FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Predictive Caching in API Mangement

António Vicente Teixeira Gonçalves de Brito

VERSÃO DE TRABALHO

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Orientador: Miguel Pimenta Monteiro, FEUP

Orientador: Ricardo Castanheira Pereira, Deloitte

21 de julho de 2021

Resumo

Numa era em que a Transição Digital vem assumindo cada vez mais relevância em todas as indústrias, o crescimento das arquiteturas baseados em serviços e microsserviços, o aumento de APIs, e plataformas em *cloud*, e a competitividade das organizações entre si acompanham esse efeito. O tempo de chegada ao mercado e a flexibilidade dos seus sistemas tornam-se fatores essenciais para superar a concorrência. É neste contexto que a camada de integração assume cada vez mais um papel decisivo para o sucesso das organizações, em que uma forma rápida e otimizada de integração pode trazer inúmeras vantagens competitivas.

A Deloitte Portugal, enquanto consultora em projetos de tecnologia, ambiciona tornar-se (ainda mais) uma referência neste sentido, tendo como palavra do dia "Inovação". Neste sentido surge o novo conceito abordado neste projeto: Predictive Caching in API Management.

Predictive Caching in API Management consiste na integração de serviços baseada em métodos preditivos. Aqui, será aplicado tendo em conta dois objetivos: fazer cache do conteúdo estático, ou seja, com uma baixa taxa de alteração ao longo do tempo; previsão de pedidos, identificação de fluxos de integração e pedidos, utilizando *machine learning*.

O Mulesoft assumiu o papel principal como plataforma de Integração. O Postman foi utilizado para fazer testes, simulando o papel do cliente, efetuando pedidos e recebendo respostas. Por sua vez, Python foi a linguagem de programação principal para análise de dados e *machine learning*, onde se recorreu a redes neuronais para algoritmo de treino e previsão.

O projeto foi aplicado e testado em dois casos de uso: Deloitte Project e DigiB Project. Deloitte project, um ambiente de teste, criado pela equipa com o objetivo de testar algumas funcionalidades e avaliar a eficácia dos algoritmos. DigiB Project já como um projeto real da Deloitte Portugal e Deloitte Netherlands. Aqui foi dada a oportunidade de utilizar dados do cliente, tendo em conta todas as medidas de privacidade e confidencialidade, com o objetivo de obter alguns resultados e identificar de que forma este novo conceito pode ser aplicado num ambiente como o projeto da DigiB.

O projeto inicou com a formação em Mulesoft, seguindo a criação do ambiente de teste, seleção, análise e tratamento de dados (tanto no Deloitte Project como DigiB), desenvolvimento dos algoritmos e integração com os casos de uso.

Terminou com a fase de obtenção e discussão de resultados, onde ficou provado que esta nova forma de integração é possível e pode acrescentar valor às organizações.

Abstract

Nowadays, Digital Transformation has a crucial role in every industries, and services or micro-services oriented software architectures, APIs and cloud follow that growth. Time to Market and flexibility of systems became crutial points in order to superate the competition. In this field, Integration Layer is extremely important, and a quick, easy and optimized way of doing integration can bring a lot of benefits.

Deloitte Portugal, as Tech consultant, aims to be a reference in this field, and "Inovation"is one of the keywords. Here, a new concept is born: Predictive Caching in API Management.

Predictive Caching consists in doing integration based on predictive algorithms. This project has two main goals: save static information in cache (data that doesn´t change a lot trough time); predict, using machine learning, request flows and patterns.

Mulesoft is the main integration platform. Postman was used for testing and to simulate the clients role, making requests and receiving responses. Python was used as main programming language and data analyses, where neuronal networks was the algotithm chosen for training and predict models.

The project was applied and tested in two use cases: Deloitte Project and DigiB project. Deloitte Projects was a testing environment which its main purpose was to test some features and evaluate the algorithms accuraccy. DigiB Project was a real project developped by Deloitte Portugal and Deloitte Netherlands. It was given the oportunity to use real data (clients data), respeting all of security and confidentiality policies, in order to have some results and understand how this concept (Predictive Caching) can be useful to a project like DigiB.

The project began with formation and certification in Mulesoft, followed by the development of the test environment (Deloitte Project). The next step was selection, analysis and data treatment (Deloitte Project and DigiB), algorithms development and integration with the use cases.

The project finished with a results discussion, where it was proved that this way of doing integration is possible and can bring value to every organization.

Agradecimentos

Quando comecei o meu percurso no ensino superior, a prioridade era terminar o mestrado integrado em 5 anos. Chegar ao final com esse objetivo cumprido, com a realização do estágio de dissertação numa empresa como a Deloitte e a sensação de que me foi dado todo o apoio e todas a oportunidades necessárias para ter sucesso neste período de tempo é algo que dificilmente imaginaria e que me deixa extremamente concretizado. Isto não foi um trabalho individual, foi um trabalho de equipa, com pessoas a quem tenho muito que agradecer e que sem elas este percurso seria muito mais complicado.

Primeiramente, quero agradecer aos orientadores Miguel Pimenta Monteiro, da FEUP, e Ricardo Pereira, da Deloitte, por acreditarem em mim, e por todo o apoio , disponibilidade e sentido crítico extremamente importante ao longo da dissertação. Contribuíram com opiniões e indicações fundamentais para o sucesso do projeto.

Quero agradecer à Faculdade de Engenharia da Universidade do Porto e a todos os docentes com quem tive oportunidade de aprender ao longo destes 5 anos, que me deram todas as condições para aprender, errar e crescer.

Agredecer também a todo o Deloitte Portugal, principalmente à equipa de PSUR, pela oportuinidade que me proporcionaram, pela forma fantástica como me integraram e disponibilidade que me deram. Aqui, um agradecimento especial ao Ricardo Pereira novamente, Emídio Silva, ao Francisco Pires e David Teixeira por todo o apoio que me deram. Agredecer ainda à equipa da DigiB Portugal, composta pelo Gonçalo Soares, Pedro Gala e Xavier Marques e Deloitte Netherlands, pelo abertura que demonstraram, integração no projeto e por tudo o que me ensinaram. A oportunidade de trabalhar com uma equipa internacional neste estágio e aplicar o projeto neste contexto foi algo que vai marcar todo o meu percurso profissional.

Quero agradecer a toda a minha família, especialmente aos meus pais, António Brito e Emília Teixeira, irmão e esposa, Pedro Brito e Ana Pinto, por todo o apoio e aconselhamento dado em todos os momentos. São modelos para mim, que me obrigam a ser melhor todos os dias.

A todos os meus amigos, tanto do Porto, que tive oportunidade de conhecer no início do ensino superior, e de Arouca, que já perdi a conta aos anos que estão comigo,e à equipa de Futsal do Futebol Clube de Arouca. Sem eles tudo, isto não seria possível , e mesmo que fosse, não seria levado com a maior das felicidades.

Quero deixar ainda uma palavra de agradecimento a todas as pessoas que não foram aqui mencionadas, e que fizeram parte do meu percurso.

Obrigado a todos, por tudo. É um capítulo que termina, mas outro que começa e pelo qual estou muito ansioso.

António Vicente Teixeira Gonçalves de Brito

”If you are the smartest person in the room,
then you are in the wrong room”

Confucius

Conteúdo

Lista de Figuras

Lista de Tabelas

Abreviaturas e Símbolos

AMS	Aplication Management Services
API	Aplication Programming Interface
API-LED-CONN	API-LED-Connectivity
DIY	Do-It-Yourself
ESB	Enterprise Service Bus
FaaS	Function as a Service
HTTP	Hypertext Transfer Protocol
IaaS	Infraestructure as a Service
IT	Information Technology
JSON	JavaScript Object Notation
ML	Machine Learning
PaaS	Platform as a Service
REST	Representational State Transfer
SaaS	Software as a Service
SOA	Service-Oriented Architecture
SOAP	Simple Object Access Protocol
XML	extensible markup language

Capítulo 1

Introdução

1.1 Contexto

Hoje em dia as arquiteturas de integração assumem um papel crucial nas organizações. Cada vez mais os clientes começam a ter arquiteturas híbridas com sistemas em *cloud* e nos seus próprios servidores. Uma arquitetura de integração faz a ponte entre todas as aplicações, criando padrões de gestão de API's (Aplication Programming Interface), monitorização, reforço de políticas de segurança, centralização de toda a comunicação, etc. O desafio das plataformas em *cloud* faz com que existam limitações para o cliente, como longos tempos de chamada para *datacenters* que estão em países e geografias longínquas, licenciamento destes sistemas, que exigem um pagamento por utilização (custo por invocação).

Atualmente, com o objetivo de tornar os sistemas mais flexíveis, escaláveis e com melhor desempenho, a aposta em arquiteturas orientadas a serviços e microsserviços tem sido cada vez mais forte. Estes sistemas destacam-se por estas três características:

- **Desempenho:** capacidade de um sistema responder às necessidades de uma forma rápida, eficiente e com poucos recursos.
- **Escalabilidade:** capacidade de um sistema crescer e gerir o aumento de carga.
- **Flexibilidade:** capacidade de um sistema evoluir e adaptar-se face às mudanças do negócio.

1.2 Motivação

O principal desafio consiste em construir uma *framework* em Mulesoft com base em *machine learning* que permita que um algoritmo aprenda com a invocação das API's e ao mesmo tempo consiga determinar quais delas podem ser "cached". Desta forma conseguir-se-ia reduzir tempos de chamada e custos de invocação.

Um segundo desafio pretendido neste estágio consiste em criar um algoritmo capaz de reconhecer padrões de chamadas de APIs, e a partir desse padrão antecipar as mesmas, automatizando esse processo, tornando-o muito mais rápido. Exemplo Prático: Quando é chamada a API x, existe

um padrão de chamadas das API's y e z. O objetivo passa por reconhecer este padrão, e uma vez que seja invocada a API x, a camada intermédia de integração invoca automaticamente as API's y e z, tornando os seus resultados disponíveis antecipadamente.

1.3 Problema

1.3.1 Definição do Problema

Com base na revisão bibliográfica realizada ao longo deste trabalho, e com a evolução no mercado, podem-se retirar os seguintes pontos que nos ajudam a definir o problema:

- Aumento das aplicações nas organizações.
- Aposta cada vez mais forte nas plataformas *cloud* .

Devido ao facto de muitas aplicações serem implementadas em *cloud*(possivelmente *datacenters* longínquos), estes dois pontos estão diretamente ligados.

O tipo de custo nas utilizações destas plataformas, com base na utilização das mesmas (*pay per use*) faz com que cada invocação tenha um custo associado, existindo um potencial de redução.

Por outro lado, o aumento de aplicações nas organizações faz com que a camada de integração assuma uma importância enorme para uma boa experiência de utilização, segurança e monitorização. É importante referir que com o crescimento das aplicações, cresce também o número de API's num sistema, o que aumenta a dificuldade da sua gestão de uma forma autónoma e automatizada, baseando as soluções em pensamentos humanos.

Pretende-se provar que há uma nova forma de integração aplicando as tecnologias preditivas na execução dos fluxos de integração.

1.3.2 Benefícios esperados com a solução do problema

1.3.2.1 Solução Proposta

A solução passa por desenvolver uma *framework* de caching, com base em algoritmos preditivos (*machine learning*). Esta solução pode separar-se em dois tópicos:

- Fazer cache de pedidos com maior incidência sem alteração de payload.
- Identificar preditivamente os fluxos subsequentes após uma chamada.

Numa fase inicial, o foco principal é o primeiro objetivo (fazer cache de pedidos com maior incidência sem alteração de payload). Para entender melhor o que se pretende, podemos considerar dois tipos de dados: estáticos e não estáticos. Os dados estáticos são os que não sofrem mudanças constantes ao longo do tempo (ou seja, baixa taxa de alteração). Os não estáticos, por outro lado, apresentam uma elevada taxa de mudança. Neste contexto, pretende-se identificar quais os dados estáticos e fazer cache dos mesmos em vez de fazer novos pedidos aos *end-systems* cada vez que se

pretende aceder à informação contida nos mesmos. Desta maneira, é possível otimizar invocações, tempos de chamada, evitar o processamento de dados já conhecidos, etc, trazendo várias vantagens à organização.

Numa segunda fase, o objetivo prende-se em identificar preditivamente os fluxos subsequentes após uma chamada. Ou seja, se após uma chamada, houver um padrão de pedidos ou de outras chamadas, pretende-se identificar o mesmo padrão e antecipar esse efeito.

1.3.2.2 Benefícios esperados

Com esta solução, um dos benefícios que esperamos ter é a otimização do uso dos sistemas em *cloud*. Quando a informação se mantém imutável e está guardada nos nossos *datacenters*, deixa de ser necessária a invocação à *cloud* para obter a resposta a dar ao cliente. Desta forma é possível reduzir custos e tempos de chamada e otimizar a utilização destes sistemas.

O segundo grande benefício que esperamos tirar deste projeto é a possibilidade de dar uma melhor experiência de utilização e uma forma mais rápida de integração dos sistemas.

Num primeiro ponto de vista, quando temos a informação em cache, é esperado que, aquando um pedido, a resposta ao cliente é muito mais rápida, visto que o tempo de chamada à *cloud* é retirado.

Por outro lado, prever e identificar fluxos de pedidos por parte dos clientes torna possível obter a resposta aos mesmos antes dos pedidos serem efetuados. Desta forma, no momento em que o pedido é feito, a resposta é imediata.

Resumidamente, esta solução possibilita uma utilização otimizada deste tipo de sistemas, promove uma melhor experiência de utilização e torna o fluxo de informação mais rápido dentro de uma organização.

1.3.3 Objetivos principais a atingir e contribuições desta dissertação

Neste projeto, os principais objetivos a atingir são:

- Fundamentos Mulesoft
- Desenhar *framework* de caching
- Modelar aplicação de *machine learning*
- Fazer cache de pedidos com maior incidência sem alteração de payload
- Identificar preditivamente os fluxos subsequentes após uma chamada.
- Construir relatório análise e sugestões de caching.

Capítulo 2

Aspetos Modernos das Aplicações Empresariais

2.1 Caracterização do domínio de intervenção

2.1.1 Contexto

Com a transformação digital e o avanço das tecnologias disponíveis no mercado, muitas empresas têm optado por soluções que permitem uma gestão mais eficiente dos seus processos, com mais agilidade e menor investimento.

Uma das grandes dificuldades das organizações prende-se em conectar as diferentes áreas de serviço. Entre outros problemas, a não integração destas soluções (CRM, ERP, SCM, BI, etc) pode provocar uma perda nas informações, ficando presas na respetiva área. Por exemplo: uma queda nas vendas poderia estar relacionada a uma queda de sistemas que suportam vendas, mas que por não estarem integradas estas informações não são utilizadas para a tomada de decisões.

Inicialmente, as organizações optaram por uma arquitetura monolítica, em que todas as aplicações e serviços de uma organização se concentram numa plataforma. No entanto, problemas como a alta dependência do código, aumento da complexidade, pouca flexibilidade, dificuldade de alteração das funcionalidades disponíveis promovidas por esta arquitetura, revelaram-se cruciais neste tipo de abordagem.

Com o objetivo de contrariar esses efeitos, optou-se por apostar em novas arquiteturas, nomeadamente microsserviços e arquiteturas orientada a serviços (SOA). Para um bom funcionamento destas arquiteturas, a integração impõe-se como parte fundamental deste desenvolvimento.

Posto isto, a revisão bibliográfica consistiu principalmente no estudo de arquiteturas de software como Microsserviços e SOA, gestão de APIs, AMS, computação em *cloud* e *machine learning*, visto como um método de automatização.

2.1.2 Arquitetura Monolítica

A arquitetura monolítica [1] é um tipo de arquitetura em que todas as aplicações e serviços de uma organização se concentram numa plataforma. Com esta abordagem, os serviços partilham os mesmos recursos, como capacidade de processamento, memória, bases de dados, etc.

Naturalmente tem algumas vantagens como a facilidade de implementação e a simplicidade no desenvolvimento.

No entanto, as desvantagens têm assumido cada vez mais destaque. Ao longo do tempo, o sistema vai ficando bastante complexo, dificultando a manutenção e atualização das funcionalidades já existentes. Todo o desenvolvimento se baseia numa única tecnologia, impedindo a utilização das melhores ferramentas para cada serviço. A alta dependência no código e aumento de tamanho ao longo do tempo, exigindo mais capacidade de memória e processamento no sistema, são outras desvantagens cruciais deste tipo de arquitetura. Na figura 2.1 é possível visualizar um exemplo de arquitetura monolítica.

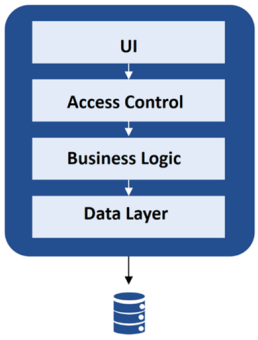

Figura 2.1: Arquitetura Monolítica

2.1.3 Microsserviços

Microserviços [2] é uma arquitetura em que uma aplicação é dividida e constituída por aplicações, componentes ou serviços de menor dimensão e independentes. Neste tipo de arquitetura cada microsserviço é responsável por uma determinada função, e pode ser desenvolvido com a sua própria linguagem de programação e modelo de dados.

Em relação à estrutura monolítica tradicional, este tipo de arquitetura traz inúmeras vantagens, tais como:

- Independência e escalabilidade de cada serviço.
- Facilidade na implementação de novas aplicações.
- Facilidade na deteção e correção de erros.
- Cada serviço tem o seu método de implementação (linguagem de programação, dados, etc).

Naturalmente as organizações também conseguem retirar inúmeras vantagens deste tipo de arquitetura. Permite às empresas alterar, remover ou adicionar facilmente funcionalidades de cada aplicação sem interferir com a restante estrutura do sistema.

Como cada microsserviço é independente, torna-se possível escolher as melhores ferramentas e tecnologias para cada função.

No entanto, apesar de cada serviço ser desenvolvido separadamente dos restantes, a comunicação com as restantes aplicações é essencial, e aqui, a integração assume um papel importante garantindo essa mesma comunicação e acessibilidade de todas as aplicações.

Com este tipo de arquitetura, cada serviço disponibiliza as suas funcionalidades para os restantes utilizadores. O código desenvolvido e a maneira como é implementado não é relevante para aceder e reutilizar as aplicações do sistema. Comunicam através de APIs, eventos ou mensagens. As organizações podem implementar os próprios serviços, garantindo a manutenção dos mesmos, ou aceder a funcionalidades de serviços disponibilizados por outras entidades. Na figura 2.2 encontra-se um exemplo de uma arquitetura baseada em microsserviços.

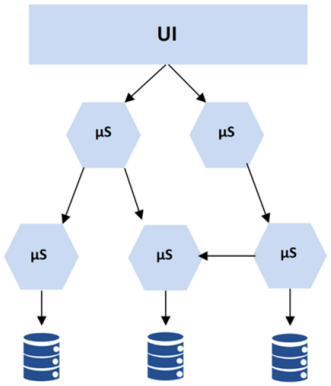

Figura 2.2: Microserviços

2.1.4 SOA (Arquitetura Orientada a Serviços)

Uma arquitetura orientada a serviços [3] divide o seu sistema em diferentes serviços e componentes mais pequenos, que comunicam entre si. Os serviços comunicam através de interfaces (APIs) e uma das grandes diferenças em relação aos microsserviços prende-se na existência de protocolos de comunicação comuns entre aplicações.

Nesta arquitetura, cada serviço pode ser implementado com a própria linguagem de programação, estruturas de dados, etc, independente dos restantes, com o objetivo de oferecer as funcionalidades desejadas. Todos os serviços são disponibilizados e acedidos através de protocolos de comunicação para troca e leitura de dados.

As grandes vantagens deste tipo de arquitetura baseiam-se na facilidade e rapidez com que podem ser acrescentadas novas funcionalidades e aplicações às organizações. Como cada serviço é desenvolvido independentemente dos restantes, a facilidade de implementação de novas aplicações

promove também uma mais rápida chegada ao mercado. O facto de cada serviço comunicar através de interfaces, e a implementação do mesmo ser irrelevante para os utilizadores, a operabilidade dos mesmos é garantida independentemente do local de implementação (próprias infraestruturas, ou externas, com recurso a *cloud* por exemplo). Podemos observar na figura 2.3 um exemplo de uma arquitetura orientada a serviços.

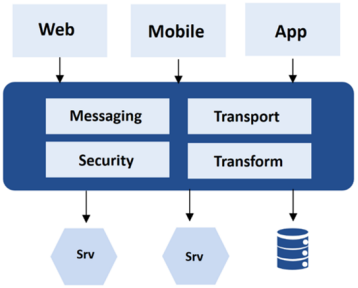

Figura 2.3: Arquitetura Orientada a Serviços

2.1.4.1 ESB (*Enterprise Service Bus*)

Neste contexto, entra um importante elemento neste tipo de arquitetura: ESB. Segundo [4], um ESB pode ser interpretado como um conjunto de regras e princípios para a integração de inúmeras aplicações, permitindo a troca de dados, independentemente da tecnologia utilizada em cada um deles. O principal conceito de ESB indica-nos que a integração entre diferentes aplicações pode ser promovida através de uma camada de comunicação (*bus*), em que as aplicações não comunicam diretamente entre si, mas sim com esta camada. Assim, o ESB é responsável pela gestão de pedidos, roteamento e garantir a acessibilidade dos diferentes serviços. Este conceito foi crescendo com o facto das organizações procurarem alternivas a integrações ponto-a-ponto.

Desta forma, um ESB aumenta a agilidade e escalabilidade dos sistemas e reduz o tempo de chegada ao mercado. Representa também uma forma fácil e rápida de acrescentar novos intervenientes no sistema, bem como a atualização dos intervenientes já existentes com novas funcionalidades. Tudo isto, sem serem necessárias alterações nos restantos componentes.

Um ESB deve garantir que o sistema atinge uma apropriada granularidade de serviços, isto é, não deve ser composto nem por grandes serviços, difíceis de gerir e atualizar, nem por uma quantidade exagerada de pequenos serviços. Este método promove a reutilização e uma gestão facilitada de todos os seus componentes.

A transformação de dados e transporte são princípios a ter em conta nesta arquitetura. Garantir que cada serviço envia e recebe a informação correta a tempo útil, deve ser um dos principais focos deste modelo.

Além disso, com o crescimento do sistema, garantir a compatibilidade dos serviços com diferentes versões, a segurança e monitorização dos mesmos vão-se revelando pontos fundamentais para o bom funcionamento das organizações.

Alguns dos softwares mais utilizados no mercado para esta função são:

- Oracle Service Bus;
- Mule ESB;
- WSO2 Enterprise Service Bus;
- IBM DataPower Gateway.

2.1.5 API-Led Connectivity

API-Led Connectivity [5] representa uma nova de conexão de dados de aplicações através de APIs. É uma nova abordagem, utilizado com o Mulesoft e considerado um dos grandes passos da arquitetura SOA. Conforme podemos ver figura 2.4, é dividida em três camadas:

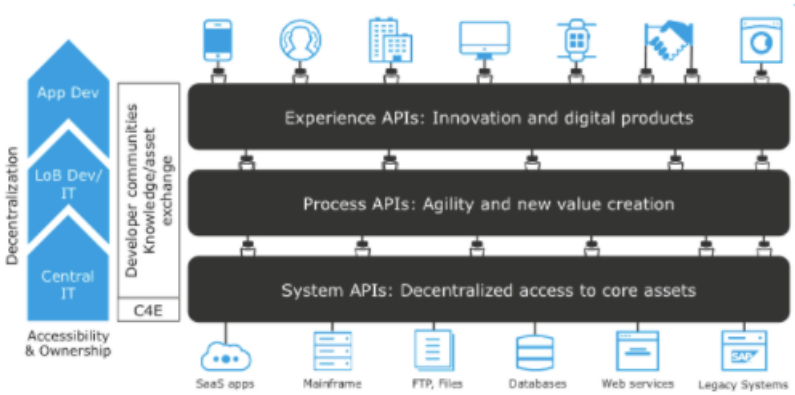

Figura 2.4: API-Led Connectivity - Arquitetura Tipo

- ***Experience API's***: responsável por reconfigurar os dados para serem consumidos com mais facilidade pelo público-alvo. Tudo isto a partir de uma fonte de dados comum, em vez de configurar integrações ponto a ponto separadas para cada canal.
- ***Process API's***: Utilizadas quando se pretende obter informações de diferentes aplicações (internas ou externas). Esta camada permite relacionar toda a informação que dos diferentes serviços e tirar conclusões que facilita a tomada de decisão. É aqui que realmente se acrescenta valor ao sistema.
- ***System API's***: Responsáveis por aceder aos sistemas de registo e expor esses dados, evitando a complexidade dos mesmos. Uma vez criadas, os utilizadores podem aceder e reutilizar as API's sem profundo conhecimento do restante sistema subjacente.

2.1.6 Arquitetura Orientada a Eventos

Uma arquitetura orientada a eventos [6] é uma metodologia de desenvolvimento de soluções baseada na geração, receção e processamento de eventos. Estes eventos fluem entre componentes independentes, sendo agnósticos entre si, ou seja, quem envia não sabe o destino e vice-versa .

Neste modelo é utilizada uma quantidade mínima de dependências, o que torna uma boa opção para as arquiteturas atuais.

Podemos considerar um evento como uma ocorrência ou mudança de estado num determinado sistema. Neste contexto é importante distinguir um evento de uma notificação, sendo esta última, apenas uma mensagem enviada pelo sistema que informa a ocorrência de um evento.

2.1.6.1 Modelos

De acordo com [7] este tipo de arquitetura pode ser dividida em dois modelos: publicação/subscrição ou transmissão de eventos.

Modelo Publicação e Subscrição: Com este modelo, os sistemas que têm interesse num determinado negócio subscrevem os sistemas relacionados com o mesmo. Após a ocorrência de um evento, os subscritores recebem a notificação relativa ao mesmo. Neste tipo de modelo há a possibilidade de notificar todos os subscritores simultaneamente após a ocorrência de um evento.

Trasmissão de eventos: Neste modelo, os eventos são gravados num registo de eventos. Aqui não há a necessidade de subscrição, e os utilizadores podem ler diretamente da transmissão. Tipos de transmissão de eventos:

- **Processamento de fluxo do evento:** usa uma plataforma de transmissão, para ingerir e processar eventos ou transformar o seu fluxo. Esse método pode ser usado para detectar padrões significativos em fluxos de eventos.
- **Processamento de evento simples:** é quando um evento aciona imediatamente uma ação no consumidor.
- **Processamento de evento complexo:** requer que um consumidor processe uma série de eventos para detectar padrões.

2.1.7 Gestão de APIs

Posto isto, surge outro ponto fundamental para as organizações, a gestão de APIs. Uma API, *aplication programming interface*, é o conjunto de funções e procedimentos que especificam como determinados componentes de um software devem interagir. A gestão de APIs [8] é o processo que engloba a criação, disponibilização, documentação e análise de APIs num ambiente seguro. Com o crescimento constante dos sistemas e aplicações dentro das organizações, uma boa gestão vem-se revelando cada vez mais fundamental. Através desta gestão, as organizações garantem que as APIs criadas são publicadas, disponibilizadas e seguras.

As soluções apresentadas no mercado para gestão de APIs podem apresentar diferentes funcionalidades [9], no entanto as fundamentais são:

- API Design - Permite aos utilizadores a possibilidade de desenvolver e publicar as suas APIs, assim como documentação correspondente, políticas de segurança, capacidade, etc.

- API Gateway - Funciona como "*gatekeeper*", que garante que as políticas de segurança são cumpridas quando do acesso à API.
- API Store - Funciona como *marketplace* de APIs, onde os utilizadores podem subscrever a mesma, suporte e acesso à comunidade, etc.
- API analytics - Permite a monitorização da API (histórico, utilização, estado, etc).

2.1.8 Padrões de Integração

Quando temos o objetivo de integrar diversas aplicações ou serviços, os padrões de integração [10] definem como essa mesma integração deverá feita.

2.1.8.1 Base de Dados partilhada

Este tipo de integração utiliza a partilha da mesma base de dados entre dois ou mais sistemas para efetuar a conecção entre os mesmos. Neste contexto, cada aplicação deve manter a ligação a uma base de dados partilhada com a informação que será integrada.

Por exemplo: a atualização de uma tabela na mesma base de dados pode provocar uma ação de atualização nas restantes tabelas que serão disponibilizadas para outras aplicações.

2.1.8.2 API Gateway

A API Gateway é o ponto de entrada para a comunição com os restantes serviços e aplicações do sistema. Quando uma aplicação pretende aceder a funcionalidades disponibilizadas por outras aplicações, não necessita de saber com que quem deve comunicar. Apenas precisa de saber que existe um *gateway* que irá fazer a ponte entre os serviços pretendidos. É também nesta camada que são feitas as autorizações, cumpridas as políticas de segurança, logs de pedidos, respostas, etc.

2.1.8.3 Messaging

O tipo de arquitetura orientada a serviços impulsionou este tipo de integração, efetuada através de trocas de mensagens. Neste caso, o responsável pela gestão, definição da estrutura, roteamento e transformação é o ESB.

Message Routing

Define como são efetuadas as trocas de mensagens entre os sistemas garantindo que as mesmas são enviadas para o servidor correto. Em sistemas mais complexos, podem existir diversos padrões de roteamento, tais como:

- Message Filter - Permite a filtragem de certas mensagens.
- Scatter-Gather - Permite a troca síncrona de mensagens. Por exemplo, a mesma mensagem ser enviada para diversos serviços ao mesmo tempo.

- Message Aggregator - Permite a agregação de diferentes mensagens.

Message Transformation

Quando se pretende estabelecer a conexão entre diferentes sistemas, é necessária a "tradução"das mensagens dos diferentes sistemas. Message Transformation é responsável por essa tarefa.

Por exemplo, um determinado sistema pode fazer um pedido em JSON, mas o sistema "destino"apenas recebe pedidos no formato XML. Esta camada é responsável por esta transformação JSON/XML e XML/JSON para tornar possível a comunicação entre os dois serviços.

Message Management

Neste tipo de integração, baseado em troca de mensagens, a gestão das mesmas é fundamental, principalmente em sistemas mais complexos. Quando existe uma grande quantidade de trocas de dados, é necessário uma solução capaz de corrigir erros, gerir tráfico, histórico de mensagens,etc. Esta camada é a grande responsável por essa tarefa.

2.1.9 AMS (*Application Management Services*)

Com a aposta cada vez mais forte em arquiteturas baseadas em microsserviços ou SOA, cresce também a quantidade, capacidade e complexidade das aplicações que as organizações utilizam, o que logicamente pode provocar um sobrecarregamento dos departamentos informáticos das mesmas. Verificando-se este cenário, acaba por existir uma queda nas aplicações procuradas pelas empresas porque deixa de haver capacidade para a manutenção e gestão de todo o sistema. Assim, em vez de inovação e crescimento, o foco concentra-se, quase exclusivamente, na gestão das plataformas já existentes, o que não é desejável.

É neste sentido que cresce este conceito de AMS [11]. AMS consiste em procurar apoio externo (*outsourcing*) para a gestão e suporte das próprias aplicações. Com isto, uma organização deixa de ter a responsabilidade de manutenção, monitorização e atualização das próprias aplicações, delegando isso para a instituição contratada para esse efeito. Para as organizações traz inúmeras vantagens, como o aumento da eficiência das aplicações, evita o sobrecarregamento do seu departamento informático (caso exista), aumenta a satisfação dos utilizadores, etc.

E quando existe um problema nos serviços, a empresa "mãe"não pára, evita o descontentamento de funcionários e quedas de produção, enquanto o problema é resolvido por outrem. Mesmo para pequenas empresas, com departamentos curtos, e sem recursos suficientes para garantir toda a oferta que desejam, esta solução pode ser bastante viável. Quando a própria equipa não é suficiente para gerir a base de dados e aplicações disponíveis, torna-se possível a contratação desse serviço fora da organização, mantendo o funcionamento desejado.

2.2 *cloud*

Computação em *cloud* [12] consiste no desenvolvimento de aplicações, servidores, armazenamento de dados, entre outros, em *datacenters* remotos. Ou seja, em vez de comprar e manter *datacenters* e servidores físicos, torna-se possível aceder a serviços tecnológicos, capacidade computacional, armazenamento de dados, utilizando um provedor de *cloud*, tal como IBM, Amazon, Microsoft, Google, etc.

As organizações utilizam este tipo de computação para diversas funcionalidades como: *backup* de dados, *email*, *desktops* virtuais, desenvolvimento de software, análise *big data*, etc.

Algumas das grandes vantagens deste tipo de tecnologia são:

Agilidade

A *cloud* oferece acesso a uma grande variedade de tecnologias para facilitar o processo de inovação. É possível utilizar recursos conforme a necessidade, como capacidade de computação, armazenamento, base de dados, IoT, *machine learning*, análises de dados, etc.

Economia de Custo

A *cloud* permite que a troca das despesas de capital (*datacenters*, servidores físicos etc.) por despesas variáveis e pagar apenas pela IT consumida. Por outras palavras, permite um pagamento por utilização, o que provoca um potencial custo de redução ao longo do tempo, que pode ser atingido com otimizações de uso da *cloud* .

Rápida Implementação Global

Com este tipo de computação, é possível ampliar as atividades para novas regiões geográficas e implementar globalmente em pouco tempo. Por exemplo, a AWS (Amazon) tem infraestruturas em todo o mundo, o que permite a implementação de aplicações em vários locais físicos com facilidade. Aproximar os aplicações dos utilizadores finais reduz a latência e melhora a experiência de utilização.

2.2.1 Tipos de Computação em *cloud*

Podemos considerar 4 grandes tipos de computação em *cloud* :

Infraestructure as a Service (IaaS)

O IaaS contém os componentes básicos da IT na *cloud* . Normalmente, o IaaS oferece acesso a recursos de rede, computadores (virtuais ou em hardware dedicado) e espaço de armazenamento de dados. O IaaS oferece o mais alto nível de flexibilidade e gestão sobre os recursos de IT. Exemplo: AWS EC2, Rackspace, Google Compute Engine (GCE), Digital Ocean.

Platform as a Service (PaaS)

Com o PaaS, não há a necessidade de gestão de hardware, sistemas operativos, etc. Sendo assim, o foco passa pela implementação e gestão das aplicações. Dessa forma, o sistema torna-se

mais eficiente, pois não existe a preocupação de aquisição de recursos, planeamento de capacidade e manutenção de software. Exemplo: AWS Elastic Beanstalk, Heroku, Windows Azure, Force.com, OpenShift, Apache Stratos.

Software as a Service (SaaS)

O SaaS oferece um produto completo, executado e gerido pelo provedor de serviços. Na maioria dos casos, quando é mencionado SaaS, correspondem às aplicações finais (como e-mail baseado na web). Com uma oferta de SaaS, não há preocupações com a manutenção dos serviços e infraestruturas. O foco passa apenas pela utilização das aplicações. Exemplos: BigCommerce, Google Apps, Salesforce, Dropbox, MailChimp, ZenDesk, DocuSign, Slack.

A figura 2.5 explica, de uma forma visual, as principais diferenças entre estes tipos de computação em *cloud*.

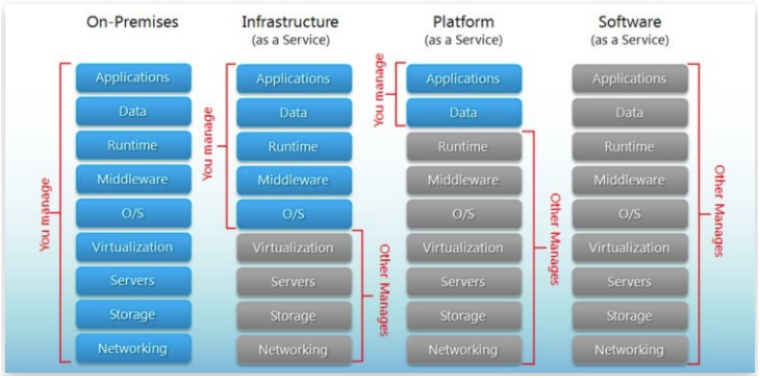

Figura 2.5: Tipos Computação cloud: On-Premisses vs IaaS vs PaaS vs SaaS

Function as a Service (FaaS)

FaaS [13] é um tipo de computação que permite aos utilizadores desenvolver, executar e gerir funcionalidades de um software sem a necessidade de gestão e manutenção dos serviços, softwares e infraestruturas. Estas funcionalidades podem ser chamadas como resposta a eventos (por exemplo). Por essa razão é um tipo de computação que tem acompanhado o crescimento dos microserviços e arquiteturas orientadas a eventos. Exemplos: AWS (Lamda), Google cloud Function.

2.2.2 Custos

No entanto, o uso desta plataforma implica custos, e este é um dos campos principais que se pretende otimizar. Como referido em cima, as *clouds* permitem que as organizações substituam as despesas físicas e permanentes, como servidores físicos e *datacenters* por despesas variáveis, mediante a taxa de utilização. Isto significa que quanto mais otimizado for este processo de acesso à *cloud*, menor é o custo incorrido com a mesma. Portanto, há dois grandes pontos que estão diretamente relacionados com estes custos: número de invocações e tempos de chamada. Neste último ponto, é importante referir que invocações a *clouds* que correm em geografias distantes

provocam tempos de chamada mais longos e consequentemente um maior custo. Consideremos uma instância como um servidor numa *cloud* que corre uma determinada aplicação, a figura 2.6 mostra as diferenças de preço entre 3 instâncias iguais, excetuando a localização do *datacenters*: Iowa (EUA), Hong Kong e Frankfurt (Alemanha). Este cálculo foi efetuado baseado nos custos da Google cloud [14].

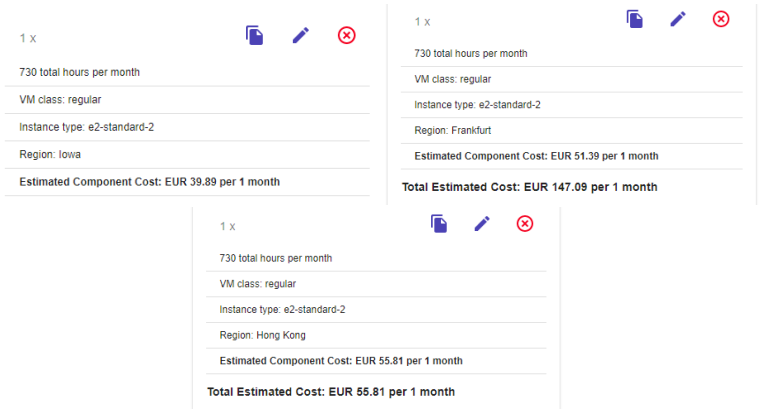

Figura 2.6: Custo Instância Iowa, Frankfurt e Hong Kong

2.2.3 *Machine Learning*

Machine Learning (ML) [15] é um ramo da inteligência artificial baseado nas aplicações que aprendem com os dados e melhoram a sua performance ao longo do tempo e do treino sem a necessidade de programação para isso.

Em ML os algoritmos são treinados com base em grandes quantidades de dados com o objetivo de detetar padrões e antecipar decisões. Quanto melhor for o treino desses algoritmos, mais viáveis serão essas soluções.

É uma área que tem crescido muito ao longo dos anos e atualmente é visível em praticamente tudo, desde músicas que um determinado *website* nos recomenda ouvir, publicidade nos é apresentada, etc. Em ramos como a saúde, também é cada vez mais utilizado, como análise de imagem que ajuda a identificar tumores por exemplo, e noutras áreas, como condução automática, etc.

O crescimento de *big data*, novas tecnologias de computação e maior capacidade dos analistas de dados em desenvolver novos algoritmos são fatores que contribuem para um crescimento cada vez maior do *Machine Learning*.

2.2.3.1 Como Funciona

Existente quatro grandes passos para desenvolver uma aplicação baseada em *Machine Learning*:

1. **Selecionar e preparar os dados de treino**:

Os dados de treino são, tal como o nome indica, a base do treino do algoritmo. É importante ter entradas, e saídas para cada entrada. Com base nos padrões descobertos, o algoritmo determina uma saída para uma nova entrada. Há um erro associado a cada treino do algoritmo, e o objetivo é ir diminuído esse valor à medida que o treino é mais intensivo. Podemos separar em 2 subconjuntos: treino (unicamente para treino do algoritmo) e de avaliação (para testar e avaliar a qualidade do algoritmo).

Portanto é muito importante ter os dados bem organizados e preparados porque são fundamentais para uma boa qualidade de treino.

2. **Escolher um algortimo**

   O algoritmo representa os processos sequenciais que com base numa determinada entrada (problema) determina uma saída (solução). Neste caso, com os dados de treino, o algoritmo torna-se capaz de aprender com os mesmos tornando-se mais eficiente. Existem 2 tipos de dados:

   Labeled - Dados que contêm uma descrição, rótulo, etc. Por exemplo: esta imagem está rotulado porque contém um gato numa foto.

   Unlabeled - Dados sem qualquer tipo de rótulo ou identificação. Geralmente são só fotografias, vídeos, tweets, raio-x, etc.

   Alguns tipos de algoritmos são:

   Regression algorithms - utilizados para prever um valor de uma variável dependente com base no valor da variável independente. É utilizado para perceber correlações entre dados.

   Decision trees - utilizado para classificar dados com base num conjunto de fatores. Por exemplo apostar numa determinada equipa com base na percentagem de vitórias, golos sofridos, etc.

   Instance-based algorithms - classifica dados determinando a probabilidade dos mesmos pertencerem a um determinado grupo.

   Clustering algorithms - identifica grupos através de padrões e características em comum e atribui um dos grupos a um determinado conjunto de dados. Isto é feito sem conhecimento prévio de cada grupo ou das suas características.

   Association algorithms - utiliza relações entre dados idênticas a "se...então"(*association rules*) para classificação.

   Neuronal Networks - utiliza uma rede de cálculos aplicados a uma determinada entrada para identificar a sua saída, onde é atribuída uma probabilidade.

3. **Treino do Algoritmo**

   O treino do algoritmo é um processo iterativo. São realizados vários testes com os dados de treino, e comparadas as saídas que o algoritmo indica com as saídas expectáveis. A cada

teste, dependendo do erro associado às saídas obtidas, são atualizados os pesos e constantes do algoritmo com o objetivo de torná-lo mais eficiente.

No fim deste processo, com resultados aceitáveis, o algoritmo treinado é considerado o modelo.

4. **Uso e evolução do Modelo**

   O último passo é utilizar o modelo, acrescentando novos dados para melhorar a eficiência e validade do modelo ao longo do tempo.

2.2.3.2 Métodos de *Machine Learning*

Existem 3 métodos de *Machine Learning*:

- ***Supervised Learning***

  Supervised Machine Learning [16] utiliza *labeled data* para o treino e depois aplica em novos dados. O treino envolve uma crítica que indica se a função está correta ou não. Neuronal Networks é um dos vários algoritmos que utilizam este método.

  Aqui é criado o algoritmo utilizando dados de treino que contêm entradas e a saída desejada. Após o teste, surge a "supervisão"com o resultado desejado e após essa verificação o algoritmo é ajustada.

  É um método que requer menos dados comparativamente com os restantes o que facilita o treino. A figura 2.7 mostra o funcionamento deste método:

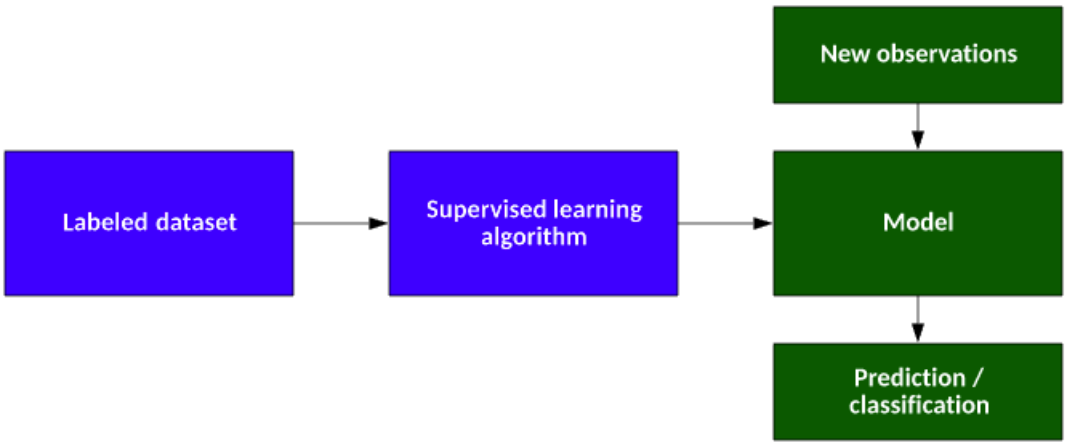

Figura 2.7: Supervised Learning Algorithm

- ***Unsupervised Learning***

  Unsupervised Learning [17] utiliza *unlabeled data* para o treino. Utiliza os dados e com base nas semelhanças e padrões que descobre divide os dados em subconjuntos, sem qualquer intervenção humana. Pode ser útil para descobrir padrões ou anomalias que de outra forma não seriam descobertos. A figura 2.8 mostra a forma como unsupervised learning funciona:

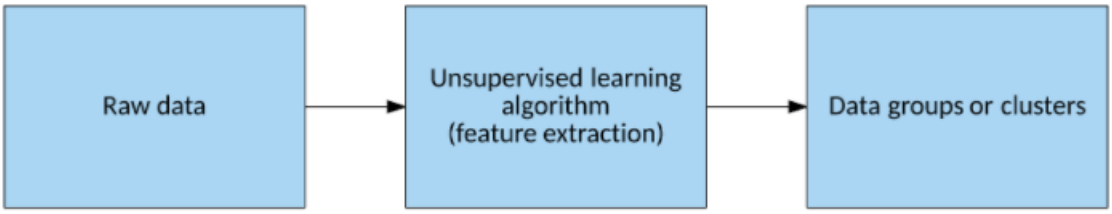

Figura 2.8: Unsupervised Learning Algorithm

- **Semi-supervised learning**

  Este método acaba por ser um meio termo entre supervised e unsupervised learning [15]. Durante o treino utiliza os dois tipos de dados (*labeled* e *unlabeled*), para extrair o máximo da informação. É também utilizado quando não há dados suficientes para o treino de supervised learning algorithms.

2.3 Outros projetos já existentes na área de intervenção

O conceito de Predictive Caching é ainda recente, mas já existem projetos neste domínio de intervenção. Um bom exemplo é a Netflix [18] que já utiliza machine learning e Predictive Caching em alguns domínios, como:

- Prever o estado da rede do utilizador (internet) para adaptar a qualidade do vídeo que transmite.
- Detetar anomalias nos equipamentos onde corre a aplicação (diminuição da qualidade do vídeo, falhas na aplicação, etc).
- Utiliza Predictive Caching para prever que episódio ou que filme o utilizador irá ver, o que permite reduzir os tempos de espera pelo início e carregamento do vídeo.

Capítulo 3

Requisitos e Arquitetura de Aplicações com Predicitive Caching

3.1 Requisitos

Ao longo do projeto definiram-se requisitos que se consideraram essenciais para um bom desempenho do projeto.

Tabela 3.1: Requisitos

Índice	Descrição
C1	Registar todos os pedidos HTTP e respetivas respostas numa base de dados e/ou ficheiros de texto.
C2	Analisar conteúdo dos ficheiros, separar os pedidos das respostas e selecionar a informação relevante para o projeto.
C3	Separar os pedidos por *queryParameters, uriParameters, payload* e cliente.
C4	Para cada tipo de pedido, o produto deverá ser capaz de comparar as respostas dadas aos mesmos.
C5	Para cada tipo de pedido, o produto deverá obter conclusões como: tempo médio, mínimo e máximo sem alteração de payload na resposta e, consequentemente, retirar conclusões sobre o tempo de cache dos mesmos.
C6	Identificar preditivamente os fluxos subsequentes após uma chamada.
C7	Para cada tipo de pedido efetuado identificar os próximos pedidos mais prováveis de surgir.
C8	Elaborar relatório de cache e resultados preditivos.
C9	Estabelecer a ligação entre estes resultados e casos de uso utilizados.

3.2 Arquitetura Tipo

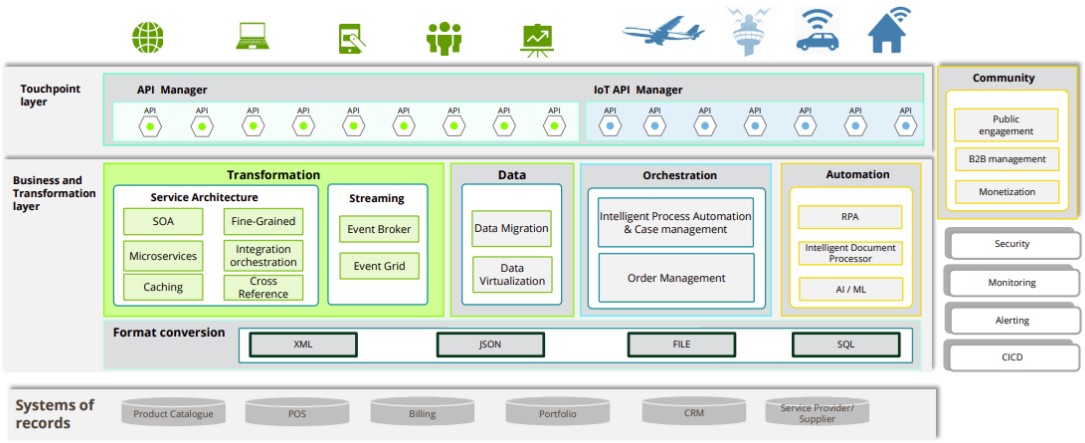

Figura 3.1: Arquitetura Tipo

Analisando a arquitetura presente em 3.1 verificamos que é na camada intermédia onde irá ocorrer maior intervenção. Essa mesma camada pode ser dividida em 3 raios de ação: Transformação, Automação e Dados. Posto isto, com base nestas 3 camadas, estão apresentadas nos pontos seguintes, as ferramentas analisadas e escolhidas para cada área de ação.

3.2.1 Transformação - Mulesoft

Mule [19] foi a plataforma de integração escolhida para o projeto. É um software baseado em JAVA e permite a integração de SaaS e aplicações na *cloud* ou no local. Tem dois grandes elementos: Anypoint Platform e Anypoint Studio.

Anypoint Platform [20] é visto como repositório principal para descoberta, consumo e gestão de ativos (aplicações, APIs, etc). É também uma ferramenta capaz de gerir os ciclos de vida de uma API (design, teste, implementação e gestão). É dividido em 3 componentes:

- **Design Center** - Plataforma o design de APIs utilizando RAML com base nas suas especificações. Permite ainda a reutilização de modelos de dados e teste das APIs.
- **Anypoint Exchange** - Plataforma para exposição, partilha e consumo de APIs.
- **Management Center** - Plataforma para desenvolvimento e gestão de APIs.

Anypoint Studio [21] é o IDE do Mulesoft para desevolvimento e teste de aplicações mais complexas. Um projeto em Anypoint Studio pode ser iniciado de raíz, ou então através do ficheiro criado no Design Center. Contém um conjunto vasto de conectores e ferramentas que permitem acrescentar complexidade e robustez ao sistema. Permite transformação, manipulação e orquestração de dados. As aplicações aqui criadas tanto podem correr no local, na *cloudhub* do Mulesoft, ou em Dockers e/ou Kubernets.

3.2.2 Automação/*Machine Learning* - Python

Como referido em cima, a integração tem assumido cada vez mais preponderância nas organizações. Um dos grandes objetivos do trabalho passa por desenvolver uma nova forma de integração baseada em métodos preditivos e automatização do processo. Para isto, será necessária uma forte componente de programação, e dentro das inúmeras linguagens disponíveis no mercado, a escolha incidiu sobre Python. Python é das linguagens mais escolhidas pelos *data scientists* e *Machine Learning developers*. Segundo [22], 57% dos analistas utilizam esta linguagem de programação, e 33% definem Python como prioritário para os projetos. Isto deve-se à grande evolução das frameworks de Python ao longo do tempo (como TensorFlow) e uma grande variedade de bibliotecas. Existem outras linguagens de programação com um peso no mercado significativo como R, Java, C/C++, etc.

3.2.2.1 Redes Neuronais

O modelo de *Machine Learning* [23] escolhido foi baseado em redes neuronais. Redes Neuronais são um tipo de algoritmo de *Machine Learning* (Supervised Learning) que é inspirado no pensamento humano, tentando replicar a forma como os neurónios humanos se conectam entre si.

Uma rede neuronal é constituída por camadas de neurónios ou nós: entrada (*input layer*), saída (*output layer*), e uma ou mais camadas "escondidas"(*hidden layers*). Cada neurónio está ligado aos neurónios da camada seguinte e tem associada um peso e um threshold. Se a entrada de um nó assume um valor superior ao threshold, então esse nó é ativado e envia informação para a camada seguinte.

Na figura 3.2 podemos observar a estrutura de uma rede neuronal com uma *hidden layer*:

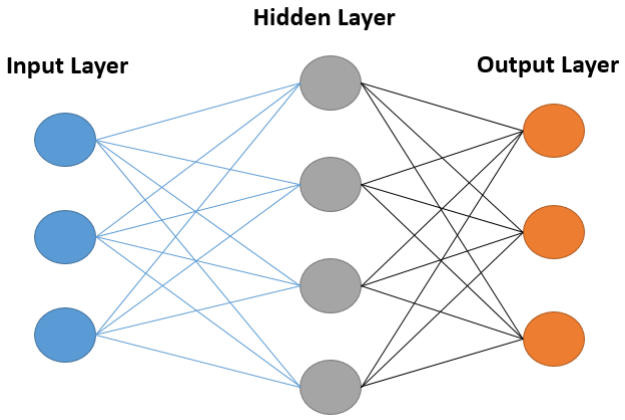

Figura 3.2: Rede Neuronal

Cada nó individual contém os seus dados de entrada, saída, threshold, ou bias, e modelo de regressão linear, cuja fórmula corresponde a algo do género:

$$\sum_{i=1}^{m} w_i x_i + bias = w_1 x_1 + w_2 x_2 + w_3 x_3$$

- *m* corresponde ao número de entradas do nó;
- *w* corresponde ao peso de cada nó de entrada;
- *x* corresponde a cada nó de entrada;

A saída de cada nó é determinada da seguinte forma:

$$output = f(x) = \begin{cases} 1 \; se \sum_{i=1}^{m} w_i x_i + bias \geq 0 \\ 0 \; se \sum_{i=1}^{m} w_i x_i + bias > 0 \end{cases}$$

Depois de determinados os dados de entrada, são atribuídos pesos a cada nó, que definem a sua importância. Todas as entradas são multiplicadas pelo seu peso, somados e de seguida passam por uma função de ativação, o que determina a saída do nó.

Durante o treino, a cada previsão feita pela rede neuronal, a rede verifica a precisão do algoritmo, verificando as previsões corretas e erradas, e recorrendo a uma função de custo (*cost* or *loss function*). No fim de cada ciclo de treino, os pesos são atualizados, utilizando um modelo que indica a forma como essa alteração deve ser feita, tendo por objetivo a minimização da função de custo.

Quando os valores de precisão são favoráveis, a rede neuronal passa do treino para a fase seguinte: aplicação nos casos de uso.

3.2.3 Base de Dados - MySQL

A base de dados escolhida para a realização do projeto foi MySQL. A escolha incidiu sobre esta tecnologia por várias razões, entre as quais o conhecimento prévio da mesma, a ligação com o Mulesoft através de conectores já presentes em Anypoint Studio e o facto de ser gratuita.

MySQL[24] é uma base de dados *open-source*, que começou a ser desenvolvida em 1995, e tem tido um grande crescimento ao longo dos últimos anos. Organizações como Youtube, Spotify, NASA optam por esta base de dados. Em MySQL, os dados são guardados em tabelas, e acedidos através de queries. Utiliza esquemas (Schemas) para definir a estrutura da base de dados, sendo que todas as linhas da mesma tabela devem ter a mesma estrutura, sendo os valores representados por um tipo específico de dados (Integer, Char, Float, etc). As figuras 3.3 e 3.4 mostram como fazer um select e insert numa tabela numa base de dados MySQL.

```
SELECT *
FROM customer
```

Figura 3.3: Select a uma tabela - MySQL

```
INSERT INTO customer (cust_id, branch, status)
VALUES ('appl01', 'main', 'A')
```

Figura 3.4: Inserir dados numa tabela - MySQL

MySQL é a base de dados indicada quando se pretende uma base de dados relacional, em que há várias relações entre tabelas. Por exemplo, uma atualização numa linha de uma tabela, provocar alterações noutras tabelas.

3.3 Casos de Uso

Neste projeto, aplicou-se o conceito a dois casos de uso:

- Deloitte Project - Caso de uso desenvolvido pela equipa, com o objetivo de testar algumas funcionalidades, obter e comparar resultados.
- DigiB - Caso de uso com um cliente real, desenvolvido em conjunto com a equipa da DigiB da Deloitte Portugal e Deloitte Holanda. Aqui o principal objetivo era poder aplicar o conceito a um caso de uso real, com dados do cliente e verificar a relevância e utilidade do projeto no mesmo.

3.3.1 Deloitte Project

O Deloitte Project foi um caso de uso criado pela equipa, que pretendia simular um sistema completo, desde os *target systems* ao cliente, passando pela camada de Integração. Sendo assim, o sistema é constituído por 3 grandes intervenientes:

- Cliente - Representado pelo Postman;
- Plataforma de Integração - Mulesoft;
- *target systems* - 3 tabelas numa base de dados MySql, RESTful Web Service (United Flights) e SOAP Web Service (Delta Flights).

A maneira como o sistema está relacionado pode ser consultada na figura 3.5:

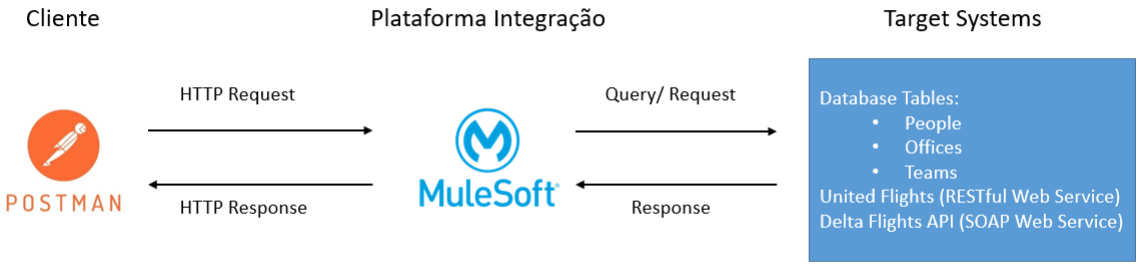

Figura 3.5: Arquitetura Deloitte Project

O Postman [25] é uma plataforma relevante para o desenvolvimento de APIs. Pode desempenhar funções como cliente de APIs (enviar e receber pedidos REST), testagem automática, monitorização, documentação, etc. Neste caso de uso, a sua principal função foi como cliente de API.

Na tabela 3.2 estão representados os pedidos que são feitos e respetiva resposta:

Tabela 3.2: Tipos de Pedidos

Pedido	Resposta
Get /people	Array em que cada objeto representa um funcionário
Get /people/id	Objeto com a informação do funcionário com determinado id
Get /offices	Array em que cada objeto representa um escritório
Get /teams	Array em que cada objeto representa uma equipa
Get /united	Array em que cada objeto representa um voo da companhia aérea United
Get /delta	Array em que cada objeto representa um voo da companhia aérea Delta

Na figura 3.6 estão representadas as respostas dadas a cada tipo de pedido.

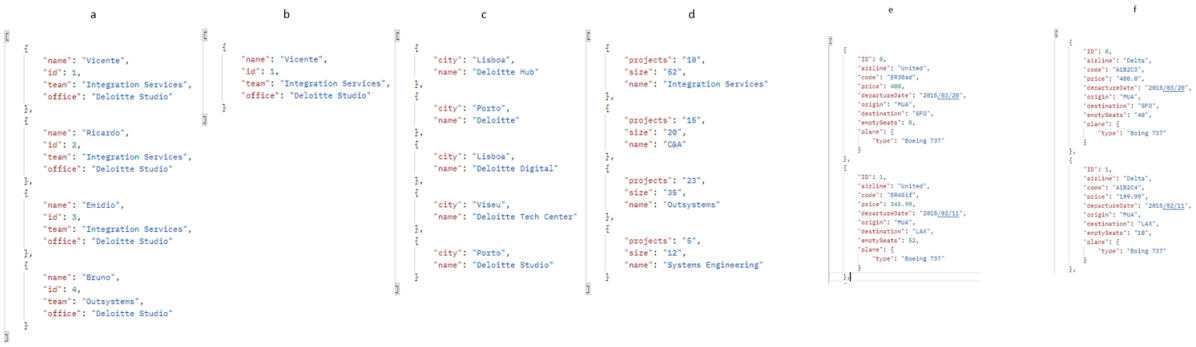

Figura 3.6: Respostas aos pedidos: a) /people b) /people/1 c) /offices d) /teams e) united f) delta

3.3.2 DigiB Project

O segundo caso de estudo do projeto trata-se já de um cliente real, DigiB. A DigiB [26] é um dos principais distribuidores de químicos a nível mundial, e está a passar por uma fase de transformação digital, onde quer trazer mudanças neste nível a toda a industria.

Juntamente com as equipas da Deloitte Portugal e Deloitte Holanda, decidiu-se utilizar uma componente do projeto para testar este novo método de integração que se pretende provar. Foi utilizada a aplicação BC-documents-Papi para este efeito. A aplicação é responsável pela gestão de documentos, desde receção de pedidos, gestão, e envio de respostas, como é possível observar na arquitetura presente na figura 3.7.

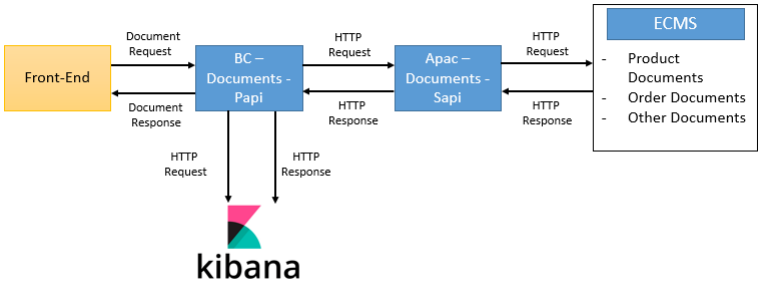

Figura 3.7: Arquitetura do processo de gestão de documentos no DigiB Project

A API BC-Documents Papi, para além de alguma orquestração, faz a ponte entre o Front-End (interface com o cliente), e o restante sistema, através da API bc-apac-documents Sapi, criada para estabelecer a conecção entre a API alvo de estudo e a camada de back-end. O Kibana [27] é utilizado para repositório e visualização de dados. Neste caso, é onde se guardam os pedidos e respostas feitos por todas as APIs do Projeto da DigiB, incluindo BC-Documents-Papi.

Assim que o cliente faz um pedido de um documento, BC-Documents-Papi guarda o evento no Kibana e faz um pedido HTTP a APAC, que por sinal faz outro pedido HTTP aos *target systems*, ECMS, que contêm documentos de produtos, ordens, entre outros. A resposta é assim enviada

fazendo o ciclo inverso e utilizando o mesmo protocolo. De novo, BC-Documents-Papi guarda a resposta no Kibana, e envia o documento ao cliente.

A API é assim responsável por entregar documentos relativos a informação de produtos, Certificados para Análise, Faturas, Documentos de Utilizador e gerar tokens para permitir o acesso a documentos de utilizadores.

Por motivos de privacidade e sensibilidade de dados, o conteúdo dos documentos em si não era guardado. Era efetuada uma encriptação, onde se atribuía uma chave aleatória a cada documento. De referir que documentos com o mesmo conteúdo continham uma chave igual. Essa chave era guardada no payload do evento.

O tipos de pedidos permitidos a esta aplicação eram:

Tabela 3.3: Tipos de Pedidos - DigiB

Método	**Pedido**
Get	/invoices:bc-documents-papi-config
Get	/documents:bc-documents-papi-config
Get	/documents:bc-documents-papi-config
Get	/invoices/(id):bc-documents-papi-config
Get	/user-documents/upload-token:bc-documents-papi-config
Get	/documents/(id):bc-documents-papi-config
Get	/user-documents/(id):bc-documents-papi-config
Get	/coaDocuments/list:bc-documents-papi-config

Capítulo 4

Implementação de uma Prova de Conceito

4.1 Fundamentos Mulesoft

Na primeira fase do projeto, a prioridade era estudar e aprender a utilizar o software de integração que iria ser utilizado, Mulesoft. Para isto, foi realizada a formação Anypoint Platform Development Platform: Fundamentals [28], fornecida pelo Training Mulesoft, cujo princicipais objetivos eram aprender a construir de uma rede de aplicações utilizando API-Led Connectivity, Anypoint Platform e Anypoint Studio, e ganhar conhecimento sobre diversas funcionalidades que este software possui.

Olhando para a estrutra da formação em si, a mesma era divida em 13 capítulos que, para além de material teórico, continha vídeos, Walkthroughs (exercícios com instruções passo-a-passo), exercícios DIY (Do-It-Yourself) para aplicar o que foi aprendido e quizzes em cada capítulo.

Depois desta fase mais teórica, foram desenvolvidos com mais detalhe alguns projetos presentes na formação, com o objetivo de aprofundar alguns dos conceitos abordados.

No entanto, era importante não esquecer os objetivos pricipais com que se iniciou o projeto. Portanto, com base no que foi aprendido investigaram-se outros tipos de funcionalidades que não estavam presentes na formação. Neste processo, encontraram-se de guardar os eventos (*Mule Events*) relativos a pedidos e respostas em ficheiros e bases de dados. De seguida, a exploração da ferramenta de Cache, transformações de dados mais complexas, etc.

Esta aprendizagem e prática revelou-se essencial para o restante projeto, visto que forneceu muito boas bases de Mulesoft, o que facilitou a linha de pensamento e a maneira como se pretendia integrar este software com a parte de análise de dados.

4.2 Ambiente de Teste

Após a primeira fase, e ainda sem conhecimento sobre o caso de uso em que o projeto iria ser aplicado, era importante pensar num ambiente de teste. Isto é, algo que permitisse efetuar pedidos,

dar respostas e coletar dados.

Assim, criou-se o Deloitte Project, cuja arquitetura pode ser visualizada na figura 4.1.

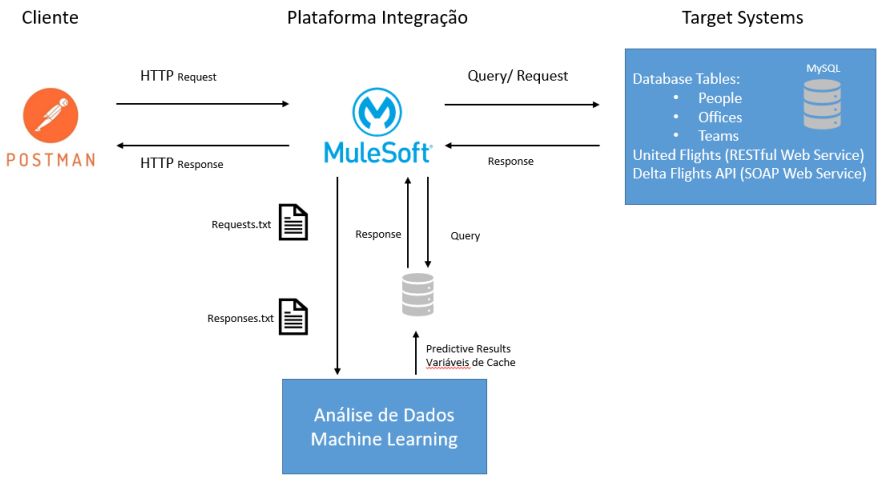

Figura 4.1: Arquitetura Deloitte Project

Seguindo a figura, é possível concluir que a arquitetura se baseia em 4 elementos essenciais.

O lado do Cliente, representado pelo Postman, é o elemento responsável por efetuar pedidos HTTP, o que inicia um conjunto de processos. É o elemento a quem é pretendido dar uma resposta. No extremo oposto, encontram-se os *target systems*, que são os *endpoints* que contêm a informação necessária para dar resposta ao cliente.

Na camada intermédia estão dois elementos: Mulesoft, que reprensenta a camada de integração, e a análise de dados e *Machine Learning*. O mulesoft é o elemento central, responsável por estabelecer a ligação entre cliente e os *endpoints*, e ainda, transmitir a informação relativa a pedidos e respostas HTTP para a parte de inteligência. A parte inteligente é responsável por dar informação útil à camada de integração, como: o que se deve guardar em cache, que pedidos devemos prever e antecipar, etc. Desta forma, é responsável pela automatização do processo.

Todo este sistema e respetivos processos são iniciados por um pedido do lado do cliente. A camada de integração, Mulesoft, recolhe esse pedido e guarda a informação relativa ao pedido num ficheiro de texto. De seguida obtém a informação necessária para a resposta ao pedido em curso. Essa informação tanto pode estar presente nos *endpoints*, como em cache. Em ambos os cenários, o conteúdo da resposta é guardado num outro ficheiro de texto. O fluxo termina quando a resposta é enviada do Mulesoft para o Cliente.

4.2.0.1 *Target systems*

O primeiro passo passou por definir *target systems*. Para isto, reutilizou-se o RESTful Web Service, United Flights, e SOAP Web Service, Delta Flights, que foram fornecidos na formação de Anypoint Platform Development Platform: Fundamentals, e que continuavam acessíveis.

Criaram-se ainda três tabelas na base de dados MySQL, como podemos ver na figura 4.2, para tornar o sistema mais robusto, com mais conteúdo, e mais fácil de controlar, visto que era a equipa que geria o conteúdo presente nestas bases de dados (ao contrário dos Web Services).

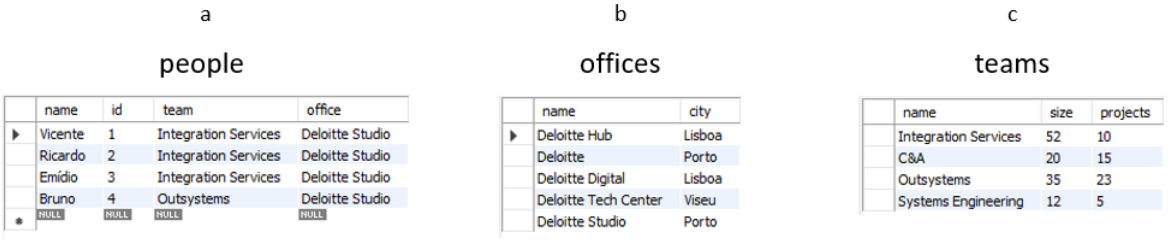

Figura 4.2: *target systems*: a) people b) offices c) teams

A informação presente nos *target systems* era acedida através de HTTP Requests. Na tabela 4.2 podemos ver os métodos permitidos, e como a informação podia ser acedida. Nas figuras A.1 e A.2 é possível consultar o fluxo principal do projeto no Mulesoft. Este fluxo é responsável por recolher o pedido HTTP por parte do cliente, obter a resposta, recorrendo a outros fluxos, e enviar para o cliente.

Tabela 4.1: Pedidos aos *target systems*

Método	**Path**	**Resposta**
Get	http://localhost:8082/people	Array json em que cada objeto representa um funcionário
Get	http://localhost:8082/people/1	Objeto json que representa o funcionário com id = 1
Get	http://localhost:8082/people/2	Objeto json que representa o funcionário com id = 2
Get	http://localhost:8082/people/3	Objeto json que representa o funcionário com id = 3
Get	http://localhost:8082/people/4	Objeto json que representa o funcionário com id = 4
Get	http://localhost:8082/offices	Array json em que cada objeto representa um escritório
Get	http://localhost:8082/teams	Array json em que cada objeto representa uma equipa
Get	http://localhost:8082/united	Array json em que cada objeto representa um voo da companhia aérea United
Get	http://localhost:8082/delta	Array json em que cada objeto representa um voo da companhia aérea Delta.

Tabela 4.2: Pedidos aos *target systems*

4.2.0.2 Alteração do Conteúdo dos *target systems*

Tendo em conta que um dos objetivos do projeto é verificar o tempo que podemos fazer cache da resposta a determinados pedidos, este ambiente de teste só faz sentido se houver alteração no conteúdo dos *target systems*. Por exemplo, se a cada hora, houver uma alteração na tabela Teams, o algoritmo deve indicar que durante esse tempo a informação pode ser guardada em cache.

Portanto, recorrendo ao Mulesoft, criaram-se fluxos próprios (figura A.3) para provocar alterações no conteúdo dos *target systems*. As alterações eram feitas na tabela people, em cada linha, e na tabela Teams. As frequências das mesmas podem ser consultadas na tabela 4.3.

As alterações a que me refiro, não necessitavam de ser uma completa reestruturação do conteúdo da tabela. Poderia ser algo tão simples como uma alteração do valor de uma coluna.

Tabela 4.3: Fluxo alteração do conteúdo das tabelas

Alteração	**Frequência**
Editar people : id = 1	15 minutos
Editar people : id = 2	8 minutos
Editar people : id = 3	15 minutos
Editar people : id = 4	20 minutos
Editar teams	30 minutos

4.2.1 Recolha e Seleção de Dados

Após a criação do ambiente de teste, o foco estava em obter uma boa quantidade de dados para, posteriormente, começar a desenvolver o algoritmo de seleção, análise e tratamento de dados.

O passo inicial passou por desenvolver fluxos no Mulesoft responsáveis por enviar os eventos relativos a pedidos e respostas para os ficheiros correspondentes. Desta forma, assim que cada HTTP Listener recebia um pedido e enviava uma resposta, todo o conteúdo Mule Event era enviado para os ficheiros de texto, requests.txt e responses.txt, respetivamente. Na figura 4.3 encontram-se exemplos de como cada Mule Event era guardado no respetivo ficheiro.

Interessava agora, ao projeto obter uma quantidade de dados significativa, com alguma lógica e automatização que permitisse reconhecer facilmente se os resultados que se iriam obter no futuro estariam corretos ou não.

Tendo em conta a metodologia utilizada anteriormente para provocar alteração nas bases de dados, concluí-se rapidamente que a mesma pode voltar a ser a base deste processo. Sendo assim, criaram-se novos fluxos no Mulesoft responsáveis por efetuar pedidos ao longo do tempo e receber respostas, figura A.4. Na tabela 4.4 podemos ver os pedidos feitos e a frequência dos mesmos.

Desta forma, ao fim de pouco tempo, o projeto já conta com uma boa quantidade de dados, o que será fundamental nos passos futuros.

a


```
{
  "request": {
    "correlationId": "d4f4a720-b260-11eb-bde5-0a0027000007",
    "date": "2021-05-11T14:57:32.788+01:00",
    "attributes": {
      "headers": {
        "x-correlation-id": "d4f4a720-b260-11eb-bde5-0a0027000007",
        "host": "localhost:8082",
        "user-agent": "AHC/1.0",
        "connection": "keep-alive",
        "accept": "*/*"
      },
      "clientCertificate": null,
      "method": "GET",
      "scheme": "http",
      "queryParams": {

      },
      "requestUri": "/united",
      "queryString": "",
      "version": "HTTP/1.1",
      "maskedRequestPath": null,
      "listenerPath": "/united",
      "relativePath": "/united",
      "localAddress": "/127.0.0.1:8082",
      "uriParams": {

      },
      "rawRequestUri": "/united",
      "rawRequestPath": "/united",
      "remoteAddress": "/127.0.0.1:51373",
      "requestPath": "/united"
    },
    "payload": ""
  }
```


b


```
{
  "response": {
    "correlationId": "d4f4a721-b260-11eb-bde5-0a0027000007",
    "requestUri": "/people",
    "date": "2021-05-11T14:57:33.316+01:00",
    "message": {
      "payload": [
        {
          "name": "Vicente",
          "id": 1,
          "team": "Integration Services",
          "office": "Deloitte Studio"
        },
        {
          "name": "Ricardo",
          "id": 2,
          "team": "Integration Services",
          "office": "Deloitte Studio"
        },
        {
          "name": "Emídio",
          "id": 3,
          "team": "Integration Services",
          "office": "Deloitte Studio"
        },
        {
          "name": "Bruno",
          "id": 4,
          "team": "Outsystems",
          "office": "Deloitte Studio"
        }
      ],
      "attributes": null
    }
  }
}{
```


Figura 4.3: a) evento relativo a um pedido HTTP b) evento relativo a uma resposta

Tabela 4.4: Sequência de Pedidos

Método	**Path**	**Frequência**
Get	http://localhost:8082/people	5 minutos
Get	http://localhost:8082/people/1	5 minutos + 1 delay
Get	http://localhost:8082/people/2	5 minutos + 2 delay
Get	http://localhost:8082/people/3	5 minutos + 3 delay
Get	http://localhost:8082/people/4	5 minutos + 4 delay
Get	http://localhost:8082/offices	8 minutos
Get	http://localhost:8082/teams	9 minutos
Get	http://localhost:8082/united	12 minutos
Get	http://localhost:8082/delta	13 minutos

Depois da recolha de dados, seguia-se a fase de configuração. Nesta fase entra o Python, principal linguagem de programação utilizada no projeto. Apesar de cada mule event ser guardado como objeto JSON, os documentos em si não eram compatíveis com as bibliotecas json do Python, o que obrigou a pequenas alterações nos ficheiros. Depois da correta leitura, seguiu-se a fase de selecção, onde foram definidos os campos presentes relevantes para o projeto em cada evento (pedido e resposta).

Do lado do pedido os campos considerados foram:

- CorrelationId
- Date
- Payload
- Attributes:
  - QueryParams
  - UriParams
  - Method
  - ListenerPath
  - Scheme
  - QueryString

Do lado da resposta:

- CorrelationId
- Date
- Payload
- RequestUri

É importante destacar campos como o CorrelationId, que é uma chave única para cada evento. É o que nos permite identificar a resposta a cada pedido, sendo portanto um campo fulcral. Destaque também para os UriParams do lado do pedido e RequestUri do lado da resposta porque é o que distingue os diversos tipos de pedido.

Do lado da resposta, o Payload é o campo que contém o conteúdo da resposta que será dada ao cliente e, portanto, um dos campos que será alvo de análise.

4.3 DigiB Project

Neste caso de uso foram utilizados dados de produção, que foram recolhidos ao longo de 18 dias úteis. A arquitetura utilizada foi muito semelhante à demonstrada em 3.7, mas com uma pequena alteração. Os logs (registos dos eventos), para além do Kibana, eram também guardados em ficheiros de texto, como podemos ver na figura 4.4:

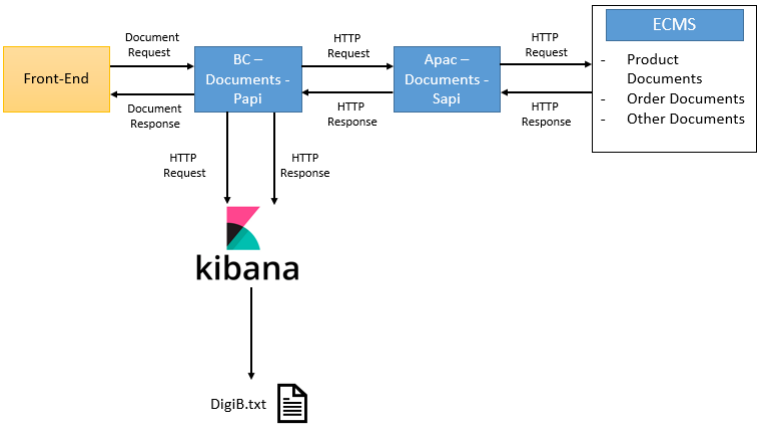

Figura 4.4: Arquitetura Projeto DigiB

4.3.1 Seleção de Dados

No apêndice C.1, podemos ver o formato em que eram guardados os eventos no Kibana e de seguida, nos ficheiros de texto. De uma forma idêntica ao que se sucedeu no caso anterior, era necessário uma seleção e filtragem destes dados, porque nem todos os campos do evento eram relevantes para o projeto.

Num primeiro momento, é importante dar destaque para o campo "LogStep". Este campo pode assumir os valores "SERVICE START", caso se trate de um pedido, "SERVICE END", no caso do evento corresponder a uma resposta, e "SERVICE ERROR"no caso de ser uma mensagem de erro.

No momento em que se fez esta distinção, começou-se a selecionar os dados dependendo do tipo de evento em questão. Do lado do pedido, os campos que se consideram relevantes foram:

- LogStep;
- CorrelationId;
- Date;
- Payload;
- TopServiceName;
- Authorization;
- Attributes:
  - headers;
  - queryParams;
  - uriParams.

Do lado da resposta e dos erros, os campos considerados foram:

- LogStep;
- CorrelationId;
- Date;
- Payload;
- topServiceName

O CorrelationId assume o mesmo papel do caso de uso anterior, em que representa a chave única de cada evento par pedido/resposta. No entanto, podemos verificar que há 2 campos aqui que não estão presentes no Deloitte Project, e são eles o topServiceName, e o Authorization. O topServiceName representa o serviço para cada tipo de documento a que a API BC-documents-Papi pretende dar resposta. Sendo assim temos um serviço para User Documents, outro para Product Documents, Invoices, etc. O campo Authorization é o que permite identificar os clientes/utilizadores. É também uma chave única e aleatória atribuída a cada sessão que é iniciada nesta aplicação.

Outro ponto que é importante realçar, é que, enquanto que no Deloitte Project o uriParam é o que identifica cada tipo de pedido, neste caso o parâmetro responsável por isso é o queryParam. Por esta razão, nas fases seguintes do projeto, o uriParam é ignorado quando o caso de uso em questão é o DigiB Project.

Nos eventos das respostas, o payload pode ser um objeto ou array JSON, ou então, quando se trata de um documento, indica-nos o tamanho do mesmo em bytes, payload size, e o checksum correspondente. O checksum é um código que depende do conteúdo do documento. Portanto, o código é igual para documentos iguais, e o contrário para os casos em que o conteúdo dos documentos é diferente.

4.3.2 Análise e Tratamento de Dados

Neste ponto, existem dois ficheiros, correspondentes a pedidos e respostas dadas pela plaforma de integração. Já com a ajuda do Python criaram-se duas listas: uma que continha todos os pedidos lidos, e outra as respostas. Cada elemento da lista representava um pedido/resposta lido, que continha apenas os campos que considerados pertinentes, tal como mostra a figura 4.5.

Com esta estrutura de dados, foram criados vários métodos que permitiam encontrar respostas para cada pedido, ou respostas com um determinado correlationId, etc. Métodos que seriam úteis para as restantes fases do projeto.

Nesta fase, o principal objetivo era organizar os dados para procurar padrões, verificar as respostas que eram dadas aos diferentes pedidos, etc. Fundamentalmente, era preparar os dados para aplicar os algoritmos de cache e previsão pensados. Portanto, a partir deste ponto, começaram-se a moldar as estruturas de dados tendo em conta cada objetivo que se pretendia atingir.

Olhando para o primeiro objetivo (analisar as respostas dadas a cada pedido), era necessário subdividir a lista de pedidos inicial por URI ou query, dependendo do caso de uso, (que era o que

identificava cada um deles), como está demonstrado na figura 4.6. O resultado desta divisão era uma outra lista, em que cada elemento representava um objeto, que continha o URI correspondente, e a lista de pedidos com o mesmo. Esta estrutura de dados denomina-se "Cache List".

Figura 4.5: a) lista de pedidos b)lista de respostas

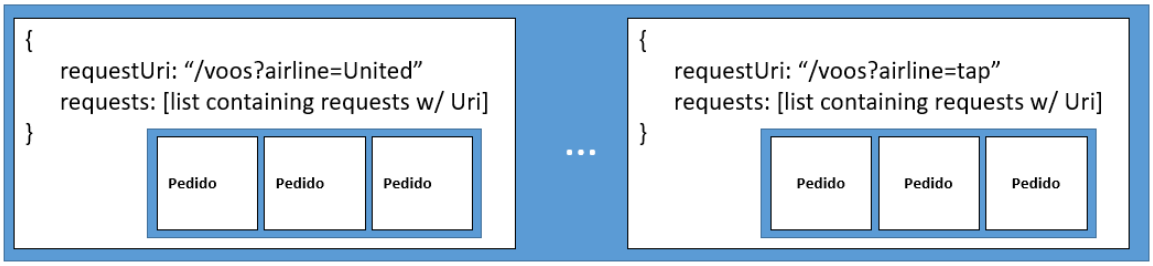

Figura 4.6: Cache List

4.4 Algoritmo de Cache

Já com os dados tratados e disponíveis para análise, o foco era pensar num algoritmo capaz responder à questão : "Quanto tempo podemos fazer cache de cada pedido? ".

Portanto, tendo em consideração a estrutura apresentada em 4.6, o primeiro passo era analisar a resposta dada a cada pedido. Por exemplo, para o primeiro elemento, que contém o requestUri "/voos?airline=United", deveria ser analisada a resposta a cada pedido presente na lista "requests", e verificar como mudava o payload ao longo do tempo. Para isto foram criadas duas novas listas auxiliares:

- Payload Check - lista em que cada elemento representa um tipo de payload presente na lista de respostas.

- Payload Values - lista em que cada elemento contém o ID do elemento do payload check cujo conteúdo é igual à resposta dada a um determinado pedido.

Na figura 4.7 ver como funciona o preenchimento destas duas listas.

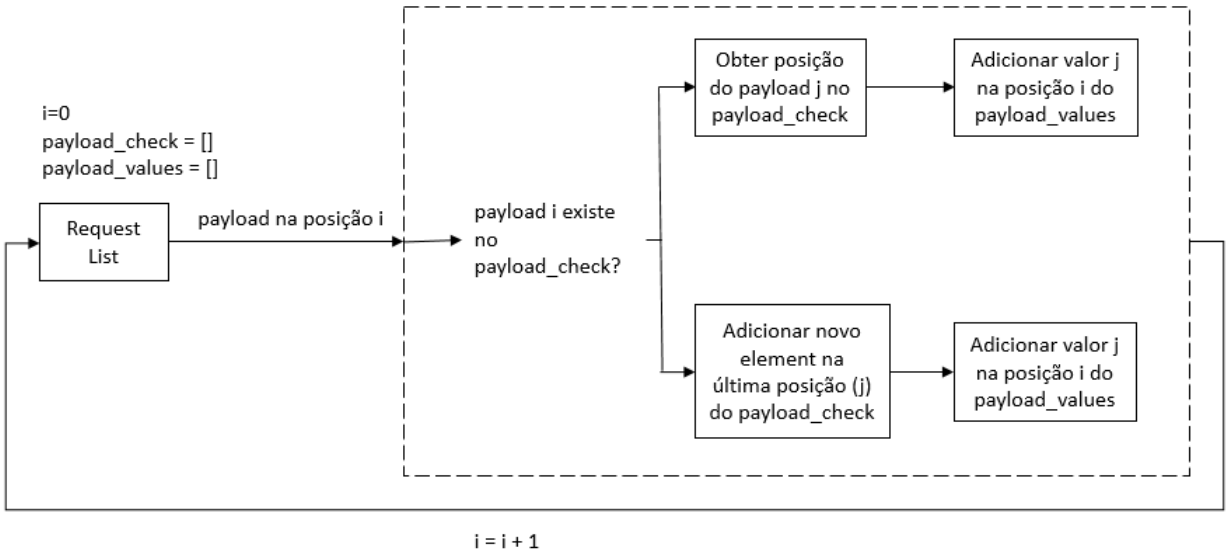

Figura 4.7: Fluxograma

Na figura 4.8 podemos ver o resultados de Payload Values e Payload Check quando este fluxograma é aplicado na lista de Pedidos em a) (dados apenas para efeito de demonstração).

a

Requests
Request1 – Uri1
Request2 – Uri1
Request3 – Uri1
Request4 – Uri1
Request5 – Uri1
Request6 – Uri1
Request7 – Uri1
Request8– Uri1
Request9 – Uri1
Request10 – Uri1

b

Responses - Payload
Response1 – a
Response2 – a
Response3 – b
Response4 – b
Response5 – b
Response6 – c
Response7 – c
Response8 – d
Response9 – d
Response10 – d

c

Payload Values
0
0
1
1
1
2
2
3
3
3

d

Payload Check
a
b
c
d

Figura 4.8: a) Lista pedidos b) Lista de Respostas c) Payload Values d) Payload Check

Na figura 4.9 podemos ver como varia o Payload Values ao longo do tempo:

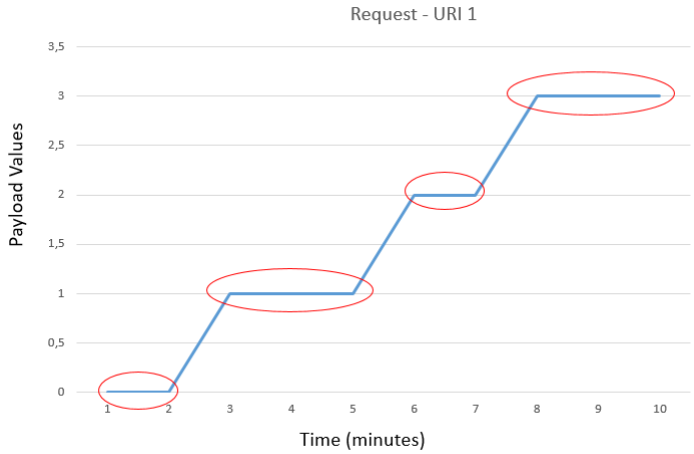

Figura 4.9: Payload Values em função do tempo

Consultando a figura, verificamos que quando o Payload Value se mantém no mesmo nível, é quando o payload se mantém estático, ou seja, não houve alteração durante um determinado período de tempo. E neste ponto, queremos calcular valores como tempos mínimo, máximo, médio, mediana, etc, em que o payload se mantém nesse mesmo nível. Aplicando esse cálculo a este exemplo obtemos os seguintes valores:

Instantes de tempo, em minutos, em que o payload se mantém no mesmo nível: [1,2,1,2]
Mínimo: 1 minutos
Máximo: 2 minutos
Média: 1.5 minutos
Mediana: 1.5 minutos
Moda: 1 e 2 minutos

4.5 Algoritmo de Previsão de Pedidos

Para este segundo o objetivo, o conteúdo das respostas dadas a cada pedido é irrelevante, logo, o foco incide apenas sobre os pedidos.

No entanto, pode ser aplicada a mesma linha de pensamento anterior com algumas mudanças. Dado que o RequestUri é o que pretendemos analisar, passa a ser esse o principal tempo de comparação. Portanto os dois vetores criados no lugar do Payload Check e Payload Values são:

- Request Check - lista em que cada elemento representa um tipo de pedido presente na lista de pedidos.
- Request Values - lista em que cada elemento contém o ID do elemento do payload check cujo pedio é igual à resposta dada a um determinado pedido.

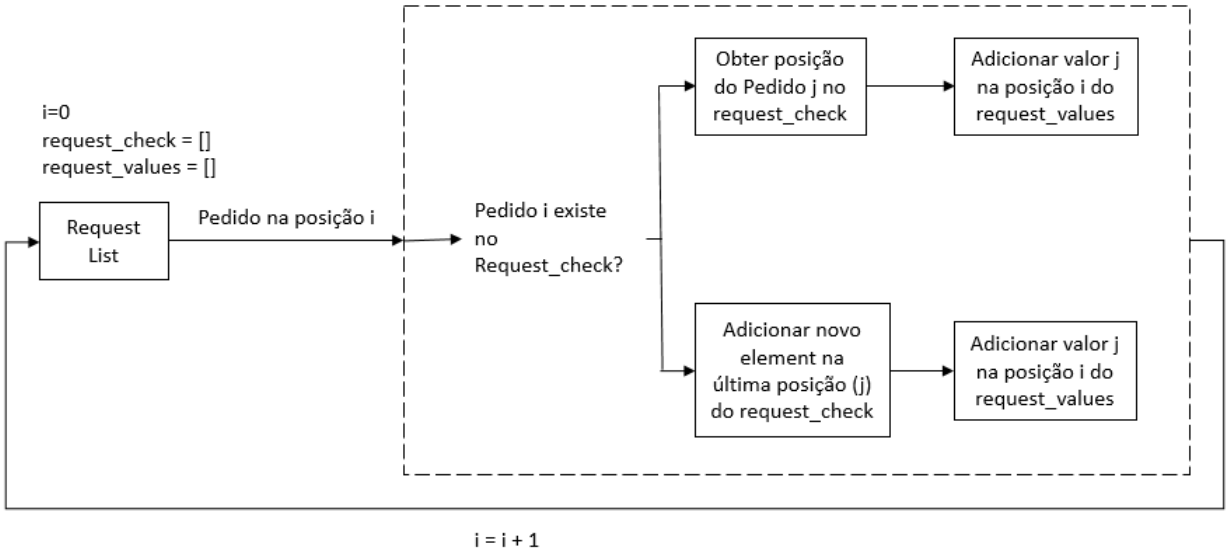

Figura 4.10: Fluxograma de Previsão

Na figura 4.10 podemos verificar como funciona o preenchimento das listas.

Aplicando este processo aos pedidos em a) na figura 4.12, podemos ver os seguintes resultados:

a

Requests
Request1 – Uri1
Request2 – Uri2
Request3 – Uri3
Request4 – Uri1
Request5 – Uri1
Request6 – Uri2
Request7 – Uri3
Request8 – Uri0
Request8 – Uri0
Request8 – Uri1
Request8 – Uri2

b

Request Values
0
1
2
0
0
1
2
0
0
1
2

c

Request Check
Uri1
Uri2
Uri3

Figura 4.11: a) Lista de Pedidos b) Payload Values c) Payload Check

A forma como o Request Value varia pode ser vista na figura 4.12.

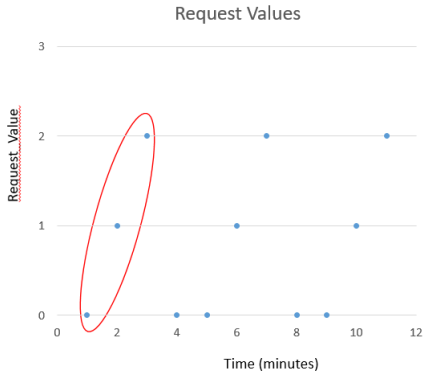

Figura 4.12: Request Values em função do tempo

Portanto, colocando estes valores num array, ordenados de forma temporal, temos o seguinte resultado:

[0 1 2 0 0 1 2 0 0 1 2]

Neste ponto, entra *Machine Learning* (recorrendo ao tensorflow, biblioteca Python, onde se utilizaram redes neuronais) para prever os pedidos. Aplicando ao exemplo que temos, o objetivo é prever para cada pedido (0, 1 ou 2), qual o pedido que seja chamado a seguir (0, 1 ou 2, que correspondem aos pedido Uri1, Uri2 e Uri3 respetivamente).

Como os Request Values estão ordenados temporalmente, obtêm-se os parâmetros de entrada e parâmetros de saída para o treino da rede:

Entrada: [0 1 2 0 0 1 2 0 0 1 2]
Saída: [1 2 0 0 1 2 0 0 1 -]

Podemos verificar que se retirarmos o primeiro elemento do primeiro vetor e deslocar os restantes para a esquerda , os parâmetros de entrada são iguais aos de saída. Este resultado diz-nos que, no primeiro caso, o pedido 1 vem a seguir ao Pedido 0, no segundo caso, o pedido 2 vem a seguir ao 1, e assim sucessivamente.

Para uma dada entrada, neste caso com o valor 0, 1 ou 2, a saída expectável da rede neuronal, é um array que contém as probabilidades de cada um dos tipos de pedidos (0, 1 ou 2) ser o seguinte. As especificidades da rede neuronal foram sendo alteradas ao longo do projeto, conforme o caso de uso que estávamos a utilizar.

4.6 Integração dos Resultados nos Casos de Uso

Nesta fase, ocorre a integração dos algoritmos pensados com os casos de uso apresentados em 4.2. Para isto criaram-se diferentes dataframes, dependendo do caso de uso, e alguma formatação adicional nos dados, principalmente nos dados do cliente (DigiB).

Posto isto, depois da análise de dados, obtenção de resultados de cache e previsão, o objetivo passa por pensar numa forma de interligar estes componentes com a camada de integração. Assim, torna-se possível colocar os resultados em prática no ambiente de teste, e verificar e utilidade dos mesmos.

Fazendo um ponto da situação, neste momento tem-se:

- Algoritmo de Cache - Para um determinado pedido (URI), retorna como resultado o tempo médio, mínimo, máximo, moda e mediana sem alteração de payload.
- Algoritmo de Previsão - Para um determinado pedido, obtém-se um array com as probabilidades de cada tipo de pedido presente no conjunto de dados ser chamado, e consequentemente o pedido mais provável (previsão).

De seguida o que criaram-se duas listas (figura 4.13). A primeira correspondia aos resultados de Cache, em que cada elemento representava um tipo de pedido (URI), com o tempo de cache (cache time), uma variável (cache var) que nos indicava se pretendíamos fazer cache da resposta a esse pedido ou não, tendo em conta o tempo de cache que era obtido.

A segunda lista correspondia aos resultados de previsão. Cada elemento representava um determinado tipo de pedido (URI), initial request, que continha o pedido/URI seguinte (next request), com a probalidade associada (prob), e tempo de resposta médio a esse pedido (response time).

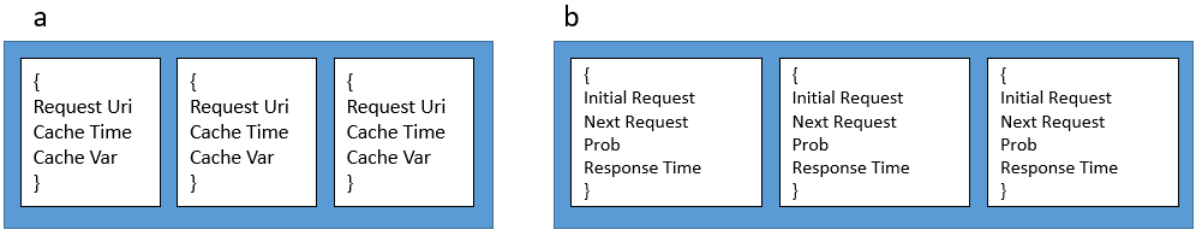

Figura 4.13: Resutados: a) cache b) previsão

4.6.1 Integração dos Resultados com o Ambiente de Teste

De forma a integrar estes resultados com a camada de integração, para poder ser aplicado diretamente nas respostas que eram dadas ao cliente (Postman), recorreu-se de novo à base de dados MySql. Criou-se uma tabela, denominada Predictive Results, que continha as seguintes colunas:

- **Initial Request** - Uri correspondente ao pedido Inicial.
- **Next Request** - Uri correspondente ao pedido seguinte previsto.
- **Prob**- probabilidade do next request ser chamado após o initial request.
- **Response Time**- Tempo médio de resposta ao Initial Request.

- **Cache Var** - Variável que assume os valores: 1 se o conteúdo da resposta ao initial request está guardado em cache; 0 caso contrário.
- **Cache Time** - Tempo considerado para cache.

Cada linha da tabela correspondia a cada tipo de pedido presente no conjunto de dados considerado.

Neste ponto, tendo em conta o ponto de vista do negócio, começou-se a pensar sobre o que realmente valia a pena guardar em cache. Portanto, dados que tinham uma taxa de mudança considerada elevada não valiam a pena guardar. E antecipar pedidos com tempos de resposta muito rápidos, abaixo de um certo limite, também não acrescentava valor do ponto de vista do negócio.

Posto isto, na camada de integração definiram-se 4 variáveis:

- **prob min** - Após um pedido inicial, esta variável representa a probabilidade mínima para a qual obtínhamos e colocávamos em cache a resposta ao pedido seguinte previsto.
- **cache time min** - Para cada pedido, o tempo mínimo para o qual colocávamos o conteúdo da resposta a um determinado pedido em cache.
- **response time min** - tempo de resposta mínimo para o qual pretendíamos antecipar o pedido.
- **pred cache time** - Ao antecipar um pedido, esta variável representa o período de tempo que pretendemos guardar a sua resposta em cache.

Neste cenário, estão encontradas agora todas as condições necessárias para aplicar os resultados diretamente na camada de integração. Para aplicar o resultado de cache, o fluxograma pensado está representado na figura 4.14:

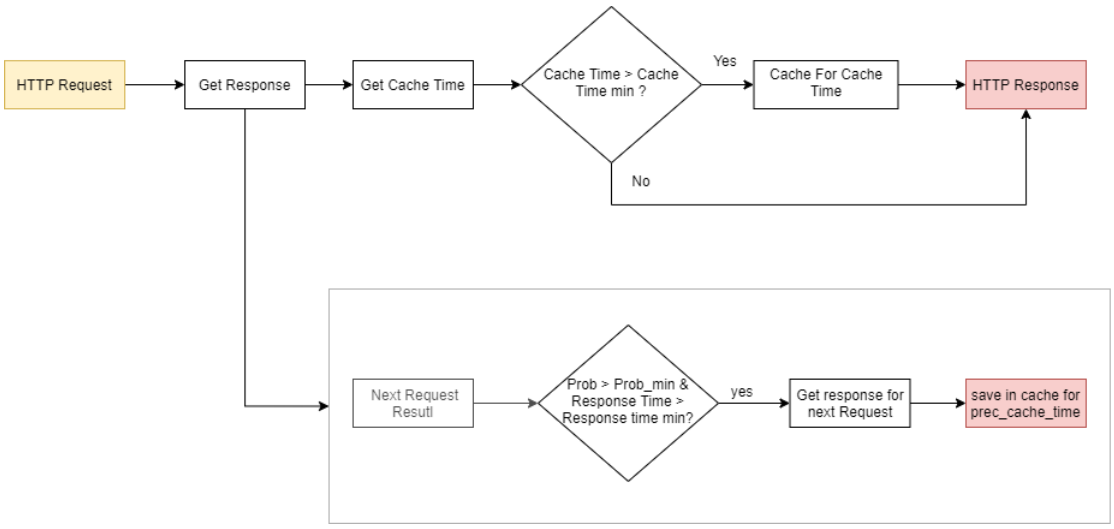

Figura 4.14: Fluxograma Integração

Como podemos ver na figura 4.14, assim que é recebido um pedido HTTP, são despoletados dois fluxos. O primeiro, o principal, é responsável por obter a resposta ao pedido e verificar se deve colocar o conteúdo em cache ou não, e de seguida envia a resposta HTTP. No primeiro cenário, caso Cache Time > Cache Time min, coloca em cache durante Cache Time.

No fluxo secundário, que corre de forma assíncrona ao principal, o objetivo é prever o pedido seguinte e se é útil antecipá-lo ou não. Isto é, obter a resposta ao pedido seguinte e guardar em cache durante pred cache time.

Na figura 4.15 podemos consultar a rede neuronal, que indicavam os resultados das previsões, utilizada neste caso:

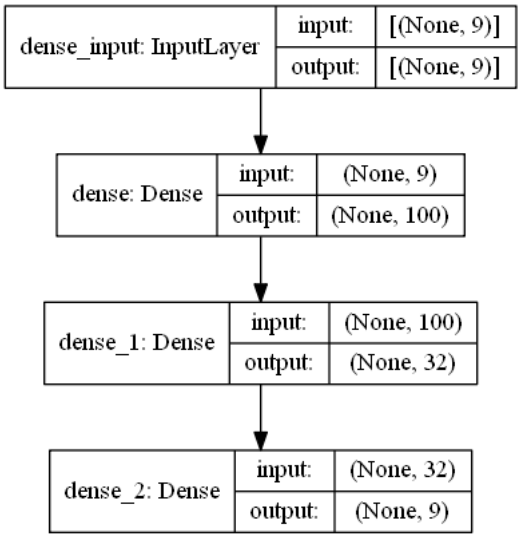

Figura 4.15: Rede Neuronal

Este diagrama indica que a rede é composta por 4 camadas: *input layer*, duas *hidden layers*, e *output layer*. A primeira camada é composta por 9 neurónios, que correspondem ao número de pedidos diferentes que estão presentes no conjunto de dados. A primeira *hidden layer* tem 9 neurónios de entrada, enquanto que a segunda tem 100. Finalmente a última camada tem 32 parâmetros de entrada, o que corresponde à saída da última *hidden layer*, e 9 parâmetros de saída.

Importante referir que este algoritmo foi treinado com uma taxa de aprendizagem de 0,01.

4.6.2 Integração dos Resultados com projeto da DigiB

Com o projeto da DigiB não foi possível aplicar diretamente na camada de integração e verificar resultados reais. Portanto, neste novo cenário objetivo passava por ter algo mostrasse os resultados dos algoritmos explicados nas secções anteriores.

A solução pensada consistiu em relatar os resultados dos algoritmos em 2 ficheiros Excel, o primeiro para os resultados do Algoritmo de Cache e o segundo para os resultados do Algoritmo de Previsão.

É importante referir que, neste contexto, houve 3 parâmetros muito relevantes neste ponto:

- **TopServiceName** - Serviço.
- **QueryParams** - Parâmetros de query, identificava os pedidos.
- **Authorization** - Identificador de sessão/cliente.

Para os resultados de cache, separaram-se os pedidos por TopServiceName, visto que não fazia sentido analisar o conteúdo de pedidos e respostas de serviços diferentes. De seguida, separou-se os pedidos por queryParams. Isto significa que o pedido 1 no Serviço A e o pedido 1, com os mesmo queryParams do anterior, no Serviço B era analisados separadamente. O objetivo era obter o tempo de cache para cada pedido em cada serviço. Na figura 4.16 podemos ver uma lista, em que cada elemento representa o resultado para um determinado parâmetro de query num serviço. O Cache Time representa o tempo de cache, a variável Cache Var assume o valor 1 caso seja vantajoso guardar em cache e 0 caso contrário. A variável Finish Dates é um array que contém a altura em que foi detetada uma mudança no conteúdo da resposta e Number of Requests, representa a quantidade de vezes que este pedido foi feito.

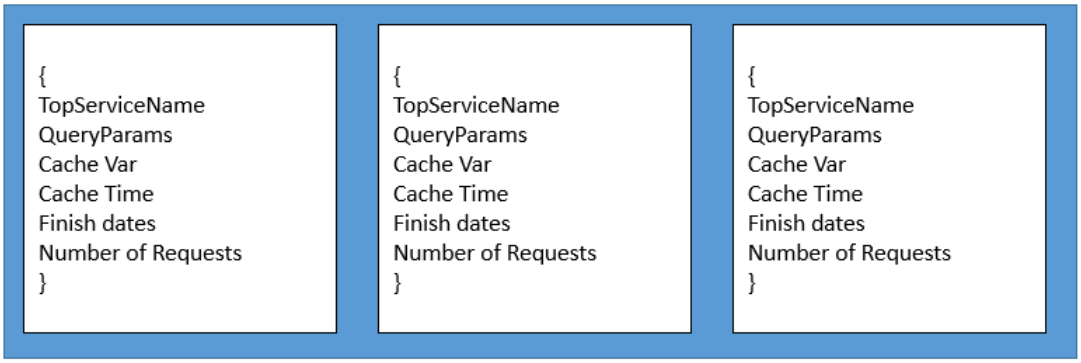

Figura 4.16: Resultado de Cache

De seguida, toda esta informação foi escrita num ficheiro Excel, como será possível ver no capítulo seguinte.

Avançando para a previsão de pedidos, e tendo em conta os parâmetros acima referidos, pretendia-se prever os serviços e/ou queryParams para cada cliente (identificado pelo campo Authorization) e de uma forma global.

É importante referir ainda que para cada cliente era treinado um modelo de *Machine Learning* próprio, com base nos pedidos que esse cliente fazia.

Com o intuito de fornecer informação com mais detalhe, não só se previu o serviço e/ou queryParams mais provável, como também os segundos mais prováveis com a respetiva confiança.

Para a previsão de serviços, obtinham-se 2 listas, figura 4.17: a primeira em que não considerávamos os diferentes clientes, e a segunda em que os mesmos eram considerados.

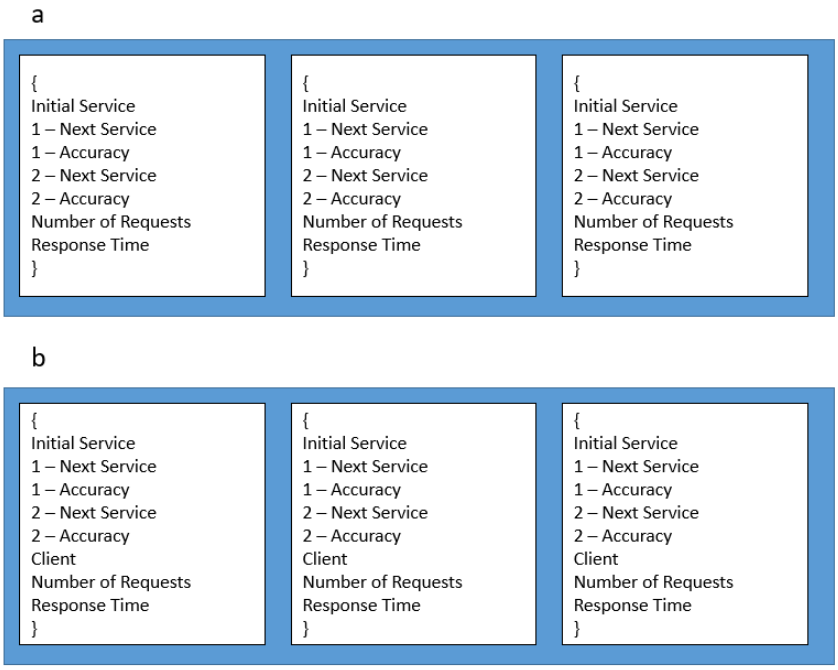

Figura 4.17: Previsão do Service: a) visão global b) Considerando clientes

De uma forma idêntica, obtém-se os seguintes resultados para a previsão dos queryParams, representados na figura 4.18.

No momento em que se pretendia prever os serviços e queryParams o resultado obtido era a lista presente na figura 4.19.

Após a obtenção destes resultados, escreveu-se toda esta informação num novo ficheiro Excel, em 5 folhas diferentes. Para além dos resultados, foram também registados os desempenhos do modelo de *Machine Learning*, de forma a avaliar a qualidade do mesmo.

Por último, para calcular a capacidade máxima de armazenamento, somou-se o payload size de todas as respostas. O resultado esperado é o valor, em bytes, do armazenamento necessário para fazer cache de todos os pedidos.

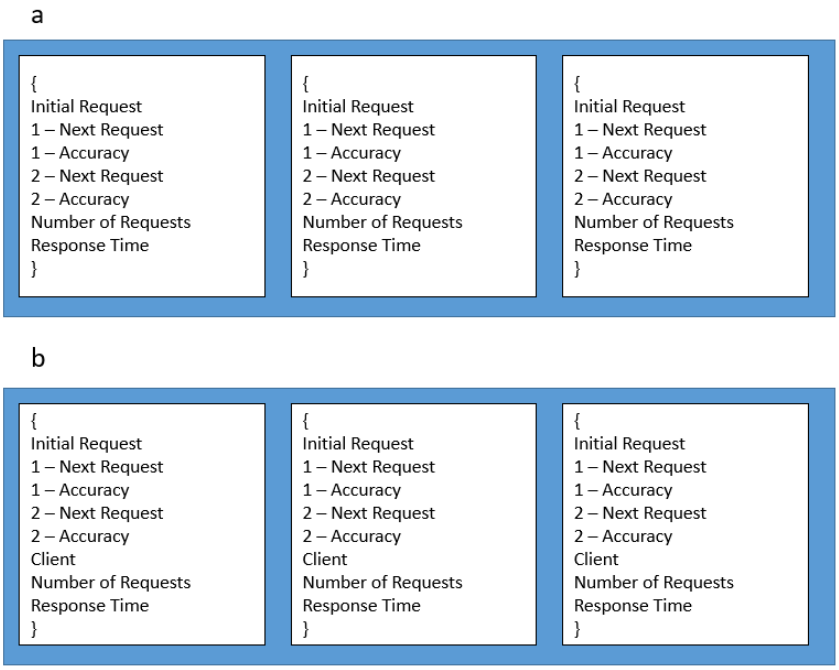

Figura 4.18: Previsão do queryParams: a) visão global b) Considerando clientes

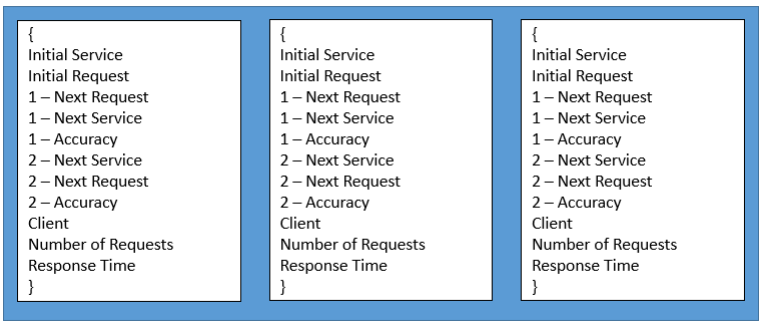

Figura 4.19: Previsão de Serviço e QueryParams

Capítulo 5

Resultados Obtidos

Este capítulo tem como principal objetivo analisar os resultados obtidos ao longo do projeto. Assim conseguiremos verificar se os algoritmos fornecem resultados úteis e se este novo conceito que queremos provar é efectivamente possível de aplicar nos contextos pretendidos.

No caso do Deloitte Project, os resultados são concretos, com aplicação direta dos algoritmos no caso de uso. Assim é possível ver o efeito do Predictive Caching no sistema.

No projeto da DigiB não foi possível aplicar estes resultados em contexto real. Desta forma não teremos uma comparação entre resultados antes e após a aplicação dos algoritmos. No entanto, ter-se-á uma ideia de como este novo formato de integração pode influenciar este sistema. Teremos portanto uma avaliação qualitativa apenas.

5.1 Resultados - Deloitte Project

No Deloitte Project, os resultados que serão alvo de análise são:

- Resultados de Cache e de Previsão;
- Tempo de Resposta dos pedidos sem cache;
- Tempo de Resposta dos pedidos com cache;
- Ciclos de Integração sem previsão;
- Ciclos de Integração com previsão.

Tendo em conta o método referido em 4.3 para recolha de dados, a aplicação responsável por essa tarefa a correr entre 60 e 65 minutos, o que resultou num conjunto de dados com 222 pedidos no total.

Ao aplicar este conjunto de dados no algoritmo de *Machine Learning* presente em 4.7.1, existem dois indicadores a ter em conta, que nos dão informação sobre a qualidade do mesmo:

- **Loss:** valor que se pretende minimizar à medida que a rede neuronal é treinada. Quanto menor for este valor, mais perto estão as previsões de corresponderem aos valores corretos.

- **Accuracy:** representa a percentagem de vezes que as previsões dadas pela rede neuronal correspondem ao valor correto.

Outra variável importante para o treino da rede neuronal é a "epochs". Esta variável representa o número de vezes que o algoritmo passa pelo conjunto de dados durante o treino. Portanto, em cada epoch, existe uma atualização nos pesos dos neurónios da rede. A figura seguinte mostra a loss e a accuracy do algoritmo em função das epochs.

O batch size é também uma variável muito relevante, que representa o número de amostras que será utilizada para treinar a rede neuronal de uma vez. Isto é, se definirmos o batch size como 20, numa primeira instância, o algoritmo utiliza as primeiras 20 amostras e treina a rede. De seguida, utiliza mais 20 amostras e volta a treinar e assim sucessivamente. E faz isto durante o número de epochs definido. Neste caso, o batch size utilizado foi o tamanho do dataset.

A figura 5.1 mostra-nos como variam a Accuracy e a Loss da rede neuronal em função das epochs, nos conjuntos de dados de teste e de treino.

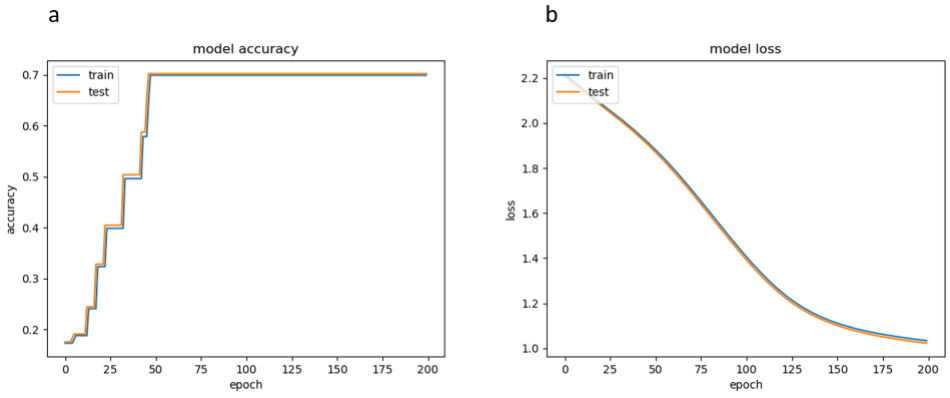

Figura 5.1: a) Accuracy b) Loss

Os valores finais da Loss e Accuracy:

- loss = 1,02
- accuracy = 70 %

5.1.1 Resultados de Cache e Preditivos

Para este caso de uso, os resultados preditivos e de tempo de cache, em minutos, estão presentes na tabela 5.1:

Tabela 5.1: Resultados

Pedido Inicial	Próximo Pedido	Prob	Cache Time
/people	/people/1	60,8%	0
/people/1	/people/2	73,4%	10
/people/2	/people/3	71,5%	5
/people/3	/people/4	81,9%	15
/people/4	/teams	89,6 %	15
/teams	/offices	78,89%	27
/offices	/united	71,5%	60
/united	/delta	48,2%	61
/delta	/people	59,8%	63

Estes resultados indicam o pedido mais provável de surgir após um determinado pedido inicial, e a respetiva probabilidade. O cache time indica o tempo que cada pedido pode ser guardado em cache, por exemplo: quando o pedido inicial é "/teams", existe 78,89 % de surgir o pedido "/offices"; e o conteúdo da resposta a "/teams"pode ser guardado em cache durante 27 minutos.

Tendo em conta o contexto mostrado em 4.2 e 4.3 podemos concluir que este resultado faz todo o sentido e está em sintonia com os fluxos que foram aplicados.

Para o pedido "/people", que é efetuado a cada 5 minutos, o tempo de cache é 0 porque em poucos minutos ocorre uma alteração em pelo uma das linhas das tabelas, logo, cada vez que este pedido é efetuado, a resposta raramente é igual à resposta dada anteriormente.

No caso de "/people/1", como é efetuada uma alteração a cada 15 minutos e os pedidos são efetuados a cada 5 minutos (com mais um minuto de delay), então é expectável que a indicação do algoritmo para Cache Time seja de 10 minutos. O mesmo cenário ocorre para "/people/2", "/people/3", "/people/4"e "/teams", com alteração nos respetivos tempos.

É importante destacar que para os pedidos "/offices", "/united"e "/delta", o cache time indicado ronda os 60 minutos devido ao facto de nunca haver uma mudança no conteúdo destes *endpoints*. Portanto, o cache time é o tempo em que a aplicação esteve ativa.

Analisando agora as previsões dos pedidos, corresponde também ao cenário previsto. Confrontando este resultado com a tabela 4.2 no subcapítulo 4.3, este seria o cenário mais expectável.

5.1.2 Tempos de Resposta

5.1.2.1 Tempos de Resposta sem cache

Na tabela 5.2 é possível consultar o tempo de resposta aos diferentes pedidos, em segundos, sem cache.

Tabela 5.2: Tempo de Resposta sem Cache

Pedido	Tempo de Resposta
/people	1,155
/people/1	1,144
/people/2	1.139
/people/3	1,140
/people/4	1,150
/teams	1.149
/offices	1,1529
/united	1,609
/delta	1,743

5.1.2.2 Tempos de Resposta com cache

Na tabela 5.3 encontram-se os tempos de resposta aos diferentes pedidos, em segundos, sem cache.

Tabela 5.3: Tempo de Resposta com Cache

Pedido	Tempo de Resposta
/people	0,120
/people/1	0,063
/people/2	0,066
/people/3	0,093
/people/4	0,056
/teams	0,127
/offices	0,071
/united	0,080
/delta	0,079

Verifica-se que existe uma clara redução entre os tempos de resposta com cache e sem cache em cada um dos pedidos efetuados.

5.1.3 Ciclos de Integração

Nesta fase, o objetivo passa por medir o tempo de integração de um ciclo completo de pedidos, conforme mostra a figura 5.2.

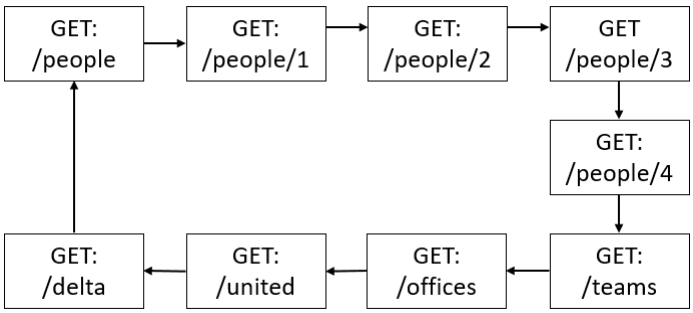

Figura 5.2: Ciclo de Integração

Neste ciclo, iniciado pelo "GET: /people", o pedido seguinte é efetuado assim que é recebida a resposta ao pedido anterior. Este ciclo foi construído tendo por base os resultados preditivos obtidos anteriormente.

Inicialmente calculou-se o tempo de ciclo de integração sem colocar em cache o conteúdo relativo ao pedido seguinte, em nenhum caso. Numa segunda fase, calculou-se o tempo de ciclo de integração colocando em cache o conteúdo relativo ao pedido seguinte, em cada um dos pedidos. Ou seja, para cada pedido efetuado, para além de dar a resposta relativa a esse pedido, era também colocado em cache o conteúdo da resposta ao próximo pedido. Essa resposta permanecia em cache durante 30 segundos, ou até o pedido seguinte previsto ser chamado. Posto isto, algumas das variáveis definidas em 4.7.1 assumem os seguintes valores:

- prob min = 0%
- pred cache time = 30 segundos
- response time min = 0 segundos

Na figura 5.3 apresentam-se os resultados obtidos, com e sem cache.

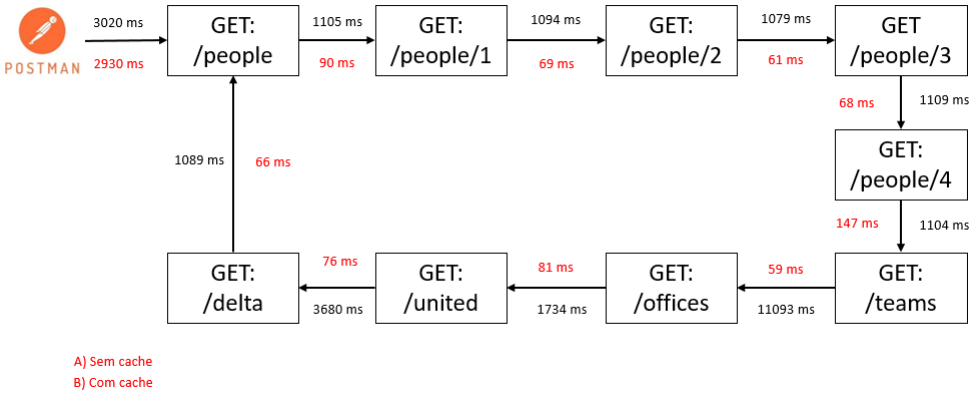

Figura 5.3: Tempos do ciclo de integração: a) Sem Cache b) Com Cache

O primeiro teste iniciou-se às 16:06:41,056h e terminou as 16:07:26,256 do dia 9 de Junho de 2021. O segundo teste decorreu entre as 16:12:36,286h e as 16:12:57,829h do mesmo dia.

Os tempos que estão entre os pedidos, na imagem, representam o tempo necessário para obter a resposta.

Verifica-se que o primeiro teste demorou 44 segundos a ser completado, e desse tempo, 26,125 segundos devem-se ao tempos de espera para obter a resposta de pedidos. O segundo demorou 21,563 segundos, em que apenas 3,647 segundos se devem a tempos de espera.

De seguida, aplicou-se o mesmo ciclo de integração, mas com uma alteração: optou-se por antecipar apenas os pedidos com probabilidade superior a 70% de surgirem. Sendo assim, as seguintes variáveis assumem os valores:

- prob min = 70%
- pred cache time = 30 segundos
- response time min = 0.5 segundos

Na figura 5.4 podemos observar os resultados obtidos.

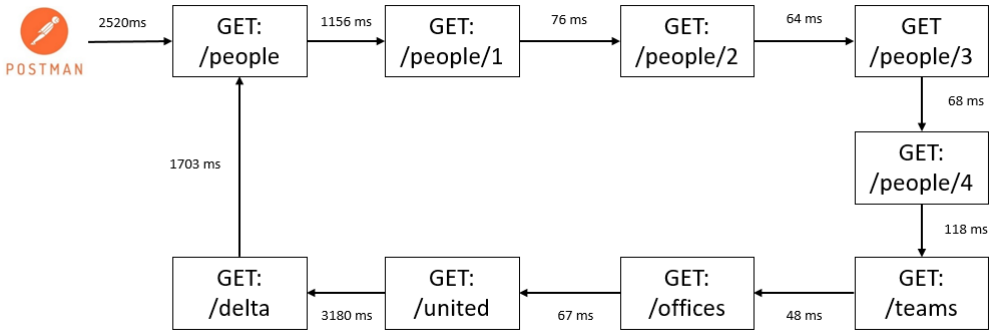

Figura 5.4: Tempos do ciclo de integração, com previsão de pedidos com probabilidade > 70%

Este teste iniciou-se às 17:26:41,125h e terminou às 17:27:05,883h do dia 9 de Junho de 2021. O tempo de ciclo foi 24,758 segundos, em que 9 segundos correspondem a tempos de espera para obtenção de resposta.

No último cenário de teste, a principal alteração ocorreu no ciclo de integração. Ao contrário dos primeiros cenários, em que a sequência de pedidos efetuados foi idêntica à prevista, neste caso isso foi alterado (figura 5.5). O tempo em que o conteúdo da resposta ao pedido previsto era guardado em Cache foi também alterado para 5 segundos. O valor final das variáveis foram os seguintes:

- prob min = 70%
- pred cache time = 5 segundos
- response time min = 0.5 segundos

Na figura 5.5 estão presentes os resultados deste teste:

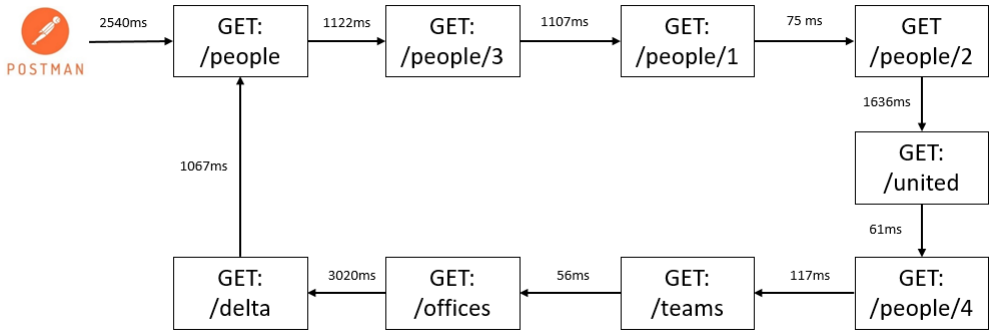

Figura 5.5: Tempos do ciclo de integração alterado, com previsão de pedidos com probabilidade > 70%

Este teste foi iniciado às 18:19:21,627h e terminou às 18:19:45,044 h de 9 de Junho de 2021. O tempo de ciclo foi de 23,417 segundos, em que 10,801 segundos corresponderam a tempos de espera.

5.2 Resultados - DigiB Project

Como referido no capítulo anterior, em 4.6.2, toda a informação relativa à previsão de pedidos está presente em 5 folhas de excel diferentes. A primeira, figura B.1 corresponde à previsão de serviços (topServiceName) e a segunda, figura B.2, corresponde à previsão de pedidos por cliente. Por sua vez, as terceira e quarta folhas, figuras B.3 e B.4, correspondem às previsões de queryParams, de uma visão global, e por cliente. Já a quinta, B.5 corresponde à previsão de serviço e queryParams em simulâneo. Cada folha contém também a qualidade dos algoritmos de *Machine Learning* aplicados (test loss e test accuracy), sendo que folhas correspondentes às previsões por cliente (tanto serviço como queryParam) contêm o resultados dos algoritmos utilizados para cada cliente. Na previsão de serviços, que corresponde à primeira folha, calculou-se também o número de clientes que efetuaram pedidos em cada serviço.

Os resultados obtidos com o algoritmo de Cache, para estimar quanto tempo cada pedido pode ser guardado em cache, está presente numa outra folha Excel, B.6. Aqui, calculou-se também o tempo de cache para cada tipo de pedido (queryParam) presente em cada serviço (topServiceName).

Por último, o método utilizado para o cálculo da capacidade máxima de armazenamento necessário para cache indicou-nos o valor de 15,8 megabytes.

5.3 Discussão de Resultados

Nesta secção será feita uma comparação entre os diversos cenários testados na secção anterior. De seguida serão abordados alguns tópicos não mencionados até aqui, como desvantagens do

predictive caching, e cenários em que o mesmo não é aconselhável.

5.3.1 Comparação entre cenários testados - Deloitte Project

Nos testes feitos em 5.1.2 podemos verificar, de uma forma clara, a redução de tempo de resposta quando o conteúdo está presente em cache. Em pedidos cujo tempo de resposta são sempre superiores a 1 segundo (tabela 5.2), a ferramenta de cache tem uma grande relevância, reduzindo esse tempo para 120 milissegundos, no pior caso. Sendo que no melhor caso o tempo de resposta foi de 56 milissegundos (tabela 5.3).

Confirmado o expectável efeito de cache, resta agora discutir os cenários em que a sua aplicação é viável.

No primeiro cenário calculou-se o tempo de ciclo de integração, sem previsão e consequentemente, sem cache, de acordo com os pedidos previstos. Isto é, começando pelo GET "/people", o pedido seguinte era sempre o pedido previsto pelo algoritmo. O ciclo termina quando GET "/people"volta a ser chamado. Neste caso, o tempo total de ciclo foi de 44 segundos, em que 26,125 segundos se deveram a tempos de espera.

No segundo cenário, em que se recorreu à previsão e cache de todos os pedidos, o tempo de espera foi reduzido para apenas 3,647, cerca de 13% do tempo obtido anteriormente. Este cenário pretende simular os casos em que a confiança na previsão é muito elevada.

No terceiro cenário, em que se optou por prever pedidos com uma certeza superior a 70%, os tempos de espera foram de 9 segundos. No quarto e último cenário, em que a ordem dos pedidos foi alterada, e os 70% de certeza para previsão foram mantidos, os tempos de espera chegaram a 10,8 segundos. O objetivo destes dois cenários era simular e testar o algoritmo nos casos em que a confiança da previsão não é tão alta, e quando a previsão não é a correta, ou seja, quando o pedido seguinte a um outro, não é o previsto pelo algoritmo.

Perante estes resultados consegue-se obter duas conclusões imediatas. Quando o tempo sem alteração do payload é elevado, a ferramenta de cache durante o tempo calculado é algo que vale a pena investir. A diferença de tempos de resposta com e sem cache é notável. No ambiente de teste pode-se considerar 60 minutos como threshold. A segunda conclusão que se consegue tirar num primeiro momento é que quando a confiança da previsão (prob) é elevada, o mecanismo de previsão e antecipação do pedido compensa. A probabilidade de falha é reduzida, e o tempo de resposta muito inferior ao anterior, o que promove uma melhor experiência de utilização para o lado do cliente.

No ambiente de teste, nos terceiro e quarto cenários, quando a confiança da previsão não é tão alta, a ferramenta de previsão e antecipação funciona bem. Mas indica-nos também que, num ambiente mais robusto, como na DigiB, o resultado pode não ser tão positivo.

Quanto o caso de uso é mais robusto, e com conjuntos de dados um elevado número de pedidos, uma solução para combater a incerteza, pode passar por prever os dois pedidos mais prováveis. Ou então prever os pedidos necessários até obter uma determinada confiança. No caso da DigiB, que será analisado no subcapítulo seguinte, optou-se por prever os dois pedidos mais prováveis. Neste sentido, poderão existir casos em que o pedido mais provável tem uma confiança de 45%, o

que é muito pouco. Mas se o segundo mais provável tiver uma confiança de 43%, antecipando os dois consegue-se obter uma confiança de 88% , o que é muito positivo. Desta forma, consegue-se promover uma melhor experiência de utilização, mesmo não tendo uma elevada confiança no primeiro pedido previsto.

5.3.2 Discussão de Resultados - DigiB Project

Relativamente ao resultado de Cache do projeto da DigiB, estamos em condições de afirmar que de facto existem pedidos que podem ser guardados em cache visto que se permaneceram inalterados durante vários dias. Muitos pedidos também têm tempo de Cache de 0 minutos, devido só terem ocorrido apenas uma vez, ou então sofreram alterações constantes.

Já nos resultados preditivos surgem várias dúvidas. No primeiro caso, previsão do serviço, consegue-se obter resultados precisos, e um bom algoritmo de treino. No entanto, acontece que muitas vezes surgem dois pedidos consecutivos iguais, o que não acrescenta valor. Isto pode dever-se a uma atualização da página web, ou ao facto de mudar de página no inventário online. Por este motivo, o facto de prever o segundo pedido mais provável ganha importância.

Nos restantes testes, os resultados são muito incertos. Não foi possível obter resultados muito precisos nem algoritmos de treino muito fiáveis, devido a diversos motivos.

No caso das previsões por cliente, o principal motivo está no volume de dados. A quantidade de dados associados a cada cliente são insuficientes para este tipo de análise. Isto acontece porque a cada sessão iniciada é gerada uma nova chave, diferente da anterior, o que impossibilita a análise temporal do comportamento de cada utilizador na aplicação. Por exemplo, o João, um utilizador da aplicação, ao iniciar a sessão, tem uma chave que é atribuída, e colocada no campo Authorization. De seguida, à medida que vai utilizando a aplicação, é efetuado o registo de todos os pedidos que faz na aplicação, e termina a sessão. No dia seguinte e João volta a iniciar sessão, e tem o mesmo comportamento. No entanto, aqui, a chave atribuída é diferente da anterior. Este fenómeno impede que seja guardado o histórico comportamental do João, guarda apenas de cada sessão, o que resulta em conjuntos de dados muito curtos para o que é pretendido.

Nas previsões dos queryParams, e serviço e queryParam simultaneamente, os resultados são também bastante imprecisos. O algoritmo ainda não é capaz de treinar da forma mais correta e de mostrar um bom desempenho. Acredita-se que com mais tempo e dados para treino é possível obter melhores resultados.

Verificou-se ainda que a capacidade de armazenamento máxima necessária para cache está entre os 15 e os 16 megabytes.

5.3.3 Vantagens do modelo para o Projeto DigiB

Neste sub-capítulo serão abordados alguns pontos onde este projeto poderia acrescentar valor. Estes pontos terão por base os pedidos e respostas efetuadas em cada serviço, ignorando os query parameters e especifidades de cada pedido dentro de cada um deles.

Com base nos resultados obtidos no capítulo anterior podemos considerar 120 ms como referência para tempos de resposta quando o conteúdo está guardado em cache.

Na tabela 5.4 pode-se consultar os tempos médios de resposta e o número de clientes obtidos em cada serviço:

Tabela 5.4: Tempos de Resposta de Serviços

Serviço	**Tempo médio de Resposta (segundos)**	**Número de Clientes**
get:/invoices:bc-documents-papi-config	1,362	139
get:/documents:bc-documents-papi-config	1,965	164
get:/user-documents:bc-documents-papi-config	0,464	415
get:/invoices/(id):bc-documents-papi-config	2,603	20
get:/user-documents/upload-token:bc-documents-papi-config	0,26	91
get:/documents/(id):bc-documents-papi-config	1,5732	21
get:/user-documents/(id):bc-documents-papi-config	0,423	23
get:/coaDocuments/list:bc-documents-papi-config	0,169	41
post:/user-documents:application/json:bc-documents-papi-config	0,230	2

Verifica-se que nenhum dos tempos de resposta é inferior à resposta dada quando o conteúdo está em cache (120 milissegundos ou 0,12 segundos). No entanto nos serviços "get:/coaDocuments/list:bc-documents-papi-config", "post:/user-documents:application/json:bc-documents-papi-config"e "get:/user-documents/upload-token:bc-documents-papi-config"a diferença, aparentemente, não é significativa, principalmente neste último.

O número de clientes em cada serviço indica-nos qual o serviço mais solicitado. O serviço "get:/user-documents:bc-documents-papi-config"é claramente o que afeta mais clientes, seguindo-se "get:/documents:bc-documents-papi-config"e "get:/invoices:bc-documents-papi-config". Daqui é possível retirar que estes devem ser os serviços prioritários para cache.

Se associarmos estes tempos médios de resposta ao números de pedidos em cada serviço, e compararmos esse resultado com o que seria obtido com a previsão (e antecipação) ou algoritmo de cache do mesmo, tendo por base o valor referência de 120 milissegundos (0,120 segundos), obtemos os seguintes resultados presentes na tabela 5.5:

Tabela 5.5: Tempos de Espera nos Serviços

Serviço	Número de Pedidos	Tempos de Espera	
		Sem Cache	Com Cache
get:/invoices:bc-documents-papi-config	590	803,55	70,8
get:/documents:bc-documents-papi-config	908	1784,41	108,96
get:/user-documents:bc-documents-papi-config	3731	1726,04	447,72
get:/invoices/(id):bc-documents-papi-config	43	111,93	5,16
get:/user-documents/upload-token:bc-documents-papi-config	140	37,07	16,8
get:/documents/(id):bc-documents-papi-config	55	86,53	6,6
get:/user-documents/(id):bc-documents-papi-config	44	18,62	5,28
get:/coaDocuments/list:bc-documents-papi-config	130	22,01	15,6
post:/user-documents:application/json:bc-documents-papi-config	4	0,92	0,48

Com estes resultados é possível verificar a influência que a ferramenta de cache tem nos tempos de espera, especialmente nos serviços get:/invoices:bc-documents-papi-config, get:/documents:bc-documents-papi-config e get:/documents/(id):bc-documents-papi-config. Até mesmo no serviço de get:/user-documents:bc-documents-papi-config, em que a diferença de tempo de espera com e sem cache é de apenas 33 milissegundos, quando aplicamos o cache em grande escala, a diferença é notável.

Isto indica que existe muito potencial de redução, e aplicar mecanismos de previsão e/ou cache que permitam este efeito acrescenta, de facto, muito valor.

Analisando agora, de uma forma superficial, os resultados de cache por serviço, consegue-se obter os valores presentes na tabela 5.6.

Esta tabela indica que para os diferentes pedidos feitos no serviço get:/invoices:bc-documents-papi-config (por exemplo), o tempo mínimo de cache obtido é de 0 minutos, o máximo de 42992 minutos, etc. Ou seja, existe um ou mais pedidos que não podem ser guardados em cache visto que o payload muda constantemente, e existem outros pedidos que podem ser guardados em cache durante 42992 minutos (29 dias), por isso obtém-se um valor muito elevado no desvio padrão. Este resultado não permite retirar conclusões claras sobre o que se sucede neste serviço, sendo por isso, necessária uma análise profunda a cada pedido realizado nele. O mesmo acontece em muitos outros serviços presentes na mesma tabela. No entanto, os serviços get:/coaDocuments/list:bc-documents-papi-config e get:/user-documents/(id):bc-documents-papi-config são exceção, e aí é possível retirar alguma conclusão.

No primeiro caso tem-se um valor mínimo de tempo de cache 7328 minutos (aproximandamente 122 horas, que correspondem a 5 dias). Isto significa que, nos pedidos feitos neste serviço,

o valor mínimo de tempo de cache que obtemos foi de 5 dias. A mediana encontra-se em 8295,257 minutos, que corresponde a 5 dias e 18 horas. O desvio padrão neste caso é de 864,67 minutos, ou seja, 14 horas. Verifica-se também uma proximidade entre estes valores e os valores máximo e médio. Este resultado indica que não há muita variabilidade nos valores, e que o conteúdo presente neste serviço pode ser alvo de cache durante um período de tempo significativo, tendo em conta o contexto em que foi aplicado.

No segundo caso, get:/user-documents/(id):bc-documents-papi-config, o desvio padrão é bastante alto, cerca de 12 dias (17798), o que mostra a grande variabilidade de valores nos tempos de cache deste serviço. No entanto o facto do valor mínimo ser 5807 minutos, o que corresponde a aproximadamente 4 dias, sugere que há aqui potencial de cache.

Em relação ao valor obtido para armazenamento para cache, de 15,8 megabytes, é muito curto, logo não será um ponto a ter conta caso se pretenda implementar este método no projeto.

Tabela 5.6: Tempos de Resposta de Serviços

Serviço	Tempo de Cache (minutos)				
	Média	Mediana	Mínimo	Máximo	Desvio Padrão
get:/invoices:bc-documents-papi-config	1523,169	0,658	0	42992,19	6231,53
get:/documents:bc-documents-papi-config	1911,881	1,372	0	37227,04	5648,379
get:/user-documents:bc-documents-papi-config	621,3	0,95	0	20213,92	2550,116
get:/invoices/(id):bc-documents-papi-config	0,06	0	0	0,25	0,124
get:/user-documents/upload-token:bc-documents-papi-config	21228,73	21228,73	21228,73	21228,73	0
get:/documents/(id):bc-documents-papi-config	5,47	0	0	21,86	10,93
get:/user-documents/(id):bc-documents-papi-config	16455,18	8471,99	5807,01	43069,73	17798
get:/coaDocuments/list:bc-documents-papi-config	8295,257	8563,287	7328,421	8994,062	864,67
post:/user-documents:application/json:bc-documents-papi-config	1264,86	1264,86	1264,86	1264,86	0

5.3.4 Prós e Contras

Quando confrontados com os resultados obtidos no ambiente de teste, os mesmos foram muito positivos, e de facto, quando a confiança é alta, o cache e previsão de pedidos é algo que pode ser muito útil a qualquer organização.

No entanto, com a DigiB, com dados reais, em que muitos dos resultados não são limpos e de conclusão imediata, surgem bastantes dúvidas.

Quando a previsão e o cache é possível, o sistema consegue dar respostas mais mais rápidas, gerir e otimizar invocações à *cloud*, etc. No entanto, dependendo do caso de uso em questão, para este mecanismo ser possível, é necessário espaço de armazenamento e maior processamento, visto que são geridos mais pedidos do que são realmente necessários. Para além de que é necessário investir em capacidade de análise de dados e *Machine Learning*, e garantir que exite uma forma "standard"de registo dos dados, para este método poder ser universal na organização. Noutro sentido, existe também o risco da informação dada ao cliente não ser a correta. Tendo por base que um dos principais objetivos do projeto é guardar em cache informação estática, ao ser dada a resposta que está guardada em cache e não nos *Target systems*, é impossível prever com 100% quando é que a mesma será alterada.

Capítulo 6

Trabalho Futuro e Conclusões

O objetivo do projeto era verificar a utilidade de uma nova forma de fazer integração recorrendo a Predictive Caching. Foi possível criar algoritmos de cache, para obter os tempos sem alteração de payload dos pedidos, e de previsão, para identificar e prever fluxos de integração.

No capítulo anterior, no ambiente de teste conseguiram-se obter resultados muito claros e positivos. Já perante os resultados obtidos no projeto da DigiB, fica a ideia de que há ainda muito trabalho neste sentido para ser desenvolvido.

Tendo em conta os resultados obtidos no Deloitte Project é possível concluir que quando a confiança dos algoritmos preditivos é alta, o Predicitve Caching é algo que acrescenta muito valor ao sistema. Concluí-se também que quando se mantém estática durante um longo período de tempo, a ferramenta de cache pode acrescentar valor e mostrar utilidade ao sistema, não só ao nível da redução de tempos de resposta como forma otimizada como os *endpoints* (muitas vezes em sistemas em *cloud*) são invocados.

Os resultados obtidos no DigiB Project não são tão conclusivos, mas há pontos onde este método pode ser aplicado, o que nos indica que existe um potencial. O feedback das equipas responsáveis pelo projeto foi positivo, referindo também que este é o "tipo de ideia e inovação que os clientes apreciam". A maneira como foi implementado este projeto sugere também que é possível alargar este conceito a outras indústrias.

De uma maneira geral, quando o conteúdo está em cache, ou porque foi lá guardado, ou porque a resposta aos pedido foram antecipadas, os tempos de resposta obtidos são muito positivos, apresentando uma redução drástica em relação aos casos "standard". Isto promove integrações mais rápidas, otimizações do uso dos sistemas em *cloud* e consequentemente uma melhor experiência de utilização, o que vai de encontro aos objetivos definidos no início do projeto.

Ficou então provado que esta nova forma de integração é possível e pode acrescentar muito valor aos sistemas.

6.1 Trabalho Futuro

Como referido anteriormente, os resultados obtidos em contexto real (DigiB) não foram suficientemente conclusivos ao ponto de permitir a tomada de decisões concretas sobre o sistema em questão. Isto indica que há muito trabalho pela frente.

Numa primeira fase, considerar ou acrescentar mais campos de análise nos eventos que são registados pela plataforma de integração, que permitisse manter um registo histórico do cliente; e promover análise mais profunda do payload das respostas (segmentação do conteúdo).

Numa segunda fase, explorar mais a área de análise de dados e *Machine Learning* de maneira a permitir retirar conclusões concretas e precisas quando os algoritmos são aplicados a ambientes robustos e complexos, como: análise temporal mais profunda da cache para obter conclusões como os dias e horas da semana em que o conteúdo é alterado, e ainda adaptar estes resultados a cada fuso horário.

Fazer uma maior filtragem e tratamento dos dados surge também como trabalho futuro, como por exemplo: ignorar pedidos iguais que são consecutivos.

Numa última fase, e já com Predictive Caching implementado, surge outro ponto que abre oportunidades a outros projetos: formas de calcular o rendimento da cache e da previsão. Alguns tópicos interessantes neste contexto podem ser:

- Calcular percentagem de pedidos previstos com sucesso.
- Calcular percentagem de vezes que a informação guardada em cache corresponde à informação presente nos *endpoints*.
- Medir o efeito deste médio na experiência dos utilização serviços intervenientes.

6.2 Conclusão Final

Este projeto foi muito desafiante, tanto a nível pessoal como profissional, que me permitiu crescer em todos os sentidos, e exceder as minhas expectativas.

Ficou provado que esta nova forma de integração é possível, e pode trazer muito valor para as organizações. Fica ainda a sensação de que há muito trabalho a desenvolver no futuro, o que não é necessariamente negativo, visto que "abre portas"a outros projetos, e mais por onde inovar.

Em termos profissionais, este projeto e o estágio na Deloitte Portugal permitiram-me trabalhar com várias tecnologias que não conhecia, e aprender muito com todos os profissionais da equipa da Deloitte e não só. A oportunidade de poder aplicar o projeto em dados reais de cliente, e consequentemente trabalhar com uma equipa internacional foi um grande promotor do meu crescimento. A responsabilidade e ao mesmo tempo privilégio de representar uma empresa como a Deloitte, e crescer com este processo é algo que será sempre relembrado por mim e marcado como um grande passo no meu percurso profissional.

Em termos pessoais, o estágio permitiu-me conhecer pessoas fantásticas. Ensinou-me a viver com pressão, problemas técnicos e frustração da melhor maneira, o que se deve a um apoio sempre

muito forte e a um trabalho de equipa fantástico, tanto por parte das equias da Deloitte, como por parte de ambos os meus orientadores.

Anexo A

Nas figuras seguintes estão presentes alguns dos fluxos implementados no Mulesoft para provocar alterações nos target Systems do Deloitte Project (A.3), para fazer pedidos automaticamente (A.4 e para obter uma boa quantidade de dados para análise. As figuras A.1 e A.2 mostram nos os fluxos principais do projeto, responsáveis por recolher o pedido do cliente, obter e enviar a resposta.

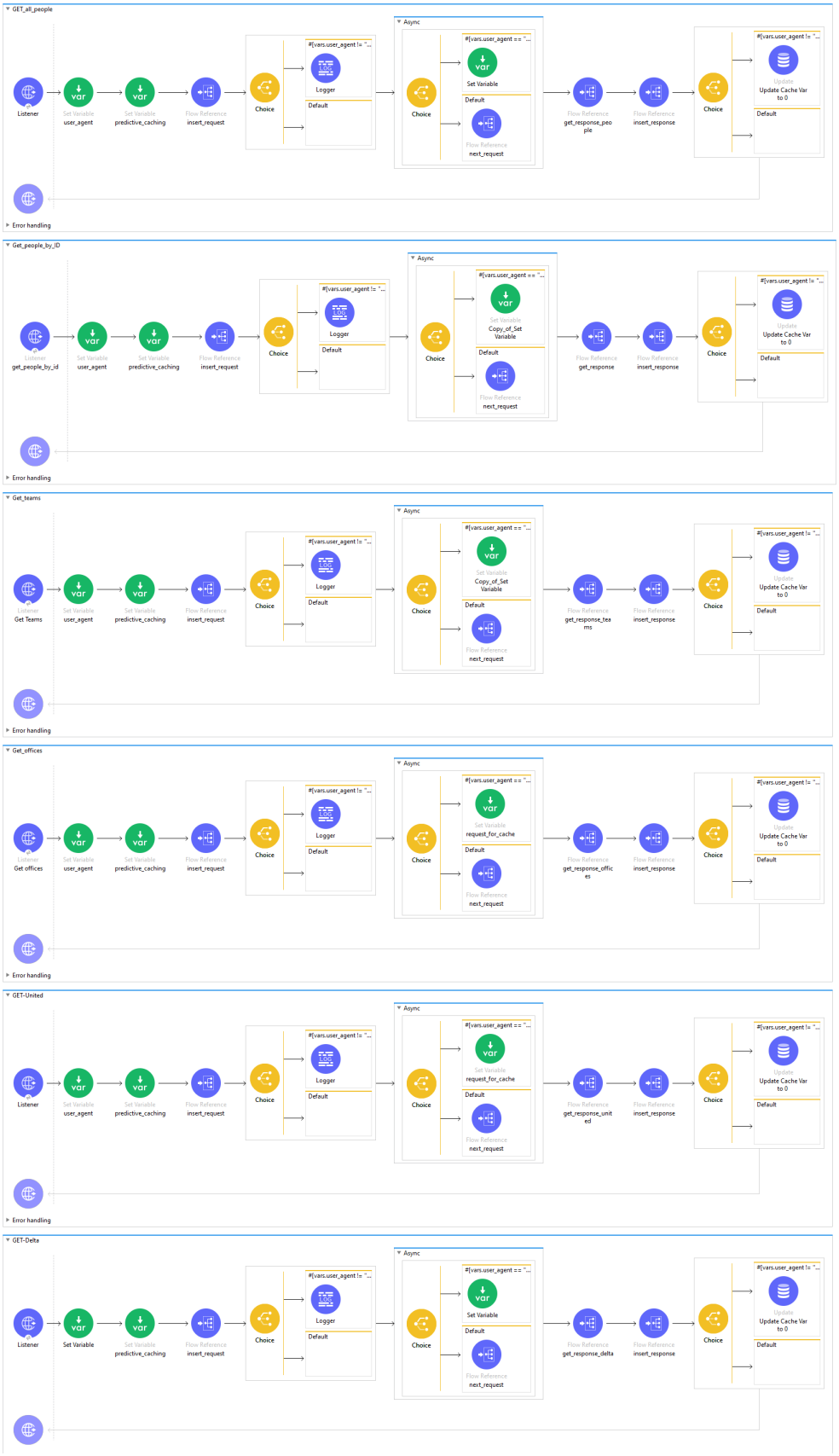

Figura A.1: Mulesoft - Flow Principal

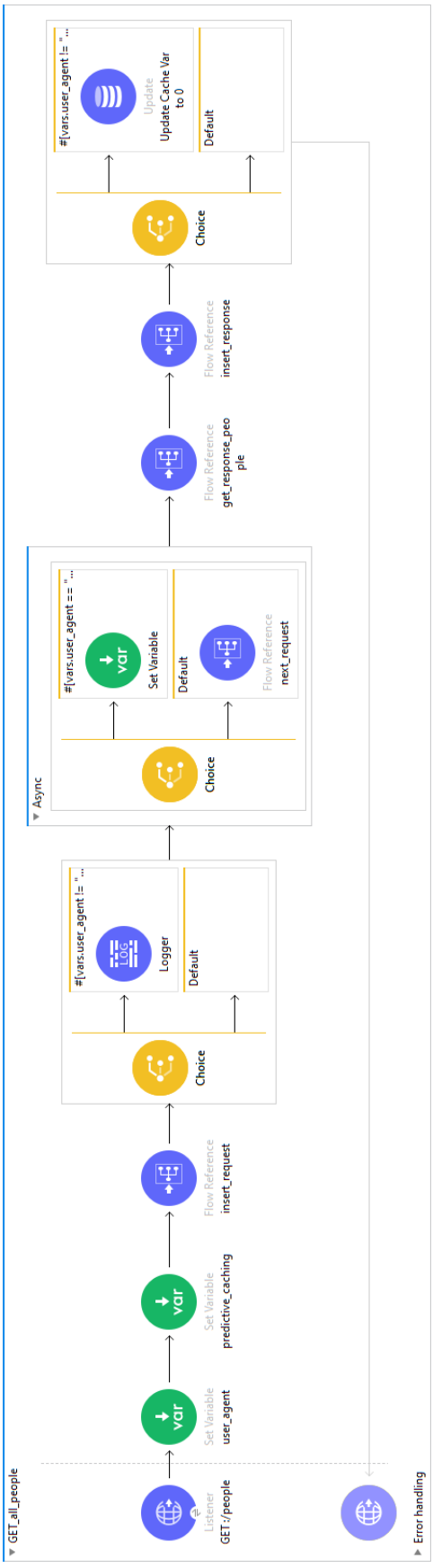

Figura A.2: Mulesoft - Get People

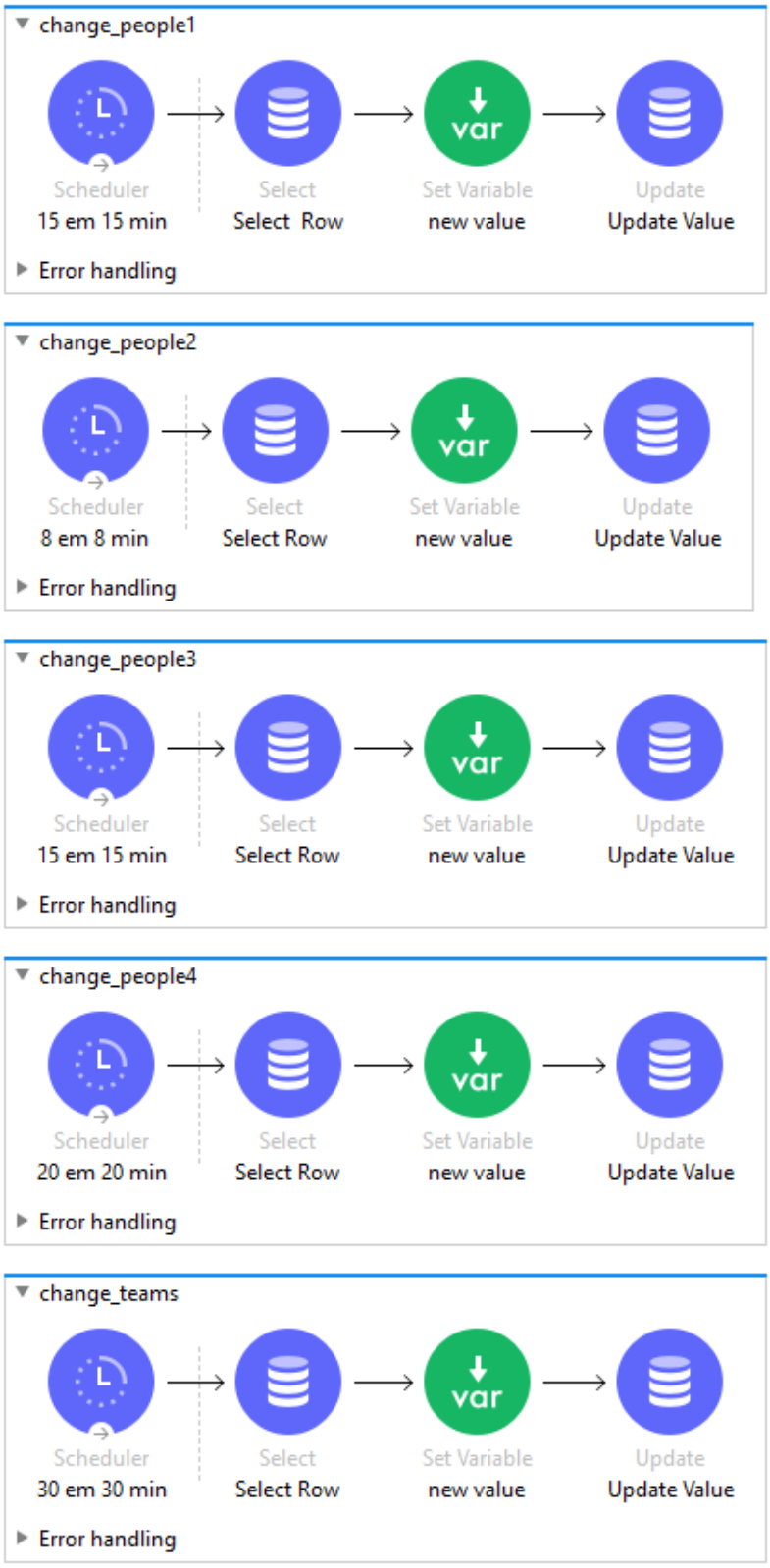

Figura A.3: Mulesoft - Change Things Flow

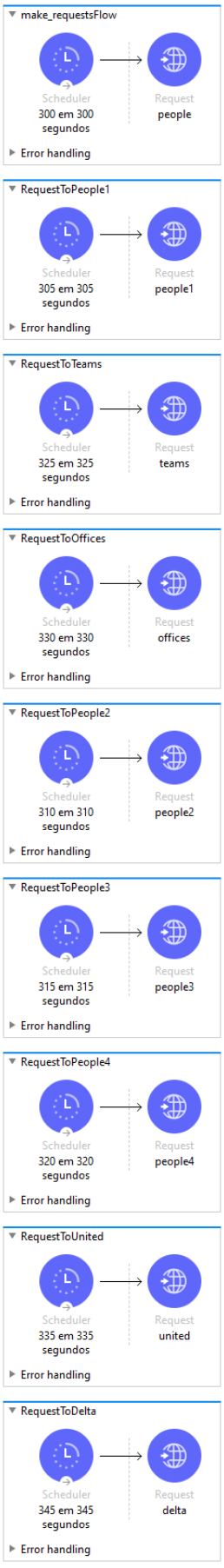

Figura A.4: Mulesoft - Make Requests Flow

Anexo B

Nas figuras seguintes estão presentes os resultados obtidos no caso de uso DigiB Project. A figura B.1 mostra o resultado dos serviços previstos após a chamada de cada um, tendo em conta todas os pedidos efetuados na API. A figura B.2 mostra alguns dos serviços previstos para cada cliente/sessão iniciada. A figura mostra alguns dos queryParams B.3 previstos após a chamada de cada tipo de pedido presente no sistema. A figura B.4, mostra uma parte os queryParams previstos para cada cliente/sessão iniciada. A figura B.5 representa uma junção das folhas 1 e 3, em que mostra uma parte dos serviços e queryParams previstos para cada tipo de combinação presente na API.

Por sua vez, a figura B.6 dá-nos o uma parte do resultado do algoritmo de cache, onde para cada tipo de pedido (distinguido pelos queryParams) presente em cada serviço, indica o tempo de cache (cache time), cache var (1 se faz sentido guardar em cache e 0 caso contrário), finish dates (instante de tempo onde foi detetada uma alteração de payload), numero de pedidos de cada tipo, tempo de resposta médio e desvio padrão da amostra. Indica também uma visão global para cada tipo de serviço, onde mostra a média, mínimo, máximo, mediana e desvio padrão do tempo de cache dos pedidos feitos em cada um.

Initial Service	1 - Next Service	1 - Accuracy	2 - Next Service	2 - Accuracy	Number o	Response Time	Number of Clients			Test Loss	0,78749
										Test Accur	0,792774
get:\user-documents\upload-token:bc-documents-papi-config	get:\user-documents:bc-documents-papi-config	0,470816374	get:\user-documents\upload-token:bc-documents-pap	0,364004374	140	0,264785714	91				
get:\user-documents:bc-documents-papi-config	get:\user-documents:bc-documents-papi-config	0,814348876	get:\coaDocuments\list:bc-documents-papi-config	0,072821409	3731	0,462622085	415				
get:\coaDocuments\list:bc-documents-papi-config	get:\user-documents:bc-documents-papi-config	0,983925581	get:\coaDocuments\list:bc-documents-papi-config	0,005825831	130	0,1693	41				
get:\documents:bc-documents-papi-config	get:\documents:bc-documents-papi-config	0,651590168	get:\user-documents:bc-documents-papi-config	0,192506284	908	1,965211454	164				
get:\invoices:bc-documents-papi-config	get:\invoices:bc-documents-papi-config	0,727584362	get:\user-documents:bc-documents-papi-config	0,104509242	590	1,361949153	139				
get:\user-documents\{id}:bc-documents-papi-config	get:\user-documents:bc-documents-papi-config	0,53392309	get:\user-documents\{id}:bc-documents-papi-config	0,230956569	44	0,423181818	23				
get:\invoices\{id}:bc-documents-papi-config	get:\invoices:bc-documents-papi-config	0,505147576	get:\invoices\{id}:bc-documents-papi-config	0,43554312	43	2,603	20				
get:\documents\{id}:bc-documents-papi-config	get:\documents:bc-documents-papi-config	0,429927498	get:\documents\{id}:bc-documents-papi-config	0,407716781	55	-125,1974909	21				
post:\user-documents:application\json:bc-documents-pap	get:\user-documents:bc-documents-papi-config	0,386022925	get:\invoices:bc-documents-papi-config	0,266616434	4	0,23025	2				

Figura B.1: Predictive Results - folha 1

Initial Service	1 - Next Service	1 - Accuracy	2 - Next Service	2 - Accuracy	Client	Number o	Response Time	Test Loss	Test Accur	Client
get:\user-documents\upload-t	get:\user-documents:bc-documents-papi-config	0,634009838	get:\user-documents\upload-token:bc	0,365990222	b325e8f7a	14	0,169	0,540914	1	b325e8f7ab1fef5b7dae9760d52843756e420325
get:\user-documents:bc-docur	get:\user-documents:bc-documents-papi-config	0,582215548	get:\user-documents\upload-token:bc	0,417784393	b325e8f7a	14	0,476231	1,301947	0,45	0bb0742b5b44cfa91f4a162f1eaf7a9ec9be182f
get:\coaDocuments\list:bc-doc	get:\coaDocuments\list:bc-documents-papi-con	0,275242358	get:\user-documents:bc-documents-pa	0,268312603	0bb0742b	40	0,201857	1,179319	0	40970cff76c37227c207029a9c084f20dd484429
get:\user-documents:bc-docur	get:\user-documents:bc-documents-papi-config	0,288919657	get:\coaDocuments\list:bc-documents	0,255928218	0bb0742b	40	0,499536	0,60312	1	ce96eebf3445a310dbcabcf388f9adb9a3be25c3
get:\documents:bc-documents	get:\coaDocuments\list:bc-documents-papi-con	0,334528357	get:\user-documents:bc-documents-pa	0,273918927	0bb0742b	40	2,381667	1,226935	0,75	4704fe67624c64728dbc8cfeddf12e920713b468
get:\user-documents\upload-t	get:\coaDocuments\list:bc-documents-papi-con	0,308986723	get:\user-documents:bc-documents-pa	0,2859478	0bb0742b	40	0,399	0,642761	0,75	cbd24d7d94f1ea3249d3acb3ba401d96755a3fe4
get:\user-documents:bc-docur	get:\user-documents:bc-documents-papi-config	1	none	0	06fa9d1d5	1	0,999	1,711316	0,285714	9df4b9716f248cfe5e02bfbfcf8f0ca2f7390011
get:\user-documents:bc-docur	get:\coaDocuments\list:bc-documents-papi-con	0,397438616	get:\user-documents:bc-documents-pa	0,336052597	40970cff7	8	0,3854	0,696009	0	d4535fbb852bd918acf538cadf30a1e0e43108eb
get:\coaDocuments\list:bc-doc	get:\coaDocuments\list:bc-documents-papi-con	0,402364165	get:\user-documents:bc-documents-pa	0,324433804	40970cff7	8	0,18	0,604876	1	3c4f8c729557aeabda0f97b2757d0cd1511fbb3f
get:\invoices:bc-documents-pa	get:\coaDocuments\list:bc-documents-papi-con	0,376813233	get:\user-documents:bc-documents-pa	0,326165974	40970cff7	8	1,5205	0,516392	1	90f8ee306de47b90700c28930aaae6e0e6c95a62
get:\coaDocuments\list:bc-doc	get:\user-documents:bc-documents-papi-config	0,539632976	get:\coaDocuments\list:bc-documents	0,460367024	ce96eebf3	8	0,169	0,618393	0,75	0a395c00a2db522ba24eeebc5384104fb8398e53
get:\user-documents:bc-docur	get:\user-documents:bc-documents-papi-config	0,547101796	get:\coaDocuments\list:bc-documents	0,452898264	ce96eebf3	8	0,402143	0,501834	1	2b4cfefc243fefa9109c17c550addc18fd09e484
get:\user-documents:bc-docur	get:\user-documents:bc-documents-papi-config	0,317785025	get:\user-documents\upload-token:bc	0,267227918	4704fe676	32	0,45516	1,004083	0,5	87bf1b19ee1f5d2ff9323a80748279484dbfa8de
get:\coaDocuments\list:bc-doc	get:\user-documents\(id):bc-documents-papi-cc	0,279009879	get:\user-documents:bc-documents-pa	0,277801931	4704fe676	32	0,203	1,122619	0	99ff4b152c24781f6d1a233682bbd98bd8f19052
get:\user-documents\upload-t	get:\user-documents\(id):bc-documents-papi-cc	0,285414875	get:\user-documents:bc-documents-pa	0,26225847	4704fe676	32	0,249	0,708706	0	be701b291c35228019380d9a132c6b7380b12a54
get:\user-documents\(id):bc-d	get:\user-documents:bc-documents-papi-config	0,312007755	get:\user-documents\upload-token:bc	0,240492746	4704fe676	32	0,513	0,651069	1	6d02a8c340321efd87cd8bc3edce652ecc7ce034
get:\user-documents:bc-docur	get:\user-documents:bc-documents-papi-config	0,524603128	get:\coaDocuments\list:bc-documents	0,475396872	cbd24d7d	7	0,441	0,554701	1	a5aa159f3416de03e32da17fee5a020bab095cd9
get:\coaDocuments\list:bc-doc	get:\user-documents:bc-documents-papi-config	0,584374189	get:\coaDocuments\list:bc-documents	0,415625811	cbd24d7d	7	0,1495	0,493828	1	394e51bc053cf38615db3103fcb5e01c94b77ae5
get:\documents:bc-documents	get:\documents:bc-documents-papi-config	1	none	0	62ffd9773	1	2,311	0,657281	0,857143	cb7df6b2c4a54429f55ff181f202674a3fc0034f
get:\invoices:bc-documents-pa	get:\user-documents:bc-documents-papi-config	0,196891531	get:\invoices\(id):bc-documents-papi-	0,177865669	9df4b9716	14	1,431	0,522916	1	6e4774db4f9a965dd261f612a2dbdfcd305a82ea
get:\invoices\(id):bc-documen	get:\invoices\(id):bc-documents-papi-config	0,186920762	get:\user-documents\upload-token:bc	0,168633968	9df4b9716	14	3,156	0,6807	0,5	b1eae3c2f21f5f6c66d8741f53510a779f1ceff0
get:\documents:bc-documents	get:\invoices\(id):bc-documents-papi-config	0,179235891	get:\invoices:bc-documents-papi-confi	0,174089372	9df4b9716	14	1,627667	0,600202	1	2b47ef0a7fa8436b6614acc4b3548c870c1bfb84
get:\coaDocuments\list:bc-doc	get:\user-documents:bc-documents-papi-config	0,212452263	get:\invoices\(id):bc-documents-papi-	0,178489983	9df4b9716	14	0,147	0,469652	1	5f8e16b1582db79ce677d9a54b9a3b902b1236f1
get:\user-documents:bc-docur	get:\user-documents:bc-documents-papi-config	0,191910088	get:\coaDocuments\list:bc-documents	0,177749917	9df4b9716	14	0,493	0,936449	0,785714	a6ca0ae3d4c3915aa24a8cb37b2557113900c9fd
get:\user-documents\upload-t	get:\invoices\(id):bc-documents-papi-config	0,183259845	get:\user-documents\upload-token:bc	0,176929995	9df4b9716	14	0,344	1,036489	0,4	00cd5c63f9341067f99abf7f58f2d55eb6c023d9
get:\coaDocuments\list:bc-doc	get:\coaDocuments\list:bc-documents-papi-con	0,501428902	get:\user-documents:bc-documents-pa	0,498571128	d4535fbb8	2	0,178	0,573978	0,666667	7db09502ed3878bf748cddba8d2b0b965987af30
get:\user-documents:bc-docur	get:\coaDocuments\list:bc-documents-papi-con	0,538986325	get:\user-documents:bc-documents-pa	0,461013675	d4535fbb8	2	0,461	0,615267	0,75	e7d253dbd5f83c5872bc3827c5bf4f7c2654599e
get:\coaDocuments\list:bc-doc	get:\user-documents:bc-documents-papi-config	0,54866761	get:\coaDocuments\list:bc-documents	0,45133239	3c4f8c729	3	0,565	1,338314	0,727273	7fa727621d97b00f7b0ecfccfc478c9ae827d35d
get:\user-documents:bc-docur	get:\user-documents:bc-documents-papi-config	0,543628454	get:\coaDocuments\list:bc-documents	0,456371486	3c4f8c729	3	0,449	0,627607	0,916667	92288ed3982877e37d781bb2cb3eabd2a5b8b3c5
get:\documents:bc-documents	get:\invoices:bc-documents-papi-config	0,596669674	get:\documents:bc-documents-papi-cc	0,403330356	90f8ee306	2	2,334	1,049221	0,5	7a8db0c64b8d953589d5cc810e8b55d276350429
get:\invoices:bc-documents-pa	get:\invoices:bc-documents-papi-config	0,542688191	get:\documents:bc-documents-papi-cc	0,457311779	90f8ee306	2	5,921	0,536606	0,75	fde894fa20ec37228f38d52aebe29620a04bd6db
get:\user-documents:bc-docur	get:\user-documents:bc-documents-papi-config	0,58088249	get:\coaDocuments\list:bc-documents	0,41911751	0a395c00a	7	0,513833	0,637764	0,75	[illegible]

Figura B.2: Predictive Results - folha 2

Initial Request	1 - Next Request	1 - Accuracy	2 - Next Request	2 - Accuracy	Number o	Response Time				Test Loss	8,948373
										Test Accu	0,074035
{}	{}	0,365403742	{'type': 'Cart', 'order': 'Q\	0,048991922	140	0,263826					
{'type': 'Cart', 'order': '5rl0Rrn	{'type': 'Cart', 'order': '5rl0Rrn	0,535611749	{'companyCode': '1241',	0,214199841	9	0,468889					
{'documentKeys': '', 'compan	{'type': 'Order', 'order': 'YXBN	0,398690492	{'order': 'ZvAA46xNVXlU	0,256727695	38	0,164632					
{'order': '2O9uAkhBOShX5Nc	{'order': 'k-X4QPuynOJI', 'incl	0,953657508	{'order': 'NC63yL0vUG6X	0,017844573	1	0,416					
{'order': 'k-X4QPuynOJI', 'incl	{'order': 'k-X4QPuynOJI', 'incl	0,633492231	{'order': 'NC63yL0vUG6X	0,07344792	10	0,5253					
{'documentKey': '0000000000	{'documentKey': '0000000000	0,409585148	{'documentKeys': '', 'con	0,404049277	5	2,3706					
{'order': 'Ctl5GJjh59F-i-FbZBx	{'documentKeys': '', 'compan	0,144220918	{'order': 'gl3UTZyKwCik',	0,127000466	6	0,503					
{'order': '!Sm2Tn6t7aWw', 'in	{'sourceIds': '9020100088,902	0,980804503	{'type': 'Order', 'order': 'i	0,00650336	1	0,999					
{'order': 'lwJyackUIIrS0h6CEC	{'order': 'lwJyackUIIrS0h6CEC	0,691821337	{'sourceIds': '9020103990	0,29199034	7	0,399143					
{'documentKeys': '', 'compan	{'type': 'Order', 'order': 'srTdr	0,174413472	{'type': 'Order', 'order': 'l	0,153170422	16	0,206625					
{'order': 'l3dMQUW57ycF', 'in	{'documentKeys': '', 'compan	0,44528091	{'sourceIds': '9150026858	0,341608673	5	0,3896					
{'sourceIds': '9020103990,902	{'sourceIds': '9020103990,902	0,99899596	{'sourceIds': '903005594	0,00033433	2	1,5205					
{'order': 'ZvAA46xNVXlUEOHi	{'order': 'ZvAA46xNVXlUEOHi	0,813447297	{'documentKeys': '', 'con	0,057548102	64	0,464641					
{'type': 'Order', 'location': '/e	{'order': 'ZvAA46xNVXlUEOHi	0,992112756	{'order': 'Ctl5GJjh59F-i-F	0,00383174	4	0,48925					
{'companyCode': '1246', 'doc	{'sourceIds': '9020103990,902	0,396241039	{'documentKeys': '', 'con	0,223075077	1	2,311					
{'sourceIds': '9150026858,915	{'location': '/en-MY/connect/	0,449871242	{'documentKey': '000000	0,27989459	4	1,5125					
{'location': '/en-MY/connect/{	{'location': '/en-MY/connect/	0,416364908	{'sourceIds': '915002584(	0,226652443	7	4,972538					
{'documentKey': '0000000000	{'sourceIds': '9150026858,915	0,963864565	{'type': 'Cart', 'order': 'Ve	0,018962132	1	2,255					
{'order': 'nl79TooyIrGJ1DxNQ {}		0,984605193	{'type': 'Cart', 'order': 'Q\	0,006667393	1	0,499					
{'documentKey': '0000000000	{'documentKey': '0000000000	0,986648142	{'sourceIds': '917002940(	0,008620244	1	1,315					
{'documentKey': '0000000000	{'documentKey': '0000000000	0,985979378	{'type': 'Cart', 'order': 'dL	0,004683935	1	1,313					
{'order': 'i2lulou-XNmoviipET	{'documentKeys': '', 'compan	0,983025551	{'type': 'Cart', 'order': 'Ck	0,00543361	1	0,597					
{'order': 'DdHr3JN8L8FP4otK' {}		0,988864899	{'type': 'Cart', 'order': 'Q\	0,003699674	1	0,383					

Figura B.3: Predictive Results - folha 3

Initial Request	1 - Next Request	1 - Accuracy	2 - Next Request	2 - Accuracy	Client	Number of Re	Response Time	Test Loss	Test Accur	Client
None	{'type': 'Cart', 'order': '	0,523328781	{}	0,270089	b325e8f7ab1fef5b7dae97604d5284375	14	0,169	1,062838	0,428571	b325e8f7ab1fef5b7dae97604d5284375
None	{'type': 'Cart', 'order': '	0,570094466	{}	0,220259652	b325e8f7ab1fe	14	0,468888889	1,995316	0,25	0bb0742b5b44cfa91f4a162f1eaf7a9e
None	{'type': 'Cart', 'order': '	0,46500507	{}	0,287712216	b325e8f7ab1fe	14	0,49275	1,42821	0,5	4097ccff76c37227c207029a9c084f20
None	{'order': 'Ctl5GJjhS9F-i	0,257341743	{'order': '2O9uAKf	0,167936593	0bb0742b5b44(	40	0,201857143	0,898764	1	ce96eebf3445a310dbcabcf388f9ad
None	{'order': 'K-X4QPuynOJ	0,232250124	{'order': 'ZvAA46x	0,159806758	0bb0742b5b44(	40	0,416	1,122499	0,8125	4704fe67624c647238dbc8cfeddf12e
None	{'order': 'k-X4QPuynOJ	0,199193448	{'order': 'ZvAA46x	0,15106231	0bb0742b5b44(	40	0,5253	0,526151	0,75	cbd24d7d94f1ea3249d3acb3ba401d9
None	{'order': 'Ctl5GJjhS9F-i	0,160769567	{'documentKey': 'I	0,160671249	0bb0742b5b44(	40	2,381666667	2,679813	0	9df4b9716f248cfe5e02bfbfcf80ca2
None	{'order': 'k-X4QPuynOJ	0,210018635	{'order': 'ZvAA46x	0,176158324	0bb0742b5b44(	40	0,503	0,025396	1	d4535fbb852bd918acf538cadf30a1e
None	{'order': 'k-X4QPuynOJ	0,20781894	{'order': 'Ctl5GjjhS	0,165832505	0bb0742b5b44(	40	0,399	0,71256	0,5	3c4f8c729557aeabdaf0f97b2757d0c
None	{'order': 'ZvAA46xNVX	0,180910319	{'order': 'k-X4QPu	0,151236027	0bb0742b5b44(	40	0,4784	0,104951	1	90f8ee306de47b90700c28930aaae6e
None	{'documentKey': '0000	0,204006091	{'order': 'ZvAA46x	0,146394208	0bb0742b5b44(	40	0,484666667	1,436497	0,5	0a395c00a2db522ba24eeebc5384104
{'order': 'ISmzTn6t7aW	{'order': 'ISmzTn6t7aW	1	none	0	06fa9d1d53ac8	1	0,999	0,073211	1	2b4cfefc243fefa9109c17c550addc18
None	{'order': 'IwJyackUIIrS(	0,79526782	{'documentKeys':	0,157485127	40970cff76c372	8	0,3854	0,82735	0,5	87bf1b19ee1f5d2ff9323a807482794
None	{'order': 'IwJyackUIIrS(	0,834258258	{'documentKeys':	0,10495843	40970cff76c372	8	0,18	0,900726	0,5	99ff4b152c24781f6d1a233682bbd98
None	{'order': 'IwJyackUIIrS(	0,692697644	{'documentKeys':	0,196756035	40970cff76c372	8	1,5205	0,262165	1	be701b291c3522801938 0d9a132c6b7
None	{'order': 'IwJyackUIIrS(	0,715261996	{'order': 'I3dMQUV	0,176175922	ce96eebf3445a	8	0,169	0,500426	1	6d02a8c340321efd87cd8bc3edce652
None	{'order': 'IwJyackUIIrS(	0,569269121	{'order': 'I3dMQUV	0,347306758	ce96eebf3445a	8	0,4335	0,101193	1	a5aa159f34116de03e32da17fee5a02
None	{'order': 'I3dMQUW57)	0,407072335	{'order': 'IwJyackU	0,34914428	ce96eebf3445a	8	0,3896	2,365268	0	15e185e10781cce303d44334700be41
None	{'order': 'ZvAA46xNVX	0,36851269	{'documentKeys':	0,284432739	4704fe67624c6	32	0,45516	0,138342	1	2e68986b43ab80450b1ec404fc45ef13
None	{'order': 'ZvAA46xNVX	0,386095673	{'documentKeys':	0,256380469	4704fe67624c6	32	0,203	0,088971	1	394e51bc053cf38615db3103fcb5e01
None	{'type': 'Order', 'locatic	0,308998913	{'order': 'ZvAA46x	0,307986081	4704fe67624c6	32	0,249	1,58219	0,142857	cb7df6b2c4a54429f55ff181f2026743
None	{'order': 'ZvAA46xNVX	0,338057458	{'documentKeys':	0,246139005	4704fe67624c6	32	0,513	0,968986	0,25	6e4774db4f9a965dd261f612a2dbdfc
None	{'type': 'Order', 'order'	0,529033959	{'companyCode': '	0,470966071	cbd24d7d94f1e	7	0,441	0,871623	0,75	b1eae3c2f21f5f6c66d87415351 0a779
None	{'type': 'Order', 'order'	0,924751461	{'companyCode': '	0,075248539	cbd24d7d94f1e	7	0,1495	0,263843	1	2b47ef0a7fa8436b6614acc4b3548c87
{'companyCode': '12(	{'companyCode': '1246	1	none	0	62ffd9773d096	1	2,311	1,46041	0,166667	5f8e16b1582db79ce677d9a54b93a3b

Figura B.4: Predictive Results - folha 4

Initial Request	Initial Service	1 - Next Request	1 - Next Service	1 - Accuracy	2 - Next Request	2 - Next Service	2 - Accuracy	Number o	Response Time		Test Loss	9,177582
											Test Accur	0,065533
{}	get:\user-documents\	{}	get:\user-documents\{u	0,425790787	{'type': 'Cart', 'orde	get:\user-documents:bc-docu	0,037209135	140	0,264785714			
{'type': 'Cart', 'order': '5rl0Rn	get:\user-documents:bc	{'type': 'Cart', 'orde	get:\user-documents:bc	0,748024523	{'type': 'Order', 'ord	get:\user-documents:bc-docu	0,131302178	9	0,468888889			
{'documentKeys': '', 'compar	get:\coaDocuments\lis	{'type': 'Order', 'ori	get:\user-documents:bc	0,680108428	{'order': 'ZvAA46xN	get:\user-documents:bc-docu	0,193763837	38	0,164631579			
{'order': '2O9uAkhBOShXSN	get:\user-documents:bc	{'order': 'k-X4QPuy	get:\user-documents:bc	0,998011708	{'companyCode': '1	get:\documents:bc-document	0,001987582	1	0,416			
{'order': 'k-X4QPuynOJI', 'inc	get:\user-documents:bc	{'order': 'k-X4QPuy	get:\user-documents:bc	0,874320924	{'documentKeys': '	get:\coaDocuments\list:bc-dc	0,073263086	10	0,5253			
{'documentKey': '000000000	get:\documents:bc-do	{'documentKeys':	get:\coaDocuments\list:	0,611330926	{'documentKey': '0f	get:\documents:bc-document	0,388157427	5	2,3706			
{'order': 'Ctl5GJjhS9F-i-FbZB	get:\user-documents:bc	{'order': 'gl3UTZyK	get:\user-documents:bc	0,372949421	{'documentKeys': '	get:\coaDocuments\list:bc-dc	0,281381249	6	0,503			
{'order': 'lSmzTn6t7aWw', 'in	get:\user-documents:bc	{'sourceIds': '90201	get:\invoices:bc-docum	0,999999046	{'type': 'Cart', 'orde	get:\user-documents:bc-docu	8,83581E-07	1	0,999			
{'order': 'lwJyackUlIrS0h6CE	get:\user-documents:bc	{'order': 'lwJyackU	get:\user-documents:bc	0,638586342	{'order': 'l3dMQUW	get:\user-documents:bc-docu	0,192030728	7	0,399142857			
{'documentKeys': '', 'compar	get:\coaDocuments\lis	{'type': 'Order', 'ori	get:\user-documents:bc	0,173976898	{'type': 'Order', 'ord	get:\user-documents:bc-docu	0,139871344	16	0,206625			
{'order': 'l3dMQUW57ycF', 'ir	get:\user-documents:bc	{'order': 'l3dMQUW	get:\user-documents:bc	0,574467957	{'sourceIds': '91500	get:\invoices:bc-documents-p	0,213315174	5	0,3896			
{'sourceIds': '9020103990,902	get:\invoices:bc-docur	{'documentKeys': '	get:\coaDocuments\list:	0,564777553	{'sourceIds': '90201	get:\invoices:bc-documents-p	0,432016522	2	1,5205			
{'order': 'ZvAA46xNVXIUEOH	get:\user-documents:bc	{'order': 'ZvAA46xN	get:\user-documents:bc	0,520060122	{'documentKeys': '	get:\coaDocuments\list:bc-dc	0,131629393	64	0,464640625			
{'type': 'Order', 'location': '/e	get:\user-documents\	{'order': 'ZvAA46xN	get:\user-documents:bc	0,999832511	{'documentKeys': '	get:\coaDocuments\list:bc-dc	0,000151915	4	0,48925			
{'companyCode': '1246', 'doc	get:\documents:bc-do	{'order': 'l3dMQUW	get:\user-documents:bc	0,999998331	{'type': 'Order', 'ord	get:\user-documents:bc-docu	1,61234E-06	1	2,311			
{'sourceIds': '9150026858,915	get:\invoices:bc-docur	{'location': '/en-M'	get:\invoices\{id}:bc-do	0,953698635	{'documentKey': '0f	get:\documents:bc-document	0,021781851	4	1,5125			
{'location': '/en-MY/connect	get:\invoices\{id}:bc-<	{'location': '/en-M'	get:\invoices\{id}:bc-do	0,433994383	{'type': 'Order', 'ord	get:\user-documents:bc-docu	0,251101583	7	2,341			
{'documentKey': '000000000(	get:\documents:bc-do	{'sourceIds': '9150(	get:\invoices:bc-docum	0,999998808	{'sourceIds': '91500	get:\invoices:bc-documents-p	1,16072E-06	1	2,255			
{'order': 'nl79TooylfrGJ1DxNC	get:\user-documents:bc	{}	get:\user-documents\{u	1	{'type': 'Cart', 'orde	get:\user-documents:bc-docu	8,90212E-11	1	0,499			
{'documentKey': '000000000(	get:\documents:bc-do	{'documentKey': '0	get:\documents:bc-docu	0,9998523	{'type': 'Cart', 'orde	get:\user-documents:bc-docu	5,62162E-05	1	1,315			
{'documentKey': '000000000(	get:\documents:bc-do	{'documentKey': '0	get:\documents:bc-docu	0,998447776	{'sourceIds': '90201(	get:\invoices:bc-documents-p	0,001325967	1	1,313			
{'order': 'i2lulou-XNmovipET	get:\user-documents:bc	{'documentKeys': '	get:\coaDocuments\list:	0,999239445	{'type': 'Cart', 'orde	get:\user-documents:bc-docu	0,000476121	1	0,597			
{'order': 'DdHr3JN8L8FP4otK	get:\user-documents:bc	{}	get:\user-documents\{u	1	{'type': 'Cart', 'orde	get:\user-documents:bc-docu	9,56827E-09	1	0,383			
{'order': '7BuiP8yL8MbQ5XU	get:\user-documents:bc	{'order': '7PEVRcim	get:\user-documents:bc	1	{'documentKey': '0f	get:\documents:bc-document	9,9781E-21	1	0,496			
{'order': '7PEVRcimu1yX', 'ini	get:\user-documents:bc	{'documentKeys': '	get:\coaDocuments\list:	0,999999642	{'documentKey': '0f	get:\documents:bc-document	4,16919E-07	1	0,402			
{'documentKey': '000000000(	get:\documents:bc-do	{'documentKeys': '	get:\coaDocuments\list:	0,999996901	{'sourceIds': '90201(	get:\invoices:bc-documents-p	3,1256E-06	3	2,324			

Figura B.5: Predictive Results - folha 5

TopServiceName	Query Params	Finish Dates	Cache Var	Cache Time	Number of Requests	Response Time	Stdev	TopServiceName	Mean	Median	Min	Max	Stdev
get:\user-documents\{i	{}	[Timestamp('2021-05-18 08:28:19.623000'	1	21228,72893	140	0,264785714	17407,63739	get:\user-documents\{ur	21228,73	21228,73	21228,73	21228,73	0
get:\user-documents:t	{'type': 'Cart', 'order	[Timestamp('2021-05-12 06:56:07.738000'	0	2,519075	9	0,468888889	1,774849806	get:\user-documents:bc	621,298	0,9511	0	20213,92	2550,116
get:\user-documents:t	{'order': '2O9uAKhB	[Timestamp('2021-05-12 03:43:23.978000'	0	0	1	0,416	null	get:\coaDocuments\{list:	8295,257	8563,287	7328,421	8994,062	864,5637
get:\user-documents:t	{'order': 'k-X4QPuyr	[Timestamp('2021-05-12 03:48:59.609000'	1	9,488775	10	0,5253	5,753480391	get:\documents:bc-docu	1911,881	1,372733	0	37227,04	5648,379
get:\user-documents:t	{'order': 'Ctl5GJjhS9	[Timestamp('2021-05-12 04:12:16.728000'	1	37,94248333	6	0,503	null	get:\invoices:bc-docume	1523,169	0,648533	0	42992,14	6231,531
get:\user-documents:t	{'order': 'l5mzTn6t7	[Timestamp('2021-05-12 00:37:58.075000'	0	0	1	0,999	null	get:\user-documents\{ic	16455,18	8471,99	5807,008	43069,73	17799,99
get:\user-documents:t	{'order': 'IwJyackUIi	[Timestamp('2021-05-12 05:40:35.697000'	1	8,839633333	7	0,399142857	null	get:\invoices\{id}:bc-do	0,062333	0	0	0,249333	0,124667
get:\user-documents:t	{'order': 'I3dMQUW	[Timestamp('2021-05-12 06:19:02.880000'	1	38,41638333	5	0,3896	null	get:\documents\{id}:bc-	5,466042	0	0	21,86417	10,93208
get:\user-documents:t	{'order': '2vAA46xN'	[Timestamp('2021-05-12 09:44:45.470000'	1	169,9094417	64	0,464640625	240,2882368	post:\user-documents:sa	1264,86	1264,86	1264,86	1264,86	0
get:\user-documents:t	{'order': 'nl79Tooyln	[Timestamp('2021-05-12 06:22:51.502000'	0	0	1	0,499	null						
get:\user-documents:t	{'order': 'i2lulou-XN	[Timestamp('2021-05-12 06:24:13.049000'	0	0	1	0,597	null						
get:\user-documents:t	{'order': 'DdHr3JN8L	[Timestamp('2021-05-12 06:24:41.463000'	0	0	1	0,383	null						
get:\user-documents:t	{'order': '7BuiP8yL8I	[Timestamp('2021-05-12 03:13:29.437000'	0	0	1	0,496	null						
get:\user-documents:t	{'order': '7PEVRcimL	[Timestamp('2021-05-12 03:13:36.856000'	0	0	1	0,402	null						
get:\user-documents:t	{'order': 'z7CDSuGuL	[Timestamp('2021-05-12 04:24:02.119000'	0	1,221533333	5	0,5288	null						
get:\user-documents:t	{'order': '0D05WAgA	[Timestamp('2021-05-12 04:22:20.813000'	0	0	1	0,439	null						
get:\user-documents:t	{'order': 'gl3UTZyKw	[Timestamp('2021-05-12 04:08:29.566000'	0	2,39475	5	0,4784	2,113259326						
get:\user-documents:t	{'type': 'Order', 'ord	[Timestamp('2021-05-12 07:08:14.885000'	0	0	1	0,453	null						
get:\user-documents:t	{'type': 'Cart', 'order	[Timestamp('2021-05-12 07:21:59.240000'	0	0	1	0,537	null						
get:\user-documents:t	{'type': 'Order', 'ord	[Timestamp('2021-05-12 07:01:56.152000'	0	2,53225	4	0,49275	null						
get:\user-documents:t	{'type': 'Cart', 'order	[Timestamp('2021-05-12 08:12:20.341000'	0	0	1	0,792	null						
get:\user-documents:t	{'type': 'Cart', 'order	[Timestamp('2021-05-13 00:04:51.947000'	1	8,541766667	5	0,473	null						
get:\user-documents:t	{'type': 'Cart', 'order	[Timestamp('2021-05-13 08:40:03.317000'	0	1,769283333	3	0,503	null						
get:\user-documents:t	{'type': 'Order', 'ord	[Timestamp('2021-05-13 08:44:12.062000'	0	3,2422	6	0,4485	null						

Figura B.6: Cache Results

Anexo C

A figura seguinte mostra o formato em que um determinado evento no projeto da DigiB era guardado no kibana e de seguida num ficheiro de texto.

```
"hits" : [
  {
    "_index" : "logstash-2021.05.28-mulesoft",
    "_type" : "_doc",
    "_id" : "8nYJs3kB3EA6J2-jXs1K",
    "_version" : 1,
    "_score" : 0.0,
    "_source" : {
      "statusMessage" : "SUCCESS",
      "@timestamp" : "2021-05-28T12:53:26.099Z",
      "serviceName" : "get:\\user-documents:bc-documents-papi-config",
      "externalKeyValue" : "",
      "engineMachineName" : "ip-172-21-18-197",
      "port" : 27168,
      "host" : "gke-eu-prod-primary-46536d9f-wb21.europe-west4-b.c.brenntag-eu-prod.internal",
      "topServiceName" : "get:\\user-documents:bc-documents-papi-config",
      "applicationName" : "bc-documents-papi",
      "tags" : [
        "mulesoft"
      ],
      "logTimeStamp" : "2021-05-28T12:53:26.076Z",
      "executionMilliseconds" : "0",
      "data" : {
        "payload" : {
          "message" : ""
        },
        "atributes" : [
          {
            "headers" : {
              "connection" : "close",
              "sec-ch-ua" : "\" Not;A Brand\";v=\"99\", \"Google Chrome\";v=\"91\", \"Chromium\";v=\"91\"",
              "accept-encoding" : "gzip, deflate, br",
              "sec-fetch-dest" : "empty",
              "sec-fetch-site" : "same-origin",
              "host" : "mule-worker-internal-bc-documents-papi.de-c1.cloudhub.io:8082",
              "authorization" : "0c2c1824a5cc12ba9d7ee41f233e022dbab13e49",
              "x-forwarded-port" : "443",
              "accept-language" : "en-US, en",
              "sec-ch-ua-mobile" : "?0",
              "cookie" : "*******",
              "x-request-id" : "9b5e82bfc439f79ff5fd13740abf0b03",
              "user-agent" : "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.77 Safari/537.36",
              "x-forwarded-for" : "fda7a13fb2ba1e7d6184a502710114ef8fb549e9",
              "x-forwarded-proto" : "https",
              "accept" : "application/json, text/plain, */*",
              "x-real-ip" : "*******",
              "sec-fetch-mode" : "cors",
              "x-forwarded-host" : "apac.brenntag.com",
              "x-scheme" : "https"
            }
          },
          {
            "uriParams" : { }
          },
          {
            "queryParams" : {
              "type" : "Cart",
              "order" : "42gXQl2o6Kjj",
              "include" : "user.permissions"
            }
          }
        ]
      },
      "logStep" : "SERVICE_START",
      "environment" : "prod",
      "requestId" : "9b5e82bfc439f79ff5fd13740abf0b03",
      "engineIpAddress" : "172.21.18.197",
```


Figura C.1: Evento

Referências

[1] Siraj ul Haq. Introduction to monolithic architecture and microservices architecture. May 2018.

[2] IBM. Microservices. URL: https://www.ibm.com/cloud/learn/microservices [último acesso em 2021-03-30].

[3] IBM. Soa (service-oriented architecture). URL: https://www.ibm.com/cloud/learn/soa [último acesso em 2021-04-7].

[4] MULESOFT. What is an esb? URL: https://www.mulesoft.com/resources/esb/what-esb.

[5] Shana Pearlman. What is api-led connectivity? July 2017.

[6] Amazon. What is an event-driven architecture? URL: https://aws.amazon.com/pt/event-driven-architecture/.

[7] Redhat. What is event-driven architecture? URL: https://www.redhat.com/en/topics/integration/what-is-event-driven-architecture.

[8] Mulesoft. How to design and manage apis.

[9] Mulesoft. What is api management? URL: https://www.mulesoft.com/resources/api/what-is-api-management.

[10] John Vester. Introduction to integration patterns. August 2018.

[11] Colton De Vos. What are application management services (ams)? May 2018.

[12] AWS. What is cloud computing? URL: https://aws.amazon.com/pt/what-is-cloud-computing/.

[13] IBM. Faas (function-as-a-service). URL: https://www.ibm.com/cloud/learn/faas.

[14] Google. Google cloud pricing calculator. URL: https://cloud.google.com/products/calculator.

[15] IBM. Machine learning. URL: https://www.ibm.com/cloud/learn/machine-learning [último acesso em 2020-07-15].

[16] M. Tim Jones. Supervised learning models. February 2018.

[17] M. Tim Jones. Supervised learning models. December 2017.

[18] Chaitanya Ekanadham. Using machine learning to improve streaming quality at netflix. March 2018.

[19] Mulesoft. Mule overview. URL: https://docs.mulesoft.com/mule-runtime/4.1/.

[20] Mulesoft. Anypoint platform. URL: https://www.mulesoft.com/pt/platform/enterprise-integration.

[21] Mulesoft. Anypoint studio. URL: https://www.mulesoft.com/pt/platform/studio.

[22] Christina Voskoglou. What is the best programming language for machine learning? MAY 2017.

[23] IBM. Neural networks. URL: https://www.ibm.com/cloud/learn/neural-networks.

[24] MySQL. What is mysql? URL: https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html.

[25] Postman. The collaboration platform for api development. URL: https://www.postman.com/.

[26] DigiB. Digib (brenntag digital sales solutions). URL: https://digib.homerun.com.

[27] Elastic. O que é o kibana? URL: https://www.elastic.co/pt/what-is/kibana.

[28] Mulesoft. Anypoint platform development: Fundamentals (mule 4). URL: https://training.mulesoft.com/course/development-fundamentals-mule4.