NOVA IMS Information Management School

MAA

Mestrado em Métodos Analíticos Avançados
Master Program in Advanced Analytics

Central Bank Digital Transformation: Building an Environment through Data

An ETL Approach

Tomás Ferreira de Sá

Relatório de Estágio apresentado como requisito parcial para obtenção do grau de Mestre em Métodos Analíticos Avançados

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

CENTRAL BANK DIGITAL TRANSFORMATION: BUILDING AN ENVIRONMENT THROUGH DATA

AN ETL APPROACH

por

Tomás Ferreira de Sá

Relatório de Estágio apresentado como requisito parcial para a obtenção do grau de Mestre em Métodos Analíticos Avançados

Orientador: Roberto Henriques

Coorientador: Pedro Calheiros Souto

Outubro 2022

Dedicado aos meus Pais, por todo o apoio, e por sempre me terem proporcionado as ferramentas necessárias para atingir este objetivo.

AGRADECIMENTOS

Em primeiro lugar, gostaria de agradecer ao Banco de Portugal, pelo nome de Pedro Calheiros Souto, Ricardo Oliveira Sousa e André Vicente Monteiro, por me terem recebido de forma tão especial, e por todo o apoio e suporte dado durante o estágio. Foi um prazer enorme e voltava a repetir tudo exatamente da mesma forma.

Uma palavra também aos meus amigos e família, pela motivação nos momentos mais difíceis, e à minha namorada, Beatriz do Ó, por garantir que estava sempre no caminho certo.

Uma palavra de agradecimento também ao Professor Roberto Henriques, por me ter acompanhado ao longo deste percurso, e por ter contribuído com todo o conhecimento científico.

RESUMO

Este relatório descreve a construção de um processo ETL como estrutura de suporte, face às necessidades de desenvolvimento tecnológico do Departamento de Averiguação e Ação Sancionatória, do Banco de Portugal. Da criação deste ETL resultaram três aplicações computacionais com finalidades distintas, e que vieram resolver problemas de extração, análise e automação de dados. O ETL foi criado com recurso à *framework* Django, tendo sido utilizado *python* e *SQL* no seu desenvolvimento. As três principais fases do processo ETL foram distribuídas de forma homogénea ao longo de todo o projeto.

A realização deste projeto abriu portas à criação de um ecossistema de inovação que servirá de base ao desenvolvimento de outras aplicações computacionais, e que vai ao encontro de um dos principais objetivos do Banco de Portugal para os próximos três anos: realizar uma profunda transformação digital.

PALAVRAS-CHAVE

Ciência dos Dados; ETL; Python; Django; Transformação Digital

ABSTRACT

This report describes the construction of an ETL process as a support structure in the face of the technological development needs of the Legal Enforcement Department of the Bank of Portugal. The creation of this ETL resulted in three computational applications with different purposes, which solved extraction, analysis, and data automation problems. The ETL was created using the Django framework, having utilized both python and SQL in its development. The three main phases of the ETL process were homogeneously distributed throughout the entire project.

The realization of this project opened the door to the creation of an innovation ecosystem that will serve as a base for the development of other computational applications, and which will meet one of Bank of Portugal's main objectives for the next three years: to carry out a profound digital transformation.

KEYWORDS

Data Science; ETL; Python; Django; Digital Transformation

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

ÍNDICE DE ANEXOS

LISTA DE SIGLAS E ABREVIATURAS

DAS Departamento de Averiguação e Ação Sancionatória

AAS Área de Ação Sancionatória

AFI Atividade de Averiguação da Atividade Financeira Ilícita

ASB Área de Supervisão Preventiva do Branqueamento de Capitais e Financiamento ao Terrorismo

AII Área de Intervenção Institucional

UPC Unidade de Planeamento e Controlo

FTE Full Time Equivalent

ETL *Extract, Transform & Load*

SQL *Structured Query Language*

RPS *Repositório de Processos e Sanções*

1. INTRODUÇÃO

A transformação digital é uma realidade cada vez mais presente no setor bancário [21], e um banco central tem o dever de acompanhar essa evolução. De forma a ter a melhor capacidade de supervisão possível e a estar sempre atualizado, é crucial construir um ambiente digital propício a este tipo de iniciativas. Uma instituição com mais de 175 anos de existência acaba por ter algumas lacunas a nível tecnológico que limitam o seu trabalho diário, e que devem ser resolvidas o mais depressa possível. A falta de digitalização dentro de um departamento leva a falhas na comunicação entre as equipas, e a que muito trabalho tenha de ser feito de forma repetida, por ação humana. A utilização do valor existente nos dados permite ao departamento reduzir o número de horas despendidas a realizar determinadas tarefas, maximiza a quantidade de material utilizado em formato digital, e permite o aumento de processos averiguados anualmente sem que para isso seja necessário recorrer a mais recursos humanos.

Já existem as fontes de dados necessárias para que haja criação de valor ao departamento. O passo seguinte foi identificar uma forma de resolver os problemas anteriormente referidos, através de um projeto que englobe todas as soluções: a criação de um processo ETL. Uma vez que a fonte de dados é alimentada com informação sobre os processos e que é constantemente atualizada pelos colaboradores, o foco está em criar um fluxo por onde esses mesmos dados passem, sejam transformados, e fiquem orientados ao resultado pretendido.

Para facilitar a identificação dos requisitos que o processo tem de ter, o resultado final do ETL foi dividido em três aplicações, sendo que cada uma delas resolve um problema distinto dentro do departamento. De modo a facilitar a extração de dados e a ser feito de forma totalmente autónoma, foi criado o DAStat – uma ferramenta self-service de extração de dados. Para que haja uma melhor comunicação dentro do departamento e para que seja possível obter informação sobre o estado dos processos em curso, foi criado o DASboard – um conjunto de dashboards com determinados indicadores e métricas. Para que a realização de tarefas repetidas seja evitada ao máximo, foi criado o File Automation – uma aplicação que gera ficheiros preenchidos de forma automática. Estas três aplicações foram disponibilizadas no portal do departamento, que é acessível por todos os colaboradores.

De maneira a facilitar a gestão dos dados ao longo de todo o percurso e a sua forma de representação, foi utilizada a *framework* Django. Uma ferramenta que permite o desenvolvimento de aplicações *web* de forma rápida, segura e que estejam orientadas ao uso de dados de forma dinâmica. O departamento já utilizava esta ferramenta antes do início deste estágio, pelo que o pressuposto da sua utilização para a construção do ETL era um requisito obrigatório cuja decisão não estive envolvido.

1.1. CONTEXTO EMPRESARIAL

O estágio profissional do presente relatório foi realizado no Departamento de Averiguação e Ação Sancionatória (DAS) do Banco de Portugal onde integrei a Unidade de Planeamento de Controlo (UPC). Uma unidade transversal a todo o departamento e que presta suporte às quatro áreas de atuação do departamento – Área de Ação Sancionatória (AAS), Área de Averiguação da Atividade Financeira Ilícita

(AFI), Área de Intervenção Institucional (AII) e Área de Supervisão Preventiva do Branqueamento de Capitais e Financiamento ao Terrorismo (ASB). O Departamento existe desde 2011, e tem como missão o desenvolvimento de ações de natureza reativa ou contraordenacional, conducentes ao cumprimento, pelas entidades supervisionadas, das normas ou das determinações a que estão obrigadas e que, por qualquer motivo, não estão a ser observadas ou integralmente satisfeitas. Deve também assegurar a supervisão em matéria de prevenção do branqueamento e do financiamento do terrorismo.

1.1.1. História e Papel na Sociedade

O Banco de Portugal, fundado a 19 de novembro de 1846 após fusão do Banco de Lisboa com a Companhia Confiança Nacional, tem o papel de Banco Central da República Portuguesa. Foi nacionalizado em 1974 tendo, até então, o estatuto de sociedade anónima e sendo maioritariamente privado. É responsável pela emissão de notas de euro e coloca em circulação moedas metálicas com a devida autorização do Banco Central Europeu. É também responsável pela supervisão prudencial das instituições de crédito e sociedades financeiras, e compete-lhe regular, fiscalizar e promover o bom funcionamento dos sistemas de pagamentos. Age como intermediário das relações monetárias internacionais do Estado, e é também da sua responsabilidade, a elaboração de estatísticas monetárias, financeiras e cambiais, e o aconselhamento do Governo nos domínios económico e financeiro. [1]

1.1.2. Enquadramento Atual

O Banco de Portugal tem a robustez do sistema financeiro como objetivo estratégico atual nº1. Para se adaptar à complexificação do mercado, está a passar por uma transformação digital em todas as áreas de negócio, através da adaptação de novas tecnologias.

Com os seus 175 anos de existência, o Banco de Portugal tem a obrigação de estar constantemente a adaptar-se às necessidades exigidas pelo sistema financeiro europeu e mundial, e a repensar a sua estratégia. Ora estas premissas levam a que alterações constantes tenham de ser feitas nos seus departamentos, seja nos fluxos dos processos, seja na capacitação dos seus funcionários para se adaptarem às necessidades que vão surgindo. Necessidades essas que passam pela transformação da informação de papel para o mundo digital, da sua gestão e da criação de valor a partir da mesma.

Em linha com uma das principais prioridades estratégicas do Banco de Portugal para 2025 - “Reforçar a capacidade tecnológica e digital do Banco: identificar oportunidades de automação, desenvolvendo projetos com as áreas de negócio.” -, é muito importante que seja construída uma estrutura sólida onde a informação vai assentar, para que a possamos utilizar no desenvolvimento de projetos concretos. Esta estrutura não é construída de um dia para o outro e, como tal, é um processo em constante desenvolvimento e que requer adaptações recorrentes em termos de tecnologias e ferramentas a serem utilizadas. [2]

2. ENQUADRAMENTO TEÓRICO

Neste capítulo, é apresentado o enquadramento teórico dos conceitos e atividades realizadas durante o estágio e tem como objetivo explicar todo o plano de fundo do que foi implementado e as suas motivações.

Tal como disse Foster Provost e Tom Fawcett (2013), a Ciência dos Dados é "um conjunto de princípios fundamentais que apoiam e orientam a extração baseada em princípios de informação e conhecimento a partir dos dados". É possível melhorar a tomada de decisões em qualquer negócio com base na informação extraída dos dados, e de forma mais responsável e baseada em factos reais (ver figura 1). Referem também que "a Ciência dos Dados envolve princípios, processos e técnicas para a compreensão de fenómenos por meio da análise (automatizada) de dados". Todas essas metodologias, quando utilizados em conjunto e de forma pró-ativa, criam mais valor a partir dos dados e aumentam a performance de qualquer instituição. [3]

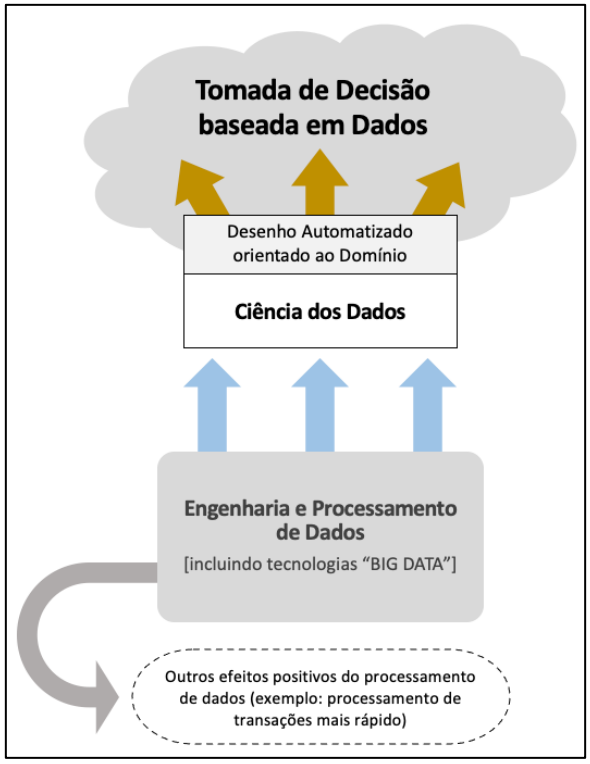


Figura 1 – Relação entre Ciência dos Dados e a Tomada de Decisão [4]

2.1. ETL – EXTRACT, TRANSFORM & LOAD

O ETL é um processo de integração de dados que está dividido em cinco fases: Extrair, Limpar, Transformar, Carregar e Analisar. Destas cinco, as fases mais relevantes são Extrair, Transformar e Carregar.

Na primeira fase, os dados são extraídos das diversas origens consideradas relevantes ao processo, de onde passam para a fase de Transformação. Nesta fase, são aplicados os métodos necessários aos dados para que estes fiquem adaptados às regras estabelecidas e, dependendo do objetivo pretendido, são carregados e armazenados em vários locais (geralmente em *Data Warehouses*, mas podem ter outros destinos, como ficheiros Excel). [5]

O processo de ETL resulta em múltiplos benefícios para qualquer negócio [7], como:

- **Contexto**: o ETL permite aos negócios, através dos dados, ter um ganho de contexto histórico;
- **Consolidação**: fornece uma visão dos dados mais consolidada e que permite uma melhor análise e comunicação dos dados;
- **Produtividade**: melhora a produtividade com a automatização de processos de negócio;
- **Precisão**: permite a melhoria da precisão dos dados e da capacidade de auditorias, muitas vezes necessários para os negócios estarem em conformidade com normas e regulamentos.

Em cada uma das três fases de um ETL, que são possíveis observar na figura 2, são aplicadas várias áreas da Ciência dos Dados, para que seja possível gerar o maior valor possível a partir dos dados.

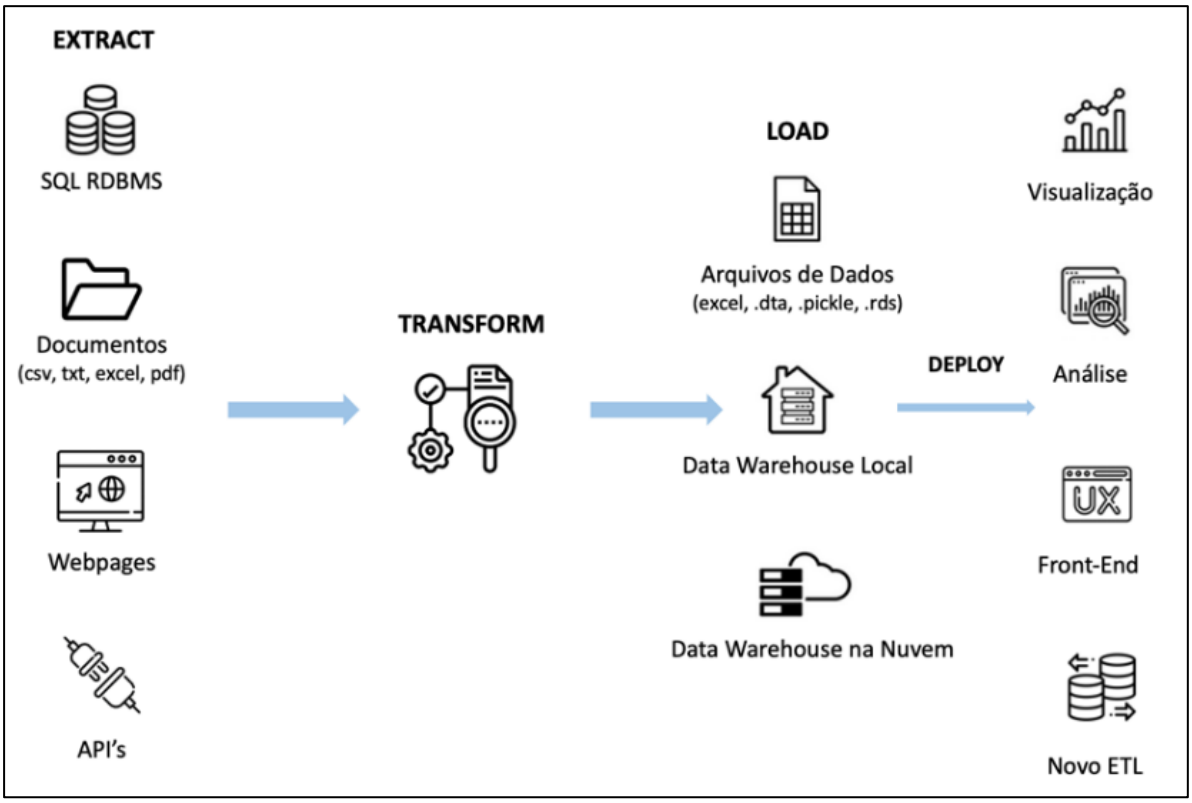


Figura 2 – Arquitetura de um processo ETL [6]

2.1.1. Extract

2.1.1.1. Extração de Dados

A extração é a primeira fase no processo de ingestão de dados (ETL). Envolve extrair a informação das fontes (exemplo: *SQL* ou *NoSQL servers*, *CRM*, sistemas *ERP*, *APIs*, *emails*, ficheiros simples, *webpages*, etc), e exportá-la para uma área de preparação temporária (*staging area*). Esta área de preparação permite fazer várias validações aos dados extraídos, antes de os mover para o *Data Warehouse*. A verificação do tipo de dados, a remoção de duplicados ou de dados fragmentados, reconciliação dos registos com os dados de origem e a certificação de que dados não desejados não sejam carregados são exemplos disso.

Neste processo, os dados podem ser extraídos de forma total e completa, ou parcial. Se não houver qualquer forma de identificar registos que tenham sido alterados, a única solução viável é extrair novamente todas as tablas da fonte dos dados, e assim garantir que os dados estão atualizados. Claro que deve ser evitado sempre que possível, uma vez que, ao extrairmos os dados na totalidade, estamos a envolver transferências de grandes volumes de dados e a sobrecarregar a rede. Se forem extraídos de forma parcial, o processo pode ser feito através da extração incremental, ou de uma notificação de atualização (*update notification*). A extração incremental é utilizada quando algumas fontes de dados não são capazes de gerar notificações ao haver atualizações, mas são capazes de identificar que registos foram modificados, e fornecem um extrato desses mesmos registos que mais tarde são propagados ao longo de todo o processo de ETL. A principal desvantagem da utilização deste método é não ser capaz de detetar registos apagados na fonte dos dados, uma vez que esses registos, por já terem sido removidos, já não se encontram na base de dados. O método de Notificação de Atualização é a forma mais fácil de extrair os dados da sua fonte, visto que é gerada uma notificação sempre que um registo seja alterado, facilitando o processo e recorrendo a menos recursos.[8]

2.1.2. Transform

2.1.2.1. Modelação de Dados

A Modelação de Dados é o primeiro passo crítico para a definição da estrutura dos dados que qualquer sistema vai receber e produzir, e é nesta fase que são definidas as relações e as associações entre os diversos tipos de dados. Conceptualmente, os dados são representados através de diagramas, símbolos e legendas, e permitem a interpretação da sua relação e clarificação dos requisitos, para que um qualquer sistema funcione da forma mais eficiente possível. Também é crucial que haja escalabilidade com a introdução de novas funcionalidades, com o mínimo de reestruturação dos modelos já existentes. Estas representações podem ser partilhadas com outras áreas de negócio sem conhecimento técnico, e permite uma melhor análise dos dados a partir de diferentes perspetivas. Dependendo do objetivo pretendido, há vários tipos de modelos de dados que podem ser utilizados, como explicado na tabela 1.

Tipos de Modelos de Dados

Conceptual	Representação visual dos conceitos e relações entre entidades de alto-nível. Em vez de se focar nos detalhes da base de dados, foca-se no estabelecimento de classes de entidade com as suas características e restrições e as relações entre si. A notação utilizada é considerada simples.
Lógico	Menos abstrato e providencia mais detalhe sobre os conceitos e relações do que está a ser analisado, e onde é utilizada uma notação formal de modelação de dados específica. Neste tipo de modelo são apontados atributos dos dados, como o seu tipo e tamanho. Os modelos lógicos não especificam qualquer requisito técnico do sistema e, geralmente, são desenvolvidos para projetos específicos, uma vez que o seu propósito é o desenvolvimento do mapa técnico de regras e estruturas dos dados.
Físico	O modelo físico consiste no esquema de como é que os dados vão ser fisicamente armazenados na base de dados e, como tal, é o modelo menos abstrato dos três. Apesar de muito idêntico ao modelo lógico, o design final do modelo físico pode ser implementado diretamente como base de dados relacional, e pode incluir propriedades de sistemas de gestão de bases de dados (DBMS)

Tabela 1 – Tipos de Modelos de Dados [9]

Este planeamento e modelação ajuda ao aumento da consistência nas nomenclaturas e nas regras de semântica utilizadas. Garante também mais segurança na manipulação dos dados, uma vez que, ao haver uma uniformização das regras e políticas governamentais dos dados, todos os colaboradores que os utilizem estarão a respeitas regras padronizadas.

Dependendo do objetivo pretendido, há vários modelos que organizam e mostram os dados de forma diferente (ver tabela 2).

Exemplos de Modelos de Dados

Entidade-Relacionamento	Utiliza entidades reais, demonstra as relações entre si e contém o conjunto de entidades, relações, atributos gerais e restrições.
Hierárquico	Organiza os dados na forma de uma árvore com uma raiz e a partir de onde os outros dados estão conectados. Representa uma relação de um-para-muitos entre dois tipos de dados.
Relacional	É o modelo mais popular e organiza os dados em tabelas, onde as colunas representam os vários atributos dessa entidade concreta, e as linhas a informação registada nesses campos. Faz também com que a relação entre pontos de dados seja mais fácil.
Dimensional	Desenhado para otimizar a velocidade de extração de dados do *Data Warehouse* para análise. Enquanto os modelos relacional e entidade-relacionamento garantem um armazenamento eficiente, o modelo dimensional aumenta a redundância para que seja mais fácil localizar a informação para extração.
Base de Dados orientada a objetos	Os objetos são agrupados em classes hierárquicas, têm funcionalidades associadas, conseguem incorporar tabelas e ainda relações entre dados mais complexas.

Tabela 2 – Exemplos de Modelos de Dados

2.1.2.2. Pré-Processamento de Dados

O pré-processamento de dados é uma etapa muito importante no processo de transformação dos dados, pois impede que dados irregulares sejam carregados e utilizados de forma a produzirem resultados imprecisos. Os problemas mais comuns são: informação incompleta, duplicada e com formatos inconsistentes. Estas adversidades podem ser originalmente causadas pela introdução de dados com erros por parte de utilizadores, por corrupção na transmissão ou armazenamento destes ou até por diferentes definições de entidades semelhantes em locais de armazenamento distintos.

O pré-processamento de dados engloba os seis passos representados na figura 3, sendo que só alguns foram abordados durante o estágio e, consequentemente, neste enquadramento teórico.

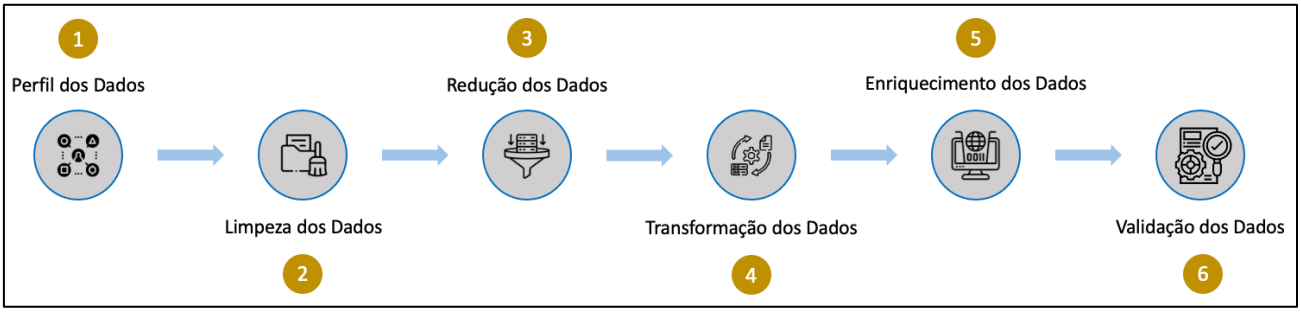


Figura 3 – Principais passos no pré-processamento de dados [10][11]

Integração de Dados

A integração de dados é aplicada quando há a necessidade de combinar dados de várias fontes distintas e, portanto, com estruturas distintas, onde temos de ter em atenção:

- **Integração de esquemas -** integração de metadados de fontes diferentes;
- **Problemas de identificação de entidades -** identificar entidades de diferentes bases de dados com nomes de campos distintos, mas que representam o mesmo;
- **Deteção e resolução de conceitos de valor dos dados -** a forma como os dados estão estruturados numa base de dados pode diferir de outra e, portanto, é importante garantir que passam a ter a mesma estrutura, visto que representam o mesmo. Exemplo: duas tabelas com o mesmo campo do tipo data podem ter formatos diferentes, e que têm de convergir.

Limpeza de Dados

O processo de limpeza envolve um primeiro passo de identificação de erros nos dados e, dependendo dos tipos de erros e dos métodos definidos de lidar com esses erros, um segundo passo onde se procede à sua eliminação, atualização ou correção. Nesta segunda fase de limpeza, uma das soluções mais comuns é a utilização de *cross-checking* com um *dataset* de validação que, entre outras coisas, permite a adição de mais elementos informativos a campos vazios ou incompletos com base em dados anteriores (*Data Enhancement*). Permite ainda a normalização de dados, que envolve a transformação de dados duplicados ou não estruturados, com base em Formas Normais específicas e que garante dados coesos e com mais qualidade.

Redução de Dados

Este passo intermédio ajuda na redução do volume de dados e do espaço de armazenamento, facilitando posteriormente a sua análise, sem que o resultado sofra alterações significativas. Podem ser aplicados diferentes métodos, entre os quais:

- **Redução de Dimensionalidade** - esta técnica é utilizada para diminuir o número de características redundantes nos dados, através da combinação de vários atributos sem que os dados percam as suas características originais;

- **Redução Numerosa** - a representação dos dados diminui através da redução do seu volume, ainda que não haja perda de informação;
- **Compressão de dados** - através de tecnologias de codificação, o tamanho dos dados é reduzido significativamente e pode haver ou não perda de informação, dependendo de se os dados originais possam ou não ser obtidos desta codificação.

Transformação de Dados

Trata da alteração do formato ou estrutura dos dados através de vários métodos [12], sendo os mais comuns:

- ***Smoothing*** - permite que haja redução do ruído nos dados, para que seja possível direcionar o foco para as características relevantes nos dados;
- **Agregação** - os dados são armazenados e apresentados na forma de resumo;
- **Discretização** - os dados contínuos são divididos em intervalos, havendo uma diminuição no tamanho dos dados. Exemplo: definir um intervalo de horas (15h-17h);
- **Normalização** - redimensiona os dados para que possam ser representados em intervalos de amplitude menores. Exemplo: todos os valores passarem a variar entre -1,0 e 1,0.

2.1.3. Load

A última fase do processo de ETL é o *Load*, e é aqui que os dados extraídos e transformados nas fases anteriores encontram o seu destino final. Os dados podem ser carregados: **(1)** de forma inicial (*Initial Load*), onde se dá o preenchimento de todas as tabelas do *Data Warehouse*; **(2)** de forma incremental (*Incremental Load*), ao aplicarem-se mudanças contínuas à medida que vão sendo necessárias; **(3)** através da atualização completa (*Full Refresh*), apagando o conteúdo de uma ou mais tabelas e recarregando-as com dados novos. Independentemente de os dados serem carregados na sua totalidade ou de forma incremental, é importante que este processo seja feito da forma mais otimizada possível, para que haja uma maximização da performance. [8]

No final, devem ser feitas várias verificações de carregamento, para garantir que os dados foram carregados de forma correta:

- Garantir que as chaves primárias não estão em falta ou a nulo;
- Verificar os valores combinados e calculados;
- Verificar os dados na tabela de dimensões e na tabela de histórico;
- Testar a modelação das *views* com base nas tabelas alteradas.

2.1.3.1. Visualização de Dados

A visualização de dados é definida como a representação gráfica onde está contida informação sobre os dados, e permite a comunicação de factos a profissionais fora do meio. Esta visualização dos dados desempenha uma função crucial em duas fases distintas da exploração dos dados. A primeira, na fase da análise exploratória, onde é possível ter uma perceção de tendências, padrões e *outliers* nos dados. Aqui, a visualização permite detetar valores nulos, incorretos e corrompidos, e dar uma visão geral do

estado dos dados. Permite também combinar categorias e tem um papel primário na fase de pré-processamento. Na segunda fase, tem um papel importante na apresentação do resultado enquanto output (em *dashboards*, por exemplo) que é depois utilizado por qualquer colaborador numa empresa.

Os limites da visualização dos dados dependem da criatividade e imaginação de quem está a trabalhar com os mesmos, mas os componentes mais comuns desta visualização são os gráficos de barras, de linhas, circulares, de dispersão, diagramas de extremos e quartis, e histogramas.

3. METODOLOGIA

3.1. ESTRATÉGIA

De forma a tornar o processo de desenvolvimento do ETL que vai acolher as aplicações computacionais o mais eficiente possível, foi necessário delinear uma estratégia onde foram definidas prioridades. Uma vez que o processo ETL está dividido nas três fases que o compõe (Extração, Transformação e Carregamento) e que este é o fluxo mais utilizado durante o seu funcionamento, determinar que tarefas necessitam de ser desempenhadas primeiro torna-se um desafio mais simples. Sendo que o resultado final deste projeto é a criação de três aplicações que terão o ETL como base, o relevante a fazer durante o planeamento é perceber que passos devem ser dados dentro de cada fase desse mesmo ETL para que este possa ser utilizado para resolver os problemas previamente identificados.

A figura 4 mostra a estrutura do processo ETL a ser implementado bem como o seu fluxo de dados, de forma a servir as três aplicações finais.

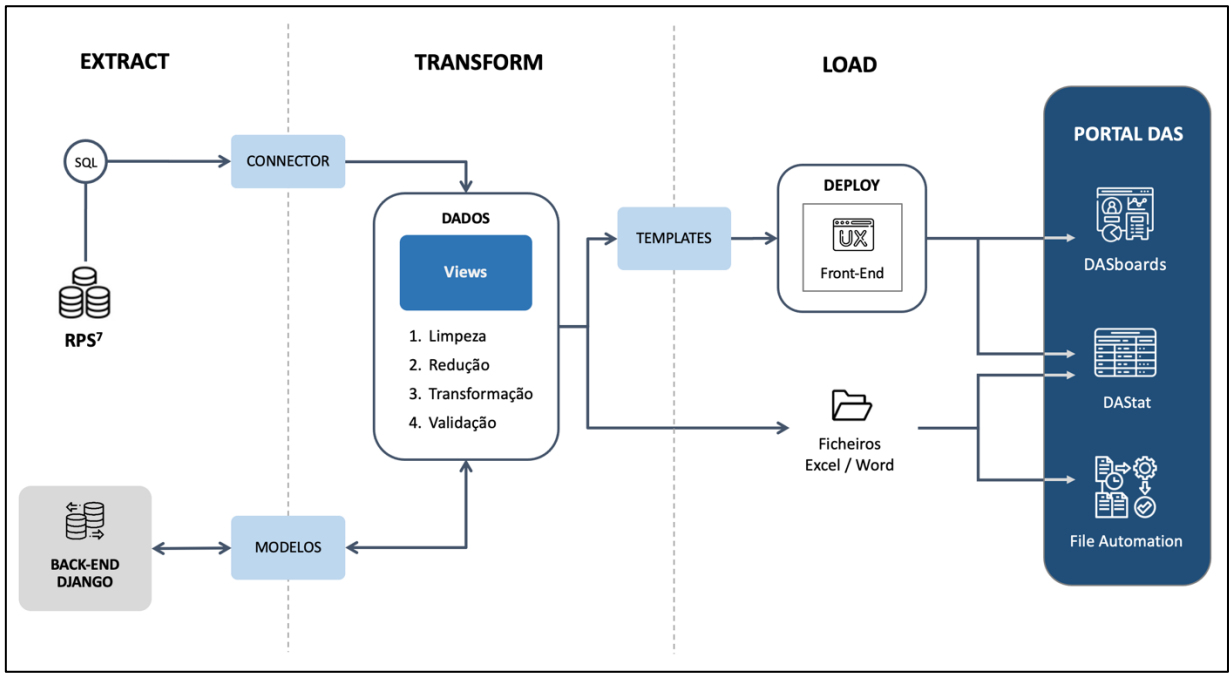


Figura 4 – Estrutura desenhada para o processo ETL

3.2. PLANEAMENTO

No início, foi definido um calendário com o planeamento para a implementação do processo ETL. Sendo que o resultado do processo serão três aplicações computacionais, os cronogramas seguintes dizem respeito ao tempo investido a desenvolver partes do processo que digam respeito a cada uma das aplicações. O planeamento inicial está representado na figura 5.

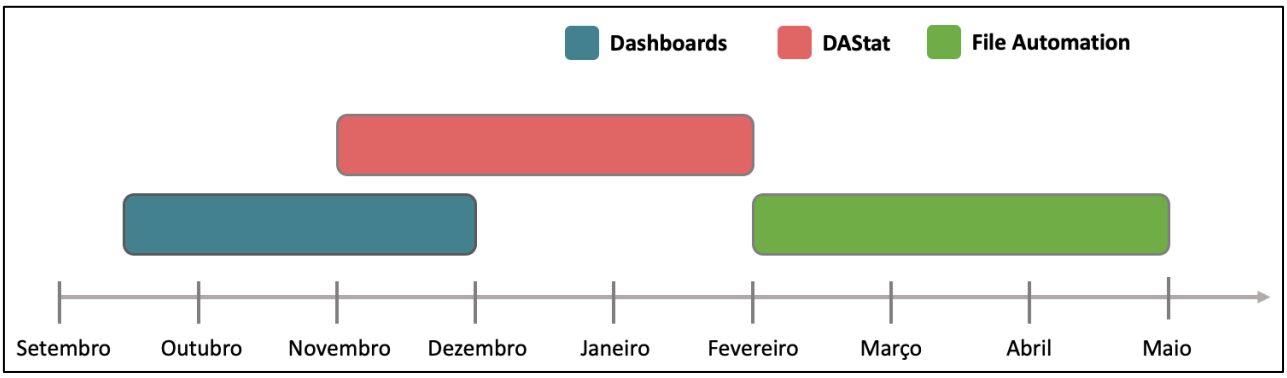


Figura 5 – Cronograma de planeamento inicial

Durante o planeamento, foi feita uma análise às aplicações a desenvolver e um levantamento dos requisitos necessários. Dito isto, como foi utilizada uma metodologia *Agile*, à medida que os projetos evoluíram, foram surgindo problemas que precisavam de ser resolvidos nos projetos já em produção e novas *features* que precisavam de ser acrescentadas. Ora, estas alterações criaram alguns atrasos que, apesar de estarem contemplados no planeamento inicial, demoraram mais tempo a resolver do que o inicialmente previsto e levaram a alterações nos prazos estipulados para os projetos. A figura 6 representa o tempo real que levou cada um dos projetos a ser desenvolvido.

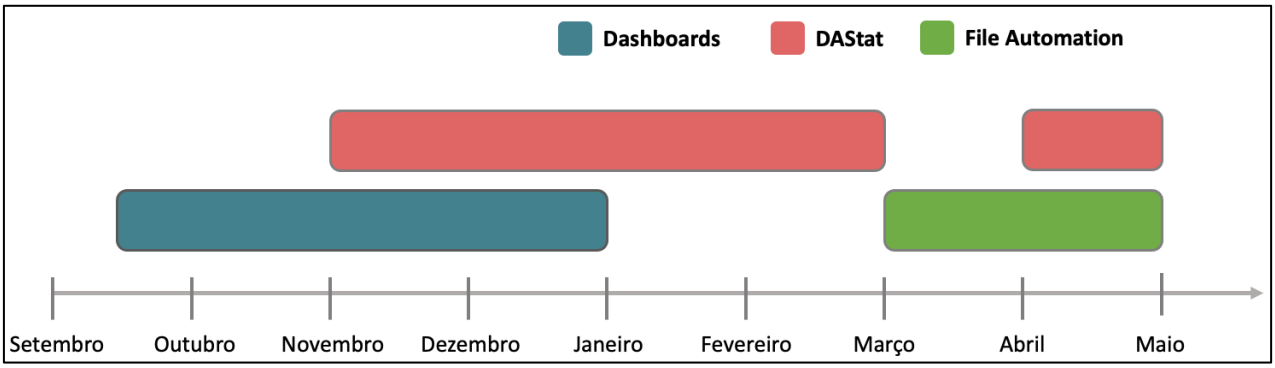


Figura 6 – Cronograma real

3.3. MATERIAIS

3.3.1. Python

Python é uma linguagem de programação de alto nível, de uso geral, projetada para ser utilizada numa ampla variedade de domínios, e é neste momento considerada a linguagem de programação mais popular do mundo. A sua versão mais atualizada conta com um diversificado leque de bibliotecas, e pode ser facilmente integrada com outras linguagens, sendo estas as características que a tornam tão conveniente e utilizada. Uma vez que permite programação orientada a objetos, mas também programação de paradigmas (processuais) e programação funcional, pode ser utilizado em *web developing*, ciência dos dados e scripts de análise, tendo também outras aplicações. Em *Python*, o código é executado de forma sequencial e a indentação das linhas tem uma função concreta, não só de aspeto visual, uma vez que indica a que bloco de código pertence cada linha a executar. [15]

3.3.2. Django

Django é um web *framework* de *Python*, *open-source* (código aberto) e de alto-nível, que segue a arquitetura *model-template-view* (MTV). Desenvolvido e mantido pela Django Software Foundation (DSF), esta *framework* permite o desenvolvimento facilitado, rápido e seguro de websites orientados a bases de dados. As principais vantagens desta *framework* são a sua reutilização, menor quantidade de código, baixo acoplamento (as classes e modelos criados são totalmente independentes entre si) e a utilização do princípio "não se repita".

Os ficheiros de *back-end* (*views.py*), os modelos de dados (*models.py*) e as definições (settings.py) utilizam *Python*, apesar de haver outras situações em que esta linguagem também é utilizada [16]. A *framework* Django também fornece uma interface administrativa de criação, leitura e eliminação de informação com base nos modelos criados.

Alguns websites bastante conhecidos como o Instagram, Bitbucket e Mozilla utilizam esta *framework* que permite a criação de um website que seja:

- **Completo**: esta *framework* segue a filosofia de "baterias incluídas" e, portanto, fornece tudo o que o programador possa vir a precisar;
- **Versátil**: pode ser utilizado para desenvolver qualquer tipo de website (desde redes sociais a sites de informação), uma vez que permite a escolha de várias bases de dados populares, *templating machines*, entre outros componentes que sejam necessários;
- **Seguro:** ajuda os programadores a evitar erros comuns de segurança, ao encapsular automaticamente certas áreas em que esta característica seja imperativa (como a gestão de contas de utilizadores, passwords e acessos a informações como cookies). Dá também proteção contra *SQL injection*[1]*, cross-site request forgery*[2] *e clickjacking*[3]*;*
- **Escalável:** cada parte da arquitetura é totalmente independente das outras e, como tal, pode ser alterada ou substituída, se necessário. Havendo esta separação entre cada uma das partes, há sempre a possibilidade cada uma delas ser dimensionada para aumentar o tráfego, ao adicionar-se hardware em qualquer dos níveis (servidores de cache, servidores de banco de dados ou servidores de aplicativos);
- **Sustentável:** utiliza princípios e padrões de design que incentivam a criação de código reutilizável, para que não haja repetições desnecessárias, que reduzem a quantidade de código. Fornece também um sistema de cache flexível, para que seja possível armazenar todas as partes de uma página que não precisem de ser renderizadas de novo, evitando o desperdício de recursos computacionais;
- **Portátil:** sendo escrito em *Python*, corre em várias plataformas, o que significa que quem o utiliza não está vinculado a nenhuma plataforma de servidor específica e, portanto, pode correr as aplicações em Linux, Windows e MacOS. O Django é também suportado por vários *web hosting providers*, que inclusive fornecem infraestrutura e documentação própria para hospedar sites em Django.

1 técnica de injeção de código que pode destruir bases de dados;

2 ataque que força o utilizador a executar ações indesejadas no website no qual está autenticado;

3 ataque que induz o utilizador a carregar num elemento da página web que está disfarçado de outro elemento ou é invisível.

3.3.2.1. Estrutura

O Django contém vários componentes que, em conjunto, permitem o correto funcionamento das aplicações, e seguem o fluxo representado na figura 7. A função de cada um dos componentes é explicada de seguida.

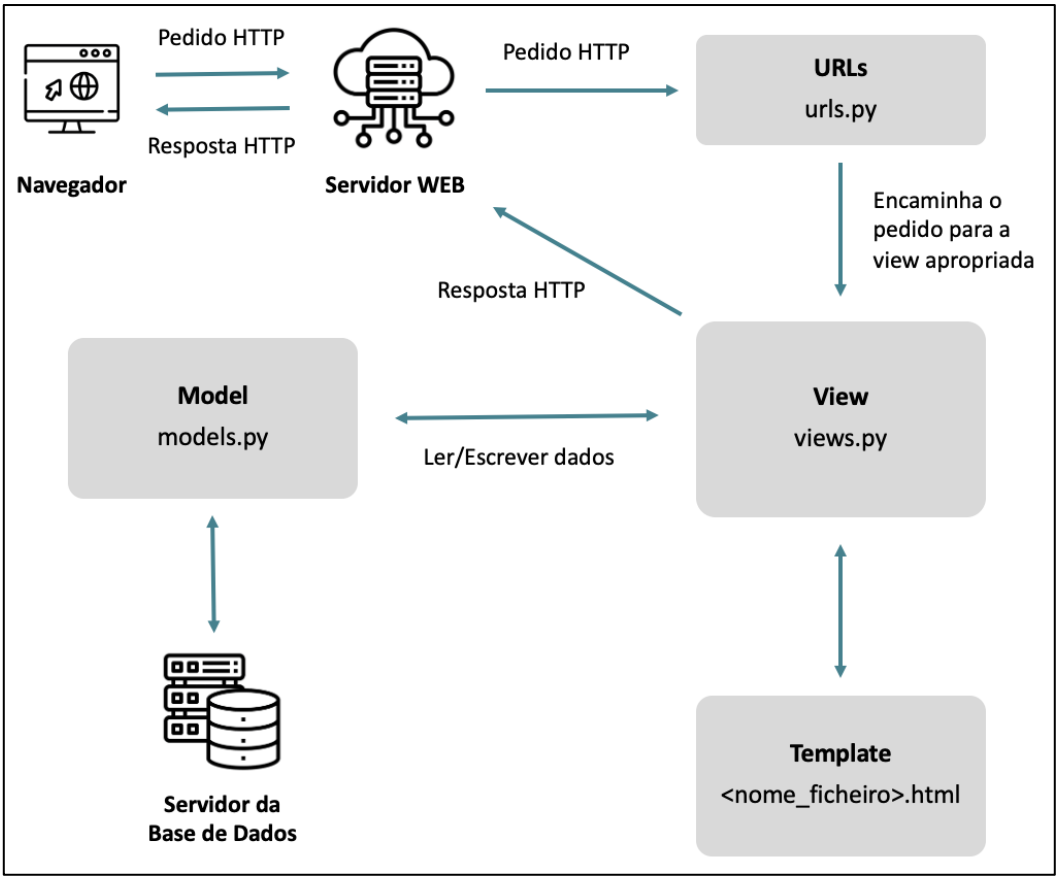


Figura 7 – Diagrama da Estrutura do Django

URLs: redireciona solicitações *HTTP* para a visualização certa com base no *URL* da solicitação. Estes *URLs* podem conter *strings* ou números específicos que representam visualizações de dados concretas;

Views: têm a função de lidar com pedidos *HTTP* que recebe, e enviar respostas também *HTTP* com os dados pedidos, que só são acedidos nesta parte;

Models: objetos *Python* que definem a estrutura de dados das aplicações, e que fornece mecanismos para gerir e consultar registos na base de dados;

Templates: define a estrutura ou layout de um ficheiro (como uma página html), com espaços reservados para representar conteúdo real e dinâmico;

***Forms*:** recolhem dados preenchidos pelos utilizadores, e que serão depois processados pelo servidor.

3.3.3. SQL

SQL, que significa *Structured Query Language,* é uma linguagem de programação standard que consiste em vários tipos de declarações consideradas sublinguagens, sendo estas: *Data Query Language* (DQL), *Data Definition Language* (DDL), *Data Control Language* (DCL) e *Data Manipulation Language* (DML). As sublinguagens incluem ações como consulta, manipulação (inserir, atualizar e apagar), definição (criação e modificação de *schema*) e controlo de acesso de dados. A linguagem está dividida em vários elementos como *clauses, expressions, predicates, queries e statements* e, apesar da existência destes *standards*, o código em SQL requer algumas alterações dependendo do sistema de bases de dados a ser utilizado. [17]

3.3.3.1. SQL Server Management Studio (SSMS)

É um sistema de gestão de base de dados relacional - *Relational Database Management System* (RDBMS) - desenvolvido e comercializado pela Microsoft. Suporta uma grande variedade de aplicações de processamento de transações, *business intelligence* e análise e que, tal como outros RDBMS, foi construído tendo por base SQL. Atualmente, funciona para ambos os sistemas operativos *Windows* e *Linux*, e é utilizado tanto para gestão de bases de dados como para consultar os dados que estas contêm (através de *queries*).

3.3.4. GitHub

O GitHub é uma plataforma de hospedagem de código fonte e de arquivos com controlo de versões, através do Git. É a plataforma de gestão de código mais utilizada do mundo e a ferramenta ideal para quem utiliza metodologias Agile. Tem várias utilidades [18], entre elas:

- Serve como repositório;
- Permite a criação de *branches* com desenvolvimento de código em paralelo;
- Criação de *commits* do novo código bem como fusão de código de diferentes origens (*branches*);
- Lançamento de novas versões para produção;
- Gestão de tarefas e *bugs* a corrigir;
- Automatização de testes a novos lançamentos;
- Criação de documentação;

3.4. MÉTODOS

3.4.1. Agile

A metodologia *Agile* permite às equipas fornecer respostas rápidas e imprevisíveis ao *feedback* que recebem no seu projeto, e cria oportunidades para avaliar a direção de um projeto durante o seu ciclo de desenvolvimento. Caracteriza-se por dividir cada projeto em pequenas partes que devem ser

completadas periodicamente. De entre as muitas vantagens da utilização desta metodologia em desenvolvimento de software [13], destaca-se:

- Envolvimento e satisfação das partes interessadas;
- Transparência;
- Entrega antecipada e previsível;
- Custos e programação previsíveis;
- Priorização flexível;
- Permite a mudança;
- Concentra-se no valor dos negócios;
- Concentra-se nos utilizadores;
- Melhora a qualidade;
- Dá propósito à sua equipa.

3.4.1.1. Kanban

O método Kanban ("quadro visual" em japonês) é organizado num quadro ou tabela, e dividido em colunas, apresentando cada fluxo dentro do projeto de produção de software. As colunas variam dependendo de quem está a implementar a metodologia, mas as mais comuns são: A Fazer, Em Execução, Em Aprovação e Finalizado, como representado na figura 8.

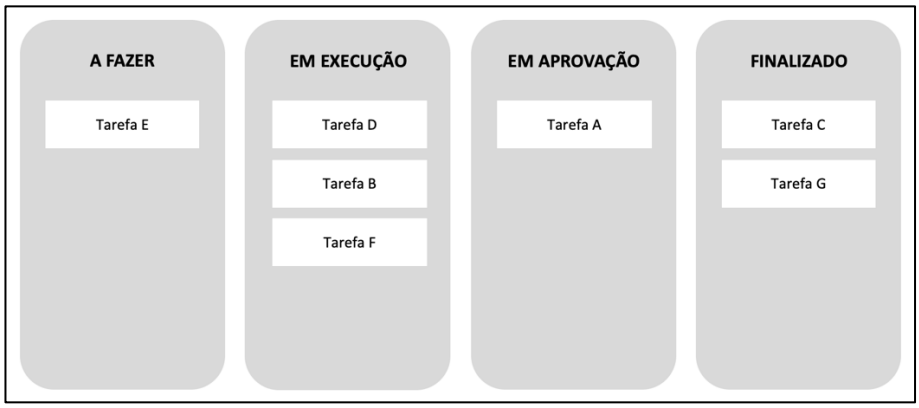


Figura 8 – Exemplo do método Kanban

Ao longo da evolução de um projeto, a informação contida na tabela é alterada, e sempre que uma nova tarefa surge, um novo "cartão" é criado. O método Kanban requer **comunicação e transparência,** para que os membros da equipa saibam exatamente em que fase se encontram os processos, e possam ver o estado de um projeto em qualquer altura. Tal como outros, o método Kanban tem vantagens e desvantagens na sua utilização, e as principais são contempladas na tabela 3.

Vantagens	Desvantagens
Capacidade de ver todas as tarefas do projeto	Possibilidade de má interpretação da informação exposta
Possibilidade de definir um limite de número de tarefas em desenvolvimento	Como não existem prazos, podem surgir atrasos em cada uma das fases
Foca-se na duração de um ciclo	
Permite entregas contínuas	
Uma das metodologias mais simples de implementar	

Tabela 3 – Vantagens e Desvantagens do método *Kanban* [14]

4. APLICAÇÕES E ETL

O departamento dispõe de um portal (Portal de Acesso à Informação) com várias aplicações. Criado há 2 anos pela Unidade de Planeamento e Controlo, é neste portal que são colocadas as aplicações desenvolvidas com o intuito de ajudar as áreas de negócio a desempenhar as suas funções, de forma mais fácil, rápida e eficaz.

As três aplicações desenvolvidas durante o estágio também integraram este portal e tiveram na sua base o ETL desenvolvido de raiz. É este ETL que tem a estrutura por onde os dados passam, do início ao fim, e que garantem o bom desempenho de cada uma das aplicações, seja qual foi a interação. Para dar suporte a este ETL, foi criado um *connector* que faz a ligação entre o Django e a base de dados, e que é introduzido em 4.1.1**.**

Para o desenvolvimento de cada uma das aplicações, foi também necessário desenhar e desenvolver vários modelos dos dados onde assenta toda a informação considerada útil ao seu correto funcionamento e que é explicado em 4.1.2. Este passo antecedeu o desenvolvimento do ETL.

4.1. ESTRUTURAS DE APOIO AO ETL

4.1.1. Connector

O *Connector* é uma ferramenta de suporte que foi desenvolvida antes da criação do ETL, e que serve como ponte entre a base de dados e a estrutura desenvolvida na fase de *Load*. É uma ferramenta transversal a todo o portal e pode ser utilizada por qualquer aplicação, bastando para isso, importá-la. Um dos principais atributos é a possibilidade de enviar mais do que uma *query* em simultâneo, e receber a respetiva informação, sem que seja necessário submeter vários pedidos.

O *Connector* segue os seguintes passos:

1) Liga-se a uma base de dados do servidor (bastando para isso indicar como argumento a base de dados que pretendemos) e uma ligação é estabelecida;
2) Uma *string* (ou um *array* de *strings*) é enviada(o) e que representa(m) as *query(ies)* que pretendemos obter;
3) Recebe como resposta uma tabela (ou *array* de tabelas) que converte no imediato para *dataframe(s)*;

4.1.2. Modelação dos Dados

Entre o processo de definição dos projetos a serem desenvolvidos e o de criação do ETL, houve um passo intermédio muito importante que facilitou a identificação da informação necessária a ser extraída da base de dados e a transformação que seria necessária ocorrer. O desenho dos modelos de dados e que servem como estrutura de suporte ao ETL, pois garantem que a informação fica organizada de forma prática e acessível [19]. À medida que os projetos foram sendo implementados,

e era necessário desenvolver novos atributos não planeados, criaram-se modelos e/ou adaptaram-se os já existentes. Para que isto acontecesse, era importante que, desde o início, estes fossem escaláveis, de forma a poderem acomodar as novas necessidades.

Depois de criados e dependendo da sua finalidade, a forma como cada modelo é preenchido difere por duas vias:

⇒ **Povoamento manual:** A informação é povoada de forma manual e serve como orientador da informação que chega da base de dados, e como a processar ao resultado desejado. Os Anexos 1.1 e 1.2 contém imagens do *back-office* do Django. O Anexo 1.1 mostra a listagem das instâncias criadas para as Guias de Pagamento (exemplo explicado abaixo) e o Anexo 1.2 o formulário para a criação dessas mesmas instâncias.
Exemplo: Para o desenvolvimento das Guias de Pagamento (ver 5.2.3), é necessário saber o(s) destinatário(s) das coimas aplicadas (sendo as hipóteses possíveis: Estado, Banco de Portugal, Fundo de Garantia de Depósitos e Sistema de Indeminização a Investidores) para que seja possível gerar uma tabela com o total a ser pago a cada um. Ora quando uma coima é aplicada, não tem associada a informação dos destinatários. Tem antes os diplomas sancionatórios aplicados a cada uma dessas coimas, sendo através destes possível identificar o(s) destinatário(s). Para que tal seja possível, e de forma a tornar a identificação dos destinatários dinâmica (caso sejam criados novos diplomas sancionatórios), foi desenvolvido um modelo de correspondência entre cada um dos diplomas e o(s) seu(s) destinatário(s). Desta forma, quando é recebido o *dataframe* com as coimas aplicadas e os respetivos diplomas para um arguido, basta fazer um pedido à base de dados do Django do(s) destinatário(s) associados aos diplomas. Para que esta extração de informação funcione corretamente, os modelos foram preenchidos manualmente, sendo possível criar correspondências ou eliminar as já existentes.

⇒ **Povoamento automático:** Para que não seja preciso processar sempre todos os dados essenciais a um pedido, determinadas informações ficam registadas em modelos de forma automática. Na primeira vez, a informação chega de forma completa, e é registada nos modelos e, a partir daí, só a informação que foi alterada é enviada pela base de dados e atualizada. Isto permite que seja mais rápida a sua consulta e que menos recursos sejam gastos.
Exemplo: Nos DASboards, a informação estatística relativa ao desempenho dos colaboradores em termos de processos é alterada com pouca frequência e, como tal, não há necessidade de pedir constantemente os mesmos dados. Assim, essa informação fica guardada na base de dados do Django.

Nota: Nunca são guardados dados sensíveis que violem a lei da proteção de dados, independentemente do tipo de modelo.

4.2. APLICAÇÕES

As aplicações construídas durante este estágio, bem como outras já existentes, são colocadas à disposição numa página própria do portal do departamento, e estão divididas por áreas. Algumas aplicações são transversais a todas as áreas, e por isso estão representadas em todos os *tabs,* e algumas exclusivas a uma área concreta. O Anexo 2 representa essa página com a listagem de todas as aplicações disponíveis.

4.2.1. DASboards

Problema: Os coordenadores das várias áreas do departamento utilizam diferentes métodos para organizar e gerir o trabalho a ser desenvolvido pelos colaboradores. Geralmente, esta informação reside em ficheiros Excel, preenchida de forma manual e com estruturas totalmente diferentes, de área para área. Ora, esta metodologia origina 3 adversidades: (1) os coordenadores precisam de despender de tempo, mensalmente, para preencher os ficheiros; (2) a informação não é atualizada em tempo real; (3) a direção tem de fazer a ginástica de interpretar ficheiros com estruturas diferentes, como forma de apoio à tomada de decisão.

Solução: Desenvolvimento de uma interface gráfica, com tanta ou mais informação do que a que consta nos *exceis*, com carregamento de forma automatizada, e que permita consultar em tempo real o estado dos processos.

Output: Criação de *dashboards* interativos, com vários níveis de granularidade divididos por separadores, para análises distintas sobre o negócio do departamento. Um dos principais requisitos é que haja restrições de acesso a determinadas informações dependendo do tipo de utilizador (entre diretor, coordenador ou técnico). Para além de informação acerca dos processos a decorrer, também consta informação acerca dos técnicos alocados a esses mesmos processos e o estado dos KPIs[4] definidos. O Anexo 3 mostra a página principal dos DASboards, acessível apenas aos diretores do departamento. O menu à esquerda tem os diversos separadores onde a granularidade aumenta à medida que descemos. Esta página principal tem uma visão geral das quatro áreas de negócio do departamento e à direita uma visão agregada de todo o departamento.

Tecnologias: Figma; SQL; *Python*; Django.

4.2.2. DAStat

Problema: Aos juristas do DAS é pedido, por várias entidades, que reportem constantemente e com dados estatísticos, toda a informação acerca do trabalho que tenham vindo a desempenhar em matéria de supervisão. Sendo que esta informação está guardada em bases de dados, e que precisa de ser processada antes de ser entregue ao colaborador, é sempre pedido a um técnico da Unidade de Planeamento e Controlo (UPC) que extraía essa informação e a trabalhe antes de esta ser entregue. Isto traduz-se em grandes dispêndios de FTE's de vários colaboradores em simultâneo.

Solução: Que qualquer colaborador, sem conhecimento prévio da estrutura dos dados, possa selecionar os campos que gostaria de ver extraídos, aplicar filtros específicos para alguns desses campos e critérios de ordenação diferentes, e os possa visualizar e/ou extrair para um ficheiro Excel, onde os possa utilizar de forma externa.

4 Key Performance Indicator *(Indicador-Chave de Performance) - métrica importante à mensuração do desempenho de uma estratégia e de processos de gestão.*

Output: Criação de uma aplicação em formato self-service de seleção e extração de dados.

O Anexo 4 representa o DAStat final. À esquerda, na parte superior, está a listagem de todos os campos disponíveis para seleção e visualização e, à medida que os campos vão sendo selecionados, estes passam da lista superior para a inferior (de onde podem ser removidos). Na lista inferior é ainda possível alterar a ordem pela qual os campos são representados na tabela ao centro.

Tecnologias: Figma; Django; SQL; *Python.*

4.2.3. Files Automation

Problema: O preenchimento manual de informação (seja para relatórios, trabalho administrativo, ou outros) requer sempre um gasto de *FTE's*[5], que poderiam ser utilizados em outras funções. Com o aumento do número de processos a serem averiguados pelo departamento ao longo dos últimos anos, tornou-se inviável o preenchimento manual destes ficheiros, sob pena do departamento ter de alocar colaboradores a tempo inteiro a estas tarefas.

Solução: Criação de um sistema de automatização do fluxo de informação de ponta-a-ponta e que possa ser escalável para o preenchimento de ficheiros com estruturas diferentes.

Output 1 (Guias): Criação de um motor de busca onde é possível pesquisar por visado e, na lista dos resultados obtidos, selecionar a extração da Guia de Pagamento para o visado pretendido, com os dados integralmente preenchidos. O Anexo 5.1 representa o modelo Word utilizado para o preenchimento dos dados para pagamento.

Output 2 (Reporte Estatístico): Criação de uma *webpage* onde estão listados vários tipos de ficheiros diferentes. Ao ser selecionado o ficheiro que pretendemos ver preenchido, é aberto um calendário onde é possível escolher o intervalo de tempo a que os dados se referem (Anexo 5.2). Depois, basta carregar no botão de descarregar, e começará o descarregamento de um ficheiro (Word ou Excel) com a informação preenchida. O Anexo 5.3 corresponde ao modelo Excel utilizado para o preenchimento dos dados estatísticos apurados.

Tecnologias: Excel; *Python*; Django; SQL.

4.3. ETL

4.3.1. Extract

A ligação da base de dados ao portal e a obtenção de informação são ambas feitas através do *Connector*, descrito em 4.1.1. A *string* que é enviada por esse mesmo *Connector*, e que contém a *query*

5 *Full Time Equivalent* (Equivalente em Tempo Integral) – unidade de medida que indica a carga de trabalho de uma pessoa a tempo integral, para que possa ser utilizada como comparação em vários contextos.

em SQL, passa por um processo de testagem, antes de ser integrada no código em *Python*. Para essa testagem, é utilizado o *SQL Server Management Studio*, onde é possível fazer as *queries* diretamente à base de dados e obter os resultados numa tabela, representada na aplicação. Só depois de feitos os ajustes à *query* nesta aplicação, é que esta é passada para um formato de *string*, e é integrada em *Python*. Utilizando esta metodologia de testagem antes de integrar as *queries* no código, é evitado repetir todo o fluxo de extração de informação através do *Connector*, sempre que alteramos alguma coisa em alguma *queryi* e a queremos testar. Ao utilizarmos o *SQL Server Management Studio* não só poupamos tempo, como recursos.

Depois de enviada a *string*, através do *Connector*, a resposta da base de dados é recebida num *dataframe*. Com a resposta neste formato, os dados podem ser manipulados diretamente em *Python*, com a ajuda das operações da biblioteca *pandas*.

As três aplicações desenvolvidos durante o estágio utilizam este método para a extração de informação da base de dados. Os únicos detalhes que variam, dependendo da operação a efetuar, são a definição da base de dados de onde extrair os dados (a partir de um dos argumentos da ligação inicial do *Connector* ao servidor) e, claro, a informação pedida em cada *query* enviada.

4.3.2. Transform

Dependendo do objetivo final dos dados que são recebidos da fase de *Load*, há uma série de passos por onde estes dados (em formato *raw*) passam, para que fiquem processados e orientados ao resultado pretendido. Dependendo da aplicação em questão, nem todos os dados passam por todas as fases.

Como em muitas das *queries* feitas à base de dados, há muitos requisitos e condições que precisam de se verificar no histórico dos dados. Por exemplo, numa *query* a processos em investigação, queremos verificar apenas os processos que tenham passado de uma fase específica para outra, dentro de um determinado período, e cujos visados associados nunca tenham estado num certo momento processual. De forma a manter as *queries* o menos extensas e complexas possíveis, mas também com menor abertura a possíveis erros, foi decidido selecionar todos os processos que correspondessem a um requisito mais geral, e receber todo o histórico desses mesmos processos, durante um determinado período. Desta forma, garantimos que não estão a ser descartados potenciais processos, logo à partida. Depois sim, em *Python* e a partir do *dataframe* recebido, verificou-se o histórico e filtrou-se apenas os processos onde se verificassem todas as condições necessárias. Deste modo foi possível tornar os resultados mais precisos, visto que é verificada uma condição de cada vez em funções concretas para o efeito, em *Python*, e não tudo de forma agregada em SQL.

4.3.2.1. Limpeza dos Dados

Uma vez que os dados chegam em bruto, há uma série de informação que é logo descartada à partida, por não preencher os requisitos mínimos para ser considerada útil e suficiente. Em nenhum caso é aplicada a substituição de dados por estes estarem em falta, uma vez que pretendemos trabalhar com os dados reais, de forma precisa e não por aproximação.

O critério de eliminação de dados que não correspondam aos critérios depende da aplicação em questão e do produto final pretendido:

⇒ **DASboards**: é feita uma limpeza dos valores nulos, uma vez que não têm representação possível num *dashboard*. A única exceção são os casos em que um valor nulo numa coluna possa ter um algum significado concreto. Por exemplo um colaborador não ter estado envolvido em qualquer processo de um determinado tipo, e daí uma coluna ter valores a nulo. Isso pode ser importante para a representação visual, e as colunas onde isso pode acontecer são identificadas desde início;

⇒ **DAStat**: é do maior interesse de todos os envolvidos que o espelhamento da base de dados seja o mais real e transparente possível e, como tal, que os dados não passem por qualquer processo de limpeza e que sejam mostrados mesmo quando contém valores nulos ou mal carregados,

⇒ ***File Automation***: para a emissão das Guias de Pagamento, só as linhas com valores não nulos são úteis para o cálculo das coimas finais e, por isso, as linhas com coimas nulas são eliminadas por não representarem informação proveitosa. Para o Reporte Estatístico, é utilizado o mesmo método dos DASboards. Determinadas informações nulas podem ter um significado por detrás e, consequentemente, são mantidas. Nas restantes colunas, os valores nulos são descartados.

4.3.2.2. Redução dos Dados

Tal como referido no início da fase de Transformação, de forma a simplificar as *queries* a fazer, há muita informação extra que chega da base de dados, e que só depois é filtrada, caso não passe nos critérios de seleção definidos. De forma a reduzir os dados, são criadas várias condições em *Python* que analisam o histórico recebido de cada um dos processos e que confirmam se determinadas situações ocorreram. Por exemplo, para calcular o número de processos fechados num período, primeiro devemos extrair da base de dados os processos que, nesse intervalo de tempo, estiveram na fase "Condenação Final", juntamente com todo o histórico anterior a essa fase (mesmo sendo anterior ao período definido). Isto porque para um processo ser considerado fechado, tem de ter passado obrigatoriamente por determinadas fases no passado e, portanto, temos de analisar todo o seu histórico, e confirmar se tal aconteceu. Isto seria possível de determinar em *SQL*, mas tornaria o pedido demasiado complexo e suscetível a erros. Esta etapa é muito útil para os DASboards e *File Automation* sendo que não é aplicada no DAStat.

4.3.2.3. Transformação dos Dados

A forma como os dados estão guardados na base de dados respeita as normas definidas e é a ideal para garantir a sua eficiência, mas, nalguns casos, esses dados não estão na forma ideal para serem carregados, seja para um ficheiro, seja para uma *webpage*. E é nesta transição, entre como é que os dados chegam e como saem, que se dá um dos passos mais importantes para o *Transform*.

Ao contrário do que acontece nas etapas anteriormente abordadas, nesta o procedimento a seguir é igual para as três aplicações. A forma como cada uma das colunas recebidas é transformada não varia consoante a aplicação sendo que, a única coisa diferente é a existência de determinadas colunas (nos DASboards, por exemplo, não consta informação sobre normas violadas).

A transformação ocorrida pode ser representada numa matriz entre os campos transformados e as técnicas utilizadas, como representado na tabela 4.

Técnica	Campo
Revisão do Formato	**DATA:** passou de AAAA-MM-DD para DD/MM/AAAA. A variável continuou a ser do tipo DATE.
	COIMAS: inicialmente um inteiro. Passou a ser uma *string* no formato '000 000,00€'
Divisão	**NORMA SANCIONATÓRIA:** divisão em três colunas – norma violada, artigo e alínea
Agregação	**SOMA TOTAL DE COIMAS:** os valores da coluna COIMAS é somado (para o mesmo processo ou arguido) e guardado numa nova coluna.
Fusão	**IDENTIFICAÇÃO DO PROCESSO[6]:** a identificação alfanumérica de um processo requer a agregação de três colunas – número de processo, ano do processo e tipo do processo

Tabela 4 – Técnicas aplicadas e campos transformados

4.3.2.4. Validação dos Dados

Só na emissão de Guias de Pagamento, na aplicação de *File Automation,* é feita a validação dos dados que chegam da base de dados. Primeiro, porque nas duas outras aplicações, DASboards e DAStat, o ideal é representar os dados tal e qual como eles estão guardados na base de dados. Segundo, porque na emissão das Guias de Pagamento, há uma responsabilidade acrescida de garantir que os dados que estão a ser utilizados são precisos. Para que tal aconteça, foi criado um sistema de validação de dados que se conecta ao *Load*, para mostrar ao utilizador mensagens de erro, caso se identifiquem anomalias nos dados.

6 Formato final de identificação de um processo: número_processo / ano_criação_processo / tipo_processo (exemplo: 171/22/CO)

O fluxo de validação que antecede a emissão das Guias de Pagamento é o seguinte:

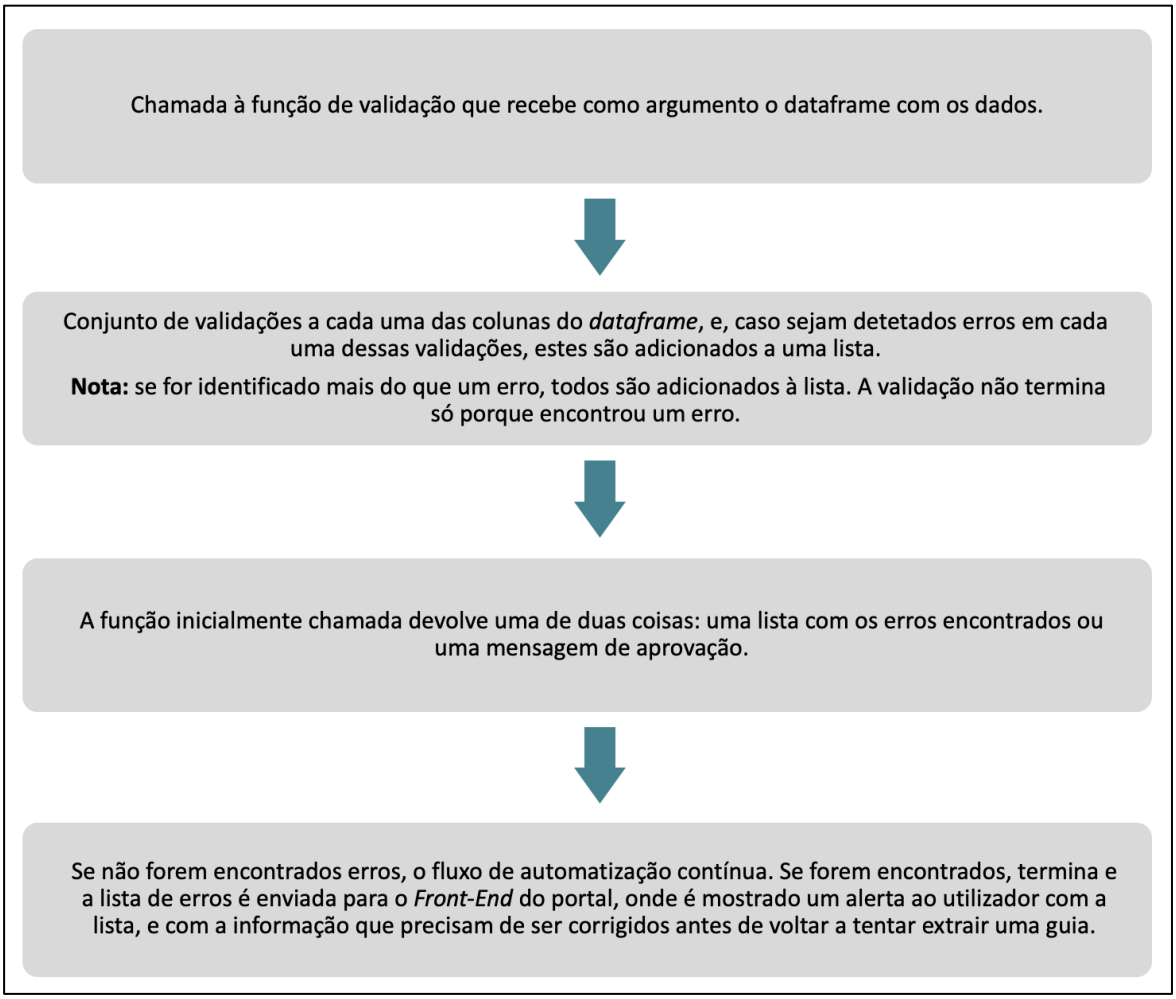


Figura 9 – Fluxo de Validação das Guias de Pagamento

O Anexo 6 representa um exemplo das mensagens de erro mostradas ao utilizador aquando da tentativa de extração de uma Guia de Pagamento.

Nota: A única opção que o utilizador tem à disposição para corrigir os erros detetados é aceder ao repositório onde os dados estão guardados e adicionar/corrigir a informação assinalada em falta/errada.

4.3.3. *Load*

Depois de garantir que os dados passam pela fase de Transformação e seguem todas as normas definidas, estes seguem um de três caminhos de forma a chegarem ao destino final.

1. Exportação para ficheiros Word e Excel

DAStat

É criado um ficheiro de Excel virtual, através da biblioteca *Openpyxl*. Os dados que estão representados na *webpage* do DAStat em formato de tabela (e que estão num *dataframe* em *Python*) são utilizados para popular esse mesmo Excel, exatamente pela mesma ordem e formato em que se encontram no *dataframe*. Uma vez terminado este processo, o ficheiro é enviado através de um *URL* gerado por um *http response*, e um *pop-up* de transferência é aberto no navegador com o ficheiro que está a ser descarregado.

***File Automation* (Guias e Reporte Estatístico)**

Todos os ficheiros gerados precisem de um ficheiro modelo como ponto de partida e, para isso, utilizam um dos ficheiros guardados no servidor do portal, conforme o ficheiro a gerar.

Quando os dados se encontram prontos para serem carregados para esse ficheiro, é criado um ficheiro virtual, que é uma copia do modelo original, e que é povoado com os dados transformados. No views, quando os dados estão prontos para serem extraídos, um dicionário é criado e preenchido com a informação toda organizada pelos campos que pretendemos preencher no ficheiro virtual. Se se tratar de um ficheiro Excel, é utilizado o *Openpyxl* para o povoamento dos dados. Se for um ficheiro Word, é utilizada a biblioteca *MailMerge* que, entre várias características, a principal é ser possível ligar os campos a serem preenchidos nesse ficheiro Word a variáveis que depois podem ser chamadas em *Python*, e preenchidas com a informação do dicionário.

Uma vez que os ficheiros estejam preenchidos e à semelhança do processo no DAStat, é gerado um *URL* e um *pop-up* aparecerá com a indicação de que o download do ficheiro começou.

2. Carregamento de dados para a base de dados do Django

Tal como referido em 4.1.2 (Povoamento Automático), certos dados sofrem alterações ocasionais e, tratando-se muitas vezes de grandes quantidades de dados, este processo gera uma grande alocação de recursos do servidor para que possa enviar toda a informação pedida. Isto torna a espera para carregamento das aplicações mais demorada, uma vez que em plano de fundo, todos os dados estão a ser pedidos e transformados antes de serem carregados. Para evitar esta situação, foi criada a possibilidade de os dados serem guardados em modelos construídos para o efeito, e de serem utilizados no futuro, sem que para isso seja necessário recarregar toda a informação desde o início.

Na primeira vez que estes dados são pedidos ao servidor, dá-se o carregamento inicial - *initial load* (ver ponto 2.1.3) -, e estes dados povoam a base de dados do Django a partir dos modelos previamente criados. A partir daí, e sempre que estes dados são requisitados, há apenas um processo de

carregamento incremental (*incremental load)*, onde apenas os dados que sofreram alterações são recebidos e atualizados.

Sempre que é feito um pedido de dados que se sabe que estão na base de dados do Django, primeiro garantimos que a informação lá guardada está atualizada e, depois sim, é criado um *QuerySet*, com os filtros para os campos desejados. Esse *QuerySet* é então carregado para o *template* da *webpage*.

Este processo tem uma aplicação real no carregamento dos DASboards. Certas informações relacionadas com técnicos e processos só é alterada casualmente e, como tal, não há necessidade de pedir essa informação constantemente. Assim, essa informação fica guardada na base de dados do Django, e o servidor só envia a informação que foi atualizada desde a data do último pedido. Isto traz inúmeros benefícios, como eficiência de recursos, rapidez no carregamento e uma maior fluidez na navegação entre separadores dos DASboards.

3. Criação de uma *Webpage* – Fase de *Deploy*

Para os dados ficarem visíveis, nem sempre estes são exportados para ficheiros, mas só os DASboards e o DAStat passam por este processo. Aliás, sempre que seja possível, o ideal é a informação ser representada numa *webpage* dentro do portal, onde se encontram as aplicações, sem que seja necessário recorrer a ficheiros. Isto porque se isso acontecer, há descentralização de informação e possibilidade de manipulação manual não desejada.
Mesmo nas aplicações **DAStat e *File Automation***, apesar do output final ser um ficheiro Excel ou Word, estas aplicações passaram por uma fase de *deploy* de uma *webpage*. Nesta fase foram carregados todos os elementos visuais que permitem ao utilizador selecionar os critérios que querem ver visíveis no ficheiro, antes de o exportar.

O processo de criação de uma *webpage* com informação dinâmica segue o seguinte fluxo (e utilizando o exemplo de envio de dados sobre os colaboradores para os DASboards):

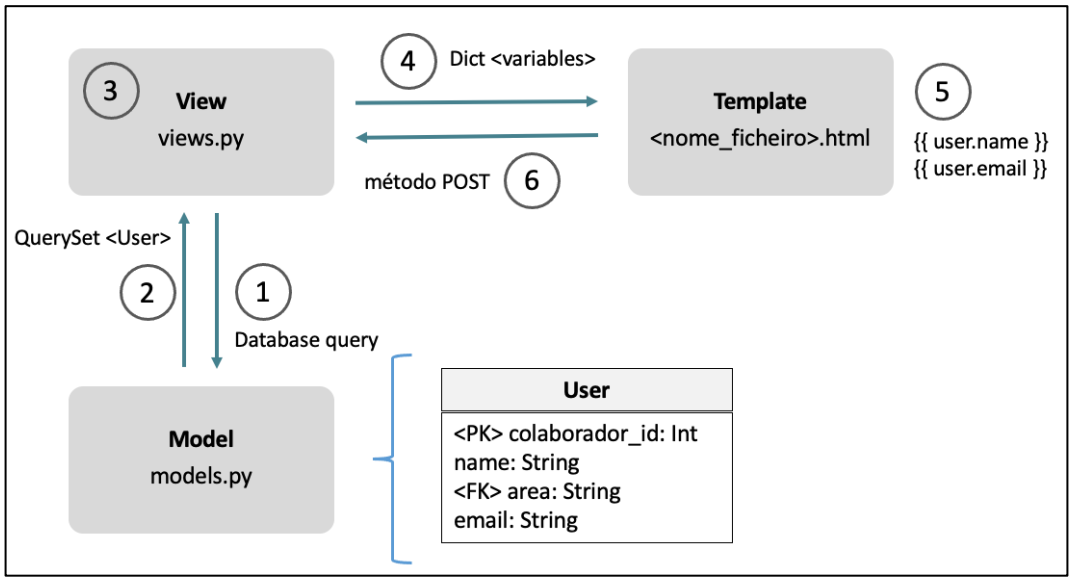


Figura 10 – Fluxo de Informação para *webpage* [20]

1) Um pedido de dados com critérios definidos para alguns campos é feito à base de dados do Django;
2) É devolvido um *QuerySet* com a lista de '*Users'* que correspondem à *query* feita em **1**;
3) O *QuerySet* é encapsulado com outras variáveis e renderizado num *http response*;
4) As variáveis encapsuladas em **3** são enviadas em formato de dicionário para o *template* definido no *views.py*;
5) É aqui, no *template*, que encontramos toda a estrutura da página. Os elementos em *html* e as classes de estilos (CSS) são responsáveis pelo esquema e *design* da página, respetivamente.
Cada uma das variáveis recebida no *template* é chamada e utilizada a partir da chave definida para essa mesma variável no dicionário enviado em **4**. Se se tratar de um *QuerySet*, campos específicos do objeto em questão podem ser também chamados, como representado em **5**. De forma a facilitar a manipulação da informação enviada em *QuerySets,* o Django dispõe de vários *tags* que podem ser utilizados em *html* para o efeito – *if else's, for loop's, extensions, tokens*, etc.;
6) Se houver alguma interação que requeira a ação do *views.py*, um *POST method* é enviado com o *id* do elemento onde houve a interação e com a informação constante desse elemento. **Exemplo:** se o utilizador carregar no botão de download de um ficheiro, é enviado para o *views* o *id* desse botão (para o *views* saber que ação tomar) e toda a informação necessária ao descarregamento do ficheiro (como o nome do ficheiro, e os intervalos de datas a selecionar).

4.3.4. Resultados Obtidos

Todos os dados passam pelo ETL antes de chegarem às aplicações finais, mas, no entanto, nem todas as fases do ETL têm o mesmo peso para o resultado obtido. Claro que esse peso varia de aplicação para aplicação, mas, ao observarmos a tabela 5, concluímos que é a fase de Transformação que teve mais relevância para a construção do ELT, seguido da fase de Carregamento. Se observarmos o peso das fases por aplicação, este não está distribuído de forma homogénea. No projeto do **DAStat**, a fase de Carregamento é o passo mais importante. Nos outros dois projetos é o passo de Transformação, mas, se repararmos na distribuição do ***File Automation*** pelo ETL, a fase de Extração tem mais do triplo do peso do de Carregamento.

	Extract	***Transform***	***Load***	**Final**
DASboards	15%	55%	30%	100%
DAStat	20%	15%	65%	100%
File Automation	35%	55%	10%	100%
Peso final	**23%**	**42%**	**35%**	**100%**

Tabela 5 – Distribuição do ETL pelas fases e aplicações

Nota: O cálculo das percentagens foi feito com base no tempo despendido em cada fase, para cada aplicação, durante o estágio.

5. CONCLUSÕES

O resultado de todo o projeto desenvolvido durante o estágio profissional superou as expectativas. O trajeto feito durante todos estes meses foi muito positivo e agradável, e esse foi um dos motivos que fez o resultado ser tão bom. Todas as propostas de implementação inicialmente definidas foram conseguias e, em alguns casos, até foi possível ir mais além.

O ETL construído durante a criação de três aplicações deu provas de ser robusto e, mesmo quando foi preciso alterar ou acrescentar requisitos, este adaptou-se de forma muito positiva e mostrou ser escalável. Como observado nos Resultados Obtidos (4.3.4), o ETL ficou bastante bem distribuído pelas três fases de atuação, o que tornou favorável que várias técnicas e métodos de diferentes áreas de Ciências dos Dados fossem abordadas em diferentes alturas do estágio.

5.1. LIÇÕES APRENDIDAS

Apesar dos desafios encontrados no início, com a ajuda de toda a equipa, foi possível ganhar confiança e autonomia que me permitiram ter mais responsabilidades e começar projetos de forma mais independente. Projetos esses que tiveram um impacto direto nos colaboradores do departamento, e isso foi visível.

A lição principal que aprendi durante estes 8 meses foi a fazer uma boa leitura de problemas e limitações ao negócio, e a convertê-los em soluções com apoio à tecnologia e aos dados. Para apresentar a solução para cada um dos problemas encontrados, foi importante fazer um planeamento estruturado das fases por onde cada um desses projetos iria passar. Apesar de, no início, terem sido feitos planeamentos que acabaram por não espelhar o desenvolvimento dos projetos, ao longo do tempo fui aprendendo a fazer um melhor levantamento dos requisitos que cada projeto precisaria, e do tempo de conclusão de cada um deles.

Com a formação inicial, mas também com o seu uso intensivo, foi possível dominar o *framework* Django. Esta veio a revelar-se muito útil e prática, e engloba várias áreas de conhecimento (desde a modelação à visualização dos dados), que me permitiu aprender a trabalhar com várias áreas da Ciência dos Dados, o que foi uma excelente rampa de lançamento.

5.2. DESAFIOS E LIMITAÇÕES

Tal como nos objetivos, houve limitações que foram identificadas logo desde o início do estágio. O Banco de Portugal, sendo uma instituição quase bicentenária, utiliza algumas tecnologias e ferramentas que por vezes não são compatíveis com as práticas mais inovadoras e avançadas, e que por isso precisam de ser migradas. Não quer dizer que com isto não haja um interesse de todas as partes que esta transição seja feita. Simplesmente, esta mudança é gradual e passa primeiro pela formação dos colaboradores. Como não é possível formar todos os colaboradores em todas as diferentes áreas em mudança de uma só vez, este processo leva tempo. E essa foi uma das maiores limitações que encontrei – por vezes os colaboradores ainda não estarem aptos para utilizarem

determinadas ferramentas que facilitariam o trabalho de todos (a título de exemplo, saber interagir com a plataforma de carregamento de dados RPS). Ora, isto cria lacunas que não são possíveis de resolver a curto prazo. A qualidade dos dados produzidos e que são o ponto de partida de todo o trabalho a desempenhar cria, logo de início, um teto limitador e aquém da capacidade máxima. Para além disso, esta limitação requer também um trabalho redobrado de validação e limpeza dos dados existentes antes de estes serem utilizados, onde muito tempo é investido.

5.2.1. DASboards

Quando foi estudada a possibilidade de desenvolver *dashboards* de apoio à decisão, uma das hipóteses que estava em cima da mesa era a sua criação através de PowerBI. Mas, depois de recolhidos todos os requisitos e informações necessárias ao seu desenvolvimento, chegou-se à conclusão de que a estrutura existente no PowerBI não seria suficiente para a complexidade dos *dashboards* que se pretendia desenvolver. Desde logo, isto criou um desafio acrescido, por todo o trabalho de visualização ter de ser criado de raiz, sem a ajuda de nenhum software. Para além deste obstáculo ao PowerBI, a complexidade dos dados também criou um outro desafio na transformação dos dados. Para ser possível obter informação sobre determinados temas (por exemplo, obter o número de dias de atraso que um processo tem, numa determinada fase), foi necessário criar um enorme nível de customização e que, em alguns momentos, foi desafiante.

5.2.2. DAStat

No DAStat, a maior batalha foi a de conseguir uma interface que fosse fácil de utilizar e com todas as características necessárias. O intervalo de tempo para o desenvolvimento desta aplicação não permitiu criar uma aplicação com um design muito desenvolvido, e isso fez com que fosse menos *user friendly* que o esperado. Sendo que esta aplicação é o espelho da base de dados, o nome dos campos também o era e, portanto, isto criava um problema de compreensão dos campos a selecionar para obter determinados dados, por parte dos colaboradores. Sendo que o *design* ainda não está como esperado, mantém-se um desafio contínuo de, pouco a pouco, o ir melhorando.

5.2.3. *Files Automation*

Na automatização do preenchimento de ficheiros, o requisito mais complexo com que me deparei foi o nível de detalhe de cada um dos relatórios. Primeiro, porque requeria analisar todos os dados que estavam em bruto para extrair de lá informação útil. Segundo, porque de forma que estes automatismos fossem utilizados pelos colaboradores, o resultado não podia ser diferente dos ficheiros que estavam habituados a preencher manualmente. Isso requereu o preenchimento dos modelos de forma minuciosa, e que se revelou um desafio.

5.3. TRABALHO FUTURO

Apesar do trabalho desenvolvido durante este estágio ser notório, ficaram algumas pontas soltas por resolver. Também durante o desenvolvimento do ETL, surgiram novas ideias para projetos, mas que tiveram de ser colocadas em fila de espera, para serem desenvolvidas mais tarde. Essas novas ideias de aplicações enriquecerão o portal já existente, tornando-o ainda mais central e útil a cada vez mais colaboradores no departamento.

O ETL existente continuará a adaptar-se às necessidades, mas manterá a estrutura atual. Uma das maiores necessidades de adaptação que se antevê é a integração de uma segunda base de dados já existente no departamento (utilizada por duas das áreas do departamento), no ETL. O objetivo é ligar essa informação às aplicações já existentes (e criar novas aplicações) para poder servir mais colaboradores. Mas, para que isso aconteça, tem de haver uma estruturação dos dados (sendo que neste momento estão semiestruturados) e um posterior tratamento. Tem também de haver um acolhimento dessa informação de forma diferente da existente para a base de dados atual, na fase de *Extract*.

Uma das propostas criada por mim para o pós-estágio, é a reformulação de algumas aplicações do portal (a maioria criadas antes de setembro de 2021, mas o DAStat também estará incluído) de forma a terem uma navegação mais amigável e percetível, do ponto de vista do utilizador.

Uma outra proposta já aceite será a criação e gestão de um repositório centralizado de documentos para as diferentes áreas do departamento, numa área criada para o efeito dentro do portal, e que estará dividida pelos vários tipos de documentos.

6. BIBLIOGRAFIA

[1] Banco de Portugal. Wikipedia.org. Retrieved 2 August 2022, from https://pt.wikipedia.org/wiki/Banco_de_Portugal.

[2] Banco de Portugal. Bportugal.pt. Retrieved 12 August 2022, from https://bportugal.pt/page/plano-estrategico.

[3] Provost, F. & Fawcett, T. (2013). Data Science and its Relationship to Big Data and Data-Driven Decision Making. Big Data. 51-59. https://doi.org/10.1089/big.2013.1508.

[4] Provost, F. & Fawcett, T. (2013). Data Science for Business. Available at: https://www.oreilly.com/library/view/data-science-for/9781449374273/ch01.html.

[5] El-Sappagh, S. H. A., Hendawi, A. M. A. & El-Bastawissy, A. H. (2011). A proposed model for data warehouse ETL processes. Journal of King Saud University - Computer and Information Sciences, Volume 23, Issue 2. https://doi.org/10.1016/j.jksuci.2011.05.005.

[6] Tobin, D. (2020). ETL & Data Warehousing Explained: ETL Tool Basics. ttps://www.integrate.io/blog/etl-data-warehousing-explained-etl-tool-basics/

[7] What it ETL? The Ultimate Guide. Matillion.com. Retrieved 19 September 2022, from https://www.matillion.com/what-is-etl-the-ultimate-guide/

[8] Taylor, D. (2022). ETL (Extract, Transform, and Load) Process in Data Warehouse. Retrieved from guru99.com: https://www.guru99.com/etl-extract-load-process.html

[9] Simitsis, A. (2005). Mapping conceptual to logical models for ETL processes. In Proceedings of the 8th ACM international workshop on Data warehousing and OLAP (DOLAP '05). Association for Computing Machinery. 67–76. https://dl.acm.org/doi/10.1145/1097002.1097014

[10] García, S., Ramírez-Gallego, S., Luengo, J., Benítez J. M. & Herrera, F. (2016). Big data preprocessing: methods and prospects. Big Data Analytics, Volume 1, 9. https://doi.org/10.1186/s41044-016-0014-0

[11] Lawton G. (2022). Data Preprocessing. Techtarget.com. Retrieved from techtarget.com: https://www.techtarget.com/searchdatamanagement/definition/data-preprocessing

[12] Sharma, R. (2022). 6 Methods of Data Transformation in Data Mining. Retrieved from upgrad.com: https://www.upgrad.com/blog/methods-of-data-transformation-in-data-mining/

[13] Gonçalves, L. (2021). Metodologia Agile, tudo o que precisa de saber sobre este tema. Retrieved from daptmethodology.com: https://adaptmethodology.com/pt-pt/o-que-e-metodologia-agile/

[14] Espírito Santo, D. (2022). Top 5 main Agile methodologies: advantages and disadvantages. Retrieved from xpand-it.com: https://www.xpand-it.com/blog/top-5-agile-methodologies/

[15] Python (programming language). Wikipedia.org. Retrieved 5 September 2022, from https://en.wikipedia.org/wiki/Python_(programming_language)

[16] Django Architecture – Detailed Explanation. Interviewbit.com. Retrieved 5 September 2022, from https://www.interviewbit.com/blog/django-architecture/

[17] SQL. Wikipedia.org. Retrieved 5 September 2022, from https://en.wikipedia.org/wiki/SQL

[18] Introdução ao GitHub: como funciona a plataforma e suas principais funções. Digitalhouse.com. Retrieved 9 October 2022, from https://www.digitalhouse.com/br/blog/github-para-iniciantes/

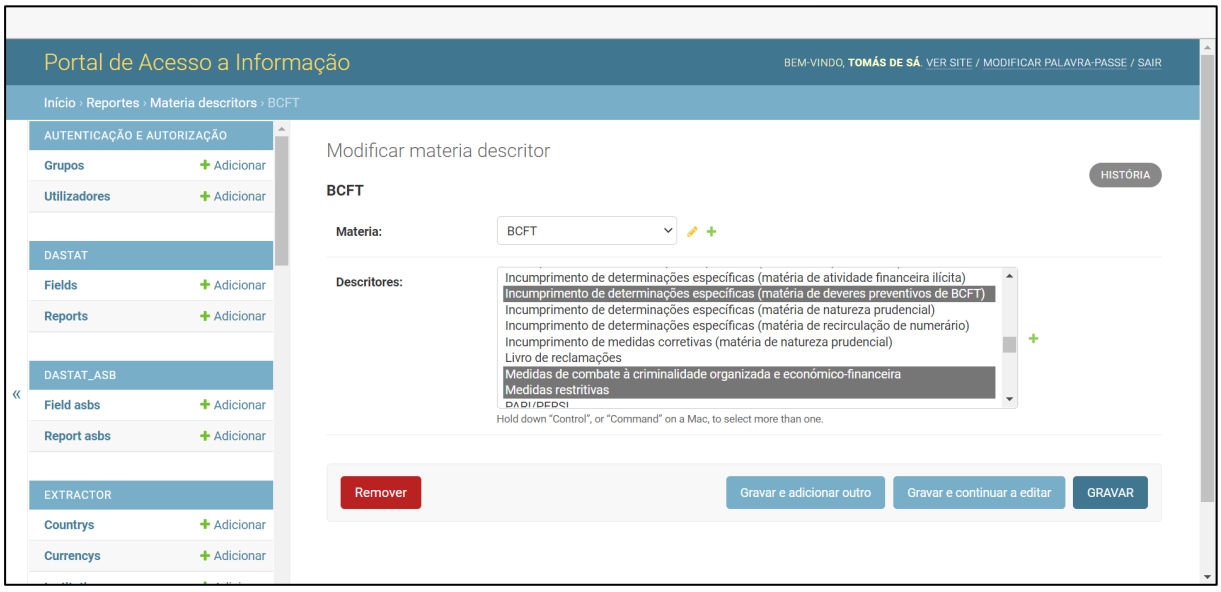
[19] Morris, A. (2021). Data Modeling Explained: Types & Benefits. Retrieved from netsuite.com: https://www.netsuite.com/portal/resource/articles/data-warehouse/data-modeling.shtml

[20] Ali, J. (2019). Understanding Django's Work Flow. Retrieved from medium.com: https://medium.com/@jvdali966/understanding-djangos-work-flow-1ee521422092

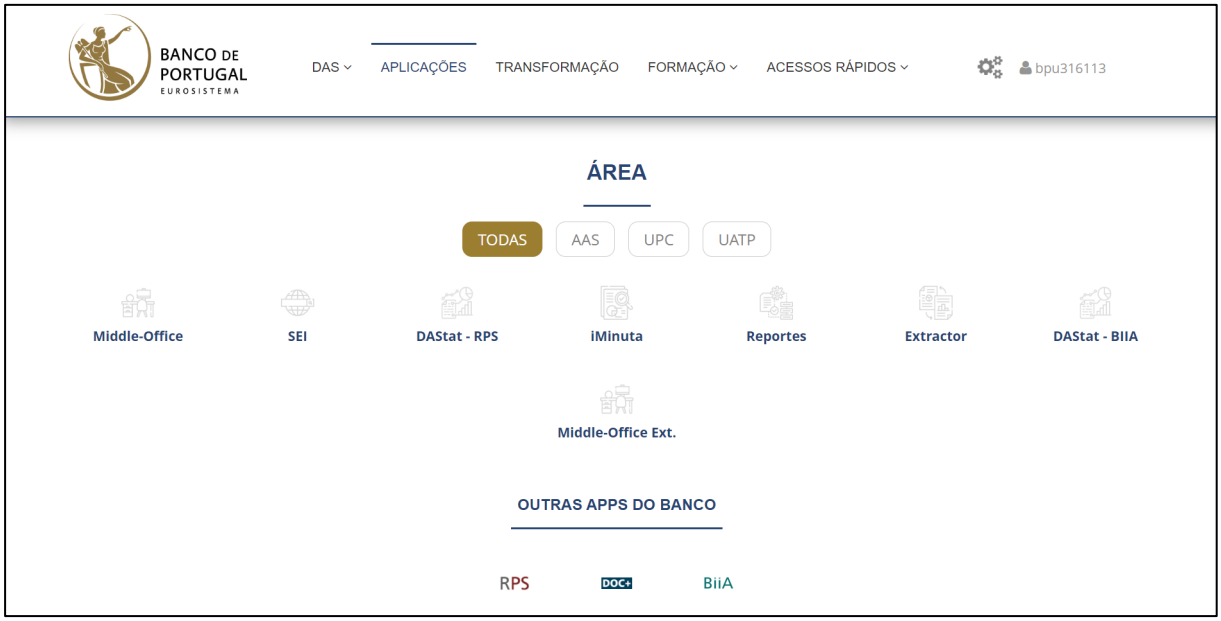
[21] Teixeira, M. (2022). A transformação digital na banca e a crescente relevância da resiliência operacional. Revista Electrónica de Direito, Volume 28, 2. http://dx.doi.org/10.24840/2182-9845_2022-0002_0008

7. ANEXOS

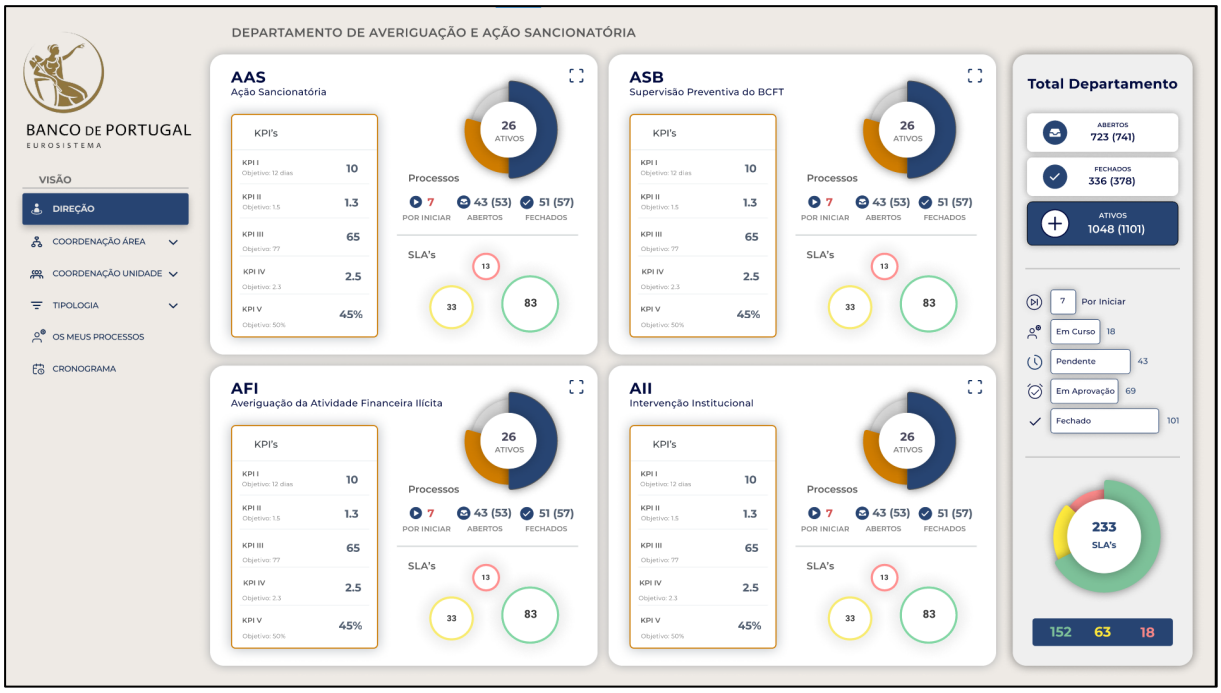
Anexo 1.1 – Página de administrador do Django – Lista de relações Matéria-Descritores



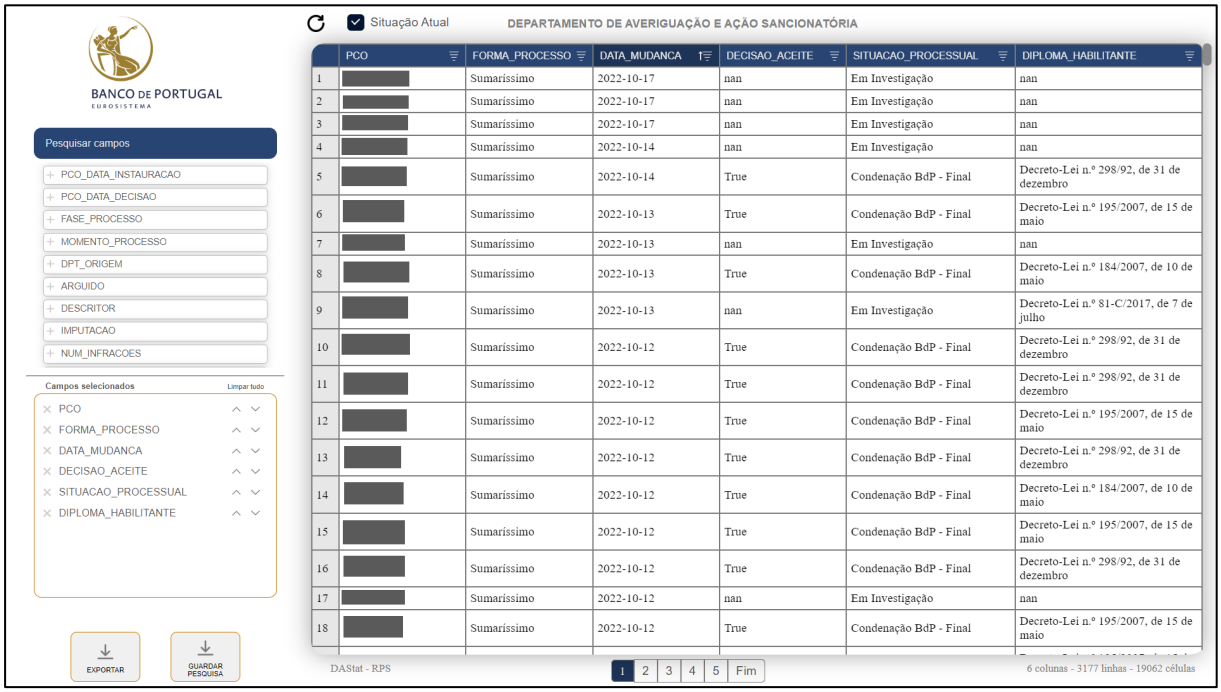
Anexo 1.2 – Página de administrador do Django – Exemplo de uma relação Matéria-Descritores



Anexo 2 – Página com a lista de aplicações do departamento

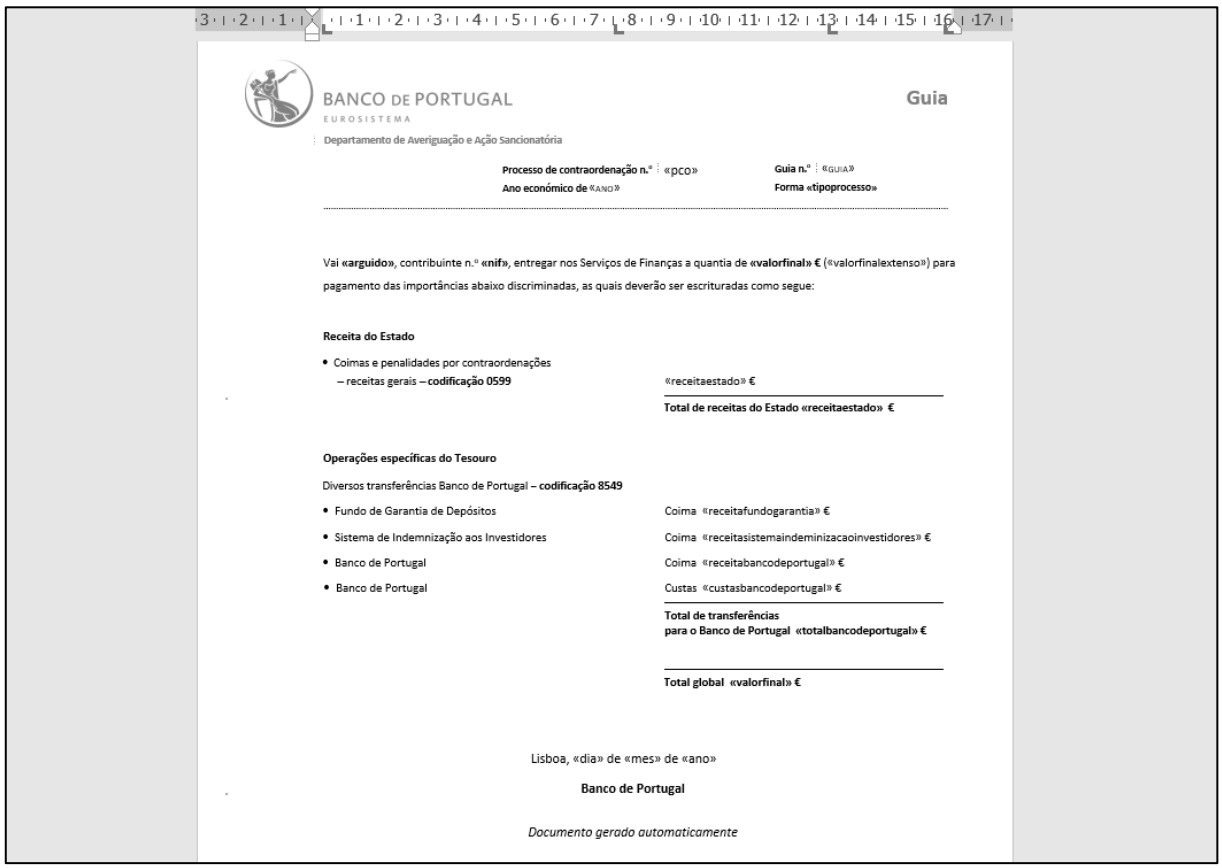


Anexo 3 – Página principal dos DASboards



	PCO	FORMA_PROCESSO	DATA_MUDANCA	DECISAO_ACEITE	SITUACAO_PROCESSUAL	DIPLOMA_HABILITANTE
1	[redacted]	Sumaríssimo	2022-10-17	nan	Em Investigação	nan
2	[redacted]	Sumaríssimo	2022-10-17	nan	Em Investigação	nan
3	[redacted]	Sumaríssimo	2022-10-17	nan	Em Investigação	nan
4	[redacted]	Sumaríssimo	2022-10-14	nan	Em Investigação	nan
5	[redacted]	Sumaríssimo	2022-10-14	True	Condenação BdP - Final	Decreto-Lei n.º 298/92, de 31 de dezembro
6	[redacted]	Sumaríssimo	2022-10-13	True	Condenação BdP - Final	Decreto-Lei n.º 195/2007, de 15 de maio
7	[redacted]	Sumaríssimo	2022-10-13	nan	Em Investigação	nan
8	[redacted]	Sumaríssimo	2022-10-13	True	Condenação BdP - Final	Decreto-Lei n.º 184/2007, de 10 de maio
9	[redacted]	Sumaríssimo	2022-10-13	nan	Em Investigação	Decreto-Lei n.º 81-C/2017, de 7 de julho
10	[redacted]	Sumaríssimo	2022-10-12	True	Condenação BdP - Final	Decreto-Lei n.º 298/92, de 31 de dezembro
11	[redacted]	Sumaríssimo	2022-10-12	True	Condenação BdP - Final	Decreto-Lei n.º 298/92, de 31 de dezembro
12	[redacted]	Sumaríssimo	2022-10-12	True	Condenação BdP - Final	Decreto-Lei n.º 195/2007, de 15 de maio
13	[redacted]	Sumaríssimo	2022-10-12	True	Condenação BdP - Final	Decreto-Lei n.º 298/92, de 31 de dezembro
14	[redacted]	Sumaríssimo	2022-10-12	True	Condenação BdP - Final	Decreto-Lei n.º 184/2007, de 10 de maio
15	[redacted]	Sumaríssimo	2022-10-12	True	Condenação BdP - Final	Decreto-Lei n.º 195/2007, de 15 de maio
16	[redacted]	Sumaríssimo	2022-10-12	True	Condenação BdP - Final	Decreto-Lei n.º 298/92, de 31 de dezembro
17	[redacted]	Sumaríssimo	2022-10-12	nan	Em Investigação	nan
18	[redacted]	Sumaríssimo	2022-10-12	True	Condenação BdP - Final	Decreto-Lei n.º 195/2007, de 15 de maio

Anexo 4 - DAStat



BANCO DE PORTUGAL
EUROSISTEMA

Departamento de Averiguação e Ação Sancionatória

Guia

Processo de contraordenação n.º : «pco» **Guia n.º** : «guia»
Ano económico de «ano» **Forma «tipoprocesso»**

Vai **«arguido»**, contribuinte n.º **«nif»**, entregar nos Serviços de Finanças a quantia de **«valorfinal» €** («valorfinalextenso») para pagamento das importâncias abaixo discriminadas, as quais deverão ser escrituradas como segue:

Receita do Estado

- Coimas e penalidades por contraordenações – receitas gerais – **codificação 0599** «receitaestado» €

Total de receitas do Estado «receitaestado» €

Operações específicas do Tesouro

Diversos transferências Banco de Portugal – **codificação 8549**

- Fundo de Garantia de Depósitos — Coima «receitafundogarantia» €
- Sistema de Indemnização aos Investidores — Coima «receitasistemaindeminizacaoinvestidores» €
- Banco de Portugal — Coima «receitabancodeportugal» €
- Banco de Portugal — Custas «custasbancodeportugal» €

Total de transferências para o Banco de Portugal «totalbancodeportugal» €

Total global «valorfinal» €

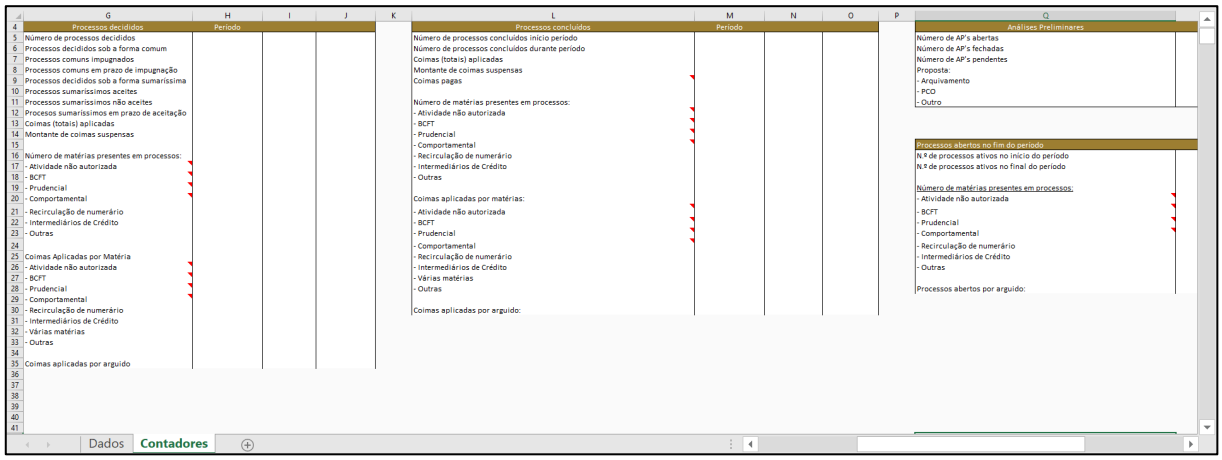
Lisboa, «dia» de «mes» de «ano»

Banco de Portugal

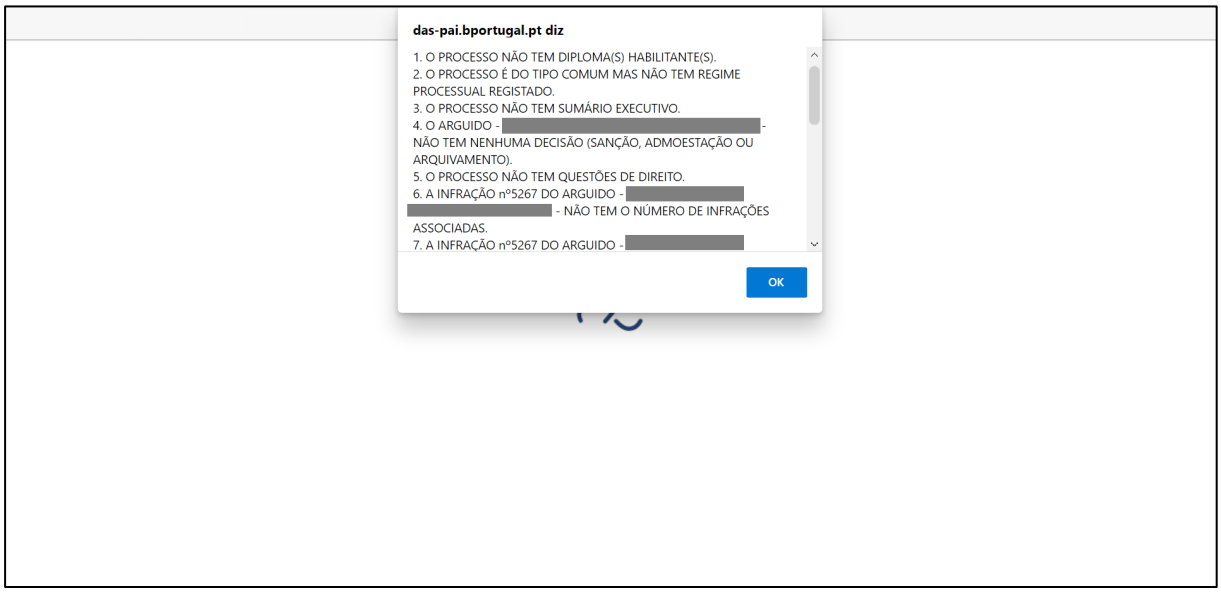
Documento gerado automaticamente

Anexo 5.1 – Modelo Word para preenchimento de uma Guia de Pagamento

Anexo 5.2 – Componente web com seleção de um intervalo de tempo para a extração do Reporte Estatístico



Anexo 5.3 – Modelo Excel para preenchimento dos dados do Reporte Estatístico



Anexo 6 - Exemplo da lista de erros gerados ao extrair uma Guia de Pagamento