POLITÉCNICO DE PORTALEGRE

Escola Superior de Tecnologia e Gestão

Plataforma de elaboração de Relatórios

Integração automatizada de dados de sistemas externos

Luís Manuel Gonçalves de Carvalho

Curso: **Engenharia Informática**

Unidade Curricular: **Projeto**

Orientador: Jorge Miguel Calha Rainho Machado

Ano Letivo: 2023 | 2024

Novembro | 2024

"You're never defeated until you surrender."

Elmer Burritt Bryan, *1921 April 28, Oakland Tribune*.

Agradecimentos

Chegada a altura de ultrapassar mais um obstáculo, não quero deixar de expressar a minha profunda gratidão a todas as pessoas que, de forma direta ou indireta, contribuíram para a concretização deste trabalho.

Em primeiro lugar, quero agradecer aos meus pais, Rui e Carla. Ao meu pai, pela sua constante presença e apoio incondicional. À minha mãe, que, embora não esteja presente fisicamente, a sua memória continua a ser uma fonte de inspiração e força. À minha irmã, Lúcia, pela sua amizade e companheirismo. À minha namorada, Margarida, por todo o amor, carinho e paciência.

Aos meus amigos, saibam que o vosso apoio foi sentido e que apesar de estarmos todos a trilhar os nossos caminhos individuais, onde nos encontrarmos será sempre motivo de celebração.

Um agradecimento especial ao meu orientador de projeto, o Professor Jorge Machado, pela sua orientação, dedicação e conhecimento. Agradecer também aos elementos do pela vossa participação neste importante momento.

A todos os professores que fizeram parte da minha formação, como Helena Taborda, João Miranda, Luís Baptista, Mónica Martins, Paula Amaral, Secundino Lopes, Sérgio Correia, Valentim Realinho e a tantos outros que, porventura, não mencionei aqui. Vocês foram fundamentais para o meu crescimento profissional e pessoal.

Por fim, quero expressar a minha gratidão à Volkswagen Digital Solutions, por me proporcionar a oportunidade de conciliar os estudos com a vida profissional.

A todos, o meu mais sincero obrigado.

Resumo

Este projeto tem como propósito o desenvolvimento de uma plataforma para consumo, transformação e distribuição de dados. Esta solução permite a criação de relatórios personalizados de apoio à decisão organizacional e tem uma forte componente de integração de sistemas, nomeadamente entre o sistema e as fontes de informação. O sistema desenvolveu-se em Python, com base no *framework* Django. O grande objetivo do sistema é criar uma plataforma de "alertas", e coloca-se alertas entre aspas porque o sistema vai além desse objetivo, uma vez que pode emitir informação periódicamente apenas com o propósito de promover a monitorização a quem decide. Os critérios de recolha de informação para composição de relatórios devem pode ser definidos na fonte ou pela configuração local do sistema. Apenas a título de exemplo e para ajudar o leitor a compreender com um caso prático o sistema pode emitir relatórios periódicos com a lista de estudantes com taxa de reprovação superior a X% (por exemplo 70%) para o coordenador de curso detetar casos anómalos rapidamente.

A aplicação é capaz de consumir dados de diferentes formatos (JSON, XML, CSV e texto) através de APIs ou ficheiros. Esses dados podem ser transformados e formatados de acordo com as necessidades específicas de cada configuração, antes de serem distribuídos em diversos formatos de saída (JSON, XML, CSV, texto, e-mail, FTP e PDF). Esta configuração é efetuada através da interface de administração do Django, que permite a gestão da plataforma, onde é possível configurar a origem dos dados, os processos de transformação e agendamento dos mesmos. Desenvolveu-se também um conjunto de funcionalidades específicas, orientadas ao suporte da plataforma escolar PAE, onde as principais diferenças se traduzem na elaboração de relatórios com base numa API própria para o efeito, assente sobre uma estrutura JSON bem definida, que permite uma melhor adaptação a este caso de uso em particular e também, o permitir aos recetores destes relatórios a gestão e consulta da informação que lhes está associada, com recurso a *tokens* que os identificam no sistema.

A plataforma foi desenvolvida, de forma modular, com a intenção de separar responsabilidades e permitir a fácil adição de funcionalidades e manutenção futura.

O suporte de tarefas assíncronas é garantido com recurso às ferramentas Celery e Redis e a base de dados utilizada é SQLite.

Palavras-Chave: plataforma de integração de dados, transformação de dados, automatização de processos, relatórios personalizados.

Lista de Abreviaturas, Siglas e Símbolos

API	Application Programming Interface
BAPI	Business Application Programming Interface
CSV	Comma-Separated Values
ESB	Enterprise Service Bus
ETL	Extract, Transform, Load
FTP	File Transfer Protocol
HTTP	Hypertext Transfer Protocol
IDoc	Intermediate Document
IT	Information Technology
JSON	JavaScript Object Notation
MVT	Model-View-Template
ORM	Object-Relational Mapping – Técnica de mapeamento entre objetos e uma base de dados relacional
PAE	Plataforma escolar – "Sistema de Planeamento Arquitetura de informação Empresarial PAE"
PDF	Portable Document Format
TDD	Test Driven Development (Inglês) – Desenvolvimento Orientado a Teste
XML	eXtensible Markup Language

ÍNDICE GERAL

ÍNDICE DE ANEXOS 9
ÍNDICE DE FIGURAS 10
ÍNDICE DE TABELAS 12
INTRODUÇÃO 13
Enquadramento e Justificação do Tema 14
Objetivos Gerais e Específicos 16
Metodologia e Meios Utilizados 17
Limitações da Pesquisa / Trabalho 18
Estrutura Geral do Trabalho 20
CAPÍTULO I – Fundamentação Teórica 21
1 Conceitos-chave 22
2 Modelos e Teorias 24
3 Padrões de Projeto 27
CAPÍTULO II – Estado da Arte 29
1 Principais desafios 29
2 Tendências 30
3 Soluções existentes 32
4 Lacunas 39
CAPÍTULO III - Metodologia 40
1 Metodologia de desenvolvimento 40
1.1 Tipo de projeto 40
1.2 Definição de Requisitos e Metodologia de desenvolvimento 40
1.3 Arquitetura 41
2 Ferramentas e Tecnologias 42
3 Metodologia de Testes 45
3.1 Testes Realizados 45
3.2 Ferramentas de Teste 45
4 Considerações Éticas 46
4.1 Impactos 46
4.2 Mitigação 47
CAPÍTULO IV – Análise e Desenho da Solução 48
1 Especificações do Problema 48
2 Arquitetura de Dados e Modelo Conceptual 49
3 Fluxo de Dados 52
4 Definição de Interfaces 53

5 Casos de Uso 56
6 Componentes e Separação de Responsabilidades 58
CAPÍTULO V – Implementação 64
1 Ambiente de Desenvolvimento 64
2 Estrutura do Projeto 65
3 Implementação das Funcionalidades 73
3.1 Processo de chamadas periódicas de Serviços 73
3.2 Envio de e-mails 74
3.3 Definição de estratégias da classe "Authentication" 75
3.4 Estratégia de manipulação com base no tipo de input 75
3.5 População de elementos HTML 76
3.6 Tratamento de dados recebidos 77
3.7 Criação de ficheiros PDF 79
4 Desafios e Soluções 80
CAPÍTULO VI – Análise de Resultados 82
1 Instalação e Utilização 82
1.1 Instalação 82
1.2 Interface de Utilizador 82
1.3 Documentação 91
2 Testes Realizados 91
3 Análise dos Resultados 97
CONCLUSÃO 99
Referências Bibliográficas 100
ANEXOS 104

ÍNDICE DE ANEXOS

Anexo 1 – “Definição de Requisitos” 104
Anexo 2 – “Casos de Uso” 104
Anexo 3 – “Diagrama Entidade-Relacionamento Completo” 105
Anexo 4 – “Diagrama de Classes Completo” 106
Anexo 5 – Exemplo de implementação - “Chamada periódica de Serviços” 107
Anexo 6 – Exemplo de implementação - “Lógica de envio de e-mails” 107
Anexo 7 – Exemplo de implementação - “Definição de estratégias da classe “Authentication”” 108
Anexo 8 – Exemplo de implementação - “Tratamento de vários tipos de resposta com base no tipo de input” 108
Anexo 9 – Exemplo de implementação - “Lógica para o popular de templates HTML para a entrega de dados” 109
Anexo 10 – Exemplo de implementação - “Algoritmo genérico para o tratamento de dados JSON” 109
Anexo 11 – Exemplo de implementação - “Lógica para a geração de ficheiros PDF” 109
Anexo 12 – “Diagrama de Classes Completo” 111

ÍNDICE DE FIGURAS

Figura 1 – Diagrama de uma Arquitetura Orientada a Serviços 24
Figura 2 – Arquitetura ETL convencional 25
Figura 3 – Representação Visual do modelo MVT do Django 27
Figura 4 – Diagrama de classes do padrão de design Strategy (exemplo) 27
Figura 5 – Mulesoft Anypoint Studio: IDE para assitir o processo de integração 32
Figura 6 – SAP Integration Suite: Ferramenta de modelação de integrações 33
Figura 7 – Zato: Exemplo da interface de utilizador #1 34
Figura 8 – Zato: Exemplo da interface de utilizador #2 35
Figura 9 – SendGrid: Relatório de alcance de serviços de e-mail 36
Figura 10 – Mailchimp: Ecrã de gestão de campanhas em curso 38
Figura 11 – Diagrama Entidade-Relacionamento da aplicação 49
Figura 12 – Fluxo de dados da plataforma 52
Figura 13 – Àrea de administração: página inicial 53
Figura 14 – Àrea de consulta de APIs: alteração de serviço 53
Figura 15 – Àrea de consulta de APIs: página inicial 54
Figura 16 – Àrea de consulta de APIs: serviços disponíveis 55
Figura 17 – Caso de Uso: criação de um Serviço 56
Figura 18 – Caso de Uso: Chamada de serviço periódica 57
Figura 19 – Caso de Uso: Chamada de serviço manual 58
Figura 20 – Componentes da plataforma 58
Figura 21 – Estrutura de pactoes do projeto 59
Figura 22 – Exemplo de um ficheiro “.env” 60
Figura 23 – Pasta “data” 61
Figura 24 – Pasta “static” 62
Figura 25 – Pasta “templates” 62
Figura 26 – Pasta “templates/pdf” 62
Figura 27 – Pasta “tests” 63
Figura 28 – Pasta “utils” 63
Figura 29 – Diagrama de classes da plataforma 65
Figura 30 – Classe “Serviço” 65
Figura 31 – Classe “Authentication” 67
Figura 32 – Classe “AuthenticationStrategy” e subclasses 67
Figura 33 – Classe “PAEAuthentication” 68
Figura 34 – Classe “Destination” 68
Figura 35 – Classe “PAEDestination” 69
Figura 36 – Classe “PAEUserAccessToken” 69
Figura 37 – Classe “PAESentReport” 70
Figura 38 – Classe “PAEEmail” 70
Figura 39 – Classe “PAEAttachment” 70
Figura 40 – Classe “RunnerResponse” 71
Figura 41 – Classe “Attachment” 71
Figura 42 – Classe “ServiceAction” 72
Figura 43 – Classe “PAEToken” 72
Figura 44 – Classe “OverwriteStorage” 72
Figura 45 – Classe “Log” 73
Figura 46 – Representação do fluxo de chamadas periódicas de Serviços 73
Figura 47 - Representação do fluxo para o envio de e-mails 74

Figura 48 – Exemplo de elemento HTTP gerado 76
Figura 49 – Exemplo da Pipeline de processamento e transformação dos dados recebidos ... 77
Figura 50 – Estrutura JSON do PAE 78
Figura 51 – Fluxo de geração dos ficheiros PDF 79
Figura 52 – DjangoWeaver: Página Inicial 83
Figura 53 – DjangoWeaver: Página de Administração 83
Figura 54 – DjangoWeaver: Consulta de destinatários configurados 84
Figura 55 – DjangoWeaver: Consulta dos serviços configurados 85
Figura 56 – DjangoWeaver: Consulta dos serviços disponíveis para chamadas imediatas 85
Figura 57 – DjangoWeaver: Exemplo de um relatório produzido #1 86
Figura 58 – DjangoWeaver: Exemplo de um relatório produzido #2 86
Figura 59 – DjangoWeaver: (PAE) Remoção da lista de subscritores 86
Figura 60 – Exemplo de um e-mail recebido #1 87
Figura 61 – Exemplo de um e-mail recebido #2 87
Figura 62 – DjangoWeaver: (PAE) Remoção da lista de subscritores 88
Figura 63 – DjangoWeaver: (PAE) Remoção da lista de subscritores - sucesso 88
Figura 64 – DjangoWeaver: (PAE) Gestão de serviços subscritos 89
Figura 65 – DjangoWeaver: (PAE) Histórico de e-mails recebidos 90
Figura 66 – DjangoWeaver: (PAE) Pedido de atualização de *tokens* - sucesso 90
Figura 67 – Estrutura da pasta de testes 91
Figura 68 – Caso de Teste: Geração de elementos HTML 95
Figura 69 – Caso de Teste: Entrega de dados via FTP 95
Figura 70 – Caso de Teste: Envio de e-mails 96
Figura 71 – Caso de Teste: Produção de ficheiros PDF 96

ÍNDICE DE TABELAS

Tabela 1 – Receita de tecnologias de desenvolvimento “low-code” (milhões US$).....31
Tabela 2 – Comparação entre os produtos MuleSoft Anypoint Platform, SAP Integration Suite e Zato 35
Tabela 3 - Relação entre produtos comerciais e open-source.....38
Tabela 4 - Resumo das diferenças: RabbitMQ vs. do Redis.....43
Tabela 5 – Exemplo do corpo de uma chamada a um serviço do PAE 50
Tabela 8 – Definição de estratégias da classe “Authentication”.....75
Tabela 9 – Tratamento de vários tipos de resposta com base no tipo de input.....75
Tabela 10 – Lógica para o popular de templates HTML para a entrega de dados.....76
Tabela 11 – Algoritmo genérico para o tratamento de dados JSON.....77
Tabela 12 – Exemplo dos registos da tabela PAEAttachment.....78
Tabela 13 – Exemplo dos registos da tabela PAEEmail.....78
Tabela 14 - Exemplo dos registos da tabela PAEDestination.....79
Tabela 16 – Exemplo de um teste de mapeamento.....92
Tabela 17– Teste de geração automática de relatórios com diferentes tipos de dados.....93
Tabela 18– Teste de cálculo do último dia do mês com base na data do sistema 94

INTRODUÇÃO

Correndo atrás do desenvolvimento tecnológico, as empresas sempre procuraram forma de tornar os seus processos mais eficientes e, consequentemente, mais informatizados. Esta evolução, heterogénea e a diferentes ritmos, criou a necessidade de um novo produto de software: soluções tecnológicas capazes de integrar e analisar grandes volumes de dados provenientes de diversas fontes.

Neste contexto, o presente trabalho propõe o desenvolvimento de uma plataforma de integração em Python, focada no consumo de dados heterogéneos e na geração de relatórios personalizados, atendendo às necessidades de diferentes sistemas e utilizadores.

A plataforma foi construída com recurso ao *framework open-source* Django, cujo propósito é o de permitir o desenvolvimento rápido e um design de soluções pragmático e simples [1]. Esta escolha foi motivada pela popularidade das ferramentas Python e Django, evidenciada pelo grande número de bibliotecas disponíveis. Adicionalmente, tomou-se em conta a disponibilidade de materiais de aprendizagem, dado ser uma tecnologia nova a explorar.

O fator diferenciador deste projeto será a capacidade de integrar dados de diversas fontes, unificando a entrega de dados a sistemas e utilizadores, sendo possível a sua entrega em formatos de saída como CSV, PDF e HTML. Aliado a isso, surge também o desenvolvimento de funcionalidades específicas para dar suporte à plataforma escolar PAE, como a criação de um modelo de geração de relatórios dedicado e o acesso de utilizadores externos à plataforma para a gestão e consulta da informação distribuída pela mesma.

Enquadramento e Justificação do Tema

A integração de sistemas e a geração de relatórios são desafios comuns em diversas áreas, especialmente no contexto empresarial. A crescente complexidade dos sistemas de informação e a necessidade de tomar decisões baseadas em dados robustos e em tempo-real impulsionam a busca das empresas por soluções eficientes e escaláveis.

No campo do software empresarial intervenientes como a SAP, a Oracle e a IBM dominam o espaço, contando com anos de experiência e desenvolvimento na área e uma forte adoção na sua gama de serviços. O principal exemplo disso é a SAP, empresa líder no mercado do software ERP [2].

Plataformas como Mulesoft Anypoint Platform, SAP Process Integration (PI) e SAP Integration Suite (IS) são exemplos de soluções comerciais consolidadas no mercado de integração de sistemas. Estas ferramentas oferecem uma ampla gama de funcionalidades, incluindo conectores pré-construídos para diversos sistemas, transformação de dados, orquestração de processos e geração de relatórios. Este tipo de soluções geralmente acarreta custos elevados de licenciamento, onde se tem verificado um aumento de soluções que seguem modelos de subscrição [3] [4] [5].

A plataforma Zato, uma solução *open-source* para a integração de sistemas desenvolvida em Python, representa uma alternativa economicamente viável e tecnologicamente robusta para este tipo de software. Com um modelo de negócio gratuito é possível oferecer este tipo de serviços a quaisquer empresas interessadas, sendo a principal fonte de rendimento a oferta de formações técnicas e serviços de suporte especializado [6].

O mercado apresenta um potencial de crescimento significativo [2] [7], impulsionado pela transformação digital. No entanto, enfrenta desafios complexos como a integração de sistemas heterogéneos, a garantia da qualidade dos dados e o cumprimento de legislação cada vez mais rigorosa. A segurança da informação, a escassez de pessoal especializado e a complexidade dos ambientes de IT são outros obstáculos a superar. As organizações que conseguirem inovar, adotar tecnologias emergentes e desenvolver uma cultura de dados sólida estarão bem posicionadas para aproveitar as oportunidades do mercado e obter uma vantagem competitiva. Em adição, a interpretação de interfaces proprietárias, como os IDocs e BAPIs do SAP, em formatos mais transversais e abertos representa um passo fundamental para a interoperabilidade entre sistemas e a democratização do acesso à informação.

Este projeto visa oferecer uma forma simples de transformar dados para integração, oferecendo a conversão de formatos e a possibilidade de os distribuir através de vários tipos de comunicação, como e-mail, PDF, FTP e HTTP. Tendo sido construído com uma base modular, de acordo com a sua responsabilidade no processo, este software pretende-se mais fácil de adicionar funcionalidades e realizar tarefas de manutenção.

A relevância deste desenvolvimento prende-se por tentar, principalmente, oferecer uma solução genérica e extensível a um grande número de casos, procurando oferecer várias formas de trabalhar e entregar vários formatos de dados de *input*.

Ao contrário dos produtos de software mencionados, que requerem custos de licenciamento e pessoal especializado para a sua operação [5] [4], ou especificamente no caso do Zato em que existe uma oferta de uso gratuito [8], este produto procurou diferenciar-se por oferecer uma alternativa mais acessível a todos os utilizadores, com a exibição de relatórios via e-mail ou PDF e sem a necessidade de infraestrutura complexa.

A escolha deste tema foi motivada pela convergência de interesses pessoais e académicos. O desenvolvimento em Python, conhecido pela sua versatilidade, é uma competência que se procurou desenvolver. Isto aliado à temática de integração de sistemas proporcionou a oportunidade de aliar estes dois interesses e desenvolver um projeto onde foi possível a aquisição de experiência prática relevante.

Objetivos Gerais e Específicos

O principal objetivo deste projeto passa pelo desenvolvimento de uma plataforma capaz de lançar alertas e elaborar relatórios com base em dados obtidos através de pedidos efetuados a sistemas externos. Com isto, pretende-se almejar a melhoria dos processos de tomada de decisão, oferecendo aos utilizadores da plataforma informação atualizada e notificações pertinentes com base em parâmetros pré-definidos.

Especificamente, os principais requisitos a atender focaram-se no seguinte:

- Integração de sistemas: a capacidade de poder receber dados de um sistema fonte e entregá-los a um outro sistema ou diretamente a um utilizador, por meio de e-mail.

- Geração de relatórios: com base nos dados recebidos, poder exibir relatórios que contenham informação relevante num formato acessível e interpretável, como documentos PDF.

- Transformação de dados: ser capaz de processar vários tipos de dados e convertê-los em outros formatos, para aplicação em diferentes cenários, sejam elementos HTML com a informação relevante, seja a conversão de dados XML para JSON ou a representação destes em documentos ou ainda, a entrega de e-mails diretamente aos utilizadores relevantes.

- Resultados: demonstrar a capacidade de a plataforma atender a, pelo menos três, chamadas de serviços com características diferentes e oferecer outputs personalizados com sucesso. A isto aliou-se o desenvolvimento específico de funcionalidades de suporte à plataforma PAE.

Metodologia e Meios Utilizados

O processo de desenvolvimento iniciou em abril, com uma análise da viabilidade das ferramentas Python e Django para a elaboração do projeto.

A plataforma foi desenvolvida segundo uma abordagem híbrida, combinando o modelo de desenvolvimento de *software Waterfall* (em português, cascata) para a estrutura geral do projeto recorrendo ao desenvolvimento orientado a testes (TDD) para funcionalidades específicas. Esta foi a abordagem escolhida pois pretendeu-se a elaboração de um MVP (Minimum Viable Product) com a maior brevidade, com o objetivo de satisfazer os requisitos principais primeiro e servir de base para o aprimorar e definição das funcionalidades seguintes. Uma vez que o projeto teve um escopro inicial bastante alargado, procurou-se assim estruturar a abordagem inicial nos seus pontos vitais e conforme o feedback obtido, ajustar a implementação de forma evolutiva, o que se veio a confirmar com o foco no suporte da plataforma PAE e definição de funcionalidades exclusivas para a integração desta plataforma.

A opção pelo TDD permitiu a implementação de funcionalidades com a expectativa que estas atendessem aos requisitos, mesmo antes de se realizarem testes mais abrangentes, destes testes destacam-se: as rotinas de elaboração de PDFs, sem a necessidade de invocar os serviços; a entrega de dados processados via FTP ou e-mail, como prova de conceito.

A arquitetura da aplicação foi baseada no padrão Model-View-Template (MVT) do Django, com o uso de tecnologias como Python, Django, Celery e SQLite. A seleção dessas ferramentas partiu do objetivo de manter o foco no desenvolvimento da plataforma em si, delegando pontos como a gestão do modelo de dados no caso do Django por meio do Django ORM, ou a gestão de tarefas assíncronas, com recurso ao Celery, utilizando a base de dados em tempo real Redis para assegurar a comunicação entre a aplicação e o Celery.

Procurou-se dividir a aplicação num contexto modular, separando as principais responsabilidades pelas aplicações: *Scheduler*, *Runner* e *Dispatcher*.

Este tópico será depois analisado em mais detalhe no “CAPÍTULO III - Metodologia”.

Limitações da Pesquisa / Trabalho

Para a realização deste projeto foram identificadas as seguintes limitações:

Informação comercial sobre plataformas empresariais não está disponível de uma forma clara, na sua generalidade, isto tornou difícil o recurso a dados que pudessem suportar o argumento de que este tipo de soluções acarreta custos elevados para as empresas.

Recorrendo principalmente a APIs disponíveis na internet de uso livre, não foi possível a elaboração de testes de integração exaustivos e com todos os modelos de autenticação previstos na implementação. Aqui também é possível ressalvar a existência de uma amostra de dados algo diminuída, sendo desejável ter acesso a informações mais heterogéneas e em maior quantidade.

O desenvolvimento tomou início no final de Abril, o que teve efeito no tempo disponível para a elaboração da plataforma como um todo.

A metodologia escolhida levou à tomada de decisões que inicialmente direcionaram o desenvolvimento num sentido que, posteriormente, dificultou o redesenhar de certas funcionalidades e limitou possíveis alterações à arquitetura inicialmente prevista.

A opção pelo uso de ferramentas novas, também levou a um tempo de desenvolvimento superior ao esperado de alguém já familiarizado com estas ferramentas.

Uma característica do Python digna de ressalva é o recurso à atribuição dinâmica de tipos, isto significa que os tipos das variáveis do sistema são determinados em tempo de execução, isto leva a que possam ocorrer erros que só sejam detetados em tempo de execução, sendo mais difícil a determinação de uma causa e dificultando a sua análise.

O recurso ao Django ORM (Object-Relational Mapping) para a realização de consultas na base de dados leva automaticamente a uma perda de desempenho na interação com a mesma, isto pode ser ultrapassado com recurso a *queries* nativas da base de dados implementada. Com isto, no entanto, perde-se alguma flexibilidade caso se queira proceder a uma futura migração.

O foco da implementação foi alterado numa fase já avançada do calendário escolar, passando de um objetivo muito genérico para algo mais particular e mensurável, como é o caso da implementação de funcionalidades de suporte à plataforma PAE. Nesse contexto, os desenvolvimentos pré-existentes foram considerados de alguma forma legados. Procurou-se assim preservar toda a funcionalidade já existente e reaproveitar, dentro do possível, o trabalho

já desenvolvido para suportar a lista de funcionalidades específicas a desenvolver de modo a dar suporte aos requisitos da plataforma escolar.

Estrutura Geral do Trabalho

Este trabalho está estruturado em seis capítulos principais.

O primeiro capítulo organiza os pressupostos teóricos que serviram de base à elaboração deste trabalho, introduzindo os conceitos mais relevantes, padrões de desenvolvimento de software e abordagens consideradas para a tarefa de desenvolvimento.

No segundo capítulo aborda-se qual o estado atual da indústria, quais são os produtos de software dominantes no mercado e como estes se diferenciam uns dos outros, bem como tendências observáveis neste espaço. Por último, serão apresentados pontos em que este tipo de soluções tende a ter dificuldade em responder.

Após a consideração destes pontos tem início, efetivamente, a descrição do processo de desenvolvimento desta aplicação. No terceiro capítulo são detalhados os processos que levaram à elaboração dos requisitos definidos para a plataforma, bem como as ferramentas e as metodologias utilizadas para o desenvolvimento da mesma.

O quarto capítulo aborda o problema que se pretende resolver e define-se qual a estrutura base que a solução deve tomar, quais os fluxos e a relação entre as entidades definidas, estes servem de base ao processo de implementação.

O desenvolvimento da solução em si é detalhado no quinto capítulo, onde são apresentadas as características do ambiente de desenvolvimento e como a solução foi estruturada, quais os principais obstáculos e como foram desenhados os processos que levaram à implementação das funcionalidades planeadas.

A sexta secção avalia os resultados obtidos e como as opções tomadas influenciaram o destino desta implementação, suportados por toda a informação resultante do fim da implementação, como a documentação e resultados dos testes efetuados.

Por último, é apresentada uma conclusão onde se reflete sobre os resultados obtidos e se oferecem sugestões para desenvolvimento futuro.

CAPÍTULO I – FUNDAMENTAÇÃO TEÓRICA

Esta plataforma foi desenvolvida com o intuito de integrar APIs de vários formatos de entrada e contextos. Procurou-se, com as várias estruturas de dados disponíveis para integração, oferecer uma solução genérica, capaz de atender aos requisitos de transformação de dados e a sua representação em vários formatos, alguns visuais, para a consulta de utilizadores, como é o caso da representação de dados em ficheiros PDF ou elementos HTML outros associados aos formatos de representação de dados como JSON ou XML.

Para este efeito, tomou-se como fonte de informação vários exemplos disponíveis na internet de forma gratuita e em adição, as interfaces disponibilizadas para a comunicação com a plataforma PAE.

Em seguida, são detalhadas as bases teóricas que serviram de suporte à estruturação desta aplicação e às decisões tomadas ao longo do processo de desenvolvimento.

1 Conceitos-chave

Integração de Sistemas:

A integração de sistemas é o foco central deste projeto, definida como a prática de conectar diferentes sistemas e aplicações para permitir a comunicação fluida e a troca de dados entre eles [9] [10]. As APIs (Application Programming Interfaces) desempenham um papel crucial neste contexto, estabelecendo interfaces para a comunicação entre sistemas distintos.

A plataforma oferece uma série de APIs, que facilitam uma integração escalável e versátil através de chamadas HTTP e formatos padronizados. A plataforma desenvolvida procura solucionar desafios de integração entre APIs diversas e atender a múltiplos formatos de dados como JSON, XML e CSV, facilitando a comunicação entre sistemas distintos.

APIs (Application Programming Interface):

APIs são componentes que estabelecem uma interface para comunicação entre diferentes sistemas, definindo regras para o envio e recebimento de dados [11]. Existem diferentes tipos de APIs, como REST e SOAP, cada uma com as suas particularidades na forma como a troca de informação tem lugar. Esta plataforma disponibiliza APIs REST para gerir alguns modelos e comunica com sistemas externos através de APIs disponibilizadas para a obtenção de dados.

Automação de Processos e Geração de Relatórios:

A automação de processos é crucial para a eficiência organizacional, especialmente em tarefas repetitivas como a recolha de dados, transformação e geração de relatórios. Automatizar processos permite que as pessoas possam passar menos tempo a atender a tarefas repetitivas e consequentemente, sejam mais produtivas [12] [13]. A aplicação desenvolvida gera relatórios em PDF e HTML e envia esses relatórios por e-mail, automatizando a entrega da informação e tornando o acesso mais prático para os destinatários. Este exemplo é uma das formas como se pode retirar valor deste tipo de ferramentas, libertando as pessoas da necessidade de produzir estes relatórios manualmente e oferecendo a informação de uma forma mais acessível, assistindo o processo de tomada de decisão.

Transformação de Dados:

Transformação de dados refere-se ao processo de manipular os dados recebidos para que atendam a diferentes formatos e requisitos de saída, um aspecto fundamental para a integração entre sistemas. Este projeto utiliza bibliotecas como o *xmltodict* para processar dados em XML

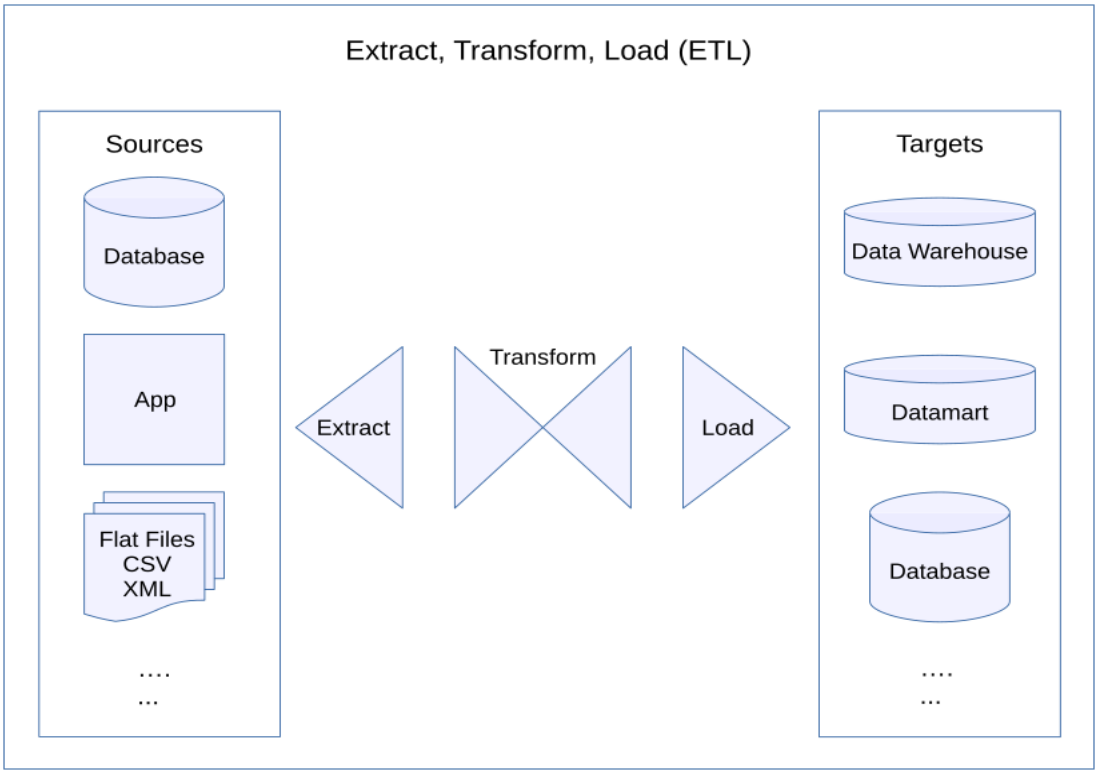
e convertê-los em formatos JSON e PDF, garantindo que as informações sejam entregues no formato mais adequado ao contexto exigido.

2 Modelos e Teorias

Arquitetura de Integração de Sistemas:

A arquitetura de integração de sistemas da plataforma incorpora elementos de duas abordagens principais: a Arquitetura Orientada a Serviços (Service-Oriented Architecture - SOA) e o modelo ETL (Extração, Transformação e Carregamento).

Figura 1 – Diagrama de uma Arquitetura Orientada a Serviços



Fonte: Letha Hughes Etzkorn - Introduction to Middleware: Web Services, Object Components, and Cloud Computing

A plataforma adota princípios da SOA, atuando como um ponto central de integração entre sistemas externos e utilizadores, neste contexto os consumidores destes serviços não têm qualquer perceção de como estes serviços estão estruturados, pois cada serviço representa uma tarefa contida em si mesma [14]. Esta abordagem assemelha-se ao conceito de Enterprise Service Bus (ESB), que pode ser definido como "um bus de alto-nível de comunicações aplicação-para-aplicação" [15], facilitando a comunicação e o fluxo de dados entre múltiplos sistemas. Ao centralizar a integração, a plataforma simplifica a complexidade das interações entre sistemas e utilizadores, proporcionando uma interface singular para a definição de vários serviços.

Além disso, a plataforma implementa o modelo ETL (Extração, Transformação e Carregamento (do inglês, *Load*)). Em cada interação, a aplicação extrai dados de serviços externos através de chamadas HTTP, realiza a transformação desses dados para os formatos desejados (como JSON, XML, PDF ou HTML), e, finalmente, carrega-os, entregando-os aos utilizadores ou outros destinatários. Este processo de ETL garante que os dados são recebidos, processados e distribuídos conforme os requisitos específicos de cada caso, sendo um modelo fundamental para a integração e automação de dados.

Estas características fazem da plataforma um ponto de integração centralizado, com o objetivo de entregar dados transformados de maneira eficiente, escalável e compatível com os padrões modernos de integração de sistemas.

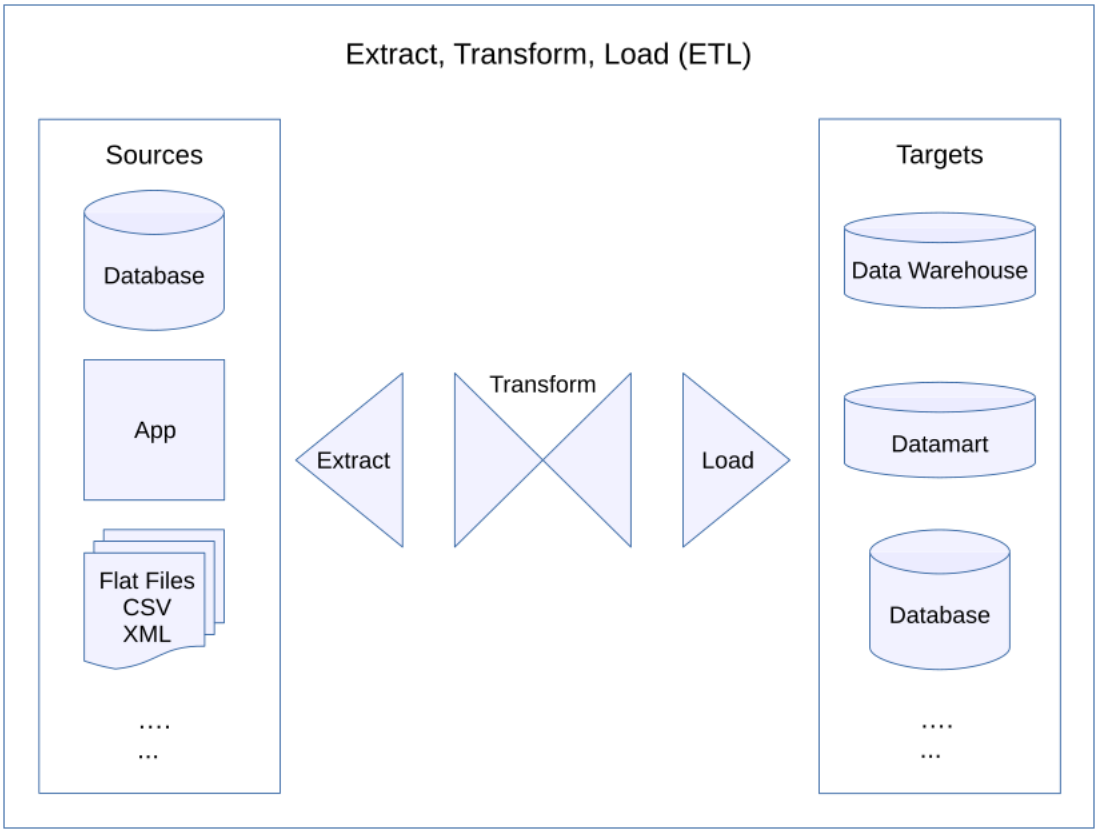
Processo Extract-Transform-Load (ETL):

O ETL é um processo de três etapas que envolve a extração de dados de várias fontes, a transformação desses dados para atender a requisitos específicos e, por fim, o carregamento dos dados transformados para um destino final, no caso desta aplicação podemos falar de sistemas externos ou utilizadores, quer por entrega dos dados via e-mail, quer através de respostas HTTP [16].

O modelo ETL (Extract, Transform, Load) é aplicado de forma eficaz nesta plataforma de geração de relatórios, orientando todo o processo de manipulação de dados.

Inicialmente, a aplicação extrai informações através de APIs, seguindo para a fase de transformação, onde os dados são formatados em estruturas como JSON ou XML. O processo culmina na etapa de carregamento, com a geração de relatórios em PDF ou HTML, que são então distribuídos via e-mail aos destinatários.

Figura 2 – Arquitetura ETL convencional



Fonte: Wikipedia - Extract, transform, load

Agendamento e Execução de Tarefas em Segundo Plano:

O processamento em segundo plano refere-se à execução de tarefas que não requerem interação imediata do usuário e podem ser realizadas independentemente do fluxo principal do sistema [17].

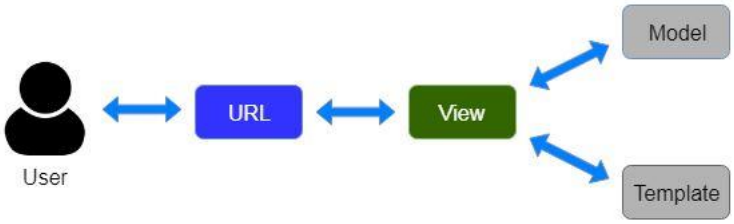
A plataforma utiliza processamento em segundo plano para lidar com tarefas de transformação e distribuição de dados, permitindo que operações de longa duração ocorram sem impactar outras atividades. Com o Celery, as tarefas são agendadas e executadas periodicamente, assegurando que o envio de relatórios e outras atividades críticas aconteçam de forma eficiente e dentro dos intervalos planeados.

3 Padrões de Projeto

Padrão Model-View-Template (MVT):

O padrão Model-View-Template (MVT), como é vulgarmente chamada a adaptação do Django ao modelo MVC (Model-View-Controller), separa as responsabilidades do sistema em camadas distintas. A camada Model representa a estrutura de dados da aplicação, a View é onde reside a lógica de processamento, é aqui que os pedidos são recebidos e caso necessário, é aqui que são definidas as interações com o modelo, a Template define a camada visual que é apresentada ao utilizador [18] [19] [20].

Figura 3 – Representação Visual do modelo MVT do Django



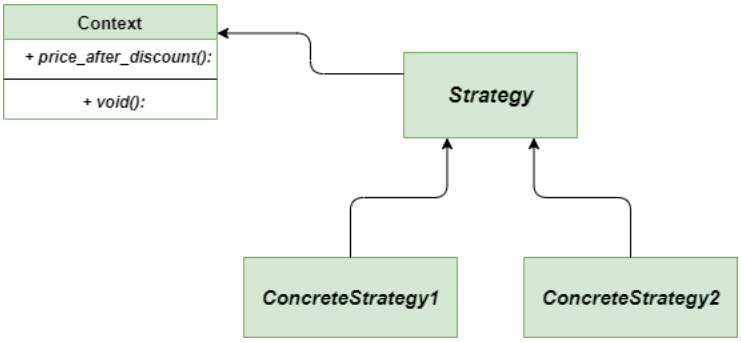
Fonte: Tutorials Point- Django – MVT Architecture

Padrão *Strategy* para Autenticação:

O padrão Strategy foi utilizado para suportar o processo de autenticação. Segundo Freeman e Robson, "O padrão Strategy define uma família de algoritmos, encapsula cada um deles e torna-os intercambiáveis. A estratégia permite que o algoritmo varie independentemente dos clientes que o utilizam" [21]. Este padrão permite definir uma estratégia específica para cada tipo de autenticação suportada pelo sistema, incluindo a implementação de um processo específico para a plataforma PAE. Isso permite que a lógica de autenticação seja adaptada conforme o contexto, sem necessidade de modificar o código principal da aplicação. Esta abordagem torna o sistema mais flexível e permite o suporte de múltiplas formas de autenticação de maneira organizada e modular.

Os tipos standard de autenticação suportados são os seguintes: Autenticação Básica (utilizador + password); Token; Digest; OAuth1.0 e OAuth2.0.

Figura 4 – Diagrama de classes do padrão de design Strategy (exemplo)



Fonte: GeeksforGeeks - Strategy Method – Python Design Patterns

CAPÍTULO II – ESTADO DA ARTE

A integração de sistemas é fundamental para a otimização de processos e a criação de ecossistemas digitais. A crescente complexidade das aplicações e a necessidade de interoperabilidade entre diferentes sistemas motivou e continua a alimentar a evolução de soluções nesta área.

1 Principais desafios

Devido ao contexto onde estas plataformas operam, sendo principalmente associadas ao tecido empresarial, estas são as principais barreiras identificadas:

Complexidade

A crescente quantidade e diversidade de aplicações, dispositivos e fontes de dados em empresas modernas torna a integração uma tarefa desafiadora. Garantir uma comunicação eficiente entre esses sistemas, sem causar inconsistências ou interrupções, requer uma abordagem abrangente que englobe aspetos tecnológicos e operacionais [22] [23].

Segurança

A integração de múltiplos sistemas que lidam com dados sensíveis de negócios e clientes apresenta riscos de segurança consideráveis. Vulnerabilidades durante o processo de integração podem expor os sistemas a violações de dados e acessos não autorizados, potencialmente causando danos financeiros e reputacionais [24].

Integração de Sistemas Legados

A integração de tecnologias modernas com sistemas legados apresenta dificuldades significativas. Sistemas antigos utilizam frequentemente tecnologias obsoletas e software proprietário, tornando-os incompatíveis com soluções mais recentes. Isso exige estratégias cuidadosas de migração gradual e uso de *middleware* para facilitar a comunicação entre sistemas novos e antigos [25].

Escalabilidade

À medida que as empresas crescem, seus sistemas de integração precisam acompanhar esse crescimento sem causar interrupções. Desenvolver uma base robusta que seja moderna, ágil e escalável é essencial para lidar com o aumento da demanda e complexidade dos sistemas [24].

2 Tendências

Na atual conjuntura do mercado de integração de sistemas, observa-se uma clara tendência para soluções que privilegiam a flexibilidade, a eficiência e a adaptabilidade às necessidades específicas das empresas. Esta evolução é impulsionada por diversos fatores tecnológicos e económicos que estão a remodelar o panorama da integração de sistemas.

Um dos desenvolvimentos mais significativos é a adoção crescente de abordagens centradas em APIs (Application Programming Interfaces). Esta estratégia "API-First" está a revolucionar a forma como os sistemas são desenvolvidos e integrados, proporcionando uma maior interoperabilidade e facilitando a criação de ecossistemas digitais robustos [26].

As APIs tornaram-se a espinha dorsal da conectividade moderna, permitindo que diferentes aplicações e serviços comuniquem de forma eficiente e segura [26] [27].

Paralelamente, a arquitetura de microserviços tem ganho uma proeminência considerável [28]. Esta abordagem modular para o desenvolvimento de software permite uma maior escalabilidade e flexibilidade, aspectos cruciais num ambiente empresarial em constante mudança. Os microserviços permitem que as organizações decomponham aplicações complexas em componentes menores e mais gerenciáveis, facilitando a manutenção e a evolução dos sistemas ao longo do tempo.

A computação em nuvem (do inglês, *Cloud Computing*) tem desempenhado um papel fundamental na transformação do mercado de integração de sistemas [29]. As soluções baseadas na nuvem oferecem uma flexibilidade sem precedentes, permitindo que as empresas escalem seus recursos de forma dinâmica e reduzam significativamente os custos operacionais [30]. Esta mudança para a nuvem também está a impulsionar a adoção de plataformas de integração como serviço (iPaaS), que oferecem capacidades de integração flexíveis e escaláveis num modelo baseado na nuvem [29].

A inteligência artificial (IA) e o *machine learning* estão a emergir como componentes críticos nas soluções de integração modernas. Estas tecnologias estão a ser aplicadas para automatizar processos complexos de integração, melhorar a qualidade dos dados e gerar insights preditivos valiosos [31]. A IA está a permitir que as empresas extraiam mais valor dos seus dados, identificando padrões e tendências que podem informar decisões estratégicas [32].

Uma tendência particularmente notável é o crescimento das plataformas *low-code* e *no-code* [33]. Estas soluções estão a democratizar o processo de integração, permitindo que profissionais não técnicos, isto é, pessoas sem o conhecimento ou formação em IT, criem

integrações complexas com pouco ou nenhum conhecimento de programação [34]. Esta abordagem está a acelerar significativamente os ciclos de desenvolvimento e a reduzir a dependência de recursos de IT especializados [34].

Tabela 1 – Receita de tecnologias de desenvolvimento "low-code" (milhões US$)

	2021	**2022**	**2023**	**2024**
Low-Code Application Platforms (LCAP)	6,324	7,968	9,960	12,351
Business Process Automation (BPA)	2,416	2,585	2,761	2,940
Multiexperience Development Platforms (MDXP)	2,081	2,508	2,999	3,563
Robotic Process Automation (RPA)	2,350	2,892	3,401	3,879
Integration Platform as a Service (iPaaS)	4,680	5,668	6,668	7,838
Citizen Automation and Development Platforms (CADP)	554	732	953	1,232
Other Low-Code Development (LCD) Technologies*	92	109	126	146
Total	**18,497**	**22,462**	**26,869**	**31,949**
*Other low-code development technologies include rapid mobile app development (RMAD) tools and rapid application development (RAD) tools. RAD tools are/were on-premises only and desktop-bound applications mainly. Low code is the evolution of RAD to cloud and SaaS models.				
Notes: Gartner defines a no-code application platform as an LCAP that only requires text entry for formulae or simple expressions. The LCAP market, therefore, includes no-code platforms. Furthermore, "no code" is not a sufficient criterion for tasks like citizen development, as many complex tooling configuration tasks are no code but still require specialist skills. Columns may not add to totals shown because of rounding.				

Fonte: Gartner - *Gartner Forecasts Worldwide Low-Code Development Technologies Market to Grow 20% in 2023*

À medida que o volume e a complexidade dos dados continuam a aumentar, a governança de dados e a segurança tornaram-se preocupações primordiais. As empresas estão a investir em soluções que não só facilitam a integração, mas também garantem a integridade, a segurança e a conformidade dos dados. Isto é particularmente crucial num ambiente regulatório cada vez mais rigoroso [34] [35].

Em conclusão, o mercado de integração de sistemas está a evoluir rapidamente, impulsionado por avanços tecnológicos e mudanças nas necessidades empresariais. As soluções que oferecem flexibilidade, escalabilidade e facilidade de uso estão a ganhar terreno, enquanto as empresas procuram formas mais eficientes e eficazes de integrar os seus sistemas e dados. Esta evolução promete transformar fundamentalmente a forma como as organizações abordam a integração de sistemas, permitindo-lhes adaptar-se mais rapidamente às mudanças do mercado e extrair maior valor dos seus investimentos tecnológicos.

3 Soluções existentes

Neste ponto, analisam-se os principais produtos disponíveis no cenário de integração de sistemas, descrevendo as suas características e funcionalidades.

Plataformas de Integração Comerciais

MuleSoft Anypoint Platform

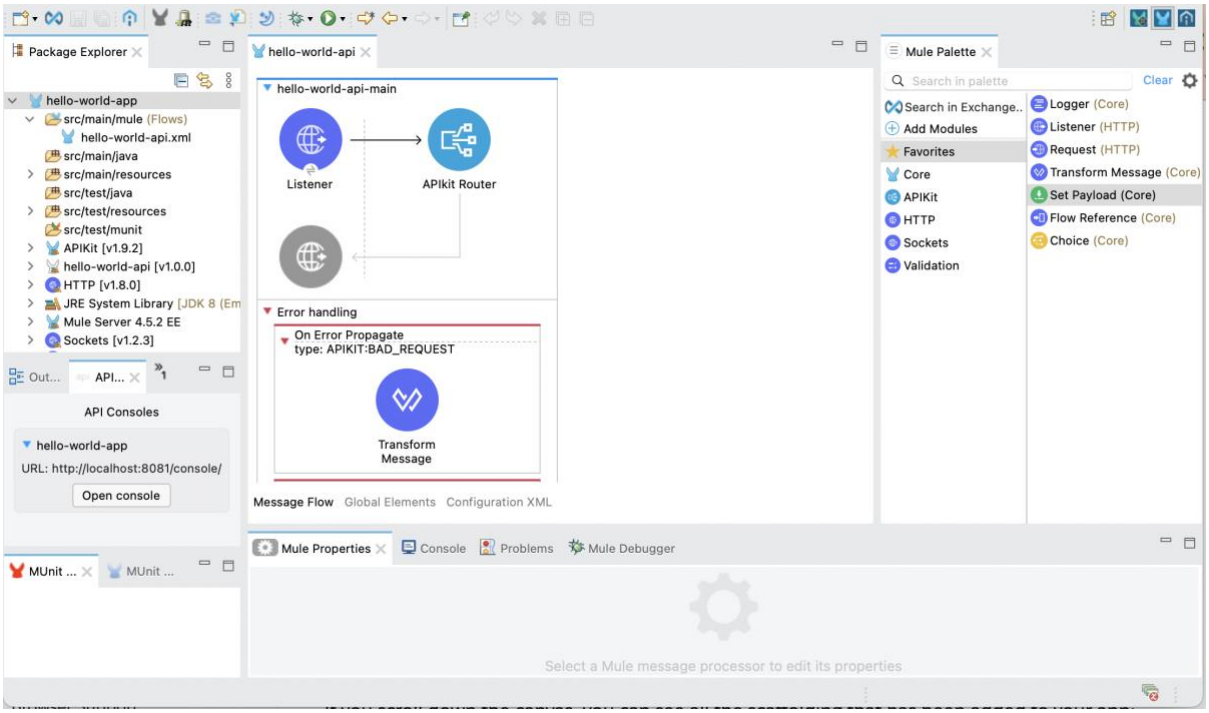
O MuleSoft Anypoint Platform é uma plataforma de integração como serviço (iPaaS), reconhecida como líder do segmento pela Gartner nos relatórios: "Magic Quadrant™ for iPaaS" e Magic Quadrant™ for API Management" [36] [37].

Características principais:

- Design, implementação e gestão de APIs
- Integração de dados e orquestração de processos
- Suporte a ambientes híbridos (on-premise e nuvem)

A plataforma oferece capacidades robustas de transformação de dados, essenciais para a geração de relatórios precisos. Utilizando o DataWeave, a linguagem de transformação proprietária do MuleSoft, os programadores podem manipular dados complexos e gerar saídas em vários formatos [38]. A arquitetura baseada em microserviços do Anypoint Platform permite uma escalabilidade eficiente e facilita a implementação de estratégias de integração contínua e entrega contínua (CI/CD).

Figura 5 – Mulesoft Anypoint Studio: IDE para assitir o processo de integração



Fonte: Mulesoft

SAP Integration Suite

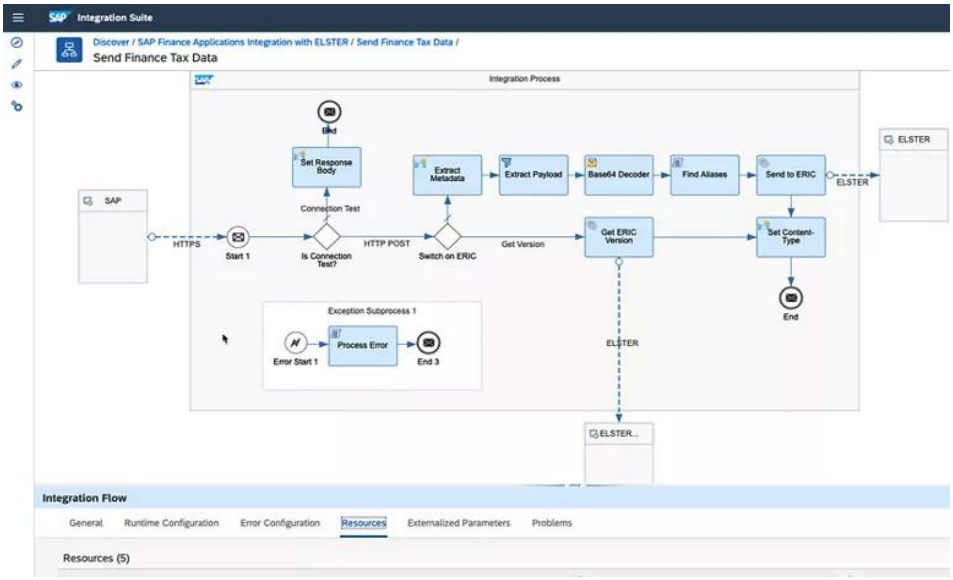
O SAP Integration Suite é uma plataforma de integração como serviço (iPaaS) projetada para facilitar a conexão de processos, serviços, aplicações e dados em ambientes heterogéneos.

Características principais:

- Integração na nuvem e gestão de APIs
- Conectores pré-construídos para sistemas SAP e não-SAP
- Conselheiro de integração (*Integration Advisor*) baseado em machine learning

O SAP Integration Suite oferece mais de 3400 conteúdos de integração pré-construídos, acelerando significativamente o processo de implementação [39]. A plataforma oferece funcionalidades avançadas na gestão de APIs, permitindo que as organizações exponham, publiquem e monetizem os seus dados e processos como APIs. Isso facilita a criação de ecossistemas digitais e novas fontes de receita [40]. Um fator diferenciador do SAP Integration Suite é o seu *Integration Advisor*, uma ferramenta baseada em *machine learning* que sugere mapeamentos de integração com base em padrões históricos e melhores práticas da indústria [41]. Isto pode reduzir significativamente o tempo e o esforço necessários para configurar novas integrações [41].

Figura 6 – SAP Integration Suite: Ferramenta de modelação de integrações



Fonte: SAP

Soluções Open-Source

Zato

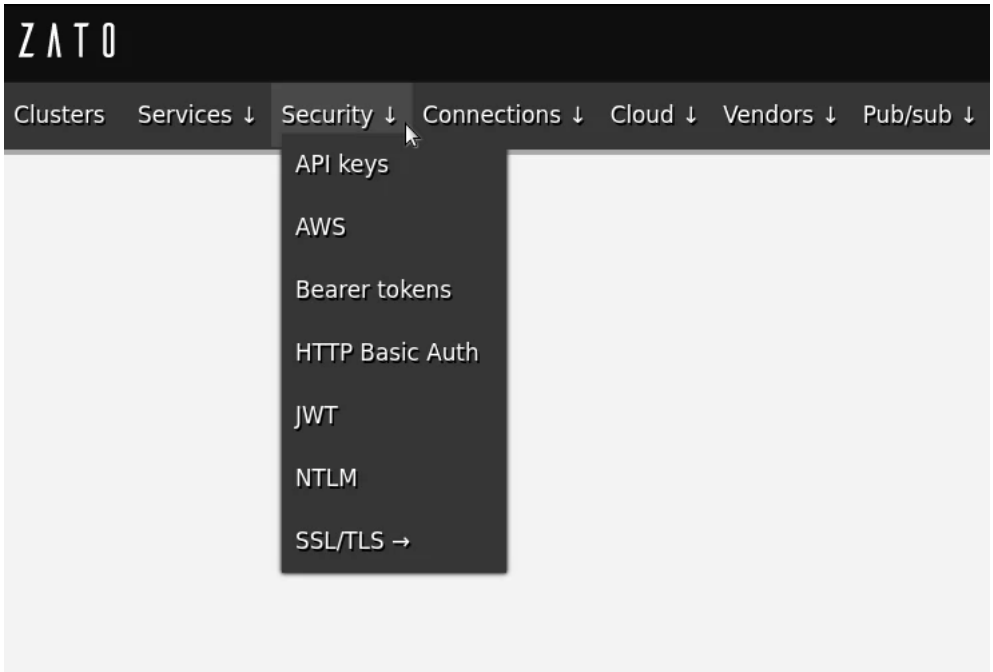
Zato é uma plataforma de integração open-source baseada em Python, focada em APIs REST, mensagens e serviços na nuvem.

Características principais:

- Suporte a múltiplos protocolos e formatos de dados
- Balanceamento de carga e clustering nativos
- Hot-deployment para atualizações sem tempo de inatividade

O Zato utiliza uma arquitetura baseada em serviços, permitindo que os programadores criem, implementem e façam a gestão de serviços de integração de forma modular [42]. A plataforma oferece recursos avançados de segurança, incluindo autenticação e autorização granular, criptografia SSL/TLS e integração com provedores de identidade externos [42]. O Zato também se destaca pela sua capacidade de *hot-deployment*, permitindo atualizações e modificações de serviços sem interromper o sistema em execução [43].

Figura 7 – Zato: Exemplo da interface de utilizador #1



Fonte: Zato

Figura 8 – Zato: Exemplo da interface de utilizador #2

Fonte: Zato

Tabela 2 – Comparação entre os produtos MuleSoft Anypoint Platform, SAP Integration Suite e Zato

Característica	**MuleSoft Anypoint Platform**	**SAP Integration Suite**	**Zato**
Modelo de Negócio	Comercial	Comercial	Open-Source
Foco Principal	Integração e APIs	Integração Empresarial	Middleware e Backend

Ambiente de Desenvolvimento	Anypoint Studio (visual)	Design Center	Baseado em Python
Suporte a Protocolos	Amplo	Amplo	Amplo
Escalabilidade	Alta	Alta	Moderada

Soluções de Automação de E-mails

Além das plataformas de integração, é importante mencionar soluções específicas para automação de e-mails, como SendGrid e Mailchimp uma vez que estas soluções partilham o âmbito do projeto, no que toca ao envio periódico de e-mails para listas de utilizadores.

SendGrid

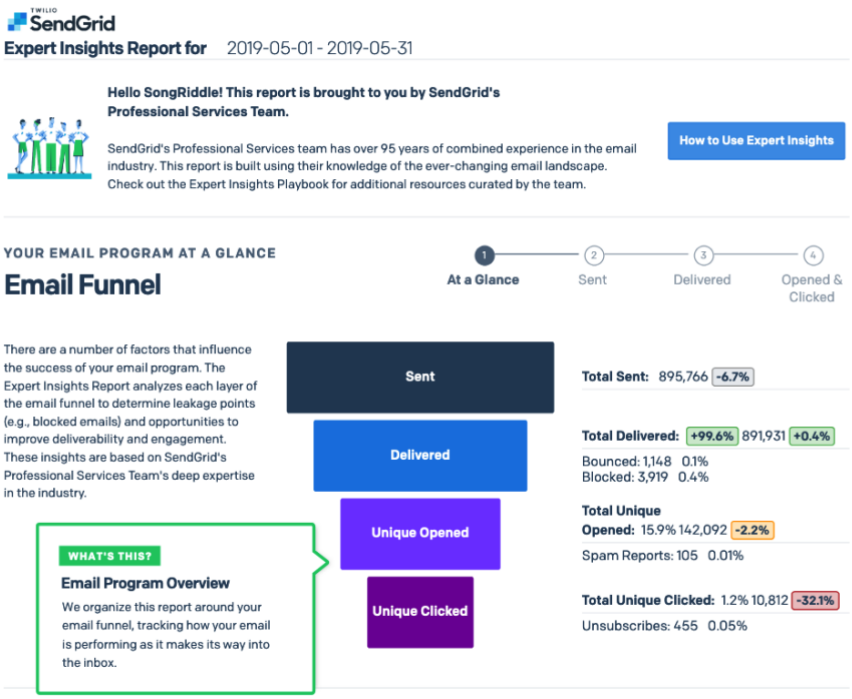
SendGrid é uma plataforma de entrega de e-mail baseada na nuvem, especializada em e-mails transacionais e de marketing.

Características principais:

- API robusta para integração com sistemas existentes
- Ferramentas avançadas de análise e relatórios
- Recursos de personalização e segmentação de e-mails

O SendGrid oferece uma API RESTful abrangente que permite a integração com sistemas existentes para automação de envio de e-mails e geração de relatórios [44]. Isso facilita a incorporação de funcionalidades de e-mail em aplicações e fluxos de trabalho existentes. A plataforma fornece análises detalhadas em tempo real, incluindo taxas de abertura, cliques e envolvimento [45]. Essas métricas podem ser utilizadas para gerar relatórios automatizados sobre o desempenho das campanhas de e-mail, auxiliando na tomada de decisões baseadas em dados. SendGrid também oferece recursos avançados de personalização e segmentação, permitindo que as organizações adaptem o conteúdo dos e-mails com base em dados do destinatário e comportamentos anteriores.

Figura 9 – SendGrid: Relatório de alcance de serviços de e-mail



Fonte: SendGrid

Mailchimp

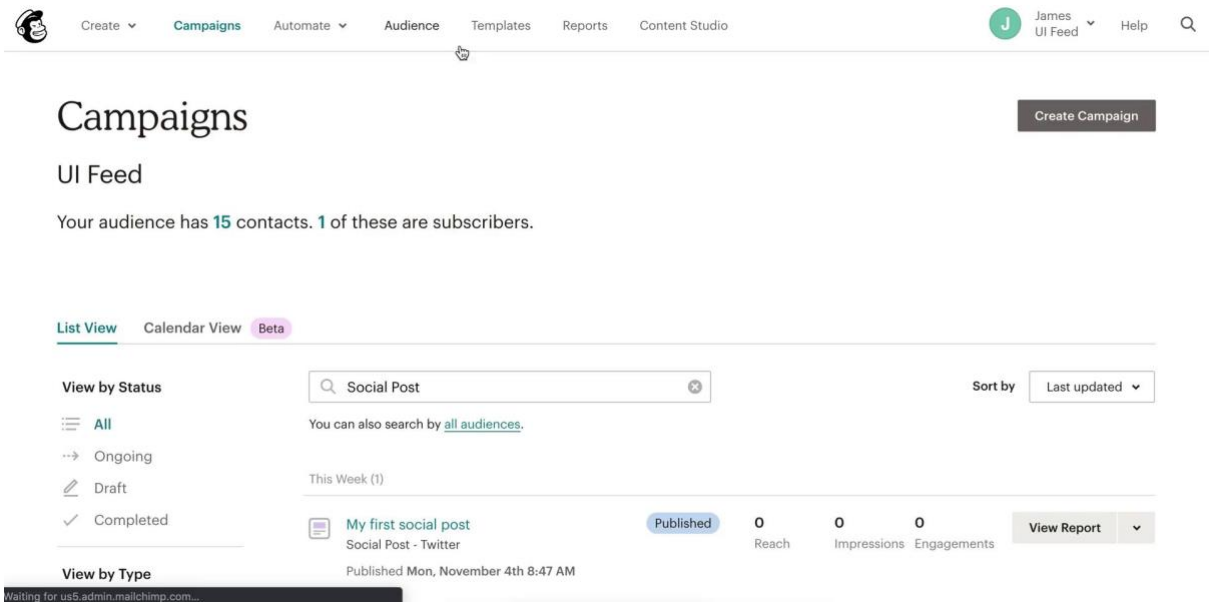
Mailchimp é uma plataforma de automação de marketing all-in-one que vai além do envio de e-mails, atualmente oferece toda uma suíte de produtos para dinamizar os processos de marketing digital, deste a integração com várias plataformas, como também a criação e gestão de formulários e anúncios nas mais variadas plataformas online [46].

Caracteristicas principais:

- Automação de marketing multicanal
- Segmentação avançada de audiência
- Ferramentas de criação de conteúdo e landing pages

A plataforma destaca-se pelas suas capacidades avançadas de segmentação, permitindo que as organizações dividam suas listas de contatos com base em uma variedade de critérios, incluindo comportamento, demografia e histórico de interações. Isso permite a criação de comunicações altamente direcionadas e relevantes. O Mailchimp também oferece ferramentas robustas para criação de conteúdo, incluindo um editor visual para a criação de e-mails e páginas iniciais [47]. Isso permite que mesmo usuários sem experiência técnica criem comunicações visualmente atraentes e eficazes. Para suportar estas atividades, o Mailchimp oferece várias APIs onde é possível automatizar várias operações, nomeadamente o envio peródico de e-mails [48].

Figura 10 – Mailchimp: Ecrã de gestão de campanhas em curso



Fonte: Mailchimp

Face aos produtos enunciados notam-se as seguintes diferenças entre os modelos de negócio deste tipo de soluções, sendo possível estabelecer a seguinte relação:

Tabela 3 - Relação entre produtos comerciais e open-source

	Plataformas Comerciais	**Soluções Open-Source**
Funcionalidades	Ampla gama de funcionalidades, suporte a diversos protocolos e formatos	Flexibilidade e personalização, mas podem exigir mais configuração
Custo	Geralmente mais caro, com modelos de licenciamento ou serviços de subscrição	Gratuito, mas pode haver custos com suporte e serviços adicionais
Suporte	Suporte técnico especializado	Comunidade ativa, mas suporte limitado
Escalabilidade	Alta escalabilidade para grandes volumes de dados	Escalabilidade pode ser limitada em algumas soluções

4 Lacunas

Face às características deste tipo de soluções, identificaram-se os pontos onde estas tendem a falhar, ou a agir de forma pouco eficaz.

Integração de sistemas real-time

A integração de sistemas que exigem baixa latência e alta disponibilidade apresenta dificuldades significativas [49]. As plataformas de integração atuam como intermediárias na troca de informações, inevitavelmente introduzindo algum atraso no processo. Isto pode ser particularmente problemático em sectores como finanças ou saúde, onde é crítica a tomada rápida de decisões [50].

Integração de dados não estruturados

Há uma carência de ferramentas eficientes para integrar dados em formatos como texto, imagem e áudio [51]. A natureza intrínseca desses dados torna difícil a sua manipulação de forma genérica [52]. Tecnologias emergentes como processamento de linguagem natural e visão computacional estão sendo desenvolvidas para abordar esse desafio [53].

Gestão de complexidade

À medida que o número de integrações aumenta, a carga sobre estes sistemas cresce exponencialmente. A heterogeneidade dos dados, protocolos de comunicação e sistemas pode levar a dificuldades na gestão de recursos, manutenção do sistema e proteção de dados [54]. Isto é principalmente observável em ambientes empresariais complexos com múltiplos sistemas legados.

Soluções personalizadas

Estes sistemas muitas vezes interligam plataformas com arquiteturas diferentes, onde as configurações e os tipos de dados esperados podem variar significativamente. Isto leva a que, para integrar um mesmo sistema com outros, muitas vezes é necessário aplicar várias transformações, baseadas nos requisitos específicos do destinatário dos dados [55]. Essa necessidade de personalização pode aumentar significativamente o tempo e o custo de implementação.

CAPÍTULO III - METODOLOGIA

1 Metodologia de desenvolvimento

1.1 Tipo de projeto

O projeto desenvolvido configura-se como um sistema de *backend* com uma interface de administração (Django Admin), para a gestão do modelo de dados, configuração de serviços, destinatários e utilizadores.

A plataforma foi desenvolvida tendo por base a *framework* Django, e as suas funções básicas resumem-se ao envio e resposta a chamadas HTTP, com o objetivo de permitir a comunicação entre sistemas e a entrega de relatórios a utilizadores.

1.2 Definição de Requisitos e Metodologia de desenvolvimento

Inicialmente foram definidos uma série de requisitos que a plataforma deveria entender, sendo os principais: a criação de ficheiros PDF e a sua entrega através de vários processos, como por exemplo: e-mail, HTTP e FTP; a criação de processos automáticos para a recolha e entrega atualizada destes relatórios e a possibilidade de configurar, caso a caso, qual a frequência com que estes seriam entregues a cada utilizador. No decorrer do desenvolvimento, surgiu a necessidade de adaptar a solução já existente aos requisitos específicos da plataforma PAE, o que levou ao desenvolvimento de funcionalidades específicas e ao adaptar da lógica pré-existente para satisfazer as necessidades dos serviços disponibilizados. Um bom exemplo desses requisitos, foi a necessidade de permitir a utilizadores da plataforma PAE a configuração da periodicidade com que serão notificados pela plataforma, bem como a consulta de relatórios recebidos no passado e que possam não estar mais disponíveis na sua caixa de e-mail.

Para a elaboração deste projeto, recorreu-se à metodologia *Waterfall* para o desenvolvimento da implementação inicial. Isto foi motivado pela forma como se abordou o projeto, definindo inicialmente quais os requisitos principais a atender e a forma como a implementação deveria ser estruturada.

Após a implementação das funcionalidades base a implementação foi complementada gradualmente por técnicas de desenvolvimento orientado a testes (TDD), quer numa perspetiva de melhorias como de implementação de novas funcionalidades. Isto com o objetivo de garantir que estas modificações não comprometiam a implementação já existente e que se estabelecia um certo grau de confiança que as novas funcionalidades atenderiam aos requisitos, isto

permitiu também que estes testes tomassem lugar sem a necessidade de invocar serviços sucessivamente, o que de alguma forma se pode considerar um desperdício de recursos computacionais.

1.3 Arquitetura

A arquitetura da aplicação baseia-se no padrão MVT (Model-View-Template) do Django, onde:

- **Model**

Representa os dados da aplicação, são as classes que, no fundo, se traduzem no modelo de dados que se procurou implementar. Envolve toda a configuração de serviços, tarefas agendadas, informação de autenticação e dados de utilizador ou destinatários.

- **View**

Responsável pela lógica da aplicação, é o componente que trata as chamadas à aplicação pelos vários URLs disponibilizados, não só tratam de atender às chamadas e tratamento das respostas dos serviços configurados para processamento "*on demand*", como também são o componente que gere a componente visual para os utilizadores da plataforma PAE poderem administrar quais as subscrições a que estão atribuídos bem como a consulta de relatórios antigos. Específico da plataforma PAE, estes utilizadores são autenticados com recurso a um *token* que é gerado toda a vez que um e-mail é enviado.

- **Template**

Define a estrutura visual das saídas, utilizada para a composição de e-mails, PDFs e demais páginas para a consulta do estado das subscrições e dados enviados.

O desenvolvimento teve por base três pilares principais, de acordo com a sua responsabilidade para com os processos definidos, vulgo módulos: Scheduler, Runner e Dispatcher.

Estes podem ser detalhados da seguinte maneira:

- **Scheduler:** Configura serviços, gere tarefas agendadas e retorna respostas imediatas ao utilizador.

- **Runner:** Executa tarefas individuais, interagindo com sistemas externos e gerando outputs (PDFs, e-mails). É utilizado pelo Celery para processamento agendado e trata a criação de ficheiros.
- **Dispatcher:** Envia os dados gerados pelo Runner para os destinos configurados.

Além disso, para o tratamento específico de dados do PAE e a distinção entre utilizadores da plataforma e destinatários do PAE, foi criado um módulo auxiliar do mesmo nome, estruturado como uma biblioteca para facilitar a reutilização de código.

2 Ferramentas e Tecnologias

Pegando nos pontos evidenciados na introdução, abaixo estão detalhadas as tecnologias utilizadas e a justificação por detrás dessas escolhas.

Linguagem de programação: O Python foi a linguagem escolhida, sendo uma linguagem que tem uma sintaxe simples e acessível. O facto de ser das linguagens mais populares do mundo do desenvolvimento de software tornou fácil esta opção, uma vez que tem uma grande comunidade a dar-lhe suporte e existe uma grande quantidade de bibliotecas, o que permitiu o acesso a recursos que facilitaram a implementação de várias funcionalidades [56].

Framework: O Django foi escolhido de entre as opções disponíveis para trabalhar com Python, particularmente pelo foco no desenvolvimento de aplicações Web e a aplicação do modelo MVT (Model-View-Template). Essas características, aliadas ao sistema de autenticação já existente, à interface administrativa (Django Admin) e ao mapeamento ORM (Object-Relational-Mapping), permitiu que os esforços de desenvolvimento se focassem principalmente na entrega de funcionalidades, abstraindo temas como a gestão de base de dados e a criação de um sistema de gestão de utilizadores. Isto, aliado à documentação extensa sobre a plataforma e os processos de migração de base de dados, transmitiu alguma confiança caso se tornasse necessária a migração da solução de base de dados inicial.

Base de dados: Utilizou-se o SQLite por ser prático e ágil, uma vez que não requer a existência de um servidor de base de dados dedicado, assentando num esquema de um só ficheiro. Isto tem algumas desvantagens no que toca à sua performance e concorrência, mas para o propósito inicial deste projeto, serviu as necessidades previstas.

Gestor de tarefas: A biblioteca Celery oferece uma fila de tarefas, ou seja, através do Celery é possível o agendamento de tarefas periódicas, que correm sobre os seus próprios processos independentes. Isto é relevante, pois significa que são instâncias separadas da própria aplicação, o que neste caso em particular seria desejável para uma boa alocação dos recursos do sistema. Isto foi suficiente para que se tenha optado por esta biblioteca para suportar as tarefas em background realizadas por esta plataforma. No entanto, para o Celery poder comunicar com a plataforma, é obrigatório a existência de um *message broker*, isto é definido pela IBM como: "um software que possibilita que aplicativos, sistemas e serviços se comuniquem e troquem informações [57], na documentação do Celery existem duas recomendações, RabbitMQ e Redis, da qual se optou pela segunda [58].

Broker: O Redis foi das duas opções mencionadas anteriormente, a escolhida. Sendo obrigatória a implementação de um *middleware* para permitir a troca de mensagens entre a aplicação e o Celery, procurou-se ver quais as mais-valias que uma opção trazia face à outra, uma vez que ambas poderiam servir de suporte à aplicação. No final, a decisão prendeu-se por ser a solução mais adequada para a transmissão de mensagens pequenas (como é o caso de uso nesta aplicação), capaz de suportar comunicação sob o modelo *publisher-subscriber* e ser a opção mais eficiente para manter dados em cache, uma vez que se comporta como uma base de dados em memória [59] [60] [61].

Tabela 4 - Resumo das diferenças: RabbitMQ vs. do Redis

	RabbitMQ	Redis
Entrega de mensagens	Entrega de mensagens garantida. Suporta para lógica complexa.	Não garante a entrega de mensagens. Requer conexão ativa dos assinantes.
Tamanho da mensagem	Tamanho da mensagem limitado a 128 MB. Pode lidar com mensagens grandes.	Não há limite de mensagens, mas a performance diminui para mensagens grandes (maiores que 1 MB).
Persistência de mensagens	Oferece suporte a mensagens persistentes e transitórias. Grava mensagens persistentes no disco.	Por padrão, não oferece suporte a mensagens persistentes.
Criptografia de mensagens	Oferece suporte à criptografia SSL.	Criptografia SSL disponível no Redis 6.0 e versões superiores.
Velocidade	Até dezenas de milhares de mensagens por segundo.	Até milhões de mensagens por segundo.
Disponibilidade	Cria vários nós ponto a ponto em um cluster.	Usa o modelo de líder/seguidor no agrupamento em cluster.

Fonte: Amazon - Qual é a diferença entre o RabbitMQ e o Redis?

Geração de PDFs: Para gerar relatórios em formato PDF, o projeto utiliza duas bibliotecas complementares. ReportLab permite criar documentos PDF a partir do zero, com todo o

controlo sobre o layout e formatação. Já o PyPDF2 é utilizado para manipular PDFs existentes, permitindo a edição, extração de dados ou combinação de arquivos PDF, facilitando a criação de relatórios personalizados, no caso desta aplicação, é utilizado sobretudo como suporte ao sistema de *templates*, permitindo oferecer um PDF com um aspeto personalizado.

Processamento XML: xmltodict é uma biblioteca que permite a conversão de dados em XML em dicionários Python, como o próprio nome indica, isto facilitou a extração e manipulação destes dados, atuando como um analisador sintático (do inglês, *parser*).

APIs: Recorreu-se ao Django REST Framework para a criação de APIs RESTful, de modo a expor os serviços do sistema numa página específica e apenas acessível a utilizadores autenticados. Utilizando essas funcionalidades como base, também se disponibilizou uma interface onde os serviços disponíveis para invocação "por pedido" são listados, o que tornou mais fácil a representação dos serviços disponíveis, esta configuração foi útil para testar ambos os processos de execução, sejam eles a vertente imediata ou periódica.

Testes: unittest - é a biblioteca padrão de testes unitários em Python e foi utilizada para suportar os testes unitários e testes de integração criados no âmbito do projeto.

IDE: Utilizou-se o PyCharm como ferramenta de ambiente de desenvolvimento integrado pois esta integra diretamente com o Django, sendo uma ferramenta de produtividade, esta permitiu uma fácil integração das bibliotecas assinaladas, bem como a sua atualização. Com a integração do terminal, comandos recorrentes, como as chamadas de migração do modelo de dados "*makemigrations"* e *"migrate"* foram mais fáceis de gerir. Serviu também como ferramenta de controlo de versões local, uma vez que o projeto foi criado de raiz, não se considerou pertinente o uso de uma ferramenta como o Git até o projeto adquirir um nível de maturidade que o exija, tal como as funções de desenvolvimento habitual.

Outras: requests, requests-oauthlib, python-dotenv, croniter

requests: Biblioteca essencial para a realização de chamadas HTTP.

requests-oauthlib: Extensão do requests que permite a autenticação através do protocolo OAuth, que é bastante comum no atual panorama de comunicação via APIs.

python-dotenv: Facilita a gestão de variáveis de ambiente, utilizando um ficheiro ".env" para o armazenar destas informações, que de outra forma, implicariam a introdução destas no sistema (envolvendo a reprodução desta configuração em qualquer sistema onde a solução seja instalada) ou no código fonte (inseguro).

croniter: Usado para interpretar expressões *cron*, que serviram de base à configuração dos períodos de atualização pelos utilizadores, optou-se pela utilização deste tipo de expressões, uma vez que segue os mesmos moldes que a biblioteca Celery, deixando assim homogéneo, o modo como estes períodos são geridos pela aplicação.

3 Metodologia de Testes

Foram realizados testes unitários e de integração para assegurar a qualidade do software. A biblioteca *unittest* do Python foi utilizada para estes testes.

O TDD foi aplicado em várias funcionalidades. Neste método, primeiro define-se um resultado esperado e, em seguida, a solução é criada e estendida até atender aos requisitos, ou seja, passar o teste, nos casos em que esta não obtém o resultado esperado, é apresentada ao programador uma mensagem de erro.

Casos relevantes incluíram a geração de PDFs e *templates*, a criação e gestão de ficheiros no sistema, a comunicação via FTP e o mapeamento entre dados recebidos de um serviço e a sua representação em formatos de e-mail e PDF.

3.1 Testes Realizados

Testes Unitários: Verificaram o funcionamento individual de componentes da plataforma.

Testes de Integração: Garantiram que as diferentes funcionalidades mantinham seu funcionamento e não introduziam comportamentos imprevistos.

Exemplo: Testes para verificar a geração de PDFs com diferentes formatos de input e a entrega de e-mails para diversos destinatários.

Testes ponta a ponta (E2E): Revisão geral da implementação e das funcionalidades existentes, com o objetivo de certificar que toda a aplicação funciona conforme esperado.

3.2 Ferramentas de Teste

unittest: Biblioteca do Python para escrever testes unitários específicos para componentes da solução, como geração de PDFs e mapeamento de dados.

Testes *Ad-Hoc*: Realizados para verificar o comportamento da plataforma em diversas situações.

4 Considerações Éticas

Cada vez que um software interage com informação que nasce de outras entidades surgem questões éticas que devem ser consideradas. Estas preocupações existem, pois, é possível, ainda que inadvertidamente, que um software exponha informação ou permita comportamentos de forma não planeada, o que pode incorrer em dano para quaisquer intervenientes.

No caso desta aplicação, são guardados dados de utilizador que podem permitir a sua identificação, dados de máquinas acessíveis pela plataforma e credenciais sensíveis que não devem ser acessíveis a utilizadores que não façam parte do conjunto de utilizadores com permissões relevantes.

4.1 Impactos

Privacidade:

A aplicação pede ao utilizador a introdução de um nome e apelido na criação de um novo utilizador, esta informação é considerada necessária, por esse motivo, essa informação só deve estar disponível ao utilizador administrador.

Os dados retornados pelos serviços são armazenados no sistema de ficheiros associado à plataforma, estes são apenas passíveis de ser obtidos pela aplicação e os respetivos destinatários, estes estão sujeitos a um período de arquivo, momento em que cessa a sua disponibilidade.

Os dados para a configuração dos serviços podem conter informação sensível, como por exemplo, credenciais de acesso ao serviço, estas devem ser mantidas privadas e ser acessíveis apenas ao utilizador que criou o serviço e ao administrador.

Atualmente, os destinatários que podem ser configurados através da plataforma de administração são apenas visíveis ao utilizador de administração, no entanto, estes podem ser adaptados para permitir a sua manutenção por um utilizador “básico”, obrigando à associação de um utilizador a estes dados.

Segurança dos dados:

A aplicação permite a autenticação de utilizadores à página de administração por um conjunto de credenciais, nome de utilizador e password, é registado um log de atividade nesta página de administração.

À página de disponibilização de serviços temos a possibilidade de aceder também pelo mesmo conjunto de credenciais, ou com recurso a um *token* de autenticação ou sessão.

A implementação atual prevê o armazenamento de dados no sistema de ficheiros, no entanto, estes têm um período de arquivo, atualmente estes ficheiros são movidos para outra localização, mas facilmente se poderia alterar esse comportamento para a eliminação dos mesmos.

Outros impactos:

O uso indevido dos dados recolhidos por esta aplicação pode permitir a que utilizadores tenham acesso a informação privada ou de acesso exclusivo a outros utilizadores. Um exemplo disto seria a informação respondente aos acessos via FTP, que disponibilizados de forma indevida, podem abrir a porta ao uso indevido, permitindo a submissão de ficheiros mal-intencionados num sistema associado a um outro utilizador.

4.2 Mitigação

Para a utilização da plataforma no seu estado atual presume-se o aceitar do uso e tratamento dos dados da plataforma nas condições acima descritas.

Adicionalmente e na perspetiva de desenvolvimentos futuros, sugere-se a remoção do acesso à plataforma de administração pelos utilizadores ditos “base”, oferecendo funcionalidades básicas de visualização, análise, e manutenção das subscrições existentes, à semelhança do que já acontece com os destinatários da plataforma PAE ou através de uma camada de *front-end* dedicada, o que permitiria um melhor controlo da informação disponível a cada utilizador da plataforma.

Reforçar os esquemas de autorização da plataforma seria uma alternativa capaz de oferecer uma maior granularidade no acesso aos dados da plataforma oferecendo assim, maiores garantias de privacidade aos utilizadores.

Relativo ao tratamento dos dados devolvidos pelos serviços configurados, estes podem ser tratados por meio da introdução da informação numa tabela da base de dados a designar, onde, por exemplo, estes possam ser encriptados. Isto dificultaria o acesso à informação mesmo assumindo que o sistema de ficheiros possa ser comprometido.

Os dados de acesso FTP devem estar encriptados e disponíveis apenas ao utilizador administrador do sistema de destino, bem como o administrador da plataforma.

CAPÍTULO IV – ANÁLISE E DESENHO DA SOLUÇÃO

1 Especificações do Problema

A plataforma desenvolvida é um sistema de integração que visa facilitar a comunicação e troca de dados entre diversos sistemas, aplicações e utilizadores. O seu principal objetivo é servir como intermediário eficiente na transferência de dados entre vários clientes.

Com base nos objetivos do projeto, foram estabelecidos três requisitos prioritários para a plataforma. Primeiramente, a solução deve suportar múltiplos formatos de comunicação, com ênfase inicial em XML e JSON, essenciais nas comunicações com APIs REST e SOAP, padrões na indústria das TI. Além disso a solução deve permitir a geração de saídas visuais, incluindo formatos HTML para visualização web e relatórios em PDF para documentação e distribuição.

Outro aspeto importante é a criação de um conector dedicado para a plataforma PAE, visando a integração e elaboração de relatórios dedicados. Essa interface específica permitirá a entrega de relatórios a entidades do Instituto Politécnico de Portalegre (IPP).

Devido às limitações de tempo disponíveis para esta implementação, o foco foi direcionado para as funcionalidades principais relacionadas à integração e automação. Para uma visão mais abrangente dos requisitos discutidos, é possível a consulta do Anexo 1 – "Definição de Requisitos". Neste contexto, a criação de um painel dedicado a cada utilizador para acesso e gestão de dados analíticos não foi incluída nos requisitos desta fase de desenvolvimento. Esta decisão foi tomada considerando as limitações de tempo e recursos, bem como a priorização das funcionalidades de integração. Em vez disso, optou-se por utilizar o painel de administração do Django para consulta e introdução de dados no sistema, uma solução que atende às necessidades imediatas do projeto.

2 Arquitetura de Dados e Modelo Conceptual

Aqui é apresentado o diagrama Entidade-Relacionamento que caracteriza a forma como as entidades previstas se relacionam na base de dados implementada para dar suporte à solução.

Na Figura 11 – Diagrama Entidade-Relacionamento da aplicação possível observar todas as tabelas que foram criadas no âmbito do desenvolvimento da plataforma, sendo que, para uma visão total, que inclui também as tabelas relativas ao Django e à biblioteca Celery, estão disponíveis no Anexo 3 – "Diagrama Entidade-Relacionamento Completo" por uma questão de conveniência.

Figura 11 – Diagrama Entidade-Relacionamento da aplicação

Observando o esquema, é possível constatar quais as entidades mais relevantes nesta implementação. Como seria espectável, é na tabela "Service" que se encontra o maior número de relações. Isto acontece, pois, é aqui que são definidas grande parte das restrições e configurações respeitantes ao Serviço a integrar. Seguindo a estrutura de nomenclatura do Django, vamos olhar para as restantes tabelas do módulo "Scheduler". A tabela

"Authentication" é onde são definidos os dados respeitantes ao processo de autenticação, qual o protocolo e as credenciais a utilizar no processo de comunicação. Já a outra tabela "ServiceAction" surgiu da necessidade de suportar pedidos POST de modo a submeter um conjunto de parâmetros e filtrar os dados que se pretende receber, isto foi particularmente útil na comunicação com o PAE, uma vez que foi necessária, com base no processo de autenticação implementado, a adição do *token* nos pedidos enviados, bem como os detalhes das funções chamadas.

Tabela 5 – Exemplo do corpo de uma chamada a um serviço do PAE


```
{
  "BACOSESS": "DOwPCGNoatMNXfr-qlgVzG_JxYRcMXwuJ-ep-sUC",
  "data": {
    "checkOnlyMode": false,
    "serviceArguments": [
      {
        "name": "importYear",
        "position": 0,
        "type": "String",
        "value": "202021"
      }
    ],
    "methodName": "pedgogicReport"
  },
  "serviceJson": "WSPeriodicApiFacade"
}
```


Para o módulo Runner, as tabelas "RunnerResponse" e "Attachment" servem apenas para guardar as respostas recebidas pelos serviços, bem como os dados transformados, com o objetivo de poder consultar quaisquer erros de comunicação, no caso da primeira e servir ficheiros antigos aos utilizadores, no caso da segunda.

A tabela "Destination" diz respeito ao módulo "Dispatcher" e neste caso, garante que a informação respeitante aos destinos, no formato de URLs ou endereços de e-mail é mantida, bem como o intervalo de atualização. A implementação específica de suporte ao PAE segue uma estrutura muito semelhante, mas optou-se pela separação de responsabilidades, uma vez que a abordagem inicial e a integração com o PAE operam em sentidos contrários, isto é, ao passo que os serviços ditos "regulares" obrigam à configuração de destinatários, no caso do PAE é esta plataforma que dita quem são os recetores da informação.

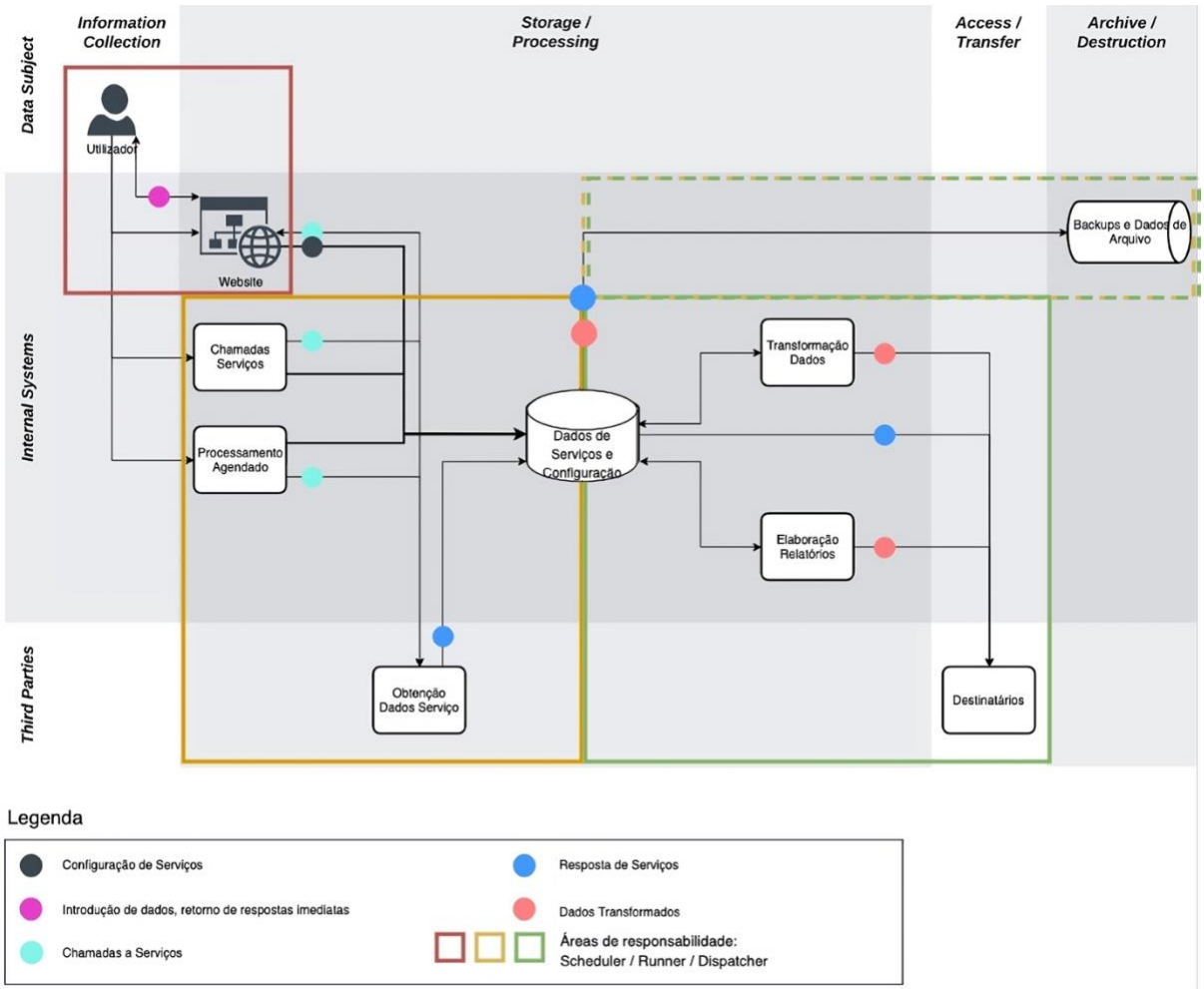
Continuando, as restantes tabelas do módulo PAE são responsáveis por alojar os e-mails e anexos enviados a cada utilizador, pela mesma razão enunciada anteriormente, optou-se pela criação de tabelas independentes para a gestão destes casos. Já a tabela "PAEUserAccessToken" é responsável pelo armazenamento dos *tokens* gerados para cada destinatário dos e-mails que resultam das chamadas deste serviço. Finalmente, a tabela "PAEToken" apenas regista quais os *tokens* de comunicação com o PAE devolvidos e a sua data de expiração, no sentido de poupar recursos.

A restante tabela de "Log", é utilizada para a recolha de informação variada de processamento, permitindo acompanhar o funcionamento da plataforma em tempo de execução.

3 Fluxo de Dados

O fluxo de dados da plataforma desenvolvida é estruturado em três componentes principais: Scheduler, Runner e Dispatcher. O Scheduler é responsável pelo agendamento de tarefas, iniciando a recolha de dados conforme as configurações definidas. Após essa etapa, o Runner processa os dados recebidos, realizando as transformações necessárias e armazenando temporariamente as informações para uso posterior. Em seguida, o Dispatcher recupera os dados processados pelo Runner, aplica transformações adicionais se necessário e gera relatórios nos formatos desejados, como JSON, XML, CSV, e PDF.

Figura 12 – Fluxo de dados da plataforma



Estes relatórios são depois distribuidos pelos destinatários configurados, utilizando métodos como envio por e-mail ou transferência via FTP. A plataforma também utiliza Celery e Redis para a realização de atualizações periódicas. Para o caso específico da plataforma PAE, o fluxo de dados inclui etapas adicionais que envolvem a recolha de informações através de uma API

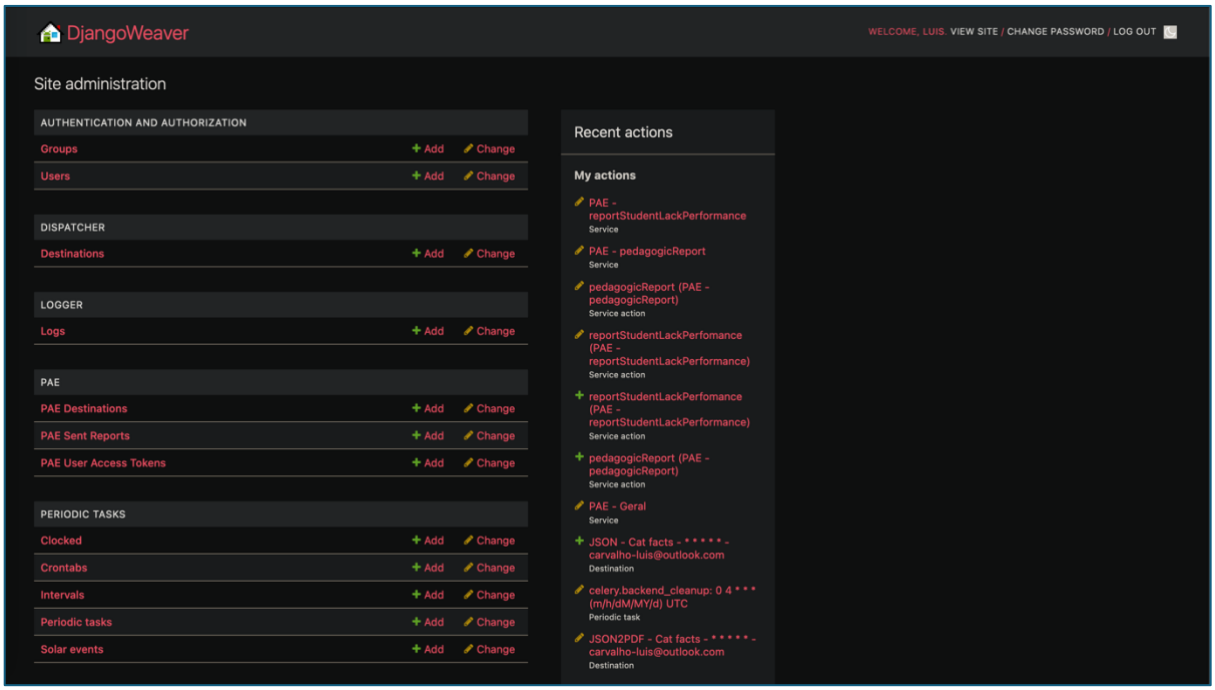
dedicada, com uma estrutura JSON rígida, o que permite a entrega de relatórios personalisados e adaptados às necessidades da plataforma.

4 Definição de Interfaces

Sendo uma aplicação em tudo similar às aplicações de contexto empresarial referenciadas, esta plataforma não foi desenhada com um processo de registo em conta, partindo do pressuposto que o utilizador Administrador seria o único responsável pela adição e criação de utilizadores na plataforma.

A estes utilizadores é assim concedido o acesso a duas áreas reservadas, a área administrativa e a área de consulta de APIs.

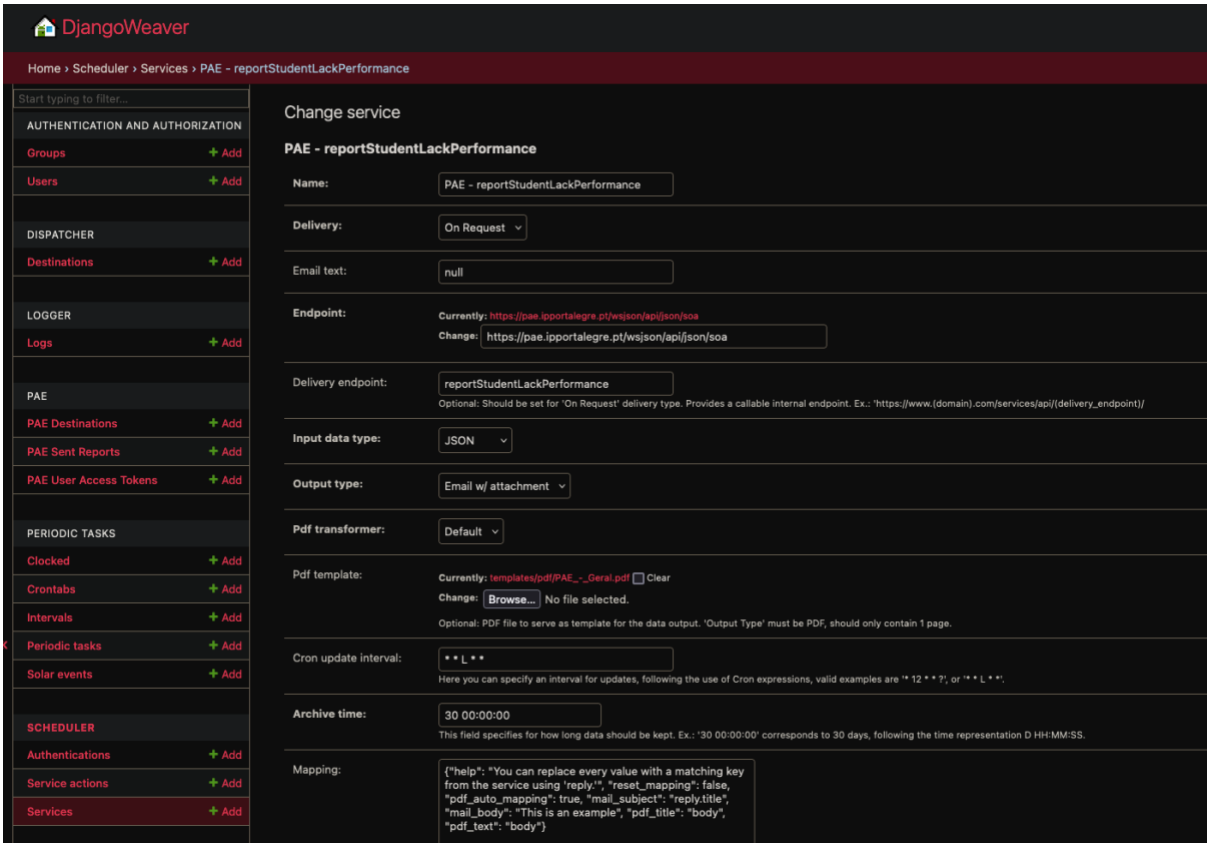
Figura 13 – Àrea de administração: página inicial



Aqui é possível a consulta de todas as entidades registadas na plataforma, bem como a possibilidade de realizar operações CRUD sobre todas elas, caso o utilizador reúna o conjunto de autorizações necessário.

É apresentada aqui a vista de manutenção de uma das interfaces disponíveis:

Figura 14 – Àrea de consulta de APIs: alteração de serviço



Nesta página, é possível navegar pelos links oferecidos e perceber quais os dados presentes nos sistema, como podem ser chamados estes serviços e também, consultar quais destes serviços estão disponíveis para ser chamados diretamente:

Figura 15 – Àrea de consulta de APIs: página inicial

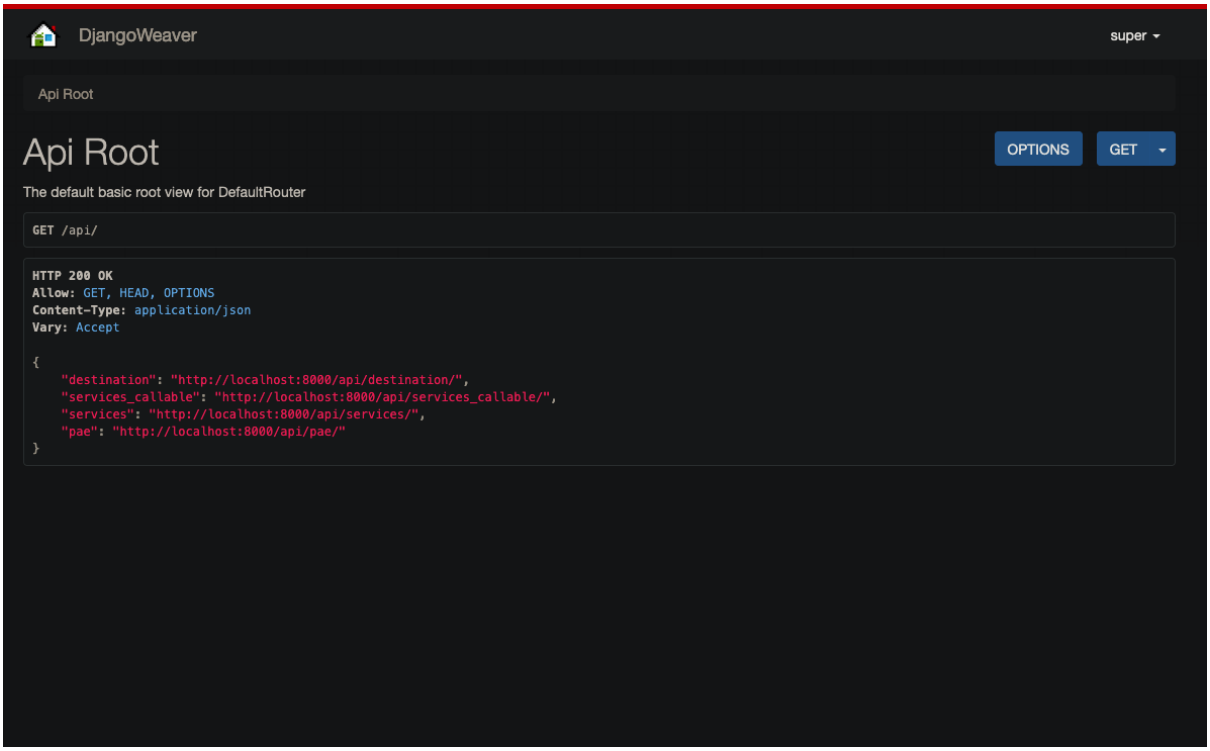
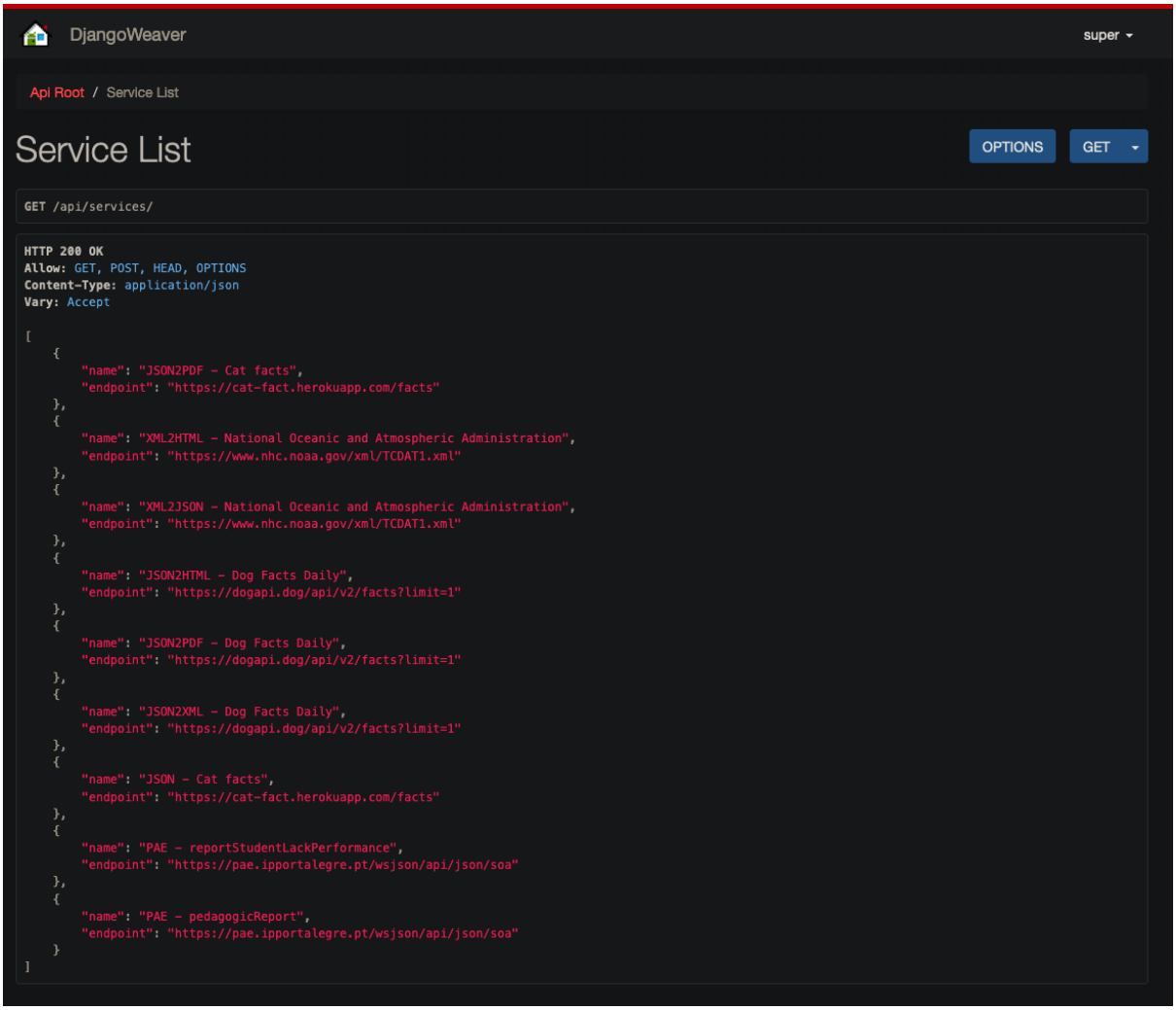


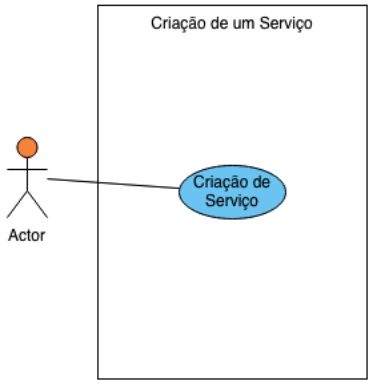
Figura 16 – Àrea de consulta de APIs: serviços disponíveis



5 Casos de Uso

Caso de Uso 1 – Criação de um serviço

Figura 17 – Caso de Uso: criação de um Serviço



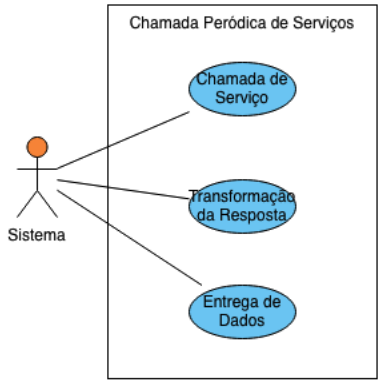
Passos:

1. O administrador acede à área de administração do Django.

2. Navega até à secção "Serviços".
3. Clica em "Adicionar Serviço".
4. Preenche os campos necessários:
    - Nome do serviço
    - URL da API
    - Método HTTP
    - Parâmetros necessários
5. Configura opções específicas:
    - Tipo de autenticação
    - Transformação de dados
    - Formato de saída
6. Define os destinatários do serviço.
7. Seleciona o tipo de serviço (genérico ou PAE).
8. Guarda as configurações do novo serviço.

Caso de Uso 2 – Processo Automático: Chamadas periódicas de serviços

Figura 18 – Caso de Uso: Chamada de serviço periódica



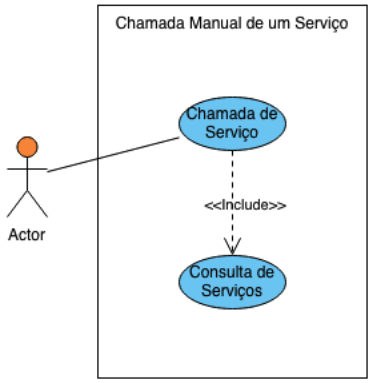
Passos:

1. O sistema inicia automaticamente as tarefas periódicas para o processamento de serviços.
2. O sistema executa a chamada ao serviço, consumindo os dados da fonte especificada.
3. Os dados são processados conforme as regras de transformação definidas.
4. O sistema gera um relatório ou notificação com os resultados da execução.
5. O relatório é distribuído automaticamente para os destinatários configurados (por exemplo, via e-mail ou outro método especificado).

6. O sistema regista a execução da chamada.

Caso de Uso 3 – Processo Manual: Chamadas de serviços

Figura 19 – Caso de Uso: Chamada de serviço manual



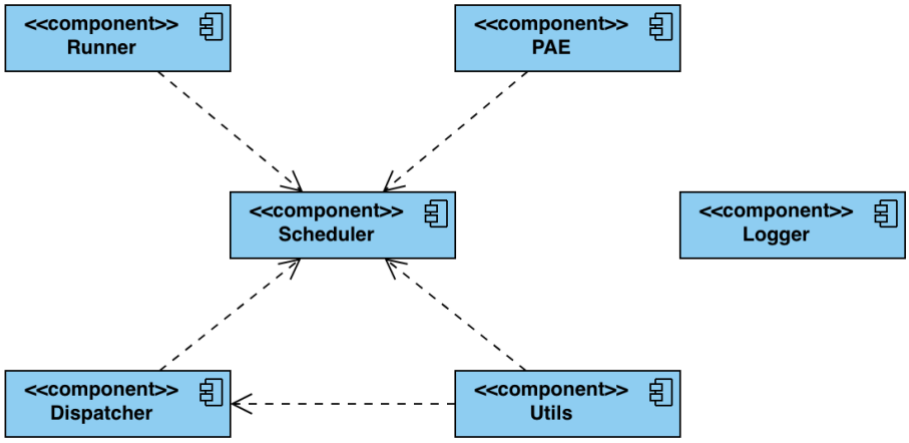
Passos:

1. O utilizador acede à interface de utilizador da plataforma.
2. Navega até à secção "Serviços Disponíveis".
3. Localiza o serviço desejado na lista.
4. Clica no link do serviço disponibilizado.
5. Aguarda a conclusão da chamada do serviço.
6. Visualiza os resultados ou recebe uma notificação de conclusão.

6 Componentes e Separação de Responsabilidades

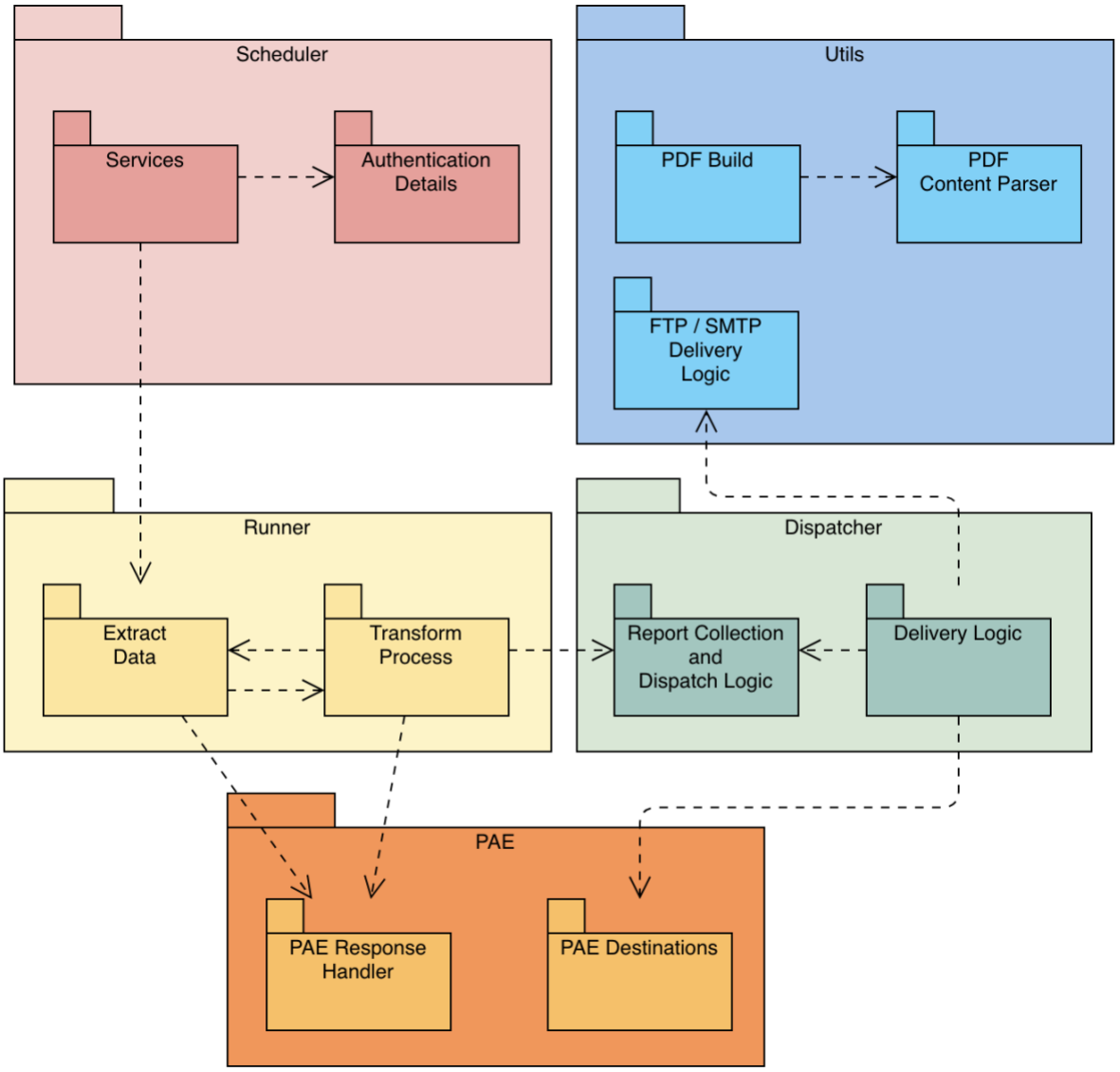
O projeto está separado em seis módulos principais, de acordo com as suas responsabilidades. No entanto, apenas estes três: “Scheduler”, “Runner” e “Dispatcher” são responsáveis pelas principais responsabilidades da plataforma. As restantes, “PAE”, “Logger” e “Utils” seguem propósitos particulares, respetivamente, os objetos de suporte dedicado à plataforma PAE, logs variados da operação do sistema e por último, vários *scripts* para processos comuns e reutilizáveis.

Figura 20 – Componentes da plataforma



Seguindo a mesma ordem lógica, será apresentada uma visão geral sobre o modo como o projeto está organizado e estruturado, ressalvando as responsabilidades de cada módulo e a forma como estes interagem de forma a oferecer as funcionalidades que compõem esta plataforma.

Figura 21 – Estrutura de pactoes do projeto



Como é possível constatar pelos dados da *Figura **21***, o projeto está definido por um conjunto de pacotes, a cada pacote remete uma responsabilidade ou um módulo diferente, que serão abordados em seguida.

O ficheiro ".env" está associado ao uso da biblioteca "python-dotenv", cujo propósito é o de definir variáveis de ambiente com base nos pares chave-valor aqui mantidos.

Figura 22 – Exemplo de um ficheiro ".env"

```
.env
#!/usr/bin/env bash
export DEBUG=1
export TEST_MODE='FALSE'
export DOMAIN="http://localhost:8000/"
export SECRET_KEY='[redacted]'
export SENDER_EMAIL='[redacted]@gmail.com'
export FALLBACK_EMAIL='[redacted]@outlook.com'
export EMAIL_PASSWORD='[redacted]'
export ALLOWED_HOSTS=* localhost 127.0.0.1 [::1]
export INPUT_PATH ='./data/output/'
export OUTPUT_PATH ='./data/archive/'
export DJANGO_SETTINGS_MODULE='djangoProject.settings'
#--> microsoft outlook credentials
export CLIENT_ID='[redacted]'
export CLIENT_SECRET='[redacted]'
export TENANT_ID='[redacted]'
export MAIL_USERNAME='[redacted]@outlook.com'
export AUTHORITY ='https://[redacted].com/{TENANT_ID}'
export SCOPE ='[redacted]'
#<-- microsoft outlook credentials
```

O ficheiro “db.sqlite3” é a base de dados utilizada pelo sistema, que, como foi mencionado, diz respeito ao SQLite3 e contém todos os dados do sistema.

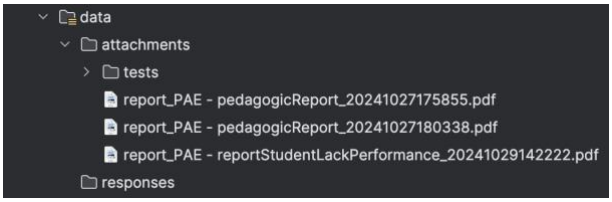
O ficheiro “requirements.txt” é um ficheiro gerado, onde são listadas todas as dependências do projeto, quais as bibliotecas utilizadas e as versões das mesmas. Isto facilita que a migração do projeto, uma vez que os requisitos estão listados de uma forma clara.

Por fim, o ficheiro “manage.py”, é um script Python do próprio Django e serve de suporte à execução de comandos específicos, deste o iniciar do servidor de desenvolvimento, como a execução dos comandos “*makemigrations*” e “*migrate*”, essênciais para a propagação das alterações dos modelos para a estrutura da base de dados, através das funcionalidades ORM do próprio Django.

Pasta “data”

Aqui são armazenados os ficheiros resultantes do processamento de serviços.

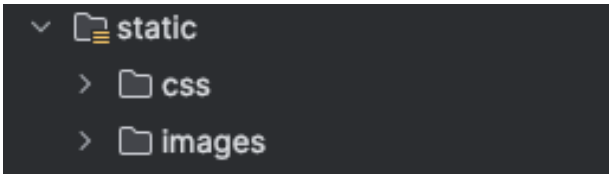
Figura 23 – Pasta “data”



Pasta “static”

Aqui são armazenados os ficheiros ".css" e as imagens utilizadas no âmbito das interfaces a que o utilizador tem acesso.

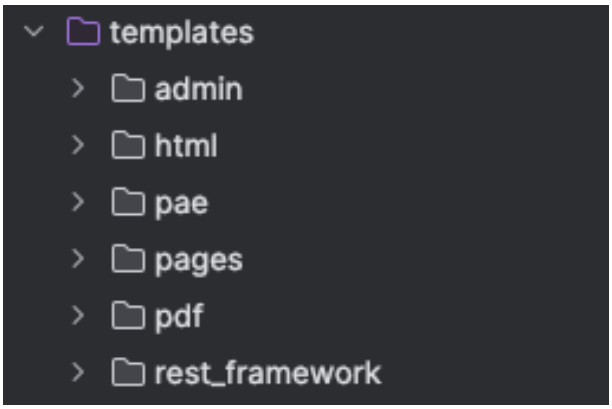
Figura 24 – Pasta "static"



Pasta "templates"

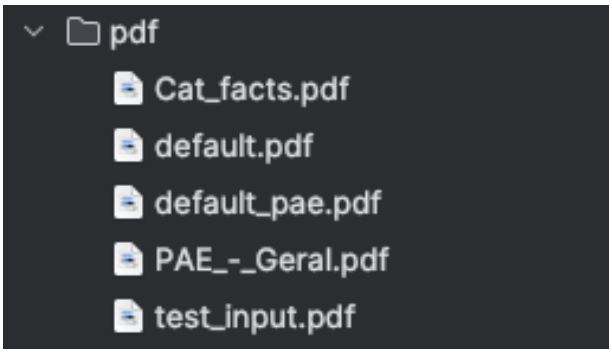
Nesta pasta é possível a consulta dos ficheiros HTML que servem de base a toda a componente visual do projeto, através do sistema de templates do Django é possível a adição de fragmentos de outras templates, ciclos e lógica condicional.

Figura 25 – Pasta "templates"



Em particular, a pasta "pdf" guarda os ficheiros que servem de template no processo de geração de relatórios, assumindo o nome dos serviços a que estão associados.

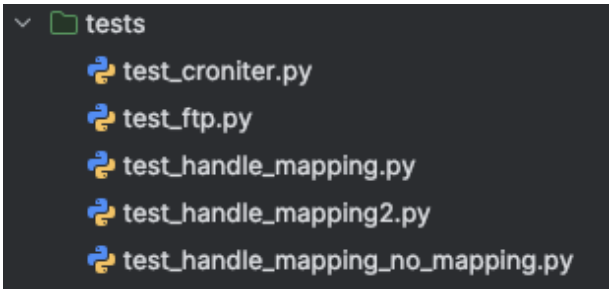
Figura 26 – Pasta "templates/pdf"



Pasta "tests"

Guardou-se nesta pasta todos os scripts gerados para a implementação e processamento de cenários de teste, para as mais variadas funcionalidades do sistema.

Figura 27 – Pasta "tests"

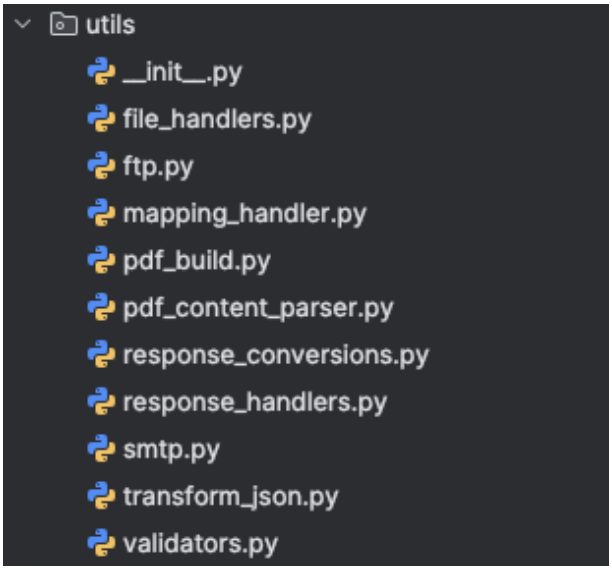


Na secção "Testes Realizados" este tópico é abordado com um foco maior no detelhe de cada teste efetuado.

Pasta "utils"

Os scripts aqui definidos servem o princípio de reutilização, uma vez que não têm nenhuma ligação rígida com quaisquer dos módulos da plataforma.

Figura 28 – Pasta "utils"



CAPÍTULO V – IMPLEMENTAÇÃO

1 Ambiente de Desenvolvimento

Hardware:

- Computador: Mac M1
- Sistema Operacional: MacOS Sonoma 14.2.1

Software:

- Editor de código: PyCharm
- Versão do Python: 3.12
- Framework Django: Django>=5.0.6
- Banco de dados: SQLite
- Ferramenta de versionamento: PyCharm (integrado)

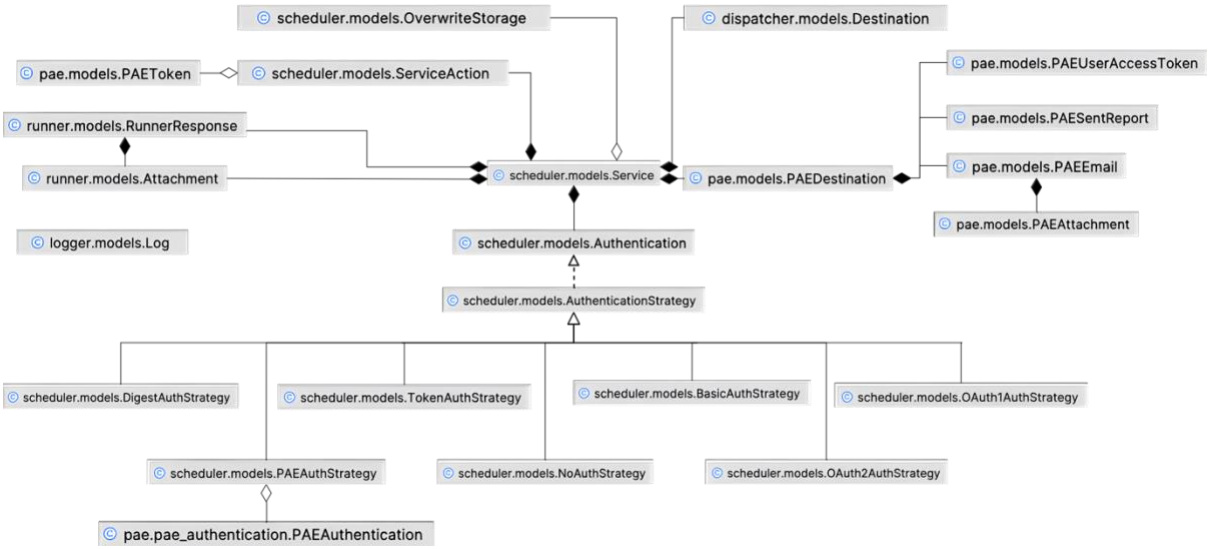
Bibliotecas e dependências:

- PyPDF2>=3.0.1
- reportlab~=4.2.0
- xmltodict~=0.13.0
- celery~=5.4.0
- djangorestframework~=3.15.1
- requests~=2.32.3
- python-dotenv~=1.0.1
- croniter~=3.0.3
- requests-oauthlib~=2.0.0

2 Estrutura do Projeto

Para servir de suporte a este capítulo, apresenta-se aqui o diagrama de classes da solução, no Anexo 4 – "Diagrama de Classes Completo" é disponibilizada outra versão mais detalhada do mesmo diagrama.

Figura 29 – Diagrama de classes da plataforma



Classe "Serviço"

Esta classe é o ponto focal do desenvolvimento, é composta por várias características que determinam o ponto de acesso à informação e a forma como esta deve ser manipulada.

Figura 30 – Classe "Serviço"

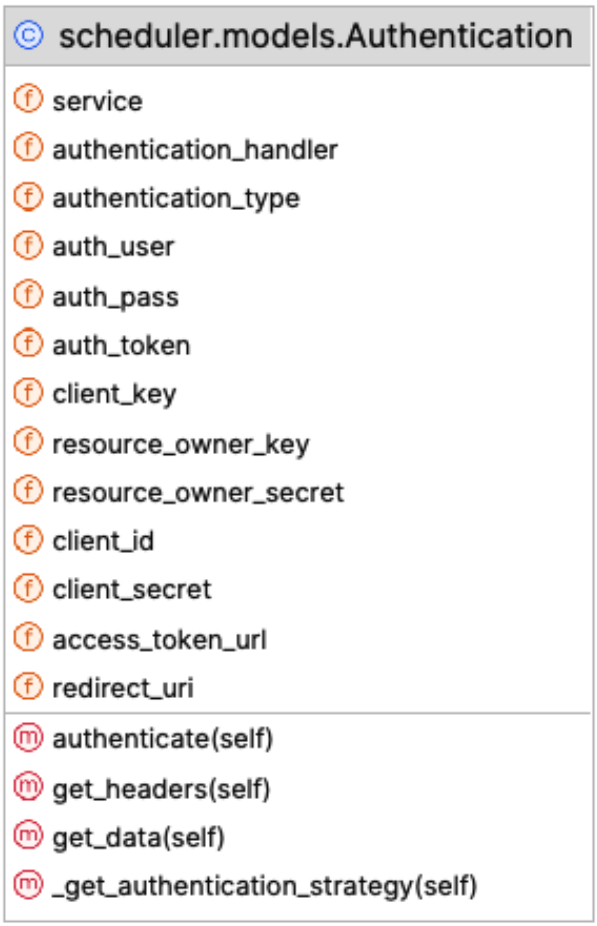
```
scheduler.models.Service
DELIVERY_CHOICES
PERIODIC
MESSAGE_TYPE_CHOICES
OUTPUT_CHOICES
INTERVAL_UNITS
AUTHENTICATIONS
name
date
delivery
email_text
endpoint
delivery_endpoint
last_updated
input_data_type
output_type
pdf_transformer
pdf_template
cron_update_interval
archive_time
mapping
response_mapping
__str__(self)
get_all_periodic(cls)
get_last_updated(self)
get_all_periodic_expired(cls)
send_request(self)
get_request(self)
```

A classe entrega vários métodos que permitem a recolha de informação sobre os serviços sob agendamento periódico e outras informações relevantes, como a última data de atualização e um método para efetuar a chamada do serviço.

Classe "Authentication"

Esta classe é responsável pelo armazenamento das informações necessárias ao processo de autenticação para a comunicação com sistemas externos.

Figura 31 – Classe "Authentication"

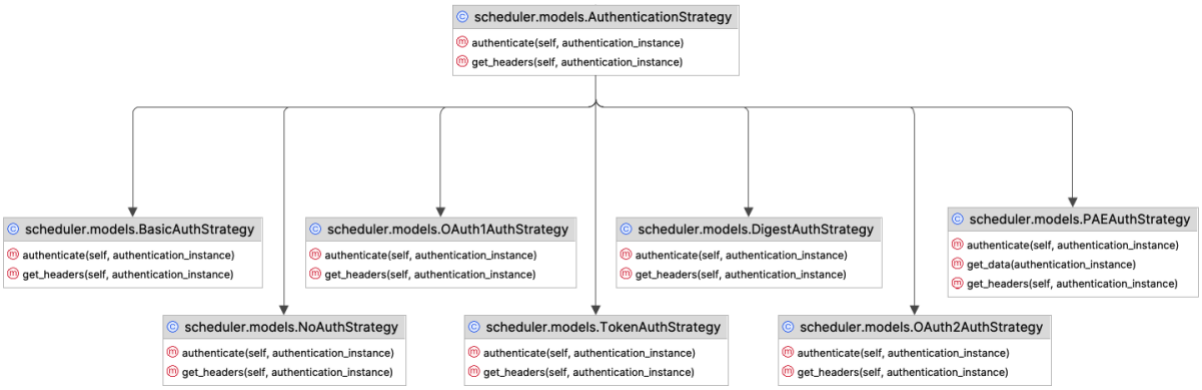


Os métodos aqui definidos são como que interfaces das implementações reais destes processos, e dependem do tipo de autenticação necessária.

Classe "AuthenticationStrategy"

Como o próprio nome indica, esta classe é uma classe "ABC" do Django, ou seja, uma "classe base abstrata" (tradução livre de "abstract base classe"). Isto significa que esta classe apenas serve de "base" a outras classes para a atribuição de atributos e métodos, assim sendo, é uma forma de herança e não envolve a criação de uma tabela na base de dados.

Figura 32 – Classe "AuthenticationStrategy" e subclasses



Esta classe define, com recurso ao padrão "Strategy" (do Inglês, estratégia), uma forma comum de aplicar todas estas formas de autenticação à classe "Authentication" original. Assim é possível suportar vários tipos de autenticação, sem a necessidade de recorrer a processos de implementação independentes.

Cada uma das classes que herda de "AuthenticationStrategy" tem a sua própria implementação dos métodos "authenticate" e "get_headers", que assistem na comunicação e criação de chamadas a serviços.

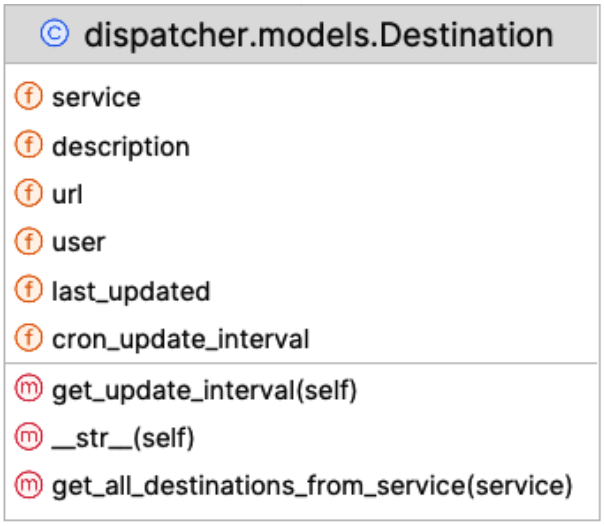
Figura 33 – Classe "PAEAuthentication"

A classe "PAEAuthentication" é um caso especial, uma vez que ela própria herda da estratégia "PAEAuthStrategy". Isto foi necessário de modo a ser possível a implementação desta classe no módulo "PAE" e não gerar dependências circulares entre os módulos do projeto.

Classe "Destination"

Esta foi a classe que contém a implementação inicial da lógica de distribuição de dados pelos subscritores de um serviço.

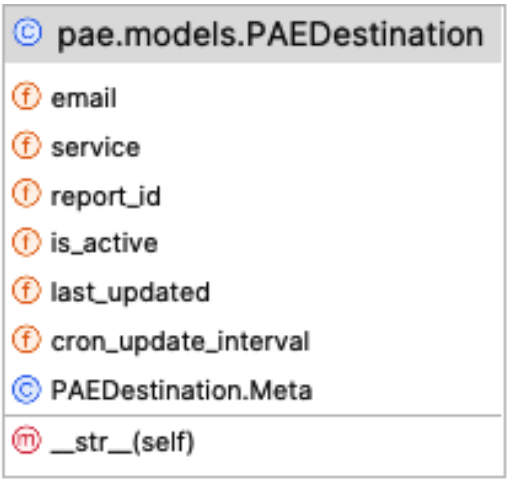
Figura 34 – Classe "Destination"



Classe "PAEDestination"

Esta é a implementação ajustada ao serviço do PAE e tem várias diferenças face à implementação inicial. De modo a suportar a plataforma e os seus requisitos específicos, ao invés de um qualquer endereço, apenas se suportam endereços de e-mail. Adicionalmente, uma *"flag"* para o controlo do estado da subscrição "*is_active*" foi adicionado.

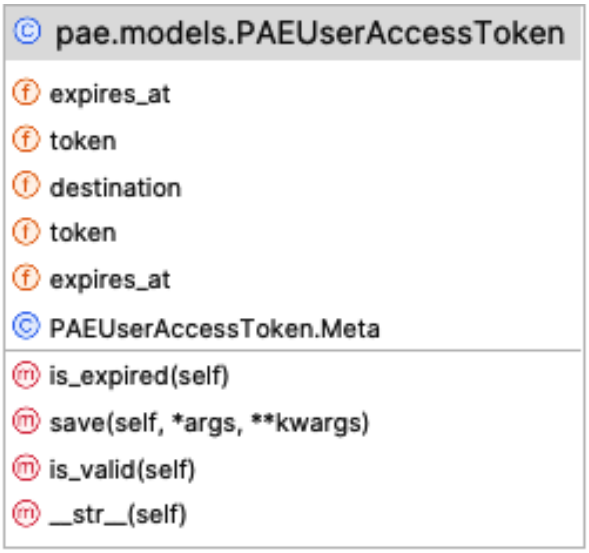
Figura 35 – Classe "PAEDestination"



Classe "PAEUserAccessToken"

Aqui são definidos os *tokens* partilhados com cada utilizador receptor das notificações provenientes do PAE. A lógica existente contempla a criação de *tokens* e a lógica de validação dos mesmos.

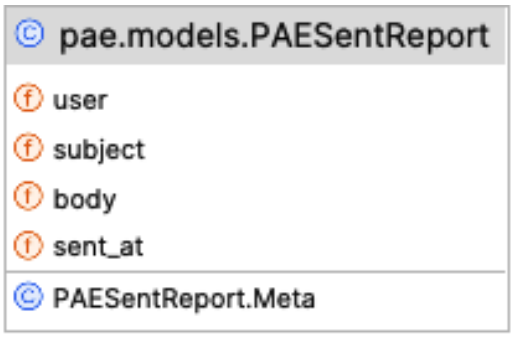
Figura 36 – Classe "PAEUserAccessToken"



Classe "PAESentReport"

Esta classe é fez parte da implementação inicial da lógica de registo dos emails enviados. Foi depois substituída pelas classes abordadas em seguida e considera-se depreciada.

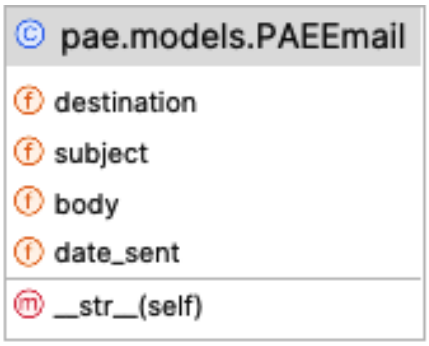
Figura 37 – Classe "PAESentReport"



Classes "PAEEmail" e "PAEAttachment"

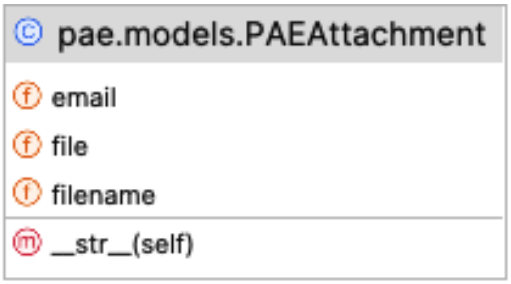
O conjunto de classes aqui descrito trata o histórico dos e-mails enviados, bem como os anexos incluídos em cada notificação.

Figura 38 – Classe "PAEEmail"



Foi separada a lógica de anexos de forma a possibilitar o associar de vários anexos a uma só notificação.

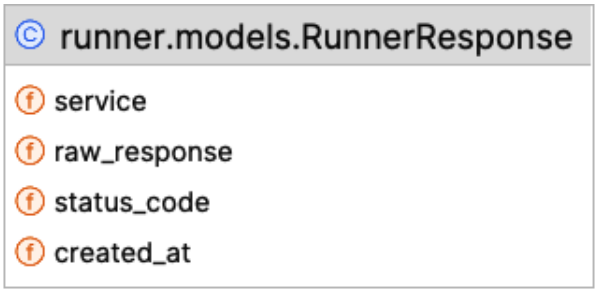
Figura 39 – Classe "PAEAttachment"



Classes "RunnerResponse" e "Attachment"

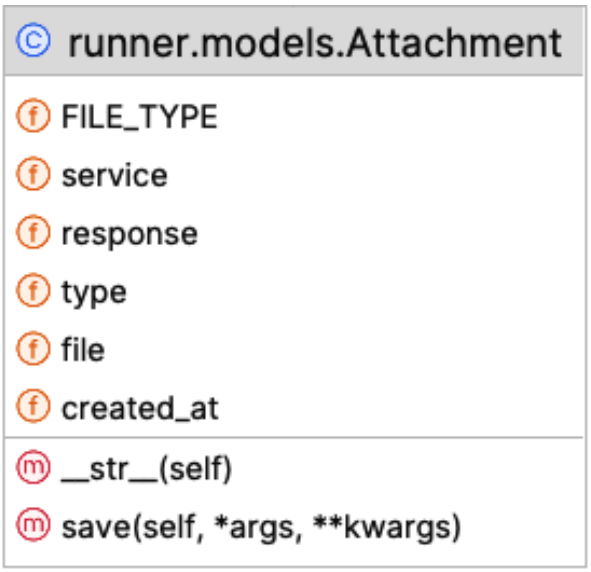
O processo de comunicação entre serviços e a transformação dos dados recebidos é assistido por estas classes, em que a primeira respeita o registo dos dados recebidos, caso seja necessária a recolha de informação sobre alguma comunicação ou falha.

Figura 40 – Classe "RunnerResponse"



Já a segunda, assegura que a informação transformada fica guardada no sistema, de forma a ser processada mais tarde pelo processo de distribuição de notificações.

Figura 41 – Classe "Attachment"



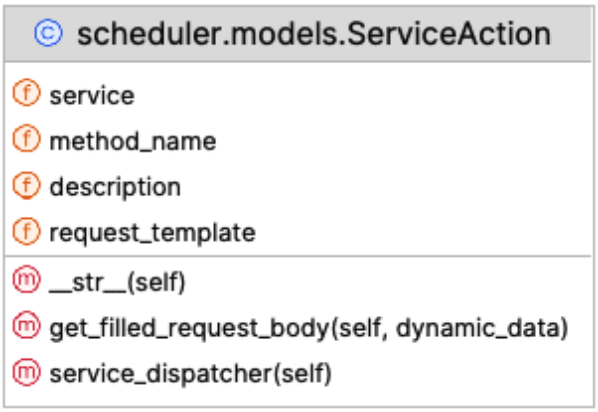
Classes "ServiceAction" e "PAEToken"

A classe "ServiceAction" foi criada com o propósito de permitir a submissão de dados em pedidos POST a serviços externos. Surgiu da necessidade da integração com a interface do PAE disponibilizada. A razão pela qual não segue a mesma estrutura de nomenclatura das

restantes é porque este processo se trata de uma funcionalidade genérica e aplicável a um qualquer serviço.

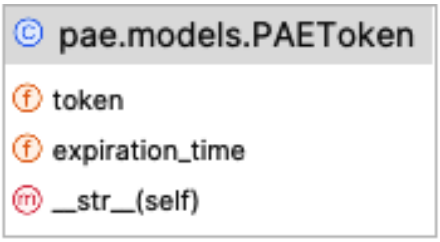
Aqui é possível a utilização de *templates* para o uso de informação dinâmica. No caso particular do PAE, isto é utilizado para o preenchimento to *token* "BACOSESS" no corpo do pedido com o resultado do processo de autenticação do serviço.

Figura 42 – Classe "ServiceAction"



Do mesmo modo, foi criada a classe PAEToken, com o propósito de gerir a data de expiração destes *tokens*. Assim, evita-se a repetição do processo de autenticação, caso estes ainda sejam válidos.

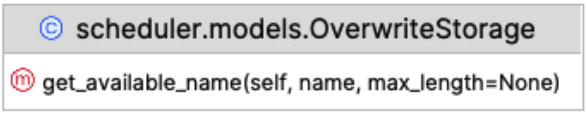
Figura 43 – Classe "PAEToken"



Classe "OverwriteStorage"

Esta classe é responsável pela gestão dos ficheiros armazenados como *templates* para o processo de geração de PDFs, garantindo que estes são substituidos corretamente e ficam alojados no diretório definido para tal no sistema de ficheiros.

Figura 44 – Classe "OverwriteStorage"

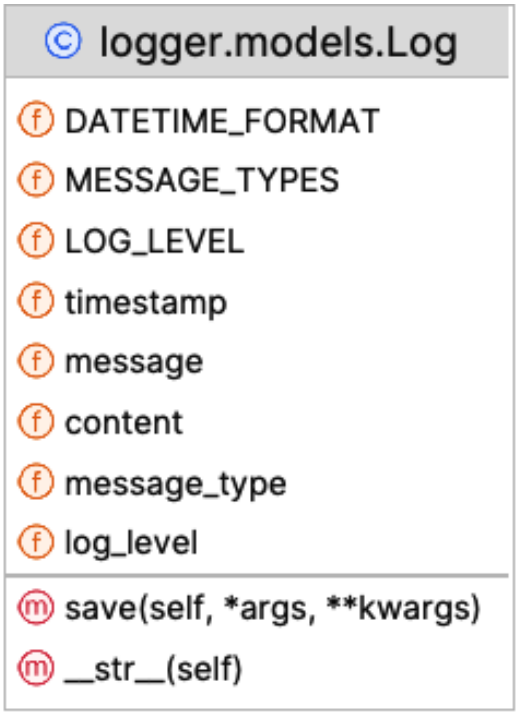


Classe "Log"

Esta classe apenas têm um método "save", que permite a recolha no modelo de dados de vária informação da execução da plataforma, foi definida uma estrutura genérica e vários

tipos de mensagem como “Sucesso”, “Aviso”, “Informação” e “Erro”, com o objetivo de dar suporte à monitorização do sistema.

Figura 45 – Classe “Log”



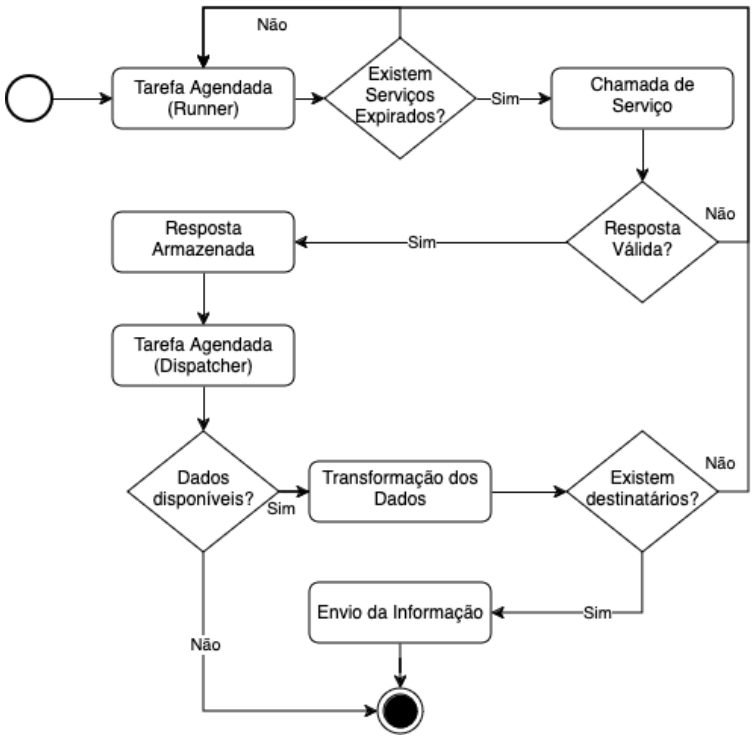
3 Implementação das Funcionalidades

As funcionalidades implementadas são descritas na seguinte secção, aqui procura-se oferecer alguma informação sobre como foram definidos os principais processos e qual o racional por detrás das funcionalidades implementadas.

3.1 Processo de chamadas periódicas de Serviços

Com recurso ao Celery, as tarefas definidas como periódicas pela anotação “shared_task” são processadas com a periodicidade definida na plataforma (por defeito, 1 minuto). Neste caso podemos ver a lógica de chamada de serviços, em que é recolhida a lista de serviços “expirados” e se trata à renovação da informação que estes prestam. É possível a consulta de parte deste processo no Anexo 5.

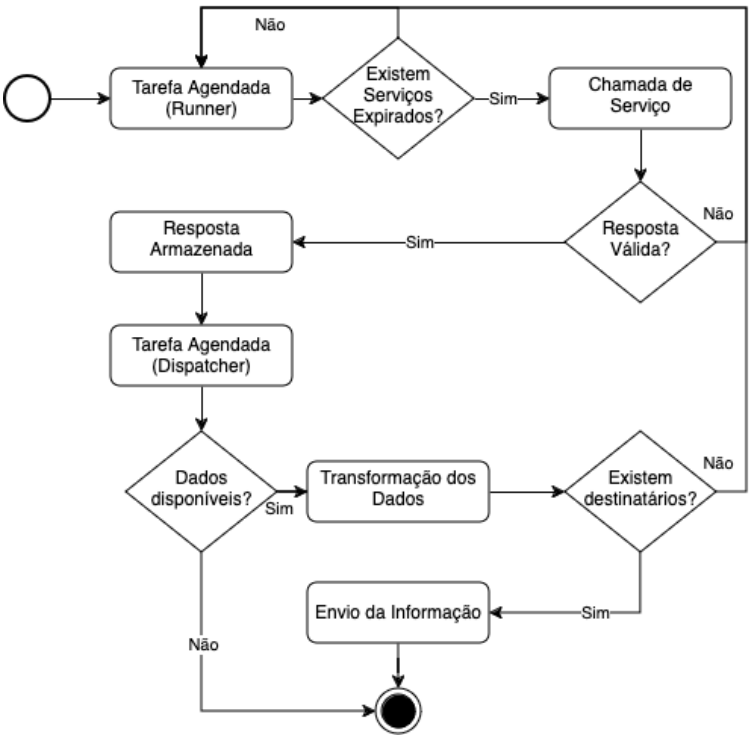
Figura 46 – Representação do fluxo de chamadas periódicas de Serviços



3.2 Envio de e-mails

Aqui é possível a consulta de como é tratado o envio de e-mails, com recurso à biblioteca "smtplib", outro ponto relevante é a lista "attachment_paths", que permite assim o anexo de vários ficheiros num só email, caso necessário. É possível a consulta de parte deste processo no Anexo 6.

Figura 47 - Representação do fluxo para o envio de e-mails



3.3 Definição de estratégias da classe "Authentication"

A classe Authentication define diferentes estratégias de autenticação que podem ser selecionadas dinamicamente com base no tipo de autenticação configurado no registo de cada Serviço.

Tabela 6 – Definição de estratégias da classe "Authentication"

```
class Authentication(models.Model):
    def _get_authentication_strategy(self):
        strategies = {
            'NONE': NoAuthStrategy(),
            'BASIC': BasicAuthStrategy(),
            'TOKEN': TokenAuthStrategy(),
            'DIGEST': DigestAuthStrategy(),
            'OAUTH1': OAuth1AuthStrategy(),
            'OAUTH2': OAuth2AuthStrategy(),
            'PAE': PAEAuthStrategy(),
        }

        if self.authentication_handler == 'DEFAULT':
            return strategies.get(self.authentication_type)
        else:
            return strategies.get(self.authentication_handler)
```

3.4 Estratégia de manipulação com base no tipo de input

A plataforma lida com diferentes tipos de respostas (JSON, XML, CSV, TEXT) baseando-se no tipo de entrada e saída especificado para um serviço.

Tabela 7 – Tratamento de vários tipos de resposta com base no tipo de input

```
def handle_response(response: HttpResponse, service: Service):
 input_type = service.input_data_type
 output_type = service.output_type

 if input_type == output_type:
    return response

 if input_type == 'JSON':
    return handle_json_response(response, output_type, service)

 elif input_type == 'XML':
    return handle_xml_response(response, output_type, service)

 elif input_type == 'CSV':
    return handle_csv_response(response, output_type, service)

 elif input_type == 'TEXT':
    return handle_text_response(response, output_type, service)
```

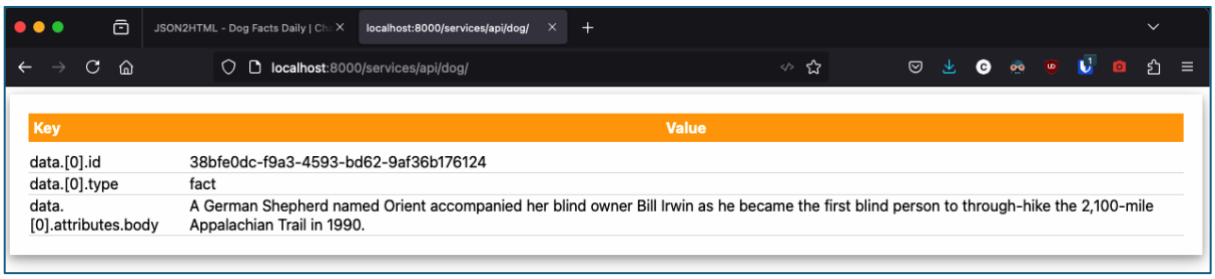
3.5 População de elementos HTML

Aqui é possível ver o comportamento que trata o preenchimento do nosso elemento HTML com base nos dados recebidos, que são convertidos para uma lista uni-dimensional antes de ser aplicádos sobre a nossa template HTML.

Tabela 8 – Lógica para o popular de templates HTML para a entrega de dados

```
def json2html(byte_data: bytes) -> HttpResponse:
 json_data = json.loads(byte_data.decode())
 flat_data = flatten(json_data)
 first_key = next(iter(flat_data))

 if first_key != "data":
    flat_data = {'data': flat_data}

 rendered_html = render(None, '../templates/html/default.html', flat_data)
 return rendered_html
```

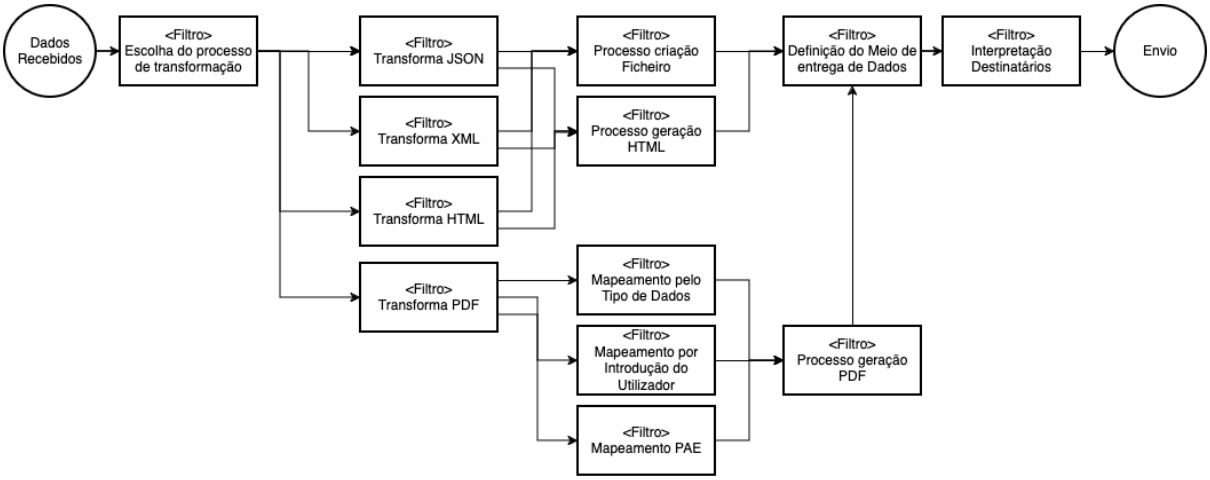
Figura 48 – Exemplo de elemento HTTP gerado



3.6 Tratamento de dados recebidos

Esta função documenta o processo de transformação dos dados recebidos, tendo cada um o seu tratamento específico de acordo com o tipo de dados a tratar, seja uma lista ou um dicionário. Por convenção, elementos com o prefixo “custom_” são tratados como texto fixo a inserir no output, esta informação é mantida no modelo de cada serviço configurado.

Figura 49 – Exemplo da Pipeline de processamento e transformação dos dados recebidos



Um outro exemplo desta mecânica, foi a definição de um processo “automático”, em que, definindo no modelo do serviço, a configuração do mapeamento como: “"pdf_auto_mapping": true”.

Aqui, não é tida em consideração, nada mais do que o tipo de dados de cada elemento da resposta para a elaboração de relatórios.

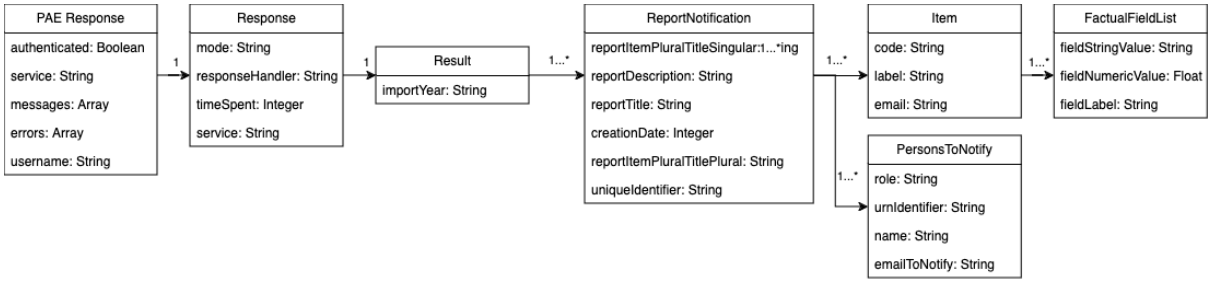
Tabela 9 – Algoritmo genérico para o tratamento de dados JSON

```
def transform_json(input_data, template: dict):
 output_data = {}

 for key, value in template.items():
    if key.startswith("custom_"):
       output_data[key] = value
    elif isinstance(value, dict):
       output_data[key] = transform_json(input_data.get(key), value)
    elif isinstance(value, list):
       output_data[key] = []
       for sub_template in value:
          if isinstance(sub_template, dict):
             output_data[key] = [transform_json(item2, sub_template) for item2 in
input_data.get(key)]
          else:
             output_data[key] = input_data.get(key)
    elif value == "display":
       output_data[key] = input_data.get(key)
    elif value == "hide":
```

```
        pass
    else:
        pass

return output_data
```

Por último, é apresentada a estrutura JSON para a elaboração de relatórios da plataforma PAE. Neste caso, foi negociada uma estrutura rígida para a comunicação entre as plataformas, deste modo é possível a entrega de relatórios personalizados. A principal vantagem passa por conseguir assim oferecer ao utilizador um qualquer tipo de relatório desde que os dados sejam entregues no formato definido.

Figura 50 – Estrutura JSON do PAE



Os dados enviados para os utilizadores do PAE são armazenados nas tabelas descritas abaixo. Isto permite a identificação dos utilizadores e do conteúdo recebido por cada um deles. Essencial para a recuperação posterior desta informação através dos *tokens* que já foram apresentados.

Tabela 10 – Exemplo dos registos da tabela PAEAttachment

#	id	file	filename	email_id
1	1	data/attachments/report_PAE - pedagogicReport_20241027180338.pdf	report_PAE - pedagogicReport_20241027180338.pdf	3
2	2	data/attachments/report_PAE - reportStudentLackPerformance_20241029142222.pdf	report_PAE - reportStudentLackPerformance_20241029142222.pdf	4
3	3	data/attachments/report_PAE - pedagogicReport_20241117155613.pdf	report_PAE - pedagogicReport_20241117155613.pdf	5

Optou-se pela separação da tabela que armazena a informação de anexos e da que regista o conteúdo de cada email, isto permite gerir de uma forma mais fácil casos em que um email possa conter mais do que um anexo.

Tabela 11 – Exemplo dos registos da tabela PAEEmail

#	id	subject	body	date_sent	destination_id
1	3	PAE - pedagogicReport Report	<email message>	2024-10-27 18:03:36.586086	5
2	4	PAE - reportStudentLackPerformance Report	<email message>	2024-10-29 14:22:20.991849	6
3	5	PAE - pedagogicReport Report	<email message>	2024-11-17 15:56:13.370963	5

Da mesma forma, a tabela que contém os registos de cada destinatário e a frequência com que estes devem receber atualizações é mantida em separado.

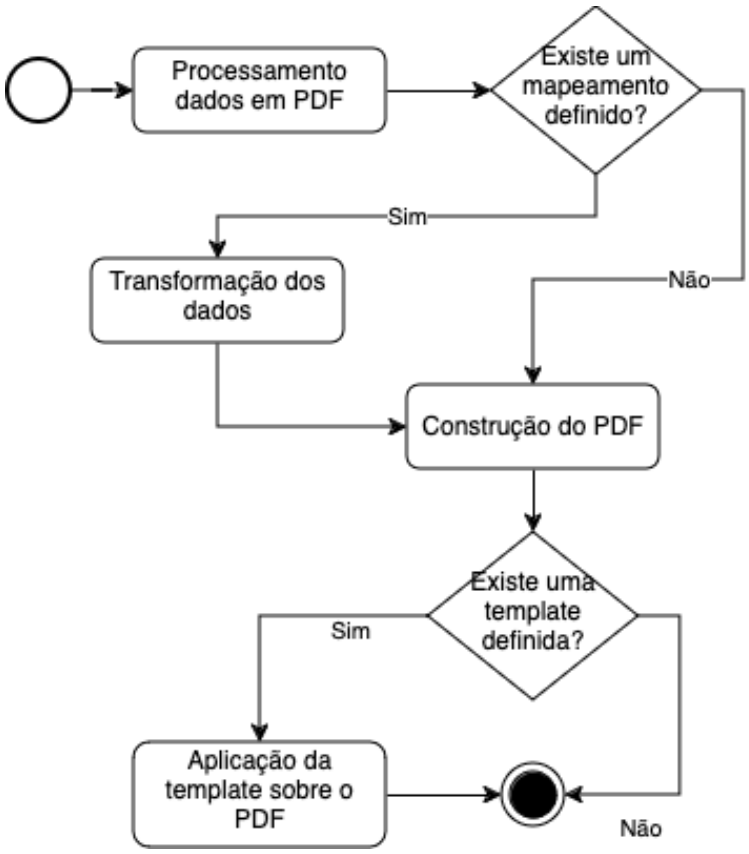
Tabela 12 - Exemplo dos registos da tabela PAEDestination

#	id	email	is_active	service_id	cron_update_interval	last_updated	report_id
1	1	xxxxx@ipportalegre.pt	1	8	* * * * *	2024-10-26 14:17:35.524320	21217
2	2	xxxxx@ipportalegre.pt	1	8	* * * * *	2024-10-26 14:17:35.524320	21217
3	3	xxxxx@ipportalegre.pt	1	9	* * * * *	2024-10-26 14:18:23.837902	pedagogicResults Report
4	5	xxxxx@outlook.com	1	9	0 0 1 * *	2024-10-27 17:17:32.773317	pedagogicResults Report
5	6	xxxxx@outlook.com	0	8	* * * * *	2024-10-26 14:17:35.524320	21217

3.7 Criação de ficheiros PDF

A geração de PDFs assenta sobre os recursos da biblioteca ReportLab. Nesta implementação o conteúdo do PDF é gerado de forma dinâmica, ou no caso do PAE, segundo a estrutura JSON pré-definida. Caso o sistema tenha associada alguma *template* esta é aplicada sobre o ficheiro gerado, sendo a primeira página tida como capa, caso haja seguintes, a ordem é mantida, sendo a última considerada a opção “por defeito”.

Figura 51 – Fluxo de geração dos ficheiros PDF



4 Desafios e Soluções

Um dos desafios encontrados foi a depreciação da autenticação básica nos provedores Microsoft e Google para o protocolo SMTP. Para garantir a continuidade da comunicação de forma segura, optou-se pela criação de uma *passkey*, o que não apenas manteve a funcionalidade do sistema, mas também aumentou a segurança na transmissão de dados.

Inicialmente, o sistema foi projetado considerando apenas períodos de atualização pré-definidos para os serviços. No entanto, à medida que a solução evoluiu, surgiu a necessidade de permitir que os utilizadores definissem seus próprios intervalos de notificação. Para atender a essa nova demanda, foi necessário adaptar a lógica de geração de relatórios para que considerasse dois períodos distintos: o período de atualização do serviço e o intervalo de notificação definido pelo utilizador. A abordagem implementada assegura que os utilizadores recebam notificações apenas quando ambos os períodos estiverem expirados, garantindo que cada relatório contenha informações atualizadas e relevantes.

A abordagem a este tipo de utilizadores, sem registo na plataforma, obrigou também a desenvolver um processo que permitisse a identificação destes, sem comprometer a segurança e a privacidade da plataforma, face a este problema, optou-se pela criação de *tokens* individuais

e temporários, deste modo a autenticação de cada utilizador fica assente sobre o seu provedor de e-mail.

No contexto de otimizações, implementou-se um sistema de cache, onde o sistema verifica se a informação necessária já está disponível e ainda é válida antes de fazer chamadas a serviços externos. Para chamadas imediatas de serviços, o sistema não segue a mesma abordagem, e as chamadas são sempre efetuadas no momento do pedido.

CAPÍTULO VI – ANÁLISE DE RESULTADOS

1 Instalação e Utilização

1.1 Instalação

A instalação deste projeto passa pela disponibilização deste e dos ficheiros correspondentes num servidor local ou público, neste último é necessária a contratação de um Provider para alojar a aplicação num servidor externo.

Seja como for, existem dois pontos na configuração do Django que são importantes antes de tornar o projeto disponível num ambiente produtivo, as variáveis "DEBUG" e "SECRET_KEY", em que a primeira deve ser marcada como "False" e a segunda deve-se garantir que não está disponível em nenhum sistema de controlo de versões ou de outro modo disponível fora desse ambiente de produção [62].

Existe também a possibilidade do Provider não permitir a persistência de dados no ambiente aplicacional, que no caso da estrutura desta aplicação implicaria a uma migração da base de dados SQLite, para uma outra.

Outro passo importante a seguir seria a configuração do Django para servir ficheiros estáticos sem passar pelo servidor web de desenvolvimento do Django, que como o nome indica, apenas deve ser usado para tarefas de desenvolvimento, principalmente por uma questão de performance.

Não esquecer as dependências relevantes para o projeto: Celery, Redis, PyPDF2, reportlab, xmltodict, Django REST Framework, Requests, Python-dotenv. que devem estar disponíveis na mesma máquina.

1.2 Interface de Utilizador

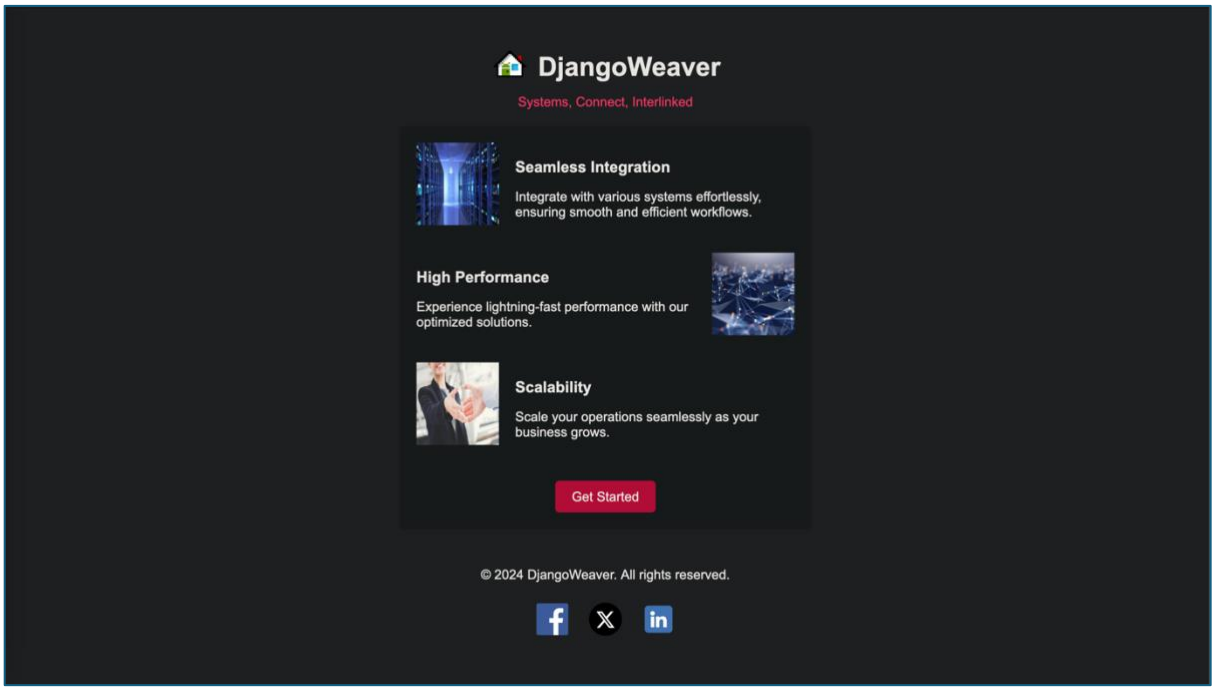
A interface de utilizador é bastante simples, uma vez que os casos de uso identificados não requerem, por norma, a utilização direta da plataforma pelos utilizadores que recebem os relatórios. A plataforma foi nomeada "DjangoWeaver", um nome que serve de *placeholder* em todas as componentes visuais da aplicação.

Em seguida são partilhados os vários pontos da plataforma onde é possível a interação direta com o sistema.

Página Inicial

A página inicial do DjangoWeaver oferece uma visão geral da plataforma e serve como ponto de entrada para a àrea administrativa.

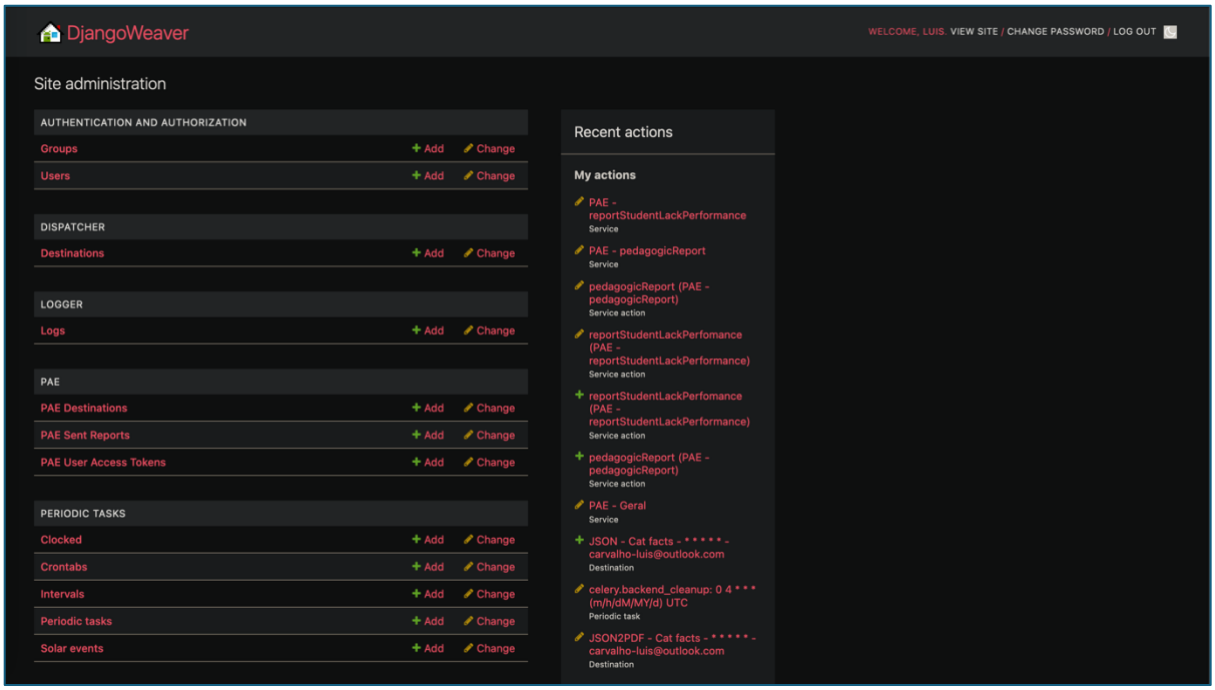
Figura 52 – DjangoWeaver: Página Inicial



Administração

A página de administração oferece acesso a funcionalidades avançadas e permita a consulta e manutenção dos dados do sistema.

Figura 53 – DjangoWeaver: Página de Administração



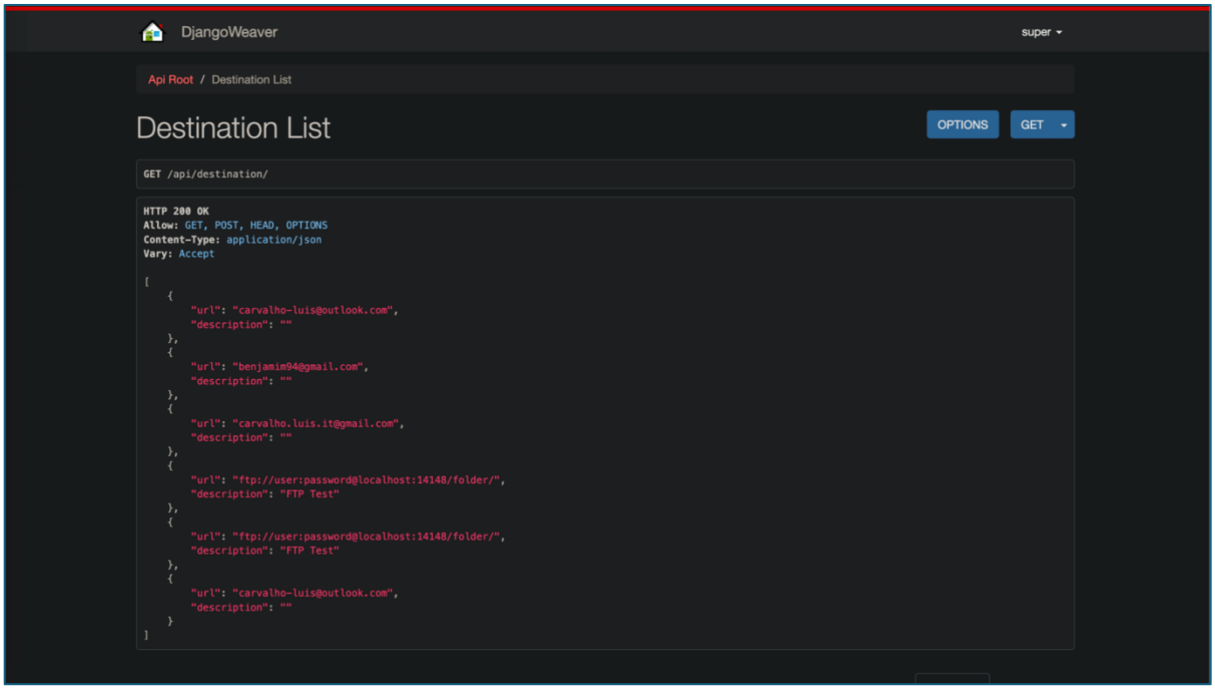
Área de API

Associado ao URL /api/, está disponível a nevageção pelas várias interfaces e serviços disponibilizados com recurso à biblioteca Django REST Framework.

Destinatários

Aqui é possível ver a interface onde se podem manter os destinatários do sistema.

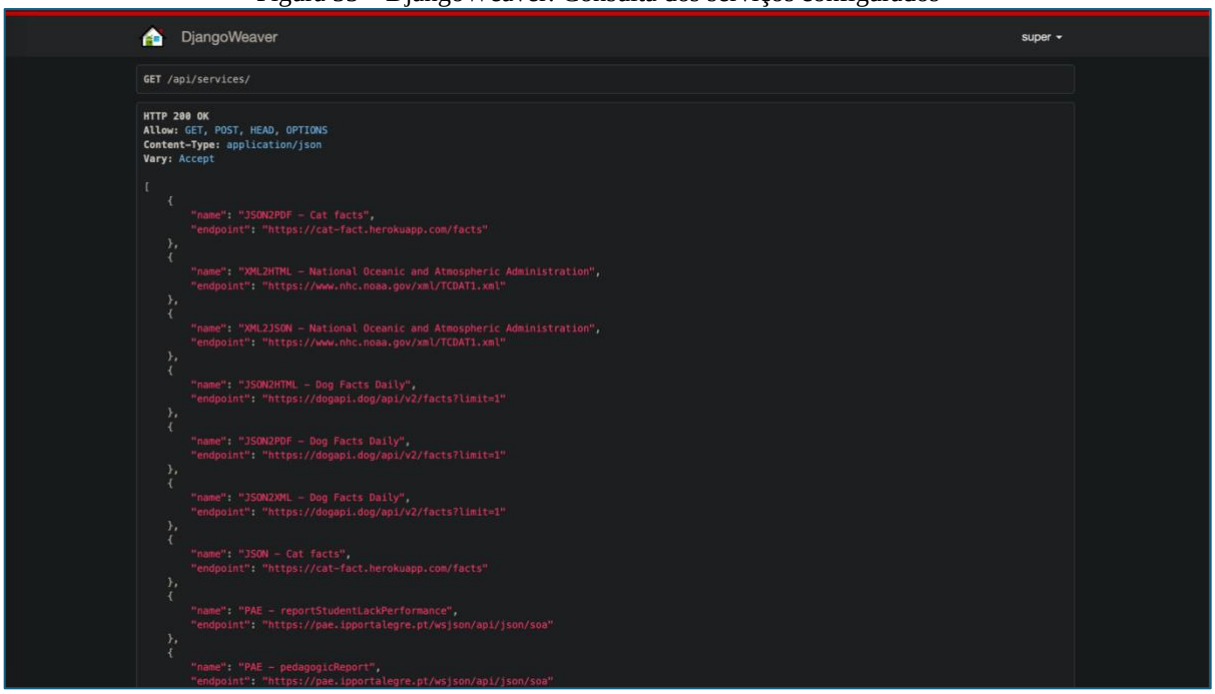
Figura 54 – DjangoWeaver: Consulta de destinatários configurados



Serviços

Esta listagem apresenta os serviços configurados no sistema, bem como os URLs originais que servem de fonte de informação.

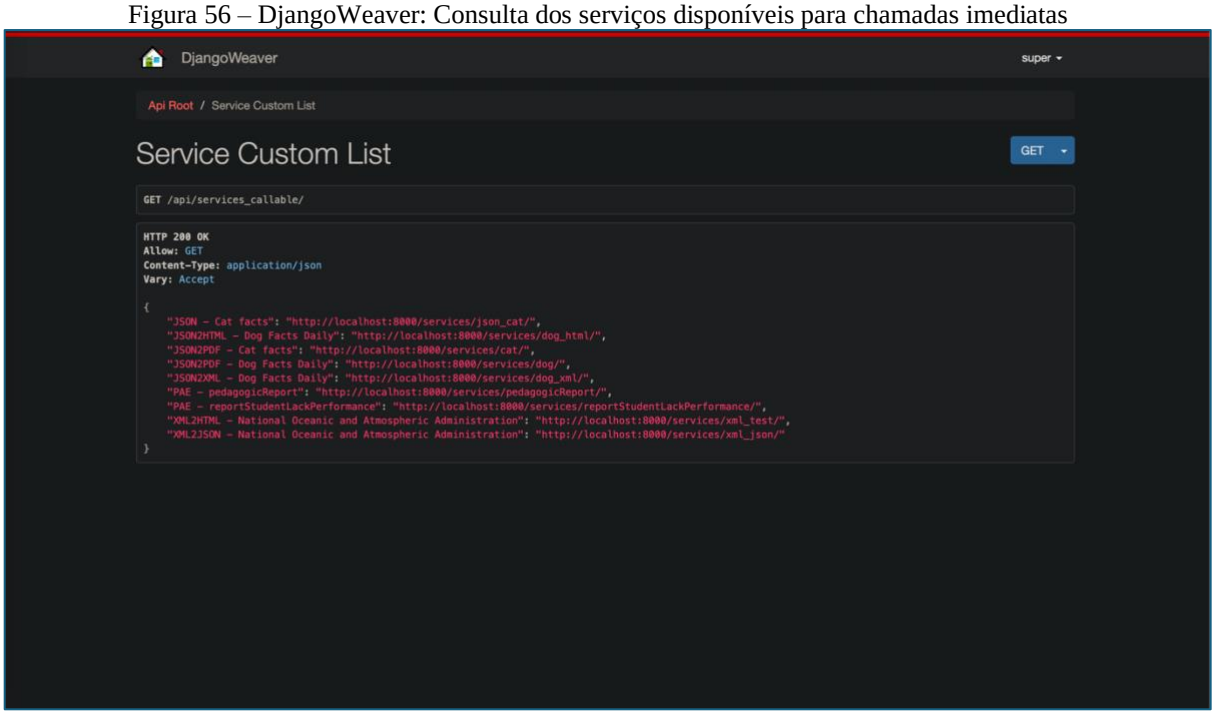
Figura 55 – DjangoWeaver: Consulta dos serviços configurados



Serviços "On-Demand"

Aqui é possível ver quais os serviços invocáveis sob-demanda, através dos links internos definidos na plataforma através da camada de administração.

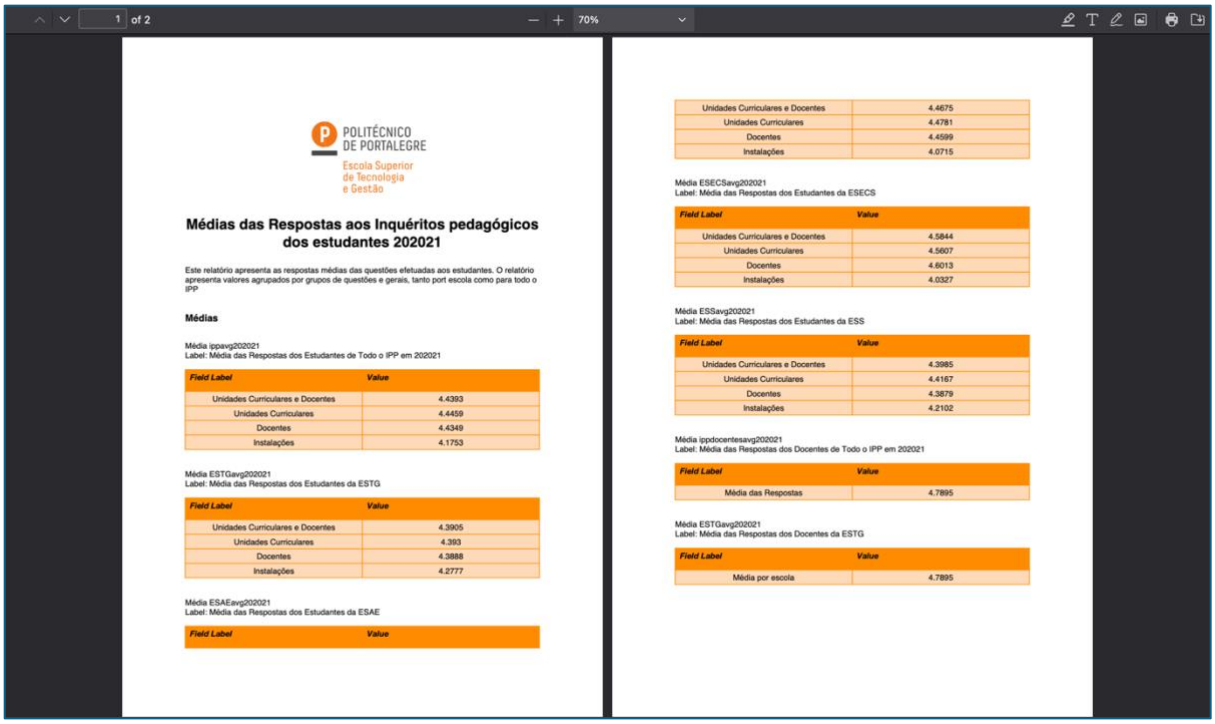
Figura 56 – DjangoWeaver: Consulta dos serviços disponíveis para chamadas imediatas



Exemplos de Relatórios

Aqui são apresentados exemplos dos relatórios gerados com base nos dados recebidos.

Figura 57 – DjangoWeaver: Exemplo de um relatório produzido #1



POLITÉCNICO DE PORTALEGRE
Escola Superior de Tecnologia e Gestão

Médias das Respostas aos Inquéritos pedagógicos dos estudantes 202021

Este relatório apresenta as respostas médias das questões efetuadas aos estudantes. O relatório apresenta valores agrupados por grupos de questões e gerais, tanto port escola como para todo o IPP

Médias

Média ippavg202021
Label: Média das Respostas dos Estudantes de Todo o IPP em 202021

Field Label	Value
Unidades Curriculares e Docentes	4.4393
Unidades Curriculares	4.4459
Docentes	4.4349
Instalações	4.1753

Média ESTGavg202021
Label: Média das Respostas dos Estudantes da ESTG

Field Label	Value
Unidades Curriculares e Docentes	4.3905
Unidades Curriculares	4.393
Docentes	4.3888
Instalações	4.2777

Média ESAEavg202021
Label: Média das Respostas dos Estudantes da ESAE

Field Label	Value
Unidades Curriculares e Docentes	4.4675
Unidades Curriculares	4.4781
Docentes	4.4599
Instalações	4.0715

Média ESECSavg202021
Label: Média das Respostas dos Estudantes da ESECS

Field Label	Value
Unidades Curriculares e Docentes	4.5844
Unidades Curriculares	4.5607
Docentes	4.6013
Instalações	4.0327

Média ESSavg202021
Label: Média das Respostas dos Estudantes da ESS

Field Label	Value
Unidades Curriculares e Docentes	4.3985
Unidades Curriculares	4.4167
Docentes	4.3879
Instalações	4.2102

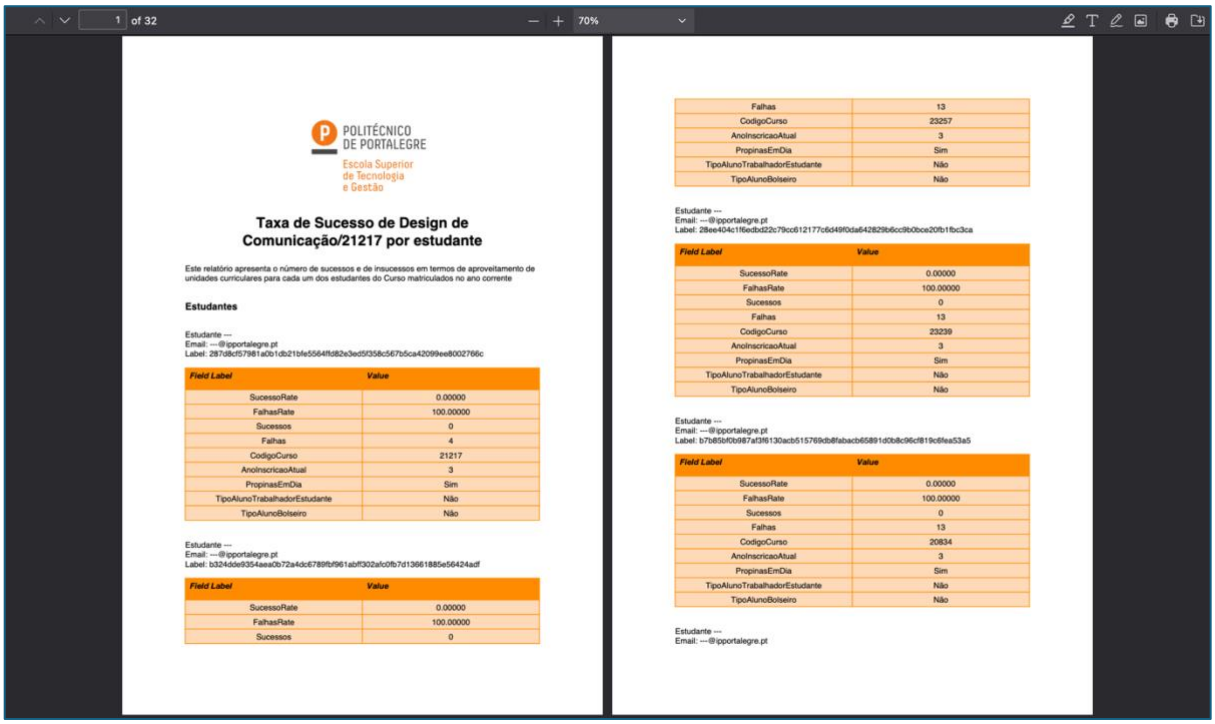
Média ippdocentesavg202021
Label: Média das Respostas dos Docentes de Todo o IPP em 202021

Field Label	Value
Média das Respostas	4.7895

Média ESTGavg202021
Label: Média das Respostas dos Docentes da ESTG

Field Label	Value
Média por escola	4.7895

Figura 58 – DjangoWeaver: Exemplo de um relatório produzido #2



POLITÉCNICO DE PORTALEGRE
Escola Superior de Tecnologia e Gestão

Taxa de Sucesso de Design de Comunicação/21217 por estudante

Este relatório apresenta o número de sucessos e de insucessos em termos de aproveitamento de unidades curriculares para cada um dos estudantes do Curso matriculados no ano corrente

Estudantes

Estudante ---
Email: ---@ipportalegre.pt
Label: 287d8cf57981a0b1db21bfe5564ffd82e3ed5f358c567b5ca42099ee8002766c

Field Label	Value
SucessoRate	0.00000
FalhasRate	100.00000
Sucessos	0
Falhas	4
CodigoCurso	21217
AnoInscricaoAtual	3
PropinasEmDia	Sim
TipoAlunoTrabalhadorEstudante	Não
TipoAlunoBolseiro	Não

Estudante ---
Email: ---@ipportalegre.pt
Label: b324dde9354eea0b72a4dc6789fbf961abff302afc0fb7d13661885e56424adf

Field Label	Value
SucessoRate	0.00000
FalhasRate	100.00000
Sucessos	0
Falhas	13
CodigoCurso	23257
AnoInscricaoAtual	3
PropinasEmDia	Sim
TipoAlunoTrabalhadorEstudante	Não
TipoAlunoBolseiro	Não

Estudante ---
Email: ---@ipportalegre.pt
Label: 28ee404c1f6edbd22c79cc612177c6d49f0da642829b6cc9b0bce20fb1fbc3ca

Field Label	Value
SucessoRate	0.00000
FalhasRate	100.00000
Sucessos	0
Falhas	13
CodigoCurso	23239
AnoInscricaoAtual	3
PropinasEmDia	Sim
TipoAlunoTrabalhadorEstudante	Não
TipoAlunoBolseiro	Não

Estudante ---
Email: ---@ipportalegre.pt
Label: b7b85bf0b987af3f6130acb515769db8fabacb65891d0b8c96cf819c6fea53a5

Field Label	Value
SucessoRate	0.00000
FalhasRate	100.00000
Sucessos	0
Falhas	13
CodigoCurso	20834
AnoInscricaoAtual	3
PropinasEmDia	Sim
TipoAlunoTrabalhadorEstudante	Não
TipoAlunoBolseiro	Não

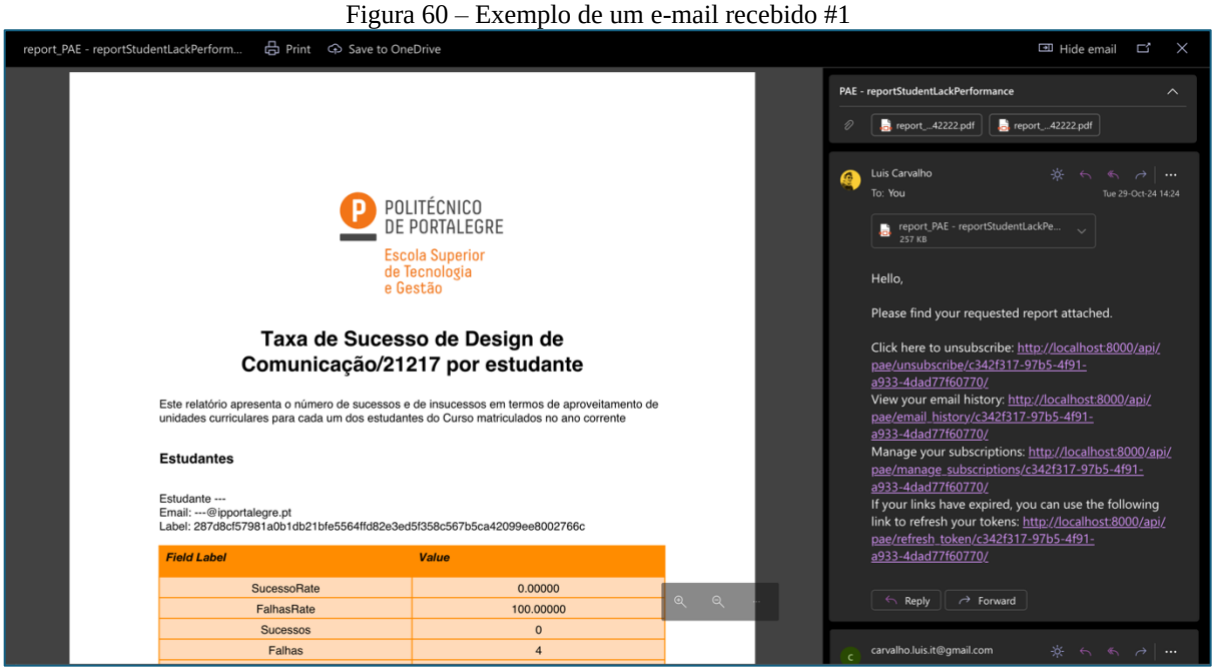
Estudante ---
Email: ---@ipportalegre.pt

Figura 59 – DjangoWeaver: (PAE) Remoção da lista de subscritores

Exemplos de entrega de e-mails

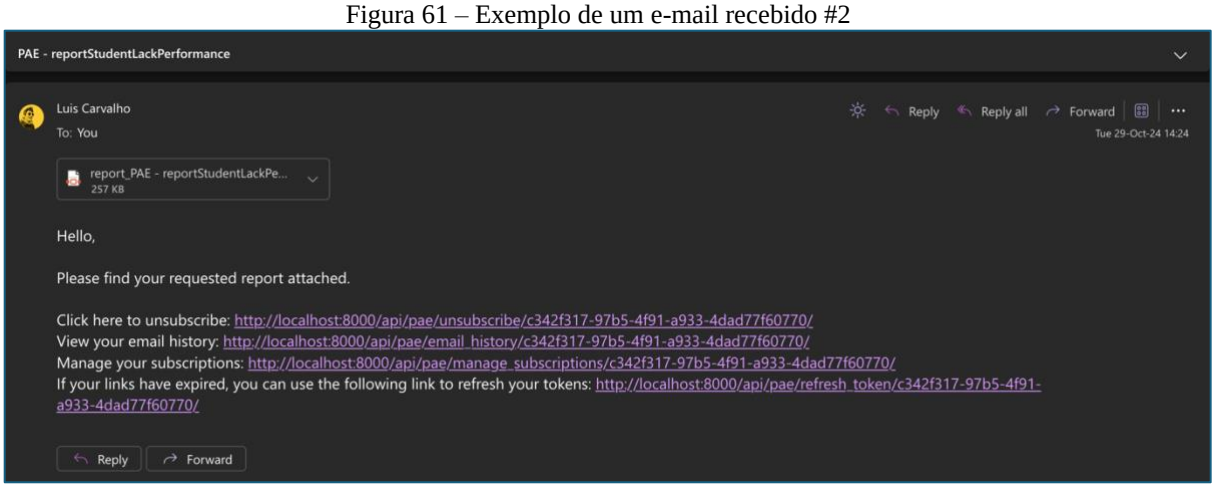
Os dados entregues via e-mail seguem esta estrutura simples, que pode ser personalizada com base na introdução de texto no mapeamento definido na área administrativa da configuração de um Serviço.

Figura 60 – Exemplo de um e-mail recebido #1



No caso do PAE, esta estrutura é mais rígida, devido à necessidade da entrega de vários links que permitam a consulta e a gestão das notificações recebidas.

Figura 61 – Exemplo de um e-mail recebido #2



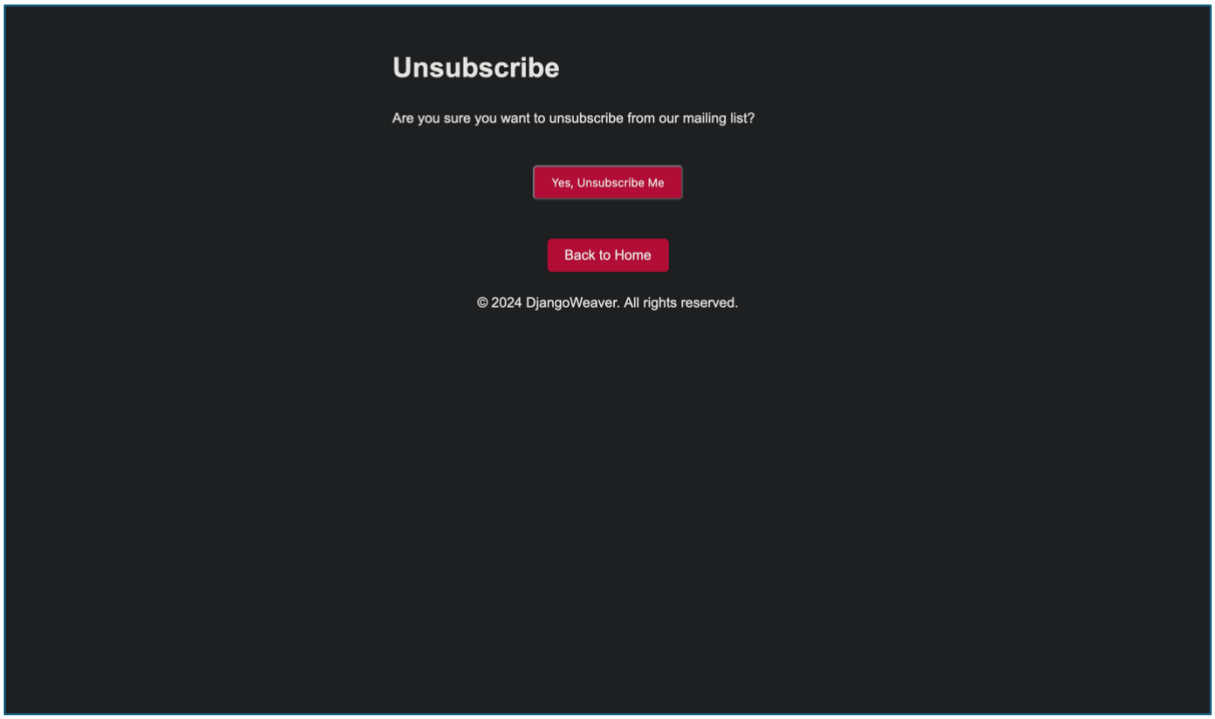
Áreas de personalização e gestão de notificações (PAE)

Com os *links* apresentados anteriormente, é possível a adesão a várias áreas, específicas para cada destinatário das notificações do PAE, isto é conseguido com o recurso ao uso de *tokens*, que são utilizados para identificar cada utilizador.

Remoção da lista de utilizadores subscritos

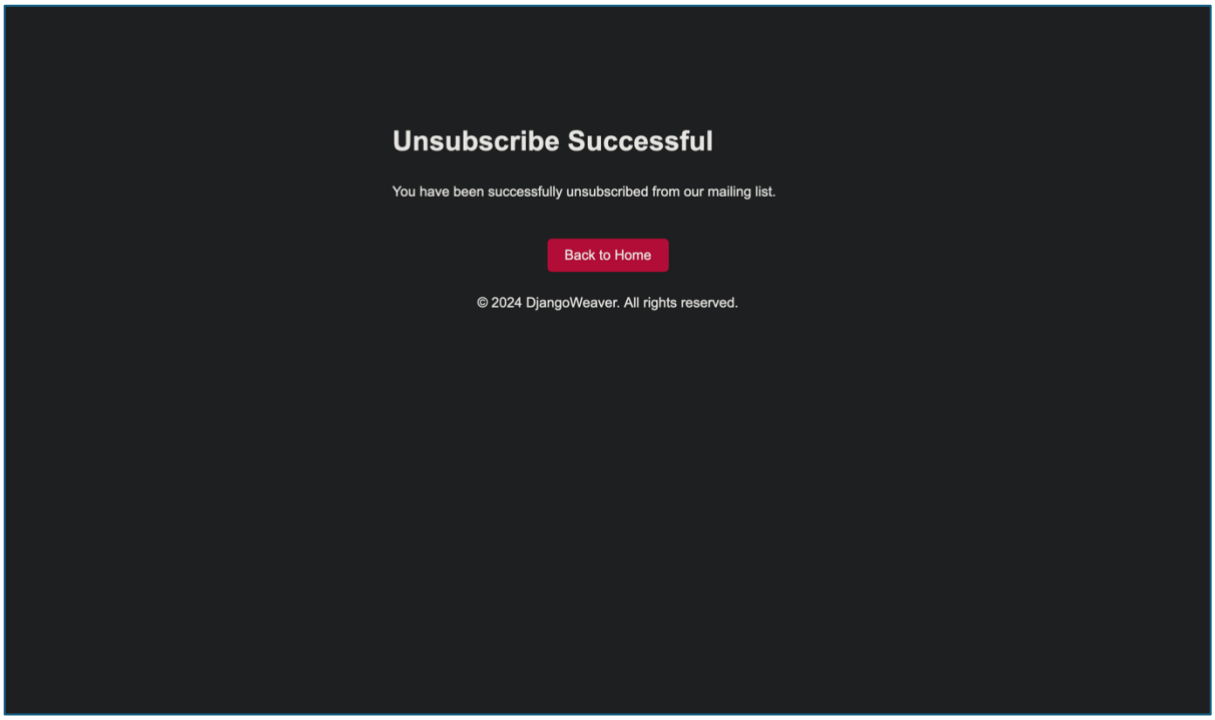
Esta interface é em tudo comum a outras disponíveis para os mais variados serviços disponíveis na Internet. É apresentado ao utilizador um botão que serve de confirmação para a tarefa que se pretende desempenhar.

Figura 62 – DjangoWeaver: (PAE) Remoção da lista de subscritores



Caso o utilizador opte por ter estes e-mails suspensos, ao contrário do que seria normal, estes endereços não são removidos da plataforma, mas sim marcados como inativos. Uma vez que a lista de destinatários é devolvida pelo sistema PAE.

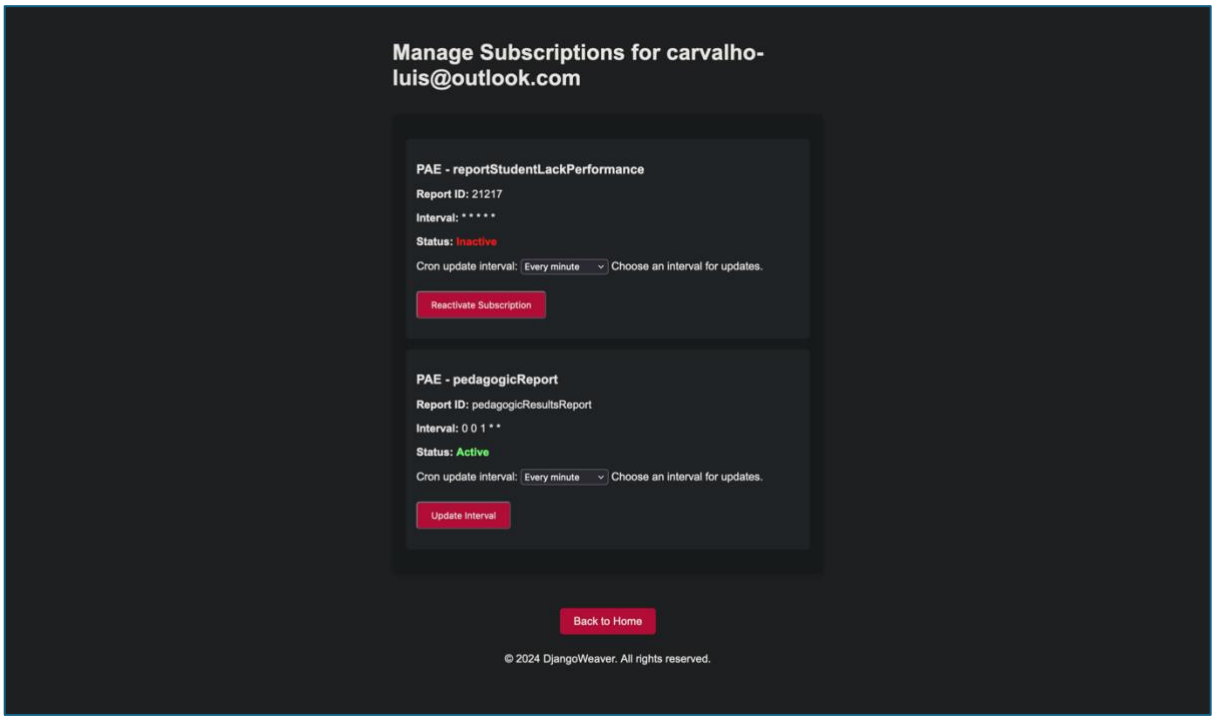
Figura 63 – DjangoWeaver: (PAE) Remoção da lista de subscritores - sucesso



Gestão de subscrições

Neste ecrã, são apresentados ao utilizador as seguintes informações relevantes. Qual o nome do serviço configurado, qual o intervalo de atualização definido para o utilizador, se a subscrição está ativa. Ao utilizador é dada a escolha se pretende atualizar estes dados.

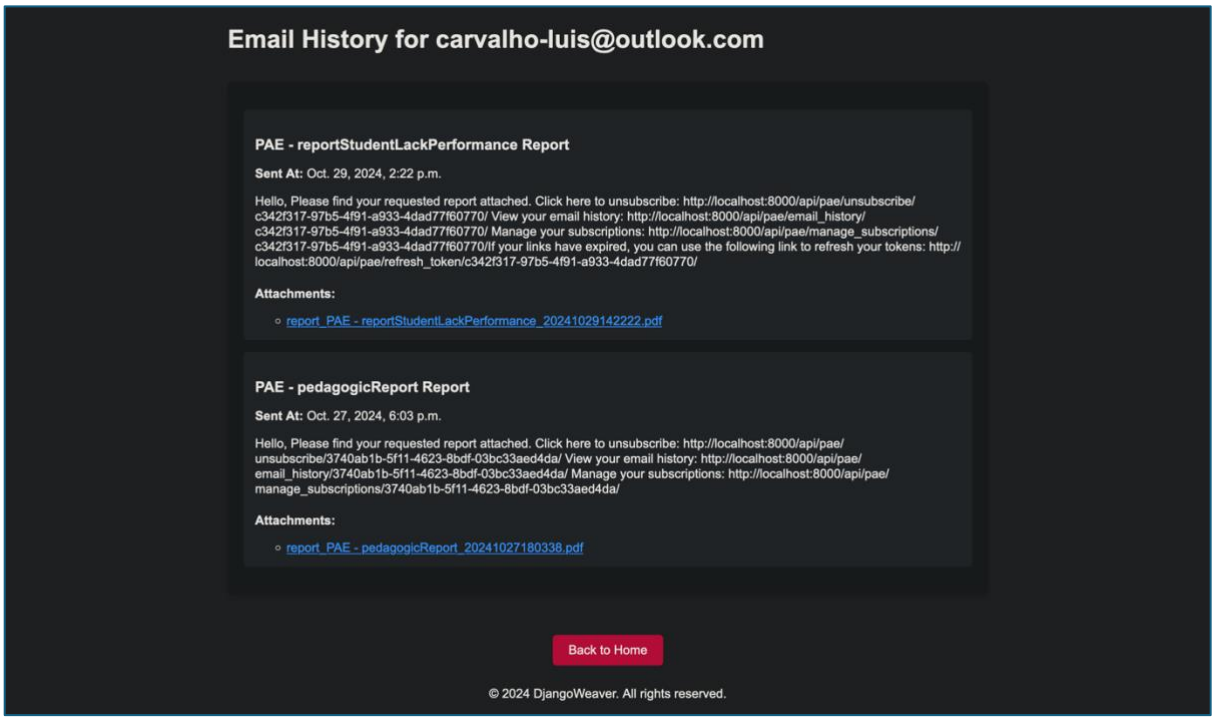
Figura 64 – DjangoWeaver: (PAE) Gestão de serviços subscritos



Histórico de e-mails recebidos

Aqui o utilizador tem a possibilidade de consultar os dados que lhe foram enviados anteriormente, assim, caso ocorra algum problema na entrega ou este tenha extraviado o e-mail, é possível a recuperação e a consulta da informação disponibilizada.

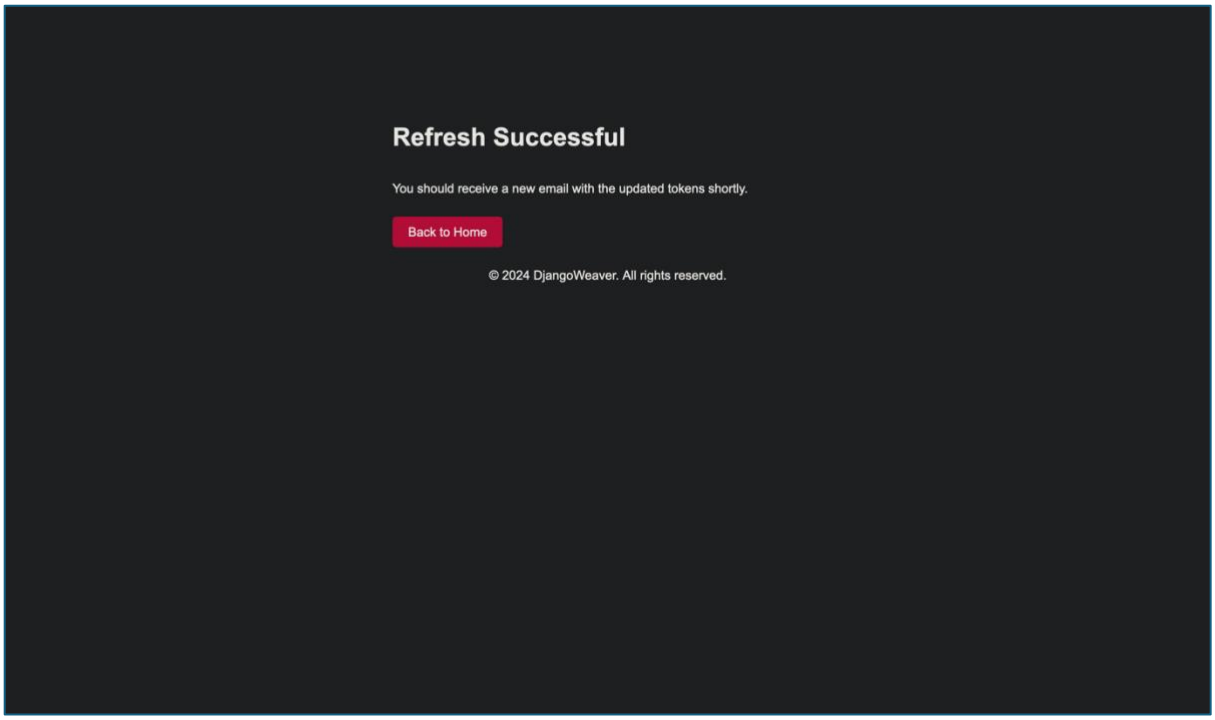
Figura 65 – DjangoWeaver: (PAE) Histórico de e-mails recebidos



Atualização de Tokens

É oferecido também ao utilizador a hipótese de atualizar os seus *tokens*. A necessidade desta funcionalidade prende-se pela questão que estes utilizadores não têm forma de se autenticar diretamente nesta plataforma e como necessidade de segurança, estes *tokens* expiram no decorrer de sete dias.

Figura 66 – DjangoWeaver: (PAE) Pedido de atualização de *tokens* - sucesso



Assim sendo, a plataforma recebe o pedido de renovação destes acessos e um novo e-mail é enviado para o endereço de e-mail associado, o que permite reestabelecer o acesso às restantes funcionalidades.

1.3 Documentação

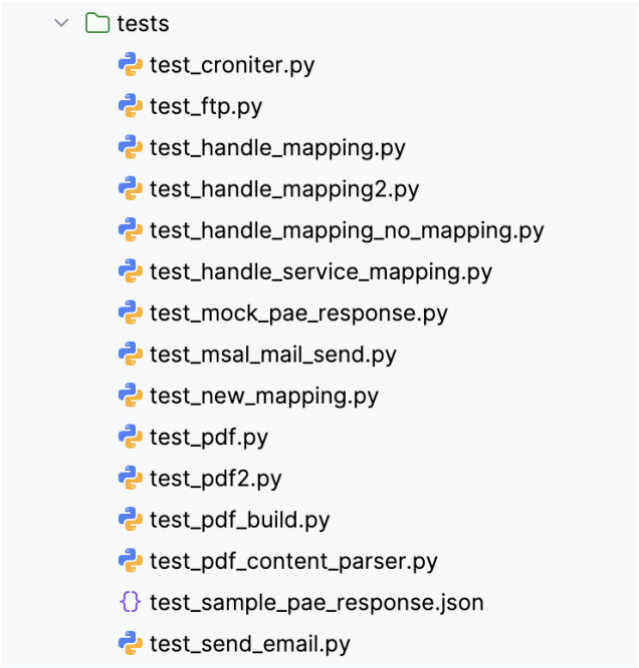
O código-fonte deste desenvolvimento será facultado, em acordo com as normas da instituição, em formato físico, bem como este relatório e outros ficheiros respeitantes ao desenvolvimento da plataforma.

Poder-se-á encontrar também, em alguns ficheiros de desenvolvimento, comentários orientadores do funcionamento esperado da aplicação ou do método respetivo.

2 Testes Realizados

No contexto do desenvolvimento da plataforma foram desempenhados uma serie de testes, sendo que a maioria se pode classificar como testes unitários e testes de integração, quer o objetivo seja o de testar funcionalidades de forma isolada, ou testar a integração com os vários serviços incorporados.

Figura 67 – Estrutura da pasta de testes



Testes de Mapeamento e Manipulação de Dados:

- **test_handle_mapping.py**
- **test_handle_mapping2.py**
- **test_handle_mapping_no_mapping.py**
- **test_handle_service_mapping.py**
- **test_new_mapping.py**

Estes testes focam-se na validação das funções responsáveis pelo mapeamento e manipulação de dados. Eles verificam se o sistema está interpreta e transforma os dados de entrada de forma correta, de acordo com as regras de mapeamento definidas. O teste "no_mapping" serviu para verificar o comportamento do sistema quando não há mapeamento definido e utiliza um algoritmo baseado apenas no tipo de dados e estruturas recebidas.

Tabela 13 – Exemplo de um teste de mapeamento

```
  class TestHandleServiceMapping(unittest.TestCase):
def test_handle_service_mapping(self):
        # Example usage
input_data = {
            "data": [
                {
                    "id": "2cd15e5a-1013-4f23-abd5-45cbdceba59c",
                    "type": "fact",
"attributes": {
                        "body": "Female dogs carry puppies for about nine weeks before birth."
                    }
```

```
            },
            {
                "id": "2cd15e5a-1234-5678-9012-45cbdceba59c",
                "type": "wololo",
                "attributes": {
                    "body": "Cotton candy is a great treat about nine weeks before birth."
                }
            }
        ]
    }

    custom_mapping = {
        'help': "You can replace every value with a matching key from the returning endpoint.'",
        'reset_mapping': False,
        'mail_subject': '',
        'mail_body': 'body',
        'pdf_title': '',
        'pdf_text': 'body'
    }

    expected_result = {
        'help': "You can replace every value with a matching key from the returning endpoint.'",
'reset_mapping': False,
        'mail_subject': '',
        'mail_body': "Female dogs carry puppies for about nine weeks before birth.",
        'pdf_title': '',
        'pdf_text': "Female dogs carry puppies for about nine weeks before birth."
    }

    result = map_json(custom_mapping, input_data)
    # Assert the result
    self.assertEqual(expected_result, result)
```

Testes de Criação e Manipulação de PDF:

- **test_pdf.py**
- **test_pdf2.py**
- **test_pdf_build.py**
- **test_pdf_content_parser.py**

Tabela 14– Teste de geração automática de relatórios com diferentes tipos de dados

```
class DataTypeTestCase(TestCase):
def setUp(self):
self.my_list = {
'text': "Hello",
        'int': 123,
        'float': 45.67,
        'bool': True,
        'null': None,
```

```
            'object': {"key": "value"},
            'list': [1, 2, 3]
        }

    def test_build_auto_pdf_from_content(self):
story = []
        story = build_auto_pdf_from_content(story, self.my_list)
self.assertIsInstance(story, list)
```

Esta série de testes está diretamente associado à geração de relatórios PDF. Estes casos serviram para testar exclusivamente a os outputs produzidos, sem a necessidade de chamar os serviços, permitindo ajustes mais rápidos.

Testes de Comunicação e Integração:

- **test_ftp.py**
- **test_msal_mail_send.py**
- **test_send_e-mail.py**

Estes testes focam nas funcionalidades de comunicação do sistema. O teste FTP verifica a capacidade de transferir arquivos via FTP. Os testes de envio de e-mail asseguram que o sistema pode enviar notificações e relatórios por e-mail corretamente. No caso da biblioteca MSAL (Microsoft Authentication Library), esta foi testada na alteração do processo de autenticação SMTP, tendo-se optado pelo recurso a *passkeys*.

Testes de Agendamento:

- **test_croniter.py**

Os testes aqui efetuados resumiram-se ao processo de cálculo da próxima data válida, com base na data do sistema e o uso de expressões *cron*.

Tabela 15– Teste de cálculo do último dia do mês com base na data do sistema

```
class CronTest(TestCase):
def test_last_day_of_month(self):
  base = datetime(2024, 9, 7)  # Use a fixed date for testing
  cron_expression = '0 0 L * *'
  iterable = croniter(cron_expression, base)

  next_date = iterable.get_next(datetime)
  expected_date = datetime(2024, 9, 30, 0, 0)  # Expected last day of the month

  self.assertEqual(next_date, expected_date)
```

Testes de Integração com o PAE:

- **test_mock_pae_response.py**
- **test_sample_pae_response.json**

Estes *scripts* estão relacionados aos testes de integração com o PAE. Em ambos se simulam respostas do PAE para testar o comportamento do sistema sem depender de efetuar pedidos reais.

Testes Ponta-a-ponta ("end-to-end)

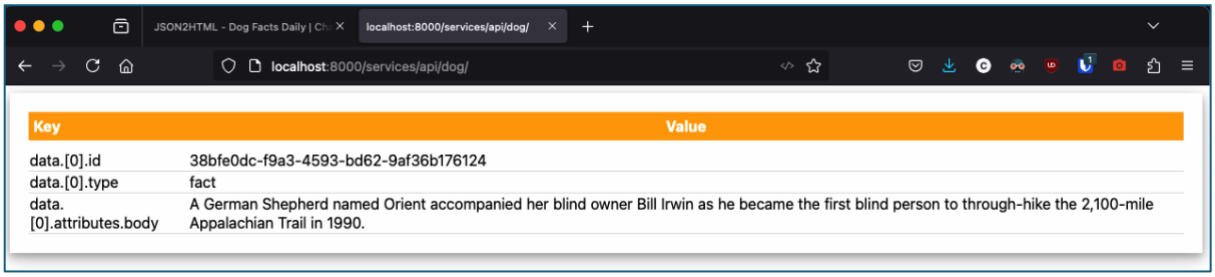
Recorreu-se várias vezes ao longo da evolução do projeto a este tipo de testes, que consiste na testagem exaustiva das funcionalidades do sistema. Isto foi necessário devido ao grau das alterações necessárias ao suporte de funcionalidades específicas do PAE, ou outras que impactaram processos de mapeamento comuns e surgiu a necessidade de garantir que certas alterações não impactavam o restante do processo.

Geração de Elementos HTML

Teste: Criação de tabelas HTML a partir de dados estruturados.

Resultado: HTML gerado corretamente, com estrutura e conteúdo validados.

Figura 68 – Caso de Teste: Geração de elementos HTML

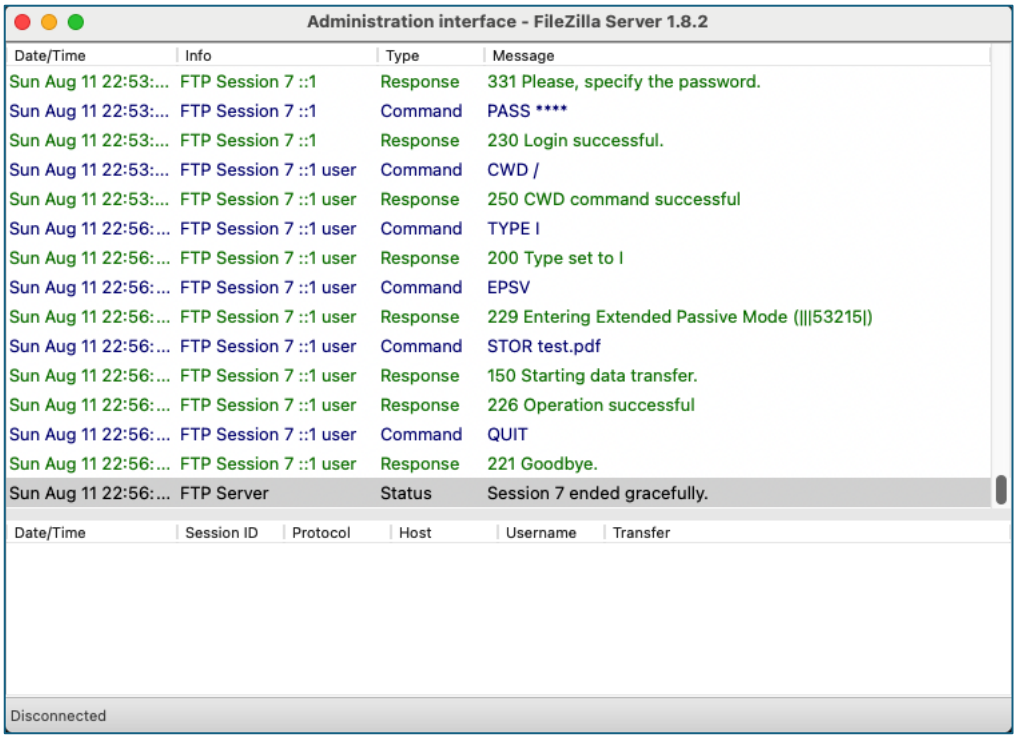


Entrega de Dados via FTP

Teste: Transferência de arquivos para um servidor FTP.

Resultado: Arquivos transferidos com sucesso, verificando integridade e permissões.

Figura 69 – Caso de Teste: Entrega de dados via FTP



Envio de E-mails

Teste: Envio de e-mails com diferentes tipos de conteúdo e anexos.

Resultado: E-mails entregues corretamente, verificando conteúdo e recebimento.

Figura 70 – Caso de Teste: Envio de e-mails

Produção de Ficheiros PDF

Teste: Geração de PDFs com gráficos, tabelas e texto formatado.

Resultado: PDFs criados com layout e conteúdo corretos, validados visualmente e por parsing.

Figura 71 – Caso de Teste: Produção de ficheiros PDF

3 Análise dos Resultados

O desenvolvimento da plataforma resultou num sistema funcional, capaz de atender aos objetivos propostos inicialmente. Os principais resultados alcançados incluem:

A plataforma demonstrou capacidade de consumir dados de diferentes formatos (JSON, XML, CSV e texto) através de APIs, processá-los e distribuí-los em diversos formatos de saída, incluindo JSON, XML, CSV, texto, e-mail, FTP e PDF. Esta funcionalidade atende ao objetivo de integração de sistemas, permitindo a comunicação eficiente entre diferentes plataformas.

O sistema foi capaz de gerar relatórios personalizados com base nos dados recebidos, apresentando-os em formatos acessíveis e interpretáveis, como documentos PDF. As Figuras Figura **60** e Figura **61** do relatório mostram exemplos de relatórios produzidos pela plataforma-

Foram desenvolvidas funcionalidades específicas para suportar a plataforma escolar PAE, incluindo um modelo de geração de relatórios dedicado a capacidade de acesso a utilizadores externos para a gestão e consulta de informações distribuídas pela plataforma.

Os testes realizados, como mostrado nas figuras Figura **57** a Figura **66**, validaram as funcionalidades críticas do sistema, incluindo a geração de elementos HTML, entrega de dados via FTP, envio de e-mails e produção de arquivos PDF.

Como possíveis melhorias listam-se principalmente, o suporte de mais formatos de dados, capacidade de oferecer dados estatísticos sobre o uso e a resposta do sistema, bem como um processo de *logs* completo, capaz de oferecer informação detalhada sobre a execução dos vários serviços e o estado de operação da plataforma. Seria desejável o suporte também de Templates HTML para as saídas de e-mail e de elementos HTTP, permitindo ao utilizador uma maior granularidade na personalização da saída.

As interfaces disponibilizadas aos utilizadores do PAE para a manutenção das suas subscrições deveriam ser extendidas e adaptadas ao resto do sistema. Bem como a apresentação do histórico de notificações deverá ser revista para ser de mais fácil navegação.

CONCLUSÃO

O objetivo principal deste projeto era desenvolver uma plataforma de integração de sistemas robusta e escalável, utilizando as tecnologias Python e Django. Nesta plataforma esperava-se a introdução de funcionalidades que permitissem a entrega de relatórios aos utilizadores da plataforma de forma periódica, bem como à sua disponibilização através de documentos PDF ou via e-mail, nessa ótica é admissível concluir que esse objetivo foi cumprido.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] Django Project, “Django | The web framework for perfectionists with deadlines.,” [Online]. Available: https://www.djangoproject.com/. [Acedido em 02 04 2024].

[2] M. N. Rizza e K. Evans, “IDC MarketScape: Worldwide SaaS and Cloud-Enabled Medium-Sized Business ERP Applications 2024 Vendor Assessment,” IDC MarketScape, 2024.

[3] C. Pettey, “Moving to a Software Subscription Model,” Gartner, 18 05 2018. [Online]. Available: https://www.gartner.com/smarterwithgartner/moving-to-a-software-subscription-model. [Acedido em 23 05 2024].

[4] SAP, “SAP Integration Suite pricing,” [Online]. Available: https://www.sap.com/products/technology-platform/integration-suite/pricing.html.

[5] MuleSoft, “MuleSoft Anypoint Platform Pricing,” [Online]. Available: https://www.mulesoft.com/anypoint-pricing. [Acedido em 29 4 2024].

[6] Zato, “Support,” [Online]. Available: https://zato.io/en/services/support. [Acedido em 13 07 2024].

[7] market.us, “Global System Integration Services Market Size - Industry Segment Outlook, Market Assessment, Competition Scenario, Trends and Forecast 2024-2033,” market.us, 2024.

[8] Zato, “Zato GNU Affero General Public License v3.0,” [Online]. Available: https://github.com/zatosource/zato/blob/main/LICENSE.txt. [Acedido em 06 05 2024].

[9] E. Stewart, “What is System Integration? Types, Components and Examples,” EM360Tech, 24 06 2024. [Online]. Available: https://em360tech.com/tech-article/system-integration. [Acedido em 24 06 2024].

[10] J. O. Grady, System Integration, CRC Press, 2019, p. 3.

[11] M. Reddy, API Design for C++, Elsevier, 2011, p. 1.

[12] Nivelics, “Automation: The Path to Operational Efficiency in Large Companies,” [Online]. Available: https://www.nivelics.com/en/blog/automation-operational-efficiency. [Acedido em 4 11 2024].

[13] J. Manyika e K. Sneader, “AI, automation, and the future of work: Ten things to solve for,” McKinsey Global Institute, 1 06 2018. [Online]. Available: https://www.mckinsey.com/featured-insights/future-of-work/ai-automation-and-the-future-of-work-ten-things-to-solve-forhttps://www.mckinsey.com/featured-insights/future-of-work/ai-automation-and-the-future-of-work-ten-things-to-solve-for. [Acedido em 23 06 2024].

[14] L. H. Etzkorn, Introduction to Middleware: Web Services, Object Components, and Cloud Computing, CRC Press, 2017, pp. 4-5.

[15] D. C. Luckham, Event Processing for Business: Organizing the Real-Time Enterprise, John Wiley & Sons, 2011, p. 38.

[16] IBM, "What is ETL (extract, transform, load)?," [Online]. Available: https://www.ibm.com/topics/etl. [Accessed 5 11 2024].

[17] Microsoft, “Tarefas em segundo plano,” Microsoft, 06 04 2024. [Online]. Available: https://learn.microsoft.com/pt-pt/azure/architecture/best-practices/background-jobs. [Acedido em 5 11 2024].

[18] T. Point, “Django - MVT Architecture,” [Online]. Available: https://www.tutorialspoint.com/django/django_mvt.htm. [Acedido em 1 11 2024].

[19] P. Bakare, “Django | Model-View-Template (MVT),” dev.to, 15 10 2024. [Online]. Available: https://dev.to/mrcaption49/django-model-view-template-mvt-5h5p. [Acedido em 2 11 2024].

[20] V. Nishant, “Django MVT Architecture: A Fresh Take on Classic MVC,” Ask Python, 20 08 2020. [Online]. Available: https://www.askpython.com/django/django-mvt-architecture. [Acedido em 4 11 2024].

[21] E. R. B. B. K. S. Eric Freeman, Head First: Design Patterns, O'Reilly, 2004, p. 24.

[22] Z. Mahmood, Cloud Computing: Methods and Practical Approaches, Springer Science & Business Media, 2013, p. 103.

[23] Orderful, “Enterprise Application Integration: A Comprehensive Guide,” Orderful, [Online]. Available: https://www.orderful.com/blog/enterprise-application-integration. [Acedido em 1 11 2024].

[24] M. F. Gholami, F. Daneshgar, G. Beydoun e F. Rabhi, “Challenges in migrating legacy software systems to the cloud — an empirical stud,” *Information Systems,* vol. 67, pp. 100-113, 2017.

[25] V. Andrikopoulos, T. Binz, F. Leymann e S. Strauch, “How to adapt applications for the Cloud environment Challenges and solutions in migrating applications to the Cloud,” *Computing,* vol. 95, p. 507, 30 12 2012.

[26] Preet Kaur, “Embracing the Future: How the API First Approach is Revolutionizing Software Development,” Moesif, 25 06 2024. [Online]. Available: https://www.moesif.com/blog/technical/api-development/Embracing-the-Future-How-the-API-First-Approach-is-Revolutionizing-Software-Development/. [Acedido em 3 11 2024].

[27] J. Lindner, “API Statistics: Businesses Prioritize Integrations, Security for Digital Growth,” WifiTalents, 6 8 2024. [Online]. Available: https://wifitalents.com/statistic/api/. [Acedido em 4 11 2024].

[28] M. Loukides e S. Swoyer, “Microservices Adoption in 2020,” O'Reilly, 15 07 2020. [Online]. Available: https://www.oreilly.com/radar/microservices-adoption-in-2020/. [Acedido em 4 11 2024].

[29] M. Michalowski, “55 Cloud Computing Statistics for 2024,” Spacelift, 27 03 2024. [Online]. Available: https://spacelift.io/blog/cloud-computing-statistics. [Acedido em 4 11 2024].

[30] B. Fitzgerald e K.-J. Stol, “The Future of Software Development Methods,” em *The Routledge Companion to Management Information Systems,* Routledge, 2018, pp. 125-137.

[31] J. Bratincevic e D. L. Giudice, “The Rise Of Application Generation Platforms,” Forrester, 27 05 2024. [Online]. Available: https://www.forrester.com/blogs/the-rise-of-application-generation-platforms/. [Acedido em 14 08 2024].

[32] T. H. Davenport e R. Ronanki, “Artificial Intelligence for the Real World,” Harvard Business Review, 1 1 2018. [Online]. Available: https://hbr.org/2018/01/artificial-intelligence-for-the-real-world. [Acedido em 3 11 2024].

[33] M. R. DeLisi e C. Howley, “Gartner Forecasts Worldwide Low-Code Development Technologies Market to Grow 20% in 2023,” [Online]. Available: https://www.gartner.com/en/newsroom/press-releases/2022-12-13-gartner-forecasts-worldwide-low-code-development-technologies-market-to-grow-20-percent-in-2023.

[34] KPMG, “81% das empresas consideram low-code um fator crítico de sucesso,” KPMG, [Online]. Available: https://kpmg.com/pt/pt/home/media/press-releases/2024/02/low-code-fator-critico-de-sucesso.html. [Acedido em 2 11 2024].

[35] KPMG; Koen De Loose; Nicolas Dubois, “What are the main data regulations impacting organizations?,” [Online]. Available: https://kpmg.com/be/en/home/insights/2021/11/rc-what-are-the-main-data-regulations-impacting-organizations.html. [Acedido em 2 11 2024].

[36] Gartner, “Magic Quadrant™ for Integration Platform as a Service (iPaaS),” Gartner, 2024.

[37] Gartner, “Magic Quadrant™ for API Management,” Gartner, 2024.

[38] Joshua Erney, “What is DataWeave?,” Mulesoft, [Online]. Available: https://developer.mulesoft.com/tutorials-and-howtos/dataweave/what-is-dataweave-getting-started-tutorial/. [Acedido em 2 11 2024].

[39] SAP, “SAP Integration Suite pricing,” [Online]. Available: https://www.sap.com/products/technology-platform/integration-suite/pricing.html. [Acedido em 1 11 2024].

[40] SAP, “Monetize APIs,” [Online]. Available: https://help.sap.com/docs/integration-suite/sap-integration-suite/monetize-apis-f6ab15146ac74bc4be6d9e9a9fef4b26. [Acedido em 1 11 2024].

[41] SAP, “What is SAP Integration Suite?,” [Online]. Available: https://help.sap.com/docs/integration-suite/sap-integration-suite/what-is-sap-integration-suite. [Acedido em 2 11 2024].

[42] Zato, “Tutorial,” Zato, [Online]. Available: https://zato.io/en/docs/3.2/tutorial/01.html. [Acedido em 1 11 2024].

[43] Zato, “Architecture,” Zato, [Online]. Available: https://zato.io/en/docs/3.2/arch/index.html. [Acedido em 1 11 2024].

[44] SendGrid, “Pricing and Plans | SendGrid,” [Online]. Available: https://sendgrid.com/en-us/pricing. [Acedido em 2 11 2024].

[45] SendGrid, “ Deliverability Insights,” [Online]. Available: https://sendgrid.com/en-us/solutions/email-api/deliverability-insights. [Acedido em 2 11 2024].

[46] Mailchimp, “Recursos do Mailchimp,” [Online]. Available: https://mailchimp.com/pt-br/features/. [Acedido em 5 11 2024].

[47] Mailchimp, “Design an Email with the New Builder,” [Online]. Available: https://mailchimp.com/help/design-an-email-new-builder/.

[48] Mailchimp, “Marketing API,” [Online]. Available: https://mailchimp.com/developer/marketing/api/automation-email/. [Acedido em 5 11 2024].

[49] Amazon, “O que é latência de rede?,” Amazon, [Online]. Available: https://aws.amazon.com/pt/what-is/latency/. [Acedido em 4 11 2024].

[50] Cisco, “What Is High Availability?,” Cisco, [Online]. Available: https://www.cisco.com/c/en/us/solutions/hybrid-work/what-is-high-availability.html#~high-availability-clusters-explained. [Acedido em 4 11 2024].

[51] Alexsoft, “Unstructured Data: Examples, Tools, Techniques, and Best Practices,” Alexsoft, 12 05 2023. [Online]. Available: https://www.altexsoft.com/blog/unstructured-data/. [Acedido em 05 11 2024].

[52] Chris Nwankwo, “Unstructured Data: Examples, Techniques, and Challenges,” insight7, [Online]. Available: https://insight7.io/unstructured-data-examples-and-tools-for-generating-insights/. [Acedido em 4 11 2024].

[53] Amazon, “O que é visão computacional?,” Amazon, [Online]. Available: https://aws.amazon.com/pt/what-is/computer-vision/. [Acedido em 4 11 2024].

[54] P. Ziegler e K. R. Dittrich, “Data Integration - Problems, Approaches, and Perspectives,” em *Conceptual Modelling in Information Systems Engineering*, Springer, 2007, pp. 41-42.

[55] M. Lenzerini, “Data Integration: A Theoretical Perspective,” em *PODS '02: Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, New York, 2002.

[56] S. Overflow, “Technology | 2024 Stack Overflow Developer Survey,” Stack Overflow, [Online]. Available: https://survey.stackoverflow.co/2024/technology/. [Acedido em 05 11 2024].

[57] IBM, “O que são message brokers?,” IBM, [Online]. Available: https://www.ibm.com/br-pt/topics/message-brokers. [Acedido em 02 11 2024].

[58] “First Steps with Celery,” Celery, [Online]. Available: https://docs.celeryq.dev/en/stable/getting-started/first-steps-with-celery.html#choosing-a-broker. [Acedido em 5 11 2024].

[59] Amazon, “Qual é a diferença entre o RabbitMQ e o Redis?,” Amazon, [Online]. Available: https://aws.amazon.com/pt/compare/the-difference-between-rabbitmq-and-redis/. [Acedido em 05 11 2024].

[60] Redis, “Community Edition,” Redis, [Online]. Available: https://redis.io/docs/latest/get-started/. [Acedido em 5 11 2024].

[61] Redis, “Redis FAQ,” Redis, [Online]. Available: https://redis.io/docs/latest/develop/get-started/faq/. [Acedido em 5 11 2024].

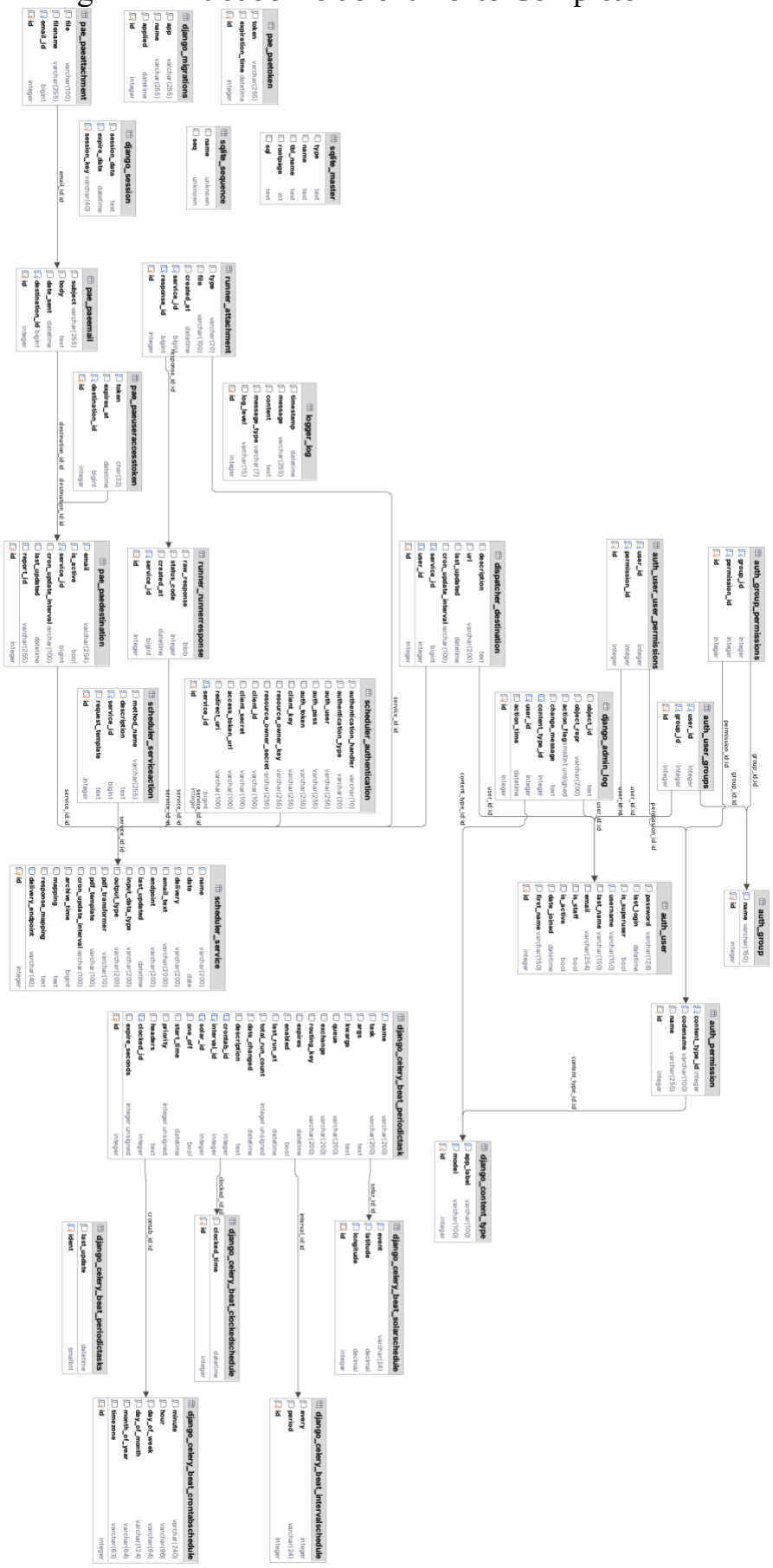
[62] Mozilla, “Django Tutorial Part 11: Deploying Django to production,” Mozilla, [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Deployment. [Acedido em 5 11 2024].

ANEXOS

Anexo 1 – “Definição de Requisitos”

Anexo 2 – “Casos de Uso”

Anexo 3 – “Diagrama Entidade-Relacionamento Completo”



Anexo 4 – "Diagrama de Classes Completo"

Anexo 5 – Exemplo de implementação - “Chamada periódica de Serviços”

```
  @shared_task()
  def check_tasks():
      Log.objects.create(log_level="1", message="RUNNER: process started", content="",
message_type="info")

      services = Service.get_all_periodic_expired()
      amount = services.count()

      Log.objects.create(log_level="1", message=f'RUNNER: tasks found: {amount}', content="",
message_type="info")

      for service in services:
          input_path, output_path = get_or_create_output_and_archive_folders(service.name)

          if os.path.exists(input_path):
              files = os.listdir(input_path)
              if not files:
                  print(f"RUNNER: No files found in './{service.name}', making new request")
              else:
                  is_update = handle_update_interval(service)
                  if not is_update:
                      continue
          else:
              print(f"RUNNER: No folder found for '{service.name}', creating new folder and making
request")

          response = service.send_request()
          RunnerResponse.objects.create(service=service, raw_response=response.content,
status_code=response.status_code)

          if response:
              timestamp = timezone.now().strftime("%Y%m%d-%H%M%S")
              save_response_to_file(response, service, timestamp)

              handled_data = handle_response(response, service)
              if handled_data.status_code != 500:
                  save_handled_data_to_file(handled_data, service, timestamp)

              print(f"RUNNER: Processed service '{service.name}'")
          else:
              print(f"RUNNER: Failed to get response from '{service.endpoint}', status code:
'{response.status_code}'")
      print('RUNNER: process finished')
```

Anexo 6 – Exemplo de implementação - “Lógica de envio de e-mails”

```
  def send_email(service: Service, recipients: Iterable, attachment_paths: Iterable, file_binary_data:
bytes) -> bool:
  …
  if service.output_type == 'EMAIL_ATTACHMENT' or service.output_type == 'PDF':
    for filepath in attachment_paths:
        filename = filepath.split("/")[-1]
        with open(filepath, "rb") as f:
            email.add_attachment(f.read(), maintype='application', subtype='octet-stream',
```

```
filename=filename)
try:
    with smtplib.SMTP_SSL('smtp.gmail.com', 465) as server:
        server.ehlo()
        server.login(os.getenv("SENDER_EMAIL"), os.getenv("EMAIL_PASSWORD"))
        server.send_message(email)
        server.close()
    return True
except Exception as e:
    print(f"Error sending email: {str(e)}")
    return False
```

Anexo 7 – Exemplo de implementação - "Definição de estratégias da classe "Authentication""

```
class Authentication(models.Model):
    def _get_authentication_strategy(self):
        strategies = {
            'NONE': NoAuthStrategy(),
            'BASIC': BasicAuthStrategy(),
            'TOKEN': TokenAuthStrategy(),
            'DIGEST': DigestAuthStrategy(),
            'OAUTH1': OAuth1AuthStrategy(),
            'OAUTH2': OAuth2AuthStrategy(),
            'PAE': PAEAuthStrategy(),
        }

        if self.authentication_handler == 'DEFAULT':
            return strategies.get(self.authentication_type)
        else:
            return strategies.get(self.authentication_handler)
```

Anexo 8 – Exemplo de implementação - "Tratamento de vários tipos de resposta com base no tipo de input"

```
def handle_response(response: HttpResponse, service: Service):
    input_type = service.input_data_type
    output_type = service.output_type

    if input_type == output_type:
        return response

    if input_type == 'JSON':
        return handle_json_response(response, output_type, service)

    elif input_type == 'XML':
        return handle_xml_response(response, output_type, service)

    elif input_type == 'CSV':
        return handle_csv_response(response, output_type, service)

    elif input_type == 'TEXT':
        return handle_text_response(response, output_type, service)
```

Anexo 9 – Exemplo de implementação - "Lógica para o popular de templates HTML para a entrega de dados"

```
def json2html(byte_data: bytes) -> HttpResponse:
  json_data = json.loads(byte_data.decode())
  flat_data = flatten(json_data)
  first_key = next(iter(flat_data))

  if first_key != "data":
     flat_data = {'data': flat_data}

  rendered_html = render(None, '../templates/html/default.html', flat_data)
  return rendered_html
```

Anexo 10 – Exemplo de implementação - "Algoritmo genérico para o tratamento de dados JSON"

```
def transform_json(input_data, template: dict):
  output_data = {}

  for key, value in template.items():
     if key.startswith("custom_"):
        output_data[key] = value
     elif isinstance(value, dict):
        output_data[key] = transform_json(input_data.get(key), value)
     elif isinstance(value, list):
        output_data[key] = []
        for sub_template in value:
           if isinstance(sub_template, dict):
              output_data[key] = [transform_json(item2, sub_template) for item2 in
input_data.get(key)]
           else:
              output_data[key] = input_data.get(key)
     elif value == "display":
        output_data[key] = input_data.get(key)
     elif value == "hide":
        pass
     else:
        pass

  return output_data
```

Anexo 11 – Exemplo de implementação - "Lógica para a geração de ficheiros PDF"

```
def generate_pae_pdf(json_input:dict, service: Service):
  packet = io.BytesIO()
  doc = SimpleDocTemplate(packet, pagesize=A4)

  story = generate_pae_story(json_input)
```

```
    doc.build(story)
    packet.seek(0) # Move to the beginning of the buffer

    # Read PDF template
    pdf_path = service.pdf_template.path or "./templates/pdf/default_pae.pdf"
    existing_pdf = PdfReader(open(pdf_path, "rb"))
    new_pdf = PdfReader(packet)

    # Create a new PDF with the added text
    output = PdfWriter()

    # Determine the number of pages in both the template and the new PDF
    num_existing_pages = len(existing_pdf.pages)
    num_new_pages = len(new_pdf.pages)

    # Iterate over all the pages in the new generated PDF
    for i in range(num_new_pages):
        # Get the corresponding template page, or if there are more new pages than template pages,
just reuse the last template page
        template_page = existing_pdf.pages[min(i, num_existing_pages - 1)]
        generated_page = new_pdf.pages[i]

        generated_page.merge_page(template_page) # Merge the generated page onto the template
        output.add_page(generated_page) # Add the merged page to the output

    output_buffer = io.BytesIO()
    output.write(output_buffer)
    output_buffer.seek(0)

    return output_buffer.getvalue()
```

Anexo 12 – “Diagrama de Classes Completo”