FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Análise e integração de dados para gestão de informação no contexto das Autarquias Locais

Rodrigo Campos Reis

Mestrado em Engenharia Informática e Computação

Orientador: Professor Doutor Gabriel de Sousa Torcato David

12 de julho de 2024

Análise e integração de dados para gestão de informação no contexto das Autarquias Locais

Rodrigo Campos Reis

Mestrado em Engenharia Informática e Computação

Aprovado em provas públicas pelo Júri:

Presidente: Sérgio Sobral Nunes

Arguente: Orlando Manuel Oliveira Belo

Vogal: Gabriel de Sousa Torcato David

12 de julho de 2024

Resumo

No contexto das Autarquias Locais, a necessidade de ferramentas confiáveis de apoio à decisão é crucial para os decisores e gestores públicos. Estes profissionais altamente escrutinados enfrentam o desafio de obter respostas precisas e rápidas sobre o desempenho global da organização, dos diversos departamentos e dos seus colaboradores. No entanto, os softwares disponíveis muitas vezes não conseguem atender diretamente a essas necessidades ou fazem-no a um custo elevado. Além disso, o crescente volume de informação, muitas vezes disperso, dificulta o cálculo das métricas necessárias para uma análise adequada dos indicadores relevantes.

Para abordar este problema, propõe-se a criação de um motor de *Business Intelligence* (BI) projetado para as Autarquias Locais. Este motor deve extrair dados dos diversos softwares utilizados pelos Municípios, processá-los de maneira eficiente e apresentá-los em formatos acessíveis como *dashboards*, gráficos e listagens. Essa solução permitirá aos decisores e gestores públicos visualizar de forma clara e integrada as informações cruciais para a tomada de decisões.

Ao disponibilizar dados processados em formatos visuais e facilmente interpretáveis, o motor de BI proposto visa aumentar a eficiência e a eficácia na gestão da informação dentro das Autarquias Locais. Com essa abordagem, os gestores poderão monitorizar o desempenho organizacional de forma mais precisa, identificar áreas de melhoria e tomar decisões informadas que promovam a eficiência operacional e o melhor atendimento às necessidades da comunidade. Em última análise, esta dissertação busca transformar a maneira como as Autarquias Locais gerem e utilizam a informação produzida, contribuindo para uma gestão pública mais transparente, eficaz e orientada por dados.

Abstract

In the context of Local Authorities, the need for reliable decision support tools is crucial for policymakers and public managers. These highly scrutinized professionals face the challenge of obtaining precise and prompt answers regarding the overall performance of the organization, its various departments, and its employees. However, the available software often fails to meet these needs directly or does so at a high cost. Moreover, the increasing volume of information, frequently scattered, complicates the calculation of necessary metrics for a thorough analysis of relevant indicators.

To address this issue, the creation of a Business Intelligence (BI) engine tailored for Local Authorities is proposed. This engine is designed to extract data from the diverse software used by Municipalities, process it efficiently, and present it in accessible formats such as dashboards, graphs, and listings. This solution will enable policymakers and public managers to clearly and comprehensively visualize crucial information for decision-making.

By providing processed data in visual and easily interpretable formats, the proposed BI engine aims to enhance efficiency and effectiveness in information management within Local Authorities. This approach will allow managers to monitor organizational performance more accurately, identify areas for improvement, and make informed decisions that enhance operational efficiency and better serve community needs. Ultimately, this dissertation aims to transform how Local Authorities manage and utilize produced information, contributing to more transparent, effective, and data-driven public administration.

Agradecimentos

Ao Professor Doutor Gabriel David, meu orientador, pela orientação e pelo acompanhamento de todas as etapas desta dissertação.

À ANO Software, a todos os colaboradores e colegas. Em particular, ao Eng. Manuel Amorim (CEO) e à Dra. Teresa Pacheco (Diretora Departamento de Software e Serviços) pela oportunidade de elaborar esta dissertação e integrá-la nas soluções da ANO. Ao Eng. João Pedro Oliveira (Diretor do Departamento de Engenharia) e ao Eng. Sérgio Almeida (*Team Leader*) por toda a orientação e conselhos no desenvolvimento técnico da solução proposta. À Drª. Isabel Ribeiro por algumas dicas e perspetivas sobre a solução. Ao meu colega Luís Ferreira que dará continuidade a esta dissertação.

À minha família que me apoiou e me incentivou a concluir com determinação esta dissertação.

À minha namorada, Helena, por todo o apoio incondicional.

Rodrigo Campos Reis

"O que impede de saber não são nem o tempo nem a inteligência,
mas somente a falta de curiosidade."

Agostinho da Silva

Conteúdo

Lista de Figuras

Lista de Tabelas

Abreviaturas e Símbolos

BI	*Business Intelligence*
TI	Tecnologias da Informação
ERP	*Enterprise Resource Planning*
OS	*Operational Systems*
DW	*Data Warehouse*
ETL	*Extract, Transform and Load*
OLAP	*Online Analytical Processing*
KPIs	*Key Performance Indicators*
SQL	*Structured Query Language*
API	*Application Programming Interface*
REST	*Representational State Transfer*
HTTP	*Hypertext Transfer Protocol*

Capítulo 1

Introdução

No contexto das Autarquias Locais, os decisores e os gestores públicos necessitam de ferramentas fiáveis de apoio à decisão, capazes de responder às principais questões relacionadas com a performance da organização como um todo, dos diferentes departamentos e dos seus colaboradores.

No entanto, os *softwares*, muitas vezes, não estão preparados para responder diretamente às questões dos decisores, ou apenas o poderão fazer com elevado custo. Com o volume de informação crescente e, por vezes, disperso, torna-se impraticável o cálculo das métricas necessárias para uma análise adequada dos indicadores pretendidos.

Assim, nesta dissertação, intitulada "Análise e integração de dados para gestão de informação no contexto das Autarquias Locais", é proposta uma solução capaz de responder a esta necessidade das Autarquias Locais, isto é, a criação de um motor de *Business Intelligence* (BI), orientado para extrair a informação criada pelos *softwares* utilizados pelos Municípios, processá-la e posteriormente disponibilizá-la no formato de *dashboards*, gráficos e listagens.

1.1 Enquadramento

Uma sociedade, onde é notório o avanço tecnológico e a constante digitalização da informação, impôs aos diferentes quadrantes da sociedade, em particular às suas organizações, a aquisição de *softwares* capazes de gerir os fluxos de trabalho, produzir e organizar documentos e otimizar os seus recursos.

O mesmo sucedeu nas Autarquias em Portugal. Surgiu a necessidade da desmaterialização dos processos e dos fluxos de trabalho, o que conduziu à procura de *softwares* de Gestão Processual e Documental, orientados para estruturar processos, documentos e informação inerentes às competências e ao funcionamento de um Município.

Algumas empresas de Tecnologia de Informação (TI), em Portugal, começaram a explorar essas necessidades do mercado, tendo desenvolvido soluções à medida, nomeadamente motores de *workflow* estruturado e não estruturado, capazes de informatizar as etapas de um processo,

que outrora se consolidavam num documento físico, tornando-o agora um conjunto de passos orquestrados em ambiente digital.

Para além de *sofwares* de Gestão Processual e Documental, nesse processo de digitalização, os municípios têm vindo a investir em produtos destinados a otimizar os seus recursos. São alguns exemplos: *softwares* de *Enterprise Resource Planning* (ERP), *softwares* de Serviços de Atendimento ao Público, *sofwares* de Gestão de Reuniões e Atas e *sofwares* de Partilha de Documentos.

Estes produtos permitiram assim a redução de custos nas autarquias, associados aos processos administrativos, nomeadamente a redução do custo das impressões e cópias de documentos e das necessidades de espaço de arquivo. Permitiram também localizar e disponibilizar imediatamente processos ou documentos, quando necessário e em qualquer parte.

Mais recentemente com a consolidação do acesso à *internet* e com o aparecimento dos *smartphones*, foi necessária a agilização da comunicação entre os munícipes e as Autarquias, surgindo no mercado soluções digitais de Atendimento aos Munícipes.

No entanto, com o processo de digitalização nas Autarquias locais, a informação dispersa pelos diferentes *softwares* necessita agora de ser consolidada e interpretada pelos gestores públicos. Ou seja, muitas vezes, com apenas os *softwares* de operação diária dos municípios não é possível interpretar a informação que caracterize a *performance* da organização, isto é obter dados de produtividade anuais, por caracterização dos processos, por departamento e até por utilizador.

É nesse sentido que esta dissertação se propõe a explorar e a desenvolver um motor de BI que permita processar os dados produzidos pelos *softwares* das Autarquias Locais e disponibilizá-los aos gestores de topo (Presidentes de Câmara, Vereadores, Diretores Municipais, Diretores de Departamento, etc), através de *dashboards*, gráficos e listagens.

Ao ser concretizada, esta dissertação representa um avanço significativo na otimização da tomada de decisões e na gestão estratégica no setor público.

1.2 Proponente

Esta Dissertação é proposta pela **ANO Software**, uma empresa de TI, sediada no Porto, com mais de 25 anos de presença no mercado (ANO).

A ANO oferece uma variedade de produtos para a gestão eletrónica de documentos e processos, especialmente direcionados às autarquias locais. Essas soluções são altamente adaptáveis, expansíveis e flexíveis, sendo capazes de se ajustar a diferentes mercados e tamanhos organizacionais. Estas incluem funcionalidades abrangentes, como gestão de expediente, arquivo documental, assinatura digital, *gateways* para SMS, pagamentos eletrónicos, *workflows* estruturados e não estruturados, além de interfaces de integração com outras aplicações.

As soluções abrangem diversas áreas, sendo a Gestão de Processos do Urbanismo uma das principais. Dada a sensibilidade da gestão urbanística, que impacta vários setores da sociedade e está sujeita a diversos interesses, os gestores públicos dessa área dependem de ferramentas confiáveis para orientar as suas escolhas e decisões, dada a exposição intensa a escrutínio.

A implementação bem-sucedida do motor de BI proposto permitirá à ANO Software consolidar sua posição no mercado, oferecendo uma solução inovadora e altamente relevante para as autarquias. A empresa estará na vanguarda da oferta de ferramentas que facilitam a gestão eficaz da informação, contribuindo para a eficiência operacional e tomada de decisões informadas no setor público.

Figura 1.1: Logótipo da ANO Software

1.3 Objectivos

A dissertação tem como principal objetivo apresentar uma solução de BI destinada aos perfis de gestão e decisão das autarquias locais.

O sistema proposto deve dar respostas a questões específicas, tais como a obtenção de indicadores de produtividade e histórico. Além disso, deve procurar fornecer informações detalhadas sobre a caracterização dos dados, com a capacidade de segmentação temporal.

Outro objetivo crucial é a capacidade de fornecer indicadores relacionados aos diversos passos de *workflow* da gestão processual, oferecendo uma visão abrangente e detalhada do progresso e eficiência dos processos em curso. O produto resultante da dissertação deve também ser capaz de gerar listagens que fundamentem os indicadores calculados, enriquecendo a compreensão do seu contexto.

A apresentação das respostas a essas questões deve ser realizada de forma intuitiva, utilizando *dashboards*, gráficos e listagens como ferramentas de visualização.

Em resumo, a dissertação deve procurar não apenas criar uma solução técnica, mas apresentar uma ferramenta prática aos gestores públicos, de forma a permitir às autarquias locais a tomada de decisões informadas e gerir eficientemente os seus recursos, otimizando a sua *performance*.

1.4 Metodologia

Na condução desta dissertação, foi adotada uma abordagem metodológica baseada nos princípios ágeis, utilizando *Scrum* como guia para a gestão e execução do projeto.

A metodologia *Agile* proporcionou uma resposta flexível às mudanças nos requisitos, promovendo a colaboração contínua com os *stakeholders*.

O controlo de versões foi assegurado através do uso do *Git*, garantindo uma gestão eficiente de *branches* e *commits* descritivos.

Além disso, foi implementada uma estratégia de testes, abrangendo tanto testes unitários para validação de componentes específicos como testes de qualidade para garantir o correto funcionamento do sistema como um todo.

1.5 Estrutura da dissertação

A dissertação segue uma estrutura clara e sequencial onde o problema é enquadrado, delineando a necessidade de uma solução inovadora no contexto das Autarquias Locais.

Em seguida, o capítulo 2 explora as teorias fundamentais, os conceitos base e o panorama atual relacionado ao problema em questão.

O capítulo 3, intitulado "Motor de construção de *data warehouses*", descreve a solução proposta, um motor de sincronização e transformação de dados versátil, capaz de se adaptar na construção de um *Data Warehouse*.

O capítulo 4, "O Future Sense no contexto das Autarquias Locais", detalha como o produto *Future Sense* pode ser utilizado de forma eficaz no tratamento e organização de dados no contexto das autarquias locais, em conjunto com um visualizador e explorador de *dashboards* e gráficos.

No capítulo 5, "Análise de dados e resultados", são apresentados e analisados os resultados alcançados com a aplicação da metodologia.

Por fim, o capítulo 6, "Conclusões e Trabalho Futuro", discute as oportunidades de expansão, melhorias e a continuidade do trabalho, delineando os próximos passos na evolução do trabalho realizado.

Capítulo 2

Revisão Bibliográfica

2.1 Conceitos Base

Neste capítulo são descritos alguns conceitos base relevantes para o enquadramento da Dissertação, tais como *Business Intelligence*, *Data Warehouse*, entre outros.

2.1.1 *Sistemas Operacionais (OS)*

Os sistemas operacionais são aqueles que suportam as atividades diárias da organização. Normalmente, estão focados no processamento de transações, de acordo com a aŕea de negócio. É comum os sistemas operacionais utilizarem uma ampla variedade de tecnologias e arquiteturas, e podem incluir alguns sistemas empacotados de fornecedores além de *software* desenvolvido com esse intuito. Os sistemas operacionais são estáticos por natureza; mudam apenas em resposta a uma mudança intencional nas políticas ou processos da organização, ou por motivos técnicos, como manutenção do sistema ou ajustes de desempenho (Imhoff et al., 2003).

No contexto desta dissertação importa reconhecer quais os sistemas operacionais das organizações que serão utilizados na análise e processamento de dados. Neste caso, um passo relevante na construção de um sistema de BI, consiste na análise dos dados presentes nesses sistemas. Por exemplo, nesta dissertação será considerado um *software* de Gestão Processual e Documental, utilizado pelas autarquias locais.

2.1.2 Business Intelligence (BI)

Business Intelligence engloba um conjunto de metodologias, processos, arquiteturas e tecnologias que convertem dados em bruto em informação significativa e útil para a tomada de decisões.

Os sistemas de BI e de apoio à decisão auxiliam gestores em vários níveis organizacionais na análise de informação estratégica. Estes sistemas recolhem vastas quantidades de dados e reduzem-nos a uma forma que pode ser utilizada para analisar o comportamento organizacional. Essa transformação de dados envolve um conjunto de tarefas que, através de processos de extração,

transformação, integração e limpeza, armazenam os dados num espaço comum denominado *Data Warehouse* (DW) (Vaisman e Zimányi, 2014).

É precisamente um motor de BI que se pretende implementar nesta dissertação, um motor capaz de auxiliar os gestores públicos, nomeadamente das autarquias locais a avaliar o comportamento organizacional do município. Desta forma o motor deve ser capaz de extrair os dados do contexto organizacional, transformá-los, limpá-los e organizá-los num *DW* para posteriormente serem utilizados para relatórios e *dashboards*.

2.1.3 *Data warehouse* (DW)

Na década de 90, perante um mundo cada vez mais competitivo e em constante mudança, as organizações perceberam a necessidade de realizar análises de dados sofisticadas para apoiar seus processos de tomada de decisão. Bases de dados operacionais ou transacionais tradicionais não davam resposta aos requisitos para a análise de dados pretendida, uma vez que eram apenas projetadas e otimizadas para suportar as operações diárias de negócios (Vaisman e Zimányi, 2022).

É neste contexto que surge o termo *data warehouse* que, segundo (Inmon, 2002), é definido como uma coleção de dados orientados para o objeto, integrados, não voláteis e variáveis no tempo, destinados a apoiar decisões de gestão.

Por outras palavras, um DW é um repositório de dados integrados obtidos a partir de várias fontes, com o objetivo específico de análise de dados multidimensional (Vaisman e Zimányi, 2014).

2.1.4 Objectivos de um *data warehouse* e *Business Intelligence*

Segundo Kimball e Ross (2013) é com base na perceção das necessidades dos gestores de uma organização que podemos encontrar os objetivos de um sistema de *Business Intelligence*/*Data warehouse*:

- "Produzimos muitos dados, mas não conseguimos interpretá-los"
- "Precisamos de aceder aos dados de uma forma mais fácil"
- "Mostrem-me apenas o mais importante"
- "Passamos reuniões inteiras a discutir sobre os números mais corretos em vez de tomar decisões."

Assim, Kimball e Ross (2013) mostra-nos como podemos tornar essas afirmações em requisitos de uma sistema DW/BI. Conclui-se assim, que o sistema DW/BI deve:

- Tornar a informação facilmente acessível;
- Apresentar a informação de maneira consistente;
- Ser capaz de se adaptar a mudanças;

- Apresentar as informações de maneira oportuna e pertinente;
- Ser um local seguro que protege a informação;
- Servir como a base confiável para a tomada de decisões.

Por fim, refere que a organização deve aceitar o sistema DW/BI para podermos considerá-lo como bem-sucedido (Kimball e Ross, 2013).

2.1.5 Extração, transformação e carregamento (ETL)

Extração, transformação e carregamento (ETL) é o processo central de integração de dados e é tipicamente associado ao armazenamento de dados. As ferramentas de ETL extraem dados de uma fonte escolhida, transformam-nos em novos formatos de acordo com as regras de negócio e, em seguida, carregam-nos na estrutura de dados de destino, ou seja processam os dados relevantes dos sistemas de origem em informações úteis para serem armazenadas no DW. Os dados originais, provenientes de diferentes fontes, são extraídos corretamente e filtrados, transformados e integrados de maneira adequada de forma a facilitar o processo de consulta, que é a espinha dorsal do DW (Goar et al., 2010).

Uma das etapas chave desta dissertação é precisamente a aplicação responsável pelo processo de ETL, já que é crucial que o processamento dos dados seja realizado de forma iterativa e incremental e que a informação a disponibilizar para os gestores seja consistente e útil para a análise que pretendem.

2.1.6 Sistemas OLAP

Uma ampla variedade de sistemas e ferramentas pode ser utilizada para aceder e explorar os dados contidos num DW. O mecanismo utilizado normalmente para essas tarefas tem sido o *Online Analytical Processing* (OLAP). Os sistemas OLAP permitem que os utilizadores consultem interativamente e agreguem automaticamente os dados contidos num DW. Dessa forma, os gestores podem acessar facilmente as informações necessárias e analisá-las em vários níveis de detalhe (Vaisman e Zimányi, 2022).

2.1.7 Indicadores-chave de desempenho (KPIs)

Os indicadores-chave de desempenho, em inglês, *Key Performance Indicators* (KPIs) ajudam a definir e medir os objetivos organizacionais que são fundamentais para o sucesso atual e futuro de uma organização. Ter KPIs sólidos é crucial para organizações que estão a implementar sistemas de gestão de desempenho (Parmenter, 2019).

Nesta Dissertação é importante reconhecer que KPIs são relevantes para o contexto organizacional dos municípios e de que forma estes se relacionam com os OS, em particular com os *softwares* dedicados à Gestão Processual e Documental. Os KPIs são os indicadores chave que os gestores pretendem considerar para avaliar o desempenho da organização, dos seus departamentos e até dos seus fluxos de trabalho.

2.1.8 Esquema em estrela versus cubo OLAP

Os modelos dimensionais implementados em sistemas de gestão de bases de dados relacionais são designados como esquemas em estrela devido à sua semelhança com uma estrutura em forma de estrela. Os modelos dimensionais implementados em ambientes de bases de dados multi dimensionais são designados como cubos OLAP (Kimball e Ross, 2013).

Tanto os esquemas em estrela como os cubos têm um *design* lógico comum com dimensões reconhecíveis, diferindo apenas na sua implementação física. Quando os dados são carregados num cubo OLAP, são armazenados e indexados utilizando formatos e técnicas projetados para dados dimensionais (Kimball e Ross, 2013).

Consequentemente, os cubos oferecem um desempenho de consulta superior devido ao pré-processamento, estratégias de indexação e outras otimizações. Os utilizadores de negócio podem fazer *drill down* ou *up* adicionando ou removendo atributos das suas análises com excelente desempenho sem emitir novas consultas. Os cubos OLAP também fornecem funções analíticas mais robustas que excedem aquelas disponíveis com *Structured Query Language* (SQL). O inconveniente é que o preço a pagar a nível de desempenho no carregamento por estas capacidades é elevado, especialmente com grandes conjuntos de dados (Kimball e Ross, 2013).

Nesta dissertação optou-se por recorrer ao esquema em estrela, de forma a simplificar a implementação e o carregamento de dados para o DW.

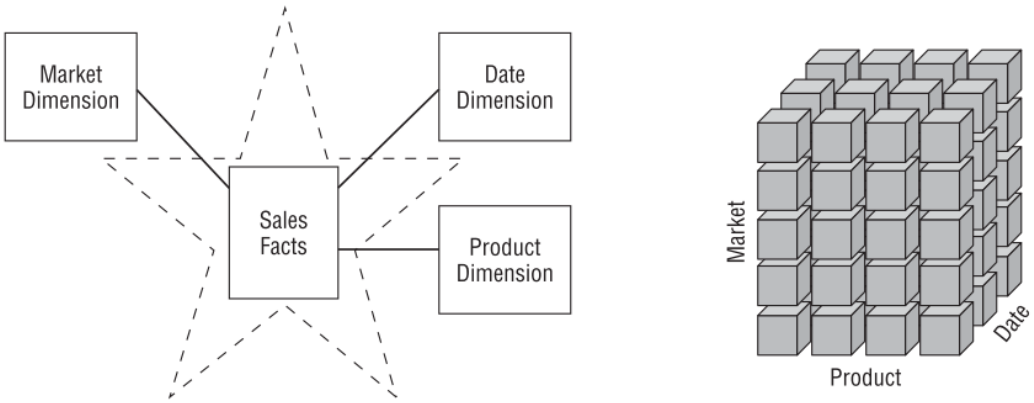


Figura 2.1: Esquema em Estrela e Cubo OLAP (Kimball e Ross, 2013)

2.1.9 Tabelas de factos

A tabela de factos num modelo dimensional armazena as métricas de desempenho resultantes dos eventos de um determinado processo de uma organização. Deve esforçar-se por armazenar os dados de medição a baixo nível resultantes de um processo de negócio num único modelo dimensional (Kimball e Ross, 2013).

O termo "facto"representa uma medida do negócio ou um KPI. Cada linha numa tabela de factos corresponde a um evento de medição. Os dados em cada linha estão a um nível de detalhe

específico, referido como granularidade, como uma linha por produto vendido numa transação de venda. Um dos princípios fundamentais da modelação dimensional é que todas as linhas de medição numa tabela de factos devem estar no mesmo nível de granularidade. Ter a disciplina para criar tabelas de factos com um único nível de detalhe garante que as medições não sejam contadas de forma inadequada (Kimball e Ross, 2013).

2.2 Utilização de BI

2.2.1 Aplicação de BI em diferentes setores de atividade

Tomar decisões rápidas e informadas a partir dos dados pode ser uma chave para o sucesso. As expectativas dos clientes, a competição global e as margens de lucro fazem com que muitas organizações, independentemente de seu tamanho ou setor, recorram a BI para obter uma vantagem competitiva (Morris, 2021).

Por exemplo, utilizar dados para apresentar anúncios personalizados com base no histórico de navegação, fornecer acesso a dados e KPIs para todos os colaboradores e centralizar dados de toda a empresa num único ecossistema digital, permitindo uma análise mais detalhada dos processos, são exemplos de como BI pode ser aplicado (Morris, 2021).

Morris (2021), *Principal Product Marketing Specialist* da *Oracle NetSuite*, apresenta no seu artigo alguns casos de estudo que ilustram como o BI fez a diferença para várias empresas, em diferentes setores de atividade, em todo o mundo:

2.2.1.1 Lotte.com: BI aumenta a receita da empresa

Lotte.com é o maior shopping virtual da Coreia, com 13 milhões de clientes.

Desafio: Com mais de 1 milhão de visitantes diários no *website*, os gestores queriam entender o motivo pelo qual os clientes abandonavam os seus carrinhos de compras.

Solução: O gestor assistente da equipa de Marketing implementou uma solução de análise do comportamento e experiência do cliente, o primeiro sistema de análise comportamental online aplicado na Coreia. O gestor usou essas informações para entender o comportamento dos clientes, e implementou marketing direcionado, transformando o *website*.

Resultados: Com os *insights* do novo programa de análises de BI, houve um aumento na lealdade dos clientes após um ano e um aumento de 10 milhões de dólares nas vendas. As mudanças ocorreram ao identificar as causas do abandono do carrinho. Detetaram um processo de *checkout* longo e tempos de entrega inesperados.

2.2.1.2 New York Shipping Exchange: BI reduz a dependência de TI

A New York Shipping Exchange (NYSHEX) é uma empresa de TI focada no Transporte Marítimo e que trabalha para melhorar o processo de transporte de mercadorias.

Desafio: Para entender o desempenho geral da empresa, a NYSHEX extraía manualmente dados da sua aplicação (OS) e de várias aplicações na *cloud*, e depois importava esses dados para

o *Excel*. Esse era um processo trabalhoso e poucas pessoas tinham acesso aos dados, e a maioria dos pedidos de relatórios recaía sobre a equipa de engenharia. **Solução:** A NYSHEX investiu em BI, centralizou os seus dados num sistema único e deu acesso a toda a empresa, capacitando aqueles sem conhecimento de codificação a realizar análises detalhadas. **Resultados:** Graças ao BI e outros esforços, em 2019 a empresa mais que triplicou seu volume de embarques entre a Ásia e os EUA.

2.2.1.3 Spear Education: BI simplifica processos internos e fluxo de trabalho

A Spear Education é líder em formação para dentistas.

Desafios: O sistema telefónico da Spear carecia de funcionalidades que poderiam tornar os seus representantes de atendimento ao cliente mais eficientes e fornecer um atendimento melhor. Por exemplo, o sistema telefónico não gravava chamadas e não estava conectado a uma ferramenta de gestão de relacionamento com o cliente (CRM). **Solução:** Após algumas pesquisas, a Spear conectou o seu *software* de **call center** com sua solução de BI para manter registos mais completos das interações com os clientes e fornecer uma visão completa das interações. **Resultados:** Após implementar uma nova solução para seu centro de contacto, a Spear aumentou a eficiência dos agentes e economizou 35 horas de tempo de atendimento por semana. Os agentes da Spear agora reinvestem esse tempo fazendo 4.000 chamadas a mais por semana.

2.2.1.4 Cementos Argos: BI melhora a eficiência financeira

A Cementos Argos é uma das maiores produtoras de cimento da América Latina com operações nos EUA, América Central, América do Sul e Caraíbas.

Desafio: A empresa procurava uma vantagem competitiva geral e uma forma de apoiar uma melhor tomada de decisões.

Solução: A Cementos Argos criou um centro dedicado de análises de negócios. A empresa investiu em analistas de negócios experientes e equipes de *Data Science* e usou BI para aproveitar os dados.

Resultados: A empresa padronizou o processo financeiro e aplicou BI para obter uma visão mais aprofundada do comportamento dos clientes, o que resultou em um nível de rentabilidade mais alto.

2.2.1.5 Expedia: BI aumenta a satisfação do cliente

A Expedia é a empresa-mãe de algumas das principais empresas de viagens, incluindo Expedia, Hotwire e TripAdvisor.

Desafio: A satisfação do cliente é essencial para a missão, estratégia e sucesso de uma empresa. A experiência online deve ser agradável para o utilizador, mas a empresa não tinha acesso à opinião dos clientes.

Solução: A empresa possuía grandes volumes de dados que eram agregados manualmente, deixando pouco tempo para análise. Utilizando BI, o departamento responsável pela satisfação

dos clientes conseguiu analisar os dados dos clientes de toda a empresa e vincular os resultados a 10 objetivos diretamente relacionados às iniciativas da empresa. Os responsáveis por esses KPIs constroem, gerem e analisam dados para descobrir tendências ou padrões.

Resultados: A equipa de atendimento ao cliente pode ver em tempo real como estão os KPIs e tomar medidas corretivas, se necessário. Além disso, outros departamentos podem utilizar os dados.

2.2.2 Conclusão

Os exemplos apresentados demonstram claramente o impacto significativo que a utilização de BI pode ter em diversos setores de atividade e em diferentes empresas em todo o mundo. Desde o aumento da receita e da lealdade dos clientes na Lotte.com até a melhoria da eficiência financeira na Cementos Argos, as soluções de BI permitem que as empresas transformem dados brutos em *insights*, promovendo tomadas de decisões mais informadas e estratégicas.

Além disso, o caso da New York Shipping Exchange ilustra como BI pode conectar os departamentos, dados e processos, reduzindo a dependência de TI e melhorando a personalização dos serviços oferecidos.

Finalmente, a Expedia exemplifica a maneira como BI pode simplificar processos complexos e melhorar a satisfação dos clientes, ao mesmo tempo em que cria oportunidades.

Em resumo, BI não apenas otimiza operações internas e externas, mas também fornece uma base sólida para a inovação e a competitividade num mercado em que os dados assumem cada vez mais importância. As empresas que adotam a BI são capazes de adaptar-se rapidamente às mudanças do mercado, antecipar tendências e tomar decisões proativas que resultam em vantagens competitivas sustentáveis e de crescimento contínuo.

Capítulo 3

Motor de construção de *data warehouses*

Para solucionar o problema apresentado nesta dissertação foi considerado o desenvolvimento de um produto capaz de extrair informação de diferentes bases de dados operacionais e transformar esses mesmos dados de forma a conseguir obter as métricas pretendidas numa análise de *Business Intelligence*. Ao longo deste capítulo será descrita toda a especificação do produto.

3.1 *Visão*

Sincronizar, agregar e transformar diferentes fontes de dados.

3.2 Descrição

O ***Future Sense*** é uma aplicação capaz de se conectar e interagir com diferentes fontes de dados (bases de dados relacionais). Na implementação foram inicialmente consideradas fontes de dois motores de bases de dados distintos: *Oracle* e *MySQL*.

A aplicação permite agendar sincronizações dessas fontes de dados, por exemplo, sincronizar diariamente às 3h da madrugada uma determinada fonte de dados *Oracle*. Para além disso, permite que essa sincronização seja realizada de forma incremental, caso sejam indicadas colunas de auditoria nas tabelas da fonte de dados a sincronizar (colunas que indicam quando ocorreu a inserção de um registo e qual foi a última atualização desse mesmo registo).

Após as sincronizações de dados, o utilizador pode definir quais as transformações de dados que devem ocorrer para mais tarde ser possível consultar, visualizar e explorar os dados já transformados.

3.3 Requisitos

Nesta secção serão apresentados os principais requisitos considerados para o desenvolvimento do sistema.

Tabela 3.1: Requisitos do Future Sense

Requisito		Descrição
RQ.1.0.0.1	Consultar, adicionar, atualizar e remover diferentes fontes de dados	Deve ser possível ao utilizador gerir diferentes fontes de dados no sistema
RQ.1.0.0.2	Testar uma ligação a uma fonte de dados	Deve ser possível ao utilizador testar uma ligação do sistema a uma determinada fonte de dados
RQ.1.0.0.3	Testar uma ligação a uma fonte de dados sem a adicionar ao sistema	Deve ser possível ao utilizador testar uma ligação do sistema a uma determinada fonte de dados sem a adicionar ao sistema
RQ.1.0.0.4	Obter as tabelas de dados das fontes de dados	Deve ser possível consultar as tabelas disponíveis na fonte de dados
RQ.1.0.0.5	Consultar, adicionar, atualizar e remover tabelas das fontes de dados para sincronização	Deve ser possível gerir, das tabelas disponíveis na fonte de dados, as tabelas a sincronizar
RQ.1.0.0.6	Consultar as sincronizações e transformações de dados que ocorreram	Deve ser possível consultar as sincronizações e transformações de dados que ocorreram, em particular as ultimas sincronizações e transformações
RQ.1.0.0.7	Consultar, adicionar, atualizar e remover transformações a realizar	Deve ser possível gerir as transformações a realizar após as sincronizações de dados

3.4 Arquitetura e design

3.4.1 Arquitetura tecnológica

Na figura 3.1 estão representadas, a alto nível, as diferentes tecnologias utilizadas no desenvolvimento do produto e a forma como se relacionam.

Descrevendo de cima para baixo deparamos-nos com a representação das bases de dados operacionais de um sistema arbitrário, onde se consideram diferentes tecnologias associadas a essas fontes de dados (*MySQL*, *Oracle*, *PostgreSQL*, ...).

De seguida, observamos uma *Application Programming Interface* (API), um serviço REST (*Representational State Transfer*), que se conecta às diferentes fontes de dados e que é responsável pelo processo de extração dos dados das bases de dados operacionais, transformação e carregamento dos dados, populando um DW, em *PostgreSQL*. À API do *Future Sense* liga-se também uma *Aplicação Web* desenvolvida em *React.js* que permite através da API configurar as fontes de dados e monitorizar as sincronizações e transformações de dados.

Por último, o DW é conectado a um visualizador de *dashboards* e gráficos *open-source*, o *Apache Superset*.

Todos os componentes descritos, com exceção das bases de dados operacionais, são *containers Docker*, o que permite agilidade no desenvolvimento e na instalação do produto.

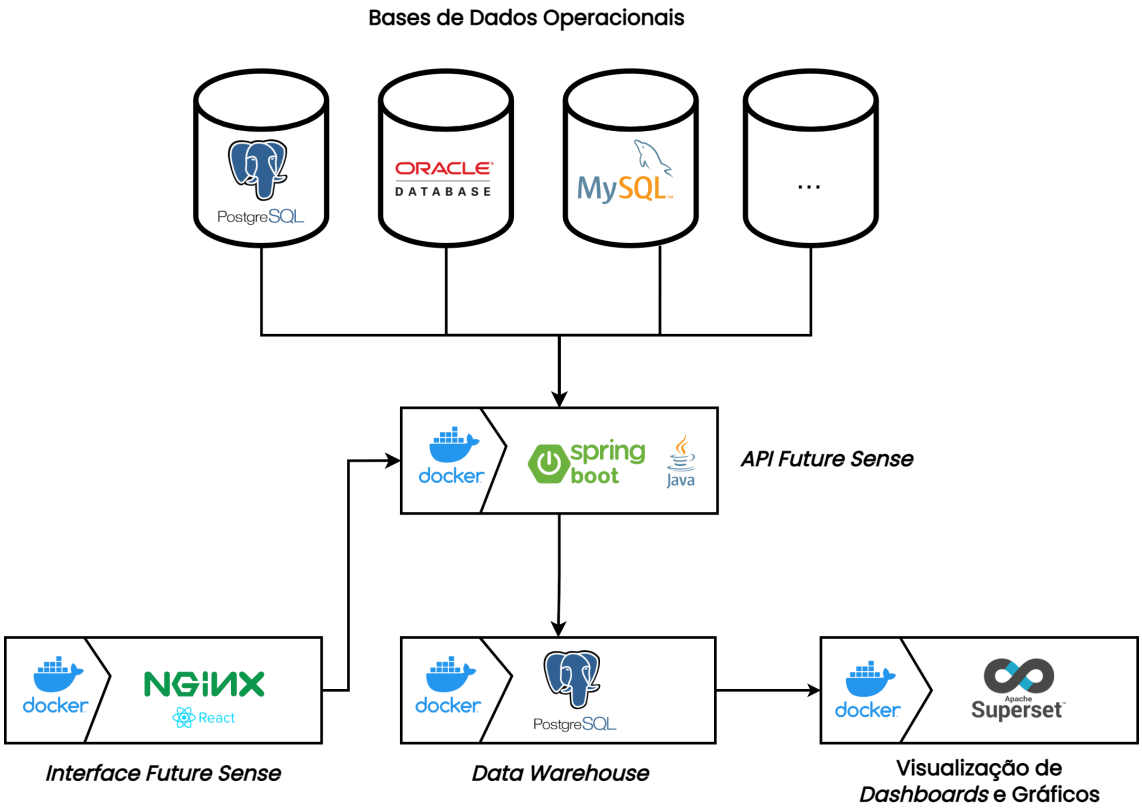


Figura 3.1: Arquitetura Tecnológica

3.4.2 Descrição das tecnologias

3.4.2.1 Docker

O *Docker* é simplesmente uma ferramenta para criar e gerir *containers* (Adetunji, 2023).

Por outras palavras, o *Docker* é uma plataforma *open-source* que permite aos programadores desenvolver, implementar, executar, atualizar e gerir *containers* executáveis e padronizados que combinam o código-fonte de aplicações com as bibliotecas e estruturas do sistema operativo necessárias para executar o código em qualquer ambiente (IBM, a).

Segundo a empresa *Docker Inc*, desenvolver aplicações hoje em dia requer muito mais do que escrever código. A utilização de múltiplos idiomas, estruturas, arquiteturas e interfaces descontínuas entre ferramentas para cada fase do ciclo de vida cria uma enorme complexidade. O *Docker* simplifica e acelera o seu fluxo de trabalho, ao mesmo tempo que dá aos programadores a liberdade para inovar com a sua escolha de ferramentas, pilhas de aplicações e ambientes de implementação para cada projeto (Docker).

Conforme referido anteriormente, no desenvolvimento desta dissertação, recorreu-se a *containers Docker* e a um ficheiro *docker-compose* para definir os *containers* que compõem o sistema. Assim, facilmente se consegue correr o sistema sem que ocorram erros de incompatibilidade das

tecnologias utilizadas mantendo a consistência no desenvolvimento em diferentes computadores, sistemas operativos e facilitando o *deploy* da aplicação considerada.

3.4.3 *PostgreSQL*

O *PostgreSQL* é um sistema de gestão de base de dados relacional de *open-source*, resultado de 30 anos de desenvolvimento contínuo, o que o torna uma das opções mais respeitadas no mercado. A sua popularidade entre *developers* e administradores de sistemas deve-se à sua notável flexibilidade e integridade. Por exemplo, o *PostgreSQL* é capaz de suportar tanto consultas relacionais quanto não relacionais. Para além disso, por ser *open-source*, é constantemente melhorado por uma comunidade com mais de 600 contribuidores, o que garante a sua evolução contínua (Azure).

O motor de base de dados *PostgresSQL* foi escolhido para armazenar os dados relativos ao DW.

3.4.4 Spring Boot

O *Java Spring Framework* (*Spring Framework*) é um *framework open-source*, destinado à criação de aplicações independentes de produção que são executadas através do *Java Virtual Machine* (JVM) (IBM, b).

O *Java Spring Boot* (*Spring Boot*) é uma ferramenta que simplifica e acelera o desenvolvimento de aplicações *web* e microsserviços com o *Spring Framework* (IBM, b).

Para desenvolver a REST API foi utilizado *Spring Boot*. Esta ferramenta facilita o desenvolvimento da solução proposta, uma vez que permite de forma ágil configurar a aplicação, simplificando as conexões às diferentes fontes de dados e permitindo a construção de uma arquitetura estruturada.

3.4.5 *Liquibase*

O *Liquibase* é uma solução de gestão de versões de base de dados que permite a revisão e implementação mais rápida e segura de alterações na base de dados, desde o desenvolvimento até à produção (Inc).

No arranque da aplicação REST API em *Spring Boot* recorremos ao *Liquibase* para criar a estrutura de dados do DW. O *Liquibase* gere as migrações da base de dados permitindo que quando a aplicação corre verifique que alterações foram executadas, apenas executando aquelas que ainda não foram consideradas previamente.

3.4.6 Next.js

A programação *web* está em constante evolução, sendo que continuamente surgem novas ferramentas e tecnologias destinadas a facilitar o desenvolvimento de aplicações. No âmbito dessas ferramentas, os *frameworks* têm ganho destaque como uma maneira de acelerar a criação de projetos e proporcionar funcionalidades adicionais (Clemente, 2023).

O *Next.js* é um *framework* direcionado para desenvolvedores *React* que almejam desenvolver aplicações de forma mais eficiente. Ao contrário de uma biblioteca, que disponibiliza apenas ferramentas, o *Next.js* é um framework para *React* que oferece uma estrutura base para projetos *React*. O *Next.js* já vem pré-configurado com várias funcionalidades, incluindo *routing* e *Server-Side Rendering* (SSR). Ou seja, não é necessário instalar e configurar bibliotecas adicionais para realizar essas tarefas, simplificando assim o processo de desenvolvimento (Clemente, 2023).

O *Next.js* é utilizado para desenvolver a interface gráfica do *Future Sense*. Ou seja, no contexto do produto corresponde ao *frontend* da aplicação.

3.4.7 *Nginx*

O *Nginx* é um *software open-source* para servidores *web* lançado originalmente para navegação HTTP. Hoje, também funciona como *proxy* reverso, balanceador de carga HTTP, e *proxy* de email para os protocolos IMAP, POP3, e SMTP (L., 2023).

No contexto da dissertação o *Nginx* é utilizado como servidor *web* para servir a interface gráfica do *Future Sense* desenvolvida em *Next.js*.

3.4.8 *Apache Superset*

O *Apache Superset* é um visualizador de *dashboard* e gráficos, é uma aplicação *web* moderna, orientada para BI. É rápido, leve, intuitivo e repleto de opções que facilitam aos utilizadores de todos os níveis de competência explorar e visualizar os seus dados, desde simples gráficos circulares até gráficos geoespaciais detalhados (Foundation).

O *Apache Superset* é talvez uma das ferramentas mais relevantes que integra com o *Future Sense*. Esta ferramenta permite visualizar em múltiplos formatos os dados previamente consumidos e processados pelo *Future Sense*, permitindo aos gestores obter as métricas pretendidas para a análise da sua área de negócio.

3.5 Arquitetura lógica

No contexto do *Future Sense* é relevante o desenho da arquitetura lógica do módulo "API*Future Sense*", visto que é este módulo que integra toda a lógica de negócio inerente às sincronizações e transformações de dados. Optou-se assim, por uma desenhar uma arquitetura que respeite os princípios SOLID:

- S — **Single Responsiblity Principle** (Princípio da responsabilidade única);
- O — **Open-Closed Principle** (Princípio Aberto-Fechado)
- L — **Liskov Substitution Principle** (Princípio da substituição de Liskov)
- I — **Interface Segregation Principle** (Princípio da Segregação da Interface)
- D — **Dependency Inversion Principle** (Princípio da inversão da dependência)

O respeito por estes princípios torna o código mais limpo, organizado e capaz de responder no futuro a modificações e alterações sem comprometer o trabalho previamente realizado.

A *Controller Layer* é a camada responsável por rececionar os pedidos, verificar permissões e reencaminhar esses mesmos pedidos para a camada *Service Layer*. Esta última, por sua vez é responsável pela lógica de negócio, as validações inerentes a cada solicitação, a instanciação de novas entidades e a criação de relações. A camada *Repository Layer* é responsável pela persistência das entidades e pela lógica da base de dados.

Na comunicação entre as camadas *Controller Layer* e *Service Layer*, na figura 3.2 conseguimos ver respresentado um *Data Transfer Object* (DTO), um objeto que mapeia os dados enviados nos pedidos ao servidor e permite filtrar os dados na resposta a esses mesmos pedidos. Os *Data Transfer Objects* (DTOs) são utilizados, pois é uma boa prática não expor diretamente as entidades (*Model*). Para fazer esse mapeamento do *Model* para DTO e, o contrário, de DTO para *Model* é utilizado um *Mapper*.

Assim, com a arquitetura representada na figura 3.2 percebemos que cada módulo/componente tem a sua responsabilidade o que permite aplicar testes unitários direcionados para cada funcionalidade/responsabilidade.

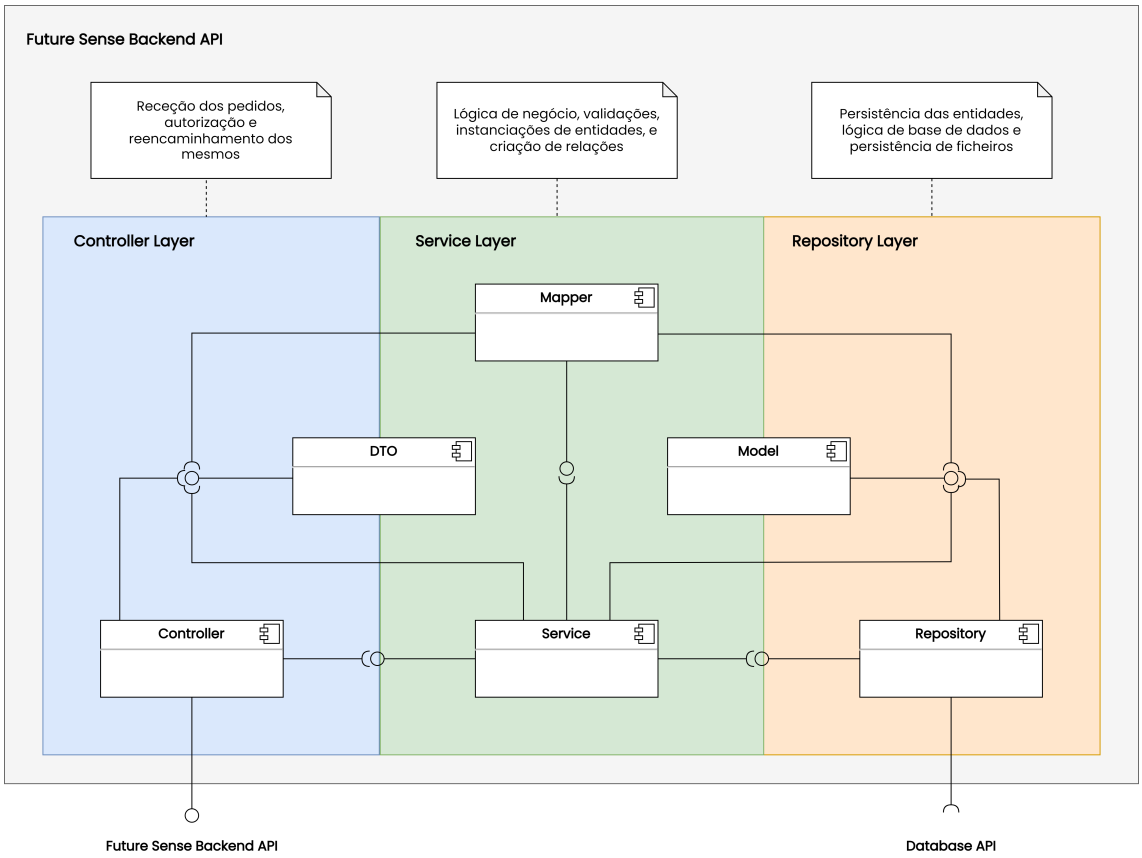


Figura 3.2: Arquitetura Lógica da API do Future Sense

3.6 Construção do *data warehouse*

Para o desenvolvimento de um produto com a visão apresentada é necessária a criação e a manutenção de um DW capaz de armazenar os dados de forma a permitir a resposta às interrogações colocadas pelos *stakeholders*.

Para a construção do DW recorreu-se a um motor de base de dados relacional, neste caso *PostgreSQL*.De forma a controlar a gestão das fontes de dados, a sincronização e a transformação de dados foi necessário a criação de uma estrutura de dados capaz de armazenar informação acerca do estado atual do DW e de que forma devem ocorrer as sincronizações e transformações de dados.

Assim, de forma a organizar a disposição dos dados armazenados no *Future Sense*, foi criado o *schema* "control", que considera as tabelas responsáveis por armazenar a informação relativa à manutenação do DW.

3.7 Modelo de dados

O modelo de dados do Future Sense mapea com o schema control da base de dados e segue assim uma estrutura desenhada especificamente para cumprir os requesitos da aplicação, uma abordagem versátil que permite conjugar e agendar sincronizações e transformações das diferentes fontes de dados. No diagrama UML apresentado na figura 3.3 conseguimos perceber de que forma foi desenhado esse modelo de dados capaz de ajustar ao requesitos do produto.

A classe ***Source*** representa uma fonte de dados a partir da qual podem ocorrer sincronizações e transformações de dados. Neste caso, uma fonte de dados pode ser uma a base de dados relacional associada a um OS de uma determinada organização. A tabela foi assim criada no DW, mapeando a classe **Source** representada no UML, conforme o exemplo da tabela 3.2.

Tabela 3.2: Exemplo de registo na tabela sources do schema control

id	driver	host	port	database	user	password	active
gsp	oracle	192.168.9.59	1521	bd01	admin	admin	true

A classe **SourceTable** relaciona as fontes de dados com as tabelas das respetivas fontes e indica se deve ser realizada uma sincronização incremental/temporal.

A classe **Sync** armazena cada sincronização realizada para uma determinada tabela de uma determinada fonte de dados, conforme exemplificado na tabela 3.3. Através da variável last_sync conseguimos perceber qual foi a última sincronização que ocorreu de uma determinada tabela de dados.

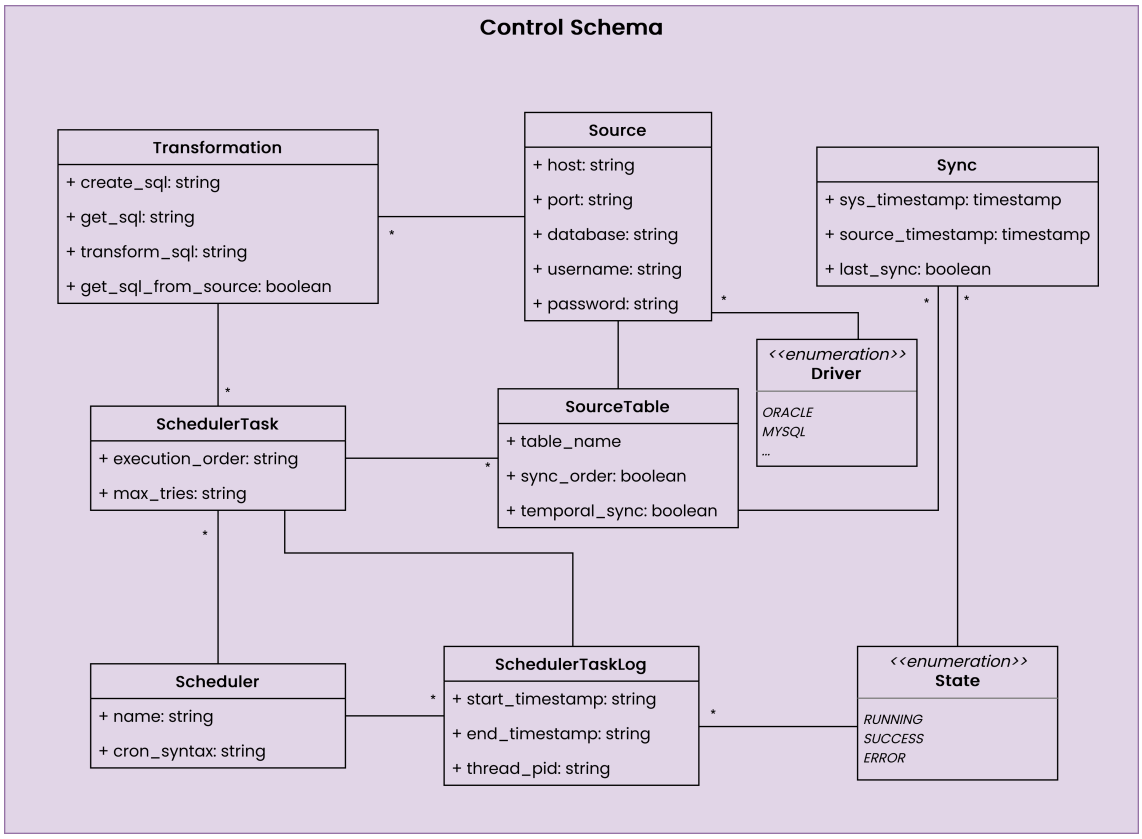


Figura 3.3: Diagrama de Classes do Control Schema (UML)

Para cada sincronização de dados é registada a data do sistema no início da sincronização (sys_timestamp) e a data da fonte de dados (source_timestamp) para garantir que a sincronização temporal é realizada com a data da fonte de dados que pode estar desfazada da data do sistema no caso de a fonte estar noutra máquina. É também indicado o estado da sincronização. Pelo exemplo apresentado conseguimos perceber que a sincronização do dia 28-12-2023 apresentou um erro, e que a ultima sincronização ocorreu no dia 29-12-2023 com sucesso.

A classe **Transformation** armazena de que forma devem ser realizadas as transformações de dados no final das sincronizações ocorrerem, isto é as *queries* que devem ser executadas de forma a transformar os dados para uma tabela de factos ou de dimensão que responda às necessidades dos gestores da organização.

Para analisar uma transformação, consideramos um exemplo de um negócio de retalho, onde pretendemos obter diferentes métricas (exemplo: os produtos mais vendidos em cada loja por mês).

Através da variável ***"create_sql"***, caso esta seja definida, é possível criar uma tabela de factos capaz de armazenar os dados relacionados com a transformação a ocorrer. Deve ser definido um comando SQL que efetue a criação da tabela, caso ela não exista, com uma coluna DW_ID que corresponde ao identificador do registo na tabela de factos:

Tabela 3.3: Exemplo de registo na tabela "syncs"do schema control

id	source_table_id	sys_timestamp	source_timestamp	state	last_sync
UUID()	UUID()	2023-12-29 10:24:32	2023-12-29 10:24:33	SUCCESS	true
UUID()	UUID()	2023-12-28 10:29:59	2023-12-28 10:30:01	ERROR	false

```
CREATE TABLE IF NOT EXISTS TRANSACTIONS_DETAIL (
    DW_ID SERIAL PRIMARY KEY,
    TRANSACTION_ID UUID UNIQUE,
    SHOP_ID INTEGER,
    CLIENT_ID UUID,
    PRODUCT_ID UUID,
    AMOUNT DECIMAL,
    TIMESTAMP TIMESTAMP,
    FOREIGN KEY (SHOP_ID) REFERENCES APP_SHOPS,
    FOREIGN KEY (CLIENT_ID) REFERENCES APP_CLIENTS,
    FOREIGN KEY (PRODUCT_ID) REFERENCES APP_PRODUCTS,
    CONSTRAINT UNK_TRANSACTION_ID UNIQUE(TRANSACTION_ID)
)
```

Exemplo de query "create_sql"

Após a execução da *query* de criação da tabela de factos, através da variável "get_sql", deve ser definida uma *query* SQL que seleciona dados do DW ou de uma das fontes registadas (pela indicação da fonte, através da variável "get_sql_from_source").

```
SELECT * FROM TRANSACTIONS
WHERE "TIMESTAMP" >= :LAST_TRANSFORMATION_TIMESTAMP;
```

Exemplo de query "get_sql"

Depois da seleção de dados, para cada registo obtido é aplicada a *query* de inserção e transformação de dados definida através da variável "transform_sql".

```
INSERT INTO TRANSACTIONS_DETAIL(
    TRANSACTION_ID,
    SHOP_ID,
    CLIENT_ID,
    PRODUCT_ID,
    AMOUNT,
    TIMESTAMP
)
VALUES (
:ID,
(SELECT s."DW_ID" as SHOP_ID FROM app_shops c WHERE ID=:SHOP_ID),
(SELECT c."DW_ID" as CLIENT_ID FROM app_clients c WHERE ID=:CLIENT_ID),
```

```
13 (SELECT p."DW_ID" as PRODUCT_ID FROM app_products p WHERE ID=:PRODUCT_ID),
14 :VALUE,
15 :TIMESTAMP
16 ) ON CONFLICT ON CONSTRAINT UNK_TRANSACTION_ID DO
17 UPDATE SET
18     SHOP_ID=(SELECT s."DW_ID" as SHOP_ID FROM app_shops c WHERE ID=:SHOP_ID),
19     CLIENT_ID=(SELECT c."DW_ID" as CLIENT_ID FROM app_clients c
20     WHERE ID=:CLIENT_ID),
21     PRODUCT_ID=(SELECT p."DW_ID" as PRODUCT_ID FROM app_products p WHERE ID=:
           PRODUCT_ID),
22     AMOUNT=:VALUE,
23     TIMESTAMP=:TIMESTAMP
```

Exemplo de query "transform_sql"

De acordo com o exemplo apresentado, é inserido por cada registo obtido na *query* "get_sql"um novo registo na tabela de factos que mapeia as relações com as tabelas previamente sincronizadas no DW, neste caso a tabela app_shops (lojas da cadeia), app_clients (clientes) e app_products (produtos). Os valores atribuídos com ":"correspondem à coluna obtida na *query* de seleção (get_sql).

Para garantir a consistência de dados é importante considerar uma *constraint UNIQUE* na criação da tabela de factos que permita ser validada quando ocorre a inserção de um novo registo. Validando a *constraint* garantimos que a transformação possa ser realizada várias vezes para o mesmo registo sem colocar em causa a integridade dos dados. Assim, é possível popular uma tabela de factos com as relações inerentes à exploração de dados pretendida.

A classe **Scheduler** permite agregar e agendar rotinas de tarefas (sincronizações e transformações) através de uma *cron_expression*, isto é uma expressão que representa através da disposição de caracteres e/ou números uma hora específica (às 15h horas, todos os dias), um intervalo de tempo (de minuto a minuto) ou uma determinada periodicidade temporal (no primeiro dia de cada mês às 12h). Desta forma é possível agendar um conjunto de tarefas a ser executada automaticamente com uma determinada periodicidade.

Tabela 3.4: Exemplo de registo na tabela "shedulers"do schema control

id	name	cron_syntax	active
UUID()	tarde	0-59 * 19 * * ?	true
UUID()	madrugada	0-59 0 3 * * ?	true

No exemplo apresentado, podemos visualizar dois registos de *schedulers* ativos, um com o nome "tarde"que é executado diariamente às 19h da tarde e outro com o nome "madrugada"que é executado diariamente pelas 3h da madrugada.

A classe **SchedulerTask** representa uma tarefa a ser executada pelo **Scheduler** e relaciona-se com a classe **SourceTable**, no caso de uma sincronização ou com a classe **Transformation** no

caso da tarefa ser uma transformação de dados. Para além disso, a *SchedulerTask* permite indicar a ordem de execução de cada tarefa do **Scheduler** (através da variável execution_order).

A classe ***SchedulerTaskLog*** permite registar a execução das tarefas realizadas por um determinado ***Scheduler*** ativo. Permite assim perceber a duração de cada tarefa (através do start_timestamp e do end_timestamp), o *Process Identifier* (PID) da *thread* responsável pela tarefa em questão, assim como o estado da tarefa (*RUNNING*, *SUCCESS* ou *ERROR*), bem como se a tarefa foi terminada ou não (*finished*).

3.8 Sincronização de dados

3.8.1 Colunas de auditoria

A fim de monitorizar as alterações nas tabelas da base de dados de operação e de garantir a consistência de dados com o DW, em alguns casos foi necessário adicionar colunas de auditoria nas tabelas das bases de dados dos OS:

```
1
2  ALTER TABLE TABLE_IA
3  ADD (
4      INS_USER_AUDIT VARCHAR2(200),
5      INS_DATA_AUDIT DATE,
6      ALT_USER_AUDIT VARCHAR2(200),
7      ALT_DATA_AUDIT DATE
8  );
```

3.8.2 *Trigger* de alterações

Através da criação de um *trigger*, que preenche as colunas de auditoria a cada operação de *INSERT* ou *UPDATE*, conseguimos perceber quando os registos são novos ou sofreram alterações na fonte de dados que estamos a analisar.

```
1  CREATE OR REPLACE TRIGGER "SCHEMA".TABLE_IA
2    BEFORE INSERT OR UPDATE ON TABLE
3    FOR EACH ROW
4   DECLARE
5    v_data_audit Date;
6    v_user_audit Varchar2(1000);
7   BEGIN
8     v_data_audit := sysdate;
9     v_user_audit := substr(to_char(sysdate,'dd-mm-yyyy hh24:mi:ss')
10    ||': '
11    || SYS_context('USERENV','OS_USER') || ', '
12    || SYS_context('USERENV','HOST') || ', '
```

```
13    || SYS_context('USERENV','IP_ADDRESS') || ', '
14    || SYS_context('USERENV','MODULE'), 1, 200);
15    if INSERTING
16    then
17         :new.ins_data_audit := v_data_audit;
18         :new.ins_user_audit := v_user_audit;
19    elsif UPDATING
20    then :new.alt_data_audit := v_data_audit;
21         :new.alt_user_audit := v_user_audit;
22    end if;
23  END;
```

Este exemplo de código destina-se a uma base de dados *Oracle*, utilizada no contexto desta dissertação.

Desta forma, é possível perceber quando ocorrem alterações nas tabelas da base de dados de operação garantindo que a sincronização de dados é realizada de forma incremental e consistente.

3.9 Interface gráfica do *Future Sense*

No sentido de melhorar a usabilidade do *Future Sense* e particularmente para contextos em que não existe conhecimento técnico para a gestão do DW foi considerado o desenvolvimento de uma aplicação *frontend*, isto é paginas *web* onde é possível fazer a gestão das fontes de dados e das sincronizações e transformações do DW através de uma interface gráfica.

Para desenvolver a interface foi utilizada a *framework* de *javascript Next.js*, uma *framework* que utiliza por base o *React.js* e que permite de forma mais ágil desenvolver uma aplicação *web*.

3.9.1 Listar fontes de dados

Na figura 3.4 podemos ver a página *web* que lista as fontes de dados disponíveis para sincronização e transformação. Neste exemplo, vemos três fontes de dados distintas: a fonte "gsp", uma base de dados *Oracle*; a fonte "futuredoc", uma base de dados *MySQL* e "teste", uma base de dados *PostgreSQL*. No canto superior direito verificamos que existe um botão "*New Source*"que encaminha para a página de adicionar uma nova fonte de dados.

3.9.2 Nova fonte de dados

Conforme indicado anteriormente através da página de listar as fontes de dados é possível desencadear a ação de associar uma nova fonte de dados. Na figura 3.5 podemos visualizar o formulário a preencher com os dados de ligação à fonte pretendida. Atribuímos um identificador da fonte de dados (id), de seguida selecionamos o *driver* pretendido, associado ao motor de base de dados da fonte (*Oracle*, *MySQL* ou *PostgreSQL*), indicamos o *host* (ip e url), o porto, o nome da base de dados e as credenciais de autenticação.

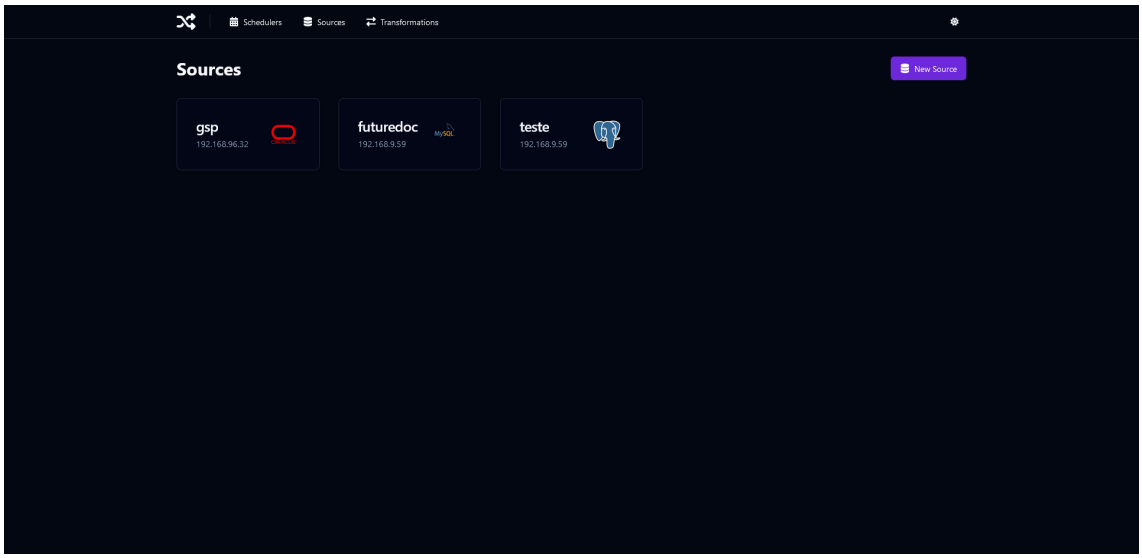


Figura 3.4: Future Sense App - Listar fontes de dados

```
id: app
driver: Oracle
host: 192.168.9.59
port: 1521
database: my-db
username: admin
password: admin
```

Posteriormente, com o formulário preenchido podemos testar a conexão à base de dados e de seguida adicioná-la como fonte de dados disponível para sincronizar dados com o DW.

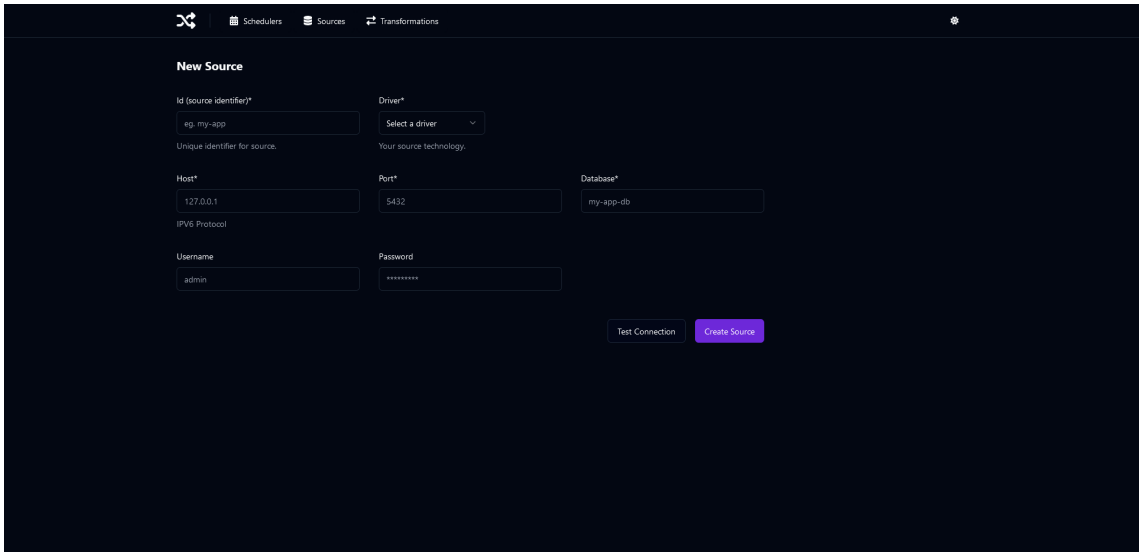


Figura 3.5: Future Sense App - Nova fonte de dados

3.9.3 Consultar fonte de dados

Através da página de listagem das fontes descrita em 3.9.1 é possível selecionar uma fonte de dados e ser encaminhado para a página de detalhe dessa mesma fonte, conforme apresentado nas imagens 3.6 e 3.7. No canto superior direito podemos testar a conexão à fonte de dados e iniciar a sincronização de dados dessa fonte.

Do lado esquerdo é apresentada uma *tab* em que é possível alternar a visualização da informação entre as últimas sincronizações (figura 3.6) e as tabelas associadas à fonte de dados (figura 3.7).

3.9.3.1 Consultar sincronizações da fonte de dados

Na figura 3.6 podemos visualizar todas as sincronizações de tabelas que ocorreram na fonte de dados. Cada sincronização tem um identificador único e está associada a uma tabela. Conseguimos também verificar o estado da sincronização (se foi realizada com sucesso, se está em processamento ou se ocorreu um erro na sincronização), a data em que se iniciou a sincronização e se estamos perante a última sincronização daquela tabela ou não.

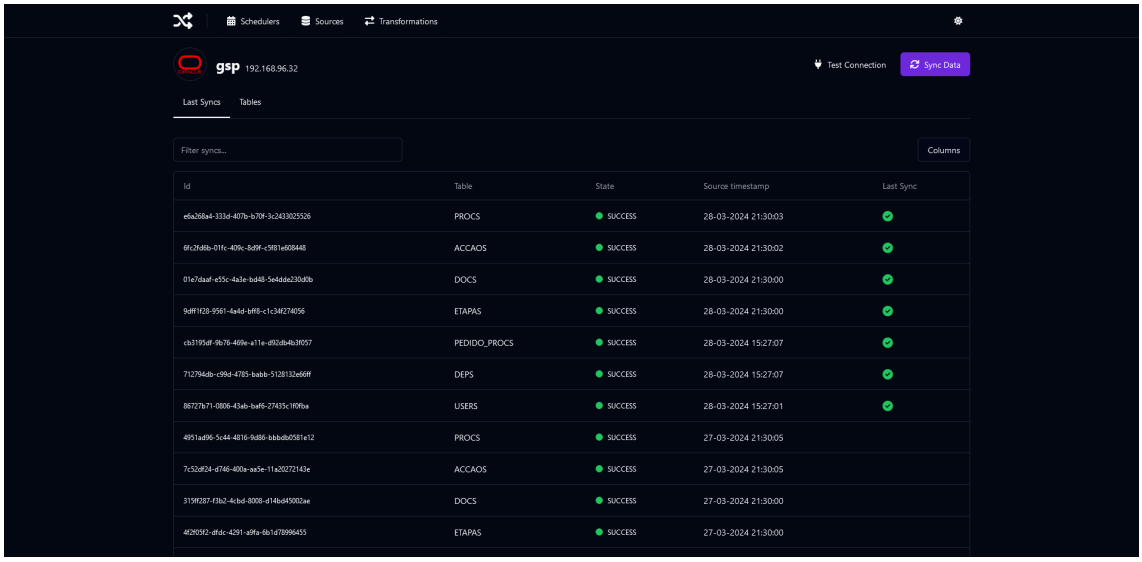


Figura 3.6: Future Sense App - Consultar Fonte de Dados | Sincronizações

3.9.3.2 Consultar tabelas da fonte de dados

Na página de consulta da fonte também é possível consultar todas as tabelas adicionadas para realizar sincronizações. Por outras palavras, neste ecrã não irão aparecer todas as tabelas da fonte de dados, mas apenas aquelas que foram adicionadas e consideradas para sincronizações e transformações de dados.

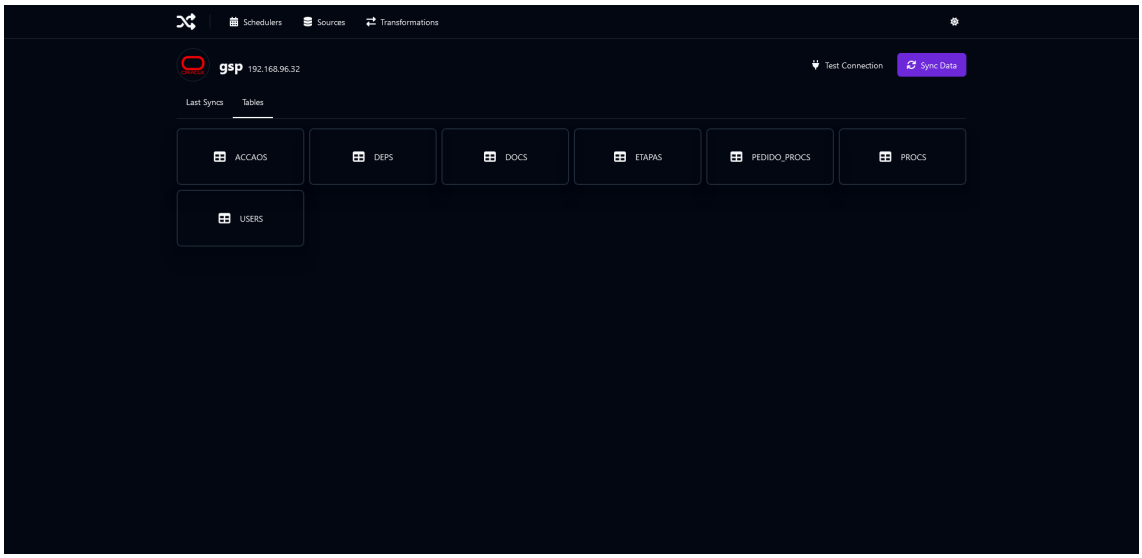


Figura 3.7: Future Sense App - Consultar Fonte de Dados | Tabelas

3.9.4 Consultar *schedulers*

Do mesmo modo que é possível consultar as diferentes fontes de dados, também é possível consultar todos os *schedulers* (agendamentos) numa página dedicada. Conseguimos visualizar o nome do *scheduler*, o seu estado (ativo e inativo) e a sua periodicidade temporal. No canto superior direito visualizamos um botão para criar um novo *scheduler*.

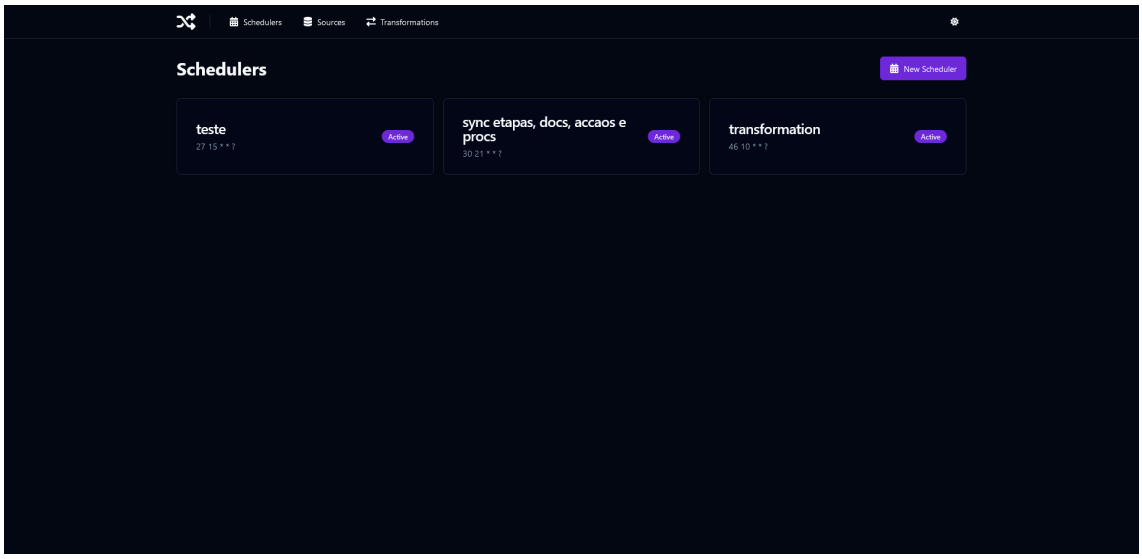


Figura 3.8: Future Sense App - Consultar Schedulers

3.9.5 Criar *schedulers*

Quando clicamos no botão de criar um novo *scheduler*, no ecrã de consulta de *schedulers* (3.8) um pequeno *modal* aparece para que possamos introduzir os dados. Devemos preencher o nome

do *scheduler*, a periodicidade temporal em *Quartz Cron syntax* e de seguida podemos adicionar uma breve descrição.

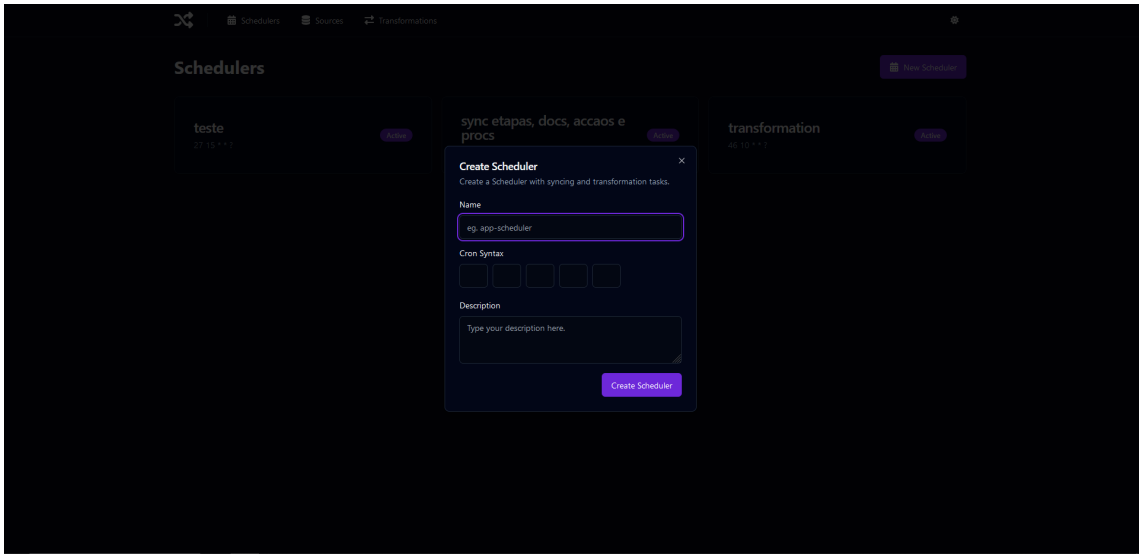


Figura 3.9: Future Sense App - Criar Scheduler

3.10 Descrição da REST API

Nesta secção são descritos os métodos em *Hypertext Transfer Protocol* (HTTP) da REST API do *Future Sense*.

3.10.1 *Sources*

Métodos HTTP responsáveis por gerir as fontes de dados:
POST /sources - Criação de uma fonte de dados.
GET /sources - Obter todas as fontes de dados.
POST /sources/test-connection - Testar a ligação a uma fonte de dados.
GET /sources/{id} - Obter uma fonte de dados pelo id.
POST /sources/{id}/test-connection - Testar a ligação a uma fonte de dados pelo id.

3.10.2 *Tables*

Métodos HTTP responsáveis por gerir as tabelas das fontes de dados:
POST /sources/tables - Adicionar uma tabela de uma determinada fonte de dados.
GET /sources/{id}/tables - Obter as tabelas previamente adicionadas de uma determinada fonte de dados (pelo id da fonte de dados).

3.10.3 *Schedulers*

Métodos HTTP responsáveis por gerir os *schedulers*:
POST /schedulers - Criação de um scheduler.
GET /schedulers - Obter todos os schedulers.

3.10.4 *Transformations*

Métodos HTTP responsáveis por gerir as *transformações* de dados:
POST /transformations - Criação de uma transformação.
GET /transformations - Obter todas as transformações.

3.11 Testes Unitários

No sentido de verificar o correto funcionamento da aplicação foram realizados alguns testes unitários. Os testes unitários aplicados às diferentes camadas da arquitetura lógica permitiram validar a consistência e o desempenho correto do *Future Sense*.

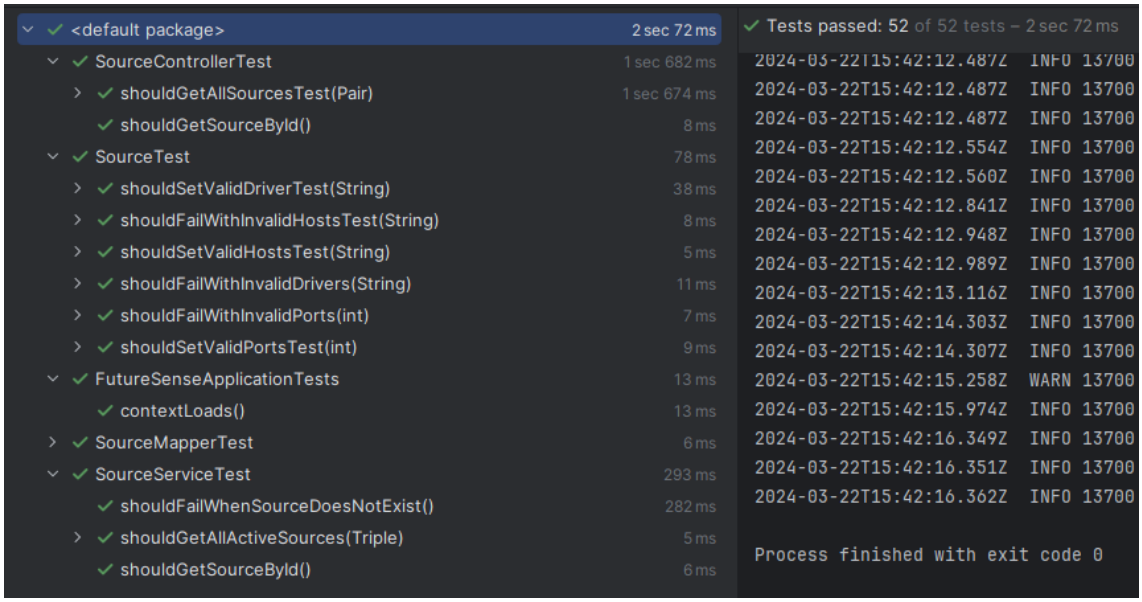


Figura 3.10: Exemplo da Execução de Testes Unitários (IntelliJ IDEA)

Capítulo 4

O Future Sense no contexto das Autarquias Locais

Neste capítulo é descrito como o produto desenvolvido, o Future Sense, consegue adaptar-se ao contexto das Autarquias Locais. Com recurso a um visualizador de dashboards e gráficos é possível os gestores de topo, nomeadamente os cargos de Presidente e Vereação explorarem os dados e filtrarem de acordo com a sua análise.

4.1 Adicionar fontes de dados

No contexto desta dissertação, apenas foi considerado o *software* de Gestão Processual e Documental da ANO Software (empresa proponente) com o intuito de construir um DW capaz de responder às necessidades dos clientes do proponente. Assim sendo, a base de dados, neste caso, com um motor *Oracle* foi adicionada como fonte ao Future Sense.

4.2 Modelo de dados - base de dados operacional

O modelo de dados da base de dados de operação, foi um modelo desenhado e estruturado ao longo do tempo, capaz de dar resposta à gestão processual e documental de uma autarquia, com recurso a um *workflow* estruturado. Para esta análise apenas foi considerada uma parte deste modelo, isto é a parte que diz respeito à tramitação processual e à performance dos diferentes departamentos e utilizadores da organização, visto que neste contexto é aquilo que se pretende analisar e de onde se pretende extrair conhecimento.

No modelo apresentado na figura 4.1 estão representados os utilizadores (USERS) e os departamentos da organização (DEPS), sendo que um utilizador pode pertencer a mais do que um departamento. Estão também representados os processos geridos pela autarquia, através da classe PEDIDO_PROCS. Cada processo tem uma chave composta constituída pelo tipo do processo (PROC_COD_PROC - mapeia com a classe TIPO_PROCS), o número do processo (NREG_P) e o ano em que é criado o processo (ANO_P). Deste modo, no software de Gestão Processual

e Documental, o processo é identificado através da chave no formato <NREG_PG>/<ANO_P> <PROC_COD_PROC>. Por exemplo o processo 3/2024 LE-EDI, é o processo número 3, do ano de 2024, do tipo Licença de Obras de Edificação. O número do processo é incrementado sempre que é criado um novo processo do mesmo tipo de processos.

No entanto, a classe que deve ser analisada com mais detalhe, e sobre a qual se vai debruçar a transformação de dados, é a classe ROTEIROS, pois a cada registo de ROTEIROS corresponde um passo do processo (workflow estruturado). Cada registo de ROTEIRO tem uma chave composta: a chave composta de PEDIDO_PROCS (ped_proc_proc_cod_proc - tipo de processo; ped_proc_nreg_p - número do processo; ped_proc_ano_p - ano do processo) e a etapa (peda_pro_et_etapa_cod_etapa - relaciona-se com a classe ETAPAS), o documento (peda_dc_cod_doc - relaciona-se com a classe DOCS), e ação (peda_accao_cod_accao - relaciona-se com a classe ACCAOS) em que o processo se encontra.

Quando um novo processo é registado, automaticamente é criado para esse processo o primeiro passo na tabela ROTEIROS que corresponde à ação predefinida na tabela PEDAS para esse tipo de processo.

A etapa é um agregador de ações e o documento é o documento que deve ser produzido no respetivo passo do processo. Cada ação do roteiro é também identificada por um número sequencial (seq_accao) de forma a podermos perceber a sequência de passos a que um determinado processo foi sujeito. Sabemos que uma determinada ação foi concluída e existe uma nova ação que foi desencadeada para um determinado processo caso a variável transpor seja 'S', no caso de ser 'N', indica que o processo se encontra na ação que está a ser analisada.

Para além disso, importa perceber que cada ação tem um departamento responsável para a tratar (DEP_COD_DEP_DESTINO) e o departamento associado à ação anterior, isto é o departamento que encaminhou o processo (DEP_COD_DEP_INICIO).

Do mesmo modo, a uma determinada ação pode ser atribuído um utilizador responsável que pertença ao departamento responsável (USERS_USR_LOGIN_DESTINO). É também registado o utilizador que termina a ação anterior e inicia a ação atual (USER_INI) e o também o utilizador que termina a ação atual (USER_FIM).

Os departamentos responsáveis para cada ação e a sequência de ações para cada tipo de processo obedecem a uma estrutura previamente definida através da classe PEDAS, que para cada tipo de processo define um *workflow* a seguir, isto é um conjunto de etapas e ações a serem desencadeados de forma sequencial, cíclica ou segundo um determinado padrão. Para além disso, os documentos podem também ser agrupados por tipo de documento através da classe TIPO_DOCS, do mesmo modo os processos também podem ser agrupados pela classe AMBITO_PROCESSOS.

Apesar da tabela ROTEIROS gerir toda a tramitação dos processos é um tabela complexa que não consegue dar resposta ao intervalo temporal dos processos, pelo que deve ser simplificada e otimizada de forma a que possa ser feita uma análise de BI.

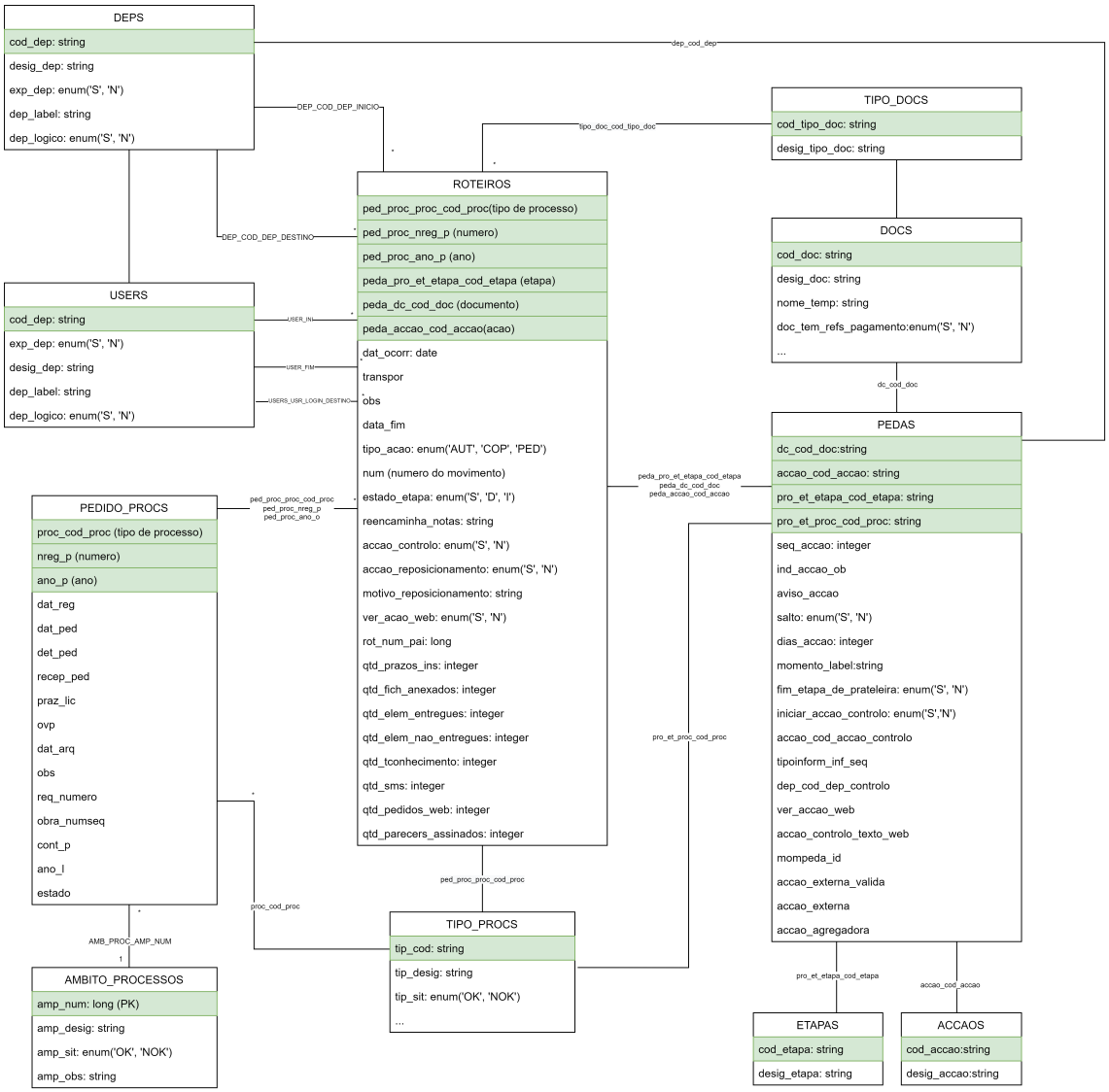


Figura 4.1: Diagrama de Classes do Software de Gestão Processual e Documental (UML)

4.3 Modelo do Data Warehouse

Para as métricas que se pretende analisar no contexto desta dissertação foi considerado um esquema em estrela no desenho do DW. Foi considerada a tabela de factos "WORKFLOW_STEPS", em que cada registo corresponde à simplificação da tabela ROTEIROS da base de dados operacional, com algumas métricas calculadas.

As métricas com mais relevo que devem ser calculadas dizem respeito ao intervalo de tempo desde a ação inicial do processo até ao passo em análise e o intervalo de tempo destinado a cada ação.

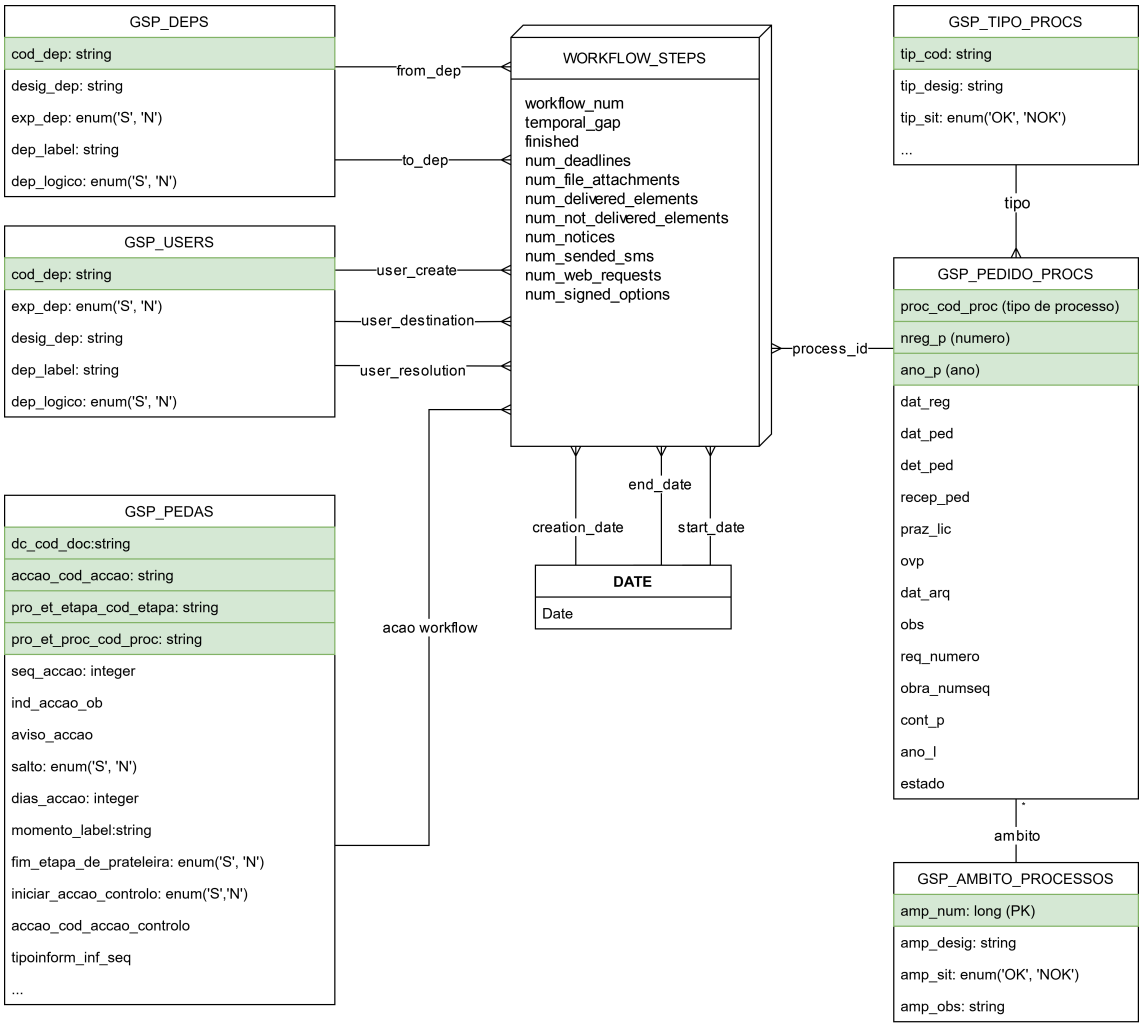


Figura 4.2: Modelo do Data Warehouse (UML)

4.4 Sincronizações

Através do mecanismo de sincronização do Future Sense é possível sincronizar as tabelas da base de dados operacional de forma incremental, caso sejam definidas as colunas de auditoria temporal em que é inserido ou atualizado um registo.

Para obter as informações e as relações inerentes à construção da tabela de factos é necessário previamente sincronizar todas as tabelas da base de dados operacional que contém relações com a tabela de factos. Neste caso a tabela GSP_DEPS, GSP_USERS, GSP_PEDAS, GSP_PEDIDO_PROCS são relações diretas com a tabela de factos, já as tabelas GSP_TIPO_PROCS e GSP_AMBITO_PROCESSOS relacionam-se indiretamente, o que significa que têm que ser as primeiras tabelas a sincronizar. Depois destas tabelas estarem sincronizadas, pode ocorrer a transformação capaz de criar e popular a tabela de factos.

4.5 Transformações

A tabela de factos é criada de acordo com o modelo de dados definido para o DW. A tabela acaba por se traduzir numa simplificação da tabela ROTEIROS da base de dados operacional com chaves estrangeiras para as tabelas previamente sincronizadas. Acaba assim por suportar toda a informação relevante para a exploração de dados da gestão processual estruturada.

```
CREATE TABLE IF NOT EXISTS WORKFLOW_STEPS (
  DW_ID SERIAL PRIMARY KEY,
  WORKFLOW_NUM INTEGER,
  PROCESS_ID INTEGER,
  TEMPORAL_GAP INTERVAL,
  CREATION_DATE TIMESTAMP,
  START_DATE TIMESTAMP,
  USER_CREATE INTEGER,
  USER_DESTINATION INTEGER,
  USER_RESOLUTION INTEGER,
  END_DATE TIMESTAMP,
  FROM_DEP INTEGER,
  TO_DEP INTEGER,
  FINISHED BOOLEAN,
  NUM_DEADLINES INTEGER, -- QUANTIDADE DE PRAZOS
  NUM_FILE_ATTACHMENTS INTEGER, -- QUANTIDADE DE FICHEIROS ANEXADOS
  NUM_DELIVERED_ELEMENTS INTEGER, -- QUANTIDADE DE ELEMENTOS ENTREGUES
  NUM_NOT_DELIVERED_ELEMENTS INTEGER, -- QUANTIDADE DE ELEMENTOS NAO ENTREGUES
  NUM_NOTICES INTEGER, -- QUANTIDADE DE TOMADAS DE CONHECIMENTO
  NUM_SENDED_SMS INTEGER, -- QUANTIDADE DE SMS ENVIADOS
  NUM_WEB_REQUESTS INTEGER, -- QUANTIDADE DE PEDIDOS WEB
  NUM_SIGNED_OPINIONS INTEGER, -- QUANTIDADE DE PARECERES ASSINADOS
  FOREIGN KEY(PROCESS_ID) REFERENCES gsp_pedido_procs,
  FOREIGN KEY(USER_CREATE) REFERENCES gsp_users,
  FOREIGN KEY(USER_DESTINATION) REFERENCES gsp_users,
  FOREIGN KEY(USER_DESTINATION) REFERENCES gsp_users,
  CONSTRAINT UNK_WORKFLOW_STEPS UNIQUE(WORKFLOW_NUM,PROCESS_ID)
)
```

Depois da criação da tabela de factos, os dados são selecionados da tabela ROTEIROS da base de dados operacional. Nesta *query* em específico existe um *join* novamente com a tabela roteiros, com o intuito de calcular o intervalo de tempo desde a primeira ação, isto é desde a criação do processo até à ação a registar.

```
SELECT r2.dat_ocorr AS DATA_NUM1, r1.* FROM ROTEIROS r1 JOIN ROTEIROS r2
ON  r1.ped_proc_nreg_p=r2.ped_proc_nreg_p
AND r1.ped_proc_ano_p=r2.ped_proc_ano_p
AND r1.peda_pro_et_proc_cod_proc=r2.peda_pro_et_proc_cod_proc
AND r2.num=1
```

```
AND r1.dat_ocorr >= [LAST_TRANSFORMATION_TIMESTAMP]
```

```
INSERT INTO WORKFLOW_STEPS(
  WORKFLOW_NUM,
  PROCESS_ID,
  TEMPORAL_GAP,
  CREATION_DATE,
  START_DATE,
  END_DATE,
  USER_CREATE,
  USER_DESTINATION,
  USER_RESOLUTION,
  FROM_DEP,
  TO_DEP,
  FINISHED,
  NUM_DEADLINES,
  NUM_FILE_ATTACHMENTS,
  NUM_DELIVERED_ELEMENTS,
  NUM_NOT_DELIVERED_ELEMENTS,
  NUM_NOTICES,
  NUM_SENDED_SMS,
  NUM_WEB_REQUESTS,
  NUM_SIGNED_OPINIONS
)
VALUES (
:NUM,
(SELECT gpp."DW_ID" as process_id FROM gsp_pedido_procs gpp WHERE "NREG_P"=:
    PED_PROC_NREG_P AND "ANO_P"=:PED_PROC_ANO_P AND "PROC_COD_PROC" = :
    PED_PROC_PROC_COD_PROC),
(:DAT_OCORR - :DATA_NUM1),
:DATA_NUM1,
:DAT_OCORR,
:DATA_FIM,
(SELECT "DW_ID"  FROM gsp_users gu WHERE "USR_LOGIN" = :USER_INI),
(SELECT "DW_ID"  FROM gsp_users gu WHERE "USR_LOGIN" = :USERS_USR_LOGIN_DESTINO),
(SELECT "DW_ID"  FROM gsp_users gu WHERE "USR_LOGIN" = :USER_FIM),
(SELECT "DW_ID"  FROM gsp_deps gd WHERE "COD_DEP" = :DEP_COD_DEP_INICIO),
(SELECT "DW_ID"  FROM gsp_deps gd WHERE "COD_DEP" = :DEP_COD_DEP_DESTINO),
(CASE WHEN :TRANSPOR=\'S\' THEN true ELSE false END),
:QTD_PRAZOS_INS,
:QTD_FICH_ANEXADOS,
:QTD_ELEM_ENTREGUES,
:QTD_ELEM_NAO_ENTREGUES,
:QTD_TCONHECIMENTO,
:QTD_SMS_ENVIADOS,
:QTD_PEDIDOS_WEB,
:QTD_PARECERS_ASSINADOS) ON CONFLICT ON CONSTRAINT UNK_WORKFLOW_STEPS DO
```

```
44 UPDATE SET
45   END_DATE=:DATA_FIM,
46   USER_RESOLUTION=(SELECT "DW_ID"  FROM gsp_users gu WHERE "USR_LOGIN" = :USER_FIM)
         ,
47   FINISHED=(CASE WHEN :TRANSPOR=\'S\' THEN true ELSE false END),
48   NUM_DEADLINES=:QTD_PRAZOS_INS,
49   NUM_FILE_ATTACHMENTS=:QTD_FICH_ANEXADOS,
50   NUM_DELIVERED_ELEMENTS=:QTD_ELEM_ENTREGUES,
51   NUM_NOT_DELIVERED_ELEMENTS=:QTD_ELEM_NAO_ENTREGUES,
52   NUM_NOTICES=:QTD_TCONHECIMENTO,
53   NUM_SENDED_SMS=:QTD_SMS_ENVIADOS,
54   NUM_WEB_REQUESTS=:QTD_PEDIDOS_WEB,
55   NUM_SIGNED_OPINIONS=:QTD_PARECERS_ASSINADOS
```

4.6 Conclusão

Depois de aplicadas as transformações e sincronizações de dados, com recurso a um software como o Apache Superset permite-nos configurar *dashboards* e gráficos capazes de responder às necessidades dos cargos de gestão da autarquia. Essa análise dos dados de forma gráfica e dos resultados práticos do motor de BI e da modelação do DW serão apresentados no capítulo seguinte.

Capítulo 5

Análise de dados e resultados

Neste capítulo serão analisados os dados provenientes da construção do DW, sincronização e transformação de dados descritas no capítulo anterior. Os dados serão analisados com recurso a um *dashbord* e gráficos adaptados aos indicadores e métricas pretendidos.

5.1 Definição de métricas e indicadores

Para os gestores de topo de uma autarquia, isto é, o Presidente de Câmara e restante executivo municipal (vereação), bem como para as chefias dos diferentes departamentos do município é relevante perceber a *performance* do município como um todo e dos seus departamentos em particular, na agilização dos processos, como por exemplo, na resolução dos requerimentos apresentados pelos munícipes, nas contraordenações aplicadas e nos licenciamentos das obras no concelho.

Assim, para dar resposta a estas necessidades é importante definir que métricas e indicadores é que estes gestores pretendem para dar resposta às suas necessidades.

Nesta secção serão abordados alguns indicadores base que permitem monitorizar no geral a *performance* da organização.

Número de processos em aberto (a tratar)

Um indicador essencial trata-se do número de processos em aberto, isto é, os processos que existem a tratar na organização. Para obter essa resposta basta aplicar uma *query* simples à tabela de factos previamente criada e populada.

```
1 SELECT COUNT(*) FROM WORKFLOW_STEPS WHERE finished=false;
```

Exemplo de query do número de processos em aberto

Podemos assim mostrar no *dashboard*, recorrendo ao *Superset* o número de processos em aberto, selecionando todos os passos do roteiro em que *finished* é falso. Ou seja, filtramos todos os passos do roteiro onde a ação não foi concluída.

Figura 5.1: Gráfico - Nº de Processos em Aberto

Número de processos em aberto por departamento

Para além de perceber a quantidade de processos que existem por tratar na organização, é também relevante perceber o número de processos em aberto alocados a cada departamento da organização.

Conforme referido anteriormente, cada ação no *workflow* estruturado é alocada a um certo departamento que tem a responsabilidade de realizar essa mesma ação e no final tramitar o processo para outro departamento, criando um novo passo no roteiro do processo.

```
1 SELECT to_dep, desig_dep, COUNT(*) FROM workflow_steps ws
2 LEFT JOIN gsp_deps deps ON deps.dw_id = ws.to_dep
3 GROUP BY to_dep, desig_dep;
```

Exemplo de query do número de processos em aberto por departamento

Podemos assim refazer a *query* do indicador anterior e agrupar agora os dados por departamento. Fazemos um *left join* com a tabela sincronizada dos departamentos (gsp_deps), neste caso, para obter as designações dos departamentos.

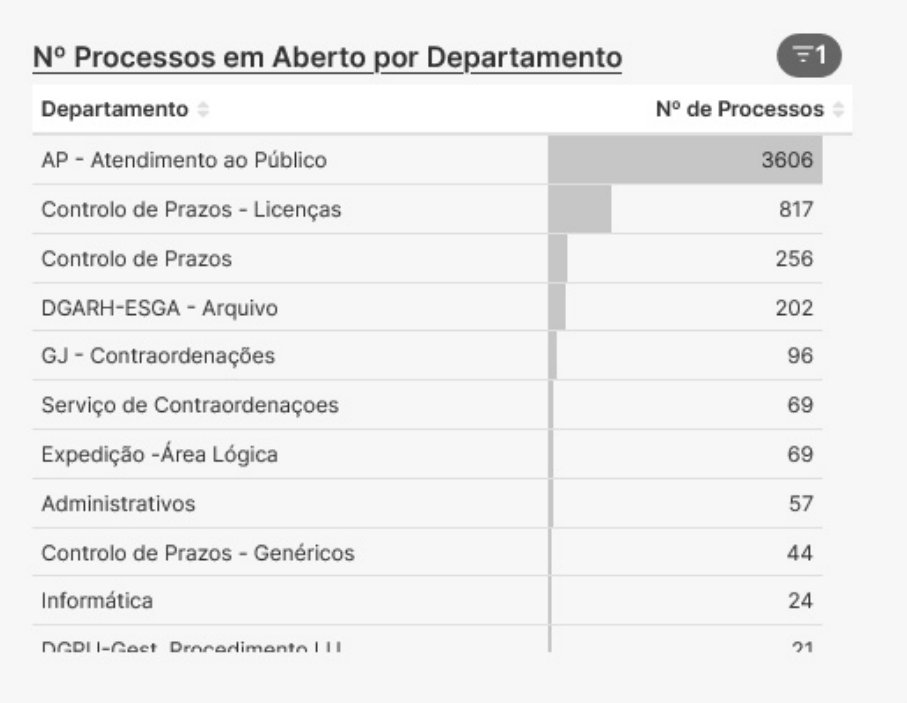
Número de processos registados (por ano/mês/dia)

É também relevante para a autarquia perceber quantos processos estão a ser registados por ano, por mês, por dia, etc. Assim, é possível perceber tendências e ajustar os recursos e fluxos de trabalho, por exemplo, para os meses em que há maior afluência de processos a dar entrada na Câmara Municipal.

```
1 SELECT COUNT(*) FROM WORKFLOW_STEPS WHERE workflow_num=1
2 GROUP BY DATE_TRUNC('year', creation_date);
```

Exemplo da query do número de processo registados por ano

Deste modo, através desta *query* obtemos os processos registados anualmente, selecionando todos os primeiros passos do roteiro (que correspondem ao registo dos processos) e agrupando-os pela unidade temporal pretendida.



Nº Processos em Aberto por Departamento

Departamento	Nº de Processos
AP - Atendimento ao Público	3606
Controlo de Prazos - Licenças	817
Controlo de Prazos	256
DGARH-ESGA - Arquivo	202
GJ - Contraordenações	96
Serviço de Contraordenaçoes	69
Expedição -Área Lógica	69
Administrativos	57
Controlo de Prazos - Genéricos	44
Informática	24
DGRH-Gest. Procedimento LLL	21

Figura 5.2: Gráfico - Nº de Processos em Aberto por departamento

Neste caso, podemos recorrer a outra abordagem, sem utilizar a tabela de factos (WORFKLOW_STEPS), ou seja selecionando os processos diretamente da tabela gsp_pedido_procs (a tabela sincronizada de todos os processos), visto que esta tabela contém a data de registo.

```
1 SELECT DATE_TRUNC('year', dat_reg) as year, COUNT(*) FROM gsp_pedido_procs
2 GROUP BY year ORDER BY year ;
```

Exemplo da query do número de processoS registados por ano através da tabela gsp_pedido_procs

A *query* pode ser ajustada mediante a granularidade temporal escolhida pelo utilizador. Este processo é feito automaticamente pelo *Superset*, conforme descrito na secção 5.3 deste capítulo.

Número de processos arquivados (por ano/mês/dia)

Para o contexto organizacional importa também perceber a quantidade de processos arquivados por granularidade temporal, pois isso indica que um determinado processo está "resolvido"e não precisa de mais atenção por parte da autarquia (deixa de estar em aberto).

```
1 SELECT DATE_TRUNC('month', dat_arq) as archive_date, COUNT(*)
2 FROM gsp_pedido_procs pp WHERE pp.estado='A'
3 GROUP BY archive_date ORDER BY archive_date;
```

Exemplo da query do número de processo arquivados por mês

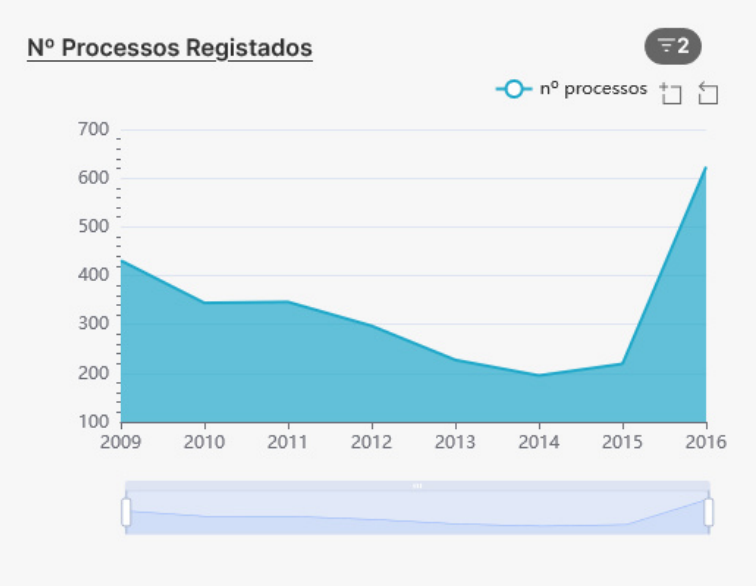


Figura 5.3: Gráfico - Nº de Processos registados por ano (entre 2009 e 2016)

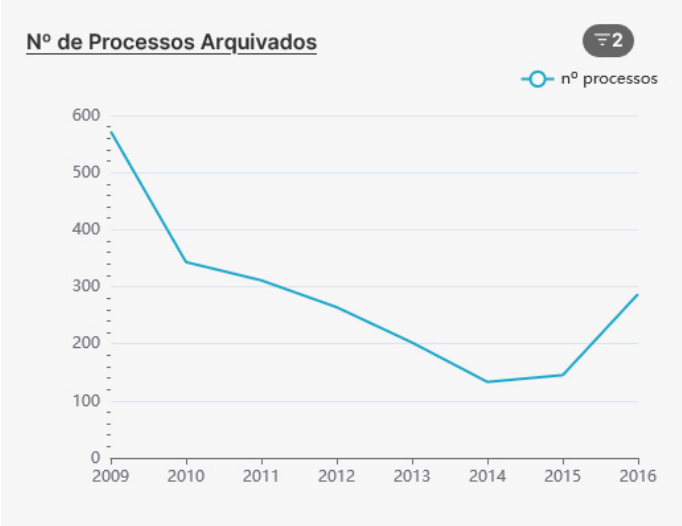


Figura 5.4: Gráfico - Nº de Processos em Arquivados por Ano (de 2009 a 2016)

Nº de ações terminadas por departamento (por ano/mês/dia)

Um dos indicadores que permite medir a *performance* de cada departamento da organização é o número de ações terminadas (tramitações de processo) por um determinado departamento. Através da variável *finished* da tabela de factos conseguimos filtrar pelas ações terminadas e juntar esses dados com o departamento responsável pela ação (to_dep).

```
1 SELECT DATE_TRUNC('year', ws.end_date) as year, to_dep, desig_dep, COUNT(*)
2 FROM workflow_steps ws LEFT JOIN gsp_deps deps on deps.dw_id = ws.to_dep
3 WHERE ws.finished=true GROUP BY to_dep, desig_dep, year ORDER BY year;
```

Exemplo da query do número de ações terminadas por ano e por departamento

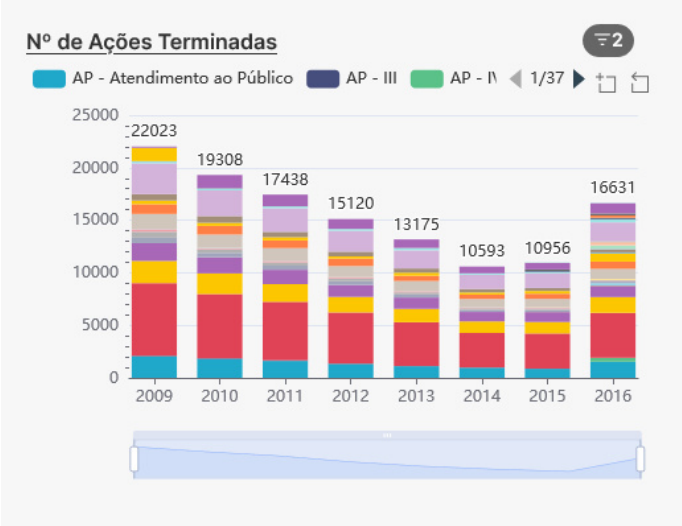


Figura 5.5: Gráfico - Nº de ações terminadas por departamento por ano (de 2009 a 2016)

Nº de ações terminadas por utilizador (por ano/mês/dia)

Do mesmo modo, podemos ir mais a detalhe e perceber a *performance* de um determinado utilizador da organização, visto que um utilizador pode pertencer a mais do que um departamento e tratar tipologias de processos diferentes.

```
1 SELECT DATE_TRUNC('year', ws.end_date) as year, to_dep, usr_desig, COUNT(*)
2 FROM workflow_steps ws LEFT JOIN gsp_users usrs on usrs.dw_id = ws.user_resolution
3 WHERE ws.finished=true GROUP BY to_dep, usr_desig, year ORDER BY year;
```

Exemplo da query do número de ações terminadas por ano e por utilizador

Ações em aberto (etapa/ação/departamento)

No contexto da gestão processual é relevante para os gestores perceber que tipo de ações estão em aberto (a tratar), isto é, em que etapa se inserem e quem é o departamento responsável por lhes dar seguimento.

Assim, através do exemplo da figura 5.7 conseguimos perceber que o departamento "AP - Atendimento ao Público"tem o maior número de ações a tratar (3513 ações) de "Entrada de Requerimento", ações essas que fazem parte da etapa "Saneamento Apreciação Liminar".

```
1 SELECT ac.desig_accao, et.desig_etapa, deps.desig_dep, COUNT(*) as nr
```

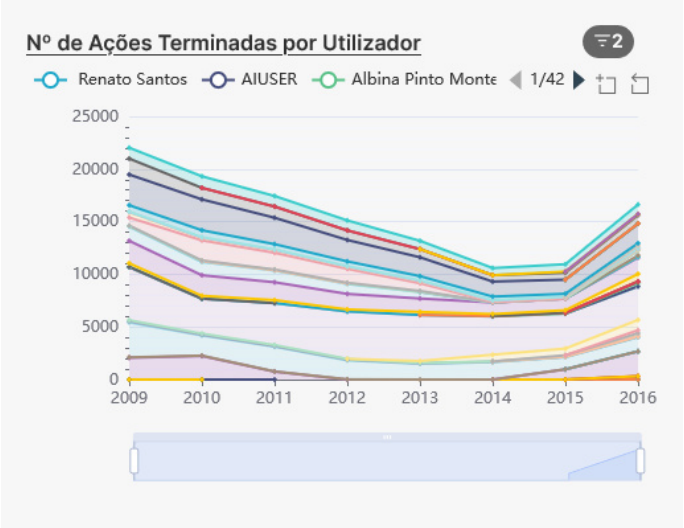


Figura 5.6: Gráfico - Nº de ações terminadas por utilizador por ano (2009 a 2016)

```
2 FROM workflow_steps ws
3 LEFT JOIN gsp_deps deps on deps.dw_id = ws.to_dep
4 LEFT JOIN gsp_accaos ac on ac.dw_id = ws.action_id
5 LEFT JOIN gsp_etapas et on et.dw_id = ws.stage_id
6 WHERE ws.finished=false GROUP BY deps.desig_dep, ac.desig_accao, et.desig_etapa
7 ORDER BY nr DESC;
```

Exemplo da query das ações em aberto

Ações em Aberto

1

Search 191 records...

Etapa	Ação	Departamento	Nº de Processos
Saneamento Apreciação Liminar	Entrada de Requerimento	AP - Atendimento ao Público	3513
Emissão Alvará Op. Urbanística	Prazo-Aguarda comun. início da obra	Controlo de Prazos - Licenças	509
Visitas Técnicas	Prazo-Decurso da obra	Controlo de Prazos - Licenças	228
Arquivo	Arquivo	DGARH-ESGA - Arquivo	202
Emissão de Aditamento Alvará	Prazo-Decurso da obra	Controlo de Prazos - Licenças	65
Apreciação do Pedido	Entrada de Requerimento	AP - Atendimento ao Público	64
Aprec. Projecto Arquitectura	Prazo-Aguarda proj. especialidades	Controlo de Prazos	57
Instrução	Registo do Auto de Notícia	Serviço de Contraordenaçoes	57
Notificações	Informação ao Processo	GJ - Contraordenações	45

Figura 5.7: Gráfico - Ações em Aberto

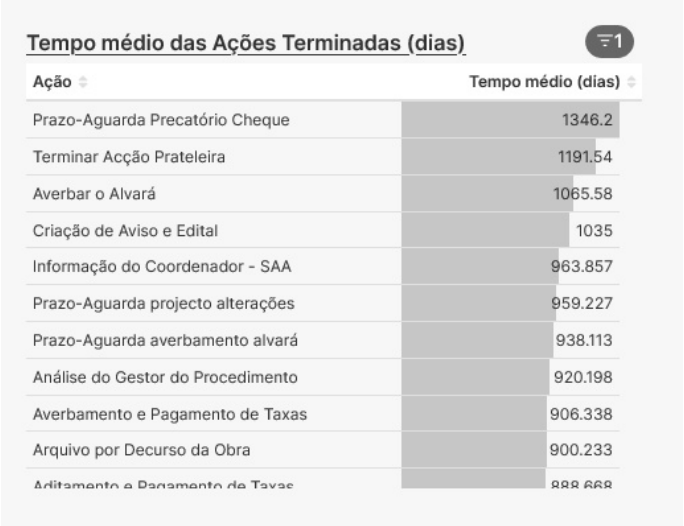
Tempo médio das ações terminadas

Para além de percebermos em que departamento existem mais ações a tratar, é também relevante perceber quais as ações onde se demora mais tempo a dar seguimento (em dias).

Este indicador permite ao município gerir melhor os seus recursos, nomeadamente considerar dar mais atenção aos procedimentos a executar e aos recursos humanos envolvidos nas ações que demoram mais tempo a dar seguimento.

```
1 SELECT ac.desig_accao, ROUND(AVG(DATE_PART('day', start_date-creation_date))) as
      avg_days
2 FROM workflow_steps ws
3 LEFT JOIN gsp_accaos ac on ac.dw_id = ws.action_id
4 WHERE ws.finished=true
5 GROUP BY ac.desig_accao
6 ORDER BY avg_days DESC;
```

Exemplo da query do tempo médio das ações



Tempo médio das Ações Terminadas (dias)

Ação	Tempo médio (dias)
Prazo-Aguarda Precatório Cheque	1346.2
Terminar Acção Prateleira	1191.54
Averbar o Alvará	1065.58
Criação de Aviso e Edital	1035
Informação do Coordenador - SAA	963.857
Prazo-Aguarda projecto alterações	959.227
Prazo-Aguarda averbamento alvará	938.113
Análise do Gestor do Procedimento	920.198
Averbamento e Pagamento de Taxas	906.338
Arquivo por Decurso da Obra	900.233
Aditamento e Pagamento de Taxas	888.668

Figura 5.8: Gráfico - Tempo médio das ações terminadas (dias)

5.2 Construção de *Dashboards*

Através dos gráficos gerados através das métricas e indicadores previamente descritos é possível construir *dashboards* (painéis) capazes de agregar informação e permitir uma visualização comparativa entre os diferentes indicadores.

Com recurso ao *Superset* é possível configurar e personalizar o *dashboard* à medida, redimensionando o tamanho dos gráficos, agrupando gráficos por *tabs* e/ou secções, criando divisórias e títulos.

Na figura 5.9 podemos visualizar o *dashboard* construido que agrega os diferentes indicadores considerados no âmbito desta dissertação.

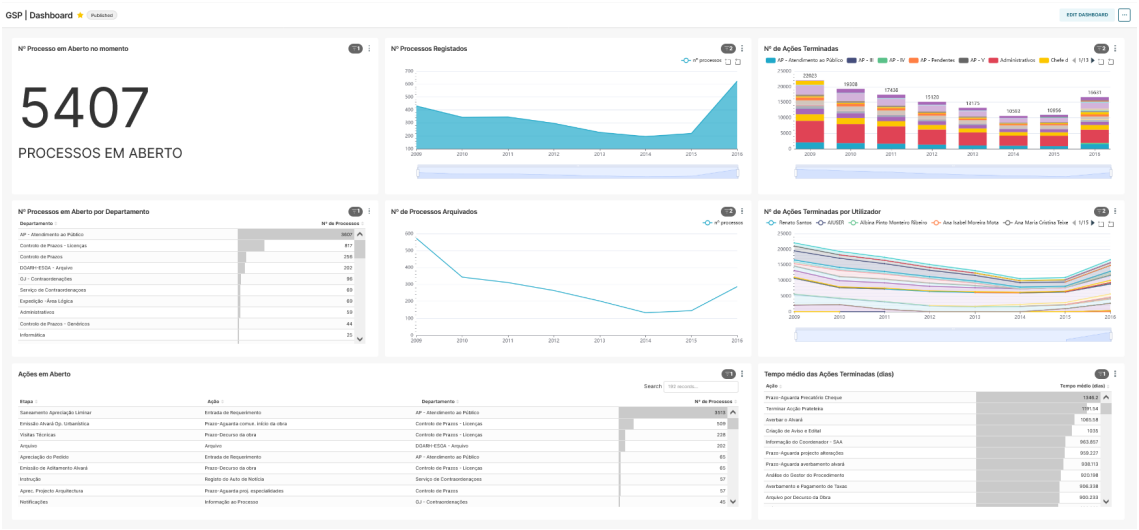


Figura 5.9: Dashboard - GSP

5.3 Filtros dinâmicos

O *Superset* disponibiliza também como funcionalidade a possibilidade de aplicar filtros aos gráficos, num *dashboard*. Assim é possível, por exemplo, alterar a granularidade temporal dos gráficos, o intervalo de tempo que queremos considerar, filtrar por uma tipologia de processo ou até por uma etapa ou ação específica.

Assim que aplicamos os filtros, os gráficos considerados atualizam-se ajustando automaticamente os seus dados.

Na figura 5.10 a granularidade temporal foi ajustada para visualizar os gráficos por mês num intervalo de tempo de 01-01-2009 a 31-12-2009. Do mesmo modo, aplicou-se o filtro para considerar apenas processos de "Licenças de Obras de Edificação", tratados pelo departamento "Administrativos".

Ao primeiro gráfico, intitulado "Nº de Processos em Aberto no momento"apenas é aplicado o filtro "Tipo de Processo". Assim podemos afirmar que existem 617 processos de "Licenças de Obras de Edificação"em aberto na organização.

Já ao segundo gráfico, "Nº de processos em Aberto por Departamento"foram aplicados dois filtros: "tipo de processo"e "tratado por". Deste modo, conseguimos visualizar o número de processos de "Licenças de Obras de Edificação"em aberto, a tratar pelo departamento "Administrativos". Aos gráficos "Nº de Ações em Aberto"e "Tempo médio das Ações Terminadas (dias)"são aplicados os mesmos filtros, permitindo assim aos gestores avaliar em que ações e etapas estão os processos de "Licenças de Obras de Edificação"a tratar pelo departamento "Administrativos", bem como o tempo médio (em dias) que cada ação costuma demorar.

Relativamente ao gráfico "Nº de processos registados"este não apresenta valores, uma vez que o departamento "Administrativos"não efetua o registo de processos.

Aos gráficos "Nº de Processos Arquivados", "Nº de Ações Terminadas"e "Nº de Ações Terminadas por Utilizador"foram aplicados os filtros: "tipo de processo", "tratado por"(departamento),

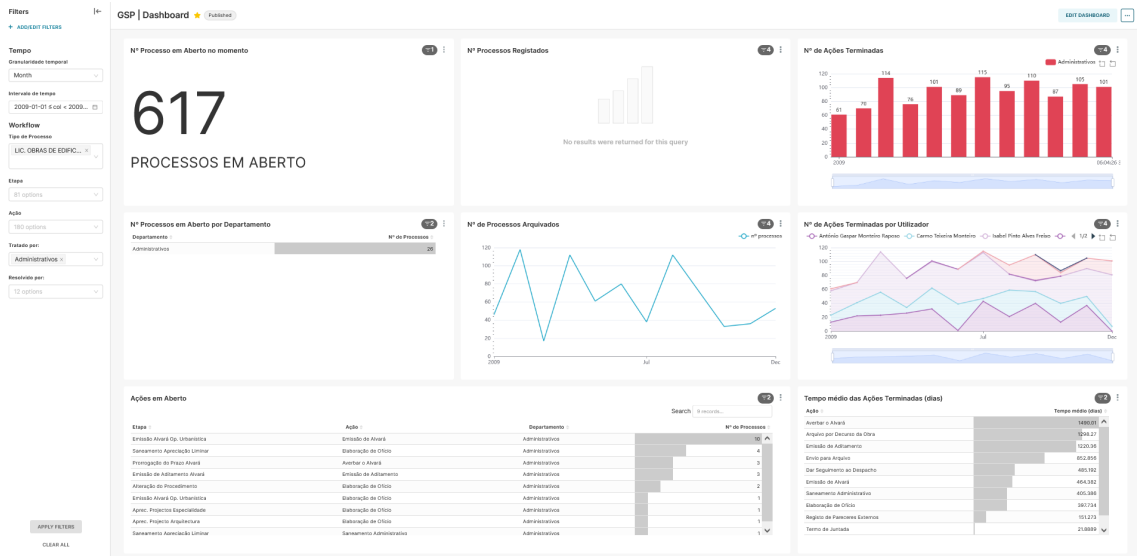


Figura 5.10: Exemplo *dashboard* com filtros aplicados - GSP

"granularidade temporal"e "intervalo de tempo". Neste caso, os dados considerados remetem para processos de "Licenças de Obras de Edificação", tratados pelo departamento "Administrativos"durante o ano de 2009 (visualização por mês).

Particularmente no gráfico "Nº de Ações Terminados por Utilizador", é ainda possível observarmos o nº de ações terminadas por cada funcionário do departamento "Administrativos".

5.4 Listagens

Por vezes, os gestores pretendem perceber em detalhe que processo ou processos estão numa ação em aberto há mais tempo. Com recurso a uma listagem com pesquisa e filtros dinâmicos conforme descrito na secção 5.3 (por tipologia, por ano do processo, por nº de dias em aberto, ...) é possível visualizar os processos em concreto para que o gestor possa analisar no software de gestão processual qual a motivação para o processo em análise estar em aberto.

Também com recurso ao *Superset* é possível utilizar formatação condicional para indicar através de cores a duração dos processos em aberto. No exemplo apresentado na figura 5.11, é aplicada uma formatação condicional, onde a verde são apresentados os processos cujo número de dias em aberto é inferior a 60 dias, a amarelo entre 60 e 90 dias e a vermelho mais de 90 dias em aberto.

Figura 5.11: Listagem de Processos

Capítulo 6

Conclusões e trabalho futuro

Nesta dissertação, foi desenvolvida e implementada uma solução de BI voltada para a avaliação da *performance* dos municípios e dos seus departamentos. Através da integração de fontes de dados e da aplicação de um mecanismo de processamento de dados demonstramos como o BI pode proporcionar uma visão abrangente e detalhada dos fluxos dos processos da autarquia, facilitando a tomada de decisões informadas e a otimização de recursos do município.

A implementação da solução permite assim identificar os departamentos onde a *performance* corresponde às expectativas dos gestores e em que áreas e departamentos deve haver investimento com o intuito de melhorar os serviços prestados aos munícipes. A capacidade de monitorizar indicadores-chave de desempenho (KPIs), diariamente, possibilita uma gestão mais eficaz e proativa, além de promover a transparência e a prestação de contas. Para além disso, a solução facilita a identificação de padrões e tendências, fornecendo *insights* valiosos para o planeamento das políticas públicas e a alocação eficiente de recursos.

Os resultados obtidos reforçam a importância do uso de tecnologias de BI no contexto da administração pública. A aplicação de metodologias de análise de dados pode transformar a forma como os municípios operam, promovendo uma gestão mais eficiente, transparente e centrada no cidadão. Além disso, a disseminação desta abordagem pode contribuir significativamente para o desenvolvimento sustentável e equilibrado da eficácia dos processos municipais.

6.1 Satisfação dos objetivos

Podemos afirmar que os objetivos propostos foram alcançados. A solução de BI desenvolvida e implementada para avaliar a *performance* dos municípios e dos seus diversos departamentos mostrou-se eficaz. Conseguimos criar uma solução capaz de responder às necessidades específicas dos gestores municipais.

Esta dissertação não apenas atendeu aos objetivos técnicos e operacionais propostos, mas também evidenciou o potencial impacto positivo de uma abordagem baseada em BI na administração pública municipal.

6.2 Trabalho Futuro

Com base nos resultados alcançados no desenvolvimento desta dissertação, podemos identificar melhorias que podem ser consideradas como trabalho futuro, nomeadamente, aprimorar e expandir a solução de BI FutureSense.

Primeiramente, um dos objetivos principais para o desenvolvimento futuro do FutureSense é torná-lo mais acessível para utilizadores com menos conhecimento técnico, especialmente em relação ao uso de SQL. Para isso, é necessário implementar funcionalidades que permitam aos utilizadores selecionar tabelas e associar dados de forma intuitiva e gráfica, eliminando a necessidade de escrever código SQL. Isso poderá ser alcançado através de interfaces e assistentes passo a passo, facilitando a análise de dados para todos os níveis de utilizadores.

Além disso, a integração de novas fontes de dados e a interoperabilidade com outros softwares utilizados pelos municípios, como os sistemas de Gestão de Recursos Humanos (RH) e Planeamento de Recursos Empresariais (ERP), são prioridades importantes. A inclusão de mais fontes de dados permitirá uma visão ainda mais abrangente das operações municipais.

Outro avanço pode passar pelo desenvolvimento da solução de sincronização em tempo real. Isso permitirá que os dados sejam atualizados instantaneamente, fornecendo aos gestores municipais informações sempre atuais e relevantes para a tomada de decisões rápidas e eficazes. A implementação dessa funcionalidade exigirá a construção de um mecanismo robusto de atualização contínua de dados e a garantia de que o desempenho do sistema permaneça eficiente.

Para além disto é também importante destacar que esta dissertação terá continuidade, na medida em que está em curso outra dissertação que implementará uma solução inovadora com recurso a assistentes virtuais, que irá integrar com o FutureSense.

Esta integração permitirá que os utilizadores interajam com o sistema de forma ainda mais intuitiva e eficiente, utilizando comandos de linguagem natural para obter informação.

Em suma, os próximos passos para o desenvolvimento do FutureSense incluem tornar a ferramenta mais *user-friendly* para utilizadores com diferentes níveis de conhecimento técnico, integrar novas fontes de dados, implementar um mecanismo de sincronização em tempo real e integrar assistentes virtuais para uma interação mais intuitiva. Acreditamos que essas melhorias não só aumentarão a eficácia e a eficiência do FutureSense, mas também maximizarão o seu impacto positivo na gestão municipal, proporcionando benefícios tangíveis tanto para os gestores quanto para a população.

Referências

Daniel Adetunji. How docker containers work – explained for beginners, 2023. URL https://www.freecodecamp.org/news/how-docker-containers-work/.

ANO. Ano software. Available at https://ano.pt/about/, last accessed in January 2024.

Microsoft Azure. O que é o postgresql? | microsoft azure. URL https://azure.microsoft.com/pt-pt/resources/cloud-computing-dictionary/what-is-postgresql.

Paulo Clemente. Introdução ao next.js - um framework para desenvolvedores react, 2023. URL https://blog.rocketseat.com.br/introducao-ao-next-js/.

Docker. Why docker | docker. URL https://www.docker.com/why-docker/.

Apache Software Foundation. Introduction | superset. URL https://superset.apache.org/docs/intro.

Vishal Goar, Prof S Sarangdevot, Govind Tanwar e Dr Anand Sharma. Improve performance of extract, transform and load (etl) in data warehouse. *International Journal on Computer Science and Engineering*, 2, 05 2010.

IBM. O que é o docker? | ibm, a. URL https://www.ibm.com/br-pt/topics/docker.

IBM. O que é java spring boot? | ibm, b. URL https://www.ibm.com/br-pt/topics/java-spring-boot.

Claudia Imhoff, Nicholas Galemmo e Jonathan G Geiger. *Mastering Data Warehouse Design - Relational and Dimensional Techniques*. 2003. ISBN 0471324213.

Liquibase Inc. Introduction to liquibase. URL https://docs.liquibase.com/concepts/introduction-to-liquibase.html.

W. H. Inmon. Building the data warehouse third edition, 2002.

Ralph Kimball e Margy Ross. The data warehouse toolkit, 2013.

Andrei L. Guia para iniciantes: O que é nginx e como funciona, 6 2023. URL https://www.hostinger.com.br/tutoriais/o-que-e-nginx.

Andy Morris. 23 real-world examples of business intelligence | netsuite, 4 2021. URL https://www.netsuite.com/portal/resource/articles/business-strategy/business-intelligence-examples.shtml.

David Parmenter. *Key Performance Indicators*. Wiley, 10 2019. ISBN 9781119620778. doi: 10.1002/9781119620785.

Alejandro Vaisman e Esteban Zimányi. *Data Warehouse Systems*. Springer Berlin Heidelberg, 2014. ISBN 978-3-642-54654-9. doi: 10.1007/978-3-642-54655-6.

Alejandro Vaisman e Esteban Zimányi. *Data Warehouse Systems*. Springer Berlin Heidelberg, 2022. ISBN 978-3-662-65166-7. doi: 10.1007/978-3-662-65167-4.