, é necessário saber o valor do Índice Aleatório (IR). Para tal, o valor de IR é obtido através de uma tabela própria (Tabela 3.4), face ao valor de n.

Tabela 3.4: Tabela de Valores de IR

n	1	2	3	4	5	6	7	8	9	10
IR	0.00	0.00	0.58	0.90	1.12	1.24	1.32	1.41	1.45	1.49

Sendo que o valor IR é 0.58, devido ao número de critérios ser 3 (valor de n), a RC é calculada da seguinte forma:

$$RC = \frac{IC}{IR} = \frac{0.05}{0.58} = 0.0862$$

Como 0.0862 < 0.1, considera-se que os valores das prioridades relativas estão consistentes.

Na fase cinco, é necessário construir a matriz de comparação paritária para cada critério, tendo em consideração cada uma das alternativas identificadas. Tal como realizado anteriormente na fase três, é obtida a prioridade relativa de cada critério para as alternativas identificadas, tal como se verifica nas Tabelas 3.5, 3.6 e 3.7.

Tabela 3.5: Matriz de Comparação para o Critério Performance

	Power BI	Tableau	Qlik	ThoughtSpot	Prioridade Relativa
Power BI	1	2	3	3	0.44
Tableau	1/2	1	3	3	0.31
Qlik	1/3	1/3	1	2	0.15
ThoughtSpot	1/3	1/3	1/2	1	0.11

Tabela 3.6: Matriz de Comparação para o Critério Integração

	Power BI	Tableau	Qlik	ThoughtSpot	Prioridade Relativa
Power BI	1	3	5	3	0.82
Tableau	1/3	1	4	2	0.35
Qlik	1/5	1/4	1	1/4	0.09
ThoughtSpot	1/3	1/2	4	1	0.24

Tabela 3.7: Matriz de Comparação para o Critério Usabilidade

	Power BI	Tableau	Qlik	ThoughtSpot	Prioridade Relativa
Power BI	1	2	3	2	0.43
Tableau	1/2	1	3	2	0.30
Qlik	1/3	1/3	1	1/3	0.10
ThoughtSpot	1/2	1/2	3	1	0.21

Face à informação apresentada nas Tabelas 3.5, 3.6 e 3.7, pode-se verificar que a ferramenta *Microsoft Power BI* destaca-se nos três critérios identificados, seguida da ferramenta *Tableau, ThoughtSpot* e, por fim, *Qlik*.

Seguidamente, na fase seis, obtém-se as prioridades compostas das alternativas, através da multiplicação da matriz prioridade, construída através dos valores das Tabelas 3.5, 3.6, 3.7, com os valores das prioridades relativas de cada critério (matriz com peso de cada critério - Tabela 3.3).

$$\begin{bmatrix} 0.44 & 0.82 & 0.43 \\ 0.31 & 0.35 & 0.30 \\ 0.15 & 0.09 & 0.10 \\ 0.11 & 0.24 & 0.21 \end{bmatrix} * \begin{bmatrix} 0.62 \\ 0.29 \\ 0.10 \end{bmatrix} = \begin{bmatrix} 0.55 \\ 0.32 \\ 0.13 \\ 0.16 \end{bmatrix}$$

Posto isto, na última fase, escolhe-se a alternativa com a prioridade mais alta relativa à ferramenta de BI, em função dos critérios definidos e das suas respetivas importâncias. A Tabela 3.8, apresenta as prioridades relativas de cada critério e a prioridade composta de cada alternativa.

Tabela 3.8: Matriz Final do AHP para Escolher Ferramenta BI

Ferramenta	Performance	Integração	Usabilidade	Prioridade Composta
Power BI	0.44	0.82	0.43	**0.55**
Tableau	0.31	0.35	0.30	0.32
Qlik	0.15	0.09	0.10	0.13
ThoughtSpot	0.11	0.24	0.21	0.16

Com a análise da Tabela 3.8, verifica-se que a alternativa com a prioridade composta mais alta é a ferramenta *Microsoft Power BI*, o que indica que é a melhor para o problema, pois é a que apresenta melhores valores nos três critérios definidos, sendo esta a ferramenta escolhida a utilizar na solução.

Capítulo 4

Análise e Design

Neste capítulo, é apresentada a solução atual, fazendo uma contextualização de como o processo é realizado, seguido da definição dos requisitos do novo processo a desenvolver. Adicionalmente, são expostas e analisadas diferentes soluções para o novo processo a implementar, culminando com a escolha da solução que melhor se enquadra para o problema. Por fim, apresenta-se a proposta de implantação para a solução escolhida.

Para retratar todos os artefactos apresentados, serão utilizados diagramas UML.

4.1 Solução Atual

O processo de desenvolvimento da solução atual pode ser representado pelo diagrama de atividades presente na Figura 4.1.

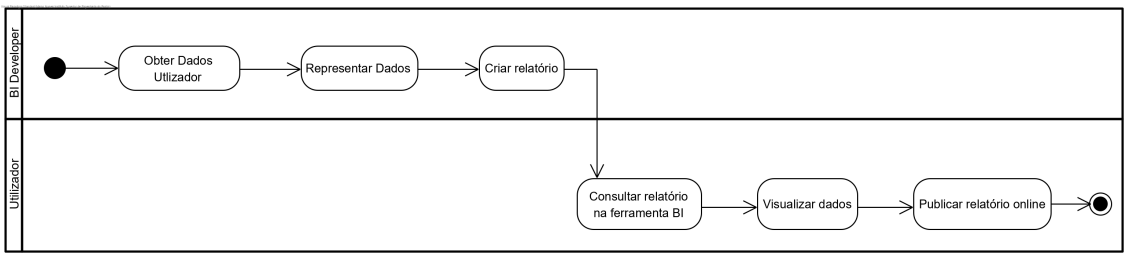


Figura 4.1: Processo Atual

Tal como apresentado no diagrama da Figura 4.1, identificam-se dois intervenientes: *BI Developer* e Utilizador (cliente final).

Para a realização deste processo, o *BI Developer* começa por obter os dados do utilizador pretendido na ferramenta de BI (dados adquiridos através de *queries* à base de dados), representando-os por via de vários visuais, o que dá origem à criação de um relatório. Seguidamente, o relatório é entregue ao utilizador (cliente) que, por sua vez, consulta o relatório para a visualização de dados, tendo a possibilidade de publicar o relatório *online*, para serem criados *dashboards* e/ou ser partilhado com outros utilizadores. Este processo, efetuado pelo *BI Developer*, repetir-se-á para cada entrega do relatório ao cliente, caracterizado-se, assim, por um processo repetitivo e moroso.

A ferramenta de BI que permite a realização das tarefas por parte dos dois intervenientes nesta solução é denominada por *Microsoft Power BI* (explicada na Secção 2.2.1).

4.2 Requisitos

A definição de requisitos é um processo importante na forma como um sistema irá ser desenhado e, posteriormente, implementado. Para tal, foram realizadas reuniões com o cliente, de modo a proceder ao correto levantamento de requisitos para que o sistema final cumpra as necessidades pretendidas.

Os requisitos dividem-se em dois tipos: Requisitos Funcionais e Requisitos Não-Funcionais. Os Requisitos Funcionais consistem nas funcionalidades que o sistema terá que desempenhar [45], enquanto que, os Requisitos Não-Funcionais, correspondem às características do sistema que não são funcionalidades (linguagem de programação, usabilidade, entre outras) que terão de ser cumpridas [46].

4.2.1 Requisitos Funcionais

De forma a representar os requisitos funcionais, é ilustrado na Figura 4.2 um diagrama de casos de uso.

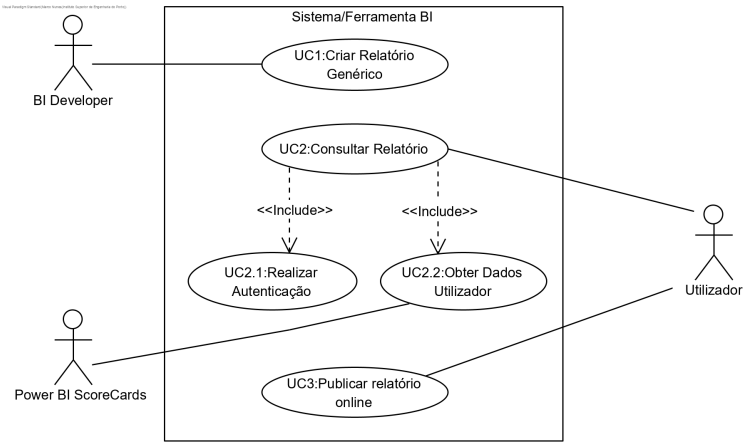


Figura 4.2: Casos de Uso do Projeto

Pela análise do diagrama da Figura 4.2, identificam-se os seguintes casos de uso:

UC1: Criar Relatório Genérico

O *BI Developer* cria um relatório genérico para representar a informação do utilizador. Este relatório deve ser genérico, sendo criado com um padrão previamente estabelecido (*template*), para ser entregue a todos os utilizadores (clientes).

UC2: Consultar Relatório

O Utilizador consulta o relatório que foi criado para visualizar a sua informação e proceder a uma análise sobre os dados. O relatório vai ser consultado e visualizado na ferramenta de BI, que terá a capacidade de realizar a autenticação com as credenciais do Utilizador (cf. UC2.1) e obter os seus dados (cf. UC2.2) que serão apresentados no relatório.

UC2.1: Realizar Autenticação

A autenticação a ser realizada permite validar as credenciais do Utilizador, de modo a garantir a sua identidade no acesso à informação.

UC2.2: Obter Dados Utilizador

O *PowerBI ScoreCards* fornece os dados do respetivo Utilizador, depois deste realizar a autenticação (cf. UC2.1), de forma a que a informação seja apresentada no relatório aquando a sua consulta (cf. UC2).

UC3: Publicar Relatório Online

O Utilizador publica o relatório na aplicação *online* da ferramenta de BI, permitindo partilhar o relatório com outros utilizadores e/ou serem criados *dashboards*.

4.2.2 Requisitos Não Funcionais

Para o âmbito deste projeto, foi definido o seguinte conjunto de requisitos não funcionais, seguindo a classificação FURPS+ [47], significando *Functionality* (Funcionalidade), *Usability*, *Reliability* (Fiabilidade/Confiabilidade), *Performance* (Desempenho) e *Supportability* (Suportabilidade), sendo que o "+" é relativo a identificar requisitos adicionais que representam restrições.

No caso deste projeto, são apresentado alguns dos atributos que esta classificação preconiza, que são os seguintes:

Funcionalidade

A ferramenta de BI deve comunicar com uma API para obter os dados da aplicação *PowerBI ScoreCards* relativos ao utilizador.

A segurança no acesso à informação por parte da ferramenta de BI à API deve ser assegurada, sendo que os processos de autenticação e autorização devem utilizar um provedor de identidade da *Microsoft*, denominado por *Azure AD*, com o protocolo de autorização *OAuth 2.0*.

Fiabilidade/Confiabilidade

Neste projeto, a confiabilidade não foi estipulada, uma vez que a API relativa a obter os dados da aplicação *PowerBI ScoreCards* dos utilizadores já está implementada e a respetiva segurança no seu acesso está ao encargo de um provedor de identidade externo.

Suportabilidade

A solução a desenvolver deve ter atenção à manutenibilidade, de modo a que o processo de alterar o *software* implementado seja facilmente corrigido ou estendido a novas funcionalidades ao longo do seu ciclo de vida.

A ferramenta de BI, que permite a criação do relatório genérico, deve permitir também a extensibilidade de visuais gráficos, com a criação/adição de novos visuais personalizados para representar a informação.

4.3 Modelo de Domínio

O modelo de domínio consiste na representação das entidades e conceitos do problema e a forma como se relacionam entre si. O diagrama da Figura 4.3 ilustra as relações entre as diferentes entidades do problema.

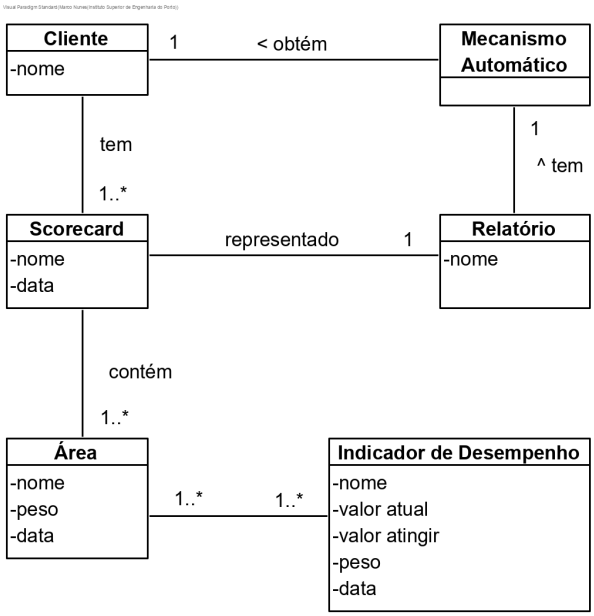


Figura 4.3: Modelo de Domínio

Seguidamente, é apresentado uma explicação para cada entidade:

- Cliente – Entidade que contém um conjunto de *scorecards*.
- *Scorecard* – Representa o *scorecard* do cliente (visão geral) que contém um conjunto de áreas.
- Área – Representa uma área de um *scorecard*, em que se atribui um peso em percentagem que esta representa no *scorecard* do cliente. É constituída por um conjunto de indicadores de desempenho.
- Indicador de Desempenho – Representa a medida de valor para verificar o valor atual face à meta que se pretende atingir.
- Relatório – Representação dos dados do *scorecard* e seus indicadores de desempenho em visuais gráficos para posterior análise.
- Mecanismo Automático – Mecanismo responsável por obter de forma automática os dados do cliente para serem apresentados no relatório.

4.4 Diagrama do Processo Proposto

Uma vez analisado o problema e o processo atual, representado na Figura 4.1, o diagrama ilustrado na Figura 4.4 visa propor uma solução que realize o processo de forma mais automatizada que o processo atual.

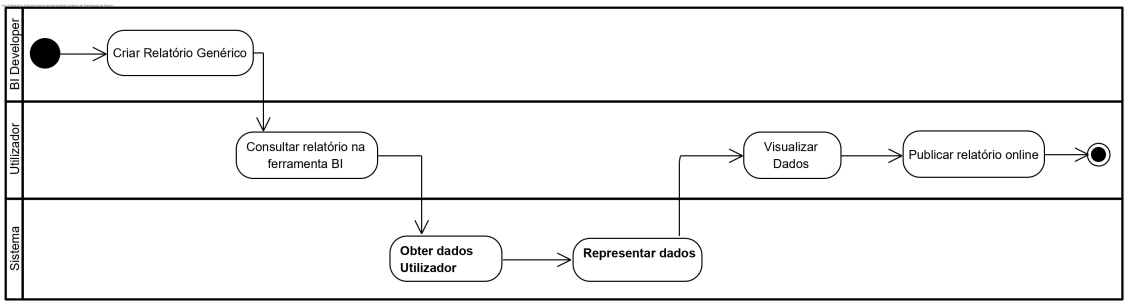


Figura 4.4: Processo Proposto

Este diagrama visa abordar um novo processo onde se identificam três intervenientes: *BI Developer*, Utilizador (cliente final) e Sistema.

Na realização deste processo, o *BI Developer* começa por criar um relatório genérico (*template*) que vai servir de base para representar os dados de cada um dos utilizadores. Desta forma, este relatório vai ser criado uma só vez e será entregue a cada utilizador. Este vai consultar o relatório na ferramenta de BI, que terá um Sistema, que funcionará como um mecanismo que obtém os dados em função do utilizador autenticado e os integre de forma automática com a ferramenta de BI. Posteriormente, a informação é apresentada no relatório, onde o utilizador pode visualizar os seus dados e publicar o relatório *online*, com o intuito de partilhá-lo com outros utilizadores e/ou criar *dashboards*.

Este novo processo caracteriza-se por ser mais automatizado e agilizado em relação ao atual, desde a obtenção de dados até à disponibilização da informação no relatório. Com isto, o *BI Developer* necessita de criar um relatório suficientemente genérico e que seja adaptativo para representar a informação dos utilizadores, sendo que, a parte de obter a informação do utilizador respetivo, é realizado pelo Sistema que se pretende implementar.

Assim sendo, existe a criação de um só relatório que vai ser entregue aos vários clientes, em oposição ao processo atual (um relatório criado para cada cliente). Desta maneira, permite a automatização do processo, onde existe a uniformização de criação de um relatório genérico, com a obtenção e integração dos dados do respetivo utilizador de forma automática na ferramenta de BI.

Conforme o supramencionado, este processo proposto, em comparação ao processo atual, permitirá colmatar e diminuir o tempo de implementação e manutenção, prevendo acabar com o fluxo repetitivo que existe.

A ferramenta de BI a utilizar será a mesma que se usa no desenvolvimento atual, uma vez que, analisando as várias ferramentas, não existem evidências destas apresentaram funcionalidades não existentes ou melhores que a ferramenta *Microsoft Power BI*, além de que é a ferramenta que os *BI Developers* estão habituados a trabalhar.

4.5 Estudo de Soluções

Nesta secção, são identificadas e analisadas as arquiteturas propostas para automatizar o processo existente, no sentido de dar resposta ao problema referido anteriormente na Secção 1.2.

Na Figura 4.5, encontra-se representado um diagrama de componentes de alto nível que idealiza, sob o ponto de vista de granularidade grossa, a arquitetura a implementar para resolver o problema específico.

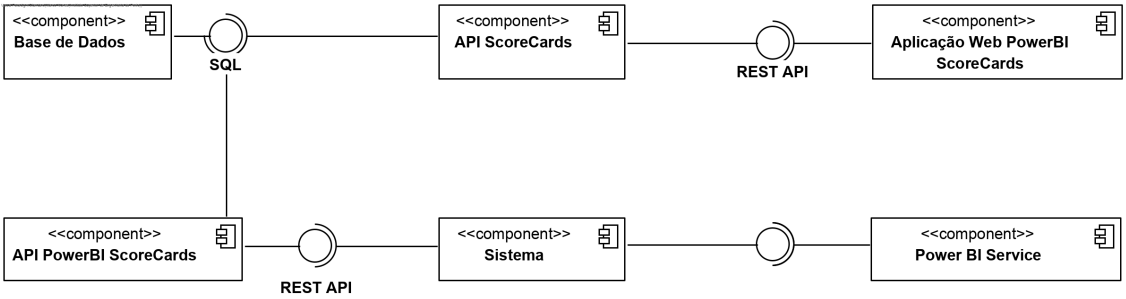


Figura 4.5: Diagrama de Componentes de Alto Nível Proposto

No diagrama da Figura 4.5, identificam-se seis componentes, sendo estes:

- Aplicação *Web PowerBI ScoreCards* – Representa a aplicação *web* onde são introduzidos dados relativos aos indicadores de desempenho para os *scorecards* definidos (abordada anteriormente na Secção 2.3.2).
- *API ScoreCards* – API que estabelece a comunicação entre a aplicação anterior e a base de dados, para permitir acesso e persistência de dados.
- Base de Dados – Armazena a informação introduzida na aplicação *web*, contendo informação sobre os *scorecards* definidos, o estado e meta a atingir pelos indicadores de desempenho.
- *API PowerBI ScoreCards* – Responsável por obter os dados da aplicação *web* e disponibilizá-los por diferentes *endpoints* para serem acedidos na ferramenta de BI. Esta API é abordada com mais detalhe na Secção 4.5.1.
- *Power BI Service* – Representa a aplicação *online* da ferramenta *Microsoft Power BI*, permitindo que os relatórios possam ser publicados e partilhados com outros utilizadores. Adicionalmente, pode-se criar *dashboards*.
- Sistema – Este componente será analisado nas secções seguintes, apresentando um detalhe ao nível de granularidade fina e representará as diferentes alternativas/soluções de como o mesmo se irá comportar para tornar o processo mais automatizado, em contraste ao processo atual.

Nesta arquitetura, podia-se considerar a ligação entre o componente Sistema e a *API ScoreCards*. No entanto, esta API é utilizada pela aplicação *web PowerBI ScoreCards*, sendo construída à volta do que esta necessita, tendo um comportamento diferente da *API PowerBI ScoreCards*, pois esta foi implementada para utilizar os dados da aplicação *web* com a ferramenta de BI (*Microsoft Power BI*). Desta maneira, a separação das responsabilidades

de cada API e as suas diferentes funções foram os fatores que permitiram decidir que o componente Sistema estabeleceria a sua ligação com a *API PowerBI ScoreCards*.

Assim sendo, o diagrama apresentado tem como objetivo automatizar o processo atual, sendo isto mais especificado nas Secções 4.5.2 e 4.5.3, com as diferentes abordagens consideradas.

4.5.1 API PowerBI ScoreCards

A *API PowerBI ScoreCards*, desenvolvida pela equipa de desenvolvimento da aplicação *web PowerBI ScoreCards*, assenta no padrão arquitetural REST [48], tendo como objetivo disponibilizar os dados da base de dados da aplicação *web PowerBI Scorecards* através do acesso aos diferentes *endpoints* que disponibiliza. Um *endpoint* de uma API é o ponto de entrada num canal de comunicação onde dois sistemas interagem entre si, no qual se acede aos recursos que o mesmo fornece [49].

Esta API está registada no Portal *Azure* da *Microsoft* [50] e suporta dois tipos de autenticação: *Basic Authentication* [51] e *OAuth* com Azure Active Directory (Azure AD). O Azure AD é um provedor de identidade da *Microsoft* que controla e gere o acesso dos utilizadores aos serviços, de modo a que estes possam se registar e aceder aos recursos que pretendem [52].

O primeiro tipo de autenticação, *Basic Authentication*, o *username* e *password* são encriptados em base 64 e enviados em cada chamada do *endpoint*, sendo, atualmente, considerado um método bastante inseguro [51].

Relativamente ao segundo tipo de autenticação, o *OAuth* funciona como um protocolo de autorização, que permite que as aplicações acedam aos dados dos utilizadores sem estes divulgarem as suas credenciais: *username* e *password*. Para tal, estas credenciais são substituídas por um *access token*. Um *access token* é um conjunto aleatório de caracteres que expira passado um certo tempo, e que é enviado nos pedidos para permitir a autenticação do utilizador, sendo considerado o mais seguro, uma vez que não envia as credenciais do utilizador [51].

A *API PowerBI ScoreCards* foi pensada e idealizada para utilizar os dados dos clientes da aplicação *web PowerBI ScoreCards*, à medida das necessidades de serem integrados com a ferramenta de BI – *Microsoft Power BI*, utilizando o tipo de autenticação *OAuth* com *Azure AD*, uma vez que é o mais seguro. Atualmente, este processo idealizado e descrito não é efetuado.

Os *endpoints* que disponibiliza são os seguintes:

- GET /scorecards – Obtém todos os *scorecards* do cliente.
- GET /scorecardsNodes/:scorecardId – Obtém o conjunto de indicadores de desempenho e áreas do *scorecard*, através do *scorecardId* que é indicado no Uniform Resource Locator (URL).
- GET /kpiDetails/:kpiId – Obtém a informação relativa ao indicador de desempenho, através do *kpiId* que é indicado no URL.

Todos os pedidos que irão ser realizados à API utilizam o *access token* relativo ao cliente, como forma de garantir segurança no acesso à informação pretendida.

4.5.2 Ligação Power BI Desktop com API PowerBI ScoreCards

Na Figura 4.6, existe a representação do componente Sistema mais detalhado, sendo que os restantes cinco componentes mantêm-se sem alterações, tal como foram apresentados no artefacto da Figura 4.5.

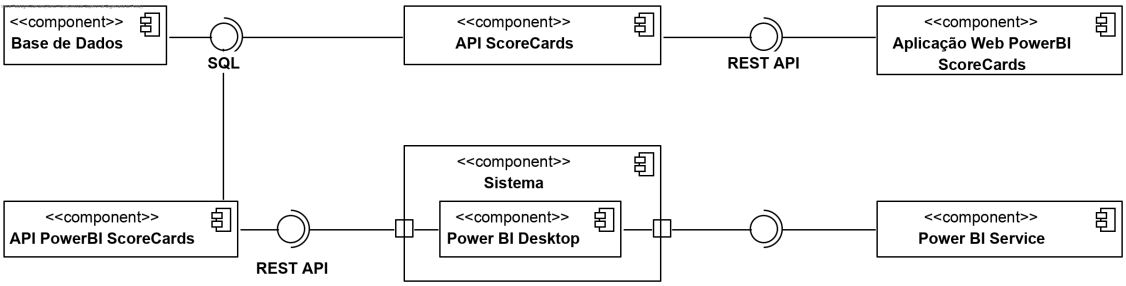


Figura 4.6: Diagrama de Componentes Proposto para Ligação Power BI Desktop com API PowerBI ScoreCards

Neste diagrama proposto, pode-se observar que o componente Sistema é composto por um componente: o *Power BI Desktop*. Este componente já foi explicado anteriormente, aquando da apresentação da arquitetura do processo atual (Secção 2.3.1), mas, neste caso, o componente *Power BI Desktop* apresenta um comportamento diferente.

Posto isto, pretende-se que o componente *Power BI Desktop* se conecte à *API PowerBI ScoreCards*, via REST API, como forma de obter os dados dos vários *endpoints* que esta disponibiliza e, posteriormente, haja a sua representação, criando um relatório para a informação obtida. Com isto, o componente *Power BI Service* permite que os relatórios possam ser publicados na aplicação *online*, sendo isto já realizado pelo cliente.

Face ao supratranscrito, o fluxo passa pela automatização do processo de obtenção dos dados e criar um relatório para depois ser utilizado pelos vários clientes. Desta forma, cada cliente ao aceder ao relatório previamente automatizado e preparado (relatório genérico) na ferramenta *Microsoft Power BI*, mais concretamente na aplicação *Power BI Desktop*, introduz as suas credenciais para permitir realizar a autenticação e autorização com a *API PowerBI ScoreCards* para obter os dados e a sua informação ser apresentada no relatório.

4.5.3 Ligação Conetor com API PowerBI ScoreCards

Uma outra alternativa, presente na Figura 4.7, apresenta os cinco componentes já referidos anteriormente (Base de Dados, *API ScoreCards*, *API PowerBI ScoreCards*, Aplicação *Web PowerBI ScoreCards* e *Power BI Service*), mais o sexto componente Sistema, proposto com uma diferente representação, que é explicado seguidamente.

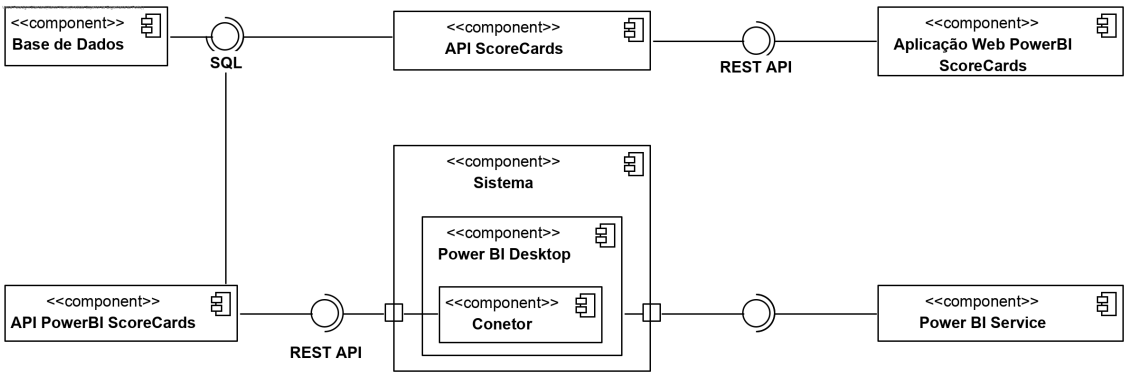


Figura 4.7: Diagrama de Componentes Proposto para Ligação Conetor com API PowerBI ScoreCards

Como ilustrado na Figura 4.7, o componente Sistema é representado por um componente na sua globalidade: o *Power BI Desktop*. Contudo, o componente *Power BI Desktop* é composto por um novo componente – Conetor. Este novo componente pressupõe a ligação com a *API PowerBI ScoreCards*, de forma a obter os dados para os representar na aplicação da ferramenta *Microsoft Power BI*, o *Power BI Desktop*, sob a forma de um relatório, podendo este ser depois publicado no *Power BI Service*. O Conetor funciona, assim, como a ligação entre a *API PowerBI ScoreCards* e aplicação *Power BI Desktop*, onde as ligações aos *endpoints* da API e a forma de autenticação e autorização com esta estão estabelecidas no componente Conetor.

Esta abordagem apresentada estabelece um processo automatizado no que diz respeito à obtenção de dados até à criação do relatório com a informação. Numa primeira fase, o componente *Conetor* tem como funcionalidade estabelecer a ligação aos *endpoints* da *API PowerBI ScoreCards*, procedendo aos corretos pedidos do cliente, de modo a adquirir a sua informação, sendo, seguidamente, representada a informação no relatório na aplicação *Power BI Desktop*. Posteriormente, este relatório pode ser publicado na aplicação *online* - *Power BI Service* - pelo cliente.

Assim sendo, tal como explicado na secção anterior, este relatório é previamente criado e preparado (relatório genérico) para ser entregue aos clientes. Seguidamente, cada cliente consulta o relatório na aplicação *Power BI Desktop* e, introduzindo as credenciais de autenticação, permite a ligação do Conetor à *API PowerBI ScoreCards*, obtendo os dados específicos que serão expostos nesse relatório, sendo este um fluxo automático.

O componente Conetor aqui apresentado é devido à aplicação *Power BI Desktop* permitir que sejam adicionados novos conetores desenvolvidos [21] aos já existentes na aplicação, tal como referenciado na Secção 2.2.1.

4.6 Escolha da Solução

Nesta secção, as alternativas propostas apresentadas anteriormente serão comparadas, de modo a escolher a melhor solução para dar resposta ao problema.

Posto isto, as soluções apresentadas anteriormente propõem um fluxo mais automatizado do processo, desde a obtenção de dados até à criação do relatório, deixando de ser um processo

repetitivo, tal como é realizado atualmente. Com esta abordagem será, primeiramente, criado um relatório genérico já preparado para obtenção dos dados e sua representação, ao qual será entregue a cada cliente, que só necessitará de abrir/consultar o relatório com a aplicação *Power BI Desktop*, realizar a autenticação necessária, como forma de serem obtidos os seus dados aquando das chamadas ao *endpoints* da *API Power BI ScoreCards* e, em seguida, os dados serão apresentados no relatório.

A utilização da *API Power BI ScoreCards* é um ponto que foi considerado nas soluções. Com a sua utilização, que suporta autenticação *OAuth* com Azure AD, permitindo aceder à informação do cliente que está a consultar o relatório na aplicação *Power BI Desktop*, o processo será mais automático, em contraste ao atual, que tem que se criar um relatório para cada cliente, com execução de *queries* à base de dados para obter os dados do mesmo e, só depois, o relatório é fornecido ao cliente (processo repetido para cada cliente).

Assim sendo, a solução a adotar passa por escolher a forma como a comunicação à *API PowerBI ScoreCards* vai ser realizada: por meio da ligação do *Power BI Desktop* (Secção 4.5.2) ou por meio da ligação do Conetor (Secção 4.5.3).

Como a API *PowerBI ScoreCards* está protegida utilizando *OAuth* com *Azure AD*, atualmente, esta ligação não é suportada pela aplicação *Power BI Desktop*, tal como apresentado na comunidade *Power BI* da *Microsoft* [53]. Em oposição, o Conetor permite definir vários tipos de ligação a fontes de dados, suportando: *OAuth, API Key, Windows Authentication* e *Basic Authentication* (referido na Secção 2.2.1), o que revela que, neste caso, o acesso à API *PowerBI ScoreCards* autenticada é assegurada. A lógica de autenticação pode, assim, ser definida no Conetor.

Além disto, com a criação de um Conetor específico para se comunicar à *API Power BI ScoreCards* (fonte de dados), permite que o mesmo seja desenvolvido à medida das necessidades. Isto significa que obtém-se apenas os dados pretendidos, uma vez que pode ocorrer primeiro uma filtragem destes antes de serem passados para a aplicação *Power BI Desktop*, para serem representados visualmente em relatórios.

Outro factor que o Conetor permite é a centralização de todos os pedidos aos *endpoints* da fonte de dados, definindo a forma como são acedidos ao mesmo. Mesmo que fosse possível a conexão do *Power BI Desktop* à API *Power BI ScoreCards*, as ligações necessárias a esta API ficavam estabelecidas no *Power BI Desktop*, onde os dados eram obtidos para depois ser elaborado o relatório.

No caso de acontecer alguma mudança no URL da API para aceder aos seus *endpoints*, isto levaria a que os relatórios que os clientes atuais tinham não funcionassem, significando que uma nova versão do relatório fosse fornecida para cada cliente (solução relativa à ligação do *Power BI Desktop com API PowerBI ScoreCards*). No Conetor, como os pedidos são aqui centralizados, a mudança ocorreria apenas e só no Conetor, não necessitando de novas entregas do relatório a cada cliente. Em termos de manutenção, o Conetor revela-se como o mais indicado.

Relativamente à segurança da informação, o Conetor protege a propriedade intelectual da empresa, uma vez que os pedidos realizados ficam omitidos no mesmo, não sendo possível de serem visualizados na aplicação *Power BI Desktop*.

Em suma, em virtude do que foi mencionado anteriormente, o *design* proposto na Figura 4.7 será a solução a adotar para resolver o problema.

4.7 Implantação

O diagrama de implantação ilustrado na Figura 4.8 representa a forma como os componentes da solução a desenvolver são instalados. Como tal, este diagrama em linguagem UML contempla o *design* escolhido da solução da Secção 4.5.3, abordando os seus componentes mais importantes.

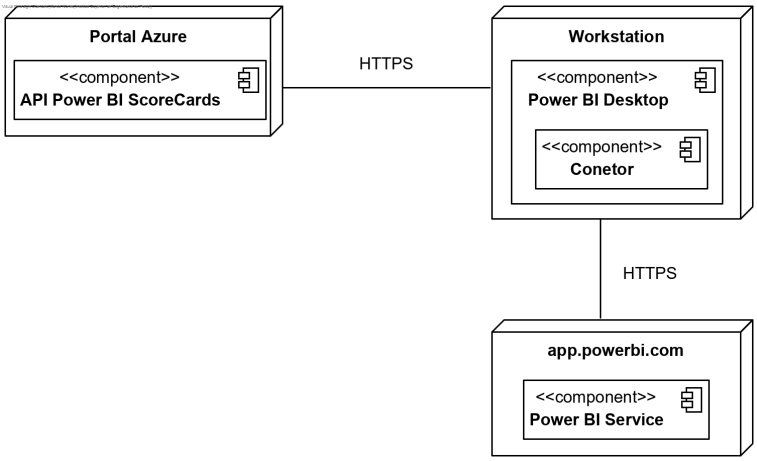


Figura 4.8: Diagrama de Implantação Proposto

Neste diagrama de implantação, identificam-se os seguintes dispositivos:

- Portal Azure – Servidor *Microsoft* responsável por alojar a *API PowerBI ScoreCards*.
- Workstation – Máquina responsável por ter a aplicação *PowerBI Desktop* instalada, com o respetivo Conetor a ser desenvolvido. O sistema operativo desta máquina deve ser *Windows*.
- app.powerbi.com – Servidor *online* que aloja a aplicação *Power BI Service*, que pode ser acedida via *web browser*, funcionado como um SaaS da ferramenta *Microsoft Power BI*.

Capítulo 5

Implementação

Este capítulo descreve a implementação da solução final resultante do desenho no Capítulo 4. Desta forma, começa-se por explicar o ambiente de desenvolvimento, referindo quais as ferramentas e tecnologias utilizadas. Seguidamente, faz-se uma introdução teórica do *OAuth* 2.0, com a explicação dos papéis envolventes e fluxo seguido, sendo este estudo importante para a implementação da autenticação no Conetor.

Posteriormente, aborda-se a implementação seguida no desenvolvimento do Conetor, a criação do relatório genérico, finalizando com a descrição da criação de um visual personalizado para a aplicação *Power BI Desktop*.

5.1 Ambiente de Desenvolvimento

Para o desenvolvimento de conetores personalizados com a ligação à fonte de dados pretendida, é necessário a instalação do *Power Query SDK*, para o editor de desenvolvimento *Visual Studio* [54]. O código implementado será escrito na linguagem M, de modo a que o Conetor desenvolvido seja reconhecido pela aplicação *Power BI Desktop*.

Complementarmente a isto, foi elaborado um relatório genérico, que serve como modelo (*template*) para a representação da informação de qualquer utilizador. Este relatório é elaborado na aplicação *PowerBI Desktop*, contendo vários visuais para a apresentação da informação.

Uma vez que a aplicação *PowerBI Desktop* permite que sejam adicionadas novas visualizações às existentes, foi desenvolvido um visual personalizado para representar a informação.

Para o desenvolvimento deste visual, foi utilizado o editor de desenvolvimento *Visual Studio Code* [55], sendo necessário instalar a ferramenta de visualizações *Power BI Visuals* [56]. O visual é desenvolvido na linguagem *TypeScript* [57], utilizando a biblioteca *D3* [58] para elaborar visualizações dinâmicas e interativas dos dados.

O sistema operativo do computador de trabalho deve ser *Windows*, devido ao facto da aplicação *Power BI Desktop* só ser possível de ser executada neste sistema.

5.2 OAuth 2.0

Os processos de autenticação e autorização são estabelecidos para garantir segurança no acesso aos dados por parte dos utilizadores. Desta forma, a autenticação é o processo que

confirma a identidade de um utilizador, enquanto que, a autorização, determina quais as permissões que o utilizador tem e a que dados deve aceder [59].

Como já referido na Secção 4.5.1, a *API PowerBI ScoreCards* encontra-se registada no Portal *Azure*, o que significa que utiliza o mecanismo de autenticação *Azure AD*. Com isto, esta API permite aos utilizadores com contas *Microsoft* (organizacionais ou de comunidade escolar) possam aceder à mesma, obtendo a informação pretendida, desde que estejam devidamente autorizados.

Este mecanismo simplifica a autenticação para quem desenvolve as aplicações (fica ao cargo de outra entidade), uma vez que fornece a identidade como um serviço (*identity-as-a-service*), suportando protocolos de autorização, tal como, o *OAuth* 2.0 [60]. No caso atual da *API PowerBI ScoreCards*, esta encontra-se com *Azure AD* a utilizar *OAuth 2.0*, para permitir autenticar e autorizar o acesso aos recursos que disponibiliza.

Desta forma, o protocolo *OAuth* introduz uma camada de autorização que, em vez de usar credenciais de *login*, utiliza um *access token*. Os papéis envolvidos neste protocolo são os seguintes [60]:

- *Authorization Server* – Provedor de identidade responsável por garantir a identidade do utilizador, de forma a conceder ou a negar o acesso a determinados recursos.
- *Resource Owner* – Entidade que possui os dados e tem o poder de permitir a terceiros aceder aos mesmos. Tipicamente, é o utilizador final.
- *OAuth Client* – É a aplicação que solicita acesso a um recurso protegido em nome do *Resource Owner*. O *OAuth Client* é a parte com a qual o utilizador final interage e que solicita *access tokens* ao *Authorization Server*.
- *Resource Server* – Local onde os recursos ou dados residem. É a API a que se quer aceder, a qual confia no *Authorization Server* para autenticar e autorizar o *OAuth Client*, utilizando *access tokens* para garantir que o acesso a um recurso possa ser concedido.

5.2.1 Fluxo Genérico

De maneira a ter-se uma perceção de como o protocolo *OAuth* funciona, a Figura 5.1 apresenta um fluxo genérico representativo do mesmo.

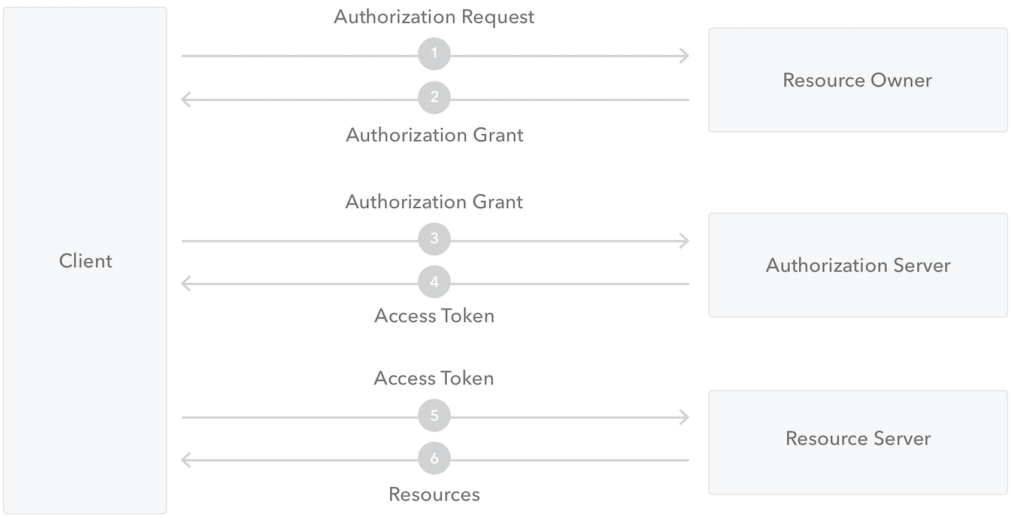


Figura 5.1: Fluxo Genérico do OAuth 2.0 [60]

Pela análise da Figura 5.1, é descrito o seguinte fluxo genérico:

1. O *OAuth Client* pede autorização ao *Resource Owner* para aceder aos seus recursos.
2. Caso o *Resource Owner* autorize o acesso, o *OAuth Client* recebe um *Authorization Grant*. Esta é uma credencial representativa da autorização do *Resource Owner*.
3. O *OAuth Client* pede um *access token*, autenticando-se com o *Authorization Server*, fornecendo o *Authorization Grant*.
4. Caso o *OAuth Client* seja devidamente autenticado e o *Authorization Grant* válido, o *Authorization Server* envia um *access token* para o *OAuth Client*.
5. O *OAuth Client* pede acesso aos recursos protegidos pelo *Resource Server*, e autentica-se com este, fornecendo o *access token* anteriormente obtido.
6. Caso o *access token* seja válido, o *Resource Server* dá resposta ao pedido previamente realizado pelo *OAuth Client*.

Contudo, o *OAuth 2.0* define quatro tipo de fluxos, suportados pelo *Azure AD*, para obter um *access token*, também denominados de *grant types*, sendo estes [60]: *Authorization Code, Implicit, Resource Owner Password Credentials* e *Client Credentials*.

No caso atual da *API PowerBI ScoreCards*, o fluxo utilizado para aceder aos seus recursos protegidos é o *OAuth 2.0 Authorization Code Grant* [61], que será analisado com detalhe na Secção 5.2.2. Como o objetivo desta dissertação é a implementação do Conetor com a ligação à *API PowerBI ScoreCards*, a lógica de autenticação a ser desenvolvida no Conetor seguirá este fluxo.

5.2.2 Authorization Code Grant

O fluxo *Authorization Code Grant* é utilizado para o *OAuth Client* obter tanto um *access token*, bem como, um *refresh token*, como forma destes permitirem o acesso aos recursos protegidos do *Resource Server* [62].

Um *refresh token* é um conjunto aleatório de caracteres que, ao contrário do *access token*, tem um longo tempo de vida e é utilizado para se obter um novo *access token* quando este expira [63]. Assim, permite que o *OAuth Client* continue a aceder aos recursos protegidos com o devido *access token*, evitando perguntar novamente ao utilizador que consinta autorização.

Como tal, é possível estabelecer uma definição concreta dos papéis que o *OAuth* pressupõe e o protocolo que preconiza, tal como, ilustrado na Figura 5.1, com a solução que se pretende implementar (Secção 4.5.3). Assim sendo, neste caso, o *Authorization Server* é o *Azure AD*, o *Resource Ower* é o utilizador final com conta *Microsoft*, *OAuth Client* é o Conetor e o *Resource Server* é a *API PowerBI ScoreCards*.

O diagrama UML de sequência da Figura 5.2 representa o fluxo *OAuth 2.0 Authorization Code Grant* entre o Utilizador, Conetor, *Azure AD* e a *API PowerBI ScoreCards*.

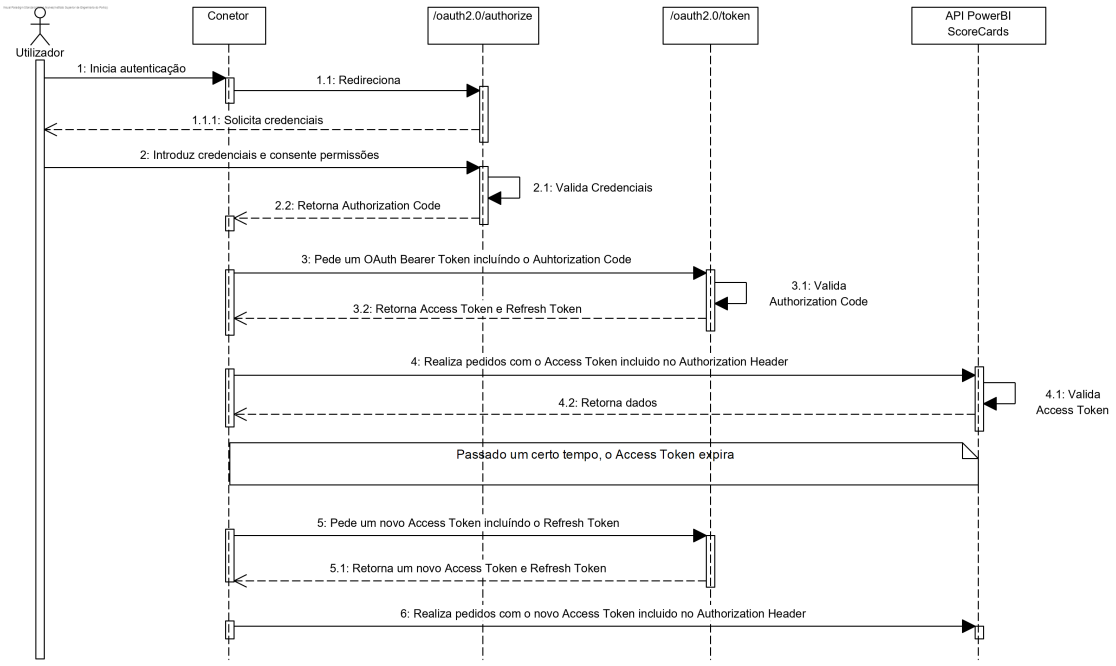


Figura 5.2: Diagrama de Sequência do Authorization Code Grant

Através do diagrama de sequência da Figura 5.2, constata-se que o fluxo começa com o *OAuth Client* (Conetor) a direcionar o utilizador para o *endpoint /authorize* do *Azure AD*. Para tal, o utilizador vai ser questionado para introduzir as suas credenciais e completar a autenticação, sendo que o *Azure AD* vai assegurar que o utilizador deu acesso às permissões que o Conetor pretende obter. Uma vez autenticado e as permissões concedidas, o *endpoint* do *Azure AD* vai retornar a resposta para o Conetor, que contém o *authorization code*.

De seguida, o Conetor pede um *access token* ao *endpoint /token* do *Azure AD*, através do método POST, incluindo neste pedido o *authorization code* anteriormente obtido. Com

isto, o Conetor autentica-se com o *Azure AD*, o que faz com que este valide o *authorization code* e, no caso de ser válido, o *Azure AD* envia para o Conetor um *access token* e um *refresh token*. Posteriormente, com o *access token*, este é incluído no *Authorization Header* aquando dos pedidos à *API PowerBI ScoreCards* para obter a informação pretendida do utilizador.

Contudo, passado um certo tempo (uma hora), o *access token* expira, e, devido a isso, é realizado outro pedido ao *endpoint /token* do *Azure AD*, através do método POST, incluindo o *refresh token* em vez do *authorization code*. Assim, o *Azure AD* vai validar este *refresh token* e, caso seja válido, emite um novo *access token* e *refresh token*. Desta maneira, o Conetor realiza o comportamento descrito anteriormente, realizando pedidos à *API PowerBI ScoreCards* com o *access token* incluído no *Authorization Header*.

5.2.3 Endpoints

Tal como observado no diagrama da Figura 5.2, existem dois *endpoints* que o *Azure AD* disponibiliza (*/authorize* e */token*), onde são efetuados pedidos a estes com o *OAuth 2.0*.

Nos pedidos Hypertext Transfer Protocol (HTTP) que são realizados a estes dois *endpoints*, utiliza-se o método HTTP associado (GET ou POST).

Posto isto, estes dois *endpoints* são analisados de seguida, para o cenário que se vai desenvolver.

/authorize

Para realizar o pedido a este *endpoint*, utiliza-se o método GET e são necessários os seguintes parâmetros [64]:

- *client_id* – Corresponde ao *client_id* da *API PowerBI ScoreCards* que é atribuído pelo Portal *Azure* quando foi registada.
- *response_type* – Permite inferir ao *Authorization Server* (*Azure AD*) qual fluxo (*grant type*) vai executar. No caso atual, o fluxo é o *Authorization Code Grant*, logo, o valor deste parâmetro é *code*.
- *scope* – Conjunto de permissões que se pretende que o utilizador dê consentimento.
- *state* – Valor referente a questões de segurança.
- *redirect_uri* – URL onde os utilizadores são direcionados após a autorização.
- *resource* – Corresponde à *Application Id* atribuída pelo Portal *Azure* à *API PowerBI ScoreCards*.

Com este pedido, vai-se obter um *authorization code*, que será utilizado no *endpoint* seguinte.

/token

Para este *endpoint*, com o intuito de se obter um *access token*, utilizando o *authorization code*, faz-se uso do método POST com os seguintes parâmetros [64]:

- *client_id* – Tal como referido anteriormente, este valor é atribuído pelo Portal *Azure* quando a *API PowerBI ScoreCards* foi registada.
- *client_secret* – Corresponde ao segredo da *API PowerBI ScoreCards* que é criado quando é registada a API no Portal *Azure*.
- *grant_type* – Corresponde ao valor que o *grant type* irá assumir consonte o fluxo a ser utilizado. No caso atual, o fluxo é o *Authorization Code Grant*, logo, este valor será *authorization_code*.
- *code* – Valor do *authorization code* obtido anteriormente.
- *redirect_uri* – URL onde os utilizadores são direcionados após a autorização.
- *resource* – Corresponde à *Application Id* atribuído pelo Portal *Azure* à *API PowerBI ScoreCards*.

Como resposta deste pedido, obtém-se um *access token* e um *refresh token*, sendo que este último pode ser utilizado para obter um *access token*, fazendo uso deste *endpoint*. Desta forma, o pedido a ser realizado é em quase todo semelhante ao descrito anteriormente, apresentando diferenças no parâmetro *grant_type* e no parâmetro *code*, uma vez que este último é substituído pelo parâmetro *refresh_token*. Assim sendo, estes parâmetros assumem os seguintes valores:

- *grant_type* – Este parâmetro é igualado a *refresh_token* (*grant_type = refresh _token*).
- *refresh_token* – Valor do *refresh token* obtido anteriormente.

Com o resultado deste pedido, obtém-se um novo *access token* e *refresh token*.

5.3 Conetor

A implementação do Conetor assentou em quatro partes essenciais:

- Autenticação – Especificação do tipo de autenticação a ser realizado com a fonte de dados pretendida (*API PowerBI ScoreCards*).
- Comunicação com a *API PowerBI ScoreCards* – Estabelece as ações que o Conetor irá realizar e qual a informação que irá obter da fonte de dados.
- Detalhes Gráficos – Forma gráfica de como o Conetor se irá apresentar na aplicação *Power BI Desktop*.
- Metadados – Relativo ao tamanho que o *icon* do Conetor terá na aplicação *Power BI Desktop*.

Face a isto, os dois primeiros pontos apresentados serão analisados nas secções seguintes, visto que, os restantes pontos, são relativos a detalhes gráficos, não existindo um concreto desenvolvimento de código.

Relativamente à implantação do Conetor no *Power BI Desktop*, a abordagem realizada é descrita na Secção 5.3.3.

5.3.1 Autenticação

O Conetor desenvolvido permitirá a conexão à *API PowerBI ScoreCards*, seguindo o fluxo do *OAuth 2.0 Authorization Code Grant* explicado na Secção 5.2.2, autenticando os utilizadores com conta *Microsoft* para, posteriormente, obter a informação relativa aos seus *scorecards*.

Para tal, aquando da definição da autenticação *OAuth* no desenvolvimento do Conetor, é esperado que contenha a seguinte definição de funções: *StartLogin, FinishLogin, Refresh* e *Logout*. Estas funções são executadas automaticamente assim que o utilizador começa a estabelecer a autenticação com o Conetor.

No Excerto de Código 5.1, apresenta-se a implementação da função *StartLogin*, que irá dar início ao fluxo do *OAuth*.

```
StartLogin = (resourceUrl, state, display) =>
    let
        authorizeUrl = "https://login.microsoftonline.com/common/oauth2/authorize?" &
            Uri.BuildQueryString([
                  client_id = client_id,
                  resource = client_id,
                  state = state,
                  response_type = "code",
                  scope = "User.Read",
                  redirect_uri = redirect_uri
            ])
    in
        [
            LoginUri = authorizeUrl,
            CallbackUri = redirect_uri,
            WindowHeight = windowHeight,
            WindowWidth = windowWidth,
            Context = null
        ];
```

Excerto de Código 5.1: Função StartLogin da Autenticação OAuth

Como se pode observar pelo Excerto de Código 5.1, o código implementado segue a definição dos parâmetros necessários para o *endpoint /authorize*, tal como apresentado na Secção 5.2.3. Em alguns parâmetros, o seu valor é atribuído por meio de variáveis globais, para serem reaproveitas futuramente, mas também, para manter confidencialidade de dados da *API PowerBI ScoreCards*. Assim sendo, este excerto de código irá perguntar pelas credenciais do utilizador e, no caso de ser a primeira vez que efetua o *login* na *API PowerBI ScoreCards*, irá perguntar se pode aceder a certas permissões, que estão definidas no parâmetro *scope*. Neste aspeto, o parâmetro assume o valor "User.Read", que permite ler o perfil do utilizador que se vai autenticar.

No caso do utilizador completar o fluxo de autenticação, o *Azure AD* redireciona-o para o URL de retorno, com um código temporário no parâmetro *code*. Assim sendo, a função *FinishLogin*, apresentada no Excerto de Código 5.2, vai extrair o código do parâmetro *callbackUri*, correspondente ao *authorization code* e, de seguida, trocá-lo por um *access token*, utilizando a função *TokenMethod*.

```
FinishLogin = (context, callbackUri, state) =>
    let
        parts = Uri.Parts(callbackUri)[Query],
        result = if (Record.HasFields(parts, {"error", "error_description"})) then
                    error Error.Record(parts[error], parts[error_description], parts)
                 else
                    TokenMethod("authorization_code", "code", parts[code])
    in
        result;
```

Excerto de Código 5.2: Função FinishLogin da Autenticação OAuth

Caso a resposta do Excerto de Código 5.2 não contenha nenhum erro, este *authorization code* é passado para a função *TokenMethod*, apresentada no Excerto de Código 5.3.

```
1  // grantType:  Maps to the "grant_type" query parameter.
2  // tokenField: The name of the query parameter to pass in the code.
3  // code:       The actual code (authorization_code or refresh_token) to send to the service.
4  TokenMethod = (grantType, tokenField, code) =>
5      let
6          query = [
7              client_id = client_id,
8              client_secret = client_secret,
9              resource = client_id,
10             grant_type = grantType,
11             redirect_uri = redirect_uri
12         ],
13         queryWithCode = Record.AddField(query, tokenField, code),
14
15         response = Web.Contents("https://login.microsoftonline.com/common/oauth2/token", [
16             Content = Text.ToBinary(Uri.BuildQueryString(queryWithCode)),
17             Headers = [
18                 #"Content-type" = "application/x-www-form-urlencoded",
19                 #"Accept" = "application/json"
20             ],
21             ManualStatusHandling = {400}
22         ]),
23
24         parts = Json.Document(response),
25         result = if (Record.HasFields(parts, {"error", "error_description"})) then
26                     error Error.Record(parts[error], parts[error_description], parts)
27                 else
28                     parts
29     in
30         result;
```

Excerto de Código 5.3: Função TokenMethod

No Excerto de Código 5.3, é realizado um pedido para o *endpoint /token*, com todos os parâmetros necessários tal como apresentado na Secção 5.2.3, através do método POST. Desta forma, é obtido um *access token* e um *refresh token*, que se encontram armazenados na variável "parts", presente na linha 24.

Apesar da função *TokenMethod* não ser uma definição exigida da autenticação *OAuth* aquando a sua especificação no Conetor, esta funciona como reaproveitamento de código para, simultaneamente, obter um *access token* a partir de um *authorization code* ou *refresh token*. Assim sendo, a variável "grantType", presente na linha 10, permitirá saber sobre qual tipo se está a tratar, realizando o pedido pretendido.

No Excerto de Código 5.4, apresentam-se as restantes funções exigidas (*Refresh* e *Logout*) da autenticação *OAuth*.

```
1  Refresh = (resourceUrl, refresh_token) => TokenMethod("refresh_token", "refresh_token", refresh_token);
2
3  Logout = (token) => "https://login.microsoftonline.com/logout.srf";
```

Excerto de Código 5.4: Funções Refresh e Logout da Autenticação OAuth

Nas funções apresentadas do Excerto de Código 5.4, a função *Refresh* é chamada quando o *access token* expira. Para tal, fazendo uso da função *TokenMethod*, é passado o valor do *refresh token* para esta função, que seguirá a lógica já descrita anteriormente aquando a análise do Excerto de Código 5.3, obtendo-se um novo *access token* e *refresh token*.

A função *Logout*, presente na linha 3, é uma implementação simples que retorna um URL fixo, expirando, por sua vez, o atual *access token* e *refresh token*.

5.3.2 Comunicação com a API PowerBI ScoreCards

A *API PowerBI ScoreCards* disponibiliza três *endpoints*, com o intuito de obter a informação do utilizador (Secção 4.5.1). Desta forma, em cada um dos pedidos realizados a estes *endpoints*, o *access token* do utilizador (obtido pelas funções de Autenticação – Secção

5.3.1) é incluído no *Authorization Header* do pedido, sendo isto feito de forma automática pela ferramenta *Power Query SDK*.

Posto isto, de modo a realizar os pedidos pretendidos, utiliza-se a função da linguagem M denominada *Web.Contents* [65], que irá retornar a informação presente de um URL indicado. Para tal, é indicada para esta função um URL base (endereço URL da *API PowerBI ScoreCards*), juntamente com o *endpoint* pretendido.

O Excerto de Código 5.5 apresenta a implementação parcial das funções relativas a realizar os pedidos a cada um dos *endpoints* da *API PowerBI ScoreCards*.

Estas funções são estabelecidas na função principal do Conetor (função *Main)*, que é sempre executada quando existe alguma interação com o Conetor. Desta forma, as funções definidas no Excerto de Código 5.5 são automaticamente chamadas e o código definido em cada uma destas é prontamente espoletado.

```
1  getScoreCardsTable = () as table =>
2      let
3          source = Json.Document(Web.Contents(baseURL & "/scorecards")),
4          convertToTable=Table.FromList(source, Splitter.SplitByNothing(), null, null, ExtraValues.Error),
5          expandColumns = Table.ExpandRecordColumn(convertToTable, "Column1", {"id", "name",
6                              "numberOfLevels", "nodeStructure", "nodeStructureConfig", "status", (...)
7                          }
8      in
9          expandColumns;
10
11 getScoreCardsNodesTable = (scoreCardsTable as table) as table =>
12     let
13         source = Table.SelectColumns(scoreCardsTable,{"id"}),
14         renameColumn = Table.RenameColumns(source,{{"id", "id_sc"}}),
15         addUrlCombine = Table.AddColumn(renameColumn , "URL",
16                         each Uri.Combine(baseURL, "/scorecardNodes?scorecardId=" & Text.From([id_sc]))),
17
18         changeTypeURL = Table.TransformColumnTypes(addUrlCombine,{{"URL", type text}}),
19         addColumnWebContent = Table.AddColumn(changeTypeURL, "WebContent",
20                               each Json.Document(Web.Contents([URL]))),
21     (...)
22
23 getKPITable = (scoreCardsNodesTable as table) as table =>
24     (...)
25         addUrlCombine = Table.AddColumn(expandEachYear, "URL", each Uri.Combine(
26         baseURL, "/kpiDetails?kpiId=" & Text.From([kpiId]) & "&year=" &  Text.From([year_temp]))),
27
28         changeTypeURL = Table.TransformColumnTypes(addUrlCombine,{{"URL", type text}}),
29
30         addColumnWebContent = Table.AddColumn(changeTypeURL, "WebContent",
31                               each Json.Document(Web.Contents([URL]))),
32     (...)
```

Excerto de Código 5.5: Funções que Realizam Pedidos à API PowerBI ScoreCards

No Excerto de Código 5.5, o primeiro pedido a ser realizado à *API PowerBI ScoreCards* é ao *endpoint /scorecards*, que se encontra presente na função "getScoreCardsTable" (linha 1).

Como o pedido anterior obterá todos os *scorecards* do utilizador, por conseguinte, pretende-se extrair a informação de cada um destes, sendo isso conseguido pela função "getScoreCardsNodesTable" (linha 11). Apesar de ser apresentada uma implementação incompleta da função, esta baseia-se, essencialmente, em obter toda a informação de cada *scorecard* (conjunto de indicadores de desempenho) ao *endpoint /scorecardNodes* e agrupar esses dados numa tabela.

Por último, para obter a informação de cada indicador de desempenho de cada *scorecard*, a função "getKPITable" (linha 23) recebe por parâmetro a informação retornada da função anterior e utiliza o *endpoint /kpiDetails*. Na resposta deste pedido, obtém-se os valores que cada indicador de desempenho registou nos meses do ano especificado no parâmetro "year". No cenário atual, para cada indicador de desempenho, é obtida a informação que o mesmo obteve desde o ano de 2019 até ao ano corrente.

Posto isto, cada uma destas funções permite estabelecer os pedidos e formatar os dados conforme pretendido, para depois os mesmos serem carregados, posteriormente, para a aplicação *Power BI Desktop*. Assim sendo, cada função agrupa um conjunto de dados, que dá origem à criação de uma tabela, que depois é apresentada na aplicação *Power BI Desktop*, por via da definição de uma tabela de navegação.

Uma tabela de navegação possibilita uma experiência mais facilitada ao utilizador aquando o uso do Conetor, permitindo visualizar quais as tabelas que o Conetor fornece, escolhendo quais as que pretende importar, sendo isto possível depois de estar devidamente autenticado [66]. Para este propósito, é utilizada a função *Table.ToNavigationTable* [67], já existente no *Power Query SDK*, que permite especificar as tabelas a retornar pelo Conetor de uma forma simples e organizada. Assim, as tabelas que a tabela de navegação exibe são:

- ScoreCards – Tabela retornada pela função "getScoreCardsTable".
- ScoreCardsNodes – Tabela retornada pela função "getScoreCardsNodeTable".
- KPI – Tabela retornada pela função "getKPITable".

5.3.3 Implantação no Power BI Desktop

Como forma de permitir que o Conetor esteja disponível na aplicação *Power BI Desktop*, é necessário realizar *build* do projeto no editor *Visual Studio*. Como resultado, obtém-se um ficheiro na extensão *.mez*, que deve ser copiado para o diretório `"Documents\Microsoft Power BI Desktop\Custom Connectors"`.

Assim sendo, o Conetor é possível de ser acedido na aplicação *Power BI Desktop*, aquando a escolha de qual a fonte de dados pretendida, procedendo à devida autenticação que o mesmo sugere.

A Figura 5.3 mostra o Conetor desenvolvido "ScoreCardsConnector_OAuth (Beta)" , indicado pelo retângulo a vermelho, na aplicação *Power BI Desktop*. Para ser utilizado, basta selecioná-lo e, de seguida, pressionar o botão "Connect".

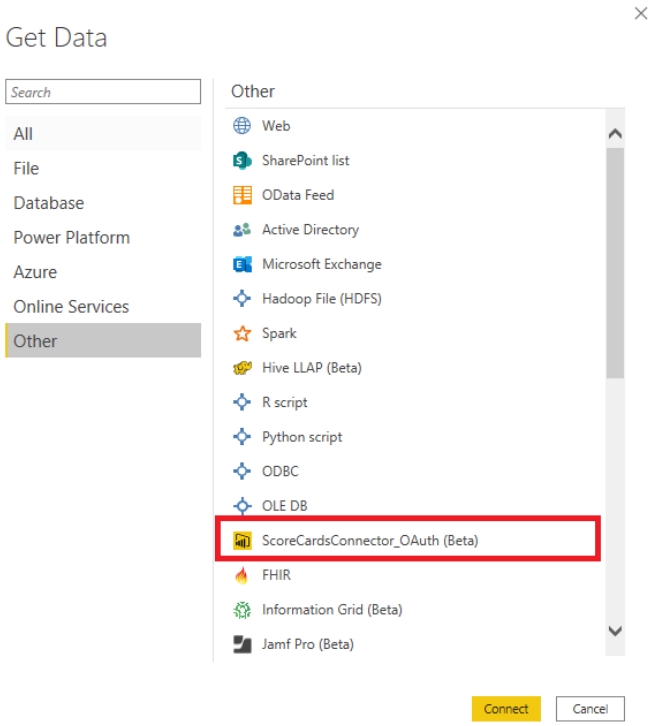


Figura 5.3: Acesso ao Conetor no Power BI Desktop

5.4 Relatório Genérico

Em virtude de se representar a informação proveniente de cada utilizador (cliente final), os seus dados podem ser apresentados por meio de diferentes visualizações gráficas.

O relatório genérico, criado na extensão *.pbit* [68], funciona como um *template* que se pretende que seja utilizado por todos os utilizadores, partilhando estes todos o mesmo padrão e estrutura do relatório definido. Desta forma, o utilizador ao consultar este relatório na aplicação *Power BI Desktop*, realizará a autenticação com o Conetor, responsável por obter os seus dados que serão, seguidamente, representados no relatório.

De forma a que se proceda a diferentes filtragens nos dados e estes se apresentem em conformidade com o que foi selecionado, é necessário realizar primeiro uma correta modelação de dados no relatório genérico. Com isto, pode-se conetar dados de várias tabelas, estabelecendo uma relação entre as mesmas através de uma coluna comum, com a definição da cardinalidade pretendida [69].

Posto isto, conforme o que foi referido na Secção 5.3.2, o Conetor fornece três tabelas, que são importadas para a aplicação *Power BI Desktop*, onde se efetua a correta modelação de dados.

A Figura 5.4 apresenta o modelo de dados criado para o relatório genérico na aplicação *Power BI Desktop*, para permitir a correta visualização da informação no relatório aquando a definição de filtragens nos dados.

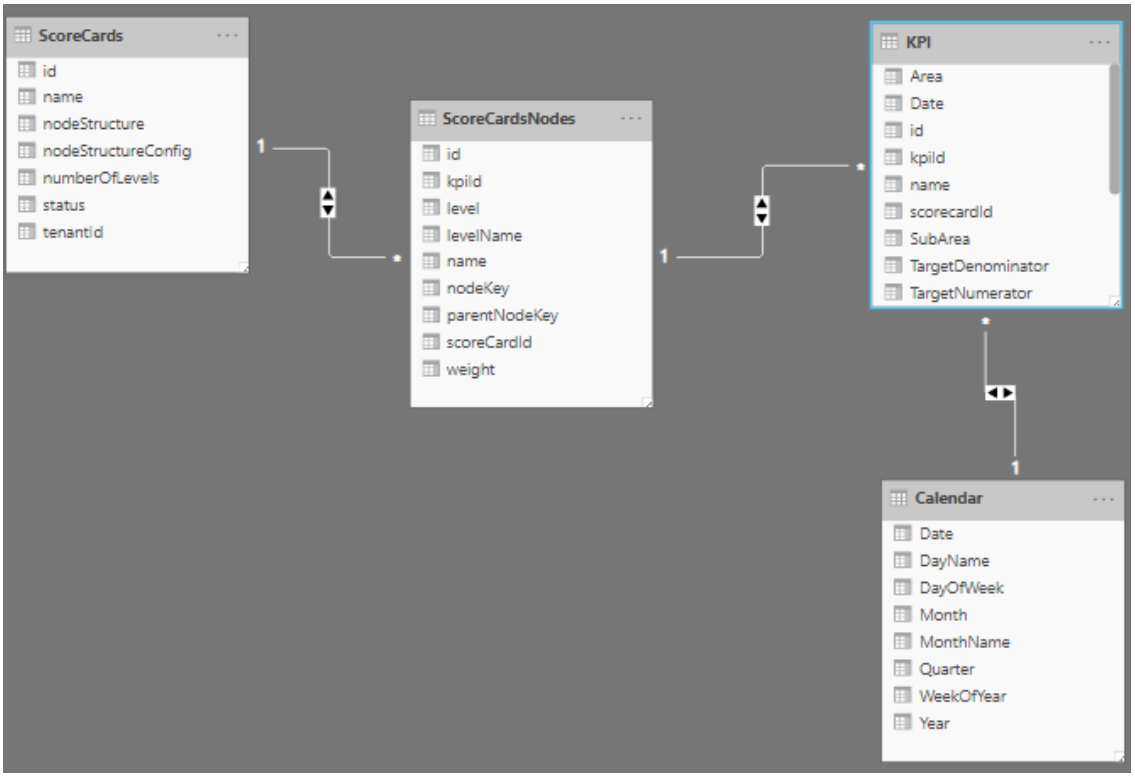


Figura 5.4: Modelo de Dados no Power BI Desktop

Tal como ilustrado na Figura 5.4, esta contempla a modelação dos dados, com a definição das relações entre as diferentes tabelas e suas cardinalidades. Além das tabelas importadas pelo Conetor, foi criada uma nova tabela, denominada por *Calendar*, que corresponde a um calendário com todas as datas desde o início do ano de 2019 até ao ano atual.

A tabela *Calendar* permitirá uma filtragem dos indicadores de desempenho aquando a definição de filtros relativos ao mês e/ou ano. Deste modo, como existe uma relação de um para muitos (1:*) entre a tabela *Calendar* e *KPI*, isto significa que, com a definição de um filtro do mês e/ou ano, várias instâncias de indicadores de desempenho vão ser visualizadas.

A Figura 5.5 ilustra a primeira página criada para o relatório que representa os dados de um *scorecard*.

Figura 5.5: Primeira Página do Relatório Genérico

Pela observação da Figura 5.5, constata-se que esta página de relatório possui a opção de selecionar qual o ano, mês e *scorecard* pretendido, apresentando os dados dos indicadores de desempenho face à filtragem definida.

A variedade de visualizações gráficas nesta página do relatório possibilita verificar o número de indicadores de desempenho que existem, a quantidade destes que já alcançou o valor a atingir (meta) estabelecido (através do gráfico com o título "Nr KPI Target Value Reached"), assim como, verificar qual o valor atual e meta de cada indicador de desempenho e a percentagem que o valor atual já atingiu para a meta definida.

Em complemento à página apresentada anteriormente, foi criada uma segunda página no relatório que promove outra análise diferente da informação, tal como apresentado na Figura 5.6.

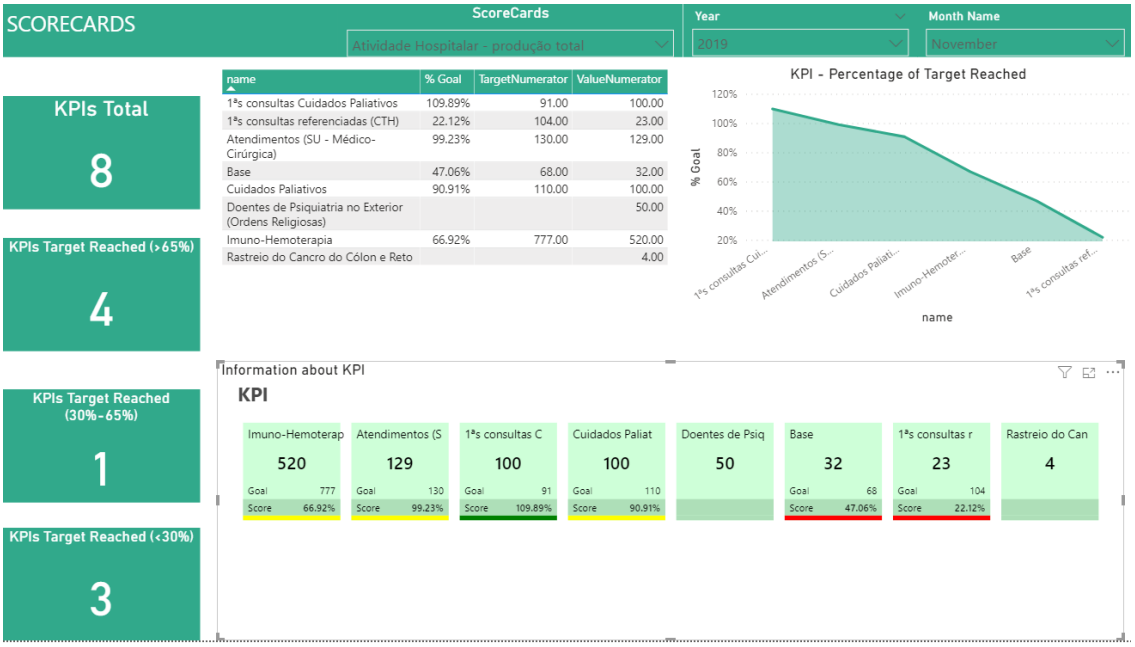


Figura 5.6: Segunda Página do Relatório Genérico

A segunda página do relatório permite selecionar o ano, mês e *scorecard*, contendo a particularidade de detalhar o número de indicadores de desempenho que se encontram em certos intervalos de percentagem definidos (percentagem de valor de meta já atingido). Para definição destes intervalos, foram criadas métricas através da linguagem DAX da aplicação *Power BI Desktop*. Os quatros visuais representativos desta informação denominam-se por *cards*, e estão representados de forma retangular, em orientação vertical, posicionando-se do lado esquerdo da página do relatório.

Relativamente às restantes visualizações, estas focam-se em representar os dados de cada indicador de desempenho (valor atual, valor a atingir e percentagem de meta já alcançada), permitindo verificar quais apresentam os resultados esperados e tomar as devidas decisões conforme a análise efetuada.

Os dados apresentados representam uma simulação da consulta do relatório genérico com a conexão ao Conetor desenvolvido, realizando a devida autenticação por parte do utilizador. Por sua vez, este relatório pode ser publicado *online*, no *Power BI Service*, para ser criado um *dashboard* e/ou ser partilhado com outros utilizadores.

5.5 Criação de Visual Personalizado

Em complemento aos visuais que a aplicação *Power BI Desktop* disponibiliza, podem ser adicionadas novas visualizações. A criação deste visual corresponde a um dos Requisitos Não Funcionais (Secção 4.2.2) relativo à criação de novos visuais personalizados para representar a informação no relatório.

O visual da Figura 5.6 com o título "Information about KPI" é um exemplo de uma visualização criada para representar a informação, neste caso concreto, ilustra os dados dos

indicadores de desempenho (cada quadrado do visual representa um indicador de desempenho).

Para a criação deste visual, foram seguidos um conjunto de passos que se encontram explicados nas secções seguintes. Este é um desenvolvimento independente ao já apresentado anteriormente (Conetor), não estando relacionado com este.

No final da implementação do visual personalizado, o mesmo é importado para a aplicação *Power BI Desktop*, para ser utilizado nos relatórios a serem elaborados.

5.5.1 Processamento de Dados

Uma vez instalada a ferramenta de criação de visualizações (*Power BI Visuals*), esta é utilizada para criar um novo projeto, executando o comando "`pbiviz new <nome do projeto>`" num terminal *PowerShell* [70]. Para a edição deste projeto, é utilizado o editor *Visual Studio Code*.

Com a definição deste projeto, que corresponde à criação de um novo visual personalizado, o projeto contém uma estrutura de código predefinida, assim como, quais os ficheiros que são utilizados e que devem ser modificados para reproduzir as mudanças pretendidas.

Como o intuito da criação deste visual é a representação da informação dos indicadores de desempenho, os dados a mostrar relativamente a cada indicador são sobre o seu nome, valor atual, valor a atingir e percentagem de valor a atingir já alcançado.

Para a definição dos campos que o visual deverá apresentar, é necessária a configuração do ficheiro *capabilities.json*, onde são definidos os *dataRoles* e os *dataViewMappings*. A primeira definição (*dataRoles*) corresponde aos campos que são especificados e que contêm os dados dos indicadores de desempenho, e a segunda definição (*dataViewMappings*) é relativa a como os campos de dados se relacionam entre si [71].

Assim sendo, com a definição destes campos no ficheiro *capabilities.json*, o utilizador da aplicação *Power BI Desktop* tem a possibilidade de indicar no painel de propriedades do visual as colunas pretendidas para cada campo. A Figura 5.7 ilustra o painel de propriedades do visual "Information about KPI", resultante da implementação final do visual realizada.

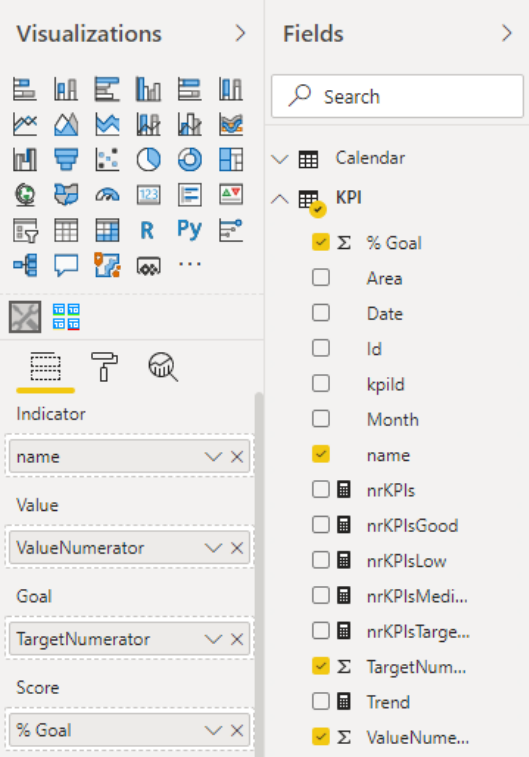


Figura 5.7: Painel de Propriedades do Visual

Na análise da Figura 5.7, as colunas que se encontram selecionadas na tabela "KPI" são as escolhidas para apresentar a sua informação nos campos definidos do visual personalizado, ou seja, os dados dos indicadores de desempenho (nome, valor atual, valor a atingir e percentagem de valor a atingir já alcançado).

5.5.2 Desenvolvimento dos Elementos Visuais

O ponto de partida para o desenvolvimento da lógica dos elementos visuais encontra-se no ficheiro *visual.ts*, que implementa dois métodos principais ("constructor" e "update") que se encontram apresentados de forma parcial no Excerto de Código 5.6.

```
export class Visual implements IVisual {
    public constructor(options: VisualConstructorOptions) {
        this.host = options.host;
        this.target = options.element;
        //...
    }

    public update(options: VisualUpdateOptions) {
        this.dataView = options.dataViews[0];
        //...
    }
}
```

Excerto de Código 5.6: Métodos da Classe Visual

Esta classe "Visual" implementa a interface "IVisual", onde o método "constructor" é responsável por inicializar todas as variáveis da construção do visual personalizado, enquanto que, o método "update", tem como finalidade desenhar o visual. O método "update" é sempre chamado quando acontece alguma interação no próprio visual ou noutros visuais presentes no relatório, como forma de atualizar os dados do visual em conformidade com o

que foi selecionado (por exemplo, a seleção de um determinado mês e/ou ano, resulta em atualizar os dados dos indicadores de desempenho).

O argumento "options" que é passado no método "update" permite saber quais os dados que vão ser consumidos neste método para, posteriormente, serem desenhados. A informação obtida deste parâmetro é guardada na variável "dataView", tal como ilustrado na linha de código 9 (Excerto de Código 5.6).

Tal como referido na Secção 5.5.1, o ficheiro *capabilities.json* permite a definição dos campos no visual. Desta forma, permite-se que exista uma passagem da informação das colunas selecionadas nestes campos para o argumento "options" do método "update".

Com o auxílio da biblioteca *D3*, é possível desenhar os elementos visuais pretendidos. O Excerto de Código 5.7 apresenta um exemplo simples da utilização desta biblioteca com a informação presente na variável "dataView" referida anteriormente.

```
1  //...
2
3  let indicatorsSVG: Selection<SVGElement> = d3.select(this.indicatorsDiv).append("svg");
4  indicatorsSVG.attr("width", "100%").attr("height", "100%");
5
6  for (let categorical of this.dataView.categorical.categories[0].values) {
7      let indicatorLane: Selection<SVGElement> = indicatorsSVG.append("rect");
8      let indicatorName: Selection<SVGElement> = indicatorsSVG.append("text");
9      indicatorName.text(<string>categorical);
10
11     //...
12 }
13
14 //...
```

Excerto de Código 5.7: Exemplo da Utilização da Biblioteca D3

Neste Excerto de Código 5.7, é apresentada uma simples implementação da biblioteca D3, onde se começa por desenhar cada quadrado representativo do indicador de desempenho, fazendo uso das propriedades "SVG" e "rect", resultando no visual da Figura 5.6 com o título "Information about KPI". A informação a ser obtida da variável "dataView" é relativamente ao nome de cada indicador, que é inserido, posteriormente, no visual (Excerto de Código 5.7, na linha de código 9).

5.5.3 Propriedades do Visual

O ficheiro *capabilities.json*, além do que foi referido na Secção 5.5.1, permite também definir um conjunto de propriedades que o visual pode apresentar, sendo o seu conteúdo de propriedades descrito em "objects" [71].

Neste visual criado, as propriedades que permite que sejam estabelecidas são relativas a definir o tamanho de cada quadrado do indicador de desempenho, assim como, a sua cor de fundo e definir um valor de limite mínimo e máximo para a sua barra de *score* (percentagem de valor a atingir já alcançado), que se encontra representada no final do indicador de desempenho.

Estes limites servem para enquadrar os indicadores de desempenho conforme o valor de *score* registado, permitindo que a cor deste varie conforme os limites estabelecidos. No caso atual, por predefinição, o limite mínimo é 60 e o limite máximo é 100.

Assim sendo, a cor da barra final de cada indicador de desempenho corresponde à seguinte informação:

- Verde - O valor do *score* é maior ou igual a 100%, o que indica que o registo do valor atual é maior ou igual que o valor a atingir.
- Amarelo - O valor do *score* está no intervalo compreendido entre o limite mínimo e limite máximo.
- Vermelho - O valor do *score* está abaixo do limite mínimo.
- Cinzento - O valor a atingir ainda não foi preenchido. Só se encontra registo do valor atual, logo, não existe um valor para o *score*.

A Figura 5.8 apresenta o visual "Information about KPI" com a definição do valor 90 para o limite mínimo e do valor 100 para o limite máximo do *score*.

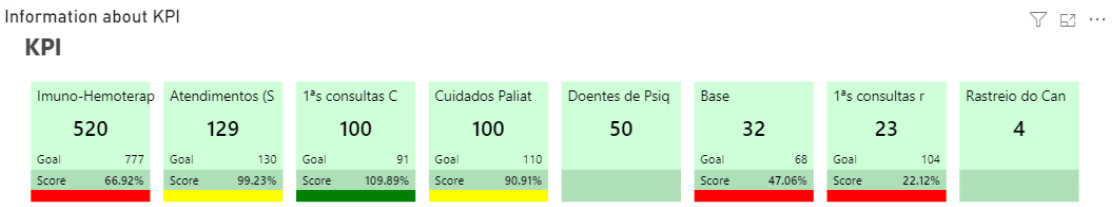


Figura 5.8: Visual com as Cores do Score Ajustadas aos Limites Definidos

Nesta Figura 5.8, a cor da barra de *score* dos indicadores de desempenho é substancialmente diferente daquela apresentada anteriormente no visual "Information about KPI", da Figura 5.6, para o mesmo conjunto de dados. Isto deveu-se à alteração dos valores dos limites mínimo e máximo, tal como referido anteriormente, o que levou à atualização de todos os elementos constituintes do visual.

5.5.4 Implantação no Power BI Desktop

Para que o visual criado possa ser utilizado na aplicação *Power BI Desktop*, é necessário realizar o seu *packaging* e distribuição.

Assim sendo, com a implementação do visual terminada, executa-se num terminal *PowerShell*, no caminho da raiz do projeto do visual personalizado, o comando "`pbiviz package`". Este comando dá origem a um ficheiro *.pbiviz* no diretório "`<nome do projeto>/dist`", representativo do visual previamente desenvolvido.

Posteriormente, este ficheiro *.pbiviz* é importado para a aplicação *Power BI Desktop*, possibilitando a sua utilização em relatórios representativos da informação sobre indicadores de desempenho, tal como ilustrado anteriormente (Figura 5.6).

Capítulo 6

Experimentação e Avaliação

Este capítulo tem como propósito definir os indicadores de avaliação, as suas hipóteses e a metodologia de avaliação a ser aplicada na solução implementada. De seguida, são atribuídos os métodos de avaliação aos indicadores considerados, finalizando o capítulo com a avaliação dos resultados conforme as hipóteses formuladas.

6.1 Indicadores de Avaliação

A definição de indicadores de avaliação tem como intuito ajudar a avaliar a solução implementada. Desta forma, identificam-se os seguintes indicadores de avaliação:

- Cumprimento dos Requisitos – Verificar se os requisitos inicialmente identificados foram alcançados e cumpridos na solução implementada.
- Satisfação dos *BI Developers* – Analisar se os *BI Developers*, que realizam as suas tarefas de BI com a aplicação *Power BI ScoreCards*, se sentem satisfeitos com a solução desenvolvida.

6.2 Hipóteses

A definição de hipóteses tem como propósito avaliar os indicadores anteriormente identificados. Assim sendo, foram estipuladas duas hipóteses:

1. Cumprimento dos requisitos da solução.
2. Satisfação dos *BI Developers* ser superior a 80%.

A primeira hipótese apresentada remete para o indicador relativo ao cumprimentos dos requisitos, ao qual se pretende confirmar que os requisitos estabelecidos foram alcançados.

Relativamente à segunda hipótese, esta prende-se com o indicador da satisfação dos *BI Developers*, pretendendo-se que a sua satisfação com a solução desenvolvida seja superior a 80%.

6.3 Metodologia de Avaliação

Esta secção tem como finalidade apresentar as metodologias de avaliação utilizadas, neste caso, são especificados testes de *software*, inquérito de satisfação e testes de hipóteses.

6.3.1 Testes de Software

Os testes de *software* permitem garantir que o código desenvolvido apresenta qualidade, a fim de evitar possíveis falhas.

No caso atual, os testes foram planeados seguindo o modelo *V-Model* [72], ilustrado na Figura 6.1. Este modelo estabelece uma associação entre cada fase dos testes com cada fase do processo de desenvolvimento.

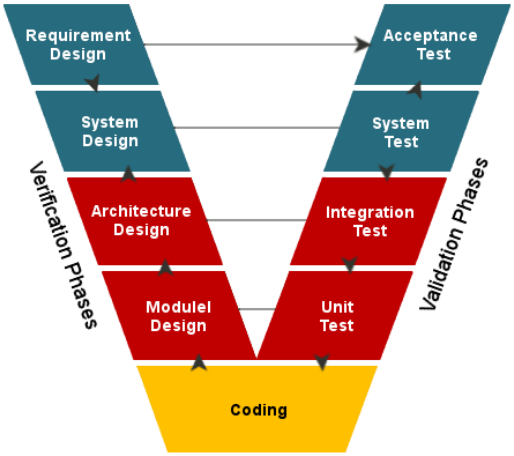


Figura 6.1: Arquitetura do Modelo V-Model [72]

Neste modelo *V-Model*, a fase dos testes é categorizada por Fase de Validação, ao passo que, a fase de desenvolvimento, é categorizada por Fase de Verificação.

Na Fase de Validação, apresentam-se quatro níveis de testes, sendo estes:

- Testes Unitários
- Testes de Integração
- Testes de Sistema
- Testes de Aceitação

Estes quatro níveis de testes são analisados na Secção 6.5.1.

6.3.2 Inquérito de Satisfação

O inquérito de satisfação visa avaliar a satisfação/concordância dos inquiridos com a solução desenvolvida, a fim de verificar se cumpre as suas necessidades. Este tipo de inquérito é utilizado a fim de avaliar indicadores de ordem subjetiva.

No contexto atual, é avaliada a satisfação/concordância dos *BI Developers* ao utilizar esta nova solução, e, para tal, é realizado um inquérito com a abordagem que é descrita seguidamente.

O inquérito de satisfação contém questões específicas e diretas, com a opção de resposta fechada, sobre a solução implementada. Cada questão é composta por cinco opções de resposta, utilizando o modelo *Likert Scale* [73], que consiste numa escala de 1 a 5. Estas opções representam:

1. Discordo Totalmente
2. Discordo
3. Neutro
4. Concordo
5. Concordo Totalmente

No final do inquérito, existe uma secção de comentários opcionais/sugestões de melhoria para os *BI Developers* preencherem, de modo a entender as suas opiniões sobre a solução desenvolvida, assim como, que partes da solução a melhorar.

6.3.3 Testes de Hipóteses

Para avaliar a solução final implementada de acordo com os indicadores de avaliação e hipóteses definidas, podem ser utilizados testes de hipóteses.

Um teste de hipóteses é um procedimento estatístico que verifica a informação de uma amostra. Para tal, averigua se a amostra está consistente com a hipótese da população sob estudo possuir determinadas características [74].

Num teste de hipóteses, definem-se dois tipos de hipóteses [74]:

- H_0: A hipótese nula que se julga rejeitar.
- H_1: A hipótese alternativa que se entende verosímil e que se pretende verificar.

Com base nos dados da amostra, o teste de hipóteses determina se a hipótese nula deve ser rejeitada. Para tal, é usado o valor de prova (*p-value)* que é o valor mínimo da significância que leva à rejeição de H_0 [75].

Assim sendo, se *p-value* <= α (nível de significância do teste), a hipótese H_0 é rejeitada. Nos testes a serem realizados, assume-se um nível de significância de 0.05, o que corresponde a um grau de confiança de 95% [75].

Os testes de hipóteses podem ser representados por testes paramétricos e testes não paramétricos. Os testes paramétricos especificam a distribuição de onde os dados são obtidos e a hipótese nula especifica um parâmetro dessa especificação, enquanto que, os testes não paramétricos, não assumem hipóteses sobre a distribuição de onde provêm os dados [76].

Assim sendo, os testes paramétricos devem ser aplicados caso se verifique a validade de certas condições, tais como, verificar se a amostra tem uma distribuição normal, especialmente, se a sua dimensão for inferior a trinta [77]. Por sua vez, os testes não paramétricos não necessitam de requisitos tão fortes, podendo ser aplicados quando a amostra tem uma

distribuição que não é normal, sendo também indicados de utilizar quando a amostra é pequena [77].

A vantagem dos testes paramétricos face aos testes não paramétricos é que possuem um poder estatístico maior, sendo mais capazes de levar a uma rejeição de H_0 [76]. Além disso, no caso de existir diferenças entre os dados, o teste paramétrico é mais provável das detetar [76].

6.4 Atribuição dos Métodos de Avaliação

Face aos indicadores de avaliação e hipóteses definidas anteriormente, torna-se relevante indicar a metodologia de avaliação escolhida.

Posto isto, na Tabela 6.1, identificam-se os métodos de avaliação utilizados para os indicadores de avaliação.

Tabela 6.1: Metodologia de Avaliação Aplicada

Indicador	**Metodologia de Avaliação**
Cumprimento dos Requisitos	Testes de *Software*
Satisfação dos BI *Developers*	Inquérito de Satisfação e Testes de Hipóteses

6.5 Avaliação de Resultados

Nesta secção procede-se à avaliação de resultados face às hipóteses definidas. Assim sendo, apresentam-se os resultados obtidos de cada método de avaliação efetuado, baseando a sua análise em função do indicador de avaliação e da hipótese formulada.

6.5.1 Testes de Software

Esta secção visa analisar os quatro níveis de testes: Testes Unitários, Testes de Integração, Testes de Sistema e Testes de Aceitação. Com isto, pretende-se avaliar a hipótese relativa ao cumprimento dos requisitos da solução.

Para cada nível de teste, é ilustrado um exemplo do mesmo realizado no projeto. Todos os testes realizados foram bem sucedidos, sendo todas as funcionalidades testadas, o que permite concluir que o cumprimento do requisitos foi alcançado. No entanto, as funcionalidades não foram testadas até à exaustão (por exemplo, não ser utilizado um grande volume de dados nos testes efetuados).

Testes Unitários

Os testes unitários têm como finalidade verificar se pequenas partes da solução (unidades) funcionam conforme esperado [78].

Na solução atual, para testar o Conetor desenvolvido, o contexto dos testes realizados seguem a definição definida na ferramenta *Power Query SDK*, em que cada teste é definido como um *Fact*, que contém um nome, um valor expectável e o valor atual [79].

O Excerto de Código 6.1 representa um exemplo de um teste unitário, na linguagem M, à tabela de navegação do Conetor desenvolvido, seguindo a especificação descrita anteriormente. Neste teste, pretende-se que a tabela de navegação retorne três tabelas.

```
1
2 navigationTable = ScoreCardsConnector_OAuth.Contents(),
3
4 nrTables = Table.RowCount(navigationTable)
5
6 //Fact(<Name of the Test>, <Expected Value>, <Actual Value>)
7 Fact("Three entries in navigation table", 3, nrTables),
```

Excerto de Código 6.1: Teste Unitário à Tabela de Navegação do Conetor

Os restantes testes efetuados ao Conetor foram delineados para abordar todas as funcionalidades descritas anteriormente na Secção 5.3. Contudo, não é possível fazer uso de uma ferramenta para averiguar a percentagem de cobertura de código implementado que foi testado, embora o propósito dos testes tenha isto em consideração.

Relativamente aos testes unitários ao visual personalizado criado, foram desenvolvidos testes para todos os métodos implementados, recorrendo à ferramenta *JavaScript Jasmine* [80]. Para se verificar a percentagem de cobertura por parte dos testes ao código implementado, utilizou-se a ferramenta *Karma* [81].

O Excerto de Código 6.2 mostra um exemplo de um teste unitário ao visual. Neste caso em concreto, pretende-se que o elemento raiz que alberga todos os indicadores de desempenho tenha sido devidamente criado. Os testes unitários efetuados testam todo o carregamento, processamento e tratamento de dados até ao seu efetivo desenho.

```
1  describe("Create KPIs", () => {
2      let visualBuilder: VisualBuilder;
3      let dataView: DataView;
4
5      beforeEach(() => {
6          visualBuilder = new VisualBuilder(230, 440);
7      });
8
9      it("root KPI DOM element is created", () => {
10         expect(visualBuilder.mainElement).toBeInDOM();
11     });
12 });
```

Excerto de Código 6.2: Teste Unitário à Criação do Elemento Principal do Visual

Conforme o Excerto de Código 6.2, o teste unitário ilustrado e todos os restantes testes seguem a abordagem apresentada. Para a criação deste teste, diversos métodos foram utilizados, sendo estes [80]:

- *describe* – Descreve um caso de teste. No âmbito da utilização do *Jasmine*, descreve um conjunto de *specs*.
- *beforeEach* – É sempre chamado antes de cada chamada do método *it*.
- *it* – Define um *spec* individual, podendo conter uma ou várias *expectations*.
- *expect* – Cria uma expressão também chamada de *expectation* para um *spec*. Assim, um *spec* é bem-sucedido caso todas as suas *expectations* não tenham falhas.

- *toBeInDOM()* – Exemplo de um método *Matcher*, podendo ser utilizados outros métodos disponibilizados pelo *Jasmine*. Os métodos *Matcher* têm como propósito efetuar uma comparação booleana entre o valor esperado e o valor a ser obtido.

Relativamente à cobertura de código implementado que os testes unitários ao visual abrangem, regista-se uma cobertura de 81%.

Testes de Integração

Os testes de integração têm como objetivo testar as interações entre os diferentes componentes, verificando o seu comportamento em conjunto [82]. Neste caso, será testado a interação entre o Conetor e a *API PowerBI ScoreCards*.

O Excerto de Código 6.3 representa um exemplo de um teste de integração, com a comunicação do Conetor à *API PowerBI ScoreCards*, mais especificamente à obtenção dos *scorecards* de um utilizador.

Será utilizada uma conta *Microsoft*, que servirá como auxílio na execução deste pedido, uma vez que a *API PowerBI ScoreCards* encontra-se protegida por *OAuth* com Azure AD.

```
1 scoreCards = ScoreCardsConnector_OAuth.getScoreCardsTable(),
2 expectedResult = [id = "1159", tenantId = "1", name = "Atividade Hospitalar - producao total",
      numberOfLevels = "0", nodeStructure = "["Scorecard","Area"]", nodeStructureConfig = [{"Name":"
      Scorecard","Color":"level1"},{"Name":"Area","Color":"level2"}], status = "0"],
3
4 Fact("First Scorecard of API PBI ScoreCards - Endpoint /scorecards", expectedResult, scoreCards{0}),
```

Excerto de Código 6.3: Teste de Integração entre o Conetor e API PowerBI ScoreCards

No caso do visual personalizado, não foram realizados testes de integração, uma vez que este não interage com nenhum componente externo. Concretamente, o visual representa os dados que foram previamente importados pelo Conetor para a aplicação *Power BI Desktop*, tendo só como tarefa desenhar os elementos visuais conforme a informação presente. Assim sendo, a parte relativa ao consumo de dados do visual e posterior desenho dos elementos visuais foi testado por via de testes unitários.

Testes de Sistema

Os testes de sistema visam testar o sistema como um todo, englobando os requisitos funcionais e requisitos não-funcionais [83]. A definição deste tipo de teste pode ser baseado em definição dos casos de uso.

No caso do projeto desenvolvido, os testes de sistema foram realizados a cada caso de uso, seguindo um conjunto de procedimentos e critérios. Em cada critério, é anotado se o mesmo passou ou não. No fim, é mostrado o resultado final do teste, bastando um critério falhar para o resultado final indicar que o teste não passou.

Na Tabela 6.2, encontra-se representado o teste de sistema para o caso de uso consultar relatório.

Tabela 6.2: Teste de Sistema para o Caso de Uso Consultar Relatório

Caso de Uso	UC2
Título	Consultar Relatório
Cenário	O relatório genérico criado previamente pelo *BI Developer* é consultado, para se visualizar a informação relativa aos indicadores de desempenho.
Procedimento	Consultar relatório genérico na aplicação *Power BI Desktop*. Efetuar a autenticação necessária para obter a informação. Visualizar a informação presente no relatório.
Critérios	1. O relatório genérico é consultado na aplicação *Power BI Desktop*. 2. É exigida a autenticação com conta *Microsoft* para obter a informação - Realizar Autenticação (UC2.1). 3. Os dados respetivos à conta autenticada são obtidos - Obter Dados Utilizador (UC2.2). 4. A informação é apresentada no relatório. 5. Existe interação com as várias visualizações presentes no relatório, aplicando os filtros de dados pretendidos, e a informação apresenta-se em conformidade com o que foi selecionado. 6. O visual personalizado criado mostra a informação conforme os filtros aplicados e as propriedades definidas no mesmo (alteração da cor de fundo, definir tamanho da barra de *score* e definir um valor limite mínimo e máximo para o *score*).
Resultados	1. Passou 2. Passou 3. Passou 4. Passou 5. Passou 6. Passou
Resultado Final	Passou

Pela análise da Tabela 6.2, face aos vários critérios especificados e aos resultados obtidos, conclui-se que o sistema é capaz de proceder à consulta do relatório, o que torna o caso de uso - Consultar Relatório (UC2) - possível de ser concretizado com sucesso.

No Anexo A, encontram-se os outros testes de sistema efetuados.

Testes de Aceitação

Os testes de aceitação têm como intuito verificar se o sistema desenvolvido cumpre com sucesso os requisitos definidos pelo cliente [84]. Desta forma, estes testes permitem aferir se o sistema satisfaz os critérios de satisfação, o que leva o cliente a aceitar ou não o sistema.

Este tipo de testes inclui, por exemplo, testes alfa e beta, testes de aceitação operacional e testes de aceitação do utilizador. No caso atual, serão realizados testes de aceitação do utilizador.

A Tabela 6.3 exemplifica um teste de aceitação relativo à consulta de relatório. Para determinar se um teste de aceitação é aceite, identificam-se um conjunto de critérios que são relativos aos requisitos do sistema, que devem ser cumpridos na sua totalidade.

Nos testes de aceitação realizados, assume-se algumas configurações prévias do sistema, como forma de simplificar a elaboração dos testes.

Tabela 6.3: Teste de Aceitação para Consultar Relatório

Cenário	Consultar Relatório
Descrição	O relatório é consultado na aplicação *Power BI Desktop*, para se proceder a uma análise da informação.
Critérios	1. É possível consultar o relatório na aplicação *Power BI Desktop*. 2. É possível realizar a autenticação com conta *Microsoft*. 3. É possível obter a informação da conta autenticada. 4. É possível representar a informação no relatório para a sua análise. 5. É possível interagir com a informação, aplicando diversos filtros e os dados apresentarem-se conforme o que foi selecionado.
Resultados	1. Cumpre 2. Cumpre 3. Cumpre 4. Cumpre 5. Cumpre
Resultado Final	Aceite

Pelo resultado final obtido na Tabela 6.3, considera-se que este teste de aceitação é aceite, devido a todos os critérios serem cumpridos.

No Anexo B, encontram-se os restantes testes de aceitação realizados.

6.5.2 Inquérito de Satisfação

O inquérito de satisfação é realizado a seis *BI Developers* que trabalham diretamente com os dados da aplicação *web PowerBI ScoreCards*. Como tal, pretende-se avaliar o nível de concordância de cada inquirido com a solução desenvolvida, que contrasta com a solução que utilizam – solução atual (Secção 4.1).

O inquérito é composto por nove questões de resposta fechada, existindo no final uma secção opcional de comentários adicionais/sugestões de melhoria. Este inquérito de satisfação encontra-se no Anexo C.

De modo a avaliar a satisfação dos *BI Developers*, à qual foi definida a hipótese da satisfação ser superior a 80%, cada questão do inquérito será avaliada de forma singular, tendo em consideração a referida hipótese. Para tal, utilizou-se testes de hipóteses. Nestes testes, assume-se um nível de significância de 0.05, com um grau de confiança de 95%. Uma vez analisadas todas as questões, tal como apresentado de seguida, pode-se afirmar que, em cada questão, a satisfação dos *BI Developers* é superior a 80%, logo, na generalidade, constata-se que todos os inquiridos apresentam uma satisfação de acordo com a hipótese formulada nesta solução desenvolvida.

O editor de desenvolvimento utilizado para realizar os testes de hipóteses é o *RSudio* [85], sendo o código escrito na linguagem R.

A Tabela 6.4 apresenta as respostas selecionadas pelos inquiridos a cada questão do inquérito de satisfação. A amostra para cada questão é independente, quantitativa e com uma dimensão de amostra de seis (número de inquiridos). Como tal, em cada questão, verificar-se-á se os dados da amostra seguem uma distribuição normal.

Tabela 6.4: Respostas Selecionadas pelos Inquiridos às Questões do Inquérito

Questão	**Resposta**				
	1	**2**	**3**	**4**	**5**
1	0	0	0	1	5
2	0	0	0	0	6
3	0	0	0	2	4
4	0	0	0	2	4
5	0	0	0	1	5
6	0	0	0	2	4
7	0	0	0	0	6
8	0	0	0	1	5
9	0	0	0	0	6

Seguidamente, são analisadas as questões que compõem o inquérito, face às respostas obtidas.

Questão 1 – A solução implementada integra automaticamente os dados da aplicação *PowerBI ScoreCards* com a ferramenta *Microsoft Power BI*?

Como a dimensão da amostra é de seis, vai-se ter que verificar se os dados da amostra seguem uma distribuição normal. Para isto, aplica-se um teste da normalidade de *Shapiro-Wilk* [74], sendo as hipóteses:

- H_0: Os dados da amostra seguem uma distribuição normal.
- H_1: Os dados da amostra não seguem uma distribuição normal.

O Excerto de Código 6.4 corresponde ao código em linguagem R, no editor *RSudio*, no qual é apresentado o teste de *Shapiro-Wilk*.

```
1 > q1<-c(5,5,5,5,5,4)
2 > shapiro.test(q1)
3
4   Shapiro-Wilk normality test
5
6 data:  q1
```

```
7 W = 0.49609, p-value = 2.073e-05
```

Excerto de Código 6.4: Teste de Shapiro-Wilk para a Questão 1

Conforme o Excerto de Código 6.4, a linha de código 1 apresenta o vetor "q1", que contém as opções de resposta escolhidas pelos inquiridos. De seguida, na linha de código 2, aplica-se o teste de *Shapiro-Wilk*, que dá origem ao valor do *p-value*, que é 2.073e-05.

Uma vez que o *p-value* é menor que o nível de significância (0.05), rejeita-se a hipótese H_0, o que se pode afirmar, com um grau de confiança de 95%, que a amostra não segue uma distribuição normal.

Posto isto, um teste paramétrico, tal como, um *t-test* [74] não pode ser utilizado, logo, será utilizado um teste não paramétrico, o teste de *Wilcoxon* [74], que é um teste que é equivalente ao *t-test*.

Tendo em consideração a hipótese da satisfação ser superior a 80%, formulam-se as seguintes hipóteses:

- H_0: Média das respostas é menor ou igual a 4.
- H_1: Média das respostas superior a 4.

Como a indicação das respostas obtidas é numa escala de 1-5, definiu-se que, o valor de 80% de satisfação, corresponde ao valor de 4 na escala de respostas.

No Excerto de Código 6.5, apresenta-se o teste de *Wilcoxon*, com o mesmo vetor de respostas do teste de *Shapiro-Wilk* anteriormente referido, e os resultados obtidos.

```
1 > wilcox.test(q1, mu = 4, alternative = "greater")
2
3   Wilcoxon signed rank test with continuity correction
4
5 data:  q1
6 V = 15, p-value = 0.01844
7 alternative hypothesis: true location is greater than 4
```

Excerto de Código 6.5: Teste de Wilcoxon para a Questão 1

Pelo Excerto de Código 6.5, o valor do *p-value* é 0.01844. Isto significa que é menor que 0.05, logo, rejeita-se a hipótese H_0. Assim sendo, afirma-se com um grau de confiança de 95% que a concordância dos *BI Developers* nesta questão é superior a 80%.

Questão 2 – A solução implementada garante a autenticação e autorização para obter os dados do respetivo cliente que está a aceder?

Nesta questão, todos os inquiridos escolheram a opção de resposta 5, o que significa que está presente uma distribuição discreta, com o valor de probabilidade 1. Desta forma, utilizando a hipótese referida da satisfação ser superior a 80% (valor 4 de satisfação, na escala 1-5), como todas as respostas são de valor 5, significa que a satisfação dos *BI Develepers* nesta questão é, sem dúvida, superior a 80%.

Questão 3 – A solução implementada agiliza e automatiza o processo desde a obtenção de dados à criação de relatórios?

Nesta questão, segue-se a lógica descrita na questão 1. Como se está perante uma dimensão de amostra pequena, verifica-se, primeiramente, se os dados da amostra seguem

uma distribuição normal. Posto isto, aplica-se um teste de *Shapiro-Wilk*, com as mesmas hipóteses já apresentadas na questão 1 relativamente à normalidade.

O código correspondente ao teste de *Shapiro-Wilk* é em todo semelhante ao apresentado no Excerto de Código 6.4, à exceção do vetor de opções de resposta, que agora é "q3<-c(5,4,5,5,5,4)".

Deste teste resulta que, o *p-value*, é de 0.001351, logo, é inferior ao nível de significância. Com isto, rejeita-se a hipótese H_0, afirmando-se com um grau de confiança de 95% que os dados não seguem uma distribuição normal.

Assim, tal como descrito na questão 1, aplica-se um teste de *Wilcoxon*, face à hipótese da satisfação ser superior a 80%, formulando duas hipóteses:

- H_0: Média das respostas é menor ou igual a 4.
- H_1: Média das respostas superior a 4.

O código aplicado para este teste é em todo semelhante ao Excerto de Código 6.5, mudando apenas o vetor de opções de resposta, que agora é "q3".

Como resultado do teste de *Wilcoxon*, o valor de *p-value* é 0.03593, significando que é inferior ao nível de significância. Como tal, rejeita-se a hipótese H_0, logo, com um grau de confiança de 95% conclui-se que a satisfação dos *BI Developers* nesta questão é superior a 80%.

Questão 4 – A solução desenvolvida, comparativamente à solução atual, é mais rápida no seu processo de desenvolvimento, resolvendo o problema de criação de relatórios repetitivos?

Nesta questão, obteve-se o seguinte vetor de respostas: "q4<-c(5,4,5,5,5,4)", significando que as respostas são idênticas à questão 3, analisada anteriormente. Como tal, a análise realizada na questão 3 pode ser aplicada nesta questão.

Conforme os valores obtidos na questão 3, e considerando a hipótese da satisfação ser superior a 80%, para esta questão 4, afirma-se, com um grau de confiança de 95%, que os *BI Developers* apresentam uma satisfação superior a 80%, o que traduz que esta solução desenvolvida resolve o problema de criação de relatórios repetitivos.

Questão 5 – O desenvolvimento do Conetor, aliado ao desenvolvimento de um relatório genérico, permite uma manutenção mais simplificada?

Para esta questão, obteve-se o vetor de respostas: "q5<-c(5,5,5,5,5,4)", sendo igual ao da questão 1. Assim sendo, considera-se para esta questão 5 a análise efetuada na questão 1.

Considerando a hipótese da satisfação ser superior a 80%, e face aos valores obtidos na questão 1, para esta questão 5, com um grau de confiança de 95% conclui-se que a satisfação dos *BI Developers* é superior a 80%.

Questão 6 – O relatório genérico/modelo é uma opção a ter em conta para poder ser utilizado por todos os clientes?

Nesta questão, o vetor de respostas é: "q6<-c(5,5,5,4,5,4)", sendo idêntico ao vetor de respostas da questão 3, portanto, considera-se a análise desta questão baseada nos valores da questão 3.

Assim sendo, com a hipótese da satisfação dos *BI Developers* ser superior a 80%, face ao valor obtido do *p-value* do teste de *Wilcoxon*, que é 0.03593, rejeita-se a hipótese H_0 (média das respostas é menor ou igual a 4), logo, com um grau de confiança de 95%, considera-se que a satisfação nesta questão é superior a 80%.

Questão 7 – O relatório genérico apresentado contém uma correta modelação de dados, ou seja, ao aplicar diferentes filtros nos dados, a informação nos visuais atualiza em conformidade?

Para esta questão, todos os inquiridos selecionaram a opção de resposta 5, significando que está presente uma distribuição discreta, com valor de probabilidade 1. Deste modo, a hipótese da satisfação ser superior a 80% (valor 4 de satisfação, na escala 1-5) é verificada, o que significa, sem dúvida, que a satisfação dos *BI Developers* nesta questão é superior a 80%.

Questão 8 – O visual personalizado criado com o título "Information About KPI" presente no relatório é intuitivo e adaptado para a representação dos indicadores de desempenho?

Nesta questão, o vetor de respostas é: "q8<-c(5,5,5,5,5,4)", idêntico ao vetor da questão 1.

Posto isto, com a análise e os valores obtidos da questão 1, conclui-se que a satisfação dos *BI Developers* é superior a 80% nesta questão.

Questão 9 – As propriedades existentes no visual personalizado criado com o título "Information About KPI" são possíveis de ser aplicadas (aplicar os temas disponíveis, mudar valores dos limites do *score*, definir tamanho do visual) e caracterizam-se como um complemento à variedade da representação da informação?

Nesta questão, todos os inquiridos escolheram a opção de resposta 5, portanto, está presente uma distribuição discreta com valor de probabilidade 1. Face à hipótese de satisfação ser superior a 80% (valor 4 de satisfação, na escala 1-5), esta é verificada nesta questão apresentada.

Feedback dos BI Developers

No final do inquérito de satisfação, existe uma secção de comentários opcionais/sugestões de melhoria. Posto isto, apresentam-se os comentários e sugestões obtidas:

- Esta solução apresentada é muito satisfatória e vai ajudar certamente a combater o processo repetitivo que atualmente se faz.
- A opção de desenvolver um conetor à medida das necessidades que se tem, com a ligação à API, suportando a sua autenticação, é uma excelente opção. O relatório genérico com esse conetor permitirá, com certeza, um processo mais rápido nas nossas tarefas.

- Esta ideia/processo apresentado está muito bem pensado e conseguido. No entanto, o relatório genérico precisa de ser melhorado, com visuais diferentes e mais informação dos *scorecards*. Por exemplo, apresentar os indicadores de desempenho por áreas. Mas a ideia está lá, o processo e solução estão definidos.
- Este processo tem tudo para correr bem. O conetor tem vantagens, uma delas é que não dá para saber onde se vai buscar os dados (não permite ver os pedidos no *Power BI Desktop*). Sugiro melhorias no relatório apresentado. Talvez mais composto em termos de visualizações. Representar os indicadores de desempenho por áreas. O visual criado para os indicadores de desempenho está bem conseguido, assim como, as suas propriedades a aplicar.
- A ideia de ter um relatório mais o conetor que obtém os dados da API e os representa no relatório segundo o cliente que faz "login" no conetor é excelente. Vai-se poupar trabalho com isto. É necessário ver este novo processo com calma e ir aplicando aos poucos com alguns clientes, para ver possíveis problemas que possam ocorrer.
- No conetor, não me parece haver muito a dizer. Talvez se possa acrescentar mais uma tabela, relativa às áreas. Contudo, também pode ser uma modulação de dados a fazer depois no *Power BI Desktop* (fazemos assim isso no processo que se usa). O relatório parece-me que falta apresentar a informação de forma mais variada, e faltam também as áreas dos indicadores. Está um relatório simples, mas tem conteúdo. Certamente que o relatório será melhorado. Em conclusão, este processo apresentado está bastante interessante.

Capítulo 7

Conclusão

Este capítulo apresenta a conclusão final da dissertação, descrevendo os objetivos alcançados, seguindo-se a explicitação das limitações encontradas e, por fim, encontra-se o trabalho futuro a desenvolver.

7.1 Objetivos Alcançados

De modo a aplicar o correto desenvolvimento da solução, analisou-se, primeiramente, o problema, com a definição dos seus objetivos. De seguida, definiram-se os requisitos funcionais e não funcionais, efetuando um desenho de soluções, seguido com a escolha da melhor solução. Posteriormente, a solução escolhida é implementada, com as boas práticas de desenvolvimento de *software*, culminando com a avaliação da solução seguindo uma metodologia definida.

Como tal, o propósito principal desta dissertação é a integração automática dos dados da aplicação *web PowerBI ScoreCards* com uma ferramenta de BI (*Microsoft Power BI*), de modo a desenvolver uma solução que seja mais automatizada e agilizada que a solução e processo atual. Para tal, conforme apresentado na Secção 1.3, são definidos dois objetivos:

- Desenvolver um mecanismo de integração automático dos dados da aplicação *web PowerBI ScoreCards* com uma ferramenta de BI para agilizar o processo de criação de relatórios.
- Criar um relatório genérico que sirva como modelo (*template*) para representar a informação de qualquer cliente.

O primeiro objetivo foi atingindo com o desenvolvimento de um Conetor para a aplicação *Power BI Desktop*. Como pretendido nos Requisitos Não Funcionais, o Conetor implementa a comunicação com uma API – *API PowerBI ScoreCards*, com o intuito de obter os dados relativos ao utilizador.

Como a *API PowerBI ScoreCards* está protegida por *OAuth* com *Azure* AD, implementou-se esta autenticação no Conetor. Assim, o Conetor permite a autenticação com o cliente que está a aceder no momento, sendo obtida a informação do mesmo pela API. Posteriormente, o Conetor organiza a informação, de modo a disponibilizá-la por tabelas, para serem acedidas na aplicação *Power BI Desktop*.

O segundo objetivo alcançado é relativo à criação de um relatório genérico, com uma estrutura e modelação previamente definida, que possibilite a sua utilização por qualquer cliente. O uso do Conetor desenvolvido juntamente com este relatório permite que, quando um

cliente consulte este relatório genérico, realize a devida autenticação com o Conetor e, no caso de a autenticação ser bem-sucedida, a informação é representada no relatório.

O relatório genérico visa acabar com o processo atual, em que, para cada cliente, é sempre construído um relatório, realizando os *BI Developers* a obtenção dos dados do cliente e sua representação no relatório. Este processo caracteriza-se por ser repetitivo e moroso.

A solução desenvolvida permite não só automatizar o processo existente, mas também simplificar o processo de manutenção do mesmo. No caso de existir, futuramente, algum problema com o cliente final na solução desenvolvida, os locais onde pode ocorrer algum erro é no Conetor ou no relatório genérico. Desta forma, ao resolver este problema do cliente, o mesmo é resolvido em todos os outros clientes, devido à uniformização do processo.

Complementarmente ao relatório genérico e aos visuais que o compõem, foi desenvolvido um visual personalizado para representar a informação dos indicadores de desempenho, sendo este um Requisitos Não Funcional. Este visual foi desenvolvido e incorporado no relatório genérico, mostrando os dados de cada indicador de desempenho relativo ao seu nome, valor atual, valor a atingir e percentagem de valor a atingir já alcançado.

Com a solução implementada, torna-se relevante efetuar a sua avaliação. Para tal, foram utilizados métodos de avaliação, tais como, testes de *software*, inquéritos de satisfação e testes de hipóteses.

Conforme as hipóteses consideradas no Capítulo 6, os testes de *software* tencionam analisar, em virtude da hipótese considerada, o cumprimento dos requisitos da solução. Com os quatro níveis de teste aplicados, todas as funcionalidades da solução foram testadas com sucesso, o que permite garantir que existiu um cumprimento dos requisitos. Contudo, deve-se aumentar ainda mais a carga de testes realizados para detetar alguma falha que possa existir e que ainda não foi encontrada.

Relativamente à hipótese da satisfação dos *BI Developers* ser superior a 80%, foi realizado um inquérito de satisfação. Posto isto, face às respostas obtidas, cada questão foi analisada por outro método de avaliação complementar, os testes de hipóteses. Assim sendo, em cada questão analisada, a satisfação foi superior a 80%, o que indica uma satisfação geral dos *BI Developers* superior a 80% com a solução implementada. No final, pelos comentários e sugestões de melhoria obtidos, regista-se uma satisfação geral de todos os inquiridos.

Conforme o supratranscrito, os objetivos desta dissertação foram alcançados, suportando a sua validação por diferentes metodologias de avaliação, o que permitir concluir, de forma positiva, que a solução implementada foi bem sucedida.

7.2 Limitações

Apesar da implementação ser bem sucedida e ter tido uma avaliação positiva, existem algumas limitações a apontar. Não obstante, estas limitações podem ser alvo de trabalho futuro.

Na ferramenta que permitiu desenvolver o Conetor com a comunicação à *API PowerBI ScoreCards*, os vários testes realizados testaram as funcionalidades implementadas. No entanto, a ferramenta não permite elaborar e desenvolver testes mais robustos, com a utilização, por exemplo, de um maior volume de dados ou aferir o tempo de resposta nos pedidos que são

executados aos *endpoints* da *API PowerBI Scorecards*. Em seu complemento, não permite o uso de outras ferramentas, tais como, de verificação de cobertura de código, para identificar qual o código que os testes delineados abrangem.

Relativamente ao relatório genérico criado, este representa os indicadores de desempenho de cada *scorecard*. Contudo, não contempla os indicadores de desempenho organizados por área, sendo algo necessário a melhorar para oferecer um maior detalhe na representação da informação. Esta limitação foi também indicada por alguns *BI Developers* nas sugestões de melhoria do inquérito de satisfação.

A avaliação efetuada permitiu obter a satisfação dos *BI Developers*, mas falta ainda analisar a avaliação e satisfação dos utilizadores finais (clientes). Isto não foi conseguido por a solução desenvolvida ainda não estar disponível para utilização real por parte dos utilizadores finais, mas uma vez que esteja efetivado o seu uso, vai-se verificar a sua satisfação, como forma de identificar problemas que possam surgir e/ou possíveis melhorias.

7.3 Trabalho Futuro

Face ao trabalho realizado e às limitações apontadas, existem tarefas a desenvolver para enriquecer mais a solução.

No Conetor desenvolvido, para este estar disponível para todos os utilizadores da aplicação *Power BI Desktop* (sejam ou não clientes da aplicação *web Power BI ScoreCards*), na parte de escolher a fonte de dados, é necessário o conetor passar pelo programa de Certificação de Conetores da *Microsoft*. Uma vez certificado, significa que o Conetor é certificado e distribuído pela *Microsoft*, sendo a manutenção e suporte assegurada pelos responsáveis do seu desenvolvimento. No entanto, até ao processo de certificação estar concluído, o Conetor pode ser utilizado pelos utilizadores finais, sendo fornecido a estes uma cópia do artefacto relativo ao mesmo, com as instruções de como o implantar no *Power BI Desktop* (Secção 5.3.3). Esta decisão de certificação é uma decisão de negócio que é da responsabilidade da empresa.

No contexto da *API PowerBI ScoreCards*, à qual o Conetor acede através dos seus *endpoints*, devem ser efetuados novos testes. Como tal, estes testes devem testar o desempenho dos *endpoints* (avaliar o tempo que demoram os pedidos a ser executados pelo Conetor) e verificar se os *endpoints* fornecem corretamente os dados conforme o utilizador em questão.

Relativamente ao relatório genérico criado, este deve ser aprimorado para oferecer mais detalhe na representação dos dados. Conforme apontando anteriormente nas limitações e baseado nas sugestões de melhoria dos *BI Developers*, a apresentação das áreas com os indicadores de desempenho respetivos é um ponto a melhorar, sendo que isto não foi conseguido porque não houve tempo no âmbito desta dissertação para ser realizado. Contudo, é possível de ser aplicado, servindo o relatório genérico criado (juntamente com o Conetor) uma representação e ponto de partida de como a solução desenvolvida é possível de ser implementada e melhorada.

O visual personalizado criado, presente no relatório genérico, pode ser certificado para estar disponível para todos os utilizadores na aplicação *Power BI Desktop*, semelhante ao que já foi referido anteriormente no Conetor. Apesar disto, pode ser utilizado na mesma pelos utilizadores até ser certificado ou não (decisão da empresa), sendo necessário fornecer o

ficheiro do visual a cada utilizador e as instruções de como ser importado para o *Power BI Desktop*, tal como referido na Secção 5.5.4.

Por último, é importante analisar a satisfação dos utilizadores finais com a solução desenvolvida, assim que esta esteja em ambiente de utilização real. Este ponto é importante para analisar como a solução está a comportar-se com a interação de vários utilizadores, de modo a analisar possíveis problemas reportados e/ou sugestões adicionais. De realçar que, qualquer adição de novas funcionalidades, assim como, mudança do *software* existente, deve seguir as boas práticas de desenvolvimento de *software*, para garantir uma implementação de novos requisitos e manutenção mais facilitada.

Referências Bibliográficas

[1] Efraim Turban, Ramesh Sharda e Dursun Delen. *Decision Support and Business Intelligence Systems*. 9th. USA: Prentice Hall Press, 2010.

[2] Carlo Vercellis. *Business Intelligence: Data Mining and Optimization for Decision Making*. Wiley Online Library, 2009.

[3] Longwen Zhao e Xiaohui Huang. «Research on the Application of Business Intelligence in Logistics Management». Em: *2009 International Conference on Management and Service Science*. IEEE. 2009, pp. 1–4.

[4] Celina Olszak e Ewa Ziemba. «Approach to Building and Implementing Business Intelligence Systems». Em: *Interdisciplinary Journal of Information, Knowledge, and Management* 2 (jan. de 2007), pp. 135–148.

[5] Petr Osadnik e Lenka Landryova. «Principles of Key Performance Indicators for Small and Medium Enterprise in European Union». Em: *2011 12th International Carpathian Control Conference (ICCC)*. IEEE. 2011, pp. 275–279.

[6] Gert HN Laursen e Jesper Thorlund. *Business Analytics For Managers: Taking Business Intelligence Beyond Reporting*. John Wiley & Sons, 2010.

[7] *What is Power BI? - Power BI | Microsoft Docs*. url: `https://docs.microsoft.com/en-us/power-bi/fundamentals/power-bi-overview` (acedido em 04/11/2019).

[8] H. P. Luhn. «A Business Intelligence System». Em: *IBM J. Res. Dev.* 2.4 (out. de 1958), pp. 314–319.

[9] Surajit Chaudhuri, Umeshwar Dayal e Vivek Narasayya. «An Overview of Business Intelligence Technology». Em: *Commun. ACM* 54 (ago. de 2011), pp. 88–98.

[10] *Dashboards vs. Reports: Similarities and Differences*. url: `https://chartio.com/blog/dashboards-vs-reports-how-theyre-the-same-how-theyre-different/` (acedido em 06/05/2020).

[11] Stephen Few. *Information Dashboard Design: The Effective Visual Communication of Data*. O'Reilly Media, Inc., jan. de 2006, pp. 26–27.

[12] *What is the difference between a dashboard and a report? - User Experience Stack Exchange*. url: `https://ux.stackexchange.com/questions/103146/what-is-the-difference-between-a-dashboard-and-a-report` (acedido em 06/05/2020).

[13] *Connect to Project Online with Power BI - Power BI | Power BI Report*. url: `https://docs.microsoft.com/en-us/power-bi/connect-data/service-connect-to-project-online` (acedido em 06/05/2020).

[14] *KPI Dashboard Software | KPI Dashboards Tools*. url: `https://www.zoho.com/analytics/kpi-dashboards.html` (acedido em 06/05/2020).

[15] P Vassiliadis et al. «A Methodology for the Conceptual Modeling of ETL Processes». Em: *CAiSE Workshop on Decision Systems Engineering*. 2003.

[16] Jiawei Han, Micheline Kamber e Jian Pei. *Data Mining: Concepts and Techniques*. 3ª ed. Elsevier, 2011, pp. 125–136.

[17] William H. Inmon. *Building the Data Warehouse Third Edition*. 3ª ed. 2003, p. 31.

[18] *Magic Quadrant Research Methodology*. url: `https://www.gartner.com/en/research/methodologies/magic-quadrants-research` (acedido em 02/04/2020).

[19] *Magic Quadrant for Analytics and Business Intelligence Platforms*. url: https://www.gartner.com/doc/reprints?id=1-3TXXSLV&ct=170221&st=sb&ocid=mkto_eml_EM526600A1LA1 (acedido em 02/04/2020).

[20] *What is the Power BI service? - Power BI*. url: https://docs.microsoft.com/en-us/power-bi/power-bi-service-overview (acedido em 04/11/2019).

[21] *Power Query SDK - Visual Studio Marketplace*. url: https://marketplace.visualstudio.com/items?itemName=Dakahn.PowerQuerySDK (acedido em 04/11/2019).

[22] *Start developing custom connectors for Power Query | Microsoft Docs*. url: https://docs.microsoft.com/en-us/power-query/startingtodevelopcustomconnectors (acedido em 04/11/2019).

[23] *Handling authentication for Power Query connectors | Microsoft Docs*. url: https://docs.microsoft.com/en-us/power-query/handlingauthentication (acedido em 04/11/2019).

[24] *Power BI Custom Visuals*. url: https://powerbi.microsoft.com/en-us/developers/custom-visualization/ (acedido em 04/11/2019).

[25] *Business Intelligence and Analytics Software - Tableau*. url: https://www.tableau.com/ (acedido em 01/12/2019).

[26] *Tableau Products | Types of Tableau Products*. url: https://www.gangboard.com/blog/tableau-products/ (acedido em 01/12/2019).

[27] *Connector Plugins Built with the Tableau Connector SDK - Tableau*. url: https://help.tableau.com/current/pro/desktop/en-us/examples_connector_sdk.htm (acedido em 01/12/2019).

[28] *Qlik Business Intelligence: Data Analytics & Data Integration | Qlik*. url: https://www.qlik.com/us (acedido em 01/12/2019).

[29] *Qlik Sense Products*. url: https://help.qlik.com/en-US/sense/November2019/Subsystems/Hub/Content/Sense_Hub/Introduction/qlik-sense-product-family.htm (acedido em 01/12/2019).

[30] *QVX custom connector*. url: https://help.qlik.com/en-US/sense-developer/November2019/Subsystems/QVXSDKAPI/Content/Sense_QVXSDKAPI/Qlik-View-QVXFileFormat/QVX-and-QlikView-custom-connector.htm (acedido em 01/12/2019).

[31] *Getting started building widgets - QlikSense*. url: https://help.qlik.com/en-US/sense-developer/November2019/Subsystems/Extensions/Content/Sense_Extensions/widgets-getting-started.htm (acedido em 01/12/2019).

[32] *Data Analytics, Data Visualization & Reporting | ThoughtSpot*. url: https://www.thoughtspot.com/product (acedido em 01/12/2019).

[33] *Mobile | ThoughtSpot*. url: https://www.thoughtspot.com/mobile (acedido em 01/12/2019).

[34] Robert Kaplan e Norton. «The Balanced Scorecard: Measures That Drive Performance». Em: *Harvard Business Review* 83 (jul. de 2005), p. 172.

[35] Robert S. Kaplan. «Conceptual Foundations of the Balanced Scorecard». Em: *Handbooks of Management Accounting Research* 3 (2009), pp. 1253–1269.

[36] *What is a Key Performance Indicator (KPI)?* url: https://kpi.org/KPI-Basics (acedido em 03/11/2019).

[37] P. Koen et al. «Providing Clarity and a Common Language to the "Fuzzy Front End"». Em: *Research Technology Management* 44.2 (2001), pp. 46–55.

[38] P. Kotler e K. L. Keller. *Creating Customer Value, Satisfaction, and Loyalty*. Pearson Education, 2009.

[39] Tony Woodall. «Conceptualising 'Value for the Customer': An Attributional, Structural and Dispositional Analysis». Em: *Academy of Marketing Science Review* 12.1 (2003), pp. 1–42.

[40] *Value Proposition Canvas – Download the Official Template*. url: `https://www.strategyzer.com/canvas/value-proposition-canvas` (acedido em 08/02/2020).

[41] Alexander Osterwalder. «The Business Model Ontology - A Proposition In A Design Science Approach». Tese de doutoramento. Université de Lausanne, Faculté des Hautes Études Commerciales, 2004.

[42] Alexander Osterwalder e Yves Pigneur. *Business Model Generation: A Handbook for Visionaries, Game Changers, and Challengers*. John Wiley & Sons, 2010.

[43] Thomas Saaty. *The Analytic Hierarchy Process: Planning, Priority Setting, Resource Allocation*. Jan. de 1980.

[44] Thomas Saaty. «Decision Making with the Analytic Hierarchy Process». Em: *Int. J. Services Sciences Int. J. Services Sciences* 1 (jan. de 2008), pp. 83–98.

[45] Suzanne Robertson e James Robertson. *Mastering the Requirements Process: Getting Requirements Right*. Addison-wesley, 2012.

[46] Lawrence Chung e Julio Leite. «On Non-Functional Requirements in Software Engineering». Em: *Conceptual Modeling: Foundations and Applications*. Springer, 2009, pp. 363–379.

[47] *Capturing Architectural Requirements*. url: `https://www.ibm.com/developerworks/rational/library/4706.html` (acedido em 05/04/2020).

[48] *What is REST*. url: `https://restfulapi.net/` (acedido em 01/02/2020).

[49] *What is an API Endpoint? | API Endpoint Definition | RapidAPI*. url: `https://rapidapi.com/blog/api-glossary/endpoint/` (acedido em 10/04/2020).

[50] *Portal do Microsoft Azure | Microsoft Azure*. url: `https://azure.microsoft.com/pt-pt/features/azure-portal/` (acedido em 13/02/2020).

[51] *API authentication and authorization*. url: `https://idratherbewriting.com/learnapidoc/docapis_more_about_authorization.html` (acedido em 23/01/2020).

[52] *What is Azure Active Directory? - Azure Active Directory | Microsoft Docs*. url: `https://docs.microsoft.com/en-us/azure/active-directory/fundamentals/active-directory-whatis` (acedido em 11/02/2020).

[53] *Unable to source data from web api secured by azure ad (OAuth 2) - Microsoft Power BI Community*. url: `https://community.powerbi.com/t5/Power-Query/unable-to-source-data-from-web-api-secured-by-azure-ad-oauth2/td-p/101990` (acedido em 06/02/2020).

[54] *Visual Studio IDE, Code Editor, Azure DevOps, & App Center - Visual Studio*. url: `https://visualstudio.microsoft.com/` (acedido em 22/05/2020).

[55] *Visual Studio Code - Code Editing. Redefined*. url: `https://code.visualstudio.com/` (acedido em 22/05/2020).

[56] *Developing a Power BI visual - Power BI | Microsoft Docs*. url: `https://docs.microsoft.com/en-us/power-bi/developer/visuals/custom-visual-develop-tutorial` (acedido em 22/05/2020).

[57] *TypeScript - JavaScript that Scales.* url: `https://www.typescriptlang.org/` (acedido em 19/05/2020).

[58] *D3.js Visual*. url: `https://appsource.microsoft.com/en-us/product/power-bi-visuals/WA104381354` (acedido em 22/05/2020).

[59] *Authentication and Authorization*. url: `https://auth0.com/docs/authorization/concepts/authz-and-authn` (acedido em 20/05/2020).

[60] *OAuth 2.0 Authorization Framework*. url: https://auth0.com/docs/protocols/oauth2 (acedido em 20/05/2020).
[61] *Authorization Code Flow*. url: https://auth0.com/docs/protocols/oauth2 (acedido em 20/05/2020).
[62] *The OAuth 2.0 Authorization Framework*. url: https://tools.ietf.org/html/rfc6749#section-4.1 (acedido em 20/05/2020).
[63] *Understanding Refresh Tokens*. url: https://auth0.com/learn/refresh-tokens/ (acedido em 20/05/2020).
[64] *Understand the OAuth 2.0 Authorization Code Flow in Azure AD | Microsoft Docs*. url: https://docs.microsoft.com/en-us/azure/active-directory/azuread-dev/v1-protocols-oauth-code (acedido em 20/05/2020).
[65] *Web.Contents - PowerQuery M | Microsoft Docs*. url: https://docs.microsoft.com/en-us/powerquery-m/web-contents (acedido em 22/05/2020).
[66] *Handling Navigation for Power Query Connectors | Microsoft Docs*. url: https://docs.microsoft.com/en-us/power-query/handlingnavigationtables (acedido em 22/05/2020).
[67] *Helper Functions for M Extensions for Power Query Connectors | Microsoft Docs*. url: https://docs.microsoft.com/en-us/power-query/helperfunctions#tabletonavigationtable (acedido em 22/05/2020).
[68] *Power BI Templates*. url: https://www.powerbitutorial.org/tutorials/power-bi-templates/ (acedido em 21/05/2020).
[69] *Create and Manage Relationships in Power BI Desktop - Power BI | Microsoft Docs*. url: https://docs.microsoft.com/en-us/power-bi/transform-model/desktop-create-and-manage-relationships (acedido em 22/05/2020).
[70] *What is PowerShell? - PowerShell | Microsoft Docs*. url: https://docs.microsoft.com/en-us/powershell/scripting/overview?view=powershell-7 (acedido em 22/05/2020).
[71] *Capabilities and properties of Power BI visuals - Power BI | Microsoft Docs*. url: https://docs.microsoft.com/en-us/power-bi/developer/visuals/capabilities (acedido em 22/05/2020).
[72] *V Model SDLC: Verification and Validation Model*. url: https://www.professionalqa.com/v-model (acedido em 14/06/2020).
[73] I. Elaine Allen e Christopher A. Seaman. «Likert Scales and Data Analyses». Em: *Quality progress* 40.7 (2007), pp. 64–65.
[74] Joaquim P. Marques De Sá. *Applied Statistics using SPSS, Statistica, MatLab and R*. Springer Science & Business Media, 2007.
[75] Douglas C. Montgomery. *Design and Analysis of Experiments*. John Wiley & Sons, 2017.
[76] Douglas C. Montgomery e George C. Runger. *Applied Statistics and Probability for Engineers*. John Wiley & Sons, 2010.
[77] Craig S. Wells e John M. Hintze. «Dealing with Assumptions Underlying Statistical Tests». Em: *Psychology in the Schools* 44.5 (2007), pp. 495–502.
[78] *Unit Testing: Tutorial, Types, Frameworks,Process,Techniques*. url: https://www.professionalqa.com/unit-testing (acedido em 16/06/2020).
[79] *Handling Unit Testing for Power Query Connectors | Microsoft Docs*. url: https://docs.microsoft.com/en-us/power-query/handlingunittesting (acedido em 16/06/2020).
[80] *Jasmine Documentation*. url: https://jasmine.github.io/ (acedido em 20/06/2020).

[81] *Karma*. url: `https://karma-runner.github.io/5.0/intro/how-it-works.html` (acedido em 20/06/2020).

[82] *What is an Integration Testing?* url: `https://www.professionalqa.com/integration-testing` (acedido em 16/06/2020).

[83] *System Testing : Complete Guide (Types,Process)*. url: `https://www.professionalqa.com/system-testing` (acedido em 16/06/2020).

[84] *What is User Acceptance Testing (UAT)?* url: `https://www.professionalqa.com/user-acceptance-testing` (acedido em 16/06/2020).

[85] *RStudio | Open Source & Professional Software for Data Science Teams - RStudio*. url: `https://rstudio.com/` (acedido em 17/06/2020).

Anexo A

Testes de Sistema

Neste Anexo, encontram-se apresentados os restantes testes de sistema realizados para o projeto. Neste caso, os testes de sistemas estão apresentados nas Tabelas A.1 e A.2.

Tabela A.1: Teste de Sistema para o Caso de Uso Criar Relatório Genérico

Caso de Uso	UC1
Título	Criar Relatório Genérico
Cenário	O relatório genérico criado funciona como modelo (*template*) para ser utilizado por todos os utilizadores, partilhando estes a mesma estrutura de visuais estabelecida.
Procedimento	É elaborado um relatório genérico baseado nos dados fornecidos pelo Conetor. Efetua-se a correta estruturação e modelação dos dados no relatório, assim como, as visualizações que deve apresentar.
Critérios	1. Obter as tabelas retornadas pelo Conetor, de modo a organizar os dados. 2. Estruturação e modelação dos dados. 3. Escolha das visualizações para representar os dados. 4. Conformidade dos dados entre os vários visuais face aos filtros aplicados. 5. Guardar relatório genérico com a extensão *.pbit*.
Resultados	1. Passou 2. Passou 3. Passou 4. Passou 5. Passou
Resultado Final	Passou

Tabela A.2: Teste de Sistema para o Caso de Uso Publicar Relatório Online

Caso de Uso	UC3
Título	Publicar Relatório *Online*
Cenário	O relatório é publicado com sucesso na aplicação *Power BI Service* (aplicação *online*), sendo partilhado com outros utilizadores e/ou criado um *dashboard*.
Procedimento	O relatório é publicado na aplicação *Power BI Service*. É partilhado com outros utilizadores. É criado um *dashboard* para representar a informação.
Critérios	1. O relatório é publicado na aplicação *Power BI Service*. 2. O relatório é partilhado com outros utilizadores. 3. É criado um *dashboard* com base na informação presente no relatório.
Resultados	1. Passou 2. Passou 3. Passou
Resultado Final	Passou

Anexo B

Testes de Aceitação

Neste Anexo, encontram-se representados os restantes testes de aceitação realizados ao sistema. Estes testes efetuados são testes de aceitação do utilizador. No teste da Tabela B.1, o teste é relativo ao utilizador com papel de *BI Developer*, enquanto que, os restantes testes, ilustrados nas B.2, B.3 e B.4 são relativos ao utilizador com papel de utilizador final (cliente).

Tabela B.1: Teste de Aceitação para Criar Relatório Genérico

Cenário	Criar Relatório Genérico
Descrição	O relatório genérico deve ser criado, funcionando como modelo (*template*), contendo uma estrutura padrão de modelação de dados, tal como, de visuais representativos da informação.
Critérios	1. É possível, primeiramente, obter as tabelas retornadas pelo Conetor. 2. É possível modelar e estruturar os dados. 3. É possível apresentar os dados pelos visuais escolhidos. 4. É possível aplicar filtros aos dados e a informação apresentar-se em conformidade. 5. É possível guardar o relatório genérico na extensão *.pbit*, de modo a que o relatório seja um relatório genérico a ser utilizado por qualquer utilizador.
Resultados	1. Cumpre 2. Cumpre 3. Cumpre 4. Cumpre 5. Cumpre
Resultado Final	Aceite

Tabela B.2: Teste de Aceitação para Realizar Autenticação

Cenário	Realizar Autenticação
Descrição	A autenticação no Conetor é possível de ser realizada por meio de uma conta *Microsoft* com a *API PowerBI ScoreCards*, aquando a consulta do relatório na aplicação *Power BI Desktop*.
Critérios	1. É possível introduzir as credenciais de autenticação (conta *Microsoft*) no Conetor para autenticar-se com a *API PowerBI ScoreCards*. 2. É possível verificar o sucesso da realização da autenticação.
Resultados	1. Cumpre 2. Cumpre
Resultado Final	Aceite

Tabela B.3: Teste de Aceitação para Visual Personalizado Criado

Cenário	Verificar Visual Personalizado
Descrição	O visual personalizado criado para a aplicação *Power BI Desktop* e que está presente no relatório deve mostrar a informação conforme o que foi selecionado (atuar em conformidade face aos filtros aplicados), e aplicar as suas propriedades escolhidas.
Critérios	1. É possível apresentar a informação dos indicadores de desempenho conforme o filtro do ano e/ou mês selecionado. 2. É possível realçar no visual o indicador de desempenho que foi selecionado noutro gráfico (aplicação de filtros entre diferentes visuais). 3. É possível definir o tamanho de cada quadrado representativo do indicador de desempenho. 4. É possível definir a cor de fundo para o visual (escolher um tema dos vários disponíveis). 5. É possível definir o tamanho da barra de *score* do visual. 6. É possível definir um valor de limite mínimo e máximo para o *score* do indicador de desempenho, e a cor da barra de cada indicador atuar em conformidade face aos limites definidos.
Resultados	1. Cumpre 2. Cumpre 3. Cumpre 4. Cumpre 5. Cumpre 6. Cumpre
Resultado Final	Aceite

Tabela B.4: Teste de Aceitação para Publicar Relatório

Cenário	Publicar Relatório
Descrição	O relatório é publicado na aplicação *online Power BI Service*, sendo partilhado com outros utilizadores e ser criado um *dashboard*.
Critérios	1. É possível publicar o relatório *online* na aplicação *Power BI Service*. 2. É possível partilhar o relatório com outros utilizadores. 3. É possível criar um *dashboard* para o relatório publicado.
Resultados	1. Cumpre 2. Cumpre 3. Cumpre
Resultado Final	Aceite

Anexo C

Inquérito de Satisfação

Este inquérito de satisfação tem como finalidade avaliar a concordância/satisfação dos *BI Developers* com a solução desenvolvida.

Questão	**Resposta**				
	1	**2**	**3**	**4**	**5**
1 - A solução implementada integra automaticamente os dados da aplicação *PowerBI ScoreCards* com a ferramenta *Microsoft Power BI*?					
2 - A solução implementada garante a autenticação e autorização para obter os dados do respetivo cliente que está a aceder?					
3 - A solução implementada agiliza e automatiza o processo desde a obtenção de dados à criação de relatórios?					
4 - A solução desenvolvida, comparativamente à solução atual, é mais rápida no seu processo de desenvolvimento, resolvendo o problema de criação de relatórios repetitivos?					
5 - O desenvolvimento do Conetor, aliado ao desenvolvimento de um relatório genérico, permite uma manutenção mais simplificada?					
6 - O relatório genérico/modelo é uma opção a ter em conta para poder ser utilizado por todos os clientes?					
7 - O relatório genérico apresentado contém uma correta modelação de dados, ou seja, ao aplicar diferentes filtros nos dados, a informação nos visuais atualiza em conformidade?					
8 - O visual personalizado criado com o título "Information About KPI" presente no relatório é intuitivo e adaptado para a representação dos indicadores de desempenho?					
9 - As propriedades existentes no visual personalizado criado com o título "Information About KPI" são possíveis de ser aplicadas (aplicar os temas disponíveis, mudar valores dos limites do *score*, definir tamanho do visual) e caracterizam-se como um complemento à variedade da representação da informação?					

Legenda: 1. Discordo Totalmente | 2. Discordo | 3. Neutro | 4. Concordo | 5. Concordo Totalmente

Comentários adicionais/Sugestões de Melhoria:

_____

_____