ALCÍRIO RENATO
FORTES DOS REIS

Implementação de uma Solução de Canais Remotos para um Banco Digital Português

Trabalho de Projeto submetido como requisito parcial para obtenção do grau de **Mestre em Engenharia de Software**.

JÚRI

Presidente: Professor Cláudio dos Santos Sapateiro, ESTSetúbal/IPS

Arguente: Professor Nuno de Pina Gonçalves, ESTSetúbal/IPS

Orientador: Professor José Moinhos Cordeiro, ESTSetúbal/IPS

Novembro, 2023

Para a minha filhota.

Nunca é tarde para alcançares os teus objetivos.

Agradecimentos

Após a realização do presente projeto de dissertação de mestrado, existem alguns agradecimentos que não podem deixar de ser feitos, a todas as pessoas envolventes que de alguma maneira estiveram presentes nesta fase.

Em primeiro lugar quero agradecer à organização Innovation Makers pela oportunidade de realizar este projeto. Um especial obrigado ao Eng.º Ricardo Portela, orientador organizacional deste projeto, por toda a confiança/responsabilidade depositada assim como por toda a sua disponibilidade e orientação que me foi dando ao longo deste projeto e no presente relatório.

Não posso deixar de agradecer em geral ao corpo docente da Escola Superior de Tecnologia de Setúbal, pela forma como lecionaram este mestrado e os valores que transmitiram ao longo destes dois anos. Deixo uma palavra de agradecimento especial ao Professor José Cordeiro, docente orientador deste projeto, toda a sua disponibilidade e orientação foram as peças essências para a conclusão do mesmo.

Por fim à minha família, um grande obrigado pois sem o seu apoio nada disto seria possível seria.

Resumo

O rápido desenvolvimento das tecnologias de informação e comunicação causou um impacto significativo na sociedade e na economia global, originando efeitos consideráveis nos serviços financeiros e bancários. O *Internet Banking* foi concebido para simplificar as transações financeiras e não financeiras dos clientes bancários através da Internet, eliminando a necessidade de deslocamento aos convencionais balcões bancários.

A Innovation Makers (INM), empresa tecnológica considerada como uma forte referência no desenvolvimento de soluções financeiras, atende a um novo cliente, um banco português, que enfrenta um desafio particular. Este banco, que possui apenas dois balcões físicos e carece de uma solução de *Internet Banking* para atender os seus clientes que preferem utilizar dispositivos móveis. Nesse sentido os clientes do banco enfrentam algumas limitações na utilização dos serviços financeiros online, uma vez que só o podem fazer através de um navegador *web* ou fisicamente numa das suas agências bancárias.

A INM atendendo a esta necessidade propõe uma das suas soluções, que consiste numa solução de *software* de *middleware/backend* Omni canal modular. Este *software* facilita a integração com o sistema bancário, proporcionando uma variedade de operações bancárias online através de diferentes canais. Nesse sentido, o objetivo da INM foi implementar e adaptar a solução Omni canal modular de acordo com as necessidades específicas do banco.

O trabalho realizado neste documento foca-se na apresentação da solução Omni canal da INM, os seus respetivos fundamentos teóricos/tecnológicos e na documentação dos desenvolvimentos efetuados de forma a garantir as adaptações necessárias à solução Omni canal a este novo cliente (banco português). Por fim serão ainda apresentadas as perspetivas de trabalho futuro de forma a dar continuidade a este projeto.

A parceria entre o banco e a INM possibilitou a integração de uma solução de *Internet Banking* mais versátil e atraente, com a capacidade de atingir um público mais vasto alcançando assim os objetivos iniciais do cliente, banco português.

Palavras-chave: Soluções financeiras, S*oftware, Middleware,* Integração, *Internet Banking,* Core Bancário.

Abstract

The rapid development of information and communication technologies has caused a significant impact on society and the global economy, leading to considerable effects on financial and banking services. Internet Banking was designed to simplify both financial and non-financial transactions for bank customers through the Internet, eliminating the need to physically visit traditional bank branches.

Innovation Makers (INM), a technological company considered a strong reference in the development of financial solutions, is serving a new client, a Portuguese bank facing a specific challenge. This bank, which has only two physical branches and lacks an Internet Banking solution to cater customers who prefer using mobile devices. In this regard, the bank's customers face some limitations using online financial services, as they can only do so through a web browser or physically at one of its bank branches.

Addressing this need, INM proposes one of its solutions, which consists of a modular Omni-channel middleware/backend software solution. This software facilitates the integration with the banking system, enabling a variety of online banking operations through different channels. In this regard, INM's goal was to implement and adapt the modular Omni-channel solution according to the specific needs of the bank.

The work carried out in this document focuses on presenting INM's Omni-channel solution, its respective theoretical/technological foundations, and documenting the developments made to ensure the necessary changes to the Omni-channel solution for this new client (Portuguese bank). Finally, future work perspectives will be presented to continue this project.

The partnership between the bank and INM enabled the integration of a more versatile and appealing Internet Banking solution, with the ability to reach a broader audience, thus achieving the initial goals of the Portuguese bank client.

Keywords: Financial Solutions, Software, Middleware, Integration, Internet Banking, Core Banking

Índice

Lista de Figuras

Lista de Siglas e Acrónimos

AD	Active Directory
API	Application Programming Interface
B2B	Business-to-Business
CIAM	Customer Identity and Access Management
DI	Dependency Injection
GDPR	General Data Protection Regulation
GoF	Gang of Four Design Patterns
IAM	Identity and Access Management
IDE	Integrated Development Environment
INM	Innovation Makers
IoC	Inversion of Control
JIT	Just-In-Time
LDAP	Lightweight Directory Access Protocol
MFA	Multi-Factor Authentication
PAM	Privileged Access Management
POM	Project Object Model
QA	Quality Assurance
RBAC	Role-Based Access Control
REST	Representational State Transfer
SEPA	Single Euro Payment Area
SMS	Short Message Service
SOA	Service-Oriented Architecture
SQL	Structured Query Language
SSMS	SQL Server Management Studio
SSO	Single Sign-On
UI	User Interface
USSD	Unstructured Supplementary Service Data
UX	User Experience
XACML	Extensible Access Control Markup Language
XML	Extensible Markup Language

1 Introdução

Este relatório tem como objetivo detalhar o trabalho de projeto, no âmbito da unidade curricular de Projeto ou Estágio do Mestrado em Engenharia de Software, na Escola Superior de Tecnologia de Setúbal, do Instituto Politécnico de Setúbal.

A unidade curricular visa proporcionar a oportunidade de colocar em prática alguns dos conhecimentos adquiridos ao longo do mestrado, assim como pensar/desenvolver um produto capaz de responder às necessidades de um contexto real de trabalho.

O projeto teve como empresa acolhedora a Innovation Makers, sobre orientação do Partner e CTO Eng. Ricardo Portela assim como do Prof. Dr. José Cordeiro como docente orientador.

1.1 Contexto

O rápido desenvolvimento das tecnologias de informação e comunicação tiveram um grande impacto na sociedade e na economia mundial. Esta transição produziu grandes mudanças nos serviços financeiros e bancários (ISAC & DRIGĂ, 2015). A combinação de inovação tecnológica e competição entre instituições bancárias têm permitido uma maior oferta de produtos e serviços bancários disponibilizados por meio de canais de comunicação eletrónicos e interativos (ISAC & DRIGĂ, 2015). Os serviços de *Internet Banking* visam facilitar aos clientes bancários a realização de transações financeiras e não financeiras pela Internet sem que estes tenham de se deslocar aos tradicionais balcões bancários. Estas plataformas disponibilizam online a maioria dos serviços bancários tradicionalmente fornecidos através do balcão local tornando-se desta forma cada vez mais importantes para os utilizadores da internet. Os serviços bancários online são uma excelente oportunidade de venda de produtos bancários e permitem que os clientes consigam executar operações financeiras em qualquer hora e em qualquer lugar do mundo quando conectados à internet (ISAC & DRIGĂ, 2015).

1.2 Problema

A Innovation Makers (INM), empresa tecnológica considerada como uma forte referência no desenvolvimento de soluções financeiras, possui uma solução Omni canal modular,

que suporta um conjunto de bancos em Angola, Moçambique e Guiné-Bissau denominada de Digital Bank Integration. Esta solução consiste num *software* de *middleware/backend* que permite a interligação dos canais digitais com o sistema bancário, oferecendo assim um conjunto de operações bancárias *online*. No projeto no qual se foca este documento, a INM atende a um cliente, um banco português, que enfrenta um desafio particular. Este banco não dispõe de uma solução de *Internet Banking* para os seus clientes que utilizam dispositivos móveis, além de ter apenas dois balcões físicos à disposição. Como resultado, os clientes enfrentam algumas limitações em relação à realização das suas transações, uma vez que só podem fazê-lo através de um navegador *web* ou fisicamente numa das suas agências bancárias. Nesse contexto, será necessário integrar uma nova solução de *Internet Banking* em conjunto com uma plataforma de *BackOffice* para a gestão dos clientes. Isso tem como objetivo facilitar e auxiliar os clientes nas suas transações proporcionando uma oferta mais ampla de produtos e serviços. Este projeto de dissertação de mestrado irá basear-se na utilização da solução modular Omni-Canal da INM, a qual será instanciada e implementada para atender às necessidades deste novo cliente. Esta abordagem permite a adaptação da solução existente, proporcionando uma implementação personalizada que atende aos requisitos específicos do referido cliente.

1.3 Objetivos

O trabalho a ser desenvolvido neste documento foca-se por acompanhar e documentar a adaptação necessária da solução existente a este novo cliente (banco português). Nesse sentido, os objetivos passam pelo levantamento e análise de requisitos de integração necessários, identificação e mapeamento dos conceitos já suportados com os conceitos que são exigidos no contexto de espaço bancário europeu, interligação da solução existente com um *provider* externo de *Identity and Access Management*, interligação com um novo core bancário através de uma camada de serviços REST e a adaptação das interfaces com os canais remotos de forma a que exista um menor número possível de alterações às interfaces *Web* e *Mobile*. Por fim será feita a implementação do produto em ambiente produtivo. Todas estas fases de integração incluem a gestão da equipa de desenvolvimento do produto assim como as decisões importantes na fase de arquitetura do mesmo.

1.4 Estrutura do documento

Este documento é organizado em diversos capítulos, cada um com uma finalidade específica. No capítulo 1, são apresentados o contexto, o problema e os objetivos subjacentes a este trabalho. No capítulo 2, é feita a apresentação da organização Innovation Makers, abordando as equipas que a compõem e as soluções que desenvolve. O capítulo 3 concentra-se na solução de *middleware* Digital Bank Integration, que é o foco central deste trabalho de mestrado. Aqui, foi explorada a arquitetura subjacente, as tecnologias utilizadas e as vantagens inerentes a essa solução. No capítulo 4, é feita uma análise dos fundamentos teóricos e tecnológicos, onde são destacados estudos e pesquisas relacionados com o tema central deste trabalho de mestrado. No capítulo 5, são apresentados os detalhes sobre a organização e gestão do projeto, fornecendo uma visão geral que destaca as equipas participantes e as diferentes fases do desenvolvimento. No capítulo 6 são abordadas todas as fases referentes à execução do projeto, descrevendo em detalhe cada uma delas e os contributos do aluno. Por último, o Capítulo 7 apresenta as conclusões finais e perspetivas de trabalho futuro deste projeto de dissertação de mestrado, resumindo os resultados alcançados.

2 Caracterização da Organização

Este capítulo tem como objetivo fazer uma breve apresentação da Innovation Makers (INM), organização acolhedora deste trabalho de projeto.

2.1 A Empresa

A Innovation Makers (INM), está presente em Angola, Moçambique e Portugal, sendo que a sede atualmente é em Portugal situada no Parque Tecnológico de Óbidos. Conta com mais de 13 anos de atividade e com mais de 70 projetos tecnológicos, sendo considerada uma forte referência no desenvolvimento de soluções em mais de 7 países. Tem como missão transformar o sector financeiro através de uma constante relação com o cliente e implementando soluções que minimizam os custos e incrementam o negócio digital das organizações. A INM conta com uma equipa multidisciplinar, nas áreas de engenharia, gestão, ciências socias e *design* que permitem criar soluções integradas e tecnológicas com grandes níveis de complexidade e qualidade funcional, capazes de responder às necessidades de cada cliente e ao contexto em que se inserem (Innovation Makers, 2023).

2.2 Equipas

A Innovation Makers é composta por vários profissionais altamente qualificados distribuídos pelas equipas de Consultoria, Gestão de Projeto, UI e UX, *Middleware*, Web *Development*, Android *Development*, iOS *Development*, *Quality Assurance* e Suporte e Assistência Técnica (Innovation Makers, 2023).

A equipa de Consultoria é constituída pelos *partners* e colaboradores seniores da INM que detêm uma vasta experiência como consultores em tecnologias de inovação tendo colaborado em inúmeros projetos de referência internacional, estando sempre disponíveis para prestar apoio aos clientes ajudando-os a encontrar as soluções tecnológicas mais adequadas a cada desafio.

A equipa de Gestão de Projeto é constituída por colaboradores especificamente formados em técnicas de gestão de projetos que acompanham e coordenam toda a atividade desenvolvida em articulação com os clientes, desde o primeiro momento de

levantamento de requisitos até à fase final de produção, testes e suporte às soluções já implementadas.

As equipas de UI/UX são constituídas por colaboradores altamente competentes que asseguram que todos os projetos tenham as melhores práticas ao nível da experiência do utilizador integrando os mais elevados princípios de UX (*User Experience*), de forma a criar interações intuitivas e gratificantes. Da mesma forma, garantem que os projetos acompanhem as mais recentes tendências ao nível de UI (*User Interface*) que possibilitam a criação de interfaces atrativas e inovadoras que funcionam como fator diferenciador para os clientes.

A equipa de *Middleware* é fortemente especializada no desenvolvimento e implementação de soluções de *middleware* para os principais sistemas core usados por Bancos e operadores de telecomunicações em todo o mundo. A equipa assegura a integração plena e funcional de sistemas pré-existentes com aplicações de negócio evoluídas e inovadoras.

A equipa de *Web Development* é composta por colaboradores dotados de grande experiência e capacidade de desenvolvimento de soluções para ambientes *web*. Nesse sentido, e dominando as linguagens de programação de cliente e de servidor, conseguem desenvolver soluções adequadas aos objetivos de negócio de cada cliente.

A equipa de *Android Development* é especializada em Android, e cria aplicações nativas para estes sistemas operativos móveis de forma a aproveitar todo o potencial dos equipamentos a que se destinam.

A equipa de *iOS Development* é especializada em desenvolvimento e programação em iOS, que possibilita a criação de aplicações que garantem os melhores níveis de produtividade e usabilidade para as plataformas da Apple.

A equipa de *Quality Assurance* (QA) é responsável por assegurar a qualidade e confiabilidade de todas as aplicações desenvolvidas. Esta efetua testes rigorosos, identifica e relata problemas e trabalha em conjunto com as restantes equipas de desenvolvimento de forma a garantir que os produtos atendem aos mais elevados padrões de qualidade antes de serem disponibilizados aos respetivos clientes.

A equipa de Suporte e Assistência Técnica assegura a todos os clientes um serviço de suporte e de assistência técnica permanente, disponível 24 horas por dia, por ligação remota ou presencial.

2.3 Soluções

A Innovation Makers oferece soluções tecnológicas para as áreas de Banking, eWallet, Gestão Financeira e Pagamentos.

A solução para *Banking* da INM oferece uma gama completa de produtos, que proporcionam uma experiência contínua e consistente aos clientes, independentemente do canal que escolham para interagir com a empresa (Omni canal). O objetivo destes produtos é ir ao encontro das expectativas das organizações e clientes atuais, fornecendo em tempo real e para todos os canais o máximo de serviços financeiros digitais. Esta solução modular e de fácil customização/adaptação permite ainda uma maior gestão dos canais, colocando o cliente no centro da relação bancária. Possibilita ainda, através da interpretação em tempo real dos dados, o fornecimento de *insights* determinantes para estratégias de *marketing*, vendas e serviços personalizados, de forma a possibilitar a escalabilidade do negócio e a fortalecer a experiência do utilizador. Esta solução disponibiliza também uma ferramenta de *BackOffice* que permite gerir e controlar todos os canais, colaboradores e atividades financeiras dos clientes. Esta solução de *Banking* tem os seguintes produtos associados:

- Digital Bank Integration – *Middleware backend* que permite a integração dos canais com o sistema bancário, de forma rápida, dinâmica e segura, como plataforma de controlo e gestão de todas as atividades internas e dos clientes.
- *Internet Banking* – Ferramenta online, que disponibiliza todas as operações bancárias online sem a necessidade do cliente se deslocar a uma sucursal.
- *Mobile Banking* – É o produto fundamental na estratégia digital dos bancos em todo o mundo e permite realizar operações bancárias, assim como receber comunicação e serviços personalizados diretamente no dispositivo movel do utilizador.
- SMS & USSD – Produto que permite a realização de operações bancárias em todo o tipo de dispositivos, incluindo os mais básicos, através dos protocolos de SMS/USSD.
- Correspondentes Bancários – Produto que permite às entidades ampliarem a rede de serviços e produtos, para que estes cheguem a um maior número de pessoas sem ter o investimento e peso logístico normalmente exigido.
- Portal Balcão – Ferramenta de gestão/atendimento das atividades de uma sucursal.

- Balcão Movel – Produto que permite às entidades, através de um gerente, oferecer serviços bancários direcionados e personalizados a um segmento de clientes premium sem a necessidade do mesmo se deslocar a uma sucursal.
- *Kiosk Hardware* – Produto que oferece a possibilidade de levar até aos clientes das entidades financeiras ou de outros setores, todos os serviços equivalentes a um equipamento ATM.
- *Kiosk Software* – Ferramenta que permite uma integração com o sistema *core* das entidades e com o *switch* financeiro da rede interbancária, e ainda o desenvolvimento ou adaptação com as diferentes necessidades do serviço e marca.

A solução eWallet consiste numa carteira digital para múltiplas funcionalidades, como, por exemplo, transferências e pagamentos eletrónicos, envio e receção de fundos, compras. Através de uma conta virtual segura, os utilizadores podem enviar e receber dinheiro diretamente via web ou mobile, sem a necessidade de transportar cartões ou dinheiro em numerário.

As soluções de gestão financeira permitem que as entidades ofereçam aos seus clientes um conjunto de ferramentas inteligentes de gestão e análise financeira de forma a garantir uma melhor gestão dos seus clientes. As soluções para gestão financeira contam com os seguintes produtos:

- *Business Financial Management (BFM)* - Ferramenta que permite às PMEs obter *insights* relevantes de forma a entender e ultrapassar os seus obstáculos. Nesse sentido, é possível entender melhor a relação com os clientes, definir e identificar as melhores estratégias de *marketing* assim como as oportunidades de crescimento. O objetivo é dar controlo às pequenas e médias empresas sobre o seu negócio, de forma a conseguirem gerir melhor o seu dia a dia e as suas atividades.
- *Personal Management Service (PMS)* – Ferramenta que permite às entidades financeiras, fornecer a um segmento particular, o controlo, previsão e bem-estar financeiro. Através de inteligência artificial, é possível gerir o dinheiro de forma inteligente, identificando os melhores hábitos de consumo, é ainda possível agregar contas, categorizar transações e planear gastos.

As soluções de pagamentos têm em conta várias necessidades do negócio e oferecem vários produtos integráveis com os sistemas internos e externos das entidades em

múltiplos setores permitindo assim pagamentos seguros de forma ágil e conveniente. As soluções para pagamentos têm os seguintes produtos:

- Pagamentos Móveis – Produto que permite fazer pagamentos móveis seguros, rápidos e práticos através do número de telemóvel sem precisar de cartão de crédito.
- *Kiosk Payments* – Trata-se de um conjunto de soluções hardware que permitem a implementação de sistemas de pagamentos em cartão ou numerário com as redes interbancárias e ainda a expansão da rede logística.
- *Voucher Server* – Produto que oferece aos operadores e prestadores de serviços uma ferramenta multicanal que garante os pagamentos, recargas e ativação de serviços. A emissão dinâmica de vouchers é protegida por vários métodos e requisitos de segurança, reduzindo consideravelmente o risco de fraude.
- Sistema de pagamentos e serviços (SPS) – Ferramenta que permite às entidades financeiras ou de serviços públicos e privados, centralizarem os fluxos de pagamentos, podendo estes serem ou não integrados com sociedades interbancárias/serviços domésticos externos. Esta fácil integração e adaptação permite gerar mais oportunidades de negócio.
- Subsistema de débitos diretos (SDD) – Ferramenta que permite às entidades oferecer aos seus clientes uma alternativa de pagamentos de serviços através de débitos diretos em conta de forma a minimizar o tempo de deslocação ou os custos adicionais por atraso no pagamento.

Estas soluções são desenvolvidas com recurso às mais recentes tecnologias, permitindo responder de forma eficaz aos requisitos necessários e acompanhar de perto a constante evolução tecnológica. Esta abordagem assegura que as soluções estão sempre em sintonia com as últimas inovações, proporcionando aos clientes uma resposta eficiente e a capacidade contínua de adaptação às exigências de um ambiente tecnológico em constante mudança.

3 Digital Bank Integration

Este capítulo tem com objetivo apresentar a solução usada no âmbito deste trabalho de projeto. Nesse sentido será feita uma breve apresentação da mesma onde será explicada a arquitetura, tecnologias utilizadas e as vantagens na sua escolha.

3.1 Descrição

A solução existente e alvo deste trabalho de projeto, consiste, em particular, no produto Digital Bank Integration, um *middleware*/*backend* Omni canal, presente em alguns bancos do continente africano e que permite a integração dos canais (*Web, Mobile e SMS/USSD*) com o sistema bancário de forma ágil, rápida e segura, totalmente independente do core financeiro. Esta solução é o centro da inteligência de produtos como o *Internet Banking*, *Mobile Banking*, SMS *Banking* & USSD *Banking* que foram descritos na secção 2.3. Este produto tira ainda proveito da tecnologia *Machine Learning* de forma a ser capaz de produzir *insights* relevantes para o entendimento comportamental e financeiro dos utilizadores (Innovation Makers, 2023). Algumas das funcionalidades presentes neste produto, a nível de *Internet Banking* para um utilizador são:

- Consulta do património, onde é possível ter um resumo daquilo que são os ativos e passivos de um determinado utilizador;
- Consulta de contas e respetivos saldos e movimentos;
- Efetuar transferências internacionais e nacionais ou carregamento de serviços e cartões;
- Efetuar pagamentos de serviços, estado, telecomunicações ou outros serviços públicos;
- Solicitar dinheiro ou efetuar levantamentos sem cartão;
- Efetuar investimentos na bolsa de valores, etc.

Além destas operações bancárias, esta solução conta com as seguintes funcionalidades a nível de *BackOffice* para os gestores bancários:

- Gestão de clientes, contas, canais ativos, relações entre entidades, utilizadores, histórico de alterações;
- Gestão de canais e respetivas operações;

- Gestão de limites de transações para clientes;
- Monitorização de operações bancárias;
- Auditorias dos utilizadores;
- Comunicações, etc.

Além destas operações, o *middleware* está preparado para comunicar com outras entidades externas ao core bancário (como por exemplo, operadoras de telecomunicações para envio SMS) ou a integração com outras bibliotecas/API's (como por exemplo, Google para *push notification*).

3.2 Arquitetura

A arquitetura da solução do *middleware* Digital Bank Integration é dividida em camadas, onde cada camada desempenha as suas próprias funções. Cada camada é constituída por um ou mais módulos com funcionalidades específicas e que poderão ter dependências diferentes, consoante a necessidade do cliente. A arquitetura deste *middleware* poderá ser representada da seguinte forma:

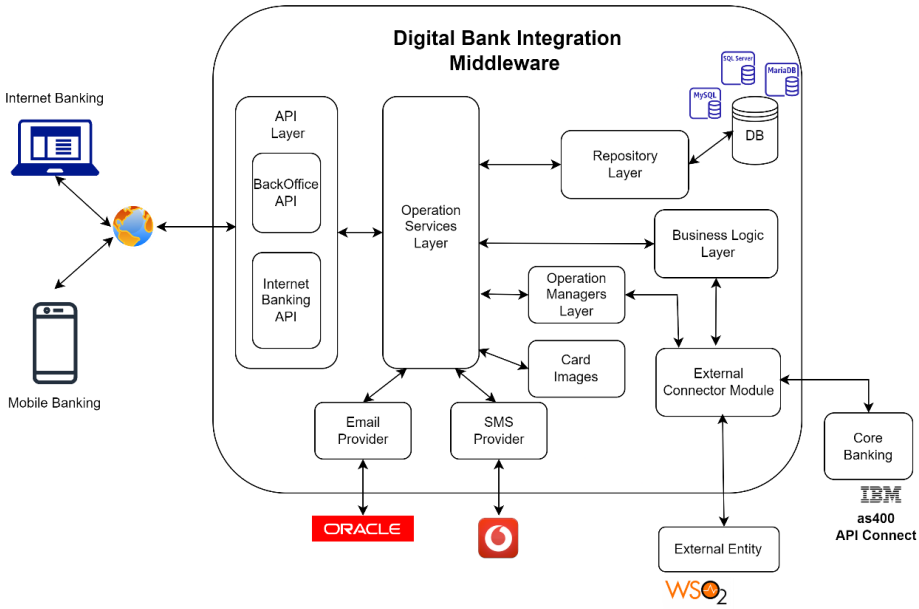

Figura 1 - Arquitetura do Digital Bank Integration Middleware

Conforme podemos verificar na figura acima, o Digital Bank Integration Middleware é composto pelas seguintes camadas:

- **API Layer** – É a camada onde estão as API's responsáveis pelas

comunicações entre o *middleware* e os canais. Cada API é um módulo e é composto por um conjunto de serviços REST onde é feita a validação dos *inputs*, o encaminhamento dos pedidos para as devidas operações e o tratamento da resposta.

- **Operation Services Layer** – É a camada onde constam as operações. Este é o primeiro nível de abstração e cada operação a este nível é composta por um conjunto de funcionalidades comuns a todos os bancos clientes. Cada operação requer também a execução de lógica específica para cada cliente, nesse sentido, essa lógica é injetada por meio de interface e implementada pelo módulo específico do cliente (por exemplo, *providers* específicos de cada cliente, módulos com imagens ou conteúdo específico, módulos com lógica de negócio de cada core bancário).
- **Business Logic Layer** – É a camada onde constam os módulos com as implementações de funcionalidades especificas para cada core bancário. A informação é solicitada ao módulo conector responsável por essa comunicação.
- **Operation Managers Layer** – Camada onde constam os módulos com as implementações de funcionalidades específicas que dependem de outra entidade externa que não o core bancário. A Informação é solicitada ao módulo conector responsável por essa comunicação.
- **Repository Layer** – É a camada onde constam os módulos que são responsáveis pelo modelo de dados, modelo este que é baseado em *code first* (abordagem de desenvolvimento em que o design e a estrutura da base de dados são derivados do código-fonte da aplicação).
- **External Connector Module** – Módulos responsáveis pela comunicação com as entidades externas. Cada módulo conector é específico para um determinado cliente e destinado a conectar com uma determinada entidade.

Cada camada tem as suas próprias responsabilidades de forma a seguir os princípios de arquitetura de software.

3.3 Tecnologias utilizadas

As principais tecnologias que suportam esta solução de *middleware* estão compreendidas entre:

- Linguagem de programação:
  - **Java** versão 8 ou superior;
- Frameworks:
  - **Maven** – *Framework* de gestão de projeto/dependências (Maven: The Definitive Guide, 2009);
  - **Jakarta Enterprise Edition -** Coleção de especificações abstratas e padronizadas que prescrevem soluções para desafios comuns enfrentados no desenvolvimento de software utilizando a linguagem Java (Saeed, 2020);
  - **Swagger** – Ferramentas que facilitam gerar e manter a documentação das API's, garantem que a documentação permaneça atualizada à medida que a API evolui (Swagger.io, 2023);
  - **Google Guice** – *Framework* de injeção de dependência. Esta facilita o uso de injeção de dependência em projetos Java (Google, 2021).
  - **Hibernate** – *Framework* de mapeamento objeto/relacional leve para Java (Elliot et al., 2008).
  - **OpenFeign** – Biblioteca declarativa usada para facilitar a criação de clientes REST em aplicações Java (OpenFeign, 2023);
- Ferramentas:
  - **IntelliJ IDEA** – Desenvolvido pela JetBrains é um ambiente de desenvolvimento integrado (IDE) escrito em Java (Jetbrains, 2023);
  - **Jenkins** – Servidor de automação de código aberto e autocontido que pode ser usado para automatizar todos os tipos de tarefas relacionadas à construção, teste, entrega e implantação de *software* (Jenkins, 2023);
  - **MySQL Workbench** – É uma ferramenta visual unificada para arquitetos, programadores e administradores de bases de dados. O MySQL Workbench oferece modelagem de dados, desenvolvimento SQL e ferramentas de administração para configuração de servidores, administração de utilizadores, backup entre outros (MySQL, 2023);

    - **SQL Server Management Studio (SSMS)** - É um ambiente integrado para gerenciar qualquer infraestrutura SQL, desde o SQL Server até bases de dados SQL do Azure. O SSMS fornece ferramentas para configurar, monitorar e administrar instâncias do SQL Server e bases de dados (SSMS, 2023);
    - **JFrog Artifactory** – É a solução única para armazenar e gerenciar todos os artefactos, binários, pacotes, arquivos, containers e componentes para uso em toda a cadeia de fornecimento de software (JFrog, 2023).
    - **GitLab** – Repositório de *software* baseado em git, com suporte a Wiki, gestão de tarefas e CI/CD (GitLab, 2023);
    - **GitKraken** – Interface gráfica que fornece uma ferramenta de gestão de repositórios git (GitKraken, 2023);
- Motores de Base de dados:
    - SQL Server – É um servidor de base de dados relacional desenvolvido pela Microsoft (Microsoft, 2023a).
    - MariaDB – É uma das bases de dados relacionais *open source* mais populares do mundo (MariaDB Fundattion, 2023).
    - MySQL – O software MySQL fornece um servidor de base de dados SQL (Structured Query Language) muito rápido, multithreaded, multiutilizador e robusto. O servidor MySQL destina-se a sistemas de produção críticos, com carga pesada, bem como à incorporação em software de implementação em massa (MySQL, 2023).

Estas são as tecnologias base utilizadas, sendo este um projeto Maven, existe ainda a possibilidade de adicionar mais dependências relativas a bibliotecas que ajudem no desenvolvimento da solução.

3.4 Vantagens

Algumas das vantagens possíveis para a solução descrita, poderão ser categorizadas por:

- Integração
    - Solução modular de canais, uma vez que cada canal poderá ser ativado ou desativado através da plataforma de *BackOffice;*

    - Integração rápida com o core do Banco sem complexos requisitos de IT, tendo em conta o nível de abstração entre as operações financeiras e o core bancário.
- Visão geral e controlo
    - Configuração, controlo, monitorização e gestão de atividades, uma vez que qualquer tipo de atividade carece de permissões para execução, e são registadas e configuradas num modelo de dados adequado de forma a permitir uma maior gestão e segurança;
    - Diferentes níveis de acesso para diferentes tipos de utilizadores.
- Diminuição de tempo de lançamento no mercado
    - Solução personalizada ou modular, uma vez que cada modulo é separado e é apenas configurado consoante as necessidades do cliente;
    - Rápida implementação e minimização do impacto económico, uma vez que não é necessário desenvolver um novo produto de raiz.
- Proximidade com cliente
    - Rápidas configurações para campanhas e comunicação permanente;
    - Foco na experiência de utilizador.
- Segurança e Suporte
    - Mecanismos que permitem o controlo e deteção de ameaças;
    - Equipa dedicada à manutenção e suporte.
- *Machine Learning*
    - Modelos analíticos que permitem analisar métricas definidas à medida e consoante as necessidades dos clientes.
    - Produção de *insights* relacionados com os comportamentos financeiros dos utilizadores que poderão ajudar na tomada de decisões ou em estratégias de *marketing.*

Estas vantagens resumem uma solução completa que permite fornecer operações financeiras aos canais que Web, Mobile e SMS/USSD.

4 Fundamentos Teóricos e Tecnológicos

Este capítulo tem como objetivo apresentar os fundamentos teóricos e tecnológicos deste projeto de dissertação de mestrado. Inicia-se com a secção 4.1, que explora conceitos fundamentais relacionados à arquitetura de software, incluindo a definição, princípios-chave, atributos de qualidade e os estilos de arquitetura. A secção 4.2 aborda o conceito de *middleware* e destaca alguns dos tipos presentes na solução deste projeto. Em seguida, a secção 4.3 explora os padrões de *design* mais utilizados neste projeto. Na secção 4.4, é apresentada a ferramenta de gestão de projetos/dependências Maven e os seus principais componentes. Por fim, na secção 4.5, é introduzido o conceito de gestão de identidade e acessos (IAM).

4.1 Arquitetura de software

Esta secção tem como objetivo apresentar o conceito de arquitetura de *software*, os seus princípios-chave e atributos de qualidade. Além disso, serão apresentados alguns estilos de arquitetura existentes e será demonstrada a contribuição deste conceito no contexto deste projeto de dissertação de mestrado.

4.1.1 Definição

Arquitetura de *software* de um sistema é o conjunto de estruturas necessárias para raciocinar sobre o sistema, que compreende elementos de *software*, relações entre eles e propriedades de ambos. Uma estrutura é simplesmente um conjunto de elementos mantidos juntos por uma relação. Os sistemas de *software* são compostos por muitas estruturas, e nenhuma estrutura isolada pode ser considerada a arquitetura, nesse sentido uma arquitetura é essencialmente uma abstração de um sistema que seleciona determinados detalhes e suprime outros (Bass et al., 2013). No coração de todos os sistemas de *software* bem projetados está uma boa arquitetura de *software* e um bom conjunto de decisões de *design*, nesse sentido a arquitetura de um sistema é o conjunto de princípios e decisões de *design* feitas durante o seu desenvolvimento e qualquer subsequente à sua evolução (Kramer et al., 2010). O objetivo da arquitetura é identificar

os requisitos que afetam a estrutura da aplicação, nesse sentido é importante considerar o efeito geral das decisões do projeto, as compensações inerentes entre os atributos de qualidade (como desempenho e segurança) e os compromissos necessários para atender aos requisitos do utilizador, do sistema e do negócio (Microsoft, 2009).

4.1.2 Princípios-chave

Os princípios de arquitetura de *software* visam tornar o *design* de *software* mais compreensível, escalável e de fácil manutenção. Estes princípios-chave ajudam a criar uma arquitetura que adere a princípios comprovados, minimiza custos e requisitos de manutenção, e promove a usabilidade e a extensibilidade (Microsoft, 2009). Estes são:

- ***Separation of concerns*** **–** Onde o principal objetivo é separar diferentes preocupações dentro do projeto ou código. A aplicação deve ser dividida em recursos distintos com o mínimo possível de sobreposição de funcionalidades. O fator importante é a minimização de pontos de interação para alcançar alta coesão e baixo acoplamento.

- ***Single Responsibility principle*** **–** Cada componente ou módulo deve ser responsável por apenas um recurso ou uma funcionalidade específica ou uma agregação de funcionalidades coesa.

- ***Principle of Least Knowledge*** **–** Também conhecido como a Lei de Demeter ou LoD, refere que o componente ou objeto não deve saber sobre os detalhes internos de outros componentes ou objetos.

- ***Don't repeat yourself*** **(DRY) –** Cujo objetivo é reduzir a repetição de padrões e duplicação de código. A intenção só precisa ser específica em um único lugar.

- ***Minimize upfront design*** **–** Refere que apenas o necessário deve ser desenhado.

Ao aplicar estes princípios na fase de arquitetura do *software*, temos uma melhor visão de como é a interligação dos componentes do sistema, por sua vez isso ajudará a implementar mudanças caso sejam necessárias ou a evitar problemas posteriores.

4.1.3 Atributos de qualidade

Os atributos de qualidade fazem parte das principais decisões que devemos tomar de forma a garantir que são considerados todos os fatores importantes ao começar e, de seguida, desenvolver iterativamente um projeto de arquitetura (Microsoft, 2009). Os atributos de qualidade são propriedades mensuráveis ou testáveis de um sistema que são usadas para indicar o quão bem o sistema satisfaz as necessidades dos seus *stakeholders* (Cervantes & Kazman, 2016), estes são os fatores gerais que afetam o comportamento em tempo de execução, *design* do sistema, e experiência do utilizador. Nesse sentido, os atributos de qualidade podem ser usados para focar o pensamento sobre os problemas críticos que o projeto deve resolver. Devem ser consideradas as seguintes questões na escolha dos atributos de qualidade que melhor se adequam às necessidades de *software*:

- Quais são os principais atributos de qualidade necessários para a sua aplicação? Estes devem ser identificados como parte do processo de projeto.
- Quais são os principais requisitos para adotar esses atributos? São quantificáveis?
- Quais são os critérios de aceitação de forma a indicar que os requisitos foram atendidos?

A escolha dos atributos de qualidade, depende dos requisitos e necessidades do sistema, nesse sentido seguem alguns exemplos de atributos de qualidade (Microsoft, 2009):

- **Disponibilidade –** A disponibilidade define a proporção de tempo em que o sistema está funcional e em funcionamento. Este pode ser medido como uma percentagem do tempo de inatividade total do sistema durante um período predefinido. Este poderá ser afetado por erros de sistema, problemas de infraestrutura, ataques maliciosos e carga do sistema.

- **Escalabilidade –** É a capacidade de um sistema lidar com aumentos de carga sem afetar o seu desempenho ou a capacidade de ser facilmente ampliada. Existem dois métodos para melhorar a escalabilidade: o dimensionamento vertical (aumento em escala, ou seja, instâncias com maior capacidade) e o dimensionamento horizontal (um maior número de instâncias). Para o dimensionamento vertical é necessário adicionar mais recursos, como por exemplo, CPU e memória. Para o dimensionamento horizontal é necessário

adicionar mais máquinas à infraestrutura que executa a aplicação e partilha a carga.

- **Interoperabilidade –** É a capacidade de um sistema ou vários sistemas diferentes operarem com sucesso, ao comunicar e efetuar a troca de informações com outros sistemas externos geridos por entidades externas. Um sistema interoperável facilita a troca e a reutilização de informações interna e externamente. Nesse sentido protocolos de comunicação, interfaces, e formatos de dados são as principais considerações para a interoperabilidade.

- **Confiabilidade –** É a capacidade de um sistema continuar a funcionar da maneira esperada ao longo do tempo, esta é medida como a probabilidade de que um sistema não falhará e que executará as funções pretendidas por um determinado intervalo de tempo.

- **Performance –** É a indicação da capacidade de resposta de um sistema para executar ações em um determinado intervalo de temo, este pode ser medido em termos de latência ou de taxa de transferência. O desempenho pode afetar diretamente a escalabilidade e a falta de escalabilidade pode afetar o desempenho.

- **Segurança –** É a capacidade de um sistema reduzir a probabilidade de ataques maliciosos ou acidentais por ações fora do uso definido que afetam o sistema e impedem a divulgação ou perda de informações. Melhorar a segurança pode aumentar a confiabilidade do sistema reduzindo assim as hipóteses de sucesso de um ataque prejudicando o sistema.

Os atributos de qualidade são os que moldam a arquitetura mais significativamente, nesse sentido as escolhas feitas durante a arquitetura de *software* vão determinar em grande parte, as formas pelas quais o sistema deverá ou não atender aos objetivos definidos pelos atributos (Microsoft, 2009).

4.1.4 Estilos de arquitetura

Estilo de arquitetura, também conhecido como estilo arquitetónico é um conjunto de princípios/padrão de granulação grossa que fornece uma estrutura abstrata para uma família de sistemas. Um estilo arquitetónico melhora a divisão e promove a reutilização do projeto ao fornecer soluções para problemas frequentemente recorrentes (Microsoft, 2009). Mais especificamente um estilo arquitetónico determina o vocabulário de componentes e conectores que podem ser usados em instâncias desse estilo, juntamente com um conjunto de restrições sobre como eles podem ser combinados (Garlan & Shaw, 1994). É importante entender que os estilos descrevem diferentes aspetos das aplicações, por exemplo, alguns estilos arquitetónicos descrevem padrões de implementação, outros estruturas de *design* e outros descrevem características de comunicação. Nesse sentido, uma aplicação geralmente usa uma combinação de um ou mais estilos de arquitetura.

Seguem alguns exemplos de estilos arquiteturais que se encontram descritos atualmente na literatura:

- **Arquitetura em camadas –** Também conhecido como padrão de arquitetura *n-tier*, é o mais comum na maioria das aplicações Java EE e, portanto, é o mais conhecido pela maioria dos arquitetos, *designers* e programadores (Richards, 2015). Este concentra-se no agrupamento de funcionalidades relacionadas em camadas distintas empilhadas verticalmente. A funcionalidade dentro de uma camada está relacionada por um papel ou responsabilidade comum (Microsoft, 2009). Embora o padrão de arquitetura em camadas não especifique o número e os tipos de camadas que devem existir no padrão, a maioria das arquiteturas consiste em quatro camadas padrão: apresentação, negócios, persistência e base de dados. Uma das principais características do padrão de arquitetura em camadas é a separação de preocupações entre os componentes, ou seja, os componentes dentro de uma camada específica lidam apenas com lógica que se relaciona àquela camada. Os conceitos-chave da arquitetura por camadas são: camada fechada *(closed layer)*, camadas de isolamento *(layers of isolation)* e camadas abertas *(open layers)* (Richards, 2015).
    - **Camada fechada –** significa que, à medida que um determinado pedido se move de uma camada para outra, este deve passar pela camada imediatamente abaixo para chegar à próxima camada abaixo dessa. Por

exemplo, um pedido originado na camada de apresentação deve primeiro passar pela camada de negócios e depois pela camada de persistência antes de atingir finalmente a camada de base de dados.

- **Camadas de isolamento –** Este conceito significa que as alterações feitas em uma camada da arquitetura geralmente não afetam ou impactam os componentes em outras camadas: a alteração é isolada nos componentes dentro dessa camada, e possivelmente em outra camada associada. De uma outra forma, significa também que cada camada é independente das outras camadas, tendo assim pouco ou nenhum conhecimento sobre o funcionamento interno das outras camadas na arquitetura.

- **Camada aberta –** Uma camada marcada como aberta, significa que as solicitações podem contornar essa camada e ir diretamente para a camada abaixo dela. Enquanto as camadas fechadas facilitam camadas de isolamento e, portanto, ajudam a isolar as mudanças dentro da arquitetura, existem momentos em que faz sentido para determinadas camadas serem abertas. Por exemplo, classes utilitárias que contem componentes de serviço comuns acedidos por componentes de outra da camada.

- **Arquitetura Orientada a Eventos –** O padrão de arquitetura orientada a eventos é uma abordagem popular para o *design* de sistemas distribuídos assíncronos, bastante utilizada para criar aplicações altamente escaláveis. A sua flexibilidade permite ser utilizada em cenários que variam desde pequenas aplicações a grandes e complexas. Esta arquitetura é composta por componentes de processamento de eventos independentes e especializados, que os recebem e processam de forma assíncrona (Richards, 2015). O modelo baseado em eventos reage a uma situação específica e toma uma ação com base nesse evento. Um exemplo disso é enviar uma licitação para um produto específico num leilão online. Enviar uma licitação não é uma solicitação feita ao sistema, mas sim um evento que ocorre após o anúncio do preço atual. O sistema deve responder a esse evento comparando a licitação com outras recebidas ao mesmo tempo para determinar quem é o licitante atual mais alto (Richards & Ford, 2020). Existem duas tipologias principais dentro

da arquitetura orientada a eventos: as tipologias *mediator* e *broker* (Richards, 2015).

- ***Mediator* –** É comum ser utilizada quando é necessário controlar o fluxo de trabalho de um processamento de um evento. Esta tipologia é útil para eventos que possuem várias etapas e exigem algum nível de complexidade de orquestração para processar o evento. Existem quatro tipos principais de componentes de arquitetura dentro da tipologia do mediador: filas de eventos, um mediador de eventos, canais de eventos e processadores de eventos. O fluxo de eventos começa quando um cliente envia um evento para uma fila, que é usada para transportar o evento para o mediador. O mediador recebe o evento inicial e orquestra-o enviando eventos assíncronos adicionais para canais de eventos para executar cada etapa do processo. Os processadores, que estão à escuta nos canais de eventos, recebem e executam lógica de negócios específica para processar o evento.

- ***Broker* –** Esta difere da tipologia de *mediator* pelo fato de não haver um mediador central de eventos; em vez disso, o fluxo de mensagens é distribuído entre os componentes do processador de eventos de maneira encadeada por meio de um corretor de mensagens. Esta tipologia é utilizada quando é necessário um alto nível de respostas e controlo dinâmico sobre o processamento de um determinado evento. Existem dois principais componentes nesta tipologia: um componente de *broker* e um de processador de eventos. Cada componente do processador de eventos é responsável por processar um evento e publicar um novo evento que indica a ação que acabou de realizar. Esses novos eventos são então capturados por outros componentes de processadores de eventos, e a cadeia de eventos continua pelo sistema até que não haja mais eventos publicados para aquele evento iniciador específico.

- **Arquitetura de Microsserviços -** O padrão arquitetónico de microsserviços está a crescer e a ganhar espaço na indústria como uma alternativa viável às aplicações monolíticas e arquiteturas orientadas a serviços. Existem alguns conceitos-chave que se aplicam ao padrão geral desta arquitetura. O primeiro é a noção de unidades de implementação separada *(separately deployed units)*, este significa que cada

componente da arquitetura é implementado como uma unidade separada, permitindo uma implementação mais fácil por meio de um *pipeline* de entrega eficaz e simplificado, aumentando a escalabilidade e um alto grau de desacoplamento de aplicações e componentes. Outro conceito e talvez o mais importante é o de componente de serviço *(service component).* Em vez de pensar em serviços dentro de uma arquitetura de microsserviços, é melhor pensar em componentes de serviço, que podem variar em granularidade, desde um único módulo até uma grande parte da aplicação. Componentes de serviço contêm um ou mais módulos que representam uma função de propósito único ou uma parte independente de uma grande aplicação de negócios. Outro conceito-chave dentro do padrão de arquitetura de microsserviços é de que se trata de uma arquitetura distribuída *(distributed architecture)*, o que significa que todos os componentes dentro da arquitetura estão totalmente desacoplados entre si e acedidos por meio de algum tipo de protocolo de acesso remoto (por exemplo, REST, SOAP, etc.) (Richards, 2015). Numa arquitetura distribuída, cada serviço é executado no seu próprio processo, originalmente isto implicava um computador físico, mas evoluiu rapidamente para máquinas virtuais e *containers*. Desacoplar os serviços a esse grau permite uma solução simples para um problema comum em arquiteturas que apresentam infraestrutura *multitenant* para hospedar aplicações (Richards & Ford, 2020). Existem várias formas de implementar o padrão de arquitetura de microsserviços, no entanto três tipologias principais se destacam entre as mais comuns e populares: a tipologia baseada em API REST *(API REST-based)*, a tipologia baseada em aplicação REST *(application REST-based)* e a tipologia de mensagens centralizadas *(centralized messaging)* (Richards, 2015).

- **Tipologia baseada em *API REST* –** Esta tipologia é composta por componentes de serviço muito granulares, que contêm um ou dois módulos que executam funções de negócios específicas independentes dos restantes serviços. Os componentes de serviço são normalmente acedidos através de uma interface baseada em REST implementada por meio de uma camada de API. Esta tipologia é útil para sites que expõem serviços individuais pequenos e autocontidos por meio de algum tipo de API.

- **Tipologia baseada em aplicação *REST*–** Esta tipologia difere da abordagem de API REST no sentido de que as solicitações são recebidas

por interfaces visuais de aplicações empresariais baseadas na web tradicionais ou clientes pesados, em vez de passarem por uma camada simples de API. Os componentes de serviço tendem a ser maiores, mais granulares e representam uma pequena parte da aplicação de negócios como um todo, em vez de serviços granulares únicos de ação.

    - **Tipologia de mensagens centralizadas –** É semelhante à tipologia de aplicações baseada em REST, exceto que, em vez de usar REST para acesso remoto, esta utiliza um *broker* de mensagens centralizadas.

- **Arquitetura cliente/servidor –** Descreve sistemas distribuídos que envolvem um sistema cliente e outro servidor separados e uma rede de conexão (Microsoft, 2009). Neste tipo de arquitetura o cliente tem uma interface para solicitar um serviço ao servidor e este retorna a resposta assim que estiver pronta, enquanto os servidores estão localizados num outro local da rede e normalmente em máquinas mais potentes (Cyber Talents, 2023).

- **Arquitetura baseada em componentes –** Esta descreve uma abordagem de engenharia de *software* para sistemas de *design* e desenvolvimento, nesse sentido este estilo concentra-se na decomposição do *design* em componentes funcionais ou lógicos que expõem interfaces de comunicação bem definidas com métodos, eventos e propriedades (Microsoft, 2009). Esses componentes podem ser armazenados em uma biblioteca de componentes e inseridos em uma aplicação sem a necessidade de modificação de outros componentes, desta forma permite mais flexibilidade e capacidade de modificação (Datamyte, 2022).

- **Arquitetura orientada a serviços (SOA) –** Este estilo permite que as funcionalidades de uma aplicação sejam fornecidas através de um conjunto de serviços fracamente acoplados (Microsoft, 2009). Os serviços usam padrões de interface comuns e um padrão de arquitetura para que possam ser rapidamente incorporados em novas aplicações, reduzindo a necessidade de recriar ou duplicar funcionalidades já existentes, bem como a complexidade de entender como estabelecer a interoperabilidade com funções pré-existentes (IBM, 2023a).

4.1.5 Contribuição para o projeto

No contexto deste projeto de dissertação de mestrado, a solução a nível de arquitetura tem diversos componentes. O sistema foi desenhado de forma modular, com o objetivo de garantir diversas camadas de abstração. Nesse sentido, a arquitetura definiu a estrutura e organização do sistema, permitindo uma clara separação de responsabilidades entre os componentes do *middleware.* Esta separação vai de encontro com os princípios-chave da arquitetura de *software (separation of concerns e single responsability principle)* e estes tornam-se essenciais tendo em conta a complexidade do sistema, onde várias camadas e módulos precisam trabalhar juntos de forma eficiente. Uma arquitetura sólida e bem definida como a presente na solução apresentada, torna o *middleware* escalável, facilitando a adição de recursos à medida que os requisitos vão mudando ou na adição de novos clientes conforme demonstrado ao longo desta integração. Esta arquitetura permite ainda garantir os atributos de qualidade, uma vez que é de fácil manutenção e evolução, minimizando riscos e facilitando também a interoperabilidade com os diferentes sistemas e tecnologias existentes. Em resumo, a arquitetura de *software* é um elemento importantíssimo tanto na construção, evolução como na manutenção da solução de *middleware*.

4.2 Middleware

Esta secção tem como objetivo introduzir o conceito de *middleware* e também apresentar os diferentes tipos disponíveis. Além disso, é apresentada a contribuição deste conceito de *middleware* para o projeto de dissertação de mestrado em questão.

4.2.1 Definição

O termo middleware apareceu pela primeira vez no final da década de 1980, era utilizado para descrever um *software* de gestão de conexão de rede. Um *middleware* pode ser descrito como uma camada de *software* que se encontra acima da rede e do sistema operativo e abaixo das aplicações (Pinus, 2004). *Middleware* é também o nome dado a um *software* ou serviços na *cloud* que fornecem funcionalidades e recursos a aplicações e ajudam os programadores e operadores a criar e implementar aplicações com mais eficiência. O *middleware* tem o papel de conectar apps, dados e utilizadores (Red Hat,

2023). O *Middleware* é o *software* que reside entre um sistema operativo e as aplicações que são executadas no mesmo. O *middleware,* que funciona essencialmente como uma camada de tradução oculta, possibilita a comunicação e a gestão de dados para aplicações distribuídas. É, por vezes, denominado de "*plumbing*" (canalização), uma vez que liga duas aplicações para que os dados e as bases de dados possam ser facilmente passados pelo "*pipe*" (cano) (Microsoft, 2023c). De uma forma resumida o *middleware* é um *software* que age como uma camada, capaz de interligar várias tecnologias de *software*, de modo que as informações provenientes de diferentes fontes sejam canalizadas assim como as suas diferenças entre protocolos, plataformas, arquiteturas e ambientes.

4.2.2 Tipos de Middleware

Neste tópico serão abordados os quatro tipos de *middleware* encontrados na literatura, sendo estes: *transactional*; *procedural*; *message-oriented; object-oriented*.

- ***Transactional Middleware* (TM)** – É um termo genérico usado para referir-se a infraestruturas de IT que suportam a execução de transações eletrónicas em ambientes distribuídos (Card et al., 1999). *Transactional middleware* (TM) ou *Transaction Processing monitors* foram projetados para suportar transações síncronas distribuídas, nesse sentido o principal objetivo é de coordenar pedidos entre clientes e servidores (Pinus, 2004). Atualmente os *transaction processing monitors* estão no centro da maioria dos servidores aplicacionais e são uma componente chave de qualquer arquitetura empresarial (Card et al., 1999)

- ***Procedural Middleware* (PM) –** Também conhecido como *Remote Procedure Calls* (RPCs), foram desenvolvidos pela Sun Microsystems no início dos anos 80. RPCs são representados em diferentes sistemas operacionais, incluindo a maioria dos sistemas Unix e Microsoft Windows (Pinus, 2004). Este tipo de *middleware* permite que aplicações chamem procedimentos em aplicações remotas como se fossem chamadas locais, este implementa um mecanismo de ligação que localiza procedimentos remotos e os disponibiliza de forma transparente para um chamador. Tradicionalmente, esse tipo de *middleware* lidava com programas baseados em procedimentos, atualmente também inclui componentes baseados em objetos (Oracle, 2023).

- ***Message Oriented Middleware* (MOM) –** É um *middleware* que permite a comunicação através de mensagens. Um cliente faz uma chamada a uma API para enviar uma mensagem a um destino gerido pelo *provider*, este depois de enviar a mensagem pode continuar a fazer outro trabalho, confiante de que o *provider* retém a mensagem até que um cliente recetor a receba (Oracle, 2023). Existem dois tipos diferentes, *message queuing* e *message passing. Message queuing* é definido como um modelo de comunicação indireto, onde a comunicação acontece através de uma fila. No tipo *message passing* a comunicação é direta, as informações são enviadas às partes interessadas (Pinus, 2004).

- ***Object-oriented Middleware* (OOM)** – O *middleware* de orientação a objetos é desenvolvido a partir de RPCs (*Remonte Procedue Calls),* adicionando os conceitos de desenvolvimento de *software* de orientação por objetos (Pinus, 2004). Trata-se de uma das técnicas mais concebidas para lidar com as complexidades acidentais e inerentes de aplicativos em rede se concentram no *middleware* orientado a objetos. O *middleware* orientado a objetos fornece recursos cujas qualidades são críticas para ajudar a simplificar e coordenar como as aplicações em rede são conectadas e como trabalham em conjunto (informIT, 2023).

4.2.3 Contribuição para o projeto

Este projeto de dissertação de mestrado concentra-se na adaptação e integração de uma solução *middleware* conhecida como Digital Bank Integration, desenvolvida pela Innovation Makers (INM), em um novo cenário com o objetivo de atender às necessidades de um banco português. O *middleware* atua como uma camada intermediária que facilita a comunicação e a integração entre várias entidades existentes, sejam elas instituições bancárias ou não bancárias. Em resumo, no contexto deste projeto o *middleware* é essencialmente uma aplicação intermédia que permite a interligação com várias outras aplicações de forma a possibilitar o cliente (banco português) disponibilizar operações bancarias online aos seus utilizadores.

4.3 Padrões de desenho

Esta secção tem o propósito de introduzir os conceitos de padrões de software, em particular os padrões de injeção de dependência e *decorator*, que são os mais relevantes para este projeto. Além disso, será destacada a contribuição desses conceitos no contexto deste projeto de dissertação de mestrado.

4.3.1 Definição

O objetivo de um padrão é uma solução específica, uma que seja comum e eficaz para lidar com um ou mais problemas recorrentes. Outra maneira de olhar é a de que um padrão é um conjunto de conselhos, e a arte de criar padrões é dividir muitos pedaços de conselhos em partes relativamente independentes, com o intuito de discuti-los e referenciá-los de forma mais ou menos separada (Flower, 2003). Os padrões de design são soluções típicas para problemas comuns que ocorrem no *design* de *software*. Eles são como plantas pré-fabricadas que podemos personalizar para resolver um problema de design recorrente em código. Os padrões são frequentemente confundidos com algoritmos, porque ambos descrevem soluções típicas para alguns problemas conhecidos. Enquanto um algoritmo define um conjunto claro de ações que podem alcançar um objetivo, um padrão é uma descrição mais abstrata de uma solução. (Refactoring.Guru, 2023). Nesse sentido, os padrões de desenho, também conhecidos como padrões de projeto ou *design patterns* em inglês, são soluções reutilizáveis para problemas comuns que surgem durante o desenvolvimento de *software*. Estes representam as melhores práticas e abordagens testadas pelo tempo para resolver problemas específicos. Os padrões fornecem diretrizes e estruturas que devem ser aplicadas no desenvolvimento de *software* de forma a melhorar a qualidade, manutenibilidade e eficiência do código. Segundo (Gamma et al., 1995)os padrões podem ser categorizados pelo seu propósito ou intenção. Conforme proposto pelos padrões GoF “Gang of Four”, estes podem ser divididos em padrões de criação *(creational)*, estruturais (structural) e comportamentais *(behavioral)*. Os padrões de criação tratam do processo de criação de objetos. Os padrões estruturais lidam com a composição de classes ou objetos. Os padrões comportamentais caracterizam as formas pelas quais as classes ou objetos interagem e distribuem responsabilidades. De seguida serão apresentados os padrões de desenhos mais utilizados no *middleware* deste trabalho de projeto de dissertação de mestrado.

4.3.2 Padrão Injeção de dependência

A injeção de dependência ou "*Dependency Injection*" em inglês, é uma forma de desenvolver *software* de modo que vários componentes estejam relacionados de uma maneira muito fracamente acoplada (Rojas, 2023). Algumas pessoas referem-se à injeção de dependência (DI) como *Inversion of Control* (IoC). O IoC baseia-se na ideia de que o controlo sobre a execução de determinadas ações é invertido ou transferido de um componente para outro, muitas vezes controlado por um container ou *framework*. Antes da injeção de dependência ter um nome, as pessoas referiam-se às bibliotecas de gestão de dependências como *"Inversion of Control Containers"* e, em consequência, o significado de IoC gradualmente se adaptou para: *"Inversion of Control over Dependencies"*. Martin Fowler introduziu o termo *"Dependency Injection"* para se referir especificamente ao IoC no contexto da gestão de dependências (Seemann & Van Deursen, 2019). Na maioria das aplicações de *software*, existe sempre algum tipo de dependência entre as várias partes e peças (Rojas, 2023). (Seemann & Van Deursen, 2019) referem que a DI é um dos conceitos mais mal compreendidos da programação orientada a objetos e trata-se de um conjunto de princípios e padrões de *design* de *software* que permite desenvolver código com baixo acoplamento.

4.3.2.1 Propósito

O principal benefício da injeção de dependência (DI) é de promover a desvinculação entre os objetos de um determinado sistema. A desvinculação *(loose coupling)* é um princípio de *design* que visa reduzir a interdependência entre os componentes do sistema. Quando os objetos são fracamente vinculados, as mudanças num determinado objeto não implicam necessariamente mudanças nos outros. Desta forma existe uma redução significativa da complexidade e é melhorada a capacidade de manutenção da aplicação. A injeção de dependência consiste em separar a responsabilidade de criar objetos da classe ou função que os utiliza. Essa separação é alcançada através do uso de interfaces, que definem os métodos e propriedades que uma classe deve implementar. A classe que requer a dependência baseia-se apenas na interface e não na implementação concreta (Rojas, 2023).

4.3.2.2 Exemplificação

A injeção de dependência é um tópico abrangente com diversas técnicas de aplicação, de forma a facilitar a sua compreensão, nos próximos passos, é explicada a implementação de uma classe cujo objetivo é imprimir "Hello DI" na consola através de injeção de dependência.

```
private static void Main()
{
    IMessageWriter writer = new ConsoleMessageWriter();
    var salutation = new Salutation(writer);
    salutation.Exclaim();
}
```

Figura 2 - DI exemplo main classe

Fonte: (Seemann & Van Deursen, 2019)

Conforme podemos observar na figura acima, o programa cria uma instância de ConsoleMessageWriter que encapsula a funcionalidade de escrita de uma mensagem na consola, este é posteriormente passado para a classe Salutation, para que a sua instância saiba onde escrever as mensagens.

```
public class Salutation
{
    private readonly IMessageWriter writer;

    public Salutation(IMessageWriter writer)
    {
        if (writer == null)
            throw new ArgumentNullException("writer");

        this.writer = writer;
    }

    public void Exclaim()
    {
        this.writer.Write("Hello DI!");
    }
}
```

Provides the Salutation class with the IMessageWriter DEPENDENCY using CONSTRUCTOR INJECTION

Guard Clause verifies that the supplied IMessageWriter isn't null

Sends the Hello DI! message to the IMessageWriter DEPENDENCY

Figura 3 - DI exemplo

Fonte: (Seemann & Van Deursen, 2019)

A classe Salutation por sua vez depende da interface IMessageWriter, que é solicitada pelo construtor. Esta prática é chamada de injeção de construtor e consiste em definir a

lista de dependências necessárias como parâmetros do construtor da classe requerente. A interface IMessageWriter é injetada na classe Salutation que neste exemplo contém apenas um método denominado write que recebe a mensagem a ser escrita.

```
public class ConsoleMessageWriter : IMessageWriter
{
    public void Write(string message)
    {
        Console.WriteLine(message);
    }
}
```

Figura 4 - DI classe implementação

Fonte: (Seemann & Van Deursen, 2019)

Conforme podemos verificar, a classe ConsoleMessageWriter implementa a interface IMessageWriter e escreve na consola a mensagem pretendida. Este exemplo simples ilustra como foram desvinculadas as funcionalidades da classe Main, que poderia ter escrito a mesma mensagem com apenas uma linha de código. No entanto, com esta desvinculação, torna-se possível fazer várias alterações ou implementar outras funcionalidades sem a necessidade de modificar as outras classes. Isto significa que é possível estender ou modificar o comportamento da aplicação de forma mais flexível e modular, sem afetar o restante do código.

Um exemplo de modificação do comportamento sem alterar as outras classes do programa, poderia ser:

```
1.  using System;
2.
3.  public class Program
4.  {
5.      public static void Main()
6.      {
7.          IMessageWriter writer = new MessageBoxMessageWriter();
8.          var salutation = new Salutation(writer);
9.          salutation.Exclaim();
10.     }
11. }
12.
13. public class MessageBoxMessageWriter : IMessageWriter
14. {
15.     public void Write(string message)
16.     {
17.         // Exibir a mensagem em uma caixa de diálogo
18.         System.Windows.Forms.MessageBox.Show(message);
19.     }
20. }
```

No exemplo acima desenvolvido, em vez de escrever a mensagem na consola, foi criada uma implementação MessageBoxMessageWriter da interface IMessageWriter para exibir a mensagem numa caixa de diálogo. O resto do código permaneceu inalterado, e a injeção de dependência permite alternar facilmente entre diferentes implementações da interface IMessageWriter.

4.3.3 Padrão Decorator

O padrão decorator fornece uma forma de adicionar ou remover funcionalidades a um componente de forma flexível, sem alterar sua aparência ou função externa. Este faz parte dos padrões GoF, classificado como sendo do tipo de padrões *design* estruturais que descrevem maneiras eficazes de dividir e combinar os elementos de uma aplicação (Stelting & Maassen, 2001).

O padrão decorator deve ser utilizado quando é necessário atribuir comportamentos extras a objetos em tempo de execução sem alterar o código que os utiliza ou quando é difícil ou impossível estender o comportamento de um objeto através da herança. Para uma classe final, a única maneira de reutilizar o comportamento existente seria envolver a classe com seu próprio invólucro, usando o padrão decorator (Refactoring.Guru, 2023).

4.3.4 Contribuição para o projeto

No âmbito deste projeto de dissertação de mestrado, a injeção de dependências está bastante presente uma vez que desempenha um papel fundamental na arquitetura da solução. O *middleware* usado foi projetado para fornecer funcionalidades personalizadas a diferentes clientes, onde cada um tem os seus próprios requisitos. Utilizando o exemplo de DI acima demonstrado na subsecção 4.3.2.2, e imaginando que a classe Main é uma determinada operação do *middleware*, dois clientes (ConsoleMessageWriter e MessageBoxMessageWriter) podem, para a mesma operação, ter comportamentos (implementações) diferentes. Nesse sentido, a injeção de dependência, destacou-se como uma abordagem essencial para implementar de forma isolada e independente as lógicas específicas de cada cliente. Este padrão não só simplificou o processo de desenvolvimento, mas também facilitou a manutenção e evolução do sistema uma vez que não afetou o funcionamento do *middleware* como um todo. A injeção de dependência permitiu também a reutilização de componentes comuns entre os diferentes módulos do

middleware (como por exemplo, operações), originando um código mais limpo, eficiente e escalável, tendo em conta que os componentes partilhados pelos diferentes clientes podem ser melhorados e atualizados centralmente.

Relativamente ao padrão decorator, este é utilizado nas operações do *middleware*, de forma a possibilitar que estas tenham algumas funcionalidades adicionais. Um exemplo dessas funcionalidades, envolve a inclusão de registos *(logs)* que possibilitam a monitorização do processo de execução de cada operação. Outro exemplo é a possibilidade de adicionar mensagens de resposta de sucesso personalizadas para cada operação com base no código de erro.

4.4 Maven

Nesta secção, o objetivo é explorar a *framework* Maven, a estrutura dos projetos que a utilizam, e os seus principais componentes, nomeadamente, a suas coordenadas, os seus repositórios e dependências. Além disso, será demonstrado como esta *framework* contribui para o contexto deste projeto de dissertação de mestrado.

4.4.1 Definição

O Maven, cujo nome é uma palavra que significa "acumulador de conhecimento", teve origem como uma iniciativa para simplificar os processos de construção no projeto Jakarta Turbine (Apache Maven, 2023). Anteriormente, existiam vários projetos, cada um com os seus próprios arquivos de construção (por exemplo Ant), tornando o processo de construção complexo e desorganizado. O Maven foi criado para estabelecer um método comum para a construção de projetos, proporcionando uma definição clara do que constitui um determinado projeto, facilitando a publicação de informações e permitindo a partilha de JARs (Java Archive) entre diferentes projetos.

Em termos mais formais, o Maven é uma ferramenta de gestão de projetos que define um Modelo de Objeto de Projeto (*Project Object Model*), segue normas, estabelece um ciclo de vida de projeto, gere dependências e executa lógica para atingir objetivos de plugins em fases definidas no ciclo de vida. Embora seja frequentemente definido como uma ferramenta de construção, o Maven vai além disso, proporcionando recursos de gestão de projeto. Este simplifica o desenvolvimento e a colaboração em projetos Java, automatiza tarefas repetitivas, gere dependências e permite a criação de relatórios e sites. O Maven é

projetado para facilitar o trabalho diário dos programadores e melhorar a compreensão de qualquer projeto baseado em Java (Maven: The Definitive Guide, 2009).

4.4.2 Estrutura Projeto

O Maven incorpora o conceito de convenção sobre configuração fornecendo comportamentos padrão para os projetos. Sem personalização, o código-fonte é assumido estar em ${basedir}/src/main/java e os recursos são assumidos estar em ${basedir}/src/main/resources. Os testes são assumidos estar em ${basedir}/src/test e o projeto é assumido produzir um arquivo JAR (Java ARchive). O Maven pressupõe que é necessário compilar o código-fonte em ${basedir}/target/classes e, em seguida, criar um arquivo JAR distribuível em ${basedir}/target (Maven: The Definitive Guide, 2009). Nesse sentido as seguintes diretorias fazem parte de um projeto maven:

```
meu-projeto/
├── src/
│   ├── main/
│   │   ├── java/
│   │   └── resources/
│   └── test/
│       ├── java
│       └── resources
├── target/
├── pom.xml
└── outras pastas e arquivos do projeto
```

Figura 5 - Estrutura projeto Maven

- **pom.xml** – Conhecido como POM (Project Object Model) é o ficheiro de configuração do projeto. Este descreve as configurações, dependências, plugins e outros detalhes. É onde também constam as informações do projeto, como nome, descrição, versão e outras configurações específicas do Maven. O Maven é executado com base em um POM efetivo, que é uma combinação de configurações do arquivo pom.xml do projeto, todos os POMs "pai", um super

POM definido no próprio Maven, onde contém um conjunto de configurações e perfis ativos (Maven: The Definitive Guide, 2009).

- **src/main** – O código-fonte e os recursos são organizados sob a diretoria src/main. Para um projeto Java simples, isso envolve algumas classes Java e arquivos de propriedades. Noutro tipo de projeto, esta diretoria pode ser a raiz de uma aplicação web ou conter arquivos de configuração para um servidor de aplicações. No caso de projetos Java, as classes Java são colocadas em src/main/java, e os recursos do *classpath* são armazenados em src/main/resources.

- **src/test** – Os testes são organizados na diretoria src/test. Nesta estrutura, incluem-se as classes de teste em Java, como as usadas nas *frameworks* de testes unitários, JUnit ou TestNG, estas são armazenadas em src/test/java. Além disso, os recursos relacionados ao *classpath* necessários para esses testes são colocados na diretoria src/test/resources.

- **target** – Esta é uma diretoria gerada automaticamente pelo Maven e contém os artefactos de compilação produzidos, como por exemplo, os arquivos JAR, WAR.

4.4.2.1 Projeto Múltiplos Módulos

A estrutura apresentada na Figura 5 pode representar um projeto, ou um único módulo num conjunto de módulos de um determinado projeto. A complexidade de um projeto de nível empresarial requer por vezes múltiplas bases de dados, integração com sistemas externos e subprojectos que podem ser divididos por vários departamentos. Esses projetos normalmente abrangem milhares de linhas de código e envolvem o esforço de dezenas ou centenas de programadores de software (Maven: The Definitive Guide, 2009). O seguinte exemplo exemplifica a complexidade de uma aplicação empresarial que produz duas aplicações: uma ferramenta de consulta de linha de comando para o *feed* de previsão do tempo do Yahoo! e uma aplicação da web:

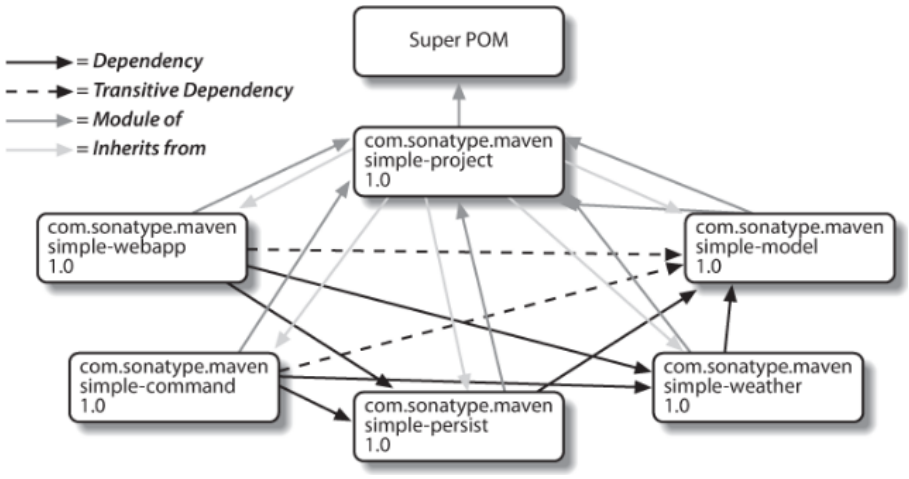

Figura 6 - Exemplo projeto de múltiplos módulos

Fonte: (Maven: The Definitive Guide, 2009)

Conforme podemos visualizar na figura acima os projetos Maven podem ser constituídos por vários módulos interligados através de um projeto POM "pai" onde cada um tem a sua própria responsabilidade. No exemplo da figura Figura 6 podemos verificar que existe um módulo simple-model que é responsável por definir um modelo de objeto simples que representa os dados retornados da previsão do tempo do Yahoo!. O módulo simple-weather é responsável por toda a lógica necessária para recuperar os dados da previsão do tempo e analisar os resultados. O módulo simple-persist contém alguns objetos de acesso a dados *(data access object)* que estão configurados para armazenar a informação relativa a previsão do tempo na base de dados. O módulo simple-webapp que representa o projeto de aplicação web e o módulo simple-command que contém uma ferramenta de linha de comando simples que pode ser usada para consultar a previsão do tempo do Yahoo!.

4.4.3 Coordenadas

O POM atribui um nome ao projeto, disponibiliza um conjunto de identificadores exclusivos (coordenadas) e estabelece as relações com outros projetos através de dependências, projetos-pai e pré-requisitos. Além disso, o POM permite personalizar o comportamento dos plugins e incluir informações sobre a comunidade e os programadores envolvidos no projeto. Por sua vez, as coordenadas do Maven consistem num conjunto de identificadores que permitem identificar de forma única um projeto,

uma dependência ou um plugin num ficheiro POM do Maven (Maven: The Definitive Guide, 2009).

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0"
  http://maven.apache.org/maven-v4_0_0.xsd"
  <modelVersion>4.0.0</modelVersion>
  <groupId>mavenbook</groupId>
  <artifactId>my-app</artifactId>          coordinates
  <version>1.0-SNAPSHOT/<version>
  <name>Maven Quick Start Archetype</name>
  <url>http://maven.apache.org</url>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
</project>
```

Figura 7 - Exemplo POM e Coordenadas Maven

Fonte: (Maven: The Definitive Guide, 2009)

Conforme podemos visualizar na figura acima as coordenadas destacadas são: groupId, artifactId, version e packaging. O groupId representa o grupo, empresa, equipa, organização, projeto ou outro grupo. A convenção para identificadores de grupo é que estes devem começar pelo nome de domínio reverso da organização que cria o projeto (por exemplo, projetos na Fundação de Software Apache devem ter o groupId como org.apache). O artifactId representa o identificador único sob o groupId que representa um único projeto. A versão *(version)* representa a versão especifica do projeto.

No exemplo acima ilustrado, as coordenadas estão representadas por mavenbook:my-app:jar:1.0-SNAPSHOT. Esta notação também se aplica às dependências do projeto tal como em qualquer outro sistema, as coordenadas do Maven são um endereço para um ponto específico no 'espaço' do geral para o específico. O Maven localiza um projeto pelas suas coordenadas quando um projeto se relaciona com outro, seja como uma dependência, um plugin ou uma referência a um projeto pai. O *packaging* de um projeto é também um componente importante nas coordenadas, no entanto, não faz parte dos identificadores exclusivos de um projeto. Nesse sentido, o *packaging* representa o tipo de pacote produzido por um determinado projeto, por exemplo, um projeto com *packaging* jar produz um arquivo JAR; um projeto com *packaging* war produz uma aplicação web. (Maven: The Definitive Guide, 2009).

4.4.4 Repositórios

Artefactos e *plugins* são descarregados de um repositório remoto e a razão pela qual o *download* inicial do Maven é tão pequeno (1,5 MB) deve-se ao facto que este não inclui muitos *plugins*. Este traz apenas o essencial e recorre a um repositório remoto quando necessário. Um repositório é definido pela sua estrutura, ou seja, é uma coleção de artefactos de projetos armazenados de forma que o Maven consiga compreendê-los facilmente. Tudo é organizado numa estrutura de diretorias que refletem de perto as coordenadas de um projeto. O Maven descarrega os artefactos e *plugins* de um repositório remoto para a máquina local e guarda-os no seu próprio repositório local. Uma vez que este tenha descarregado um artefacto a partir do repositório remoto, nunca mais irá precisar de o descarregar novamente, dado que o Maven verificará sempre a sua existência no repositório local antes de procurar noutro local.

Com frequência, ao desenvolver projetos que dependem de bibliotecas que não são de acesso gratuito ou públicas, torna-se necessário configurar um repositório personalizado dentro de uma rede organizacional ou descarregar e instalar manualmente as dependências. Nesse sentido, os repositórios remotos padrão podem ser substituídos ou complementados com referências a repositórios Maven geridos e mantidos por uma organização. Em grandes organizações, um repositório interno do Maven pode ser uma maneira eficiente para diferentes departamentos partilharem *releases* ou *snapshots* de desenvolvimento entre si. Existem vários produtos disponíveis que permitem que as organizações gerenciem e mantenham espelhos dos repositórios públicos do Maven (Maven: The Definitive Guide, 2009).

4.4.5 Dependências

As dependências no Maven referem-se a artefactos ou *plugins* de software que um projeto ou aplicação precisa para funcionar corretamente. Essas dependências são essenciais para que o código-fonte do projeto possa ser compilado, executado e funcione conforme esperado. A seguinte imagem ilustra um exemplo de algumas dependências necessárias para uma aplicação meteorológica:

```
<project>
  [...]
  <dependencies>
    <dependency>
      <groupId>log4j</groupId>
      <artifactId>log4j</artifactId>
      <version>1.2.14</version>
    </dependency>
    <dependency>
      <groupId>dom4j</groupId>
      <artifactId>dom4j</artifactId>
      <version>1.6.1</version>
    </dependency>
    <dependency>
      <groupId>jaxen</groupId>
      <artifactId>jaxen</artifactId>
      <version>1.1.1</version>
    </dependency>
    <dependency>
      <groupId>velocity</groupId>
      <artifactId>velocity</artifactId>
      <version>1.5</version>
    </dependency>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
  [...]
</project>
```

Figura 8 - Dependências Maven

Fonte: (Maven: The Definitive Guide, 2009)

Conforme podemos visualizar na figura acima, este ficheiro POM conta com cinco dependências, sendo elas log4j, dom4j, jaxen, velocity e junit. As coordenadas para estes artefactos podemos ser encontradas em http://www.mvnrepository.com, onde é fornecida uma interface de pesquisa para o repositório do Maven.

4.4.6 Contribuição para o projeto

No âmbito deste projeto de dissertação de mestrado, o Apache Maven desempenha um papel fundamental no suporte à arquitetura descrita na secção 3.2. A solução de *middleware* Digital Bank Integration da INM, é um projeto desenvolvido em Java que se baseia na abordagem do Maven e adota uma estrutura composta por vários módulos interconectados. Cada um desses módulos tem funções e responsabilidades específicas dentro do contexto do projeto global e são armazenados/geridos por um repositório Maven interno. O Maven atua como a ferramenta central que viabiliza a construção, gestão e integração de todos os componentes abrangidos pelo projeto. A abordagem modular não apenas facilita a organização eficiente do código-fonte, mas também simplifica a gestão de dependências, possibilitando, assim, a escalabilidade do sistema.

4.5 Gestão de identidade e acessos

Nesta secção será apresentado o conceito de gestão de identidade e acessos, os seus conceitos chave e os seus respetivos tipos. Seguidamente é apresentado um exemplo de

uma solução para gestão de identidade e acessos e por último e será demonstrada a contribuição deste conceito no contexto deste projeto de dissertação de mestrado.

4.5.1 Definição

A gestão de identidade e acessos, ou "*Identity and Access Management"* no inglês, é uma disciplina de ciber segurança focada na gestão das identidades de utilizadores e nas permissões de acesso numa rede de computadores. Embora as políticas, processos e tecnologias de *Identity and Access Management* (IAM) possam variar entre empresas, o objetivo de qualquer iniciativa de IAM é assegurar que os utilizadores e os dispositivos certos possam aceder aos recursos adequados pelas razões apropriadas no momento adequado (IBM, 2023b). A IAM desempenha um papel fundamental na ciber segurança, pois ajuda uma organização a encontrar o equilíbrio adequado para manter os dados e recursos críticos fora do alcance da maioria, enquanto os torna acessíveis a quem precisa. Além disso, a importância da IAM é ainda mais evidente devido à constante evolução dos métodos utilizados por criminosos cibernéticos. Ataques sofisticados, como os e-mails de *phishing*, tornaram-se uma das fontes mais comuns de *hacking* e violações de dados, visando utilizadores com acesso existente. As regulamentações como o GDPR e o PCI-DSS estabelecem requisitos rigorosos para determinar quem tem acesso a dados e com que finalidades, nesse sentido os sistemas de IAM capacitam as empresas a estabelecer e implementar políticas estritas de controlo de acesso em conformidade com essas regulamentações. Além disso, as empresas têm a capacidade de monitorizar a atividade dos utilizadores para comprovar a conformidade durante os processos de auditoria. Com a crescente adoção de ambientes *cloud*, dispositivos IoT *(Internet of Things)*, trabalho remoto e BYOD *(bring your own device)*, as empresas enfrentam a necessidade de proporcionar um acesso seguro a uma variedade maior de utilizadores e recursos. Desta forma, os sistemas de IAM têm a capacidade de centralizar o controlo de acesso para todos os utilizadores e ativos numa rede, garantindo a segurança da rede sem prejudicar a experiência do utilizador (IBM, 2023b). Sem a implementação da IAM, a gestão do acesso aos sistemas de uma organização torna-se desafiadora, o que pode resultar em dificuldades tanto na identificação de quem tem acesso quanto na revogação do acesso de utilizadores comprometidos (Microsoft, 2023b).

4.5.2 Conceitos chave

As redes das empresas são únicas, as políticas, processos e ferramentas que cada empresa utiliza para construir um sistema de gestão de identidade e acesso são igualmente únicos. Nesse sentido, a maioria, se não todas, as implementações de IAM abrangem quatro funções-chave: gestão do ciclo de vida da identidade; controlo de acesso; autenticação e autorização e gestão da identidade (IBM, 2023b).

- **Gestão do ciclo de vida da identidade** – A gestão do ciclo de vida da identidade envolve o processo de criar e manter uma identidade digital para cada entidade, seja humana ou não, numa rede. Essa identidade digital fornece informações à rede sobre quem ou o que cada entidade é e quais ações ela está autorizada a realizar na rede. Normalmente, essa identidade inclui informações padrão do utilizador, como nome, número de identificação, credenciais de *login*, entre outras, bem como detalhes sobre a função da entidade na organização, das suas responsabilidades e das permissões de acesso correspondentes.

- **Controlo de acessos –** Cada identidade digital tem um determinado nível de acesso aos recursos da rede, dependendo das políticas de acesso da empresa. Muitos sistemas de IAM adotam o princípio de controlo de acesso com base em funções *(role-based access control – RBAC)* para estabelecer e aplicar políticas de acesso. No RBAC, os privilégios de cada utilizador são determinados de acordo com a função ou posição que ocupam. Além disso, certos sistemas de IAM implementam também abordagens específicas para a gestão de acesso privilegiado *(privileged access management - PAM)*. O PAM envolve a administração das permissões de contas altamente privilegiadas, como administradores encarregados pelas bases de dados, sistemas ou servidores.

- **Autenticação e autorização –** Os sistemas IAM desempenham um papel importante não apenas na criação de identidades e na atribuição de permissões, mas também na aplicação efetiva dessas permissões por meio de autenticação e autorização. A autenticação é o processo pelo qual os utilizadores confirmam sua identidade quando solicitam acesso a recursos. O sistema verifica as credenciais do utilizador em relação às informações armazenadas e, se houver

correspondência, concede o acesso. No entanto, o uso exclusivo de palavras-passe para autenticação é cada vez mais considerado inseguro, uma vez que as palavras-passe são fáceis de adivinhar ou roubar através de ataques de preenchimento de credenciais (Okta, 2023).

- **Autenticação multifatorial (MFA) –** Exige que os utilizadores forneçam dois ou mais fatores de autenticação para comprovar as suas identidades. Os fatores comuns incluem um código de segurança enviado para o telefone do utilizador, uma chave de segurança física ou biometria, como verificações de impressão digital.

- **Entrada única (*Single sign-on* - SSO) -** Permite que os utilizadores acedam a várias aplicações e serviços com um único conjunto de credenciais de login. O portal SSO autêntica o utilizador e gera um certificado ou *token* que atua como uma chave de segurança para outros recursos.

- **Autenticação adaptativa –** Também conhecida como "autenticação baseada em risco", ajusta os requisitos de autenticação em tempo real, conforme o nível de risco varia. Por exemplo, quando um utilizador faz login a partir do seu dispositivo habitual, pode ser necessário apenas um nome de utilizador e palavra-passe. No entanto, se esse mesmo utilizador tentar fazer login a partir de um dispositivo não confiável ou aceder a informações sensíveis, poderá ser exigido fatores de autenticação adicionais. Isso garante que a segurança seja reforçada quando necessário, sem criar um impedimento desnecessário para os utilizadores em situações de menor risco.

- **Gestão da identidade –** Envolve o acompanhamento das ações dos utilizadores em relação ao seu acesso aos recursos. Os sistemas de IAM monitorizam os utilizadores de forma a garantir que não abusam das suas permissões e de forma a identificar potenciais intrusões de *hackers* na rede.

4.5.3 Tipos

A maioria das empresas lida com diferentes grupos de utilizadores que têm necessidades de identidade distintas: os seus colaboradores, os parceiros de negócios e os clientes. Mesmo que esteja a utilizar a mesma plataforma de IAM para todos esses grupos, será necessário abordá-los de maneira diferenciada. Existem os seguintes tipos de mecanismos de IAM: Identidade da força de trabalho; Gestão de Identidade e Acesso de Clientes e Identidade B2B (Okta, 2023).

- **Identidade da força de trabalho** –A maioria das empresas utiliza uma ampla gama de aplicações, algumas adquiridas e disponibilizadas aos funcionários, enquanto outras são desenvolvidas internamente. Independentemente do caso, é essencial adotar uma abordagem de IAM para a força de trabalho que seja capaz de integrar todas essas diferentes aplicações.

- **Gestão de Identidade e Acesso de Clientes (CIAM)** – Constitui uma vertente independente da IAM e é responsável por controlar o acesso às aplicações externas. Nesse sentido, é comum a utilização em logins de redes sociais, com ênfase na simplificação do processo (ao mesmo tempo que se mantém a segurança adequada). Uma plataforma de CIAM não se limita apenas à experiência de login do utilizador final, mas também se dedica a reunir informações sobre esses utilizadores a fim de criar perfis de identidade ricos em dados. Quanto melhor for o seu conhecimento sobre os utilizadores, maior será a sua capacidade de proporcionar valor a eles.

- **Identidade B2B *(Business-to-Business)*** – A IAM para clientes empresariais pode ser vista como um dos ramos mais especializados da gestão de identidade. Além disso, é também um dos mais desafiadores, uma vez que as grandes empresas operam em diversas pilhas tecnológicas de *backend*. Soluções de identidade personalizadas muitas vezes enfrentam dificuldades nesse contexto e lutam para se expandir quando precisam integrar vários clientes de grande porte. Nesse sentido, é comum que as empresas optem por adotar plataformas IAM de terceiros quando começam a atender um mercado mais amplo, permitindo-lhes

integrar rapidamente a identidade com qualquer provedor e em qualquer tecnologia.

4.5.4 WSO2 Identity Server

No domínio da integração e API, à medida que as empresas continuam a aumentar o número de APIs internas e de terceiros, é cada vez mais importante que as APIs sejam integradas e geridas com segurança. O WSO2 Identity Server tem como objetivo abordar tanto os domínios de API quanto de utilizador, enquanto oferece uma experiência aprimorada para o utilizador. Trata-se de um produto IAM altamente extensível, projetado para proteger APIs e microsserviços e possibilitar a Gestão de Identidade e Acesso de Clientes (CIAM). Este oferece capacidades de nível empresarial, como vinculação de identidade, SSO, autenticação forte e adaptativa, gestão de contas e identidades, de forma a auxiliar as organizações a se tornarem ágeis na integração por meio de CIAM e segurança de APIs (WSO2 IAM team, 2019). As principais capacidades do WSO2 estão compreendidas entre:

Figura 9 - Capacidades WSO2

Fonte: (WSO2 IAM team, 2019)

O WSO2 Identity Server simplifica a gestão de identidades ao permitir a integração com diversos repositórios de utilizadores, como *Java database connectivity* (JDBC), *lightweight directory access protocol* (LDAP) e *Active Directory* (AD). Além disso, oferece a capacidade de aplicar o controlo de acessos com base em funções ou atributos usando da Linguagem de *Access Control Markup Language* (XACML). Essa abordagem

é essencial, uma vez que os repositórios de utilizadores por si só não são suficientes para fornecer o SSO.

A autenticação adaptativa com o WSO2 Identity Server permite autenticar um utilizador tendo em consideração fatores como o perfil/comportamento de risco, atributos de identidade, atributos ambientais, tipo de dispositivo, geolocalização, algoritmos *machine learning* e parâmetros de pedido.

A interoperabilidade de identidade facilita a troca de atributos e decisões de autenticação entre sistemas de identidade heterogéneos e protocolos de forma transparente.

Os utilizadores finais podem gerir os seus próprios perfis e definir opções de recuperação de conta de forma autónoma. O WSO2 Identity Server suporta o provisionamento de utilizadores de entrada, saída e "just-in-time" (JIT). Estas funcionalidades ajudam eficazmente e economicamente, de forma fiável e segura as organizações a gerir as informações armazenadas em diversos sistemas e aplicações. Em relação à gestão de utilizadores e grupos, esta simplifica o processo de gestão de perfis, permitindo a associação de múltiplas contas de utilizador pertencentes a um único utilizador.

Com o WSO2 Identity Server é possível controlar o acesso a aplicações no fluxo de login, com algumas políticas de controlo de acesso detalhadas, que atuam como um ponto de decisão de políticas para aplicações de terceiros. Este também auxilia na gestão de direitos de utilizador e no controlo de acesso baseado em funções.

A segurança de APIs são expostas através de *tokens* de acesso OAuth2 e tipos de permissões associadas, incluindo APIs de controlo de acesso.

O servidor é otimizado para regulamentações de privacidade, como o *General Data Protection Regulation* (GDPR), incluindo a implementação de gestão de consentimento.

Este produto inclui ainda ferramentas avançadas de monitorização e análise para garantir o bom funcionamento do sistema IAM da empresa dentro do ambiente de produção. O sistema de análise pode gerar e avaliar tentativas de início de sessão realizadas ao servidor. Além disso, é capaz de produzir e analisar informações relacionadas com sessões específicas ocorridas, o que ajuda na deteção e prevenção de atividades fraudulentas.

4.5.5 Contribuição para o projeto

No âmbito deste projeto de dissertação de mestrado, uma das alterações mais significativas e importantes, foi a necessidade de incluir na solução *middleware* um *identity and access management (IAM)* externo. Nesse sentido, todas as operações de autenticação e autorização que antes eram da responsabilidade da solução *middleware*, tiveram de ser adaptadas de forma a permitir que fossem interligadas com a solução IAM disponibilizada pelo cliente (banco português). O banco utilizou o WSO2 Identity Server como solução IAM, e essa integração foi realizada através de serviços REST. Esta integração bem-sucedida proporcionou uma maior flexibilidade e adaptabilidade ao *middleware* no que diz respeito às soluções de IAM.

5 Organização e Gestão do Projeto

Este capítulo tem como objetivo fornecer uma visão abrangente do projeto em questão, nesse sentido é feita uma breve contextualização e descrição do mesmo, de seguida é fornecida uma visão detalhada das estruturas das equipas envolvidas e por fim são apresentadas as distintas fases do processo de desenvolvimento.

5.1 Descrição

A Innovation Makers, empresa responsável por acolher este projeto, atende a um cliente no setor bancário europeu. O cliente possui uma necessidade crítica e especifica de integrar uma nova solução de *Internet Banking* juntamente com uma plataforma de *Backoffice* para a gestão eficaz de seus clientes e operadores. Com base nas soluções oferecidas pela INM e descritas acima na secção 2.3, o objetivo principal deste projeto foi configurar e implementar com sucesso o produto de *Internet Banking*, juntamente com a plataforma de *Backoffice* e o *middleware Digital Bank Integration*. Este objetivo passa por conseguir estabelecer uma interconexão eficaz entre os canais digitais e o sistema bancário do cliente, de forma a proporcionar aos clientes do banco uma gama de operações bancárias online. A nível macro, este projeto abordou uma série de mudanças substanciais para atender às necessidades específicas do cliente. Isso incluiu modificações significativas na aparência visual das plataformas de *Internet Banking* e *Backoffice*. Foram realizadas adaptações e implementadas novas interfaces tanto para versões *web* quanto *mobile*, permitindo uma personalização completa de acordo com os requisitos do cliente. Além disso, para garantir o pleno suporte a essas plataformas, o *middleware* foi configurado e adaptado de forma abrangente. Essa configuração incluiu a modificação e/ou implementação de novas operações, alinhadas com as funcionalidades das interfaces visuais. Isso garantiu que as operações do sistema estivessem perfeitamente alinhadas com as interfaces, criando uma experiência coesa e consistente para os utilizadores finais. Além disso, vale a pena destacar que as adaptações no *middleware* não se limitaram apenas à harmonização com as interfaces visuais. Elas também foram contempladas para possibilitar comunicações eficazes com outras entidades e *stakeholders* do projeto que não estavam diretamente relacionados com o core bancário.

5.2 Equipas

A elaboração deste projeto contou com equipas multidisciplinares dedicadas a garantir o sucesso do mesmo. Estas equipas incluíam a equipa de gestão, a equipa de UI/UX, a equipa de *middleware*, a equipa de *web development* e a equipa de *quality assurance*. A equipa de gestão, composta por profissionais experientes em gestão de projetos, desempenhou um papel fundamental na coordenação e na garantia de que todos os elementos se encaixam perfeitamente de forma a responder às necessidades do cliente. A equipa de UI/UX esteve focada em criar/melhorar a experiência do utilizador, projetando novas interfaces intuitivas e atraentes para as novas funcionalidades assim como melhorar/adaptar as interfaces existentes. A equipa de *middleware*, especializada em desenvolvimento *backend*, desempenhou um papel crítico ao garantir o desenvolvimento necessário para as funcionalidades desejadas, assim como garantir a conectividade necessária para o projeto. A equipa de *web development* não apenas criou a interface *front-end*, mas também desempenhou um papel crucial no desenvolvimento *mobile*, aproveitando abordagens híbridas de forma a garantir a acessibilidade em diversas plataformas (Web, Android, iOS). Além disso, a equipa de *Quality Assurance* (QA) desempenhou um papel crítico ao assegurar a qualidade e a confiabilidade de todas as etapas do projeto, realizando testes rigorosos e identificando áreas de melhoria. A colaboração harmoniosa entre estas equipas é a espinha dorsal do nosso projeto, e cada um dos membros desempenhou um papel fundamental na realização deste projeto. É importante ainda destacar que o núcleo central deste trabalho de projeto concentrou-se na equipa de *middleware*, da qual o aluno faz parte. Essa equipa desempenhou um papel crítico ao liderar as adaptações e configurações essenciais no *middleware*, garantindo a integração eficaz das diferentes partes do sistema e a comunicação com as várias entidades e *stakeholders* envolvidos.

5.3 Equipa middleware

A equipa de *middleware*, liderada pelo CTO, Eng. Ricardo Portela, e apoiada por dois team leaders, é composta por um total de quinze programadores, incluindo o aluno deste projeto de mestrado. No contexto deste projeto, a estrutura da equipa permaneceu inalterada; no entanto, uma distinção significativa foi introduzida, uma vez que o aluno assumiu a direção técnica do projeto. Isso implicou não apenas o desenvolvimento das

adaptações necessárias à solução existente, mas também a gestão das complexas interações entre os vários *stakeholders* internos e externos.

Este trabalho de projeto também incluiu a coordenação de algumas tarefas de outros membros da equipa do *middleware*, bem como o fornecimento de apoio essencial às equipas de *web development* e *quality assurance*. É importante referir que esta atribuição de responsabilidades ao aluno na gestão técnica e liderança do projeto foi uma aposta da empresa INM de forma a possibilitar um maior crescimento do próprio.

5.4 Fases de desenvolvimento

A realização de um projeto desta dimensão requer um planeamento adequado de forma a conseguir garantir que as entregas são cumpridas dentro dos prazos estipulados. Nesse sentido, o planeamento deste trabalho de projeto contou com uma análise de requisitos de integração onde foram levantadas todas as interfaces necessárias para a interligação com um novo core bancário assim como outros tipos de requisitos funcionais e não funcionais. Posteriormente a isso a solução foi desenvolvida em várias fases e através da metodologia *Scrum* adaptada às necessidades do cliente, sendo que cada *sprint* exige a entrega de um conjunto de funcionalidades distintas. O processo de desenvolvimento e adaptação de cada uma das funcionalidades entregues em cada uma das fases foi uma jornada colaborativa que envolveu todas as equipas mencionadas antes. O seguinte cronograma exemplifica como as fases foram planeadas e entregues:

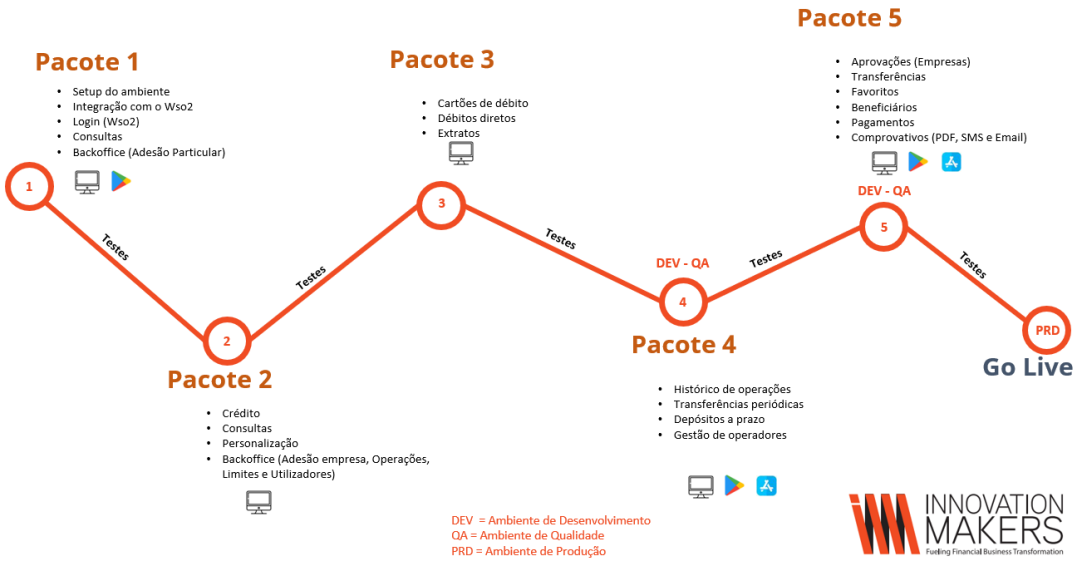

Figura 10 - Cronograma de entregas

Na Figura 10 podemos observar que tipo de funcionalidades foram entregues em cada *sprint*, e os tipos de canais/plataformas em que estes estarão disponíveis. Essas são as funcionalidades específicas dos produtos de *Backoffice* e *Internet Banking*. No entanto, antes de proceder à instanciação desses produtos, é necessário desenvolver os conectores para o core bancário e o *Identity and Access Management*, além de realizar a configuração do novo cliente e a integração dos serviços do core bancário. Podemos ainda verificar na figura acima que cada *sprint* é concluído após os testes internos realizados pela equipa de QA e pelos testes de aceitação do cliente.

6 Execução do Projeto

Este capítulo tem como objetivo apresentar o trabalho realizado pelo aluno ao longo deste projeto de dissertação de mestrado. Nesse sentido, serão detalhados todos os aspetos importantes ao longo da implementação da solução Omni canal modular da INM no novo cliente (banco português). Para melhor compreender o trabalho realizado pelo aluno neste projeto, foram introduzidas anotações na arquitetura do sistema, e um ícone foi estrategicamente colocado no modelo do diagrama de forma a indicar onde essas modificações foram implementadas.

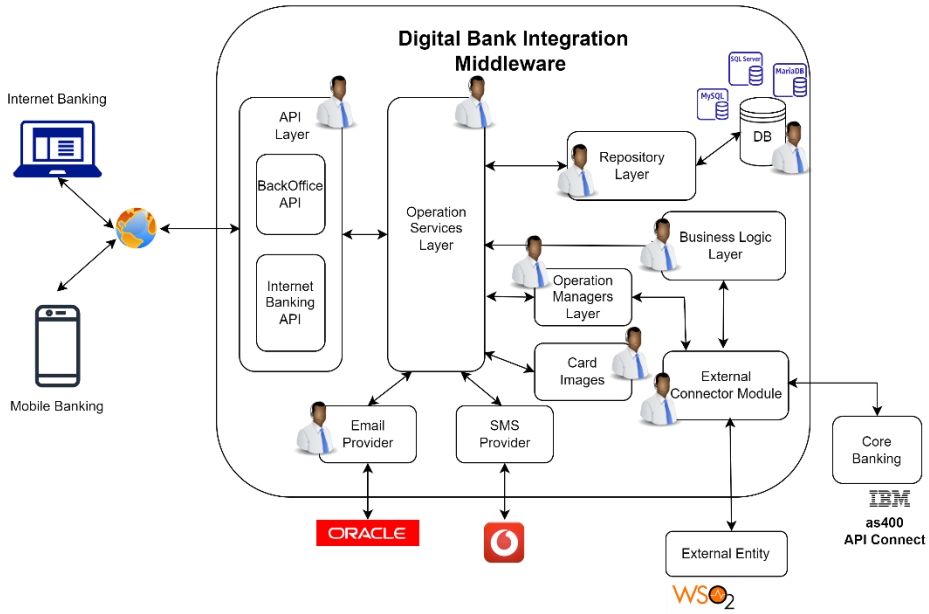

Figura 11 - Contribuições relação arquitetura

Conforme podemos visualizar na Figura 11, o ícone serve como uma representação visual das contribuições do aluno no projeto, destacando a área em que desempenhou um papel na adaptação, desenvolvimento e personalização das funcionalidades da solução. Nas camadas das APIs, o aluno desempenhou um papel central, criando os serviços REST para incorporar novas funcionalidades e adaptando as já existentes, conforme necessário. Na camada de operações, o aluno desenvolveu novas operações e ajustou as operações já existentes para atender às necessidades dos serviços REST da camada anterior. Além disso, o aluno assumiu a responsabilidade pela implementação de módulos específicos destinados a este cliente, incluindo um módulo para obter as imagens de cartões de débito

e outro para gerir o envio de e-mails. Na camada da lógica de negócios, o aluno concentrou-se na criação de implementações específicas para o core bancário deste cliente (banco português), garantindo que essas implementações atendessem às necessidades da camada de operações.

Na camada de *managers* das operações, desempenhou um papel crítico, uma vez que o aluno acrescentou a lógica necessária às operações que necessitavam de comunicação com o conector de *Identity and Access Management*. No que diz respeito à camada de repositório, foram introduzidas novas entidades no modelo de dados, juntamente com a inclusão de novos campos nas entidades já existentes, a fim de satisfazer as necessidades das operações e das novas funcionalidades. Por fim, na camada de conectores externos, o aluno criou os módulos essenciais para estabelecer conexões com o core bancário e o *Identity and Access Management*, e também implementou a maioria dos serviços disponibilizados por essas entidades. Durante essa fase, o aluno assumiu um papel estratégico na coordenação, garantindo uma distribuição meticulosa de responsabilidades em estreita colaboração com os demais membros da equipe. Essas adaptações e implementações nas várias camadas foram cruciais para atender às complexas exigências deste projeto e garantir uma integração bem-sucedida com o novo cliente.

6.1 Levantamento de Requisitos

Nesta fase de levantamento de requisitos o objetivo foi o de apresentar ao banco todas as interfaces necessárias para a integração entre a solução e o core bancário. Nesse sentido, foi disponibilizado ao cliente um documento que contém todas as interfaces de comunicação com o core bancário existentes para que o banco pudesse identificar quais os serviços a disponibilizar em API’s. Um exemplo de uma das interfaces enviadas no documento é a AccountCbInterface:

1. ACCOUNTCBINTERFACE

GETACCOUNTS(CLIENTNUMBER)

Descrição do método	Consulta da lista de contas de um determinado cliente		
Assinatura	List<CbAccount> getAccounts(String clientNumber) throws CoreBankingException;		
Retorno	Lista de CbAccounts (ver entidade no anexo "Entidades")		
Argumentos		Descrição	Restrições / Regras
Nome	Tipo		
clientNumber	String	Número de cliente	Numérico, left-padded com zeros, até atingir dimensão necessária

CREATEDOACCOUNT(CLIENTNUMBER, BRANCH, PRODUCTCODE, SUBPRODUCTCODE)

Descrição do método	Criar uma conta DO (conta à ordem)		
Assinatura	CbAccountDO createDoAccount(String clientNumber, String branch, String productCode, String subProductCode) throws CoreBankingException;		
Retorno	CbAccountDO (ver entidade no anexo "Entidades")		
Argumentos		Descrição	Restrições / Regras
Nome	Tipo		
clientNumber	String	Número de cliente	
branch	String	Balcão	
productCode	String	Código do produto	
subProductCode	String	Código do sub produto	

Figura 12 - Exemplo de Interface de integração

Através do exemplo da Figura 12, foi possível o banco identificar que é necessário que a API tenha um método que consulta a lista de contas de um determinado cliente assim como um método que cria uma conta à ordem para um determinado cliente. É ainda possível identificar quais os parâmetros de entrada necessários assim como quais os campos necessários nos objetos de retorno. A interface AccountCbInterface contém todos os métodos relacionados com contas de utilizadores, ao banco foram disponibilizadas no documento um total de vinte e três interfaces, onde cada uma está relacionada com um determinado assunto no core bancário (Por exemplo, Clientes, Cartões, Depósitos, entre outros…). Com esta informação foi possível o banco providenciar todas as API's e métodos necessários para a integração com a solução a implementar.

6.2 Conector para core bancário

O desenvolvimento de um conector para o core bancário teve início após a entrega das API's identificadas no ponto anterior e consistiu no desenvolvimento de um módulo externo que permite a integração dos serviços REST disponibilizados pelo core bancário, o objetivo deste conector é canalizar todos os serviços do core bancário. Conforme explicado na secção 3.2, este módulo consiste em um dos conectores (External Connector Module) existentes na solução e é responsável por toda a comunicação e troca de informações entre a solução e o core bancário. Os serviços foram entregues através de ficheiros ".yaml" que contêm toda a informação sobre a estrutura necessária para consumir os serviços disponibilizados. Através destes ficheiros foram criadas no conector todas as estruturas necessárias (métodos, objetos para pedidos e respostas) utilizando a

biblioteca OpenFeign (OpenFeign, 2023). Este módulo posteriormente será injetado nas dependências do cliente de forma a responder às chamadas realizadas ao core bancário.

6.3 Conector para gestão de identidade e acessos (IAM)

A solução de *middleware* possui os seus próprios mecanismos de gestão de identidade, acessos, autenticação e autorização, no entanto um dos requisitos do cliente era ter toda esta responsabilidade, uma vez que possui um produto próprio para este efeito. Nesse sentido, esta fase consistiu em criar um conector que canalizasse toda a comunicação entre as operações do *middleware* e o WSO2 (solução de *Identity and Access Management* utilizada pelo cliente), de forma a permitir toda a gestão de entidades e de acessos. A documentação dos serviços REST disponibilizados pelo WSO2, foram entregues pelo cliente de forma a fornecer toda a informação necessária para criar as estruturas necessárias no conector utilizando a biblioteca OpenFeign. Este módulo específico será injetado posteriormente única e exclusivamente nas dependências do cliente.

6.4 Configuração de novo cliente

Este passo teve início após o desenvolvimento do conector e consistiu em efetuar todas as configurações necessárias de forma a instanciar este novo cliente (banco português) na solução Omni canal modular (Digital Bank Integration) da INM. Sendo um projeto Maven foi necessário criar perfis de ambiente e as suas respetivas variáveis, com intuito de diferenciar os ambientes de desenvolvimento, qualidade e produção. Foi ainda necessário criar toda a estrutura do injetor e módulos necessários para garantir as corretas injeções de dependências a este novo cliente.

6.5 Integração com Core bancário

Este passo teve início após a configuração do novo cliente na solução de *middleware Digital Bank Integration* e consistiu na criação de implementações específicas (classes) para as interfaces existentes, conforme explicado na secção 6.1. Nesse sentido, para a interface AccountCbInterface acima exemplificada, foi criada uma classe AccountCbInterfaceClientImpl onde foram feitas as implementações dos seus respetivos

métodos. Conforme explicado na secção 3.23.2, as implementações especificas deste core bancário (cliente) encontram-se em um dos módulos de *Business Logic Layer* e são responsáveis por garantir toda a lógica necessária para chamar os serviços deste core bancário e tratar as respostas para que a informação seja retornada em objetos conhecidos pelas operações do *middleware*. Posteriormente é feita a injeção de dependência para cada interface para vincular a sua implementação através da *framework* Google Guice (Google, 2021). Este processo foi replicado para todas as interfaces existentes e necessárias para as operações de *Backoffice* e *Internet Banking*.

6.6 Instanciação do *Backoffice*

A aplicação de *Backoffice* permite gerir e controlar todos os canais, assim como colaboradores e atividades financeiras dos clientes. A instanciação do *Backoffice* consistiu em tornar disponíveis todos os serviços e operações de *middleware* necessárias para todas as funcionalidades existentes e desejadas pelo cliente. Nesse sentido, de forma a descrever o trabalho realizado, serão descritas no ponto seguinte as funcionalidades desejadas pelo cliente (banco português) e o trabalho realizado em torno das mesmas.

- **Login –** A entrada na aplicação é feita através de um ecrã de login, onde o utilizador deverá ter credenciais válidas para conseguir aceder à aplicação. Existem duas formas de autenticação: Através do protocolo *Lightweight Directory Access Protocol* (LDAP) e através de credenciais que são geridas internamente pelo *middleware* garantindo os níveis de segurança e confidencialidade. A nível de autenticação LDAP foram configuradas novas propriedades de conexão para cada ambiente, nomeadamente *endpoint*, domínio, e credenciais válidas para pesquisa de utilizadores no servidor.

As restantes funcionalidades escolhidas pelo cliente podem ser acedidas através dos seguintes menus:

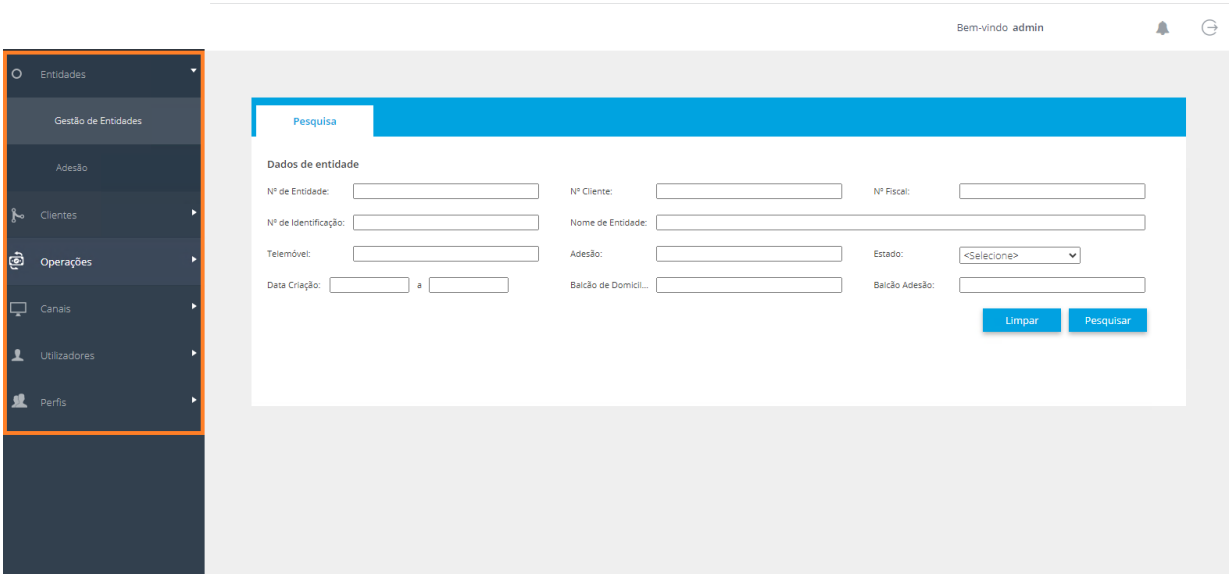

Figura 13 - Menu Backoffice

- **Entidades** – Este menu permite criar uma adesão para o *Internet Banking* de uma determinada entidade assim como gerir adesões de entidades já existentes. Uma adesão pertence a uma entidade e refere-se a uma pessoa jurídica, esta pode ter acesso a contas de outras entidades, tanto particulares como de empresas.

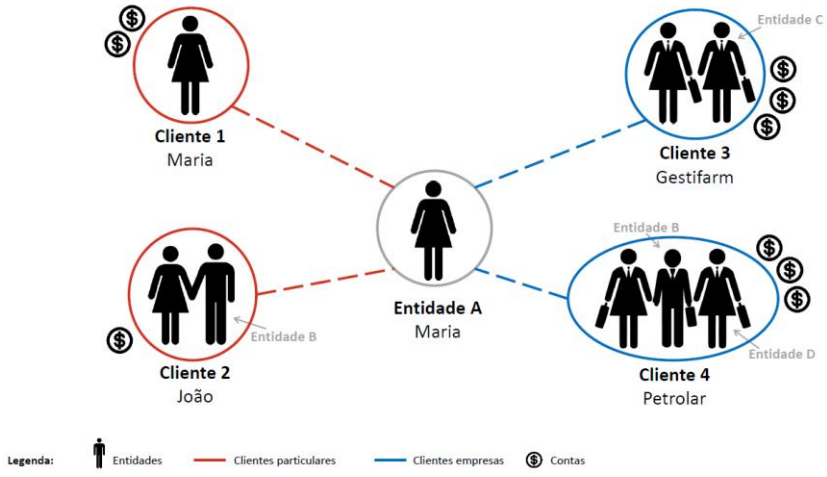

Figura 14 - Contexto entidades

Conforme podemos visualizar no exemplo ilustrado acima, a Maria pode ter acesso à conta que tem em conjunto com João (entidade particular) e às contas das empresas que está associada (entidades empresas). Estas funcionalidades a nível de *middleware* necessitaram de algumas adaptações, nomeadamente na criação e gestão de adesões, tendo em conta que os acessos conforme explicado na secção 6.3 são geridos por uma entidade externa. Nesse sentido as operações referentes à criação/gestão tiveram de ter a

injeção de um fluxo adicional de forma a comunicar com a entidade. As restantes operações de consulta de detalhes, contas e relações da entidade necessitaram das implementações especificas conforme explicado na secção 6.5.

- **Clientes** – Este menu permite associar contas de empresa a uma adesão já criada assim como consultar adesões de um determinado cliente. A nível de *middleware* foi necessário efetuar as implementações especificas conforme explicado na secção 6.5.

- **Operações** – Este menu permite pesquisar operações feitas nos canais digitais, de forma a obter uma melhor gestão das atividades dos utilizadores.

- **Canais** – Este menu permite configurar os plafonds diários e os limites máximos transacionáveis para cada tipo de operação, com base no tipo de contrato (particular ou empresas) e na moeda.

- **Utilizadores** – Este menu permite criar utilizadores para o *Backoffice,* assim como pesquisar e gerir os utilizadores existentes. A criação poderá ser feita conforme explicado no ponto acima referente ao Login, ou seja, utilizadores com credenciais internas e geridas pelo *middleware, ou* através de credenciais que serão geridas por LDAP.

- **Perfis** – Este menu permite criar, editar e eliminar perfis de utilização para o *Backoffice*.

As funcionalidades referentes aos menus de **Operações, Canais e Perfis** dependem única e exclusivamente das operações do *middleware* sem a intervenção de qualquer entidade externa. Nesse sentido e para estas funcionalidades não foram necessárias adaptações.

6.7 Instanciação do *Internet Banking*

A aplicação de *Internet Banking* é uma ferramenta, que disponibiliza todas as operações bancárias online sem a necessidade do cliente se deslocar a um balcão físico. Nesse sentido, a instanciação do *Internet Banking* consistiu em tornar disponíveis todos os serviços e operações de *middleware* necessárias para as funcionalidades existentes e

escolhidas pelo cliente. De seguida serão descritas as funcionalidades instanciadas e o trabalho realizado em torno das mesmas.

- **Login** – Esta funcionalidade e conforme explicado na secção 6.3, passou a ser da responsabilidade do cliente, nesse sentido e de forma a responder a essa necessidade foi criado um novo fluxo de autenticação e um novo serviço de *middleware* para este efeito. O novo fluxo criado possibilita que o utilizador faça o login na infraestrutura do cliente, onde este deve inserir as suas credenciais e caso estejam corretas deve introduzir um OTP *(One Time Password)* enviado para um dos seus contactos, após todas estas validações o utilizador será redirecionado para o *Internet Banking*. Sendo uma funcionalidade nova, foi criado um serviço REST no *middleware* cuja operação é responsável por garantir que após o redireccionamento do cliente para *Internet Banking* seja criada uma sessão válida para o uso das restantes operações de *middleware* assim como obter junto do WSO2 um *token* de autorização necessário para uso das operações do core bancário.

As restantes funcionalidades escolhidas pelo cliente podem ser acedidas através dos seguintes menus:

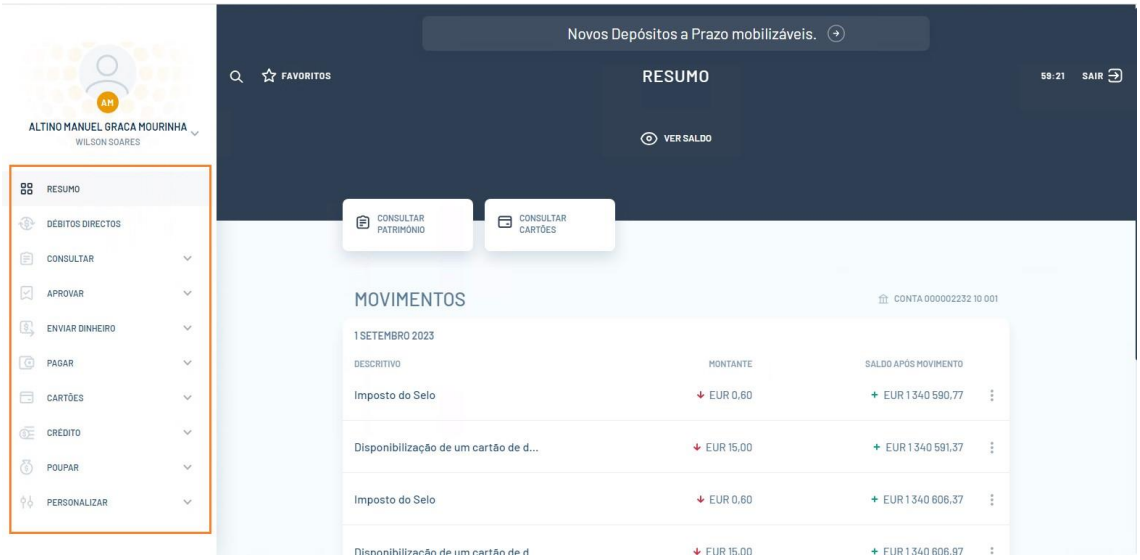

Figura 15 - Menu Internet Banking

- **Resumo** – Nesta funcionalidade é possível verificar um resumo das últimas atividades do utilizador. Esta é uma funcionalidade já existente para os restantes clientes, nesse sentido a nível de *middleware* foi necessário efetuar as

implementações especificas conforme explicado na secção 6.5 para que as operações consultassem a informação necessária.

- **Débitos Diretos** – Esta funcionalidade permite visualizar todos os débitos diretos ativos e inativos do utilizador. Esta é uma das funcionalidades novas exigidas pelo cliente e nesse sentido foi necessário desenvolver um novo serviço REST para este efeito. O novo serviço desenvolvido responde à operação que é responsável por consultar toda a informação ao módulo do core bancário e devolver a informação à interface do utilizador. No módulo do conetor com o core bancário foi também adicionado o método de invocação deste serviço assim como foi necessário criar uma interface e implementação para que fosse possível injetar a dependência na operação.

- **Consultar** – Esta funcionalidade é composta pelos seguintes submenus:
    - **Património** – Esta funcionalidade permite analisar todo o património, ou seja, é possível ter um resumo daquilo que são os ativos e passivos da adesão. Esta é uma funcionalidade já existente no *middleware* nesse sentido foi necessário efetuar as implementações especificas para este core bancário, nomeadamente, para pesquisa de contas de um determinado utilizador e para conversão de saldos de contas de outras moedas além da moeda europeia de forma a apresentar os ativos e passivos para cada moeda e convertidos para a moeda da zona euro.

    - **Saldos e movimentos** – Esta funcionalidade permite visualizar os saldos e os movimentos de uma determinada conta. Esta é uma funcionalidade já existente nesse sentido foram necessárias as implementações específicas para este core bancário, nomeadamente, pesquisa de movimentos para uma determinada conta assim como o detalhe de um determinado movimento.

    - **Detalhes da conta** – Esta funcionalidade permite verificar os detalhes de uma determinada conta da adesão, nomeadamente o nome, o tipo, moeda, NIB, IBAN e SWIFT. Esta é uma funcionalidade já existente no *middleware* nesse sentido foram necessárias as implementações

especificas para este core bancário, nomeadamente, o de pesquisa de conta do utilizador.

- **Extratos** – Esta funcionalidade permite consultar e descarregar os extratos existentes para o utilizador no core bancário. Esta é uma funcionalidade já existente no *middleware* nesse sentido foi necessário efetuar as implementações especificas para este core bancário, nomeadamente um serviço para consulta de extratos e outro para obter um determinado extrato.

- **Histórico de operações –** Esta funcionalidade permite analisar o histórico de operações efetuadas (por exemplo: Transferências, Pagamentos, Operações com cartões). Esta é uma funcionalidade já existente no *middleware* e a gestão é feita pelo *middleware* através dos dados armazenados, nesse sentido foram configuradas na base de dados os tipos de operações desejados para históricos.

Figura 16 - Menu consultar

- **Enviar dinheiro** – Esta funcionalidade é composta pelos seguintes submenus:
  - **Transferir** – Permite efetuar transferências entre contas do mesmo banco (Intra bancárias) e efetuar transferências entre contas de bancos diferentes (Interbancárias). Esta é uma funcionalidade já existente no *middleware* nesse sentido foi necessário efetuar as implementações especificas para este core bancário, nomeadamente os serviços de transferências internas e SEPA.

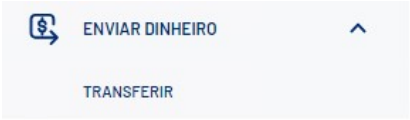

Figura 17 - Menu Transferir

- **Pagar** – Esta funcionalidade permite efetuar os pagamentos disponíveis no *Internet Banking*. Os pagamentos são geridos por outro *middleware*, denominado Sistema de pagamentos e serviços (SPS). Esta ferramenta conforme explicado na secção 2.3 é responsável por manter toda a lógica necessária para efetuar os pagamentos de uma determinada entidade, podendo esta ser uma entidade bancária ou qualquer outra entidade de pagamentos. Constituído pelas suas próprias operações e base de dados este *middleware* de pagamentos comunica com as operações do *middleware* do *Internet Banking* através de serviços REST. Nesse sentido, foi feita a instanciação do SPS para o cliente e criados todos os serviços necessários para cada tipo de pagamento requisitado pelo cliente. Posteriormente foram também criados serviços destes pagamentos no *middleware* do *Internet Banking* para que as suas operações invocassem os serviços do SPS. A funcionalidade "Pagar" é composta pelos seguintes submenus:
    - **Serviços** – Onde a adesão pode efetuar pagamentos de serviços como os de: pagamentos de serviços e compras; pagamentos ao estado e carregamentos e outros (outras entidades, por exemplo, Vodafone, MEO).

    - **Via Verde** - Onde a adesão pode ativar o serviço via verde para um determinado cartão, assim como substituir a associação de um cartão ao disposto via verde.

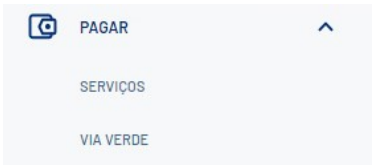

Figura 18 - Menu Pagar

- **Cartões** – Esta funcionalidade é composta pelos seguintes submenus:
    - **Consultar** – Onde é possível consultar todos os cartões disponíveis para a adesão e os seus respetivos movimentos e detalhes. Esta é uma funcionalidade já existente no *middleware* nesse sentido foi necessário

efetuar as implementações especificas para este core bancário, nomeadamente os serviços para consulta de cartões e movimentos.

- **Ativar** – Onde é possível ativar um cartão previamente requisitado. Esta funcionalidade já é existente e foi necessário efetuar as implementações especificas para este core bancário, nomeadamente os serviços de pesquisa de cartões e ativação.

- **Cancelar** – Esta funcionalidade permite cancelar um determinado cartão que esteja ativo no core bancário. Esta é uma operação existente e foi necessário efetuar a implementação para este core bancário referente ao serviço de cancelamento de um cartão.

- **Requisitar** - Permite como o próprio nome indica requisitar um determinado tipo de cartão ao core bancário. Esta é uma funcionalidade já existente nesse sentido foi necessário efetuar as implementações especificas para este core bancário, referente à requisição de cartões.

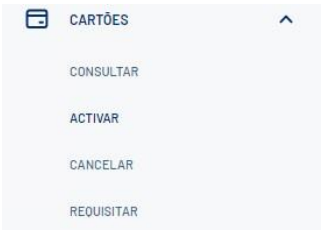

Figura 19 - Menu Cartões

- **Crédito** – Esta funcionalidade é composta pelos seguintes submenus:
  - **Consultar –** Esta funcionalidade permite consultar os créditos ativos. É possível para um determinado crédito, consultar os detalhes do mesmo, os movimentos e as suas respetivas prestações. Esta é uma funcionalidade já existente no *middleware* e foram necessárias as implementações referentes à consulta de créditos para um determinado utilizador assim com os seus respetivos planos financeiros.

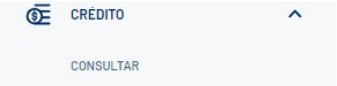

Figura 20 - Menu Crédito

- **Poupar** – Esta funcionalidade é composta pelos seguintes submenus:
    - **Depósitos a prazo –** Esta funcionalidade permite gerir os depósitos a prazo da adesão em sessão. É possível consultar os depósitos a prazo constituídos assim como mobilizar ou reforçar os mesmos. É ainda possível contrair um determinado tipo de deposito a prazo existente no core bancário. Estas são funcionalidades existentes no *middleware* e nesse sentido foram necessárias efetuar as implementações referentes à criação, pesquisa, mobilização e reforço de depósitos a prazo.

Figura 21 - Menu Poupar

- **Personalizar** – Esta funcionalidade é composta pelos seguintes submenus:
    - **Perfil** – Permite editar o perfil da adesão em questão, nomeadamente alterar a foto de perfil e os detalhes. Esta é uma funcionalidade já existente no *middleware* nesse sentido foi necessário efetuar as implementações especificas para este core bancário, nomeadamente, consulta de uma entidade e respetivos detalhes.

    - **Definições –** Esta funcionalidade permite gerir as contas visíveis, canais ativos, alterar a palavra-chave e gerir os consentimentos ativos.
        - **Contas ativas –** As contas visíveis permitem que o utilizador ative e desative uma determinada conta de forma que os restantes menus/operações apenas tenham em consideração as contas aqui definidas. Esta é uma funcionalidade que requer apenas do modelo de dados existente no *middleware*.
        - **Canais Ativos –** Esta funcionalidade oferece ao utilizador a capacidade de configurar quais canais, entre *Internet Banking* e *Mobile Banking*, estão habilitados ou desabilitados para utilização. De forma a possibilitar uma personalização de opções de acesso,

escolhendo quais canais estão ativos ou inativos de acordo com as suas preferências e necessidades. Esta funcionalidade requer apenas de configuração no modelo de dados existente no *middleware*.

- **Palavra-chave –** Esta funcionalidade permite solicitar um pedido de alteração de palavra-chave ao WSO2. Conforme definido no novo fluxo, o utilizador ao selecionar um pedido de alteração de palavra-chave será redirecionado para efetuar novo login. Após o novo login conforme explicado acima, é solicitado ao utilizador a introdução de uma nova palavra-chave e confirmação da mesma. Esta é uma funcionalidade que devido a este novo fluxo, necessitou de uma operação especifica, criada no *middleware* e no conector do WSO2 (Conector para Identity and Access Management).
- **Consentimentos ativos –** Permite consultar todos os consentimentos ativos para o utilizador. Esta é uma funcionalidade nova e necessitou de uma operação nova no *middleware* e no conector do WSO2, que é o responsável por gerir todos os consentimentos do utilizador.

- **Favoritos –** Permite consultar as operações favoritas previamente definidas. Esta é uma funcionalidade já existente no *middleware* e a gestão é feita através dos dados armazenados na base de dados.

- **Beneficiários –** Permite gerir os beneficiários para determinadas transações. Esta gestão é feita pelo *middleware* através dos dados armazenados na base de dados, no entanto é uma operação que carece de um segundo método de autenticação e para isso foi necessário implementar o mecanismo de desafios que não dependem do core bancário conforme explicado na subsecção 6.7.2.

- **Operadores –** Este submenu é destinado exclusivamente a adesões do tipo empresa e possibilita a uma determinada empresa criar várias adesões para a mesma. Esta é uma funcionalidade existente no *middleware* e a gestão é feita pelos dados armazenados na base de dados, no entanto conforme explicado na secção 6.3, a gestão das adesões passou a ser da

responsabilidade do WSO2, nesse sentido foi criada uma nova interface de forma que a implementação deste cliente conseguisse criar, editar e eliminar operadores juntos do WSO2. Sendo esta uma funcionalidade critica que requer desafio, foi ainda necessário conforme explicado na subsecção 6.7.2 OTP via criar um mecanismo que gerasse e validasse ao segundo método de autenticação lançado pela IAM *(Indentity access Management).*

Figura 22 - Menu Personalizar

- **Aprovar –** Este menu é apenas destinado a adesões do tipo empresa e possibilita à empresa em questão aprovar ou consultar operações que carecem de assinatura. Uma operação para ser executada, deve a adesão ter 100% do peso, caso contrário a mesma fica a aguardar a aprovação dos restantes assinantes até concluir os 100%. Os assinantes podem assinar uma ou mais operações, nesse sentido as mesmas podem ou não ser executadas consoante o peso. Por exemplo, uma adesão vai assinar duas operações uma já tem 50% e outra 20%, o peso atual da adesão é 50%, ou seja, a primeira operação irá ser assinada e executada com um total de 100% assinado e a segunda operação continuara pendente de assinatura pois apenas ficará com 70%. Esta funcionalidade teve de ser adaptada de forma a solicitar autorização ao core bancário para assinar ou executar um conjunto de operações. Esta funcionalidade é composta pelos seguintes submenus**:**
    - **Operações pendentes –** Permite consultar todas as operações pendentes da respetiva assinatura.
    - **Histórico de aprovações –** Esta submenu permite consultar o histórico de todas as aprovações efetuadas pela respetiva assinatura.

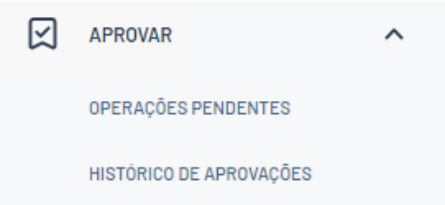

Figura 23 - Menu Aprovar

6.7.1 OTP via Core Bancário

No contexto do produto *Internet Banking*, o termo "desafio" refere-se a um processo crítico que é acionado quando uma operação requer um segundo método de autenticação para garantir a segurança da transação do cliente. A operação iniciada e que carece de desafio, fica a aguardar a inserção e validação de um OTP (*One Time Password*) que é enviado para um dos contactos definidos pelo utilizador (SMS ou E-mail). Anteriormente, a necessidade de um desafio era definida na própria operação, o que significava que a decisão de aplicar esta camada de segurança era da responsabilidade do produto *Digital Bank Integration*. No entanto, como parte deste projeto e como um dos principais requisitos do cliente, foi implementada uma mudança significativa no produto de forma que a determinação da necessidade de um desafio não é mais uma escolha estática na configuração da operação, mas sim uma resposta dinâmica do core bancário. Isso significa que o próprio sistema central do banco avalia os fatores de segurança e contextos específicos para determinar se uma operação requer ou não um desafio adicional.

O novo fluxo de desafio lançado pelo core bancário envolve três intervenientes essenciais: o core bancário, a INM e o Wso2. Primeiramente, quando um utilizador inicia uma operação na interface da INM, é feita uma chamada ao core bancário para a operação correspondente, caso essa operação seja considerada crítica e requeira uma segunda camada de autenticação, o core bancário avalia essa necessidade e envia uma resposta indicando a obrigatoriedade do desafio. Em seguida, quando a necessidade de um desafio é confirmada, a validação do OTP (*One Time Password*) é realizada em conjunto com o WSO2, que é responsável por autorizar a operação somente se o OTP fornecido for válido e corresponder às informações da transação. A autorização bem-sucedida no WSO2 concede permissão à INM para executar a operação, garantindo que apenas transações seguras sejam concluídas. De forma a responder a este requisito, foi realizado um conjunto de adaptações à solução *Digital Bank Integration* de forma que esta funcionalidade fosse adaptada às necessidades deste novo cliente (banco português) e da

mesma forma não alterasse o normal funcionamento dos restantes. A seguinte imagem ilustra a interface gráfica de quando uma operação fica a aguardar a confirmação de desafio:

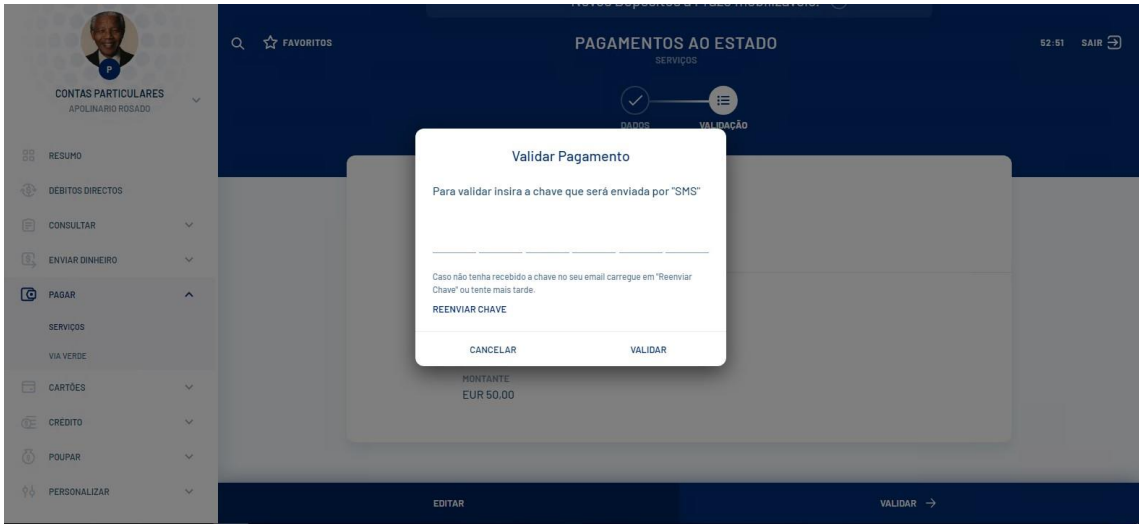

Figura 24 - Desafio lançado pelo Core bancário

Após a inserção do OTP enviado para um dos contactos do cliente, é feita a validação junto do WSO2 e é executada a operação no core bancário.

6.7.2 OTP via WSO2

Conforme explicado no ponto acima para este cliente a necessidade de desafio não é mais uma escolha estática da operação de *middleware*, mas sim uma resposta dinâmica do core bancário. No entanto, existem alguns casos em que não existe uma operação concreta do core bancário para confirmar a necessidade de desafio, como por exemplo a criação de operadores ou beneficiários. Estas operações são operações que carecem de um segundo método de autenticação, mas apenas dependem da gestão interna dos dados do *middleware.* Nesse sentido, para este novo cliente (banco português) foi ainda necessário criar um fluxo específico para estas operações de forma a conseguir gerar e validar o segundo método de autenticação. Para este novo fluxo foi necessário criar uma interface de forma a injetar para estas operações especificas uma implementação que adiciona uma chamada adicional ao WSO2 para gerar um pedido de segundo método de autenticação. Após a validação do OTP junto do WSO2 o *middleware* executa a operação e cria os respetivos dados. A seguinte figura ilustra a interface gráfica quando uma operação fica a aguardar a confirmação de desafio lançado pelo WSO2:

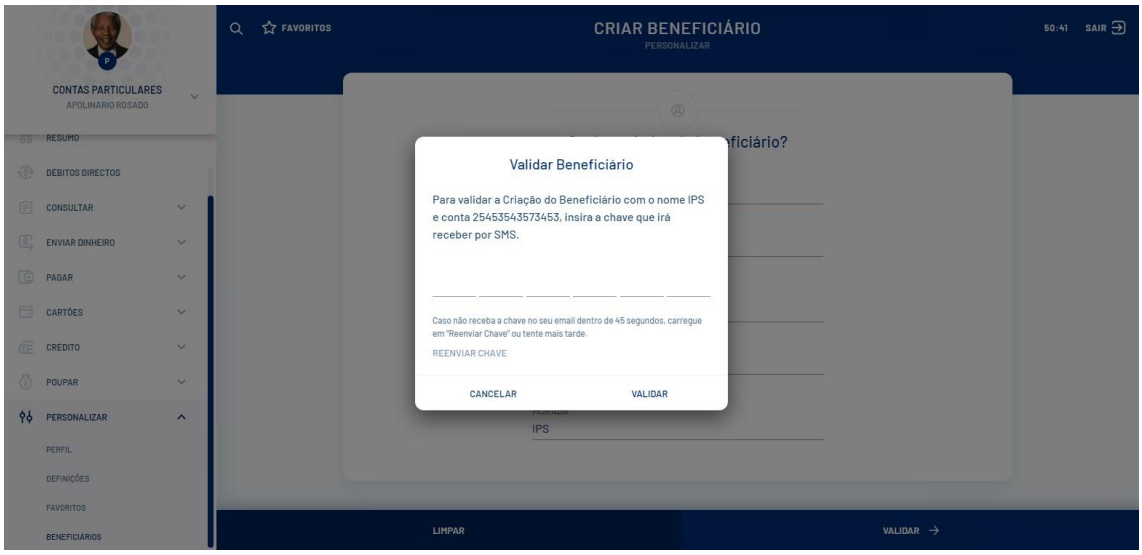

Figura 25 - Desafio lançado pelo WSO2

Semelhante ao desafio lançado pelo core bancário após a inserção do OTP enviado para um dos contactos do cliente, a validação é feita junto do WSO2 e posteriormente é executada a operação no *middleware* criando os dados necessários.

6.8 Suporte

Após o desenvolvimento ou adaptação de cada uma das funcionalidades acima descritas, foi essencial fornecer suporte às equipas internas de *web/mobile development* e *quality assurance* (QA). Nesse sentido, o aluno prestou suporte de forma a esclarecer dúvidas e questões relacionadas com os serviços desenvolvidos/adaptados o que facilitou a integração desses serviços nas interfaces visuais desenvolvidas pela equipa de *web/mobile development*. Além disso, o aluno prestou suporte à equipa de *quality assurance*, contribuindo para a identificação e resolução eficaz de erros reportados pela equipa, assegurando assim a entrega do produto com a qualidade desejada.

Durante todo o processo de desenvolvimento, foi necessário também investigar e diagnosticar as razões pelas quais algumas operações apresentavam falhas com as entidades externas. Para isso, foram recolhidas as devidas evidências das chamadas aos serviços, tanto do core bancário quanto do WSO2. Essas evidências eram essenciais para reportar ao cliente e permitir eventuais correções. Após a entrega das funcionalidades no ambiente de qualidade, o aluno também forneceu suporte adicional para identificar e resolver quaisquer erros ou problemas identificados pelo cliente de forma a assegurar o compromisso contínuo com a qualidade e satisfação do cliente face ao produto entregue.

Em resumo, o suporte prestado foi indispensável uma vez que garantiu que as funcionalidades fossem entregues com a qualidade desejada e que os problemas fossem resolvidos de forma eficaz. Essa contribuição foi essencial para o sucesso global do projeto.

6.9 Desafios encontrados

Ao longo deste projeto, foram enfrentados alguns desafios que exigiram criatividade, resiliência para superar. Estes desafios surgiram em diversas etapas do desenvolvimento do projeto, e cada um deles representou uma oportunidade para melhorar as habilidades e abordar questões mais complexas. Nesse sentido, neste ponto serão explorados os principais desafios encontrados e as estratégias adotadas para os superar.

6.9.1 Incompatibilidades da base de dados

O desafio em questão surgiu durante a entrega das funcionalidades do *Sprint* do terceiro pacote para o ambiente de qualidade (QA), conforme ilustrado na Figura 10. Este marco representou não apenas a conclusão de um conjunto significativo de funcionalidades, mas também marcou o primeiro *deploy* na infraestrutura do cliente. Foi nesse momento que nos deparamos com um desafio inesperado, exigindo uma abordagem ágil e soluções rápidas para garantir que o processo de implementação fosse bem-sucedido e atendesse aos requisitos de infraestrutura estabelecidos pelo cliente. O desafio em questão surgiu devido à diferença nos servidores de base de dados entre o ambiente de qualidade e o ambiente de desenvolvimento. Enquanto o ambiente de qualidade utilizava o MariaDB como servidor de base de dados, durante todo o processo de desenvolvimento, havíamos utilizado o SQL Server. A princípio, essa diferença não resultaria num problema, visto que o projeto utiliza o Hibernate, uma estrutura de mapeamento objeto-relacional, que permite o mapeamento dos objetos independentemente do motor de base de dados utilizado, no entanto enfrentamos dois desafios distintos que afetaram a transição entre os ambientes de desenvolvimento e qualidade. O primeiro problema surgiu durante a criação do modelo de dados, decorrente do fato de que alguns nomes de tabelas eram longos, ultrapassando o limite estabelecido pelo MariaDB, que é de 64 caracteres, enquanto no SQL Server esse limite é de 128 caracteres. Para resolver o primeiro problema, foi adotada uma abordagem dupla. Primeiro, foi atualizada a versão do

Hibernate para garantir uma melhor compatibilidade com o MariaDB. Em seguida, foi implementada uma estratégia personalizada conhecida como 'Hibernate Physical Naming Strategy', que permitiu substituir os nomes de tabelas que excediam o limite por identificadores únicos.

Contudo, após a resolução do primeiro problema, surgiu um segundo desafio. Algumas *queries* não funcionavam corretamente e, em alguns casos, não encontravam o nome de determinadas colunas. Para resolver esse problema, foi necessário realizar ajustes nas *queries* afetadas para torná-las compatíveis com a nova versão do Hibernate. Após a bem-sucedida resolução destes desafios, prosseguimos com a implementação das funcionalidades no ambiente de qualidade, marcando assim a conclusão da entrega do respetivo sprint.

6.9.2 Gestão de identidade e acessos (IAM)

Conforme explicado na secção 6.3 um dos requisitos fundamentais do cliente era assumir a total responsabilidade pela gestão de identidades e acessos. Isso representou um desafio significativo, considerando que a gestão de identidades e acessos estava anteriormente integrada na solução de *middleware*. Nesse sentido, era importante encontrar uma abordagem que permitisse a transição dessa responsabilidade para uma entidade externa. A gestão de identidades e acessos está refletida em todas as operações de *Internet Banking* e afeta algumas operações de *Backoffice*. Desta forma, a solução de *middleware* teve de se adaptar e criar mecanismos que respondessem a essa necessidade, nesse sentido foi feita uma revisão abrangente das estruturas e processos existentes, a fim de permitir uma integração segura e funcional com a solução específica do cliente, WSO2. Seguidamente serão detalhados os desafios encontrados na solução de *middleware* no que diz respeito ao contexto do *Backoffice* e *Internet Banking*.

6.9.2.1 IAM no Backoffice

No âmbito das operações de *Backoffice*, o WSO2 desempenha um papel relevante em algumas operações específicas. Essas operações permitem ao operador efetuar alterações em determinadas adesões (identidades). Entre essas operações, incluem-se a capacidade de criar uma adesão, alterar o número de telefone associado, modificar o seu estado e gerar uma nova palavra-chave.

- **Criar adesão** – Essa funcionalidade concede ao operador a capacidade de criar uma adesão específica para utilização do Internet Banking. Anteriormente, a criação de adesões era uma responsabilidade interna da solução de *middleware*. No entanto, com a integração do *Identity and Access Management*, houve a necessidade de modificar esse fluxo, permitindo que a criação de adesões também fosse realizada em conjunto com o WSO2.

- **Alterar o número de telefone** – Esta funcionalidade desempenha um papel essencial, permitindo que o operador altere o número de telefone predefinido da entidade, selecionando um dos números de telefone existentes no core bancário. Nesse sentido, a implementação desta funcionalidade exigiu uma reavaliação deste processo e modificações no *middleware* uma vez que o número de telefone é um dos contatos associados à adesão (identidade) e deve estar atualizado no WSO2, responsável pelo envio de mensagens de desafios (conforme explicado nas subsecções 6.7.1 e 6.7.2. A solução envolve a incorporação de uma camada adicional a esta funcionalidade, de modo que, sempre que houver uma alteração no número de telefone, essa informação seja sincronizada automaticamente com o WSO2. Isso garante que qualquer modificação nos números de telefone seja refletida em tempo real no sistema responsável pelo envio de mensagens, mantendo assim a consistência e a atualização dos dados.

- **Alterar o estado da adesão** – Esta funcionalidade concede ao operador a capacidade de efetuar alterações no estado da respetiva adesão. Esse estado é flexível, podendo ser ajustado para as seguintes opções: ativo, bloqueado e encerrado. Cada um desses estados implica diferentes níveis de acesso ao *Internet Banking*, permitindo ao operador conceder ou restringir o acesso conforme necessário. A possibilidade de alternar entre esses estados oferece um controlo preciso sobre o estado de cada adesão, permitindo a ativação, o bloqueio temporário ou o encerramento definitivo do acesso, conforme as exigências da gestão da identidade e dos utilizadores da solução de *middleware*. No entanto, o estado da adesão passa também a ser uma necessidade do WSO2, que é encarregue do controlo de acessos às API's do cliente (banco português). Diante dessa necessidade, a funcionalidade foi adaptada de modo a permitir a sincronização do

estado da adesão com o WSO2. Isso garante que o acesso às API's do cliente seja estritamente controlado de acordo com o estado da adesão.

- **Gerar nova palavra-chave –** Esta funcionalidade proporciona ao operador a capacidade de solicitar uma renovação da palavra-chave a uma determinada adesão. Originalmente, esta funcionalidade gerava uma nova palavra-chave internamente no *middleware* e a enviava ao cliente. No entanto, com a integração do WSO2, houve a necessidade de uma adaptação significativa e esta funcionalidade foi também ajustada. Agora o *middleware* comunica a intenção de alteração da palavra-chave, originando uma ação por parte do WSO2, que atua como o responsável e envia uma palavra-chave temporária. O utilizador recebe a palavra-chave temporária e, ao inseri-la, inicia-se o processo de renovação diretamente na infraestrutura do cliente. Esse novo processo garante uma abordagem segura, refletindo as alterações na gestão de identidades e acessos estabelecidas pelo WSO2.

6.9.2.2 IAM no Internet Banking

No contexto do *Internet Banking*, a presença e o papel do WSO2 são amplamente requisitados, especialmente considerando que a plataforma é projetada para atender a uma adesão (identidade) específica. Consequentemente, o WSO2 está envolvido em todas as etapas, desde o momento em que o utilizador faz o login até ao momento de cada pedido para executar uma operação específica onde é feita a validação da sessão. Em situações particulares, como gestão de operadores, gestão de beneficiários, aprovações de operações e as operações de desafios houve a necessidade de adaptar os fluxos para incorporar de forma eficaz a integração do WSO2.

- **Login** – Esta funcionalidade desempenha um papel fundamental ao permitir que o utilizador aceda a plataforma de *Internet Banking*. No entanto, como detalhado na secção anterior 6.7, o fluxo de login passou por uma reformulação completa. Agora, a introdução das credenciais de login ocorre diretamente nos ecrãs e na infraestrutura do cliente, representando uma mudança substancial em relação à gestão anterior, que era centralizada no *middleware.* Essa transição implicou um desafio significativo, pois exigiu uma revisão profunda das operações de login do *middleware*. Anteriormente, o *middleware* era responsável por gerar e gerir um

único *token* válido para o acesso aos seus serviços. No entanto, com a nova abordagem, foi necessário ajustar as operações do *middleware* para gerar um *token* válido para o seu próprio uso, bem como outro *token* válido junto do WSO2 para utilização dos serviços do core bancário.

- **Validação de sessão** – Esta funcionalidade interna do *middleware* desempenha um papel crítico ao validar o *token* gerado durante o processo de login sempre que um utilizador solicita a execução de uma operação no *Internet Banking*. Por outras palavras, a validade da sessão é verificada a cada pedido de operação, garantindo que a sessão do utilizador seja autenticada e autorizada corretamente. Com a introdução e integração com o WSO2, a necessidade de autorização para o uso das operações do core bancário e todo o processo de validação de sessões teve de ser adaptado. Agora, além de verificar a validade do *token* no *middleware*, é necessário validar também a autorização e a validade do *token* junto do WSO2.

- **Operadores** – A gestão de operadores consiste em gerir adesões para uma determinada entidade (adesão) do tipo empresa. Dentro do contexto do *Internet Banking*, é possível realizar diversas operações relacionadas a essas adesões, como criar, editar, eliminar e bloquear conforme explicado acima (secção 6.7). Essas operações são consideradas críticas devido à sua natureza sensível e ao potencial impacto nas entidades, como tal, todas essas operações requerem um desafio de segundo método de autenticação. A introdução do WSO2 nas operações de operadores do *middleware* trouxe consigo o desafio de conceber uma solução que abrangesse não apenas a sincronização eficaz da informação entre as plataformas, mas também a integração do envio e validação de desafios em conjunto com o WSO2. Nesse sentido agora, quando se cria, edita, elimina ou bloqueia uma adesão específica, essas informações são também refletidas no WSO2, de forma a garantir que a gestão de adesões seja consistente em ambas as plataformas, mantendo os dados de uma determinada identidade sincronizados. Além disso, foi adicionado a estas operações um mecanismo que possibilitada o pedido e validação de um segundo método de autenticação, conforme explicado na subsecção 6.7.2.

- **Beneficiários** – A gestão de beneficiários engloba a capacidade de adicionar, remover ou editar beneficiários frequentes para transações financeiras específicas no *Internet Banking*. Essas operações, devido à sua natureza sensível, requerem um segundo método de autenticação para garantir uma camada adicional de segurança. Para acomodar essa necessidade, essas operações passaram por uma revisão completa, conforme explicado detalhadamente na subsecção 6.7.2 OTP via.

- **Operações de desafios** – As operações relacionadas aos desafios referem-se a dois aspetos cruciais: a operação que gera o desafio e a operação que valida esse desafio. Nesse sentido e conforme explicado nas subsecções 6.7.1 e 6.7.2 a responsabilidade pela gestão dos desafios deixou de ser atribuída ao *middleware,* passando a ser do core bancário ou do WSO2, dependendo do contexto. A validação do desafio, por outro lado, é sempre conduzida no WSO2. Nesse sentido no contexto do *Internet Banking* juntamente com a integração do WSO2 foi necessário encontrar um mecanismo que fosse capaz de gerar desafios para operações específicas e, posteriormente, validar esses desafios junto do Wso2.

A análise dessas operações permitiu identificar os requisitos específicos, as alterações necessárias nos fluxos e as adaptações essenciais para a integração eficaz do WSO2 no *middleware*. Cada etapa desse processo foi cuidadosamente planeada e implementada para garantir que as operações continuassem a funcionar de forma eficiente, enquanto cumpriam os requisitos estabelecidos pelo cliente (banco português).

6.9.3 Core bancário

No contexto do core bancário, como detalhado nas secções 6.2 e 6.5 o desafio predominante foi a criação de um módulo externo que possibilitasse a integração com um novo core bancário através da camada de serviços REST disponibilizada pelo cliente. Esse módulo foi projetado para ser posteriormente importado na solução de *middleware*, tornando os serviços acessíveis para a camada de *business logic*, como ilustrado na Figura 1. Uma das principais distinções deste core bancário em comparação com outros clientes do *middleware,* foi o fato de que este foi o primeiro a ser integrado e disponibilizado via serviços REST. Isso representou um desafio adicional, pois exigiu uma abordagem

diferente de integração, adaptando-se às novas tecnologias e padrões introduzidos pelo cliente.

Em resumo, a conclusão deste tema destaca a complexidade das mudanças introduzidas na estrutura do *middleware*, que exigiram uma análise detalhada, adaptações em operações críticas e a integração bem-sucedida de novas entidades externas. A superação desses desafios foi fundamental para garantir os todos os requisitos necessários e exigidos no contexto do projeto.

7 Conclusões e Trabalho Futuro

Neste capítulo serão apresentadas todas as conclusões acerca deste projeto desenvolvido e exposto nesta dissertação de mestrado bem como as perspetivas de trabalho futuro.

7.1 Conclusões

Este projeto de mestrado foi uma jornada desafiadora e profundamente recompensadora que abordou a integração de uma solução de *Internet Banking* e *Backoffice* para um banco português. A motivação para essa integração surgiu da necessidade do banco superar algumas limitações significativas em relação à oferta dos seus serviços bancários online. Com apenas dois balcões físicos e uma solução de *Internet Banking* restrita a plataformas web, o banco enfrentava alguns obstáculos para atingir um público mais amplo, particularmente clientes que preferem utilizar dispositivos móveis para as suas operações bancárias. Em parceria com a Innovation Makers (INM), empresa especializada no desenvolvimento de soluções financeiras, o banco português encontrou a solução que lhe permitiu expandir o alcance dos seus serviços bancários e disponibilizá-los de forma eficaz em plataformas web e móveis. A parceria entre o banco e a INM possibilitou a integração de uma solução de *Internet Banking* mais versátil e atraente, com a capacidade de atingir um público mais vasto. Além disso, a implementação de uma plataforma de *Backoffice* ofereceu ao banco a capacidade de gerir eficientemente as suas operações e melhorar a experiência dos clientes.

A solução de *middleware* Omni-Canal da INM que suporta estes dois produtos desempenhou um papel crucial na resposta às necessidades específicas do banco português. Várias adaptações foram necessárias para garantir que o banco pudesse disponibilizar todos os seus serviços bancários de forma eficaz e personalizada. As duas principais alterações foram a gestão de identidades e acessos, que passou a ser realizada pelo próprio *Identity and Access Management* do banco, e a disponibilização dos serviços do core bancário que passou a ser feita através de serviços REST. Essas modificações foram bem-sucedidas graças à flexibilidade e capacidade modular do *middleware*, uma vez que solução Omni-Canal da INM pode ser facilmente adaptada para atender às necessidades específicas de diferentes clientes. Isso permitiu que este projeto fosse customizado de forma rápida e eficaz, alinhado com os requisitos do banco.

Ao longo do desenvolvimento deste projeto, as equipas intervenientes desempenharam papéis essenciais para garantir o seu sucesso. A colaboração multidisciplinar demonstrou o compromisso de todas as equipas envolvidas em entregar um produto que atendesse às expectativas e requisitos do banco. A capacidade de cada equipa em desempenhar o seu papel de maneira eficaz foi fundamental para o sucesso deste projeto como um todo.

Durante o desenvolvimento deste projeto, o aluno desempenhou um papel central na direção técnica do projeto, no desenvolvimento das adaptações necessárias, na definição da arquitetura das novas funcionalidades, e no suporte das funcionalidades já entregues. Além disso, o aluno atuou como ponto de comunicação entre as várias partes interessadas internas e externas. Essas responsabilidades permitiram-lhe melhorar as suas habilidades tanto a nível de desenvolvimento de software quanto nas áreas de suporte, gestão de equipas e arquitetura de sistemas.

Em resumo, este projeto de mestrado é um exemplo notável de como a colaboração entre instituições financeiras e empresas de tecnologia pode superar desafios e expandir o acesso aos serviços bancários, promovendo uma maior inclusão financeira e melhorando a comodidade para os clientes. Este representa uma contribuição significativa para o setor bancário, destacando a importância da adaptação às mudanças tecnológicas para atender às crescentes necessidades dos clientes no mundo digital. A superação de desafios técnicos, a integração e adaptação do *middleware* assim como a colaboração bem-sucedida das equipas multidisciplinares destacam a importância deste projeto na evolução do setor de serviços financeiros da empresa.

7.2 Trabalhos futuros

À medida que este projeto se desenrolava, o cliente (banco português) expressou interesse em integrar funcionalidades adicionais como perspetivas de trabalho futuro. Estas funcionalidades representarão um complemento valioso às já existentes, introduzindo novas operações e melhorando ainda mais a experiência do utilizador. Nesse sentido as seguintes funcionalidades adicionais estão atualmente em análise para futura implementação:

- **Pagamentos:**
    - **Segurança social** – Um novo tipo de pagamento que irá permitir que o utilizador faça os seus pagamentos à Segurança Social.

- **Agendamento de pagamentos** – Esta funcionalidade irá proporcionar uma maior flexibilidade aos utilizadores permitindo que estes façam o agendamento de determinados pagamentos. Com esta funcionalidade, os utilizadores têm a capacidade de programar pagamentos recorrentes, como as faturas mensais de serviços públicos, rendas ou outras despesas regulares.

- **Cartões:**
  - **Bloqueio de cartões** - A plataforma já disponibiliza uma variedade de operações relacionadas com cartões, como consultar, ativar, requisitar e cancelar. No entanto, em resposta às necessidades do cliente para funcionalidades futuras, esta funcionalidade irá oferecer aos utilizadores um controlo adicional sobre a segurança dos seus cartões, permitindo-lhes bloquear um cartão de forma conveniente e imediata quando necessário.

- **Transferências:**
  - **Editar transferências periódicas** – A plataforma atual já oferece a capacidade de agendar transferências periódicas, onde os utilizadores podem programar a execução das mesmas com base em intervalos temporais específicos. Esta funcionalidade irá dar a flexibilidade de editar as transferências agendadas, permitindo aos utilizadores alterar o período de execução previamente definido e o montante a transferir, sem a necessidade de criar uma nova instrução.

  - **Transferências Internacionais –** Atualmente, a plataforma permite efetuar transferências entre contas do mesmo banco e contas de instituições financeiras de bancos diferentes, desde que estes estejam inseridos na área SEPA (Área Única de Pagamentos em Euros). No entanto, como parte da expansão e melhorias futuras, esta funcionalidade irá proporcionar aos utilizadores possibilidade de enviar dinheiro para contas internacionais localizadas fora do espaço europeu.

- **Proxy LookUp –** Este é um requisito funcional que deve ser integrado na funcionalidade de transferências entre contas de bancos diferentes inseridas na área SEPA. Este requisito diz respeito à integração de um serviço do Banco de Portugal que irá permitir enviar dinheiro entre bancos através do contacto de telemóvel ou NIPC (Número de Identificação de Pessoa Coletiva) invés do tradicional IBAN.

- **Débitos diretos:**
  - **Alterar limites** – A solução atual, permite que os utilizadores consultem os débitos diretos ativos e inativos. No entanto, em resposta às necessidades do cliente para funcionalidades futuras, esta funcionalidade irá permitir que os utilizadores modifiquem os limites previamente estabelecidos para um débito direto específico.

  - **Desativar** – Esta funcionalidade irá possibilitar simplificar a gestão dos seus débitos diretos, uma vez que o utilizador poderá desativar facilmente um débito direto específico, conferindo-lhe maior controlo sobre as suas finanças.

- **Comunicação com o cliente –** A funcionalidade de comunicação direta com clientes é uma parte da solução *middleware* no âmbito da plataforma de BackOffice. O seu propósito é facultar aos gestores bancários a capacidade de estabelecer comunicações diretas com clientes individuais ou grupos de clientes através de notificações. Nesse sentido o banco está a considerar a integração desta funcionalidade no BackOffice implementado, como parte das funcionalidades planeadas para o futuro.

Este projeto de integração da solução Omni Canal da INM pode ser considerado uma etapa inicial na trajetória de inovação e expansão contínua das capacidades do cliente. As funcionalidades futuras irão refletir as mudanças dinâmicas no cenário dos serviços financeiros e manterão a solução relevante e competitiva no mercado.

Referências

Apache Maven. (2023, outubro 25). *Maven*. https://maven.apache.org/what-is-maven.html

Bass, L., Clements, P., & Kazman, R. (2013). *Software architecture in practice*.

Card, S. K., Mackinlay, J. D., & Shneiderman, B. (1999). *Readings in Information Visualization: Using Vision to Think* (Número 3). Oxford University Press.

Cervantes, H., & Kazman, R. (2016). *Designing Software Architectures: A Practical Approach*. www.EBooksWorld.ir

Cyber Talents. (2023). *What is cliente/server architecture*. https://cybertalents.com/blog/client-server-architecture

Datamyte. (2022). *Component Base Architecture*. https://datamyte.com/component-based-architecture/

Elliot, J., M. O'brien, T., & Fowler, R. (2008). *Harnessing Hibernate*.

Flower, M. (2003). *Patterns of Enterprise Application Architecture*.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design Patterns: Elements of Reusable Object-Oriented Software*.

Garlan, D., & Shaw, M. (1994). *An Introduction to Software Architecture*.

GitKraken. (2023). *The world's most powerful suite of Git tools*. https://www.gitkraken.com/

GitLab. (2023). *AI-powered DevSecOps Platform*. https://about.gitlab.com/platform/

Google. (2021). *Guice*.

IBM. (2023a). *SOA*. https://www.ibm.com/topics/soa

IBM. (2023b). *What is identity and access management*. https://www.ibm.com/topics/identity-access-management

informIT. (2023). *Design Challenges, Middleware Solutions, and ACE*. https://www.informit.com/articles/article.aspx?p=25486&seqNum=4

Innovation Makers. (2023, fevereiro 18). *INM*. https://www.inm.pt/. https://www.inm.pt/

ISAC, C., & DRIGĂ, I. (2015). *INTERNET BANKING SERVICES-A BUSINESS NECESSITY IN THE THIRD MILLENNIUM CLAUDIA ISAC, IMOLA DRIGĂ *.*

Jenkins. (2023). *Jenkins*. https://www.jenkins.io/doc/

Jetbrains. (2023). *IntelliJ IDEA – the Leading Java and Kotlin IDE*. https://www.jetbrains.com/idea/

JFrog. (2023). *JFROG Artifactory*. https://jfrog.com/artifactory/

Kramer, Jeff., ACM Digital Library., & ACM Special Interest Group on Software Engineering. (2010). *Proceedings of the 32nd ACMIEEE International Conference on Software Engineering - Volume 2. Volume 2.* ACM.

MariaDB Fundattion. (2023). *MariaDB Server: The open source relational database*. https://mariadb.org/

Maven: The Definitive Guide. (2009). *Maven - The definitive guide*. O'Reilly.

Microsoft. (2009). *.NET application architecture guide.* Microsoft.

Microsoft. (2023a). *SQL Server*. https://www.microsoft.com/pt-br/sql-server/sql-server-2022

Microsoft. (2023b). *What is identity and access management (IAM)*. https://www.microsoft.com/en-us/security/business/security-101/what-is-identity-access-management-iam

Microsoft. (2023c, fevereiro 25). *What is middleware*. https://azure.microsoft.com/pt-pt/resources/cloud-computing-dictionary/what-is-middleware

MySQL. (2023). *MySQL*. https://dev.mysql.com/

Okta. (2023). *What is IAM?* https://auth0.com/blog/what-is-iam/

OpenFeign. (2023). *OpenFeign*. https://github.com/OpenFeign/feign

Oracle. (2023). *RPCs*. https://docs.oracle.com/cd/E19798-01/821-1798/aeraq/index.html

Pinus, H. (2004). *Middleware: Past and Present a Comparison*.

Red Hat. (2023, fevereiro 25). *What is middleware*. https://www.redhat.com/pt-br/topics/middleware/what-is-middleware. https://www.redhat.com/pt-br/topics/middleware/what-is-middleware

Refactoring.Guru. (2023). *Design Patterns*. https://refactoring.guru/design-patterns

Richards, M. (2015). *Software architecture patterns : understanding common architecture patterns and when to use them*.

Richards, M., & Ford, N. (2020). *Fundamentals of Software Architecture*.

Rojas, D. (2023, abril 14). *All you Need to Know about Dependency Injection*. https://medium.com/javarevisited/all-you-need-to-know-about-dependency-injection-c8f896e08993

Saeed, L. (2020). Introducing Jakarta EE CDI. Em *Introducing Jakarta EE CDI*. Apress. https://doi.org/10.1007/978-1-4842-5642-8

Seemann, M., & Van Deursen, S. (2019). *Dependency Injection Principles, Practices, and Patterns*. Manning Publications.

SSMS. (2023). *SQL Server Management Studio (SSMS)*. https://learn.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver16

Stelting, S., & Maassen, Olav. (2001). *Applied Java™ Patterns*.

Swagger.io. (2023). *API Documentation Made Easy with OpenAPI & Swagger*. https://swagger.io/resources/articles/documenting-apis-with-swagger/

WSO2 IAM team. (2019, abril). *A Guide to WSO2 Identity Server*. https://wso2.com/whitepapers/a-guide-to-wso2-identity-server/