DM

Serviços de Iniciação de Pagamentos

DISSERTAÇÃO DE MESTRADO

Jacinto Duarte Afonso Costa

MESTRADO EM ENGENHARIA INFORMÁTICA

UNIVERSIDADE da MADEIRA

A Nossa Universidade

www.uma.pt

julho | 2018

Serviços de Iniciação de Pagamentos

DISSERTAÇÃO DE MESTRADO

Jacinto Duarte Afonso Costa
MESTRADO EM ENGENHARIA INFORMÁTICA

ORIENTADOR
Filipe Magno Gouveia Quintal

CO-ORIENTADOR
Pedro Miguel Santos Camacho

Abstract

The present thesis is focused on identified issues from the banking industry, namely the absence of a common interface for all banks; issues related to security; or the intermediaries' dependency for purchases.

Several bodies are significantly investing in order to resolve these challenges. Among them, one may highlight the European Commission, which proposes the use of an Application Program Interface (API) centralised for all e-banking services. This API was suggested through the Revised Payment Service Directive (PSD2).

In this thesis, the issues identified were taken into account, and an effort was made to overcome them through the use of the PSD2. The PSD2 directive was introduced by the European Commission to resolve many issues in the banking industry. We explored the PSD2 directive through a case study focused on creating an intermediary payment, and its incorporation in an e-commerce website.

An payment system called NearSoft Payment Provider (NPP) was implemented. It is easily integrable into a website through a set of widgets, which allows payments through a transactions' request under the PSD2 directive. In addition, other features such as payment management between different accounts, purchase history visualization, and payment security mechanisms were added.

During the development we had the concern of making the platform as easy as possible when it comes to its expansion, integration and maintenance. There was as well the concerb on properly documenting all the developed code.

The approach taken was tested according to its usability (8 participants) and ease of integration (2 programmers/developers), producing encouraging results. However, additional tests will be needed in order to prove the validity of the solution proposed.

Keywords

PSD2, PISP, AISP, e-commerce, Microservices, Payments

Resumo

Esta tese foca-se em problemas amplamente identificado no sector bancário, a inexistência de uma interface comum para todos os bancos, problemas relacionados com segurança ou a dependência de intermediários para realização de compras.

Diversas entidades estão a investir significativamente para a solução dos mesmos. Entre ela a comissão Europeia, que com o *Revised Payment Service* Diretive (PSD2), propôs a utilização de uma *Aplication Program Interface* (API) centralizada para todo os serviços de *e-baking*.

Nesta tese foram tidos em conta os problemas identificados, e realizado um esforço para ultrapassa-los, através da utilização da norma PSD2. A norma PSD2 foi apresentada pela Comissão Europeia com intuito de resolver diversos problemas no sector bancário. Nesta tese o PSD2 foi explorado, através de um estudo de caso focado na criação de um intermediário de pagamento e sua integração num site de e-commerce.

Foi implementado um serviço de pagamentos ao qual denominamos de *NearSoft Payment Provider*, este serviço é facilmente integrável nas aplicações de *e-commerce* através de um conjunto de *widgets*, que permitem realizar pagamentos através de um pedido de inicialização de transações, utilizando a norma PSD2. Adicionalmente foram adicionadas outras funcionalidades como gestão de pagamentos entre diferentes contas, visualização de históricos de compras e mecanismos de segurança de pagamento.

Durante o desenvolvimento tivemos a preocupação de tornar a plataforma o mais, fácil de expandir, integrar e manter possível, houve também um foco em documentar apropriadamente todo o código desenvolvido.

Ao sistema desenvolvido foi realizada uma avaliação, de acordo com a usabilidade (utilizando 8 participantes) e facilidade de integração (utilizando 2 desenvolvedores) ambos os testes produziram resultados encorajadores, contudo novos testes serão obviamente necessários para provar a validade da solução apresentada.

Palavras-chave

PSD2, PISP, AISP, Micro Serviços, Pagamentos

Agradecimentos

A finalização desta etapa só foi possível com o apoio das pessoas que me acompanharam ao longo de todo este processo.

Agradeço, por isso, a toda a minha família e namorada Sandra Santos que sempre me apoiaram e deram força e motivação para que concretizasse esta etapa tão importante. Gostava de agradecer também aos meus amigos, tanto os que já estavam presentes na minha vida e que me acompanharam durante a etapa académica, como aos que criei durante o percurso académico e que caminharam comigo nos bons e nos maus momentos.

Por último, mas não menos importante, gostaria de agradecer a todos os docentes da Universidade da Madeira, especialmente ao meu orientador Excelentíssimo Professor Doutor Engenheiro Filipe Quintal e aos colaborados da NearSoft Solutions, especialmente ao meu coorientador Engenheiro Pedro Miguel Santos Camacho.

Tabela de Conteúdos

Lista de Figuras

Lista de Tabelas

Lista de Gráficos

Lista de Acrónimos

AISP	Account Information Service Providers
AIS	Account Information Service
API	Application Programming Interface
BCE	Banco Central Europeu
DNS	Domain Name System
EBA	European Banking Authority
EEE	Espaço Económico Europeu
FDIC	Federal Deposit Insurance Corporation
HTTPS	Hyper Text Transfer Protocol Secure
MFA	Multi-Factor Authentication
JWT	JSON Web Token
NPP	NearSoft Payment Provider
OB	Open Bank
OBI	Open Banking Initiative
OBP	Open Bank Project
OBP API	Open Bank Project - API PSD2
PIN	Personal Identification Number
PIS	Payment Initiation Service
PSD	Payment Services Directive
PSD2	Revised Payment Services Directive
PSP	Payment Service Provider
PISPs	Payment Initiation Service Providers
SCA	Strong Customer Authentication
SDK	Software Development Kit
SEPA	Single Euro Payments Area
SSL	Secure Socket Layer
TANs	Transaction authentication number
TPPs	Third-Party Providers

1. Introdução

A necessidade de armazenar as riquezas com segurança é sentida já desde os primórdios da humanidade, sendo que inicialmente os templos eram usados para esse efeito. Com o aparecimento da civilização Grega, começaram a surgir os primeiros edifícios públicos onde se realizavam os depósitos e o dinheiro podia ser emprestado. Mais tarde, a civilização Romana adotou as mesmas práticas [1], resultando na formação gradual do atual sistema bancário.

O crescimento da Internet e da sua utilização em massa, na década de 90 [2], levou à criação de um ecossistema favorável ao surgimento do *e-banking*. A evolução da Tecnologia da Informação (TI), tornou-se o fator mais importante no desenvolvimento futuro dos bancos, sendo que, nos últimos anos, a adoção do *e-banking* como canal de distribuição de serviços financeiros começou a ocorrer em longa escala. Esta rápida transformação dos bancos foi possível graças às forças influentes no ambiente económico, tais como as inovações das TI e de produtos financeiros, a consolidação dos mercados financeiros, a desregulamentação de intermediação financeira, entre outros [3].

O sector bancário, tal como acontece em outras áreas, está em constante desenvolvimento. De modo a que essa evolução se adapte às novas realidades e aos utilizadores, que cada vez mais pretendem serviços simples e com mais funcionalidades, existem diversas instituições reguladoras que lançam normas e diretivas, de cumprimento obrigatório, para todos os bancos.

Na Europa, os bancos são controlados por entidades como o Banco Central Europeu (BCE) e a Autoridade Bancária Europeia (EBA), que contribuem para o trabalho regulatório da Comissão Europeia. De entre as várias funções, estas instituições são responsáveis pelo fornecimento de assessoria técnica, elaboração de normas técnicas, e publicação de diretrizes e recomendações para assegurar a aplicação consistente e efetiva das regras no Espaço Económico Europeu (EEE) [4].

1.1. Definição do Problema

O setor bancário, assim como outros setores, também enfrenta desafios. No que se refere ao *e-banking,* esses problemas podem estar relacionados com diversos fatores desde a segurança, à indisponibilidade de uma interface comum fornecida por todos os bancos, ou até mesmo a dependência de intermediários – PayPal[1], cartão de crédito, etc. –, na realização de compras online.

Os sistemas de *e-banking* estão diariamente sujeitos a vários tipos de ataques, alguns deles até muito conhecidos como o *phishing*, *pharming*, *man-in-the-browser*, cavalos de troia, entre outros [5]. Estes ataques de caracter malicioso, possuem objetivos diversos como, por exemplo, aceder aos dispositivos e/ou dados privados da conta de um utilizador, ou realizar uma transação fraudulenta [5]. Apesar de existirem procedimentos que permitam evitar estes ataques, a atenção que as entidades reguladoras

[1] PayPal - É um sistema de pagamentos de e-commerce https://www.paypal.com/pt/home.

têm sobre este assunto leva a que estas lancem normas de segurança, cuja implementação é imposta a todos os bancos.

A inexistência de uma interface comum disponibilizada pelos bancos deve-se muito ao facto da gestão de conta estar limitada ao sistema de *home banking* de cada entidade bancária. Esta condição torna-se inapropriada quando o utilizador pretende gerir diversas contas simultaneamente. Neste caso, o mesmo terá de aceder às contas separadamente, e só depois então cruzar a informação de forma manual [6]. O facto dos sistemas bancários não disponibilizarem uma API[2] comum, dificulta o surgimento de novas aplicações de gestão que se adaptem a qualquer banco e a qualquer utilizador, permitindo aumentar o número de funcionalidades disponíveis, desde a comparação de pacotes, a simulações, facilidade de transferência, etc. Abrir as APIs bancárias significaria trazer imensos benefícios para o ecossistema bancário, sendo o consumidor o principal beneficiado. O aumento da concorrência e da inovação no setor bancário e o surgimento de uma grande diversidade de serviços bancários resultaria numa maior variedade de escolha do consumidor, assim como num serviço mais bem adaptado aos clientes para fins de pagamento e de consulta das informações de conta [7].

Por fim, a dependência de terceiros na concretização de compras online, torna impossível a realização de um pagamento sem intermediários, sendo sempre forçada a utilização de um *Payment Service Providers* (PSP) como, por exemplo, o PayPal, cartão de crédito, entre outros. Estes intermediários, para além de requererem que o utilizador faça mais um registo, tenha mais uma palavra-passe, etc. [6], estão sempre associados a um aumento de sobretaxas.

1.2. Solução do Problema

De modo a solucionar os problemas descritos em 1.1., em outubro de 2015, a Comissão Europeia aprovou, em Parlamento, uma nova diretiva denominada *Revised Payments Services Directive* (PSD2).

O PSD2 é o resultado de uma revisão feita à norma lançada em 2007 – *Payments Services Directive* (PSD) –, cujos principais objetivos passavam por tornar os pagamentos internacionais mais fáceis, eficientes e seguros, abrangendo todo o tipo de pagamento eletrónico e não monetário [8]. Ao rever esta norma, a Comissão Europeia chegou ao consenso de que havia a necessidade de atualiza-la para um contexto atual, surgindo então o PSD2. Fruto desta atualização, surgem igualmente novos objetivos como tornar os serviços de pagamento online mais fáceis e seguros, proteger os consumidores, promover pagamentos online mais inovativos e móveis, e aumentar os direitos dos consumidores, a competitividade no sistema bancário e as responsabilidades do EBA [8].

O PSD2 irá, portanto, permitir aos clientes dos bancos utilizar qualquer fornecedor de serviços para gerir a informação e os pagamentos das suas contas bancárias. Para que tal seja possível, os bancos serão obrigados a disponibilizar APIs abertas a qualquer entidade que queira desenvolver um software para o sistema bancário [9]. Com esta norma, dois novos tipos de conceitos são introduzidos no

[2] API (Application Program Interface) - É um código que permite que dois programas de software se comuniquem http://searchmicroservices.techtarget.com/definition/application-program-interface-API.

ambiente financeiro – *Account Information Service Provider* (AISP) e *Payment Initiation Service Provider* (PISP).

Através do AISP, o surgimento de serviços de análise de gastos do utilizador e o cruzamento de informações de contas em vários bancos tornam-se possíveis, permitindo ao utilizador usufruir de uma visão global das suas contas bancárias, agregadas numa só plataforma.

O PISP, por sua vez, possibilita a opção de pagamento realizada através de uma transferência de crédito online em nome do utilizador. Este novo mecanismo de pagamento torna-se uma verdadeira alternativa aos pagamentos com cartão de crédito, pois oferece um serviço de pagamento fácil e acessível a todos, uma vez que o consumidor só precisa de possuir uma conta de pagamento online [10]. Este novos mecanismos de pagamentos tornam-se uma verdadeira inovação, uma vez que permitem que os fundos sejam transferidos diretamente da conta do cliente para a conta do comerciante, sem que sejam utilizadas quaisquer contas intermediárias [11].

1.3. Estágio

Este estágio foi proposto pela empresa NearSoft Solutions aos alunos do 2º Ciclo do curso de Engenharia Informática da Universidade da Madeira.

A NearSoft Solutions é uma empresa sediada na Região Autónoma da Madeira, que opera na área de Engenharia de Software. Esta empresa oferece uma diversidade de serviços, nomeadamente *Enterprise Integration*, *Containers and Microservices Architecture*, *Nearshore Software Development*, *Cloud Migrations*, entre outros.

1.4. Objectivos

O principal objetivo deste estágio, é a criação de um módulo de suporte a qualquer aplicação *e-commerce*, que funcionará como um PISP e estará de acordo com a norma PSD2, sendo, portanto, necessária a utilização de uma API que suporte esta diretiva. Este módulo de suporte permitirá ao utilizador fazer compras online beneficiando do PSD2.

Na segunda fase do projeto, é pretendida a integração do AISP no módulo de suporte. Esta integração possibilitará a realização de pagamentos “inteligentes”, tirando partido das informações das contas. Este tipo de funcionalidade caracteriza-se, portanto, pela sugestão de uma conta de pagamento, aquando da existência de várias alternativas, que tenha fundos suficientes para a realização da compra.

1.5. Estrutura do Documento

Este documento encontra-se dividido em seis capítulos.

O capítulo dois apresenta, de forma aprofundada, as questões que levaram ao lançamento da norma PSD2, pela União Europeia (UE), e o próprio conceito de PSD2. É, também, possível encontrar nesta secção, a descrição da arquitetura e das ferramentas utilizadas. O capítulo três é dedicado à

abordagem da Engenharia de Software utilizada, assim como aos requisitos do projeto e à exploração da API utilizada. Este capítulo contém, ainda, alguns diagramas de sequência que demonstram o processo de funcionamento das partes mais importantes do projeto. O capítulo quatro, por sua vez, é reservado à implementação, e aborda todos os componentes de software implementados no projeto. No capítulo cinco, encontram-se referida as avaliações realizados ao sistema e os resultados obtidos. O capítulo seis, por fim, apresenta as conclusões desta tese.

2. Contextualização

Neste capítulo, será abordado mais detalhadamente o problema descrito no capítulo 1 e a proposta de resolução do mesmo, aprovada pela Comissão Europeia – o PSD2. Por fim, será apresentada uma descrição pormenorizada dos objetivos da tese, da abordagem de arquitetura utilizada e das ferramentas utilizadas na implementação prática da mesma.

2.1. Internet Banking

O sistema de *Internet Banking,* também conhecido como *Online Banking*, *e-banking* ou *Virtual Banking*, consiste num sistema eletrónico, que permite realizar uma grande diversidade de transações através de websites ou aplicações disponibilizadas pela entidade bancária [12].

Com o surgimento do primeiro *Internet Banking*, e face à adesão massificada por parte dos clientes, foram muitos os bancos que aderiram a este sistema. Esta forte adesão foi possível por diversos fatores, sendo um deles a facilidade e comodidade que o sistema ofereceu. Com banco à distância de apenas um login, os clientes já não precisam de se deslocar ao banco, no seu espaço físico, e esperar a sua vez para poderem aceder às suas contas. Outros fatores responsáveis por esta forte adesão ao sistema, são a rapidez nas operações, que passam a ser feitas quase instantaneamente, e a disponibilidade do serviço. A dependência de apenas uma ligação à internet fez ultrapassar o horário laboral, estendendo-se a um período de 24h diárias.

A oferta diversificada de recursos tornou-se igualmente disponível online. De entre outros, destacam-se os formulários e as ferramentas de juros e de análise de investimentos, que podem ser realizadas/consultadas pelo utilizador sem que este tenha de sair de casa. Este acesso online das contas garante, ainda, uma redução das taxas.

Por fim, o facto de o *Internet Banking* ser um sistema amigo do ambiente, por reduzir a utilização de papel e dos deslocamentos [13], [14], torna-o um sistema benéfico à sua adesão.

2.1.1. Problemas

Apesar das vantagens apresentadas na secção 2.1, os sistemas de *Internet Banking* também apresentam diversas desvantagens como, por exemplo, a dificuldade de utilização por pessoas pouco dotadas de conhecimento tecnológico; a necessidade de haver grande disponibilidade dos servidores e da internet; e a oferta de apenas serviços gerais a todas as pessoas [15], [16]. Contudo, existem desvantagens mais delicadas que se relacionam com a segurança; a dependência de intermediários para a realização de compras online; e a falta de competitividade e unificação dos serviços, tendo este último na sua origem, o facto de não existir nenhuma API *standard* para sistemas de *e-banking*.

Nos subcapítulos que se seguem, os problemas levantados serão abordados de uma forma mais minuciosa, assim como a maneira como estes levaram a Comissão Europeia a apresentar a norma

Revised Payment Service Directive (PSD2), que, por sua vez, será apresentada no subcapítulo 2.1.2 [17].

2.1.1.1. Segurança

A segurança é uma questão muito frágil no que se refere a esta temática. Sem ela, os serviços de *Internet Banking* ficam incapacitados de operar. O bom funcionamento do *Internet Banking* deve ser privilegiado a todo o custo, assegurando a proteção dos dados das contas no mesmo. Para que tal aconteça, torna-se necessário reconhecer os seus principais problemas e tentar encontrar soluções que sejam adotadas pela maior parte dos sistemas, de modo a ultrapassar a barreira de segurança e permitir que os sistemas de *Internet Banking* escalem e evoluam para outro nível [18].

As principais ameaças destes sistemas são os ataques de terceiros como, por exemplo, o *phishing*, *pharming*, *man-in-the-browser*, cavalos de Troia, entre outras atividades não autorizadas [5].

No que diz respeito ao *phishing*, este tipo de ataque, geralmente realizado através do e-mail, tem como objetivo o cliente aceder a uma página web falsa, semelhante à do site do seu banco, permitindo aos hackers conseguir obter informações sensíveis, tais como *Personal Identification Number* (PIN), *passwords*, número de cartão de crédito, etc. [5], [19].

Quanto ao *pharming*, este ataque consiste em redirecionar o tráfego de um website do *Internet Banking* para outro website falso, sendo o seu objetivo muito semelhante ao do *phishing*. O *pharming* pode ser conseguido através de um ataque ao servidor *Domain Name System* (DNS[3]) da instituição bancária ou da alteração do arquivo de *hosts* do computador da vítima, sendo que ambas as formas são feitas através de um ataque malicioso [5],[19].

O *man-in-the-browser*, por sua vez, é realizado através de uma infeção do navegador web, com um cavalo de Troia[4]. Esta infeção, faz com que o navegador web fique vulnerável e permita a alteração das páginas. Este tipo de ataques, permite que sejam feitas transações fraudulentas, que têm na sua origem o computador da vítima. Estas transações são realizadas, sem que o utilizador se aperceba, durante o tempo em que este se encontra na plataforma do banco [5], [19].

Por fim, os ataques bancários através de cavalos de Troia podem ser de caracter simples – *keyloggers*[5] – ou mais sofisticados, controlados remotamente. Estes ataques são disfarçados através de Engenharia Social, querendo isto dizer que o utilizador pode ser infetado ao clicar num link que aparenta ser relevante, mas que, no entanto, não passa de um meio para a realização de um ataque. Os cavalos

[3] DNS (Domain Name System) - Atribuição de nomes aos IPs https://www.dns.pt/pt/dominios-2/o-sistema-dns/
[4] Cavalo de Troia - Porções software, geralmente enviadas por e-mail na forma anexo ou incluído num download da Web, mas que podem comprometer informações importantes https://www.tomsguide.com/us/banking-trojan-definition,news-18457.html
[5] Keyloggers - Software malicioso que regista a ordem dos cliques no teclado http://searchsecurity.techtarget.com/definition/keylogger

de Troia permitem que os hackers acedam a diversas informações bancárias importantes como é o caso das *passwords*, números de cartões de créditos, entre outros [5], [20].

Apesar de não haver uma abordagem totalmente aceite pelas instituições financeiras para proteger os utilizadores destes ataques, existem alguns procedimentos imprescindíveis que devem ser tidos em consideração. Tais procedimentos incluem a utilização de sites certificados e de confiança; a não abertura de links, no correio eletrónico, cujo remetente seja desconhecido; a utilização de um web *browser* com o protocolo de comunicação *Secure Socket Layer* (SSL[6]), para proteger os acessos ao *Internet Banking*; ou, ainda, o uso de uma autentificação através de um sistema PIN/Transaction Authentication Number (TAN). Neste último procedimento, o PIN refere-se a uma senha utilizada para o login, enquanto que os TANs representa as senhas geradas e enviadas pelo banco ao cliente, sendo cada senha da lista de TANs utilizada apenas uma única vez [21].

No que se refere a ataques específicos, existem várias contramedidas que tentam evitar a ocorrência dos mesmos. A título de exemplo, contra o *phishing* e o *pharming*, são utilizados os certificados digitais, filtros nas caixas de correios, encriptação de informações sensíveis, etc. Para proteger os sistemas e o utilizador contra cavalos de Troia, é recomendada a utilização de *Hyper Text Transfer Protocol Secure* (HTTPS) para prevenir a interceção dos dados. Outros procedimentos básicos podem ser igualmente adotados como a utilização de um bom antivírus e o cuidado extra, por parte do utilizador, ao descarregar software e/ou anexos de e-mail [22], [23].

Existem algumas regras e diretivas gerais, lançadas por entidades reguladoras, que visam o bom funcionamento dos sistemas de *Internet Banking* e a proteção dos utilizadores.

Em 2001, o ***U.S. Federal Financial Institutions Examination Council***, emitiu orientações para a utilização de *Multi-Factor Authentication* (MFA) e ordenou, ainda, que estas estivessem utilizáveis no final de 2006 [24].

Em 2012, a ***European Union Agency for Network and Information Security*** aconselhou todos os bancos a utilizarem processos de segurança onde o utilizador pudesse verificar os dados como, por exemplo, SMS TAN onde os dados da transação são enviados juntamente com o número de TAN por SMS [25].

A nível europeu existem diversas leis aplicadas a nível financeiro, que visam a proteção e o bom funcionamento dos bancos [26].

2.1.1.2. Gestão de Diferentes Contas

A gestão de contas é, também, outro problema que se levanta atualmente. Num cenário em que um utilizador possui várias contas de débito, e de crédito, em diferentes bancos, se o mesmo pretender realizar a gestão das contas corretamente, terá de realizar login em todas as contas dos diversos bancos. Todo este processo faz com que se torne difícil, para o utilizador, ter uma noção geral do

[6] SSL (Secure Socket Layer) Certificados que estabelecem uma conexão segura e confiável https://www.digicert.com/ssl/

património e fazer uma gestão adequada. Para além disso, também dificulta o desenvolvimento de aplicações financeiras por parte das empresas de desenvolvimento de tecnologia financeira, muitas vezes denominadas de *FinTech* (*Financial Technology*).

Portanto, se as informações bancárias estivessem todas agregadas, seria mais fácil para as instituições controladoras fazerem cálculos como, por exemplo, o cálculo do Imposto sobre o Rendimento (IRS) [6], [27].

A figura 1 ilustra a gestão isolada de contas no *Internet Banking* por parte de um utilizador que possui várias contas em diferentes bancos.

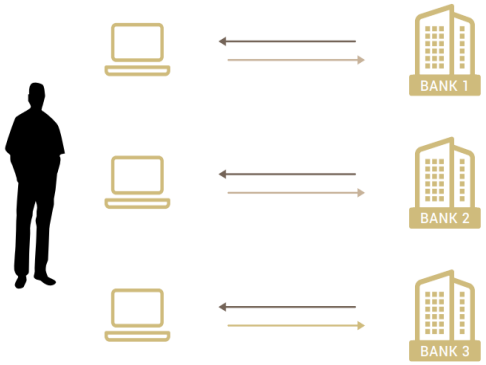

Figura 1 Gestão isolada das contas [6]

2.1.1.3. Dependência de Intermediários

A nível de pagamentos, no *Internet Banking,* o cliente é incapaz de realizar um pagamento sem intermediários, sendo sempre forçado a utilizar um *Payment Service Provider* (PSP) que gere as transações entre clientes e comerciantes. De entre os vários PSPs existentes, podemos referir, como exemplo, o PayPal e o PayMill[7]. Estes PSP oferecem serviços online de pagamentos eletrónicos, utilizando uma variedade de métodos de pagamento, desde a utilização do cartão de crédito, ao débito direto, transferência bancária, entre outros [6]. Outra função dos PSP é assegurar que as transações são feitas sem fraudes, sempre garantindo a proteção do consumidor.

A utilização de intermediários geralmente está associada a custos, pois, na maior parte das vezes, as compras estão associadas a taxas que variam entre 1,5 e 3% [28].

Na figura 2 é apresentado um exemplo da impossibilidade de o utilizador iniciar a compra sem que um intermediário intervenha na mesma.

[7] PayMill - Sistema de pagamentos de e-commerce https://www.paymill.com/en/features/

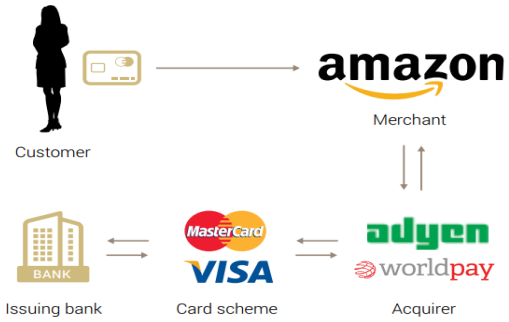

Figura 2 Entidades presentes num pagamento online [6]

2.1.1.4. Falta de Uniformização dos Serviços

Atualmente, a enorme vantagem que os bancos têm, em relação às *FinTech*, e no que diz respeito à fidelização de clientes e às oportunidades de venda cruzada, é o facto destes possuírem os dados dos clientes.

As *FinTech* têm ideias boas e imaginativas. No entanto, veem-se impossibilitadas de as porem à prova, pela falta de dados dos clientes. Permitir o acesso a terceiros aos dados dos clientes nos diversos bancos, significa não só que as *FinTech* poderiam oferecer serviços semelhantes aos dos bancos, mas mais importante, que surgiriam outras soluções de pagamento novas e inovadoras [29].

Para haver essa abertura de dados bancários dos clientes a terceiros, torna-se necessário que os bancos adiram a uma API *standard*, que permita uma forma *standard* de entrada nos bancos. A título de exemplo, na Alemanha, os bancos e outras instituições financeiras utilizam uma API aberta, gratuita e totalmente funcional. No Reino Unido, o *Open Banking Working Group* (OBWG) publicou recentemente uma *framework* para o UK *Open Banking Standard*, um ponto de partida no processo de criação de uma API bancária padronizada.

Neste momento, grande parte dos países da UE encontra-se a trabalhar no sentido de cumprir a regulamentação mencionada na diretiva PSD2, apresentada pela Autoridade Bancária Europeia (EBA) e aprovada pela Comissão Europeia [30].

Uma API *standard* implicaria que os bancos desenvolvessem uma API comum que ficasse disponível publicamente. Abrir as *APIs* bancárias a terceiros beneficiará imenso o ecossistema bancário, proporcionando, de certa forma, um aumento da concorrência e da inovação no setor bancário, levando, assim, a um maior grau de escolha do consumidor. Para além disso, fará com que surja uma grande diversidade de novos serviços bancários lvadaptados aos clientes, para fins de pagamento e de consulta de informação de conta [7]. Neste sentido, as empresas *FinTech* serão fortemente beneficiadas por haver a possibilidade de aumento do seu valor no mercado ao desenvolverem aplicações que possam ter diferentes focos na área bancária, adaptando-se aos clientes de qualquer banco [31].

Apesar destas mudanças serem do principal interesse das *FinTech*, também os bancos poderão tirar partido das mesmas, pois passam a ter a possibilidade de aumentar a quantidade de serviços oferecidos, conectando-se facilmente a outras APIs do mercado [7]. Contudo, as APIs *Open Bank,* podem ser uma ameaça para estes, na medida em que permitem que as empresas *FinTech* acedam abertamente aos dados bancários. Para ilustrar o que aqui foi dito, imaginemos uma *startup* de *FinTech* que decide utilizar uma API *Open Bank* para criar uma aplicação móvel onde os clientes orçamentam as suas finanças, administram as suas dívidas e recebem conselhos de investimentos e financeiros. Caso o cliente autorize essa aplicação móvel a aceder ao seu banco livremente, e, ao oferecerem este tipo de serviços que os bancos não oferecem, estas empresas poderão ser responsáveis pela criação de um distanciamento entre o cliente e o banco [7].

Portanto, não obstante, dos problemas mencionados anteriormente, e face às respetivas resoluções, entidades reguladoras, como a Comissão Europeia e o EBA lançaram uma nova diretiva de pagamentos – a *Revised Payment Services Directive* (PSD2) –, que irá ser abordada na secção 2.1.2.

O PSD2 não só veio resolver os problemas referidos como, de certa forma, "abrir" o sistema bancário, proporcionando diversas oportunidades que podem estar ao alcance de qualquer pessoa.

2.1.2. PSD2

De modo a resolver os problemas mencionados na secção 2.1.1, em outubro de 2015, a Comissão Europeia aprovou, em Parlamento, uma nova diretiva denominada *Revised Payments Services Directive* (PSD2).

Esta norma é uma atualização da diretiva lançada em 2007 – PSD –, abordada na próxima secção.

2.1.2.1. PSD

O *Payment Services Directive* (PSD) é uma diretiva que foi introduzida em 2007, e que se aplicava aos serviços de pagamento no mercado europeu.

O PSD fornece um quadro legislativo europeu para que os consumidores estejam protegidos, e procura tornar os pagamentos internacionais mais fáceis e eficientes, abrangendo todo o tipo de pagamento eletrónico e não monetário, tais como transferências de crédito, débitos diretos, pagamentos com cartão, e pagamentos móveis e online [32], [8]. Esta diretiva, também introduz um novo conceito de prestadores de serviços de pagamento – os *Payment Initiation Services* –, que não têm de ser necessariamente bancos.

O principal intuito do PSD foi aumentar a concorrência e a escolha dos consumidores. Segundo os seus termos, os prestadores de serviços de pagamento devem fornecer as informações adequadas aos consumidores; garantir um serviço eficiente e rápido; e compensar o consumidor, caso os serviços não sejam fornecidos corretamente [8], [33]. Estas regras também estabelecem as bases para a *Single Euro Payments Area* (SEPA), permitindo que os consumidores e empresas façam pagamentos nas mesmas condições.

2.1.2.2. Motivação

A Comissão Europeia revisa frequentemente as suas diretivas, com o intuito de verificar se as normas permanecem de acordo com a realidade atual. Ao rever o PSD, em 2012, foi verificado que esta diretiva tinha, de facto, trazido alguns benefícios ao setor bancário, dos quais se destacam o aumento da competitividade com a entrada dos *Payments Services*, a melhoria das economias de escala, pois serviu de base para a implementação da SEPA, e o aumento da transparência com informação requerida nas transações [33]. No entanto, uma diretiva deve estar sempre atualizada de acordo com o ambiente em que se encontra inserida.

Nestes últimos anos, verificou-se um aumento de necessidades dos clientes, e um grande desenvolvimento do mercado e a nível tecnológico. Para fazer face a estas mudanças, o PSD teve de sofrer uma atualização [33], [8], para o PSD2, de modo a:

- Tornar os serviços de pagamento na internet mais fáceis e seguros
- Proteger os consumidores
- Promover pagamentos na internet mais inovativos e móveis
- Aumentar os direitos dos consumidores
- Aumentar as responsabilidades do EBA para coordenar os supervisores

2.1.2.3. Evolução

O PSD2 entrou em vigor a 12 de janeiro de 2016, e teve um prazo de dois anos para a sua implementação no EEE [33]. Na figura 3, é possível visualizar o cronograma com todas as datas exatas, desde a tomada de decisão ao prazo máximo de implementação.

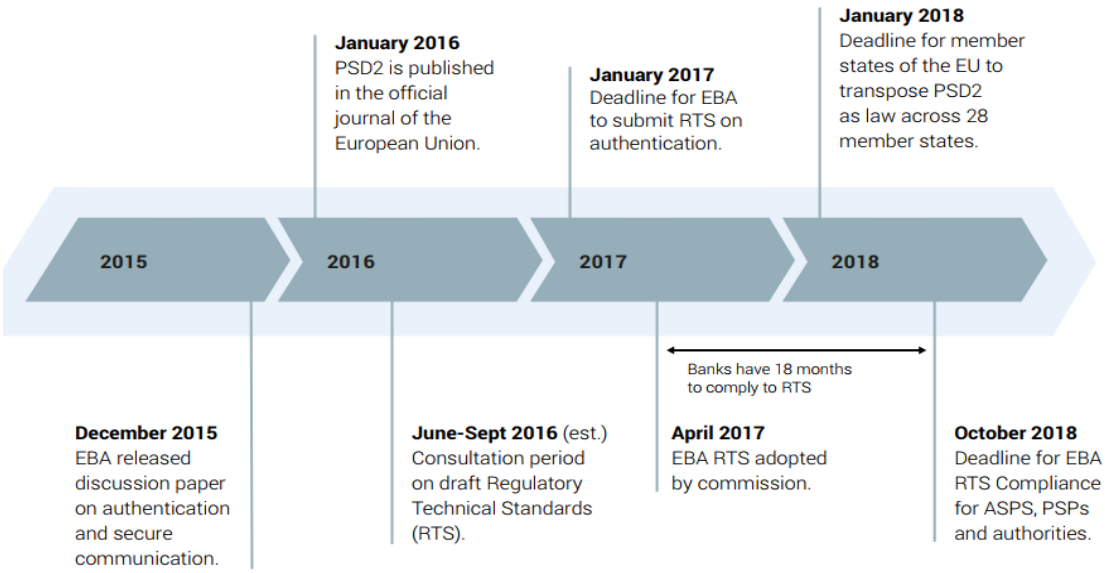

Figura 3 Cronograma PSD2 [6]

De modo a cumprir com a lei, os bancos têm até janeiro de 2018 para implementar certos aspetos do PSD2, que irão ser abordados, de forma pormenorizada, nas próximos secções. De entre vários aspetos, destacam-se a introdução de terceiros no sistema bancário; o acesso às informações da conta;

a possibilidade de pagamentos por terceiros; a transparência; e os diferentes níveis de transação em determinadas áreas [6].

2.1.2.4. Objetivos do PSD2

O PSD2 implementará um novo quadro jurídico para todos os membros do EEE. Os principais objetivos desta diretiva centram-se em [34], [6], [33]:

- Conduzir a indústria financeira europeia para um mercado de pagamentos integrado e mais eficiente
- Aumentar os requisitos de segurança a fim de proteger os consumidores contra fraudes, abusos e incidentes nos pagamentos
- Criar um ambiente para (novos) fornecedores de serviços de pagamento e aumentar a competitividade
- Promover a inovação nos espaços de pagamento e reduzir os custos das transações
- Harmonizar os preços e melhorar a segurança de processamento de pagamentos em todo EEE

De uma forma geral, a ideia base que está por detrás do PSD2 é melhorar a experiência do consumidor, e aumentar a competitividade e a inovação do mercado, ao mesmo tempo que garante a segurança do consumidor.

2.1.2.5. Aspetos Importantes do PSD2

Os subcapítulos que se seguem destinam-se à descrição detalhada dos seguintes aspetos:

- Eficiência e a integração do PSD2 no mercado;
- Direitos que os consumidores passam a ter com o PSD2;
- Novas oportunidades que surgirão juntamente com o mesmo;
- Segurança;
- Mudança que o PSD2 trará para o panorama competitivo,

2.1.2.5.1. Eficiência e Integração do Mercado

No que diz respeito à eficiência e integração do mercado, depois da implementação do PSD2, este irá passar a abranger transações *one-leg* e em todo o tipo de moeda.

As transações *one-leg,* são transações em que o consumidor, ou o PSP, encontra-se localizado fora do EEE. Para que as instituições de pagamento estejam autorizadas a oferecer serviços de pagamento, estas terão de cumprir vários requisitos, principalmente os que se relacionam com a segurança. Quanto ao consumidor, este passará a poder realizar compras online sem a necessidade de um cartão de crédito. Os PSPs, por sua vez, também terão de preencher certos requisitos, principalmente em relação ao capital e à possibilidade de realização de indeminizações pessoais. A supervisão dos serviços oferecidos pelos PSPs ficará na responsabilidade do Estado-Membro de origem, mesmo que a transação seja feita fora desse país, reforçando o seu poder e, por sua vez, facilitando a comunicação com as autoridades [33], [35].

2.1.2.5.2. Direitos dos Consumidores

Com o PSD2, os direitos dos consumidores ficarão reforçados, principalmente a nível de proteção contra fraudes e incidentes de pagamentos. Estes, estarão sob proteção do PSD2, mesmo quando o PSP estiver fora do EEE, ou a moeda seja diferente do Euro.

O PSD2 permitirá tirar partido da redução de custos de transferências e ampliará a transparência das transações. Caso seja necessário, os consumidores passam a ter direito a, pelo menos, 8 semanas para pedir reembolso de transferências por débito direto. Será ainda proibida a cobrança de taxas adicionais relacionadas com o uso de cartões e de taxas de câmbio de pagamentos abrangidas pelo regulamento de Taxas de intercâmbio e da SEPA. Os PSPs terão um prazo de 15 dias úteis, após o recebimento, para responder por escrito às queixas provenientes de clientes.

No que diz respeito a transações, cujo valor só é conhecido posteriormente, como é o caso de aluguer de viaturas, reservas em hotéis, etc., o individuo que recebe a transação só pode bloquear os fundos da conta do cliente, caso este tenha conhecimento do valor exato do serviço.

Por fim, os Estados-Membros são obrigados a recorrer a entidades competentes, como a EBA, entre outros, para lidar com disputas entre PSPs e clientes [10], [33], [36].

2.1.2.5.3. Novas Oportunidades

As novas oportunidades introduzidas com o PSD2, podem ser resumidas com os dois novos serviços que surgiram no regulamento do mesmo – o *Payment Initialization Service* (PIS) e o *Account Information Service* (AIS).

O PIS é definido no artigo 4(15) como "*Um serviço de iniciação de uma ordem de pagamento através de um pedido do cliente do serviço de pagamento em respeito a uma conta de pagamento detida por outro prestador de serviço de pagamento."* [37].

O AIS, por sua vez, é definido como "*Um serviço online para fornecer informações consolidadas de uma ou mais contas de pagamento detidas pelo utilizador do serviço de pagamento, com qualquer outro fornecedor de serviços de pagamento ou com mais de um prestador de serviços de pagamento*." [37].

Estes prestadores de serviços serão denominados de *Payment Initiation Service Providers* (PISPs) e *Account Information Service Providers* (AISPs), e muitas vezes referidos na sua generalidade como *Third Party Providers* (TPP's).

O PSD2 introduz, ainda, uma outra definição – *Account Servicing Payment Service Provider* (ASPSP) – que distingue os prestadores de serviços de pagamento que fornecem e mantêm uma conta de pagamento de um cliente, mais precisamente os bancos [33], [33].

O grande objetivo do PISP é simplificar os pagamentos online. Com a nova diretiva, é permitida a opção de transferência bancária como alternativa ao cartão de crédito. O PIS pode ser oferecido juntamente

com o AIS como meio de transferência de fundos de uma conta para outra, com base nas informações relacionadas com a transação [10], [37], [33].

Como pode ser observado na figura 4, novos prestadores de serviços podem fornecer serviços de inicialização de pagamento diretamente com o *Merchant* e a conta bancária do cliente, evitando, dessa forma, o uso de cartões de crédito e outros intermediários. A figura 4 pode, ainda, ser contrastada com a Figura 2 da secção 2.1.1.3, que ilustra a forma como os pagamentos eram realizados antes do aparecimento do PSD2 [39] .

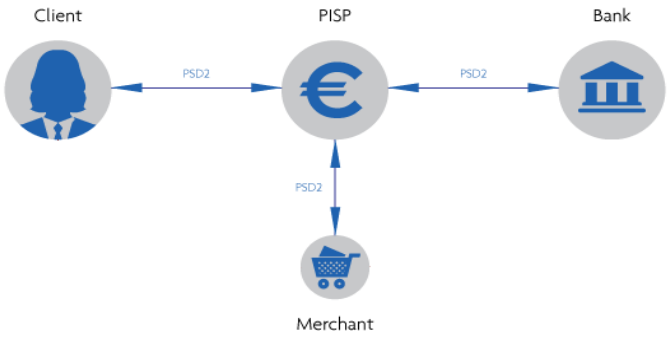

Figura 4 Esquema representativo do PISP (adaptado de [38])

O AISP tem como objetivo permitir que os consumidores e empresas obtenham uma visão consolidada das suas contas, e utilizem ferramentas de análise de transações e padrões de gastos, despesas e necessidades financeiras, de uma forma amigável [10], [37], [33].

Na figura 6, é possível observar a gestão de múltiplas contas através de uma única plataforma, um dos grandes benefícios dos AISP [6].

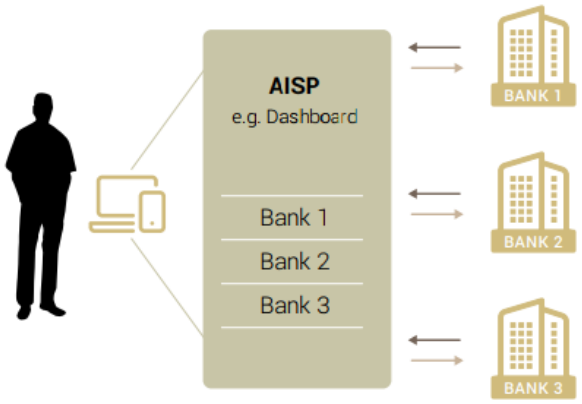

Figura 5 Esquema representativo do AISP [6]

Qualquer PSP pode fornecer o serviço de PIS e de AIS, desde que tenha a autorização do cliente.

O AIS requer mecanismos de autorização mais moderados, sendo que esta autorização pode ser dada, pelo cliente, para apenas uma única transação. No que diz respeito aos ASPSP, os PISP e os AISP podem confiar nos seus procedimentos de autenticação providenciados aos clientes. Os PISP e os AISP, por sua vez, podem prestar serviços sem dependerem diretamente de um ASPSP, desde que o cliente providencie a devida autorização. Todos os fornecedores de serviços, quer seja o PISP, AISP

ou ASPSP, devem respeitar os requisitos de segurança estabelecidos pela Comissão Europeia no jornal oficial[8] ou em outros regulamentos técnicos.

Existem ainda padrões de comunicação impostos pelo EBA para garantir uma comunicação segura entre os PISP, AISP e os ASPSP, e tornar a comunicação numa comunicação *standard,* de modo a permitir interoperabilidade tecnológica, entre outros intuitos. Caso o PISP promova um pagamento em que o cliente seja prejudicado como, por exemplo, um pagamento fraudulento, sem efeito, etc., o ASPSP deve reembolsar o cliente. Se for provado que a culpa desta ocorrência foi do PISP, este deve reembolsar o pagamento ao ASPSP. De acordo com o artigo 46(1), os ASPSP devem notificar o cliente, caso este pretenda realizar um pedido de pagamento a um PSP com um cartão de crédito que não tenha fundos suficientes para a realização do mesmo [37], [40], [33].

Até agora, era muito difícil para as TPP entrarem no mercado oferecendo serviços acessíveis em larga escala, pois eram muitas as barreiras existentes. Com o PSD2 e o surgimento dos novos fornecedores de serviços (PISP, AISP, ASPSP), essas barreiras desapareceram, sendo esperada uma maior concorrência e serviços mais baratos em todo o EEE. Contudo, as TPP deverão seguir determinadas regras, principalmente no que toca à segurança. A acessibilidade dos dados dos clientes às TPP, representa uma mudança significativa em toda a Europa em termos de concorrência [10], [37].

2.1.2.5.4. Segurança

Tendo em conta os serviços mencionados na secção 2.1.2.5.3, as empresas, ou *startups,* que pretendam prestar serviços de inicialização de pagamentos, deverão solicitar uma autorização, sendo que para o serviço de informação de conta apenas é necessário um registo. Como parte do processo de autorização e registo, as empresas, ou *startups,* necessitarão de demonstrar que implementaram sistemas efetivos de controlo de segurança, acesso a dados sensíveis e gestão de incidentes [41], [37].

Com o PSD2, todas as instituições de pagamento estão obrigadas a fornecer um documento de políticas de segurança, que inclua uma avaliação de riscos e as medidas tomadas para proteger os clientes de fraudes e uso ilegal dos seus dados pessoais. É requerido que, pelo menos uma vez por ano, os PSP entreguem aos Estado-Membros competentes uma avaliação atualizada dos riscos operacionais e de segurança, o relatório sobre a adequação das medidas de controlo implantadas e estatísticas de fraude relacionadas com os diferentes meios de pagamento. A EBA é a entidade responsável pelo estabelecimento e monitorização de orientações de medidas de segurança [33], [37].

Em caso de acidente de grande dimensão a nível de segurança ou de operacionalidade, os PSP devem comunicar ao Estado-Membro respetivo, sem demora. Caso o acidente tenha impacto no interesse financeiro do cliente, os PSP devem informar o cliente do mesmo e das respetivas medidas de resolução [37], [42].

A nível de autenticação, os Estados-Membros devem garantir, antes de cada operação bancária, que o prestador de serviços aplique uma *Strong Customer Authentication* (SCA), que consiste numa

[8] PSD2 - http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32015L2366

autenticação baseada na utilização de dois ou mais elementos de segurança. Esses elementos são categorizados como conhecimento (algo que o utilizador conhece, por exemplo, uma senha), posse (algo que apenas utilizador possui, por exemplo, um telemóvel e número) e algo específico do cliente (algo que o utilizador é ou tem, por exemplo, uma impressão digital ou padrão de íris). Uma vez que estes elementos são independentes, isto é, a violação de um não compromete os outros, é possível proteger a confidencialidade dos dados de autenticação. A SCA deve ser feita sempre que o cliente utilize um AISP, PISP ou ASPSP [43], [37]. Certas operações estão livres da SCA como, por exemplo, um pagamento que não seja superior ao valor estipulado, ou um pagamento cujo valor acumulado dos pagamentos sem SCA não exceda o valor estipulado [44], [45] . Outras operações excecionais, em que não é necessária uma SCA, podem ser vistas nos artigos 4(29) ao (31) do PSD2 [43], [37].

Os Estados-Membros devem garantir, ainda, que os ASPSP oferecem mecanismos de segurança confiáveis aos PISP e aos AISP. Caso os comerciantes ou os PSP não adotem uma autenticação forte, serão responsabilizados em caso de pagamentos não autorizados [33],[37].

2.1.2.6. Mudança no Panorama Competitivo

Uma vez implementado o PSD2, e para o bom funcionamento dos novos serviços oferecidos (PISP, AISP, ASPSP), torna-se fundamental que os bancos comuniquem através de uma API publica. A API pública do banco será não só utilizada pelas suas próprias aplicações bancárias online, como também pelas aplicações de terceiros, que utilizam as API públicas dos bancos para que tenham acesso a informações da conta e possam providenciar serviços [6].

Na figura 6, podemos observar facilmente a mudança do sistema bancário. À esquerda podemos ver o modelo tradicional bancário antes do PSD2. Neste modelo, apenas as aplicações proprietárias têm acesso a API do banco, que é fechada. No modelo bancário da direita, modelo pós PSD2, podemos visualizar que as aplicações de terceiros podem se ligar à API dos bancos [6].

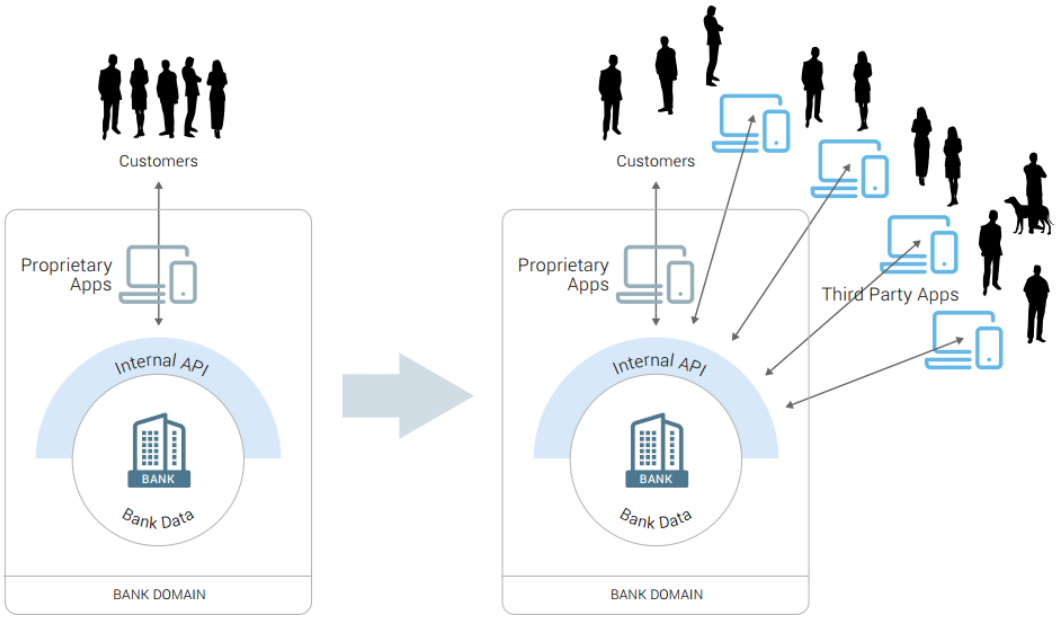

Figura 6 Panorama bancário antes e após o PSD2 [6]

O PSD2 terá um grande impacto no panorama competitivo dos bancos. Antes do PSD2, os bancos geriam e controlavam os seus próprios dados e aplicações bancárias online e móveis. Além disso, beneficiavam de relacionamentos individuais com seus clientes.

Até então, as *FinTech* e novas *Startups*, lutavam para conquistar clientes, pois não possuíam os seus dados bancários. No entanto, com o PSD2, o acesso aos dados bancários dos clientes é possibilitado, permitindo a possibilidade de estas empresas entrarem no mercado e oferecer os seus próprios serviços [6].

2.2. Objetivos Específicos do Estágio/Tese

Tendo em conta os problemas abordados em 2.1.1., e de como o PSD2 os tenta resolver, o grande objetivo desta tese é utilizar a diretiva PSD2 para criar um *Payment Services Provider* (PSP).

2.2.1. Fase I - PIPS

A primeira fase deste projeto centra-se então na criação de um módulo de suporte a aplicações de *e-commerce* e/ou outro tipo de aplicações que estejam envolvidas em compras de serviços.

Este módulo consiste na criação de um botão de compra imediata em aplicações *e-commerce*. Com esse botão, o consumidor inicia o pagamento através de um PISP, que, por sua vez, dá a instrução de pagamento ao banco do consumidor, que autorizou previamente este comerciante, não necessitando, portanto, de fazer mais nada, a não ser esperar que o produto ou serviço seja entregue. Para que este objetivo se cumpra, será utilizada uma API que suporte a norma PSD2, de forma a que o módulo seja compatível com qualquer instituição financeira, possibilitando aos fornecedores de serviço que o adquirem, efetuar todo o processo de pagamento com o mínimo de “fricção” e de forma rápida.

Após a compreensão da norma referida, deverá então ser criada uma API, cujo funcionamento deve ser testado numa simples página *web* que sirva de simulação a um site de *e-commerce*.

2.2.2. Fase 2 – AISPs (Account Information Service Providers)

Nesta fase é pretendida a sugestão “inteligente” de uma conta alternativa de pagamentos aquando do ato da compra.

Para clientes que possuam várias contas, o sistema deverá escolher/sugerir a melhor alternativa de pagamento, comportando-se, dessa forma, como um *Financial Advisor*. Esta funcionalidade, presente no PSD2, permitirá que o botão criado na fase anterior escolha a conta do cliente em que a quantia necessária para a transação esteja disponível, excluindo a hipótese da transação se realizar, caso o saldo de uma conta seja insuficiente.

2.3. Arquitetura Utilizada

No que diz respeito à arquitetura, a empresa NearSoft Solutions recomendou a utilização de uma arquitetura de micro serviços. Este tipo de arquitetura consiste na implementação de pequenos serviços, que são responsáveis por apenas uma única tarefa, permitindo a escalabilidade e diminuindo a acoplamento entre os diferentes serviços.

Na figura 7, podemos observar um exemplo de uma arquitetura de micro serviços. O cliente faz um pedido à API *Gateway* e a mesma encaminha o pedido para o micro serviço pretendido. O *Service Discovery* é responsável por listar os serviços e a sua respetiva localização, para que o cliente chegue, de certa forma, mais facilmente ao serviço em questão. O *Management*, por sua vez, é responsável por colocar os micro serviços nos nós adequados e identificar possíveis falhas [46].

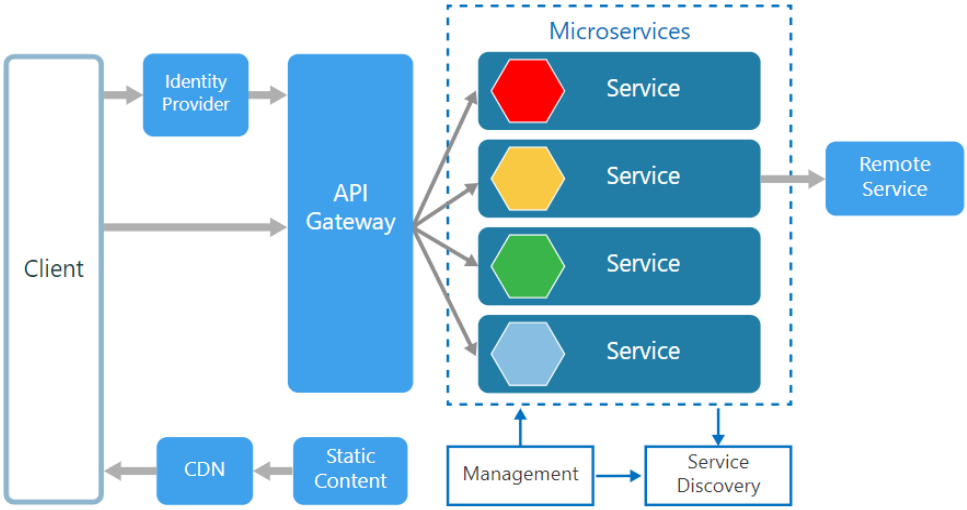

Figura 7 Arquitetura Microserviços [45]

2.4. Ferramentas Utilizadas

Para a resolução deste projeto, torna-se necessário escolher um leque de ferramentas adequadas a cada situação. As ferramentas escolhidas permitirão ultrapassar determinados problemas, como a exploração do funcionamento de uma API *Open Bank*, facilitação do desenvolvimento, realização do *deployment* do projeto de uma forma fácil e eficiente, entre outras.

Nas secções 2.4.1, 2.4.2, 2.4.3 e 2.4.4, irá ser descrito sucintamente cada ferramenta e o porquê dessa escolha.

2.4.1. Open Bank API

A escolha da *Open Bank* API foi uma das decisões mais importantes do projeto, pois é a API que permite utilizar o PSD2. É a partir dela que se torna possível o acesso aos dados dos bancos fictícios, que já possuem APIs abertas e a norma PSD2 implementada, permitindo-nos, dessa forma, simular o

ambiente PSD2 de uma forma bastante semelhante à real. De uma forma geral, esta ferramenta irá permitir a preparação do sistema, fazendo com que este se aproxime de uma aplicação real.

No que diz respeito à API *Open Bank* escolhida, foi-nos recomendado, pela NearSoft Solutions, a utilização da plataforma *Open Bank Project* (OBP) [9].

A secção 2.4.1.1 apresenta a descrição de alguns aspetos do OBP. Posteriormente, irão ser referidas outras ferramentas existentes com uma breve descrição das mesmas.

2.4.1.1. Open Bank Project

O *Open Bank Project* (OBP) consiste numa plataforma criada pela TESOBE[10], que fornece APIs bancárias abertas para terceiros para que estes possam entrar no mercado, uma das quais contempla o ambiente PSD2 e de outros ecossistemas financeiros oferecidos pela empresa.

O OBP disponibiliza conjunto de APIs *REST open source,* disponível para os bancos que querem que outras instituições financeiras aprimorem a segurança e ofertas digitais, sem a necessidade de utilização de dados reais de utilizadores.

Na figura 8 podemos visualizar como pode ser utilizado o OBP [47].

Figura 8 Diagrama de funcionamento do OBP API [46]

O OBP oferece a possibilidade de serem desenvolvidas aplicações financeiras num ambiente fictício, muito semelhante a um ambiente real. A utilização do OBP é muito vantajosa devido [48]:

- Integração das informações das contas bancarias via API *REST*
- Uso de um sistema de autenticação seguro através da implementação do *OAuth*
- Disponibilidade em qualquer sítio
- Interface bancária consistente

[9] Open Bank Project, open source open banking API https://openbankproject.com/
[10] TESOBE - Empresa com sede em Berlim que esta por trás do Open Bank Project http://tesobe.com/

Para o desenvolvimento de uma aplicação financeira, o fornecedor de serviços deve pedir uma chave da API para aceder à mesma livremente as contas bancárias dos utilizadores, caso estes o autorizem [49].

Entre diversas *sandboxes*[11], este projeto oferece uma *sandbox* para o PSD2, que demonstra a solução PSD2 API, com diversos pedidos que permitem tirar partido da nova norma europeia de pagamentos. Ao longo da documentação da API, é possível fazer testes aos pedidos e tirar partido das funcionalidades [50].

Ainda, é disponibilizada uma UTILITY APP em que o utilizador pode aceder a alguns pedidos da API, de uma forma gráfica, tirando partido da simplicidade e menor esforço cognitivo [51].

Esta ferramenta foi uma das mais importantes do projeto, uma vez que permitiu testar a norma PSD2, utilizando os pedidos destinados para tal.

2.4.1.2. Outras Ferramentas *Open Bank*

De entre muitas outras ferramentas *Open Bank*, podemos destacar as APIs disponibilizadas pela Nordia[12], banco e grupo financeiro que atua no norte da Europa, e pela Apigee[13], empresa de gestão de APIs pertencente à Google.

No que diz respeito à API lançada pela Nordia – *Nordia's Open Banking Initiative* (OBI) API –, esta é também uma *sandbox* que tem como principal intuito familiarizar os desenvolvedores com o OBI, antes da sua versão oficial. Esta API também oferece funcionalidades PSD2, tais como os serviços AIS, PIS, mecanismos de segurança *OAuth,* entre outros [52].

A API desenvolvida pela Apigee consiste também numa versão *sandbox*, que está de acordo com o PSD2, e que oferece igualmente os serviços PIS e AIS, e utiliza mecanismos de autenticação *OAuth*.

Todas estas *Open Bank* APIs são boas e completas. No entanto, e na nossa opinião, não dispõem de amadurecimento, organização e leque de funcionalidades que o OBP comtempla [53].

2.4.2. Front-End

Para todo o *Front-End* deste projeto irão ser utilizadas as linguagens HTML, CSS e JavaScript (JQuery). Os tópicos que se seguem encontram-se reservados à apresentação detalhada de cada uma destas linguagens.

[11] Sandbox - Ambiente de testes com características idênticas a um ambiente de produção https://smartbear.com/learn/api-testing/what-is-an-api-sandbox/
[12] Nordea - Banco e grupo financeiro que atua no norte da europa https://www.nordea.com/en/
[13] Apigee - Empresa líder na área das APIs https://apigee.com/about/apigee

2.4.2.1. HTML

O *HyperText Markup Language* (HTML) é uma linguagem de programação que permite a criação de *websites*, e caracteriza-se como acessível, fácil de aprender, com muitas funcionalidades e em constante evolução.

O "*HyperText*" refere-se aos links que conectam páginas entre si, dentro ou fora de um domínio. O "*markup*" (ex: <img>, <p>, , etc), permite adicionar texto, imagens e outros conteúdos às páginas web. O código HTML geralmente está entre parênteses angulares <> para diferenciar o código de outro texto que seja escrito pelo desenvolvedor para dar conteúdo à página [54], [55].

O HTML consiste num conjunto de códigos, geralmente denominados por "*tags*", que são escritos num ficheiro de texto. Esse ficheiro de texto é depois guardado com a extensão .html e pode ser interpretado por um *browser* como por exemplo o Internet Explorer ou o google Chrome. Para uma boa estruturação de uma página *web,* as *tags* HTML devem ser utilizadas corretamente encadeadas, sendo provavelmente a tarefa mais complexa [55]. Esta linguagem de programação permite estruturar todas as interfaces *web* que irão ser necessárias à realização deste projeto.

2.4.2.2. CSS

O *Cascading Style Sheets* (CSS) é uma linguagem que define a aparência da linguagem de marcação como por exemplo HTML, XHTML, etc. O CSS permite configurar a maneira como as marcações , ou seja, os elementos contidos num documento de código, são exibidas [56], [57].

Com a evolução dos recursos de programação, as tecnologias passaram a adotar estilos cada vez mais atrativos para os utilizadores. Com isto, as linguagens de marcação simples, como o HTML, também passaram a ser mais aprimoradas, nascendo daí o CSS. O CSS permite que as marcações num documento possam apresentar estilos diferentes como a cor, o tipo de letra, a posição, entre outros. Esta linguagem permite também uma grande flexibilidade no estilo das marcações, uma vez que cada desenvolvedor pode personalizar o estilo das mesmas [57].

Neste projeto, o CSS terá como objetivo personalizar as páginas HTML criadas.

2.4.2.3. JQuery

O JQuery é uma biblioteca *open source* de JavaScrip, que tem como principais características a rapidez e o grande leque de funcionalidades. Esta biblioteca torna bastante simples a manipulação de documentos HTML, como tratar eventos, construir animações, e processar pedidos Ajax, que irão ser amplamente utilizados aos longo do projeto, permitindo assim aceder facilmente a APIs terceiras. Uma outra vantagem é o seu funcionamento na maior parte dos navegadores [58]. Para a utilização do JQuery, é fundamental importar a biblioteca para a página HTML. Esta importação é feita localmente, ou através de um link para recurso web [59].

No âmbito deste projeto, o JQuery será utilizado principalmente para comunicar com o servidor, sem bloquear o cliente, utilizando os pedidos Ajax. Outras funcionalidades, como embutir o código HTML dinamicamente sem fazer *reload* a página, também serão utilizadas.

2.4.3. Back-end

Para o *Back-End*, foi decidido utilizar o Flask, uma *microframework* de Python baseada em Werkzeug e Jinja2. Na secção que se segue, podemos encontrar uma apresentação mais detalhada desta ferramenta. Esta ferramenta foi escolhido pela simplicidade que esta oferece na implementação de API, para além de que permite a fácil utilização da arquitetura *REST* [60].

Para a persistência dos dados, optamos pelo MongoDB, devido à sua flexibilidade, conseguida através do armazenamento dos dados em formato semelhante ao JSON, e à sua disponibilidade. Nos subpontos seguintes será descrito com mais algum detalhe o conceito MongoDB [61].

2.4.3.1. Python Flask's

O Python é uma linguagem de programação de alto nível, interpretada e de propósito geral. Um dos principais focos desta linguagem consiste na legibilidade do código. A sintaxe, no Python, ajuda os programadores a codificar em menos etapas quando comparado com outras linguagens. O Python é amplamente utilizado em organizações, devido aos seus múltiplos paradigmas de programação, que geralmente envolvem programação imperativa e orientada a objetos. Possui uma ampla biblioteca padrão, que faz a gestão automática da memória e dos recursos. O Python oferece ainda uma diversidade de *frameworks*, que se ajustam facilmente aos requisitos do programador [62].

A *microframework* Flask, por sua vez, foi projetada para ser fácil de utilizar e estender. A ideia por detrás do Flask consiste na construção de uma base sólida para aplicações *web* de diferentes complexidades. A partir daí, é possível conectar todas as extensões necessárias. Além disso, podem ser criados e adicionados módulos próprios gratuitamente. A suas principais características são o suporte, *secure cookies, RESTful request dispatching* ,desenvolvimento de servidores, entre outras [60], [63].

Um dos intuitos da utilização desta *microframework* no projeto é a sua capacidade de manipulação com os pedidos HTTP, e a facilidade oferecida na implementação de APIs *REST*.

2.4.3.2. MongoDB

O MongoDB é uma tecnologia de base de dados *open source* NoSQL, que não utiliza linhas nem tabelas como bases de dados relacionais. É composto por uma arquitetura própria, através do uso de documentos e coleções.

Como mencionado em 2.4.3, o MongoDB é bastante flexível, e o seu formato é semelhantes ao formato JSON, possibilitando, desta forma, a variação dos campos de documento para documento e a própria estrutura que também pode ser alterada ao longo do tempo. Os modelos para os objetos podem ser definidos diretamente no código da aplicação, tornando bastante simples a sua utilização. Para além

dos benefícios apresentados esta ferramenta é uma base de dados distribuída, assegurando, dessa forma, uma alta disponibilidade e capacidade de escalar [64], [61].

Devido à sua flexibilidade e simplicidade na utilização, foi decidido usar o MongoDB para dar persistência os dados.

2.4.4. *Deployment* e documentação do Sistema

Para o *deployment* do sistema, foi optada a utilização do Docker[14], por ser uma ferramenta bastante eficiente a nível do *deployment* das aplicações. No que diz respeito à documentação da API do projeto, será utilizado o Swagger[15].

Nos próximos pontos, é possível encontrar uma descrição mais detalhada das duas ferramentas mencionadas.

2.4.4.1. Docker

O Docker é uma ferramenta *open source*, que possibilita a *deployment* de aplicações em *containers* autossuficientes, que podem ser executadas na *cloud* ou localmente. Uma vez utilizado o Docker, diversos fatores são garantidos, desde a segurança à diminuição de custos e ao aumento da portabilidade. Os micro serviços ajustam-se facilmente aos *containers,* acelerando o desenvolvimento e a distribuição de uma aplicação. Quando bem projetada, a arquitetura torna-se muito fácil de escalar, uma vês que o Docker permite definir o número de *containers* que estão a correr num determinado micro serviço. Com o Docker, a aplicação pode ser colocada em qualquer *cloud*, pois as dependências são acopladas dentro dos *container* possibilitante, assim, uma fácil portabilidade [65], [66].

Para gerar os *container* são utilizadas Docker *images*, que podem correr em qualquer ambiente independentemente de serem, ou não, do OS [66].

Por recomendação da NearSoft Solutions, o Docker será utilizado para fazer o *deploy* do projeto na *cloud*.

2.4.4.2. Swagger

O Swagger oferece uma interface padrão para o desenvolvimento de API *REST* e a sua correta documentação. Uma API *REST*, permite a interação com serviços remotos sem conhecimento do código fonte dos mesmos ou grande esforço de programação.

O Swagger permite ainda a utilização de duas abordagens – a *Top-Down* e a *Bottom-up*.

A abordagem *Top-Down,* caracteriza-se pela criação, por parte do utilizador, da definição da API no Swagger *editor,* que, por sua vez, possui um detetor de erros. Após a definição desta API, o Swagger gera então a definição dos métodos do servidor através do Swagger Codegen.

[14] Docker, ferramenta para distribuição portátil de aplicações https://www.docker.com
[15] Swagger, framework para o desenvolvimento de API https://swagger.io/

A abordagem *Bottom-Up* permite ao utilizador gerar a definição de uma API através de uma API *REST* já implementada. Esta funcionalidade só está disponível caso o utilizador esteja a utilizar estruturas suportadas. O Swagger possui ainda a ferramenta Swagger UI para a publicação da documentação completa de uma API [67].

O Swagger foi a ferramenta escolhida para a documentação da API, devido à sua simplicidade e às funcionalidade que facilitam a construção e documentação de uma API, tornando possível que a mesma seja facilmente utilizada por terceiros.

3. Abordagem/Método

Neste capítulo, irá ser descrita a abordagem escolhida para implementar a solução para o problema abordado (*ver secção 1.1 e 2.1.1*), utilizando a solução aprovada pela Comissão Europeia – o PSD2 (*ver secção 1.2 e 2.1.2*). Nas secções 3.1 e 3.2 serão abordadas a metodologia de desenvolvimento utilizada e os requisitos da presente tese. Na secção 3.3, será apresentada, em detalhe, a exploração da API PSD2 do *Open Bank* Project, que irá permitir a utilização do PSD2. Pro fim, a secção 3.4 apresenta alguns diagramas de sequência, correspondentes a uma das possíveis implementações, juntamente com a explicação dos mesmos.

3.1. Metodologia de Desenvolvimento

Para que o processo de desenvolvimento de software se realize da melhor forma, torna-se fundamental escolher uma metodologia de desenvolvimento de software. Existe uma grande diversidade de metodologias, desde a tradicional em Cascata às mais recentes Agile [68]. Nesta tese, e devido à falta inicial de tempo para o levantamento de todos os requisitos, foi optada por a metodologia de Prototipagem. Dessa forma, foi possível utilizar os requisitos existentes para construir um protótipo e avalia-o com o intuito da aceitação do mesmo ou ao estabelecimento de novos requisitos [69].

3.1.1. Metodologia de Prototipagem

A metodologia de Prototipagem é uma metodologia de software, cuja principal ideia é o rápido desenvolvimento de um protótipo avaliável, desenvolvido a partir dos requisitos conhecidos. O protótipo geralmente não tem grande detalhe, sendo apenas constituído pelas funcionalidades principais. A este protótipo é feita uma avaliação por parte do cliente, resultando na descoberta de novos requisitos. Através do processo, o cliente consegue ter uma ideia real do sistema, dando, dessa forma, um melhor feedback aos desenvolvedores. Esta metodologia é normalmente utilizada em projetos de grandes dimensões, e com alguma complexidade, em que não existe nenhum processo manual que ajude a compreender os requisitos [69] [70] [71]. Na figura 9, é possível visualizar o funcionamento desta abordagem.

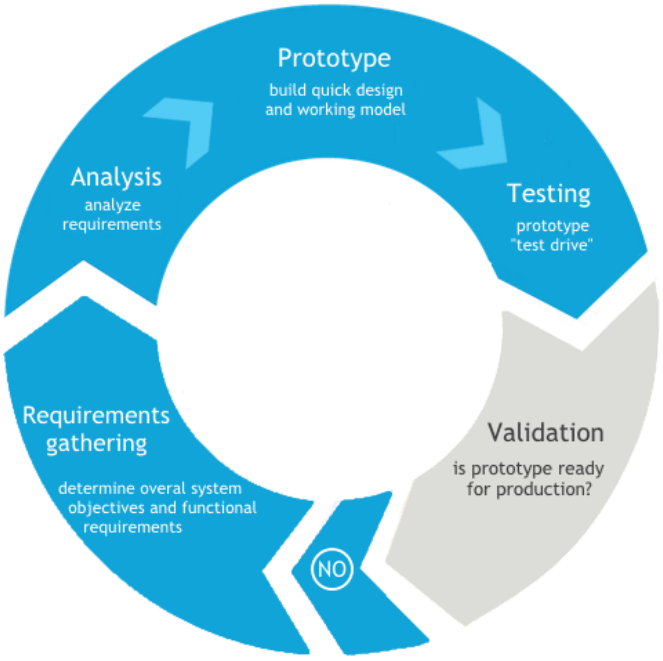

Figura 9 Metodologia de Prototipagem [70]

3.2. Requisitos

Como em todos os projetos de software, a elicitação/levantamento de requisitos é uma das etapas fundamentais. Os requisitos são uma etapa muito importante no ciclo de desenvolvimento, pois permitem dar uma ideia mais concreta do problema em si. No início, as pessoas têm uma ideia de alto nível do projeto. O ato de olhar para o potencial do projeto, e ver o que se pretende, pode ajudar significativamente a ter uma ideia mais concreta. Na Engenharia de Software, os requisitos são vistos como as fundações do projeto e o sucesso deste depende da qualidade dos mesmos [72].

Inicialmente, a nossa fonte de requisitos era apenas o enunciado da tese. O primeiro passo consistiu na interpretação do enunciado e na verificação do que se pretendia. Neste caso, era pretendido a implementação de um módulo de gestão dos pagamentos, de acordo com a norma PSD2, que suportasse aplicações de *e-commerce* e/ou outro tipo de aplicações onde estivessem envolvidas uma compra de serviços online.

Uma vez que um dos principais objetivos desta tese consiste na utilização do PSD2, através de uma API que suporte o mesmo, e que esta norma já tem estabelecidos os seus requisitos, muitos dos requisitos desta tese estão intrínsecos na legislação do PSD2 e na documentação da API utilizada. No que diz respeito à diretiva PSD2, os requisitos estão, de certa forma, dispersos ao longo de vários documentos fornecidos pela Comissão Europeia. Tais documentos podem ser consultados na página *web* https://ec.europa.eu/info/law/payment-services-psd-2-directive-eu-2015-2366_pt.

Por sua vez, os requisitos relacionados com a utilização do OBP API, podem ser encontrados na documentação da API e nas suas instruções de utilização, disponíveis em https://psd2-api.openbankproject.com/.

Mudando o foco para os requisitos mais importantes, nas subsecções que se seguem serão descritos alguns requisitos de negócio, funcionais e não funcionais.

3.2.1. Requisitos de Negócio

Os requisitos de negócio têm como objetivo, geralmente, definir ou restringir os aspetos que o negócio tem como intenção afirmar [73].

Após uma discussão com os orientadores deste projeto, foi possível chegar aos seguintes requisitos de negócio:

Tabela 1 Requisitos de negócio

RN	Requisitos de Negócio
RN.1	O sistema deverá interagir apenas com APIs que suportem a norma PSD2
RN.2	O sistema deverá ser fácil de integrar num site de um vendedor, por exemplo, um botão que possa ser embebido num site 1. Inserir o botão deverá ser a única alteração necessária no site de um vendedor

3.2.2. Requisitos Funcionais

No que diz respeito aos requisitos funcionais, os mesmos estão inteiramente relacionados com o comportamento e funções do sistema, ou seja, com o que o sistema deverá fazer [74]. Com base nos requisitos/objetivos de negócio apresentados em 3.2.1, e após diversas discussões com experts na área do sistema bancário, membros da NearSoft Solutions, consultas da documentação disponível e prototipagem foram identificados vários requisitos funcionais, que se encontram apresentados em várias tabelas.

Na tabela 2, encontram-se apresentados os requisitos funcionais comuns (*Commons*) a todo o sistema. Na tabela 3, por sua vez, encontram-se apresentados os requisitos funcionais relacionados com a gestão de contas (*AIS*). Na tabela 4, encontram-se apresentados os requisitos funcionais relacionados com o pagamento (*PIS*). Na tabela 5, encontram-se aparentados os requisitos funcionais relacionados com os papéis (*Roles*). Os papéis são atribuídos de acordo com o tipo utilizador, para permitir ou restringir certas operações no sistema. Nas tabelas 6, 7, 8, encontra-se apresentados os requisitos funcionais relacionados com os diferentes tipos de utilizadores do sistema, estes podem ser clientes (*Client),* administradores *(Admin)* e ou comerciantes *(Merchant)*. Os clientes são os utilizadores do módulo de pagamento. Os administradores são os gestores do sistema. Os comerciantes são os que utilizam o módulo de pagamento. Por fim, na tabela 9, são apresentas os requisitos funcionais relacionados com as transações realizadas pelo módulo de pagamentos

Tabela 2 Requisitos funcionais comuns

RFCommons	Requisitos Funcionais Comuns
RFCommons.1	O sistema deverá disponibilizar todas as operações através de uma API
RFCommons.2	O sistema deverá disponibilizar toda a documentação necessária à utilização da API
RFCommons.3	O sistema deverá guardar um histórico de todas as operações efetuadas

Tabela 3 Requisitos funcionais relacionados com a gestão de contas, utilizando o AIS do PSD2

RFAIS	**O sistema deverá permitir ao cliente escolher, através do PSD2, a conta com que deseja efetuar o pagamento**
RFAIS.1	O sistema deverá apresentar ao cliente todas as contas de pagamento existente: 1. O sistema deverá apresentar alguma informação detalhada da conta, como por exemplo o saldo
RFAIS.2	O sistema deverá permitir a escolha de uma conta predefinida de pagamentos
RFAIS.3	O sistema deverá apresentar as informações da conta escolhida
RFAIS.4	O sistema deverá possibilitar a alteração da conta escolhida

Tabela 4 Requisitos funcionas relacionados com o pagamento, utilizando o PIS do PSD2

RFPIS	**O sistema deverá realizar pagamentos utilizando a norma PSD2**
RFPIS.1	O sistema deverá inicializar a compra após a seleção da mesma
RFPIS.2	O sistema deverá verificar se a conta de pagamento escolhida tem fundos suficientes para efetuar a compra 1. O sistema deverá apresentar uma conta alternativa, se existente, caso a conta de pagamento escolhida não tenha fundos suficientes 2. O sistema deverá cancelar a compra, caso não existam alternativas

RFPIS.3	O sistema deverá apresentar ao cliente a *charge* (taxa) inerente à transação: 1. O sistema deverá permitir ao cliente aceitar, ou rejeitar, a *charge* 2. O sistema deverá cancelar a compra, caso a *charge* seja rejeitada
RFPIS.4	O sistema deverá gerar um *challenge* em situações que se justifiquem: 1. O sistema deverá apresentar o *challenge* ao cliente 2. O sistema deverá verificar o *challenge* 3. O sistema deverá cancelar a compra, caso o *challenge* não seja bem-sucedido
RFPIS.5	O sistema deverá notificar o cliente o resultado da transação

Tabela 5 Requisitos funcionais relacionados com os Roles

RFRoles	**O sistema deverá permitir a definição de *Roles***
RFRoles.1	O sistema deverá atribuir um determinado *Role* a cada utilizador no momento do registo
RFRoles.2	O sistema deverá possibilitar a criação de novos *Roles*
RFRoles.3	O sistema deverá possibilitar a atribuição de *Roles* aos clientes
RFRoles.4	O sistema deverá restringir certas operações a *Roles* específicos

Tabela 6 Requisitos funcionais relacionados com os clientes do sistema

RFClient	**O sistema deverá permitir a utilização do sistema a qualquer cliente**
RFClient.1	O sistema deverá permitir o registo a qualquer cliente 1. O sistema deverá atribuir o *Role* de “cliente” no momento de registo
RFClient.2	O sistema deverá permitir a realização do login aos clientes
RFClient.3	O sistema deverá permitir a associação do cliente através das credenciais da conta do OBP 1. O sistema deverá utilizar os dados do OBP do cliente para obter uma autorização (*token*) para aceder às contas do mesmo 2. O sistema não deverá guardar as credenciais do OBP do cliente

	3. O sistema deverá guardar apenas a autorização obtida com os dados do OBP
RFClient.4	O sistema deverá fornecer sempre um feedback ao utilizador

Tabela 7 Requisitos funcionais relacionados com o admin

RFAdmin	**O sistema deverá permitir a sua gestão por parte de administradores**
RFAdmin.1	O sistema deverá permitir o registo a administradores 1. O sistema deverá atribuir o *Role* de "*Admin*" no momento de registo
RFAdmin.2	O sistema deverá permitir a gestão dos *Roles*
RFAdmin.3	O sistema deverá permitir a visualização das informações de registo de todos os clientes
RFAdmin.4	O sistema deverá permitir a eliminação de utilizadores
RFAdmin.5	O sistema deverá permitir a visualização das transações dos utilizadores

Tabela 8 Requisitos Funcionais relacionados com o Merchant

RFMerchat	**O sistema deverá permitir a atribuição do *Role Merchant* por parte dos administradores**
RFMerchat1	O sistema deverá permitir a definição de uma conta bancária de recebimento das transações
RFMerchat2	O sistema deverá apresentar a conta de recebimento definida
RFMerchat3	O sistema deverá permitir a associação do *Merchant* a uma determinada conta

Tabela 9 Requisitos funcionais referentes as transações

RFTransactions	**O sistema deverá permitir a consulta de transações**
RFTransactions1	O sistema deverá permitir a consulta de transações aos clientes 1. O sistema deverá permitir a consulta de transações de uma determina conta

RFTransactions2	O sistema deverá permitir a consulta de transações dos clientes aos administradores
RFTransactions2	O sistema deverá permitir a pesquisa dinâmica de transações

3.2.3. Requisitos não Funcionais

Os requisitos não funcionais têm a responsabilidade de especificar a maneira como o sistema deverá funcionar e quais as restrições do seu comportamento [74]. Os principais requisitos não funcionais deste projeto são a Segurança, a Integrabilidade e a Usabilidade.

A Segurança está, desde o início, inerente às operações bancárias e, por isso, não poderia deixar de estar presente nos requisitos não funcionais; a Integrabilidade, está relacionada com o facto de se pretender uma verificação do grau de dificuldade que a integração do sistema de pagamentos apresenta para os *Merchant,* nas suas aplicações de *e-commerce*; a Usabilidade, por sua vez, é pretendida para simplificar ao máximo as tarefas do cliente e do *Merchant,* com o objetivo de verificar a facilidade de utilização do PSD2.

Nas tabelas 10, 11 e 12 encontra-se apresentados os requisitos não funcionais de segurança, integrabilidade e usabilidade respetivamente.

Tabela 10 Requisitos não funcionais de segurança

RNFSegurança	**O sistema deverá ser o mais seguro possível para o utilizador**
RNFSegurança.1	O sistema deverá possuir uma API *Key* para aceder ao OBP API
RNFSegurança.2	O sistema deverá utilizar o *Direct Login (ver secção 3.3.1)* para aceder ao OBP API
RNFSegurança.3	O sistema deverá implementar o mecanismo de autenticação por *token* para todas as operações
RNFSegurança.4	O sistema deverá encriptar toda a sua comunicação utilizando os certificados SSL
RNFSegurança.5	O sistema deverá ter as bases de dados autenticadas
RNFSegurança.6	O sistema deverá minimizar o armazenamento dos dados dos clientes

Tabela 11 Requisitos não funcionais de integrabilidade

RNFIntegrabilidade	**O sistema deverá ser fácil de integrar em qualquer site de *e-commerce***
RNFIntegrabilidade.1	O sistema deverá ser integrado num site de e-commerce com pouco esforço 1. O sistema deverá ser integrado num site de e-commerce, por um Engenheiro experiente, num período máximo de 10h, considerando os testes
RNFIntegrabilidade.2	O sistema deverá funcionar em, pelo menos, 90% dos web *browsers*

Tabela 12 Requisitos não funcionais de usabilidade

RNFUsabilidade	**O sistema deverá ter o maior grau de facilidade possível na sua utilização**
RNFUsabilidade.1	O sistema deverá ser utilizável por utilizadores com conhecimentos básicos em informática
RNFUsabilidade.2	O sistema deverá possibilitar a associação de clientes num período máximo de 5 minutos no que se refere a utilizadores comuns
RNFUsabilidade.3	O sistema deverá possibilitar interfaces intuitivas, para que o processo de definição/alteração da conta de pagamento não exceda os 5 minutos
RNFUsabalidade.4	O sistema deverá possibilitar uma boa interface, para que uma compra não exceda o período máximo de 5 minutos no que diz respeito a utilizadores comuns
RNFUsabilidade.5	O sistema deverá apresentar uma interface intuitiva, e com grande feedback, de modo a que o tempo, desde o registo à compra, não exceda os 15 minutos

3.2.4. Requisitos do Projeto

Conjugando algumas restrições/recomendações à implementação do projeto, por parte dos orientadores, foi possível adicionar mais alguns requisitos, esses requisitos podem ser observados na tabela 13.

Tabela 13 Requisitos de Projeto

RP	Requisitos do Projeto
RP.1	A documentação da API deverá ser feita utilizando o Swagger
RP.2	O botão deverá ser implementado em JavaScript
RP.2	A API deverá oferecer uma aplicação web que permita fazer os pedidos diretamente

3.3. Open Bank Project – API PSD2

Como referido na secção 2.4.1, para a simulação de um ambiente PSD2, será utilizado o *Open Bank Project* (OBP) [47]. Esta plataforma oferece uma *sandbox* PSD2 [51] que possibilita a implementação de aplicações que estão de acordo com a diretiva PSD2. Esta *sandbox* resume-se a uma API *REST*.

Qualquer desenvolvedor pode utilizar a Open Bank Project API PSD2 (OBP API) requerendo, apenas, que este solicite uma API *KEY*, após o registo na plataforma. Essas credencias consistem numa *Consumer Key* e *Consumer Secret,* que são utilizadas no processo de autenticação para identificar o fornecedor de serviço.

Para aceder a qualquer pedido do OBP API é necessária uma autenticação, que pode ser feita utilizando o *Oauth*[16] ou o *Direct Login*[17]. Inicialmente, foi optada a utilização do *Oauth*, por ser suportado através de um SDK[18] em Node.js. No entanto, e devido a limitações que surgiram relacionadas com a necessidade de ser preciso guardar os *tokens* para conseguir acrescentar novos pedidos ao SDK, foi recomendado, pela equipa do OBP, a utilização do *Direct Login*.

O *Direct Login* oferece uma liberdade adicional, o que permitiu desenvolver o software de raiz sem a utilização do SDK. Para todas as rotas existe um endereço base – https://psd2-api.openbankproject.com/obp/v3.0.0/ – ao qual é acrescentada a rota do pedido, por exemplo https://psd2-api.openbankproject.com/obp/v3.0.0/my/accounts.

Os subtópicos que se seguem, dedicam-se à abordagem da forma como utilizar o *Direct Login* e alguns outros pedidos importantes neste projeto. No *header* de todos os pedidos, deverão estar presentes os parâmetros – *Content-Type* e *Authorization.* O *Content-Type* deverá ser "*application/json*". A *Authorization,* que identifica o utilizador, deverá corresponder à *string* "*Direct Login*", concatenada com *token* que provem do *Direct Login*. A resposta de todos os pedidos é dada pela API do OBP em formato JSON.

[16] OAuth - https://github.com/OpenBankProject/OBP-API/wiki/OAuth-1.0-Server
[17] Direct Login - https://github.com/OpenBankProject/OBP-API/wiki/OAuth-1.0-Server
[18] SDK (Software Development Kit) - Porção de software implementado utilizado para o desenvolvimento de aplicações https://techterms.com/definition/sdk

3.3.1. Direct Login

Para aceder aos diversos pedidos da API, é necessária uma autenticação, que pode ser obtida utilizando um pedido POST **/my/logins/direct.** Neste pedido, o *body* deve ser deixado vazio e o *header* deverá conter os seguintes parâmetros – *Content-Type* e *Authorization.*

A *Authorization,* deverá obedecer a uma estrutura *"DirectLogin username='Username do cliente', password='Password do cliente', consumer_key='Consumer-key do desenvolvedor".*

A *consumer_key,* identifica o desenvolvedor do *Payment Initiation Service Provaider* (PISP) ou *Account Information Service Provaider* (AISP), também conhecidos como *Third-Party Providers* (TTP).

O *Username* e a *Password,* por sua vez, identificam o cliente que autoriza aquele fornecedor de serviços de pagamento a realizar o *Payment Initiation Service* (PIS) ou *Account Information Service* (AIS). Como resposta, é obtido um *token,* que autoriza determinado desenvolvedor de serviços utilizar livremente os pedidos do PSD2 (PIS e AIS) para o específico cliente. Esta autorização tem um período de vida de 1 mês, necessitando depois ser renovada. Este período de vida é definido nos ficheiros de configuração do OBP API pelos administradores da API.

3.3.2. Get Transaction Request Types for Account

Este pedido permite verificar quais são os *Transaction Request Types* de uma conta. Os *Transaction Request Types* podem ser SEPA, COUNTERPARTY, FREE FORM ou SANDBOX_TAN.

Por *default,* as contas criadas na *sandbox* apenas têm disponível, sem que seja necessário configurar, o *Transaction Request Type* SANDBOX_TAN, que é forma mais comum utilizada em modo *sandbox* [50]. Nesta tese, apenas será utilizado o *Transaction Request Type* SANDBOX_TAN.

Para cada *Transaction Request Type* está definida uma *charge,* que representa a taxa inerente à transação. Os valores da taxa e a moeda correspondente, podem ser obtidos na resposta do pedido. Para aceder a esta informação, é necessário realizar um pedido HTTP GET **/banks/BANK_ID/accounts/ACCOUNT_ID/Owner/transaction-request-types**. Os parâmetros BANK_ID e ACOUNT_ID, presentes no *path,* identificam o banco e a conta em questão. A resposta a este pedido corresponderá aos *Transaction Request Types* da conta e à *charge* correspondente.

3.3.3. Create Transaction Request

O *Create Transaction Request* é um pedido fundamental neste projeto, pois permite inicializar o pagamento pela TPP, que é um dos pontos fundamentais do PSD2. Este pedido pode ser feito por qualquer TPP, devendo satisfazer alguns requisitos do PSD2, nomeadamente [51]:

- O utilizador/cliente deverá ser informado da taxa inerente à transação
- O pedido deverá utilizar *delegated authentication* (*OAuth*)

De acordo com as regras impostas pela Comissão Europeia, descritas na secção 2.1.2, caso a valor da transação ultrapasse o limite, o *status* da transação será *INITIATED,* sendo necessário um *Answer*

Transaction Request Challenge para que a transação seja *COMPLETED*. Para as transações abaixo desse valor, o *status* será *COMPLETED,* sem que seja necessário o *Answer Transaction Request Challenge.* Nesta API, o valor está predefinido para 100 unidades monetárias, podendo este valor ser alterado.

Para aceder a este pedido é necessário fazer um pedido POST para o *endpoint* **/banks/BANK_ID/accounts/ACCOUNT_ID/VIEW_ID/transaction-request-types/TRANSACTION_REQUEST_TYPE/ transaction-requests**. Alguns dos parâmetros necessários ao pedido estão contidos no *path*, como são exemplos o BANK_ID e o ACCOUNT_ID, que correspondem ao ID do banco e à conta do utilizador/cliente respetivamente. A VIEW_ID corresponde à *view*, que é geralmente utilizada para delegar acessos a entidades como *accountants*, *shareholders*, etc.

Neste projeto será utilizada a *default view*, muitas vezes designada na documentação como OWNER. O TRANSACTION_REQUEST_TYPE consiste na forma em como irá ser feita a transação, sendo, neste caso, a SANDBOX_TAN (*ver secção 3.3.2*). No que diz respeito ao *body* da transação, este será estruturado da seguinte forma:

```
{
  "to":{"bank_id":"gh.29.uk", "account_id":"8ca8a7e4-6d02-48e3-a029-
0b2bf89de9f0"},
  "value":{"currency":"EUR", "amount":"10"},
  "description":"Good"
}.
```

O "`bank_id`" e o "`account_id`", representam o ID do banco e a conta de destino respetivamente. A "`currency`", identifica o tipo de moeda com que a transação é feita. O "`amount`", por sua vez, corresponde à quantidade de unidades monetárias. Por fim, a "`description`" diz respeito a alguma descrição que possa estar associada à transação.

No que diz respeito à resposta do pedido, se tudo for realizado conforme o pretendido, é devolvido um resumo da informação da transação, juntamente com o *status* e o *challenge*. Caso o *status* seja INITIATED*,* o parâmetro *challenge* contém um conjunto de informações necessárias à *Answer Transaction Request Challenge*. Por outro lado, se o *status* for COMPLETED*,* o parâmetro *challenge* deverá ser *null*.

No próximo tópico será apresentado o procedimento a ter para concluir transações com *status* INITIATED.

3.3.4. Answer Transaction Request Challenge

Como foi explicado em 3.3.3, após o pedido *Create Transaction Request*, as transações cujo o valor ultrapasse o limite, deverão estar identificadas como *status INITIATED.* De modo a que seja possível concluir este tipo de transações, o OBP disponibiliza o pedido POST **/banks/BANK_ID/accounts/ ACCOUNT_ID/VIEW_ID/transaction-request-types/TRANSACTION_REQUEST_TYPE/ transaction-requests/ TRANSACTION_REQUEST_ID/Challenge.** Os parâmetros presentes no *path*,

nomeadamente o BANK_ID, ACCOUNT_ID, VIEW_ID e TRANSACTION_REQUEST_TYPE, são os mesmos que os encontrados para o pedido *Create Transaction Request (ver secção 3.3.3)*. O parâmetro TRANSACTION_REQUEST_ID, refere-se ao ID da resposta do pedido *Create Transaction Request (ver secção 3.3.3)*. No *body* do pedido deverão ser apresentados os seguintes parâmetros:

```
{
  "id":"7e6bc76-7e9d-4553-bc6a-fbd54cb67789",
  "answer":"12345"
}
```

O "`id`", corresponde o ID do *challenge* da resposta ao pedido *Create Transaction Request* podendo o parâmetro "`answer`" ter qualquer valor. A resposta será a mesma do pedido anterior, com a diferença do *status* que estará como *COMPLETED*. Num ambiente real, o A*nswer Transaction Request Challenge* pode representar um dos passos SCA, um dos requisitos do PSD2.

3.3.5. Get Accounts at all Banks

Um outro pedido bastante importante, principalmente no que se refere ao AISP, é o pedido *Get Accounts at all Banks*, que permite obter todas as contas de um utilizador em todos os bancos. Para fazer este pedido basta fazer um pedido GET **/my/accounts**. O retorno deste pedido corresponde às informações gerais das contas bancárias, como o "`account_id`" e o "`bank_id`", sem referência ao valor monetário da conta. Caso seja pretendido a consulta de valores detalhados da conta, deverá ser utilizado o pedido *Get Account by Id,* que irá ser abordado *(ver secção 3.3.6).*

3.3.6. Get Account by Id

Este pedido, de certa forma, complementa o pedido anterior, quando é pretendido saber informação detalhada à cerca de uma determinada conta, como os valores monetários da conta e o tipo de moeda. Para realizar este pedido, é necessário realizar um pedido HTTP GET para o pedido **/my/banks/BANK_ID/accounts/ACCOUNT_ID/account.** Os parâmetros BANK_ID e ACCOUNT_ID, presentes no *path*, representam o ID do banco e da conta do utilizador. A resposta a este pedido corresponde às informações da conta, como o valor monetário, tipo de moeda, informações sobre o tomador da conta, informações de *routing* necessárias a alguns tipos de *Transasctions Request Types*, etc.

3.4. Proposta de Solução

Com os requisitos iniciais, e os obtidos através das reuniões presentes com os orientadores, foi possível chegar a uma proposta de solução.

Os próximos subtópicos estão reservados à apresentação de alguns diagramas de sequência[19], que demonstram a proposta de funcionamento, de acordo com os requisitos estabelecidos. Neles, será possível visualizar algumas entidades comuns que comunicam entre si, nomeadamente o *NearSoft*

[19] Diagramas de sequência - Diagramas que demonstram um fluxo lógico de um sistema, http://www.agilemodeling.com/artifacts/sequenceDiagram.htm

Payment Provider (NPP) API, *NearSoft Payment Provider* (NPP) APP, *Open Bank Project* (OBP) API, e por fim o *Widget* (*Payment Button*).

O NPP API diz respeito a todo o *Back-End* do sistema, onde são atribuídos os *Roles,* geridas as permissões de cada utilizador, e as autorizações, quer de entrada da própria API, quer da comunicação com OBP API. É ainda no NPP API que são guardados todos os dados do utilizador, necessários à realização das transações (compras), assim como o registo das mesmas.

O NPP APP, é uma aplicação web onde o utilizador pode fazer o registo na plataforma, associar a conta OBP, escolher a conta de pagamento predefinida e ver o histórico das transações (compras). De uma forma bastante resumida, o NPP APP é á interface visual de uma parte do NPP API.

Por último, o *Widget* (*Payment Button*) corresponde aos botões de pagamento que são embebidos no *e-commerce* website. Este *Widget*, é responsável por gerir o fluxo do pagamento, ligando-se também diretamente ao NPP API.

O OBP API corresponde à API utilizada para usufruir do ambiente PSD2 (*ver secção 2.1.2 e 3.3*).

3.4.1. Operações Comuns a Todas as Operações

De forma a não sobrecarregar as secções, nesta seção serão explicadas algumas operações que são comuns a todos os pedidos.

Focando nos pedidos realizados pelo NPP APP e o *Payment* Widget ao NPP API, todos os pedidos, à exceção do pedido de registo, devem ser acompanhados por uma autorização (*token*) que identifique o cliente. Apesar de não estar representado nos diagramas que se seguem, para não sobrecarregar os mesmos, em todos os pedidos recebidos pelo NPP é verificado se o cliente tem permissões para aceder ao pedido através do seu *Role*. Para todos os pedidos realizados pela NPP API ao OBP API é requerida uma autorização (*token*), obtida através do *Direct Login*. Esta autorização e única para cada utilizador, de forma a permitir identificar o mesmo (*ver secção 3.3.1*).

3.4.2. Obtenção da API KEY

O diagrama de sequência apresentado na figura 10, representa o processo de obtenção da API KEY. Este processo é fundamental para adquirir a autorização para aceder às diversas funcionalidades da API.

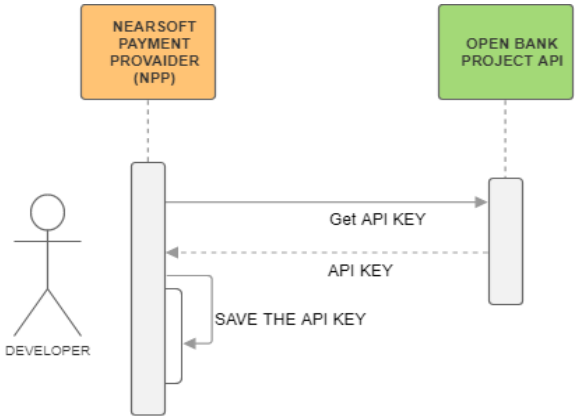

Figura 10 Processo de obtenção da API KEY

A obtenção da API KEY é feita, inicialmente, antes de qualquer desenvolvimento, através de um registo, do desenvolvedor, na página do OBP (https://psd2-api.openbankproject.com/consumer-registration). Uma vez obtida a API KEY, torna-se indispensável salvá-la, uma vez que será necessária sempre que seja preciso pedir uma autorização para um determinado cliente, como iremos ver no próximo diagrama.

3.4.3. Gestão de Associação da Conta OBP

Para que o NPP possa realizar operações bancárias, para um determinado cliente, é necessário que este esteja associado ao OBP API. Este processo de associação, corresponde à obtenção de uma autorização fornecida pelo OBP API, para que o NPP possa utilizar os dados do utilizador em questão nas diversas operações bancárias.

O diagrama da figura 11, demonstra o processo de associação da conta do *Open Bank Project* (OBP).

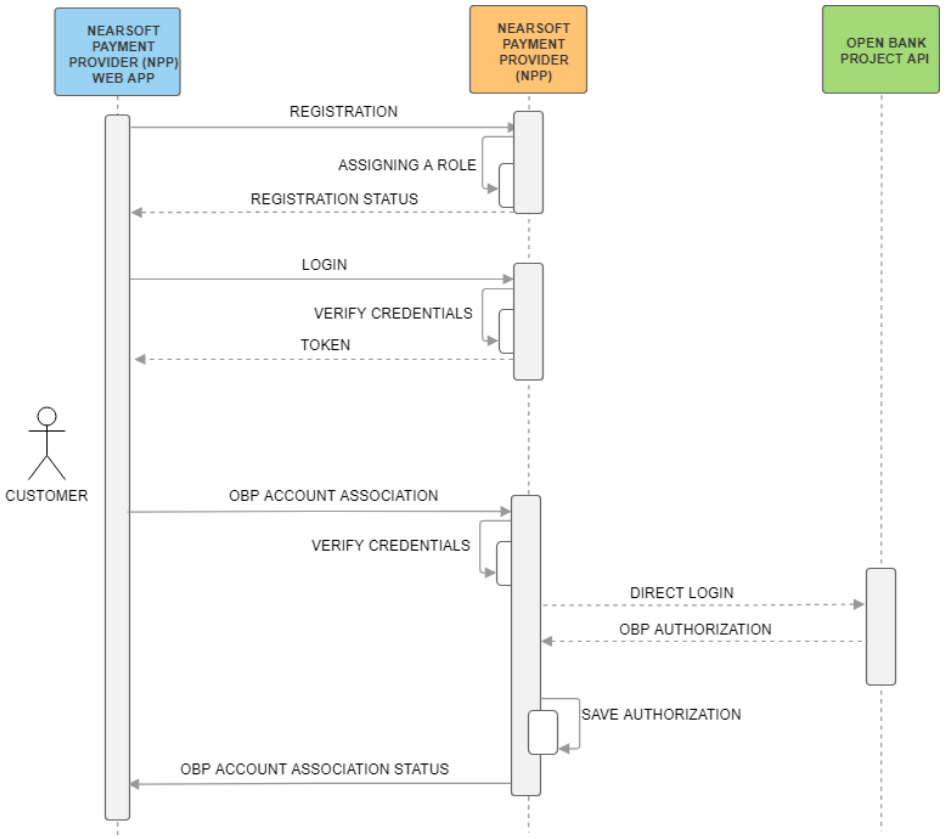

Figura 11 Processo de associação da conta OBP com o NPP

Inicialmente, e caso o cliente queira utilizar o NPP, este tem de efetuar um registo com a suas credenciais. Após o registo, é necessário fazer login para que seja obtida a *token* (autorização) para utilizar os diversos recursos do sistema. Para uma utilização produtiva do NPP, é necessário o utilizador/cliente ter uma conta num dos bancos do OBP[20]. Caso o utilizador já tenha a conta, basta efetuar a associação. O processo de associação corresponde ao *Direct Login* (*ver secção 3.3.1*). Através do *Direct Login* é obtida uma autorização, que permite o NPP poder fazer pedidos ao OBP API para o utilizador em questão. Esta autorização é armazenada na base de dados, pelo NPP. Em qualquer momento, se pretendido, o cliente pode remover a sua associação, não dando mais autorização ao NPP para o acesso às suas contas bancárias. O processo de remoção da associação pode ser verificado no diagrama da figura 12.

[20] Pode ser efetuado um registo num dos bancos do OBP API, através do seguinte link https://psd2-api.openbankproject.com/user_mgt/sign_up

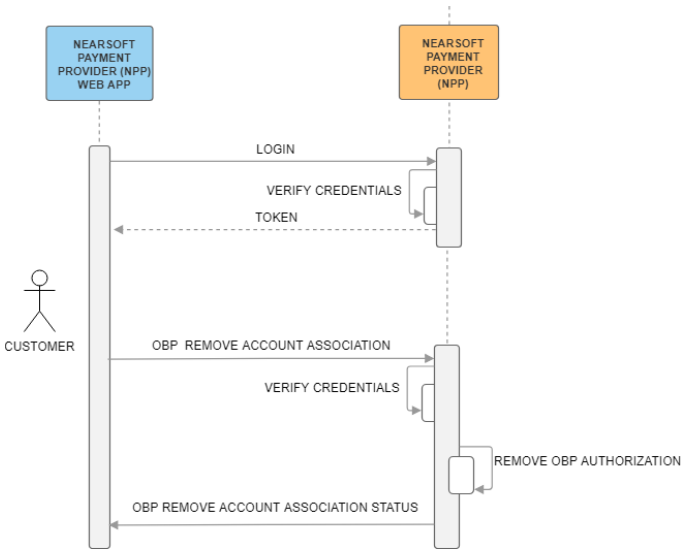

Figura 12 Processo de remoção da associação da conta OBP API

Este processo é baste semelhante ao processo de associação, sendo os primeiros passos iguais, diferindo no intuito que, neste caso, consiste em remover a associação.

3.4.4. Integração com o *Merchant*

Um dos aspetos obrigatórios de um sistema de pagamento, é permitir uma quantidade diversificada de *Merchants.* Esta funcionalidade trás certos desafios, tais como identificar a que *Merchant* pertence uma dada compra, definir e atualizar a conta de receção de pagamentos de um determinado *Merchant*, entre outros. De modo a clarificar estes problemas, será apresentado na figura 13 uma das possíveis propostas de solução, juntamente de uma breve explicação da lógica de funcionamento.

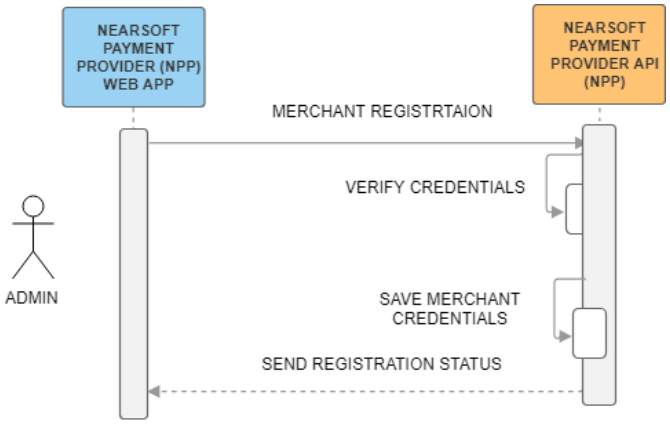

Figura 13 Processo de registo do Merchant

Aquando da criação do registo do *Merchant* por parte dos administradores, é definida, nesse momento, a conta de recebimento dos pagamentos e gerada uma KEY que tem com função identificar o *Merchant.* Essa KEY é utilizada em todos os pagamentos, com o objetivo de identificar o *Merchant* a que pertence uma determinada compra, e dar a conhecer o mesmo, para obter os dados da conta de recebimentos

das compras. A conta de recebimento poder ser alterada pelo *Merchant* em questão sempre que este o pretender, e uma vez que se encontre devidamente identificado.

3.4.5. Definir uma Conta de Pagamento Predefinida

A plataforma NPP irá tirar partido do AIS (*ver secção 2.1.2.5*), introduzido através do PSD2, providenciando funcionalidades relacionadas com a gestão de informação da conta. Uma dessas funcionalidades, consiste na possibilidade de escolha de uma conta de pagamento predefinida, quando o utilizador possui mais do que uma conta bancária.

No diagrama da figura 14, pode ser observada a sequência de execução das funcionalidades referidas anteriormente.

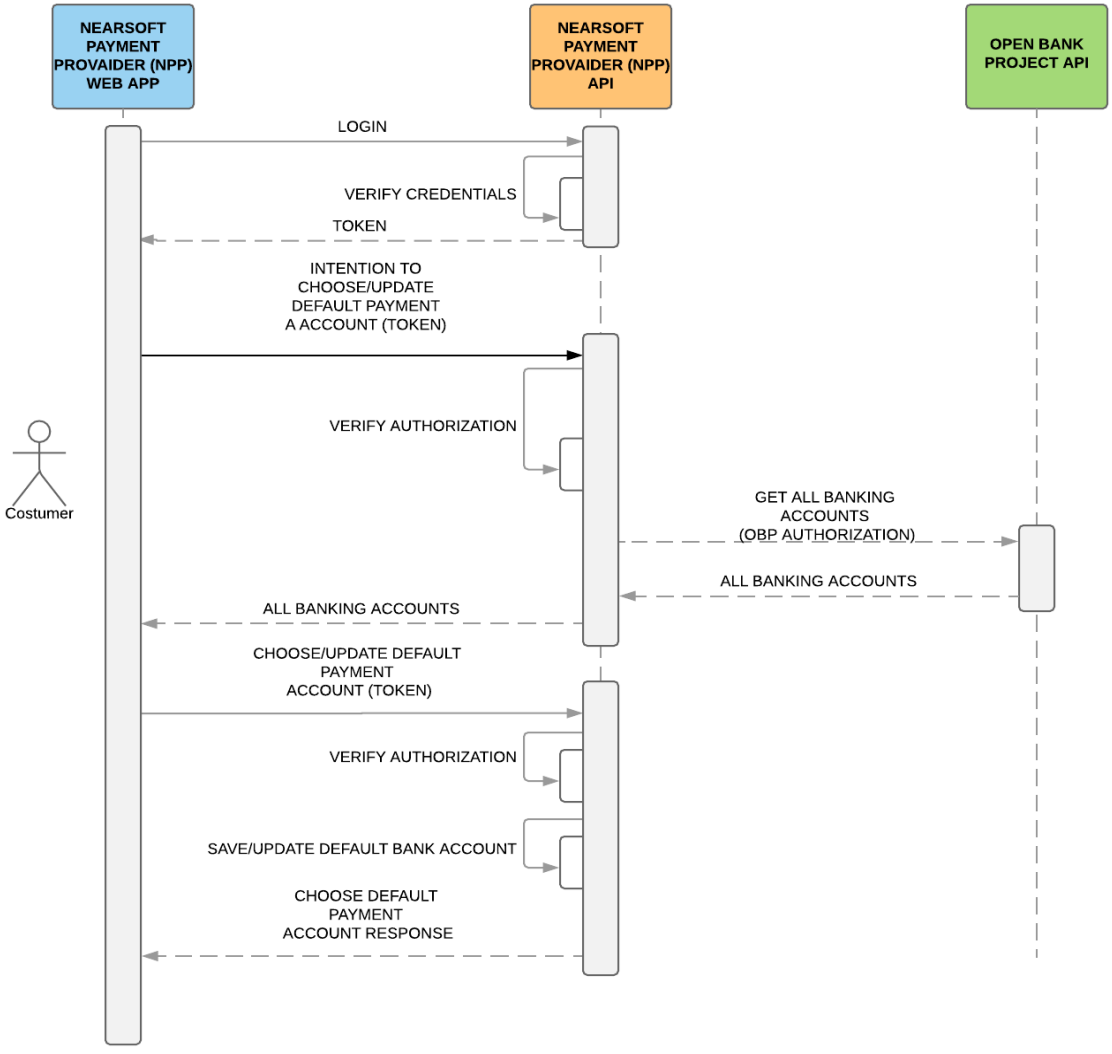

Figura 14 Definição de uma conta de pagamento predefinida

Se houver o caso em que o cliente pretende alterar, ou definir, a sua conta de pagamentos predefinida, este terá de fazer um pedido ao NPP API. Após ser identificado, são efetuados pedidos às rotas **/my/accounts** e **/my/banks/BANK_ID/ accounts/ACCOUNT_ID/account**, (*ver secção 3.3.5 e 3.3.6*), a fim de obter os dados das contas bancárias do cliente, disponíveis OBP API. Uma vez presentes os

dados no NPP API, este cruza os dados e apresenta no NPP WEB APP, para facilitar a escolha por parte do utilizador/cliente.

Após a escolha do cliente, um novo pedido é feito ao NPP, desta vez por parte da NPP WEB APP, no sentido de salvar os dados da nova conta de pagamentos predefinida. Esses dados são armazenados na base de dados do NPP, para a eventualidade de ser necessário realizar um pagamento.

3.4.6. Fluxo de Pagamento

Dado que um dos principais objetivos desta tese é a criação de um modulo de gestão de pagamentos, o PIS (*ver secção 2.1.2.5*), proveniente do PSD2, não poderia deixar de estar presente neste projeto. Este serviço tem como principal funcionalidade, a capacidade de inicialização de um pagamento através de um pedido de transação, fazendo com que se torne possível executar pagamentos através de transferências de fundos entre duas contas bancárias.

Ao PIS foi ainda decidido acrescentar funcionalidades do AIS, no fluxo de pagamento, para facilitar a usabilidade. Desta forma, é possível adicionar ao fluxo de pagamento a funcionalidade da sugestão "inteligente" de uma outra conta, quando a conta de pagamentos predefinida não tem os fundos suficientes para a concretização da compra. Esta funcionalidade pode ser despoletada num cenário em que o utilizador pretenda comprar um produto num site de *e-commerce* e escolha pagar com NPP. Não tendo os fundos suficientes na conta que configurou como predefinida, e em vez de deixar a compra em *standby* para ir ao site da plataforma NPP configurar uma nova conta com fundos suficientes, o próprio sistema encarrega-se de sugerir outra conta que tenha fundos suficientes, caso esta exista. Esta sugestão é feita diretamente da aplicação *e-commerce*, sendo da escolha do utilizador/cliente aceitar, ou não.

De acordo com os requisitos, é pretendido um botão que, aquando de um clique no mesmo numa página *e-commerce*, realize o pagamento a partir de um PISP, tornando a compra, um ato efetuado com a menor fricção possível. Contudo, e de acordo com o PSD2, algumas regras têm de ser cumpridas, como por exemplo a informação ao utilizador da taxa da transação, frequentemente designada na documentação da API como *charge*, e a resposta a um *challenge* em determinadas circunstâncias (*ver secção 3.3.2 e 3.3.4*).

No diagrama de sequência apresentado na figura 15, é possível observar um dos possíveis fluxos deste processo.

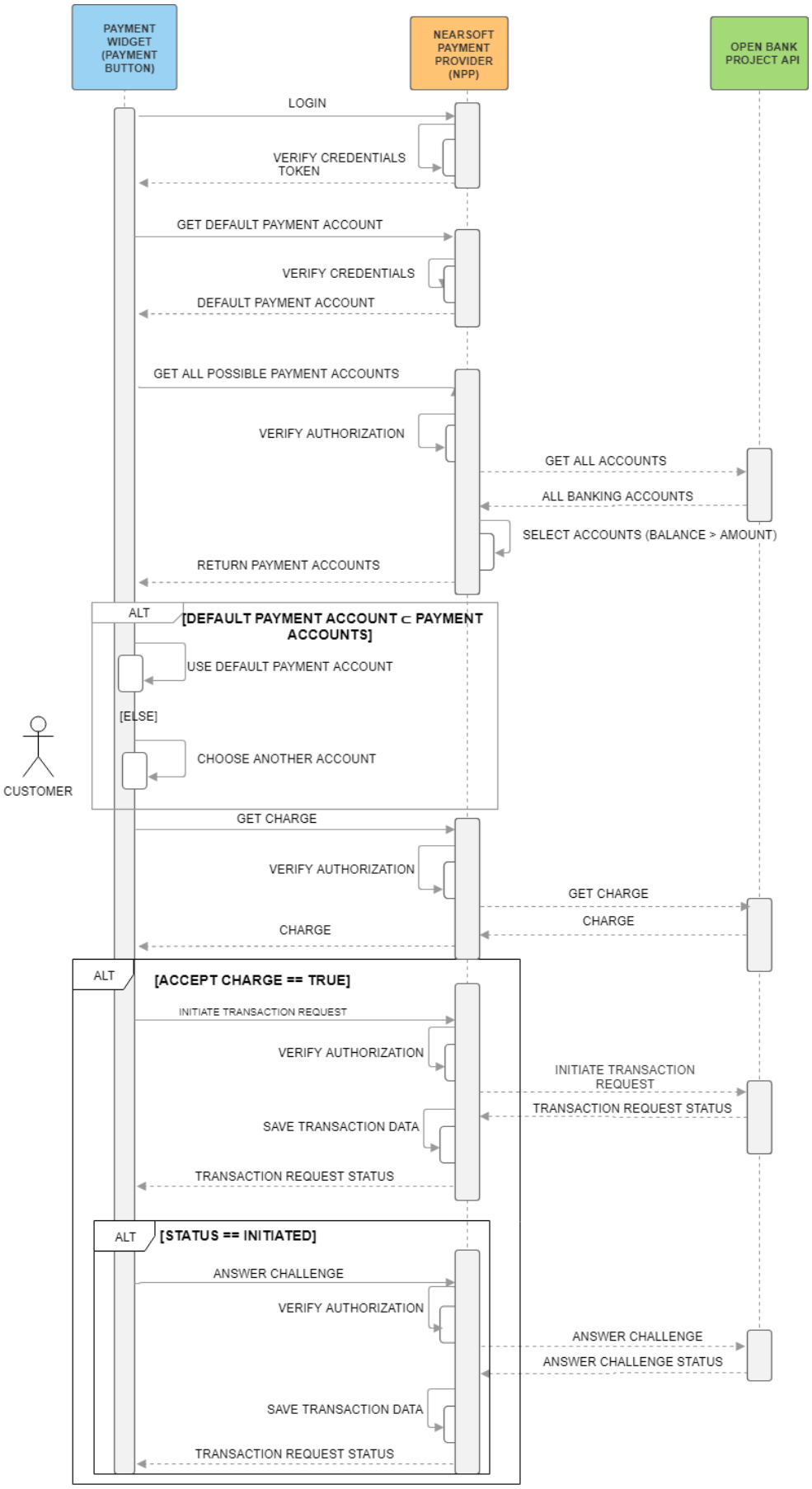

Figura 15 Processo de realização de um pagamento utilizando o botão do NPP

Quando o utilizador/cliente clica no botão "*Buy Now*" ou "*Buy All*", é aberto um *pop up* (integrado com os botões) no site de *e-commerce,* para fazer login através do NPP, para obter então, a autorização necessária para realizar todas as etapas do pagamento, uma vez que este permite identificar o utilizador e obter a autorização do OBP da base de dados.

A primeira etapa que é realizada, corresponde a um pedido ao NPP API para obter a conta de pagamento predefinida do respetivo utilizador/cliente. Quando o PAYMENT WIDGET recebe os dados da conta de pagamento predefinida, envia um novo pedido ao NPP API a solicitar todas as contas de pagamento daquele utilizador que tenham uma quantidade monetária superior ao valor do produto. Para tal, o NPP API faz pedidos ao OBP API para as rotas **/my/accounts** e **/my/banks/BANK_ID/accounts/ACCOUNT_ID/account** (*ver secção 3.3.5 e 3.3.6*), e cruza a informação das contas selecionando apenas as que têm um valor monetário superior ao valor do produto enviando posteriormente essas mesmas contas para o PAYMENT WIDGET. Lá, é verificado se a conta de pagamento predefinida está incluída nas contas possíveis de pagamento. Caso esteja contida, o sistema avança para a próxima etapa do pagamento com a conta predefinida. Caso não esteja contida, é selecionada uma das possíveis contas de pagamento apresentado ao utilizador a possibilidade. No pior dos cenários, caso não exista nenhuma conta de pagamento disponível, a transação é automaticamente cancelada.

Uma vez definida a conta de pagamento a ser utilizada para efetuar a compra, e segundo as regras do PSD2, é necessário informar ao utilizador a taxa inerente à transação, muitas vezes designada na documentação da API por *charge*. Para que tal aconteça, o PAYMENT WIDGET realiza um pedido ao NPP API a fim de obter a taxa correspondente à conta de pagamento definida. O NPP API, por sua vez, faz um pedido GET ao OBP API para o pedido **/banks/BANK_ID/accounts/ACCOUNT_ID/Owner/transaction-request-types** (*ver secção 3.3.2*) obtendo então a taxa da transação, que é posteriormente comunicada ao PAYMENT WIDGET e apresentada ao utilizador, sendo da escolha do mesmo continuar, ou cancelar, a transação.

Posto este processo, o utilizador/cliente fica apto a avançar com a compra, e, para que se conclua, o PAYMENT WIDGET comunica ao NPP API a intenção de avançar com a mesma. Entretanto, no NPP API, é verificado de que *Merchant* corresponde a compra, e são obtidos os dados da sua conta de recebimento. Uma vez obtidos todos os parâmetros necessários, o NPP API envia o pedido POST **/banks/BANK_ID/accounts/ACCOUNT_ID/VIEW_ID/transaction-request-types/TRANSACTION_REQUEST_TYPE/transaction-requests** (*ver secção 3.3.3*), com os devidos parâmetros*,* a transação é executada e o *status* da transação é enviado para o cliente, como é possível verificar no diagrama anterior. Caso o valor da transação seja inferior ao valor limite, o *status* será CONCLUDED e o utilizador é informado sobre o sucesso da mesma. Por outro lado, se o *status* for INITIETED, é requerida uma resposta a um *challenge* pelo utilizador. Em ambos os casos, toda a informação essencial à transação é salva nos *log* e na base de dados das transações efetuadas.

Para a resposta ao *chellange,* o PAYMENT WIDGET envia um pedido ao NPP API que, por sua vez, faz um pedido POST à rota do OBP API **/banks/BANK_ID/accounts/ACCOUNT_ID/VIEW_ID/transaction-request-types/TRANSACTION_REQUEST_TYPE/transaction-requests/**

TRANSACTION_REQUEST_ID/Challenge (ver secção 3.3.4). O *challenge* é respondido e o utilizador/cliente é informado do sucesso da transação. Tal como no pedido de inicialização da compra, as informações importantes à transação são igualmente salvas.

3.5. Conclusão

Uma das etapas mais importantes na realização do projeto, foi a definição de uma abordagem metodológica coerente. Devido aos escassos requisitos no início do projeto, foi determinado, em conjunto com os orientadores que, a utilização da técnica de prototipagem seria o caminho a seguir. Através dela, foi possível adquirir uma perceção mais detalhada do projeto e, dessa forma, estabelecer mais concretamente os requisitos.

Tendo em conta que um dos requisitos fundamentais consistia na construção de um módulo de pagamento utilizando a norma PSD2, tornou-se fundamental a exploração de uma API que suportasse a mesma. Assim sendo, e por recomendação da NearSoft Solutions, a API utilizada foi a OBP API.

Para que qualquer desenvolvedor possa utilizar a OBP API, é necessário obter uma API KEY. De forma a obter uma autorização para aceder aos dados de qualquer utilizador/cliente, o *Direct Login* tem de ser realizado (*ver secção 3.3.1*). A OBP API disponibiliza uma diversidade de pedidos que permitem a implementação do AIS e o PIS, serviços introduzidos com o PSD2 e que estão presentes neste projeto.

Uma vez definidos os requisitos, e explorada a API, implementação da plataforma de pagamentos passou a ser o foco do projeto. Sendo este um tema inovador que requer algum esforço de perceção, foi optado o uso do desenho de vários diagramas de sequência divididos por tarefas. Estes diagramas de sequência (*ver secção 3.4*), permitem ver ao pormenor algumas das etapas mais importantes que são pretendidas no projeto, nomeadamente o fluxo do registo e associação do utilizador, a definição da conta de pagamento e o próprio pagamento. Após a conclusão do desenho dos diagramas de sequência, foi iniciada então a implementação do módulo de pagamentos, abordada no próximo capítulo.

4. Implementação

O presente capítulo é reservado à abordagem da implementação dos principais aspetos do sistema desenvolvido. Este pode ser complementado com a secção 3.4, onde se encontra a explicação dos procedimentos mais importantes do sistema, através dos diagramas de sequência e da interpretação dos mesmos.

Como pode ser observado na figura 16, a arquitetura completa deste sistema está dividida em 4 componentes – *NearSoft Payment Provider* (NPP) API, *NearSoft Payment Provider* (NPP) APP, *Payment Widget* e *Open Bank Project* (OBP) API. A comunicação destes componentes é feita através do mecanismo de comunicação síncrono HTTP [75], utilizando a notação JSON. A esta comunicação é, ainda, adicionada o protocolo SSL, de forma a garantir que os dados não sejam enviados sem encriptação.

O NPP API é uma API *RESTful*, que contém toda a lógica do sistema e que é implementada utilizando a *micro-framework* Flask [60]. Este componente, tem a capacidade de resolver/responder os pedidos do NPP APP e do *Payment Widget*, sendo também necessária a comunicação com o OBP API, que utiliza o PSD2, no sentido de realizar as operações bancárias do utilizador. O NPP API será abordado em detalhe na secção 4.1.

O NPP APP, por sua vez, caracteriza-se como uma aplicação web onde qualquer pessoa se pode registar e tirar partido dos serviços de pagamento e de gestão oferecidos. Esta aplicação será apresentada com pormenor na secção 4.2.

O *Payment Widget*, que será descrito na secção 4.3., corresponde ao botão desenvolvido com a linguagem JavaScript, que poderá ser embebido em qualquer site *e-commerce*.

O OBP API (*ver secção 3.3*), corresponde a API externa PSD2 [76].

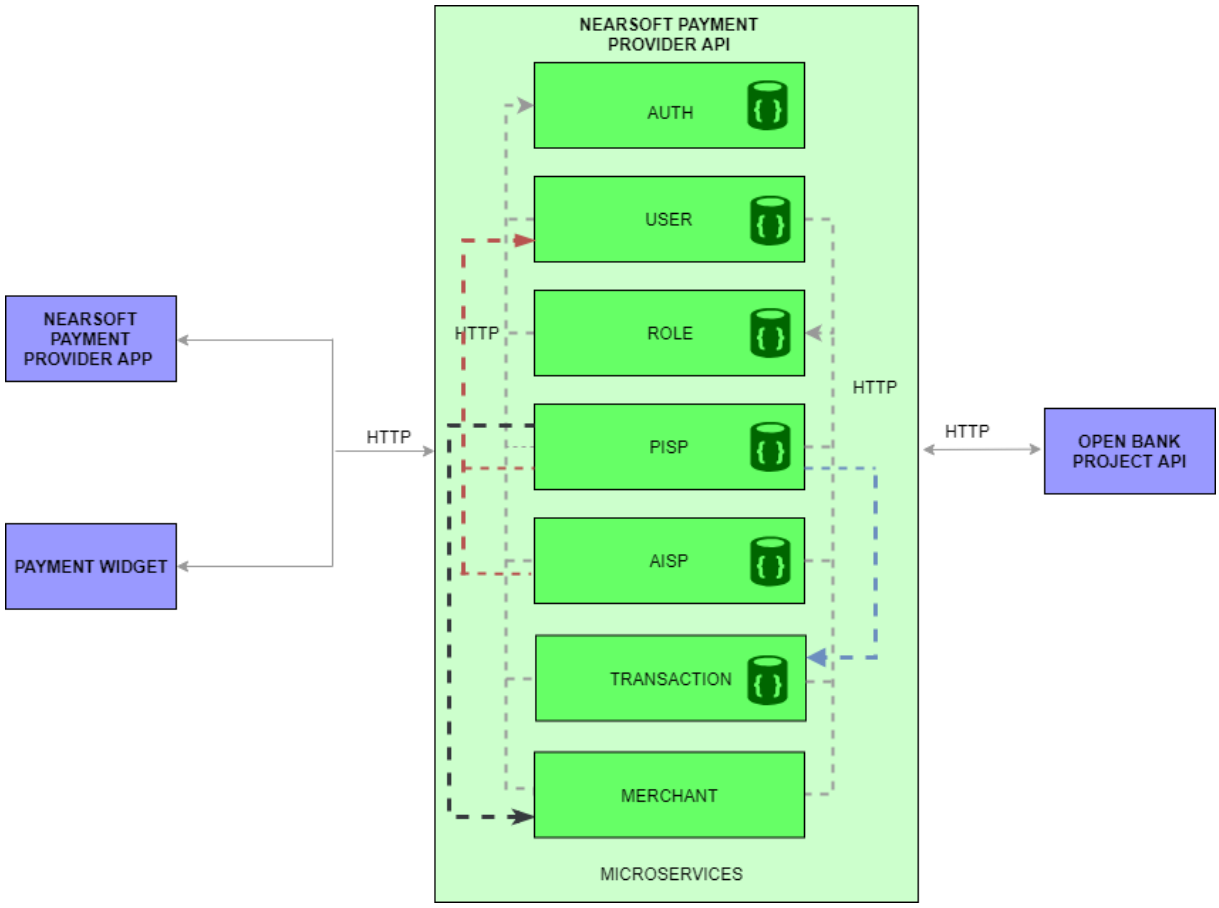

Figura 16 Arquitetura do Sistema

4.1. NearSoft Payment Provider API

Como referido anteriormente, o NPP API é o componente central de todo o sistema, e encontra-se dividido em vários micro serviços – *Auth*, *User*, *PISP*, *AISP*, *Role*, *Transactions* e *Merchant*. Cada micro serviço é completamente isolado dos outros, tendo a sua própria base de dados em MongoDB, ficheiros de configuração, API, *requirements*, entre outros. As bases de dados dos micro serviços estão protegidas pelo sistema de autenticação do MongoDB, utilizando o mecanismo de autenticação "SCRAM-SHA-1".

De modo a não tornar esta secção exaustiva, não serão abordados quaisquer parâmetros necessários aos pedidos dos diversos micro serviços. Assim sendo, será apenas apresentado a explicação do propósito de cada um dos parâmetros.

Todos os micro serviços apresentam uma resolução para exceções que possam acontecer, apresentando, dessa forma, a informação ao utilizador e mantendo execução do micro serviço. A documentação dos micro serviços foi feita utilizando o Swagger [67]. Desta forma, é possível aceder à documentação e realizar os pedidos numa página web.

Na figura 37 do **Anexo C,** pode ser visualizado um exemplo da documentação de um dos pedidos. Todos os micro serviços têm um registo *log* de todas as operações realizadas. A figura 39 do **Anexo C,** ilustra um exemplo do ficheiro *log* do micro serviço PISP.

Existem algumas operações que são comuns à maior parte dos micro serviços, e que operam como *decorators* que "decoram" os pedidos. Na linguagem Python de programação, um *decorator* é uma função que recebe uma outra e estende o comportamento da mesma, sem modificá-la explicitamente. Os *decorators* são geralmente utilizados para adicionar uma determinada funcionalidade a uma ou mais funções [77]. Neste sistema, existem três *decorators* diferentes, sendo um para verificar a autorização do utilizador/cliente dentro do sistema, outro para gerir a autorização de aceder ao OBP API e outro para a verificação do *Role* do utilizador.

Focando um pouco na autorização no sistema, é fundamental que a maior parte dos pedidos estejam sujeitas a uma autorização através de um *authorization token*. A esse *authorization token* é necessário fazer uma verificação, que é conseguida através da comunicação entre o *decorator,* que autêntica os clientes dentro do sistema, e o micro serviço *Auth.* É durante essa comunicação que o determinado *authorization token* é enviando. O micro serviço *Auth,* por sua vez, verifica a validade do *authorization token* e identifica o cliente enviando de volta a sua identificação.

No que diz respeito à autorização para aceder ao OBP API, este *decorator* está apenas presente nos micro serviços AISP e PISP. A sua função é utilizar a identificação do utilizador que se encontra no sistema, e que é obtida através do *decorator* que verifica a *authorization token* no sistema, para obter a autorização para aceder ao OBP API, que por sua vez está armazenada na base de dados do micro serviço *User (ver secção 3.3.1 e 3.4.3).*

Por fim, o *decorator* que verifica os *Roles*, permite, ou nega, o aceso ao utilizador a certas funcionalidades do sistema. Para que tal aconteça, é utilizada a identificação do utilizador, através de um pedido ao micro serviço *Role*, para obter os seus *Roles*. Depois de obtida a lista de todos os *Roles,* para determinado utilizador, é verificado se o *Role* necessário para aceder à funcionalidade está na mesma.

Nos pontos que se seguem encontram-se apresentados alguns aspetos importantes de cada um dos micro serviços.

4.1.1. Auth

O *Auth* é um micro serviço inteiramente dedicado à autenticação e autorização, sendo disponibilizada uma rota para cada uma das tarefas.

Na figura 17, é possível visualizar as rotas disponibilizadas através da documentação da API gerada pelo Swagger [67].

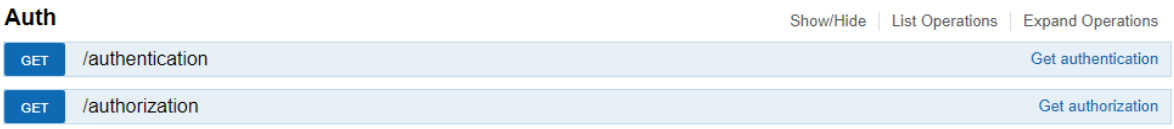

Figura 17 Pedidos disponibilizadas pelo Micro Serviço Auth

Ao utilizar o pedido GET ***/authentication,*** é atribuído um *authorization token* com uma determinada validade, que identifica cada utilizador/cliente. O pedido GET ***/authorization*** recebe um *authorization token* e após a sua verificação, autoriza, ou não, a entrada no sistema. Este pedido é utilizado pelo *decorator* que permite a entrada no sistema.

4.1.2. Role

O *Role,* é responsável pela gestão completa dos *Roles* dos utilizadores. Tirando partido do MongoDB, foi possível adicionar um *array* de utilizadores a cada documento Mongo associado a um determinado *Role,* abstraindo o resto do sistema dos mesmos. Esta abstração é importante por não comprometer as informações dos utilizadores, caso deixe de existir algum *Role* ou seja adicionado um novo. Este micro serviço contém três *Roles* inicialmente definidos e *"Customer"*, "*Merchant", "Admin".*

O *Role* "*Customer*" é atribuído a todos os clientes no momento de registo. Este é o *Role* mais básico do sistema permitindo apenas, caso esteja associado ao OBP API, efetuar pagamentos e gerir as contas bancárias.

O *Role* "*Merchant*", é atribuído aos diferentes comerciantes que utilizam o *Payment Widget*. Este role permite identificar a que comerciante pertence uma determinada compra e configurar as contas de recebimento das transações por parte dos comerciantes. Este *Role* apenas pode ser atribuído pelos "Administradores".

O *Role* "*Admin*" é o *role* que tem mais permissões no sistema, podendo gerir *Roles,* utilizadores, transações dos utilizadores, eliminar utilizadores, entre outros. Este *Role* é atribuído apenas a utilizadores que estejam registados como administradores, através de um pedido exclusivo para o registo.

Na figura 18, são ilustrados os diversos pedidos deste micro serviço.

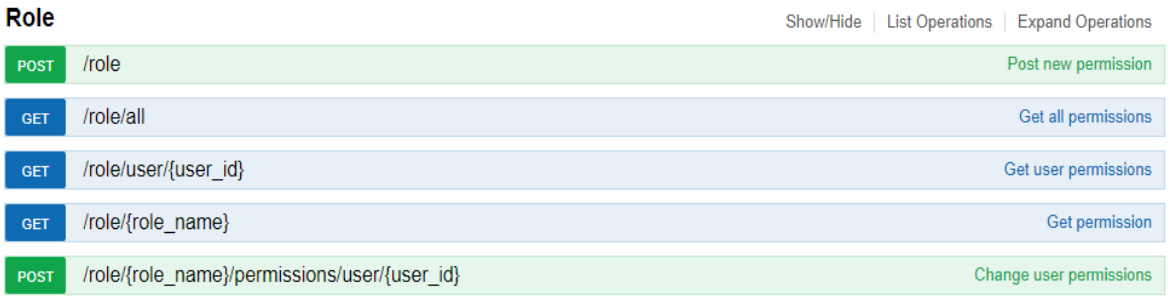

Figura 18 Pedidos disponibilizadas pelo Micro Serviço Role

Os pedidos POST ***/Role*** e **/Role/{Role_name}/permissions/user/{user_id},** permitem adicionar um novo *Role* e/ou atribuir um determinado *Role* a um determinado utilizador. Estes dois pedidos só podem ser acedidos por utilizadores que tenham o *Role* de "*Admin*".

No que diz respeito ao pedido GET **/Role/user/{user_id}**, este pedido retorna os *Roles* de um determinado utilizador, e é utilizado pelo *decorator* que faz a verificação dos mesmos. Os restantes pedidos permitem retornar todos, ou apenas um determinado *Role*.

4.1.3. User

O *User* agrega as tarefas relacionadas com o utilizador/cliente. É este micro serviço que gere as ações do utilizador, pelo próprio ou pelo administrador, e a associação no OBP API.

Na figura 19, são ilustrados os pedidos disponibilizados pelo *User*.

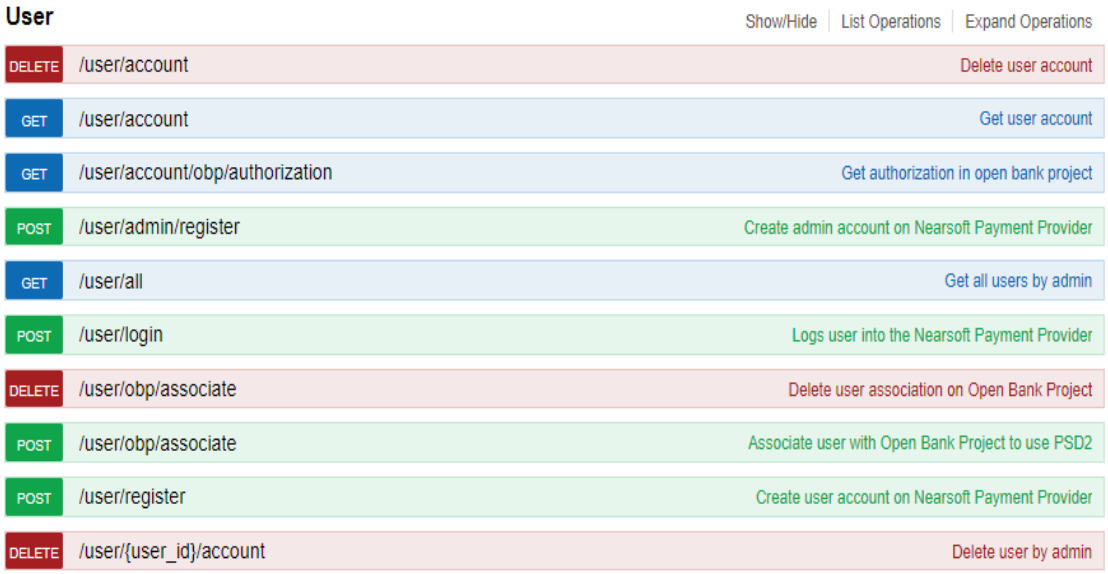

Figura 19 Pedidos disponibilizados pelo Micro Serviço User

O pedido POST ***/user/admin/register*** visa ao registo dos administradores no sistema.

O pedido e POST ***/user/register***, destina-se aos clientes e/ou *Merchants,* sendo que os últimos têm permissões adicionais, fornecidas pelos "*Admin*" através do *Role* de "*Merchant*".

O pedido POST ***/user/obp/associate*** refere-se à associação do cliente com o OBP API (*ver secção 3.4.3*). A realização do pedido DELETE ***/user/obp/associate***, faz com que o cliente deixe de ter a sua conta OBP API associada, não permitindo mais o acesso aos seus dados bancários por parte do NPP.

O pedido GET ***/user/account/obp/authorization***, tem como objetivo a verificação, por parte do *decorator* que verifica a autorização do OBP API*,* da autorização do utilizador para aceder ao OBP API. Caso o utilizador/cliente esteja associado a autorização é retornada.

Os pedidos DELETE ***/user/{user_id}*** e GET ***/user/all*,** são pedidos de gestão que apenas podem ser realizados por utilizadores que detenham o *Role* de "*Admin*". Estes pedidos, permitem eliminar um determinado utilizador e retornar a lista de todos os utilizadores registados no sistema.

Os pedidos GET e DELETE ***/user/account*****,** podem ser efetuados por qualquer utilizador do sistema e permitem retornar ou eliminar a própria conta.

4.1.4. AISP

Este micro serviço gere as informações das contas bancárias, possibilitando ainda a definição de uma conta de pagamento predefinida. O AISP utiliza diversos pedidos PSD2 do OBP API para apresentar ao utilizador os dados das suas contas bancárias. A base de dados deste micro serviço armazena os dados da conta de pagamentos predefinida pelo utilizador/cliente.

Na figura 20, pode ser verificado quais os pedidos possíveis de fazer ao micro serviço.

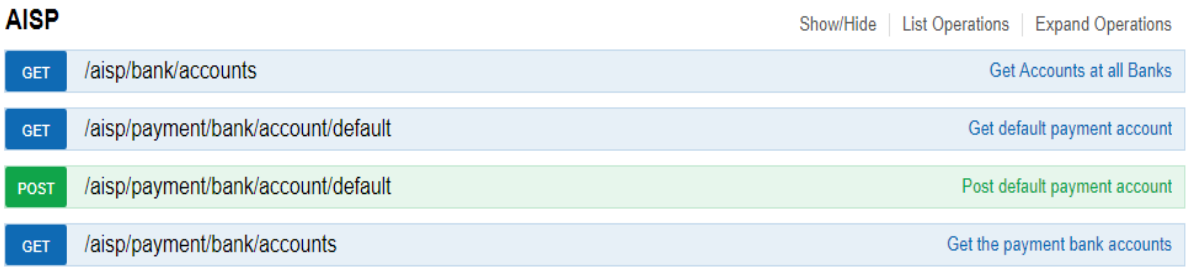

Figura 20 Pedidos disponibilizadas pelo Micro Serviço AISP

Os pedidos POST e GET ***/aisp/payment/bank/account/default***, têm como função adicionar/atualizar a conta de pagamentos predefinida e retornar a mesma, respetivamente.

O pedido GET ***/aisp/payment/bank/accounts,*** retorna todas as contas que têm fundos suficientes para a realização de um pagamento num determinado valor. Este pedido irá permitir apresentar, ao utilizador, uma alternativa no momento de pagamento, no caso de a conta de pagamentos predefinida não tiver fundos suficientes.

O pedido GET ***/aisp/bank/accounts***, por sua vez, agrega a informação de vários pedidos ao OBP API e retorna à informação detalhada de todas as contas do utilizador em todos os bancos.

4.1.5. PISP

No que diz respeito ao PISP, este micro serviço é responsável pelos pagamentos. É nele que são realizadas as três etapas fundamentais do pagamento, nomeadamente a apresentação da *charge*, a inicialização do pagamento e o *answer challenge* (*secções 2.1.2, 3.3 e 3.4.6 para mais informações sobre os três passos*). Na figura 21, é possível visualizar os pedidos disponibilizados por este micro serviço.

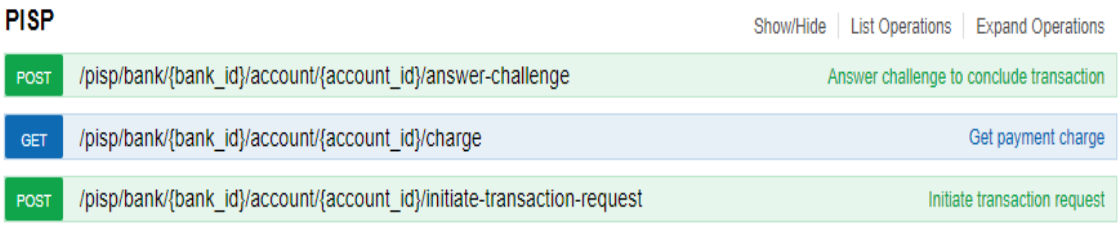

Figura 21 Pedidos disponibilizados pelo Micro Serviço PISP

Os pedidos deste micro serviço baseiam-se nas três etapas do pagamento.

A primeiro etapa é o pedido GET ***/pisp/bank/{bank_id}/account/{account_id}/charge***, que retorna o valor da *charge* correspondente a uma determinada conta bancária.

A segundo etapa, é a inicialização do pagamento através do pedidos POST ***/pisp/bank/{bank_id}/account/{account_id}/initiate-transaction-reques***.

A terceira e ultima etapa, que pode, ou não, existir (*ver secção 3.3.3 e 3.3.4*), corresponde ao *answer challenge* que é processado através do pedido POST ***/pisp/bank/{bank_id}/account/{account_id}/answer-challenge***.

4.1.6. Transactions

O micro serviço *Transactions* são responsáveis por gerir o armazenamento das transações, como compras, efetuadas pelo NPP.

Na figura 22, é possível verificar os pedidos que são disponibilizados por este micro serviço.

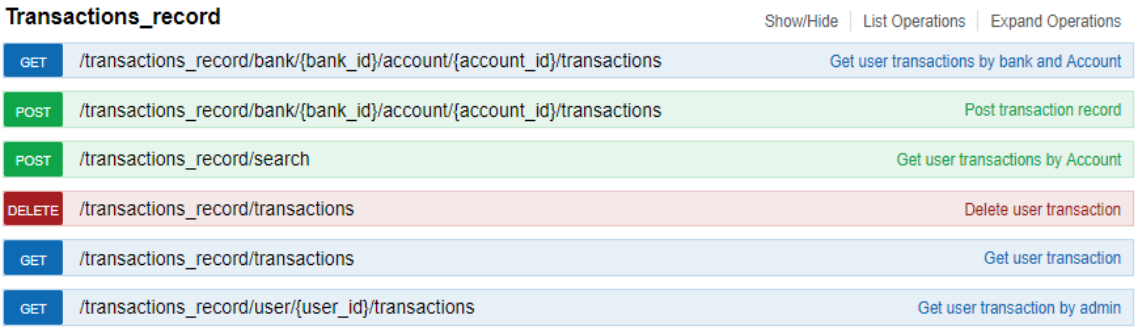

Figura 22 Pedidos disponibilizados pelo Micro Serviço Transactions

No que diz respeito ao pedido POST ***/transactions_record/bank/{bank_id}/account/{account_id}/Transactions,*** este pedido é realizado para efetuar o armazenamento das transações efetuadas pelo micro serviço PISP.

O pedido GET ***/transactions_record/bank/{bank_id}/account/{account_id}/transactions***, por sua vez, retorna todas as transações de um cliente de uma determinada conta.

Os pedidos GET e DELETE ***/transactions_record/transactions,*** são responsáveis por retornar e eliminar todas as transações. Além destes pedidos, é também possível retornar todas as transações

de um cliente através do pedido GET ***/transactions_record/user/{user_id}/transactions***. No entanto, este pedido apenas pode ser efetuado por utilizadores que tenham o *Role* de "*Admin*".

O pedido POST ***/transactions_record/search***, por sua vez, retorna as transações filtradas por parâmetros de pesquisa, que podem existir isoladamente ou em simultâneo.

4.1.7. Merchant

O micro serviço *Merchant* foi criado com intuito de gerir as informações dos vários comerciantes que possam existir. Dessas informações, destacam-se a conta bancária, que deverá receber os fundos correspondentes aos produtos vendidos e a KEY de identificação do *Merchant*.

Os pedidos disponibilizados por este micro serviço podem ser visualizados na figura 23.

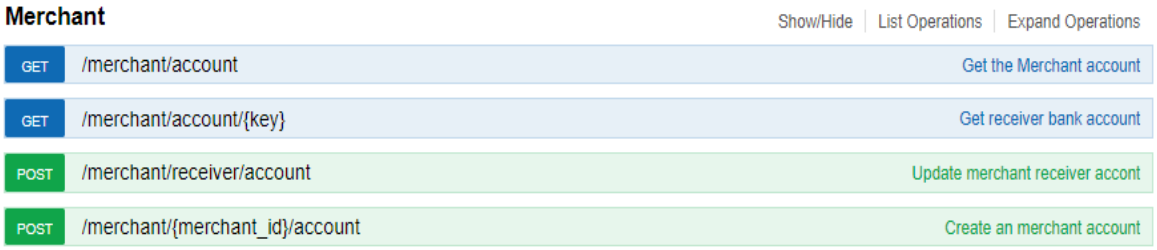

Figura 23 Pedidos disponibilizados pelo Micro Serviço Merchant

O pedido POST ***/merchant/receiver/account,*** corresponde ao pedido de criação do *Merchant*. Este pedido apenas pode ser efectuado por utilizadores que tenham o *Role* de "*Admin*".

O pedido GET ***/merchant/account,*** permite ao *Merchant* obter os dados da sua conta. Este retorno de dados da conta do *Merchant* pode ser igualmente conseguido através do pedido GET ***/merchant/account/{key}***. Contudo, este pedido pode ser realizado também por clientes devidamente autenticados.

O pedido POST ***/merchant/receiver/account***, por fim, permite ao *Merchant* atualizar a conta bancária de recebimento dos pagamentos.

4.2. NearSoft Payment Provider (NPP) APP

Tendo em conta grande parte da lógica do NPP APP, que já foi abordada na secção 3.4, esta secção apenas fará referência dos aspetos em falta e apresentará a interface gráfica das partes mais importantes da aplicação.

As primeiras etapas a serem efetuadas por um cliente que queira utilizar os serviços do NPP, consistem no registo e no login. Essas duas etapas podem ser visualizadas no **Anexo C**, figura 35 e 36.

Considerando que o cliente/utilizador está registado no OBP e tem, pelo menos, uma conta bancária, caso este pretenda realizar pagamentos de compras online utilizando o NPP, basta preencher um pequeno formulário de associação (*ver secção 3.4.3*). Na figura 24, é possível ter uma perceção da view de associação.

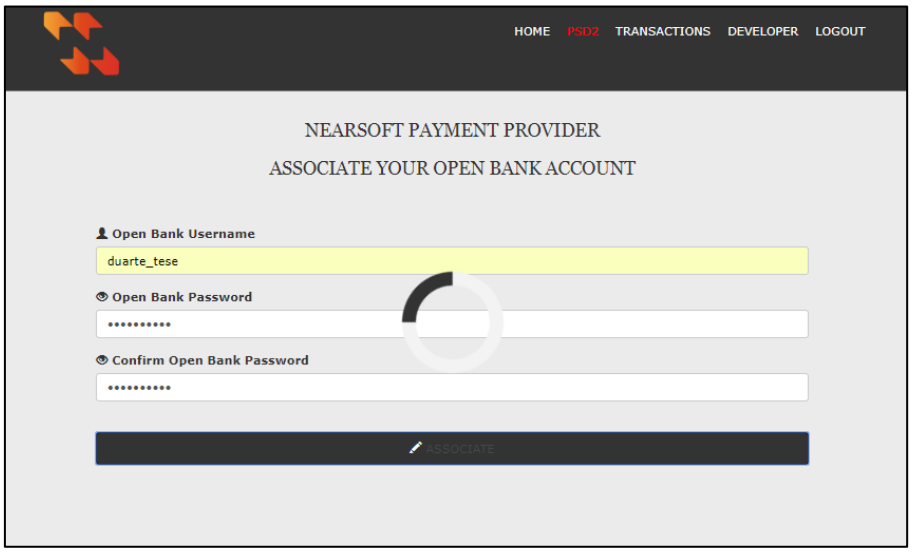

Figura 24 View de associação da conta NPP com a conta OBP

Caso o utilizador já se encontre associado, este pode escolher/alterar a conta de pagamentos predefinida.

Na figura 25, é possível visualizar *pop-up* que possibilita a escolha/alteração da conta de pagamento predefinida.

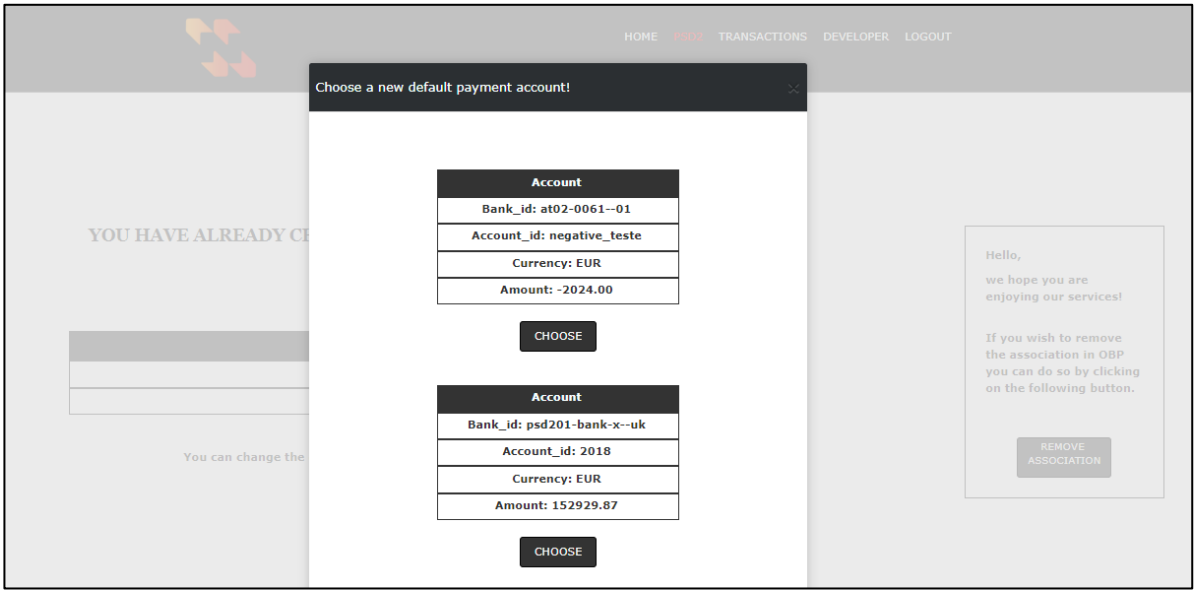

Figura 25 View com modal de escolha da conta de pagamento predefinida

Considerando a hipótese de que o utilizador já tenha escolhido a conta de pagamento predefinida, é-lhe apresentada uma *view* com os dados da conta escolhida; um botão "*Change*" com a possibilidade de alterar a conta. Além disso, existe a possibilidade de remover a sua associação no OBP API, através do botão "*Remove Association*".

Na figura 26 pode ser observada a *view* referida.

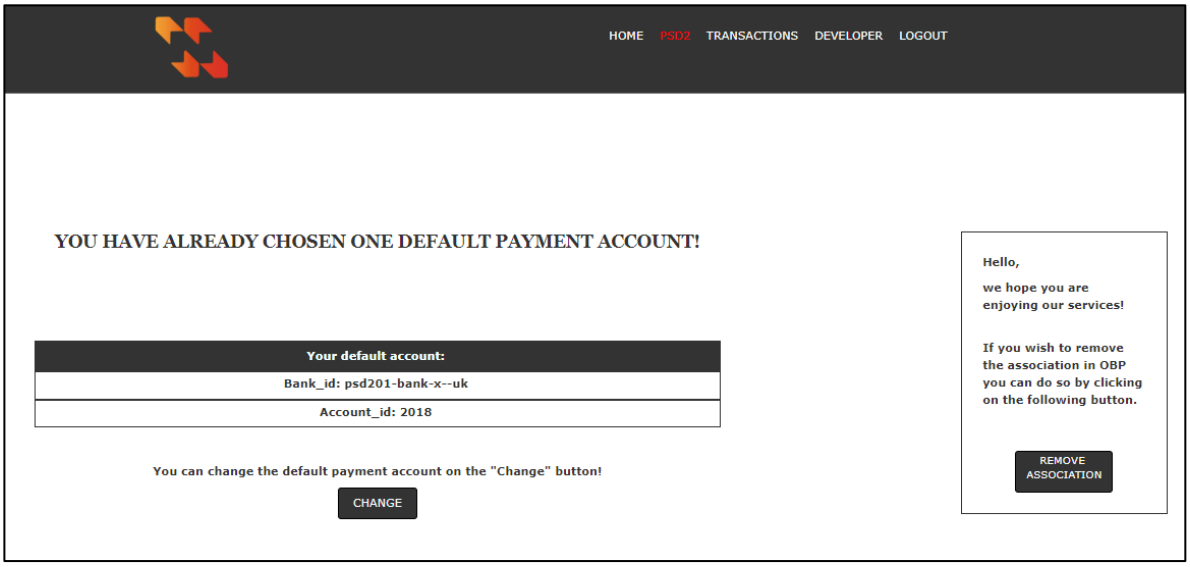

Figura 26 View com a conta de pagamentos predefinida

Quando pretendido, o utilizador pode consultar as suas transações realizadas. Esta consulta é feita por conta bancária, com o intuito de o utilizador adquirir uma melhor perceção dos seus gastos nessa determina conta.

Na figura 27, é possível visualizar um exemplo do histórico de transações realizadas. Existe ainda uma

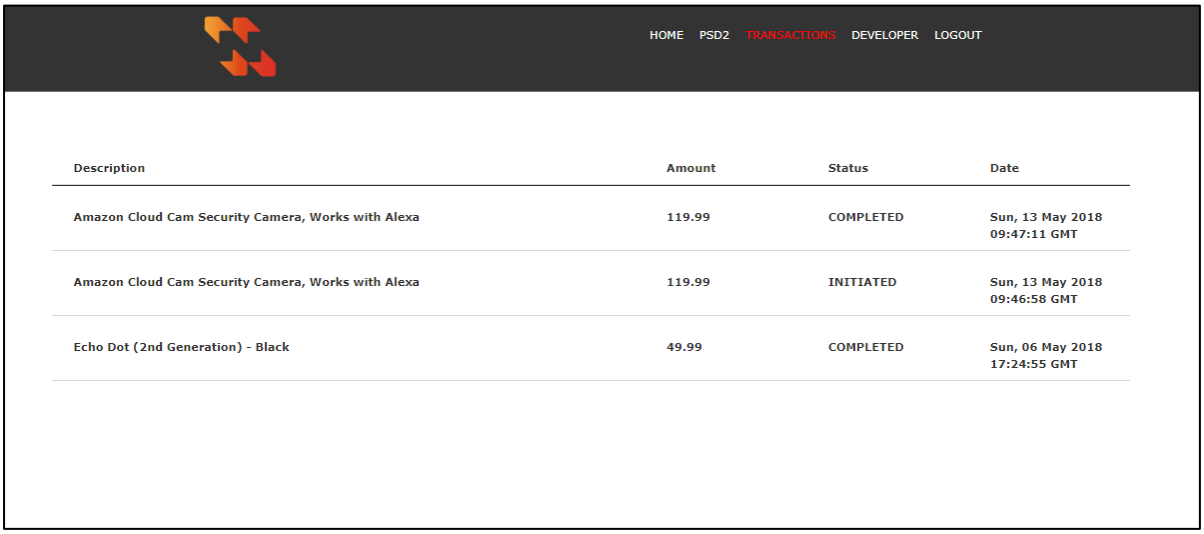

Figura 27 View com o registo das transações realizadas pelo NPP

secção dedicada às informações de desenvolvimento. Nesta página, é apresentada uma hiperligação para a documentação das APIs dos diversos micro serviços. Desta forma, torna-se possível que qualquer pessoa, que queira desenvolver a sua própria aplicação, possa utilizar a API desenvolvida.

4.3. Payment Widget

O *Payment Widget* consiste em dois botões de pagamentos, que realizam o pagamento de um ou vários produtos, utilizando o PSD2 e com a menor fricção possível. Para testar o *Payment Widget* foi

desenvolvido um website, que simula uma aplicação e-*commerce,* utilizando a *framework* de PHP, Laravel [78].

Neste capítulo, iremos apresentar algumas *views* do website desenvolvido, sendo, contudo, o principal objetivo compreender a forma como o *Payment Widget* pode ser utilizado numa aplicação de *e-commerce*.

Quanto ao site de *e-commerce,* este é uma aplicação, onde existe uma lista de produtos que podem ser comprados pelo utilizador.

Na figura 28, é possível ter uma perceção da *view* onde estão dispostos os diversos produtos.

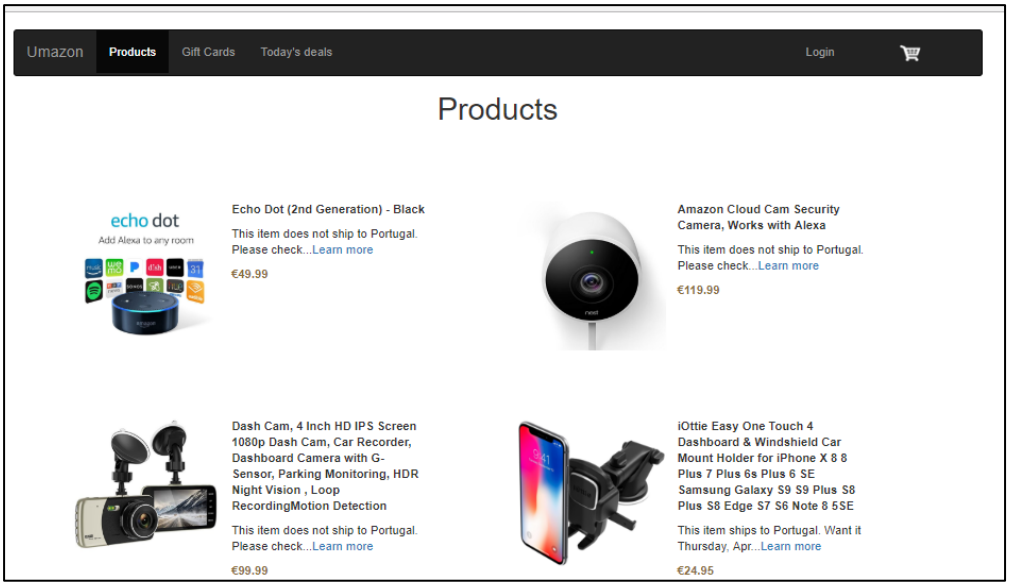

Figura 28 View com a lista de produtos

Caso o utilizador tenha interesse num determinado produto, e pretenda mais informação acerca do mesmo é apresentada uma *view* com informação mais detalhada como é possível observar na figura 29.

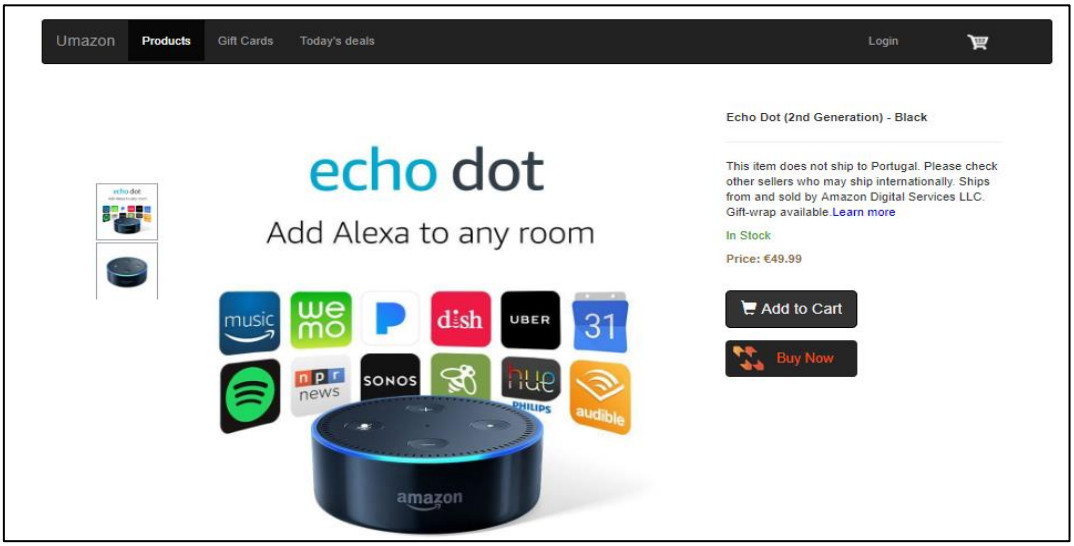

Figura 29 View com o produto detalhado

Nesta *view*, são apresentadas duas possibilidades ao utilizador, sendo a de adicionar o produto ao carrinho procedendo à sua compra mais tarde, e a de comprar no momento com o botão "*Buy Now*" desenvolvido por nós e embebido neste site. Caso o utilizador/cliente escolha a opção "*Buy Now*", o pagamento é realizado seguindo o fluxo de pagamentos abordado na secção 3.4.6.

Por outro lado, se o utilizador decidir aceder às compras do carrinho, é-lhe apresentada a *view* ilustrada na figura 30.

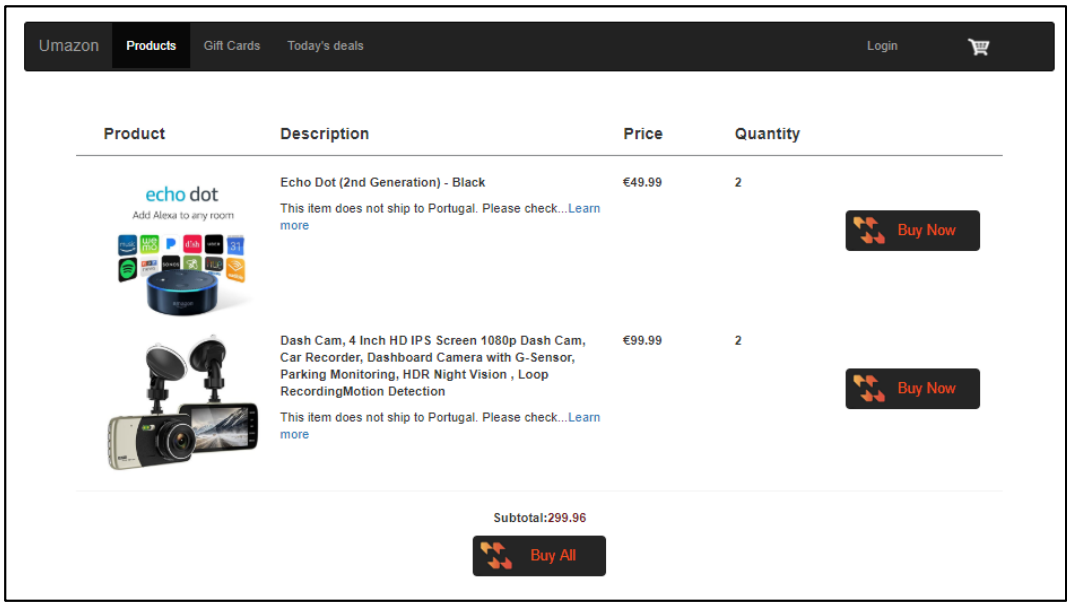

Figura 30 View de apresentação do carrinho de compras

Neste cenário, o utilizador tem a opção de proceder ao pagamento dos produtos individualmente, clicando no botão "*Buy Now*". Se a quantidade for maior do que um, o fluxo de pagamento (*ver secção 3.4.6*) é repetido por essa quantidade, sendo necessário que o utilizador faça login apenas uma única vez. Por exemplo, se o utilizador/cliente tiver dois produtos no carrinho, o fluxo de pagamento será repetido duas vezes e o login é realizado apenas um única vez.

Se o utilizar pretender realizar a compra total dos produtos, basta clicar botão de pagamentos “*Buy All*”, e as compras serão efetuadas. Neste caso, para cada produto é executado o fluxo semelhante ao fluxo do botão “Bay Now”, explicado acima, sendo necessário realizar login apenas uma vez.

4.3.1. Etapas do Pagamento

Quando um dos botões referidos anteriormente é despoletado, a compra é automaticamente inicializada. Uma das etapas mais importantes no processo da compra é a verificação da autorização, que é feita através de um login. O formulário de login consiste num *pop-up* que se abre automaticamente, como é possível observar na *view* da figura 31.

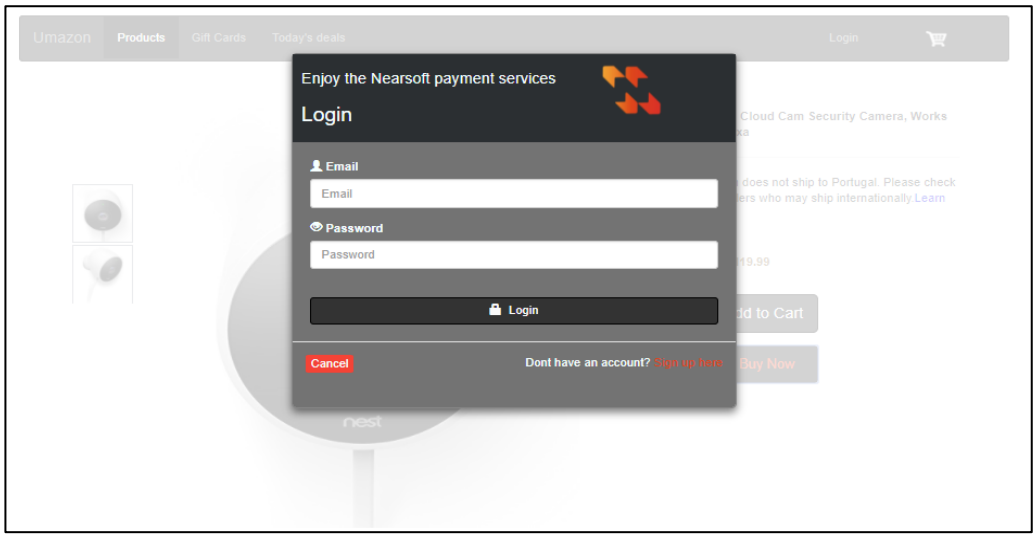

Figura 31 Login no NPP

Ao longo do processo de pagamento, quando a conta bancária predefinida não tem fundos suficientes, é necessário dar a conhecer ao cliente as informações da confirmação de uma conta bancária alternativa à predefinida, caso exista. Além da conta predefinida é necessário apresentar a *charge*, e o *challenge*, quando necessário (*ver secção 3.3.4*). Para tal aconteça, foi implementado um *pop-up* de confirmação. Na *view* apresentada na figura 32, é possível visualizar um exemplo do *pop-up* de confirmação.

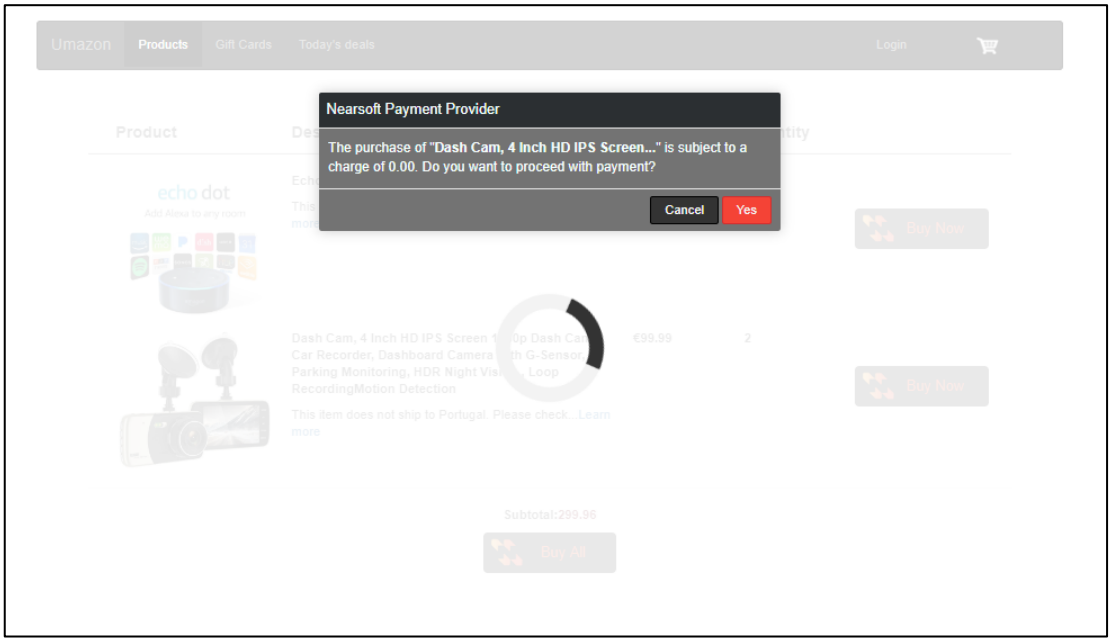

Figura 32 Confirm dialog de aceitação da charge

Aquando da conclusão/cancelamento da compra, ou falta de fundos para a realização da mesma, o utilizador é informado através de um *pop-up* de alerta. Este *pop-up* de alerta é apresentado durante 4 segundos desaparecendo depois automaticamente.

Na figura 33, encontra-se à ilustração do referido *pop-up* de alerta.

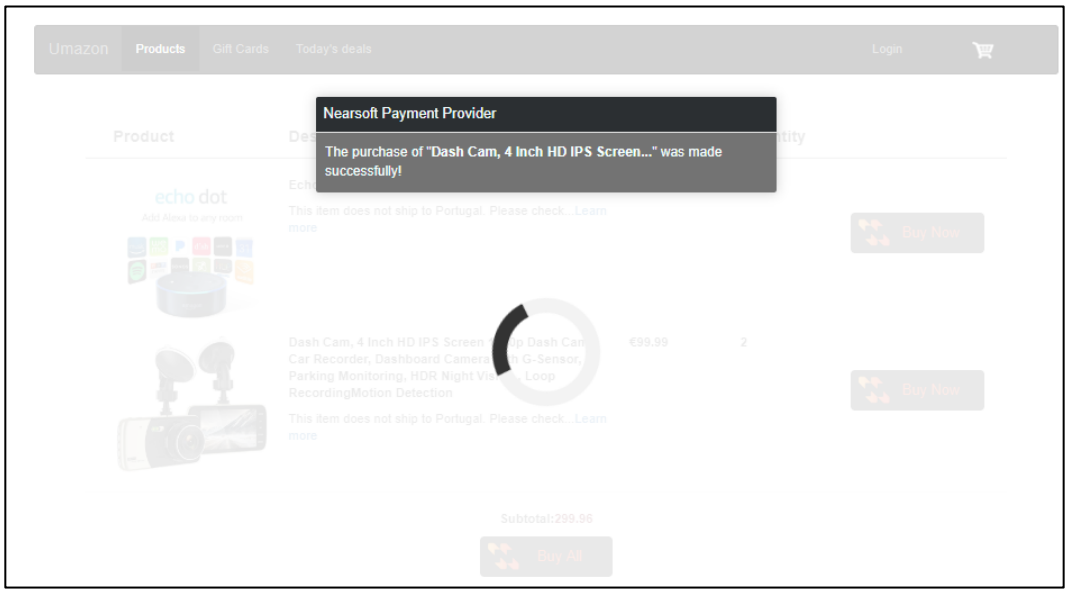

Figura 33 Alert dialog a informar o sucesso da transação

Os *pop-up* de login, confirmação, alerta e o *loader* estão integrados com o botão de pagamento.

4.3.2. Integração do *Payment Widget*

No que diz respeito à integração do *Payment Widget,* foi tentado, ao máximo, diminuir o esforço da mesma. O *Payment Widget* é composto por sete scripts, em que um deles, o **buy_buttons.js**, corresponde aos botões "*Buy Now*" e "*Buy All*", que são implementados em JavaScript, e que englobam toda a lógica do pagamento em si. Foi criado um ficheiro de configuração (**config.js**), onde é possível definir/alterar alguns parâmetros de configuração.

Cinco ficheiros CSS são utilizados para formatar a apresentação dos componentes, nomeadamente os botões (**buttons_style.css**), *pop-up* de login (**login_modal.css**), *pop-up* de confirmação (**confirm_box.js**), *pop-up* de alerta (**alert_box.js**) e ainda o *loader* (**loader.css**).

É de salientar que o *Payment Widget* vem integrado com todos os componentes referidos anteriormente, para dar feedback ao utilizador a cerca do progresso de transação.

Os botões de pagamento "*Buy Now*" e "*Buy All*", têm apenas duas diferenças entre si – o nome e o local onde são colocados. Todo o resto é partilhado entre os dois, incluído a implementação para os dois botões. Para que os botões de pagamento funcionem devidamente, existem alguns requisitos que têm de ser satisfeitos. Estes requisitos são pequenas mudanças necessárias no ficheiro de configuração, nos métodos de "retorno" da compra e na nomenclatura dos parâmetros do formulário. Os requisitos necessários à integração referidos anteriormente são:

- Cada produto deverá conter um formulário, com *inputs,* com os dados dos seguintes parâmetros*: amount, currency, product_name, product_id e quantity.* O *amount* e a *currency,* como o próprio nome indica, correspondem ao valor e à moeda do produto. O *product_name,* refere-se ao nome do produto. O *product_id* corresponde ao ID do produto, que pode se tornar necessário para retirar o produto do carrinho. Por fim, o parâmetro *quantity* deverá indicar a quantidade do produto que é pretendido ser comprado. Por *default,* esse valor poderá ser definido como 1 correspondendo a uma única unidade.
- A classe do formulário dos produtos, deve ser comum ao parâmetro **product_form** do ficheiro de configuração (**config.js**). Este requisito é importante para embeber o botão "*Buy Now*" no produto, e para aceder aos *inputs* do formulário, quando o botão "*Buy Now*" é despoletado. A classe do formulário do produto, é ainda utilizada para aceder aos dados de todos os formulários, quando o botão "*Buy All*" é despoletado, uma vez que este recolhe dados de todos os formulários.
- Ao botão "*Buy All*", e uma vez que não estará presente em qualquer formulário, é necessário definir a classe da onde é pretendido embeber o botão. Essa classe também deve ser definida no ficheiro de configuração.
- A responsabilidade de retirar os produtos do carrinho, após a compra da mesma, deverá ser da aplicação *e-commerce*. Para tal, basta implementar o método *cart_remove*, localizado na parte inicial do script **buy_buttons.js***,* que recebe o objeto do produto que foi comprado, *product_purchased,* como parâmetro no seguinte formato:

```
{
    amount: "49.99",
    currency: "EUR",
    product_name: "Echo Dot (2nd Generation) - Black",
    product_id: "1"
}
```

- Após finalizada a compra, quer de um produto individual quer do carrinho completo, é também da responsabilidade do site de *e-commerce* atualizar o estado da página. Essa alteração pode também ser feita implementando o método *return_from_purchase(),* presente igualmente na parte inicial do script **buy_buttons.js.**
- Os ficheiros **.css** são adicionados à página através do script **buy_buttons.js.** No entanto, estes ficheiros têm de fazer parte do projeto. Dessa forma, o *path* dos ficheiros, que deverá ser comum a todos os ficheiros **.css**, tem de ser adicionado no ficheiro de configuração **config.js.**
- A KEY que identifica o *Merchant,* também deve se encontrar definida num parâmetro do ficheiro de configuração **`MerchantKey`**, que se encontra a *null* por predefinição.
- O script **buy_buttons.js** deve ser adicionado no *header* de todas as páginas que necessitem conter um dos botões embebidos.

4.4. Conclusão

No capítulo 3, foram tomadas várias decisões importantes relativas à metodologia que iria ser utilizada no projeto. Foram, ainda, tidos em consideração diversos procedimentos importantes, desde o levantamento de requisitos ao estudo da API PSD2 e à proposta de solução definida. Esses passos foram fundamentais para a implementação, tema abordado no presente capítulo.

A metodologia, consistiu em encontrar uma abordagem de engenharia para desenvolver a proposta de solução. A nossa abordagem consistiu num *Back-End* completamente isolado do *Front-End*. Para o *Back-End* ao qual intitulamos de *NearSoft Payment Provider API* (*ver secção 4.1*), foram desenvolvidos vários micro serviços em Python, utilizando a *microframework* Flask [77].

No que diz respeito ao *Front-End,* este encontra-se dividido em dois módulos – *NearSoft Payment Provider APP* (*ver secção 4.2*) e *Payment Widget* (*ver secção 4.3*). A *NearSoft Payment Provider APP*, consiste no local onde os clientes, que pretendam usufruir dos serviços de pagamento e de informação de conta, podem gerir as suas informações. O *Payment Widget,* por sua vez, consiste numa espécie de *Plug-in*[21], que contém botões de pagamento de acordo com o PSD2. Este *Plug-in* encontra-se implementado em JavaScript, e pode ser embebido em qualquer aplicação *e-commerce*. Uma vez implementados os componentes, torna-se fundamental incluir a fase de avaliação. Esta fase permitirá verificar os requisitos mais importantes e testar a implementação das funcionalidades. Por este ser um procedimento com um elevado grau de importância neste tipo de projetos, o capítulo 5 será dedicado à abordagem do mesmo.

[21] Plug-in - Segmento de software que permite aumentar as funcionalidades do software existente. https://www.computerhope.com/jargon/p/plugin.htm

5. Avaliação

A avaliação é um processo fundamental neste tipo de projeto, pois permite verificar se os requisitos foram cumpridos. Os requisitos de segurança, usabilidade e integrabilidade, foram considerados os requisitos não funcionais mais importantes deste projeto.

A nível da segurança, este requisito é um dos mais comuns nos sistemas de pagamento online. Neste projeto, e por opção da empresa NearSoft, não foi dado muito enfase à segurança, uma vez que o principal objetivo era explorar a norma PSD2 e verificar a facilidade da sua implementação. Contudo, para que o sistema não ficasse demasiado vulnerável, foi optada pela utilização dos protocolos de SSL, sistema de autenticação JWT[22] e a utilização de um mecanismo de segurança nas bases de dados, como referido no capítulo 4.

Aos requisitos de usabilidade e facilidade de integração foi dado um foco especial com o intuito de avaliar a utilização dos sistemas de pagamento que suportem o PSD2, e se os mesmos são fácies de integrar nas aplicações de *e-commerce*.

As próximas secções são destinadas à abordagem detalhada da avaliação realizada para estes requisitos, e dos os resultados obtidos.

5.1. Usabilidade

A avaliação da usabilidade teve como principal objetivo, determinar a percetibilidade do sistema a facilidade de aprendizagem, e de operação, e o nível de atração do mesmo para os utilizadores. Dessa forma, foi avaliada a interface do sistema, e outros fatores humanos que afetam a maneira como as pessoas utilizarão o sistema, garantindo sempre que o design (layout e sequência, etc.) permita que as funções de negócios sejam executadas da maneira mais fácil e intuitiva possível [79].

A primeira etapa realizada nesta avaliação, consistiu na escolha daqueles que seriam os possíveis utilizadores do sistema. Esta escolha, recaiu maioritariamente para utilizadores que pertencem a faixa etária de entre 20 e 40 anos, e que utilizam frequentemente equipamentos tecnológicos, tendo mesmo a maioria já realizado compras online. Para este tipo de teste, foi possível a recruta de oito avaliadores, que serão identificadas de P1 a P8.

Nas próximas secções é possível encontrar uma descrição sobre em que consistiu esta avaliação, e quais os resultados obtidos.

5.1.1. Teste

O teste de usabilidade foi realizado individualmente com os participantes.

[22] JWT (JSON Web Tokens) - Forma compacta e segura de enviar informação através da web https://jwt.io/

No início, foi feita uma breve descrição do projeto, assim como uma referência do PSD2. A segunda etapa, consistiu numa apresentação da plataforma, ou seja, do NPP APP e do *Payment Widget (ver secção 4.2 e 4.3)*. Após esta introdução, foi então discutido o procedimento do teste, ou seja, foi explicado que seriam fornecidas certas tarefas aos utilizadores que, por sua vez, deveriam realiza-las seguindo a abordagem tradicional *"Think Aloud*" [80].

As tarefas foram escolhidas de forma a testar as operações mais importantes, nomeadamente:

1. Aceder à plataforma NPP e fazer um registo na mesma.
2. Associar na plataforma NPP, a conta OBP fornecida.
3. Escolher de entre duas contas bancárias criadas, uma conta de pagamentos predefinida (*default*).
4. Aceder à Umazon e realizar a compra de um produto que custasse mais de 100€.
5. Fazer uma nova compra de qualquer outro produto.
6. Consultar o histórico das compras.

No que diz respeito à realização das compras, foi determinado propositadamente que, aquando da realização da segunda compra, os participantes não tivessem fundos suficientes na conta de pagamentos predefinida. Esta decisão, justifica-se com a necessidade de fazer aparecer a sugestão de uma conta alternativa no momento do pagamento.

Quanto à realização das tarefas, e enquanto o participante realizava cada uma delas, foram anotados os aspetos mais relevantes como a contabilização do tempo e possíveis erros cometidos. No fim de cada tarefa, foi debatido, juntamente com os participantes, os aspetos que poderiam ser melhorados, a avaliação da aparência da interface e a facilidade de realização do teste.

Após a conclusão de todas as tarefas, foi também solicitado ao participante que realizasse o questionário relacionado com *System Usability Scale* (SUS).

O SUS, é um dos questionários mais conhecidos utilizados em UX. Este tipo de questionários deve ser dado a realizar aos avaliadores, após o fim da sessão de testes [81]. Este questionário é composto por 10 afirmações, às quais os participantes devêm atribuir uma avaliação que varia de 1 a 5, sendo 1 correspondente a discordo completamente e 5 a concordo plenamente.

As afirmações do SUS seguem-se na seguinte lista:

1. Eu acho que eu gostaria de usar este sistema com frequência.
2. Achei o sistema desnecessariamente complexo.
3. Eu pensei que o sistema fosse fácil de usar.
4. Eu acho que irei necessitar do apoio de um técnico para poder usar este sistema.
5. Eu encontrei as várias funções neste sistema que foram bem integradas.
6. Eu pensei que havia muita inconsistência neste sistema.
7. Eu imagino que a maioria das pessoas aprenderia a usar este sistema muito rapidamente.
8. Eu encontrei este sistema muito complicado de usar.
9. Eu me senti muito confiante usando este sistema.

10. Eu precisava aprender um monte de coisas antes que eu pudesse utilizar este sistema de forma fácil.

O enunciado, entregue aos avaliadores, assim como as informações recolhidas e o questionário de SUS, está presente no **Anexo A**.

5.1.2. Resultados

Tendo em conta o tempo de realização das tarefas, o gráfico 1 dispõe o tempo médio, em minutos, para a realização de cada uma delas.

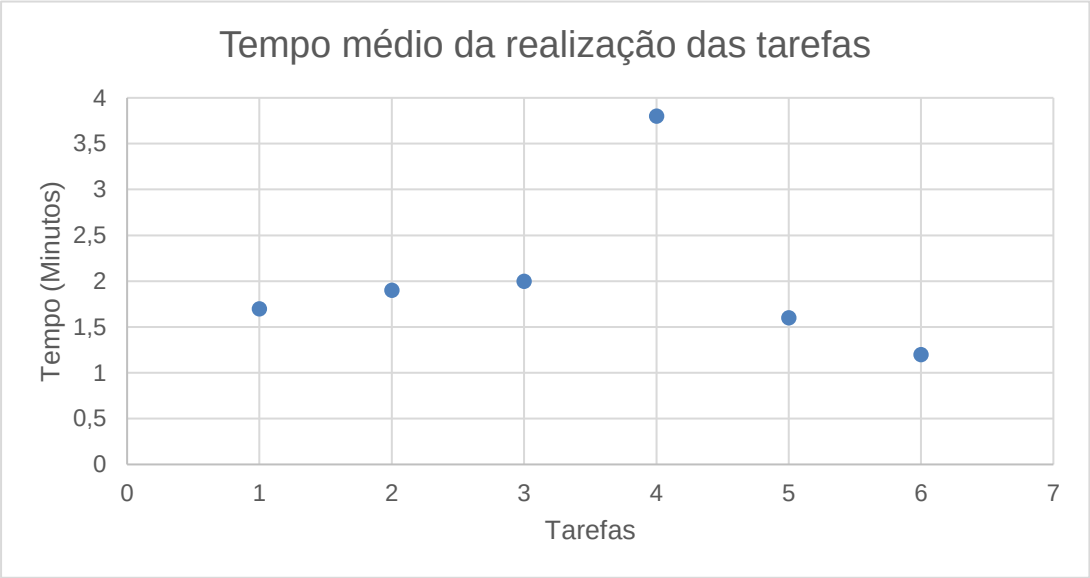

Gráfico 1 Tempo médio necessário para realização das tarefas

Através do gráfico 1, podemos inferir que em média as tarefas foram realizadas de forma rápida. Se considerarmos uma tarefa agregada, em que um utilizador sem conta no NPP pretende realizar um pagamento com o mesmo, o tempo de realização da mesma pode ser obtido através da soma do tempo das 4 primeiras tarefas. Dessa forma, arredondando o tempo de execução das tarefas as unidades, temos um tempo total de 13 minutos (3+3+3+4) no pior dos caso.

No que diz respeito à facilidade e ao aspeto da interface, a média das avaliações, para cada uma das tarefas, numa escala de 1 a 10, sendo 1 correspondente a pouco fácil e/ou com um mau aspeto, e 10 a fácil e/ou com bom aspeto, pode ser visualizada no gráfico 2.

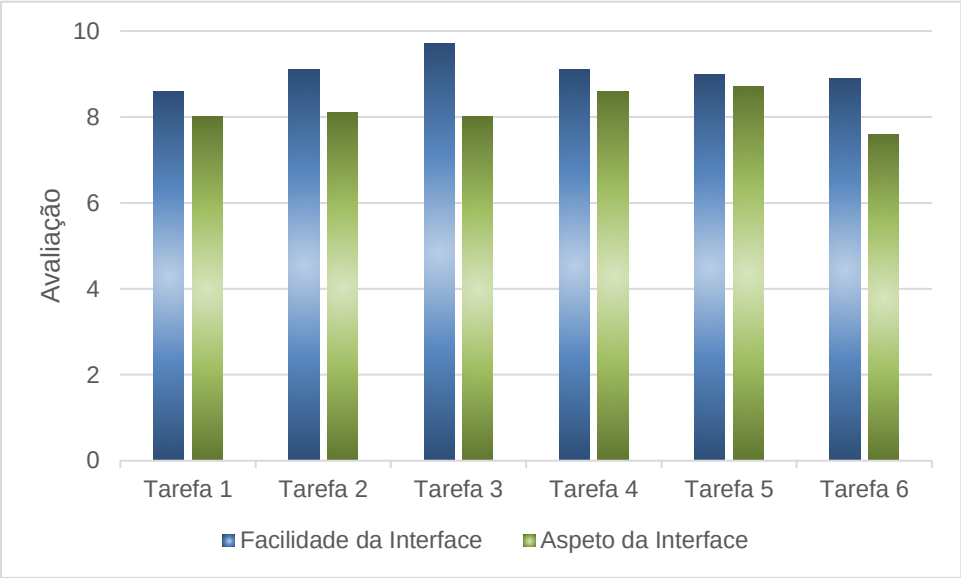

Gráfico 2 Classificação da facilidade e aspeto da interface

Os resultados para estes dois aspetos da interface foram muito positivos, ficando, na maior parte dos casos, acima dos oito pontos. Contudo, e observando o gráfico com alguma atenção, podemos verificar que a facilidade da interface obteve sempre uma melhor média. Esta tendência pode ser corroborada com a observação e diálogo com os avaliadores, em que estes mostraram muito agrado na facilidade de operar do sistema, deixando, no entanto, ciente que o aspeto da interface poderia ser melhorado, principalmente no que diz respeito ao preenchimento dos espaços vazios da interface da aplicação.

Através da observação e do diálogo, foi ainda possível tirar outras conclusões muito importantes de aspetos que poderiam ser melhorados.

A título de exemplo, na tarefa 1 os utilizadores, em vez de tentarem fazer um registo no NPP, tentaram realizar o login com as credenciais do OBP, que eram necessárias na tarefa 2, para a associação da conta NPP com a conta OBP.

Na tarefa 2, um dos utilizadores recomendou utilizar apenas um campo para a *password* em detrimento do uso de dois. Outra observação relativa a esta tarefa refere-se com o facto de os participantes não perceberem, de todo, qual seria o objetivo da associação da conta OBP.

Quanto à tarefa 3, alguns avaliadores sugeriram que, aquando da apresentação da conta predefinida de pagamentos, deveriam ser apresentados mais dados informativos acerca da conta bancária (P2: "*Podiam ser apresentados todos os dados da conta de pagamentos predefinida.*").

Analisando a tarefa 4, o resultado obtido foi muito bom, tendo os avaliadores gostado da nova forma de realizar pagamentos. No entanto, foram sugeridas ainda algumas recomendações como, por exemplo, a existência de outros idiomas à escolha.

A tarefa 5, por sua vez, revelou-se um fator surpresa para os avaliadores. Isto pode ser explicado tendo em conta a tarefa 4, em que estes tiveram de realizar uma compra no valor de 100€, usando uma conta bancária com saldo contabilístico de 120€. Ao procederem a uma segunda compra, de valor superior a

20€, valor atual do saldo, a conta predefinida de pagamentos deixa de estar disponível por não conter os fundos suficientes. Assim, o sistema, em vez de invalidar ou cancelar a compra, levando o utilizador a ter de definir uma nova conta de pagamentos, sugere uma conta bancária alternativa de pagamento (caso existente), permitindo, desta forma, a continuação da compra. Esta funcionalidade fez com que a tarefa recebesse críticas bastante positivas, tendo apenas sido recomendada uma apresentação mais intuitiva da conta de pagamentos predefinida.

Por fim, no que diz respeito à tarefa 6, esta tarefa foi a que mais recomendações obteve. Estas centram-se no facto de ser necessário proceder à escolha de uma conta antes de ser possível visualizar as transações, fazendo com que se torne complicado, caso o utilizador queira consultar transações de diversas contas. Outras recomendações remetem para a adicionação de um *sidebar* com filtros, para que o utilizador possa visualizar as transações que pretende, independentemente da conta e sem sair da *view* (P5: *"Deveria estar disponível um filtro de pesquisa."*).

5.1.2.1. System Usability Scale

Em termos do *SUS*, através do gráfico 3 podemos aferir a média das avaliações, numa escala de 1 a 5, atribuídas pelos avaliados.

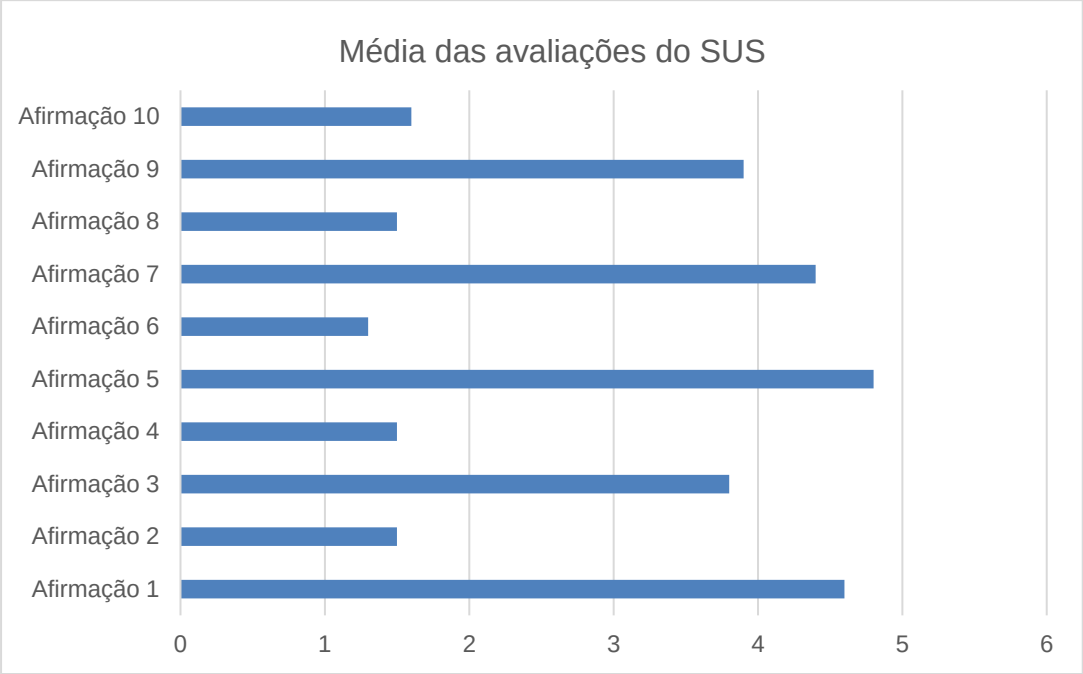

Gráfico 3 Resultados da avaliação SUS

Uma vez que as Afirmações se encontram escritas através de fases positivas e negativas, é impossível obter uma classificação linear em que 1 seja o pior nível da escala e 5, o melhor. Dessa forma, de modo a facilitar a interpretação dos resultados, foram apresentados, no gráfico 3 as médias de cada uma das Afirmações.

Os resultados desta forma demostram que os participantes utilizariam o sistema (Afirmação 1). A complexidade do sistema é facilmente ultrapassável (Afirmação 2). No que diz respeito à facilidade de

uso (Afirmação 4), os utilizadores consideram o sistema fácil de utilizar, não sendo necessário o apoio técnico para a utilização (Afirmação 5). Focando-nos na integração do sistema (Afirmação 5), a maioria dos utilizadores classificou esta componente com a escala máxima, considerando, dessa forma, o sistema bem integrado e sem inconsistências (Afirmação 6). Os avaliadores classificaram, ainda, o sistema como fácil de aprender (Afirmação 7) e fácil de utilizar (Afirmação 8). Quanto à confiança na utilização do sistema (Afirmação 9), pode ser inferido que os avaliadores se sentiriam confiantes. Por fim, os utilizadores acham que não necessitariam de aprender muitas coisas para utilizarem o sistema (Afirmação 10).

De modo a ser possível obter uma avaliação mais concreta, foi decido aplicar o algoritmo de interpretação de resultados do SUS. Para tal, a média das avaliações ímpares foi subtraída por 1 unidade. Quanto às avaliações pares, estas foram subtraídas a um valor constante de 5. Após subtraídas devidamente as unidades, somou-se as pontuações de cada uma das questões e multiplicou-se por 2.5.

O resultado de todas as operações tem um valor máximo de 100. Se o resultado obtido for inferior a 68, considera-se que o sistema tem problemas de usabilidade. Por outro lado, se o resultado for superior a 68, o sistema é considerado “fácil de utilizar”. Caso o resultado ultrapasse 80,3, o sistema é tido como “muito fácil de utilizar” [82].

Na função que se segue é possível visualizar a aplicação do algoritmo e o seu resultado.

$$SUS = ((4,6-1) + (5-1,5) + (3,8-1) + (5-1,5) + (4,8-1) + (5-1,3) + (4,4-1) + (5-1,5) + (3,9-1) + (5-1,6)) * 2,5$$

$$SUS = 85, 25$$

Tendo em conta o valor obtido, podemos concluir que o sistema está muito fácil de utilizar. Obtendo uma classificação A (a maior) da escala do SUS [82].

5.1.2.2. Erros Obtidos

Em relação à existência de erros no sistema, não foram encontrados quaisquer erros de código, tendo os micro serviços mantido a sua execução durante todos os testes.

No que diz respeito a erros de funcionamento, houve um caso em que um dos avaliadores falhou nos dados de associação e o *loader* não parou a sua execução. Ainda em relação ao *loader*, no momento do registo e do login, este não se iniciou.

5.2. Facilidade de Integração

O principal objetivo da avaliação da facilidade de integração, consistiram na verificação da facilidade com que o *Payment Widget* (*ver secção 4.3*) pode ser integrado na aplicação de *e-commerce.*

Para a realização deste teste, foi necessário recrutar profissionais com algum conhecimento na área de desenvolvimento de software. Encontrou-se dessa forma dois desenvolvedores juniores, recém-

licenciados em Engenharia Informática, com pouca experiência. Os participantes serão identificados como P1 e P2.

5.2.1. Teste

O objetivo deste teste foi verificar o requisito não funcional – "Facilidade de Integração" –, verificando o grau de facilidade da integração do *Payment Widget* numa aplicação de *e-commerce.* Para tal, foram fornecidos aos avaliadores dois elementos de software*,* um site de *e-commerce* com um botão de pagamento tradicional e o *Payment Widget*.

Num momento inicial, foi explicado o código fonte do *site* de *e-commerce* pois, num ambiente real, os desenvolvedores que irão integrar a componente de pagamento serão os administradores das aplicações de *e-commerce* com conhecimento do seu código fonte*.*

Após a explicação, foi fornecido aos desenvolvedores alguma documentação explicativa de como integrar o *Widget*. O teste fornecido aos avaliadores pode ser consultado no **Anexo B**.

Enquanto que os avaliadores realizavam o teste, foi mantido um diálogo com os mesmos, no sentido de perceber o que estes estavam a pensar. Desta forma, foi possível anotar aspetos do procedimento dos avaliadores que eram importantes.

No final do teste, foram ainda questionados alguns aspetos adicionais importantes.

Para que este teste não se tornasse exaustivo, os métodos *cart_remove* e *return_from_purchase,* cuja sua implementação deve ser realizada pelos administrados da aplicação de *e-commerce* (*ver secção 4.3.2*), foram fornecidos aos avaliadores, encontrando-se implementados.

5.2.2. Resultados

A nível do tempo decorrido na realização do teste, o resultado superou as espectativas. Em média, os avaliadores conseguiram integrar o *Payment Widget* em 26 minutos. Contudo, se estes tivessem de implementar os métodos referidos na secção anterior, o tempo decorrido seria maior, pois os participantes necessitariam de compreender mais uma parte do código.

Num ambiente real, a implementação dos métodos *cart_remove* e *return_from_purchase*, poderá não ser um problema para os desenvolvedores, porque, para além do desenvolvedores estarem por dentro do seu código fonte, parte dessas tarefas já se encontram implementadas.

Durante a integração *Payment Widget,* não foram verificados quaisquer erros, revelando-se um sinal muito positivo. No decorrer do teste, as maiores dificuldades encontradas foram o preenchimento, no ficheiro de configuração, da classe do formulário de compra e da "div" para o botão "*Buy All*", pois era necessário incluir a notação ".nomeDaClasse" e os avaliadores colocaram "nomeDaClasse".

Uma outra dificuldade, prendeu-se na adição dos ficheiros ao projeto, devido a um dos avaliadores tentar utilizar o caminho absoluto dos ficheiros, em vez de adicionar os ficheiros à pasta do projeto e colocar o caminho relativo.

No final do teste, foi questionado aos participantes se existiam algumas dependências que podiam ser evitadas, ao que ambos responderam que é difícil fazer este procedimento em menos passos (P1: "*Não existe forma de fazer isto em menos passos.")*. Quando questionados por recomendações, um dos avaliadores referiu que as dificuldades encontradas poderiam ser ultrapassadas se o teste se encontrasse mais bem documentado.

5.3. Discussão

Através dos resultados obtidos já nos foi possível inferir algumas conclusões sobre ambos os testes.

Nos testes de usabilidade, os resultados foram muito positivos. O sistema foi considerado, pelos avaliadores, como muito fácil de utilizar, sendo, no entanto, necessário fazer algumas melhorias sobretudo na interface.

Nos testes de facilidade de integração, o sistema também se mostrou fácil de integrar, não tendo os avaliadores mostrado dificuldades significativas, nem sugerido grandes melhorias. Contudo, para que seja possível obter conclusões mais completas, seria necessário realizar este teste com mais participantes e com alguma experiência e em outras aplicações de *e-commerce*.

Em relação ao que estávamos à espera, os resultados foram muito positivos, principalmente do ponto de vista de facilidade de utilização, uma vez que o paradigma de utilização deste sistema de pagamento diverge do paradigma dos sistemas de pagamento tradicionais.

A nível da facilidade de integração, os resultados foram dentro do esperado, pois foi dado ênfase a essa parte durante o processo de desenvolvimento. No entanto, não foi conseguido que a integração fosse um processo trivial, muito devido às dependências inerentes aos botões e aos produtos disponibilizados nas aplicações de *e-commerce*.

A nível das melhorias que poderiam ser feitas para que o sistema fosse utilizável num ambiente real, seria necessário fazer uma revisão da interface*,* em termos de aspeto. Para tal, seria fundamental a contratação de uma equipa de designers gráficos para auxiliarem o processo.

Uma outra melhoria que afeta um pouco a usabilidade, e que deverá ser implementada ainda nesta fase, é a consulta do histórico de compras. Esta melhoria é prioritária, não só por não serem apresentadas quaisquer informações sobre o *Merchant* nas transações, como também por ser necessário escolher previamente a conta que foi utilizada, não existindo possibilidade de utilizar qualquer filtro de pesquisa, tornando-se um incómodo, caso existam muitas compras efetuadas.

5.4. Melhorias

Devido à falta de tempo, decidimos implementar apenas as melhorias essenciais, que neste caso têm a ver com a usabilidade. Dessa forma, foi decidido melhorar a apresentação do histórico de compras efetuadas pelo NPP.

No que diz respeito às melhorias, ficou determinado adicionar informações do comerciante no qual foi feita a transação; apresentar todas as transações em detrimento das transações de uma determinada conta; e implementar uma barra de pesquisa lateral, que possibilitasse a pesquisa por conta bancária e por intervalo de tempo. A pesquisa implementa é uma pesquisa *Ajax*[23] sem qualquer botão de pesquisa, e esta permite ainda ser realizada por um ou diversos parâmetros em simultâneo.

Para além das melhorias na apresentação do histórico de compras realizadas, foi acordado implementar alguns procedimentos importantes do Regulamento Geral de Proteção de Dados (RGPD), que entrou em vigor em 25 de Maio de 2018 [83]. Dessa forma, qualquer utilizador que se registe no nosso sistema, tem de aceitar os termos e condições do RGPD. O sistema está, ainda, de acordo com os artigos 17 e 20 do RGPD.

De acordo com o artigo 17, direito ao esquecimento, os utilizadores podem solicitar que toda a sua informação pessoal seja eliminada [84].

O artigo 20, por sua vez, diz respeito ao direito de portabilidade dos dados, isto significando que, a qualquer momento, os utilizadores podem solicitar todos os seus dados, tendo os detentores dessa informação o dever de fornecer os mesmos e num formato legível.

Para implementar o artigo 17, foi decidido criar um ficheiro JavaScript que faz diversos pedidos à API, de modo a remover a conta predefinida de pagamento, bem como o registo das compras realizadas, e a eliminação da própria conta. Esse ficheiro é acionado através de um *click event* do botão destinado à eliminação dos dados pessoais. Após os dados serem eliminados, o utilizador é encaminhado para a página de login, não podendo fazer qualquer operação sem voltar a se registar.

O artigo 20, também foi implementado através dum ficheiro JavaScript, que é acionado através de um *click event*. Nesse ficheiro, são feitos igualmente diversos pedidos à API, no sentido de obter as informações da conta de pagamento, do histórico de compras e da conta do utilizador. Essas informações são utilizadas para construir um ficheiro JSON, que é descarregado automaticamente.

Na figura 34, encontra-se apresentada a *view* da página do utilizador. Nela, o utilizador pode consultar o seu histórico de compras, assim como acionar um dos artigos do RGPD.

[23] Ajax - Update a web page without reloading the pagehttps://www.w3schools.com/xml/ajax_intro.asp

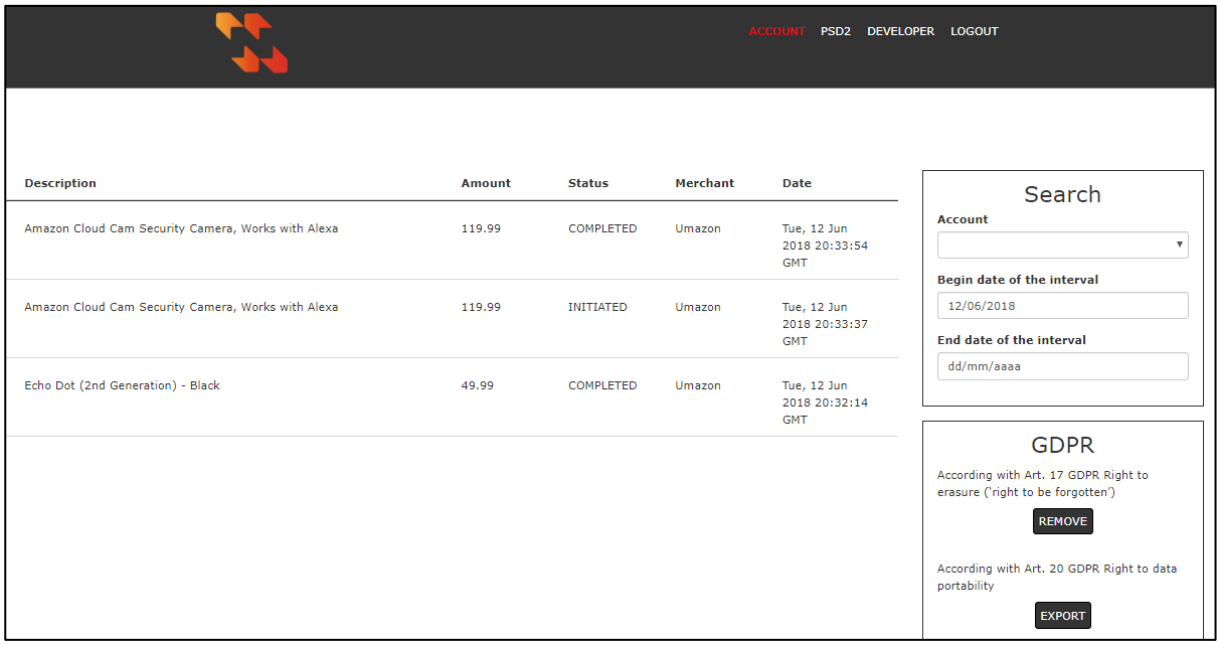

Figura 34 View correspondente a página do utilizador

6. Conclusão

Esta tese foca-se em problemas amplamente identificados do sector bancário como, por exemplo, a inexistência de uma interface comum para todos os bancos, problemas relacionados com a segurança ou a dependência de intermediários para realização de compras. Diversas entidades estão a investir significativamente para a resolução dos mesmos. Entre elas podemos destacar a Comissão Europeia, que propôs uma API centralizada para todo os serviços de *e-banking* com o PSD2. O estudo desta API foi o foco central desta tese.

Esta secção encontra-se dividida em sete secções, onde serão discutidas as contribuições que esta tese teve nos sistemas de *e-banking*, as ferramentas, os métodos utilizados, a implementação, a avaliação, por fim, as lições aprendidas e o trabalho futuro.

6.1. Contribuições

Nas secções que se seguem irão ser referidas algumas contribuições obtidas através da realização desta tese.

6.1.1. Contribuição 1

O problema inerente à dependência de intermediários na realização de compras online foi um dos aspetos tidos em conta nesta tese. Para a resolução do mesmo, foi utilizada a API PSD2 e o serviço inerente ao PSD2 – o PIS.

O PIS é um serviço que foi introduzido com o PSD2, no sentido de resolver os problemas relacionados com a inicialização de pagamentos e com as dependências excessivas com intermediários no momento do pagamento de compras online. Através deste novo serviço, foi implementado um conjunto de botões – "*Buy Now*" e "*Buy All*" –, que têm a capacidade de realizar um pagamento, diretamente entre o cliente e o comerciante. Estes botões contemplam componentes visuais adjacentes, que são integradas conjuntamente nas aplicações de *e-commerce*.

6.1.2. Contribuição 2

A nível da inexistência de uma interface de gestão comum a todos os bancos, o nosso sistema também utiliza uma das soluções introduzidas pelo PSD2, no sentido de resolver esse género de problemas. Esta solução corresponde ao AIS, através do qual é possível, a qualquer cliente, obter uma visualização agregada e centralizada de todas as suas contas. Nesta tese, e através do serviço AIS, foram implementadas algumas funcionalidades que agregam informações de diversas contas como, por exemplo, a possibilidade do utilizador configurar uma conta de pagamentos predefinida e a apresentação do histórico de compras, independentemente da conta em que a compra foi realizada.

6.1.3. Contribuição 3

A abordagem tida nesta tese não se focou no requisito não funcional – segurança –, pois não era um requisito da mesma. Contudo, foi decidido implementar opcionalmente a comunicação, utilizando o protocolo de transporte HTTPS, um sistema de autenticação através de JWT, assim como bases de dados autenticadas com algoritmos de segurança. Esta abordagem de segurança complementa-se com os mecanismos de segurança intrínsecos à API PSD2.

6.2. Ferramentas Utilizadas

No que diz respeito às ferramentas utilizadas, a nível de *Back-End* a combinação do MongoDB [61], Python Flasks [60] e Swagger [67] revelou-se muito positiva, tendo sido possível construir um SDK com vários micro serviços e estabelecer a comunicação devida entre eles. Com a utilização do Swagger, foi possível acrescentar, de forma simples, a documentação e a possibilidade de realizar os pedidos diretamente no HTML.

A nível do *Front-End*, foi optado pelo isolamento do mesmo, sendo desenvolvido inteiramente nas linguagens HTML [54], CSS [56] e JavaScript [59], sem qualquer *framework*. Esta escolha não foi, contudo, a melhor, devido à grande quantidade de pedidos estabelecidos entre o *Front-End* e o *Back-End* e à construção dinâmica dos elementos da DOM[24] mediante a resposta obtida.

Para testar o PSD2, foi utilizada a *sandbox* do OBP[9] que contem uma API. Esta API encontra-se completa, na medida em que oferece um conjunto diversificado de pedidos para o PSD2, e tem um excelente suporte, sendo mesmo possível colocar questões aos desenvolvedores no Slack [25]. Tendo a sua principal desvantagem a indisponibilidade do OAuth 2.0, que é um requisito do PSD2, como mecanismo de autenticação na API.

6.3. Método

O método de desenvolvimento utilizado, foi o método baseado em Prototipagem (*ver secção 3.1.1*). Esta abordagem foi benéfica no início, pois os requisitos não estavam bem claros, e os orientadores não dispunham de muito tempo para definir exaustivamente todos eles. Contudo, esta metodologia apresentou alguns problemas, por surgirem, em certas alturas, novos requisitos que fizeram com que certas partes do desenvolvimento e da escrita fossem refeitas diversas vezes.

Dada a inovação do *state-of-art,* qualquer método de desenvolvimento muito formal seria complicado de seguir. Contudo, um método ágil como o Scrum [85] talvez fosse o mais indicado, pois permitiria organizar melhor a implementação e os requisitos.

[24] Document Object Model – Modelo em árvore com os objetos da página HTML https://www.w3schools.com/js/js_htmldom.asp
[25] Slack Open Bank https://slack.openbankproject.com/

No início, seriam levantados os requisitos, juntamente com os orientadores. Com esses requisitos era gerada uma lista de tarefas a serem realizadas, conhecida como *Product Backlog*, onde pode haver opcionalmente uma priorização das mesmas*,* antes da inicialização do *Sprint.* Uma vez finalizado o *Sprint,* uma nova reunião seria agendada, para realizar o *Sprint Review* e definir o próximo *Sprint* [85].

6.4. Implementação

Tendo em conta os objetivos definidos no enunciado da tese (*ver secção 2.2*), os mesmos foram implementados na sua totalidade, desde a criação de um modulo de suporte, às aplicações de *e-commerce* (fase 1) e à sugestão inteligente da melhor opção do pagamento (fase 2).

Durante a implementação dos objetivos, foi decido adicionar certas funcionalidades que favoreceram a arquitetura do sistema. Dessas funcionalidades, temos um micro serviço para a gestão dos *Roles*, que permite adicionar ou remover um *Role* facilmente do sistema. Para a autenticação, existe um micro serviço dedicado a essa funcionalidade, possibilitando, a qualquer momento, a adição de robustez a nível de segurança. Foi criado, ainda, um micro serviço para a gestão das informações do *Merchant* em que facilmente podem ser adicionadas novas funcionalidades de gestão. Também foi acordado separar o pagamento e as informações da conta bancária, através de dois micro serviços diferentes, fazendo com que possam ser adicionadas novas funcionalidades relacionadas com o pagamento (PIS) ou com a gestão de contas de forma simples (AIS).

Adicionalmente, o facto dos micro serviços implementados construírem uma API *REST* documentada através do Swagger [67], faz com que o nosso sistema funcione igualmente com SDK, possibilitando, a qualquer desenvolvedor, desenvolver um *Front-End* com ferramentas a escolha.

Uma outra vantagem, consiste no fácil *deployment* no Docker [65] dos micro serviços, uma vez que estes foram desenvolvidos a pensar nesse cenário. Por exemplo, no projeto já se encontra, para cada micro serviço, um ficheiro com os *requirments* (bibliotecas necessárias), e outro de configuração com os IPs, para a comunicação entre os diferentes micro serviços, e para a configuração das bases de dados.

6.5. Avaliação

Para validar a implementação do projeto, foram realizados dois tipos de avaliações. A primeira, foi realizada tendo em conta a usabilidade, realizado com intuito de perceber a facilidade de utilização e o apelo estético geral do sistema. A segunda, foi de caracter mais técnico, com o objetivo de verificar com que facilidade o *Payment Widget* pode ser integrado numa aplicação de *e-commerce*.

O teste de usabilidade, consistiu na realização de uma lista de tarefas fornecidas aos avaliadores, para que estes a executassem (*ver secção 5.1*). Esta avaliação foi muito positiva, visto que mostrou que os utilizadores conseguem utilizar o sistema com alguma confiança e consideram-no fácil de utilizar, sendo, contudo, necessário melhorar ligeiramente a interface com base no feedback dos participantes. Para complementar esta avaliação, foi pedido aos participantes que avaliassem o sistema de acordo

com a escala SUS [82]. Os resultados desta avaliação, indicam que o sistema tem uma usabilidade "muito boa" nota A (de A à F) [82] (*ver secção 5.1.2*).

A facilidade de integração foi avaliada por dois desenvolvedores juniores. Esta avaliação, também demonstrou que o sistema pode ser integrado facilmente (*ver secção 5.2.2*). Porém, seria necessário recrutar mais avaliadores, de preferência administradores de aplicações de *e-commerce*, de forma a testar a integração numa aplicação real.

6.6. Lições Aprendidas

Após a elaboração desta tese, é possível fazer um balanço geral dos conhecimentos que foram aprendidos, principalmente no que diz respeito àquilo que voltaria a utilizar da mesma forma no futuro e ao que faria de forma completamente diferente.

O conceito mais importante aprendido neste projeto foi, sem dúvida, o conceito de API *REST*, pois este foi necessário para a exploração da API PSD2 do OBP [76] e para a criação de uma API própria para o sistema de pagamento.

No futuro, voltaria a utilizar o Python Flask [60], para a construção de uma API, pela sua simplicidade, flexibilidade e pelo facto de a maior parte das funcionalidades poderem ser adicionadas aos pedidos através de *decorators* [77]. Esta condição faz com que o código se torne muito simples de organizar, ficando uma implementação limpa, e fácil de manter.

A nível de persistência de dados, a utilização do MongoDB [61], teve um resultado positivo. Este tipo de base de dados, permite uma grande flexibilidade na definição do modelo de dados, devido a inexistência das relações. Além disso, é possível realizar certas abstrações no modelo de dados, que permitem alterações de forma fácil, possibilitando *upgrades* nas plataformas. O MongoDB contempla, ainda, um mecanismo de gestão das bases de dados, através de *Roles* e permissões muito flexíveis e úteis para gerir as bases de dados.

No que diz respeito à documentação da API, um outra experiência enriquecedora consiste na utilização do Swagger [67]. Esta ferramenta torna trivial a elaboração da documentação de uma API, gerando automaticamente o HTML da documentação da API com a possibilidade de executar os pedidos diretamente no HTML gerado.

Um dos pontos menos positivos, durante o desenvolvimento, foi a não utilização de testes adequados. Sempre que um pedido era alterado, tornava-se necessário voltar a introduzir todos os dados novamente para testar o mesmo. Contudo, esta condição poderia ser facilmente ultrapassada através da realização de testes unitários na API. Estes testes iriam permitir que, quando um pedido fosse alterado, apenas fosse necessário executar os testes relacionados com o mesmo, efetuando, quanto necessário, alterações pontuais ao teste.

A nível do *Front-End,* a decisão de implementar o mesmo em JavaScript, CSS e HTML, sem a utilização de uma *framework* de *Front-End,* nem de CSS, revelou-se uma opção pouco acertada, pois tornou-se

muito complicado organizar os ficheiros e estabelecer uma ordem sequencial para os eventos em JavaScript, devido à sua natureza assíncrona, principalmente quando é executado um pagamento em que são realizados seis pedidos à API para cada compra (se o botão "*Buy All*" é clicado com 10 produtos, são 60 pedidos). Além disso, sem a utilização de uma *framework,* é muito difícil obter uma aplicação multiplataforma.

6.7. Trabalho Futuro

Para tornar o sistema utilizável num ambiente real, e melhorar a qualidade de implementação, robustez e segurança, seriam necessárias diversas melhorias.

Uma vez que o tempo não foi suficiente para a implementação completa, foi dissidido deixar algumas melhorias e funcionalidades para trabalho futuro. Na lista que se segue, encontram-se apresentada as tarefas classificadas como "trabalho futuro" por ordem decrescente de prioridade:

1) Revisão da segurança do sistema, nomeadamente a utilização de certificados confiáveis, criação da autenticação utilizando o *OAuth* 2
2) Avaliação da segurança do sistema por uma empresa certificada
3) Utilização do SCA
4) Realização de testes unitários do sistema
5) Teste com uma API de um banco real
6) Revisão do layout da interface com o intuito de deixar o sistema mais apelativo
7) Implementação de novas funcionalidades relacionadas com gestão de contas. Por exemplo, realização de movimentos entre as contas, consulta dos movimentos, entre outras operações relacionadas
8) Criação de uma interface de administrador
9) Criação de uma interface de *Merchant*
10) Criação de aplicações móveis para Android e IOS

7. Referências

[1] «HISTORY OF BANKING». [Em linha]. Disponível em: http://www.historyworld.net/wrldhis/PlainTextHistories.asp?historyid=ac19. [Acedido: 16-Jan-2018].

[2] B. M. Leiner *et al.*, «A brief history of the Internet», *ACM SIGCOMM Comput. Commun. Rev.*, vol. 39, n. 5, pp. 22–31, 2009.

[3] M. Salehi e M. Alipour, «E-Banking in Emerging Economy: Empirical Evidence of Iran», *Int. J. Econ. Finance*, vol. 2, n. 1, p. 201, Jan. 2010.

[4] «Who we work with», *European Commission - European Commission*. [Em linha]. Disponível em: https://ec.europa.eu/info/law/payment-services-psd-2-directive-eu-2015-2366/who-we-work_en. [Acedido: 17-Jan-2018].

[5] K. Miko, «Internet Banking Attacks» [Em linha]. Disponível em: https://pt.slideshare.net/DCIT/internet-banking-attacks-karel-miko

[6] Backbase, «The PSD2 Playbook» [Em linha]. Disponível em: https://backbase.com/wp-content/uploads/2017/04/Backbase_The_PSD2_Playbook.pdf

[7] «Benefits of Adopting Open Banking APIs», *MuleSoft*, 05-Abr-2017. [Em linha]. Disponível em: https://www.mulesoft.com/resources/api/open-banking-apis-benefits. [Acedido: 21-Out-2017].

[8] «Payment services», *European Commission - European Commission*. [Em linha]. Disponível em: https://ec.europa.eu/info/business-economy-euro/banking-and-finance/consumer-finance-and-payments/payment-services/payment-services_en. [Acedido: 30-Out-2017].

[9] «PSD2 - the directive that will change banking as we know it». [Em linha]. Disponível em: http://www.evry.com/en/news/articles/psd2-the-directive-that-will-change-banking-as-we-know-it/. [Acedido: 17-Jan-2018].

[10] «European Commission - PRESS RELEASES - Press release - Payment Services Directive: frequently asked questions». [Em linha]. Disponível em: http://europa.eu/rapid/press-release_MEMO-15-5793_en.htm?locale=en. [Acedido: 01-Nov-2017].

[11] «temenos_psd2_whitepaper_v2.pdf». [Em linha] Disponível em: https://www.temenos.com/globalassets/mi/wp/16/temenos_psd2_whitepaper_v2.pdf. [Acedido: 15-Out-2018]

[12] «Bancario», *http://bancario.pt*. .

[13] «History of Online Banking: How Internet Banking Became Mainstream», *GOBankingRates*, 07-Out-2016. [Em linha]. Disponível em: https://www.gobankingrates.com/banking/history-online-banking/. [Acedido: 05-Out-2017].

[14] N. Kokemuller, «Advantages of E-Banking». [Em linha]. Disponível em: http://budgeting.thenest.com/advantages-ebanking-24063.html. [Acedido: 25-Out-2017].

[15] «Advantages and Disadvantages of Internet Banking», *ToughNickel*. [Em linha]. Disponível em: https://toughnickel.com/personal-finance/Advantages-and-Disadvantages-of-Internet-Banking. [Acedido: 05-Out-2017].

[16] «The Disadvantages of Electronic Banking | Synonym». [Em linha]. Disponível em: http://classroom.synonym.com/disadvantages-electronic-banking-24007.html. [Acedido: 25-Out-2017].

[16] P. Martino, C. Belancray, e G. Bruni Roccia, «Waiting for PSD2"» [Em linha]. Disponível em: https://www2.deloitte.com/content/dam/Deloitte/lu/Documents/financial-services/Banking/lu_en_waiting-for-psd2_122015.pdf

[18] G. Lao e L. Wang, «Application of e-commerce security management strategy in banking», em *Proceedings of the 7th international conference on Electronic commerce*, 2005, pp. 627–632.

[18] Gutlic Belma «Common types of attacks on online banking». [Em linha]. Disponível em: https://sgros-students.blogspot.com/2015/02/common-types-of-attacks-on-online.html

[20] «How Banking Trojans Can Rob You Blind», *Tom's Guide*, 12-Mar-2014. [Em linha]. Disponível em: https://www.tomsguide.com/us/banking-trojan-definition,news-18457.html. [Acedido: 06-Nov-2017].

[21] A. Wiesmaier, M. Fischer, M. Lippert, e J. Buchmann, «Outflanking and securely using the PIN/TAN-System», *arXiv:cs/0410025*, Out. 2004.

[22] «Phishing Attack Prevention: How to Identify & Avoid Phishing Scams», *Digital Guardian*, 23-Jun-2015. [Em linha]. Disponível em: https://digitalguardian.com/blog/phishing-attack-prevention-how-identify-avoid-phishing-scams. [Acedido: 11-Dez-2017].

[23] «Online banking security. Online banking protection.» [Em linha]. Disponível em: http://www.safensoft.com/Online_Banking_Security/. [Acedido: 11-Dez-2017].

[24] F. F. I. E. Council, «Authentication in an internet banking environment», *Financ. Inst. Lett. FIL-103-2005 Wash. DC Fed. Depos. Insur. CorpFDIC Retrieved March*, vol. 18, p. 2005, 2005.

[25] «Transaction authentication number». [Em linha]. Disponível em: https://ipfs.io/ipfs/QmXoypizjW3WknFiJnKLwHCnL72vedxjQkDDP1mXWo6uco/wiki/Transaction_authentication_number.html. [Acedido: 31-Out-2017].

[26] «EU banking and financial services law», *European Commission - European Commission*. [Em linha]. Disponível em: https://ec.europa.eu/info/eu-law-topic/eu-banking-and-financial-services-law_en. [Acedido: 27-Out-2017].

[27] «FinTech - A definition by FinTech Weekly», *FinTech Weekly Definition*. [Em linha]. Disponível em: https://fintechweekly.com/fintech-definition. [Acedido: 11-Dez-2017].

[28] «Functions of the PSP and the Acquirers | sellXed.com | Payment Extensions». [Em linha]. Disponível em: https://www.sellxed.com/en/functions-psp-and-acquirers. [Acedido: 12-Fev-2018].

[29] «PSD2 and the future of payments » Banking Technology». [Em linha]. Disponível em: http://www.bankingtech.com/675841/psd2-and-the-future-of-payments/. [Acedido: 21-Out-2017].

[30] «10 burning questions about banking API are answered here». [Em linha]. Disponível em: https://kontomatik.com/post/10-questions-banking-api. [Acedido: 09-Out-2017].

[31] «The State of APIs in Banking | Currencycloud». [Em linha]. Disponível em: https://www.currencycloud.com/news/blog/the-state-of-apis-in-banking/. [Acedido: 09-Out-2017].

[32] «Payment Services Directive (PSD) - Bank of Cyprus Country». [Em linha]. Disponível em: http://www.bankofcyprus.com.cy/en-GB/Cyprus/Services--Rates/Payment-Services-Directive-PSD/. [Acedido: 30-Out-2017].

[32] Payments UK, «The Second Payment Services Directive (PSD2)». Jul-2016. [Em linha]. Disponível em: http://paymentsystemsconsultancy.com/download/payments-uk-the-second-payment-services-directive-psd2-briefing-july-2016/

[34] M. Kuppinger, «THE FUTURE OF BANKING: Innovation & Disruption in light of the revised European Payment Services Directive (PSD2)». .

[35] «EBA consults on requirements for home-host cooperation under PSD2 - View press release - European Banking Authority». [Em linha]. Disponível em: https://www.eba.europa.eu/-/eba-consults-on-requirements-for-home-host-cooperation-under-psd2. [Acedido: 31-Out-2017].

[36] European Banking Authority, «Consultation Paper on Draft Guidelines on fraud reporting requirements under Article 96(6) of Directive (EU) 2015/2366 (PSD2)». 02-Ago-2017.

[37] «EUR-Lex - 32015L2366 - EN - EUR-Lex». [Em linha]. Disponível em: http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32015L2366. [Acedido: 01-Nov-2017].

[38] «EUR-Lex - 32015L2366 - EN - EUR-Lex». [Em linha]. Disponível em: http://eur-lex.europa.eu/legal-content/EN/TXT/?lang1=PT&lang2=choose&lang3=choose&uri=CELEX%3A32015L2366. [Acedido: 03-Dez-2017].

[39] «5 Answers - What is PSD2? - Quora». [Em linha]. Disponível em: https://www.quora.com/What-is-PSD2. [Acedido: 11-Dez-2017].

[39] P. Fritsch e O. Marechal, «The revised Payment Services Directive (PSD2)». 2016. [Em linha]. Disponível em: https://www.ey.com/Publication/vwLUAssets/Regulatory_agenda_updates_PSDII_Luxembourg/$FILE/Regulatory%20agenda%20updates_PSDII_Lux.pdf

[41] «Security requirements for providers of account information or payment initiation services seeking authorisation or registration under Payment Services Directive 2 (PSD2)», *FCA*, 03-Ago-2017. [Em linha]. Disponível em: https://www.fca.org.uk/firms/revised-payment-services-directive-psd2/applications/security-requirements. [Acedido: 04-Nov-2017].

[42] «EBA consults on Guidelines on security measures for operational and security risks under the PSD2 - View news - European Banking Authority». [Em linha]. Disponível em: https://www.eba.europa.eu/-/eba-consults-on-guidelines-on-security-measures-for-operational-and-security-risks-under-the-psd2. [Acedido: 04-Nov-2017].

[42] Finney Chris «PSD2: “strong customer authentication” – what, when & how?», *Cooley Fintech*, 19-Ago-2016. [Em linha]. Disponível em: https://www.lexology.com/library/detail.aspx?g=d0d14bd7-29cd-4876-9f87-ff2146fc0200

[44] internationalbanker, «PSD2: Are you ready for Strong Customer Authentication (SCA)?», *International Banker*, 13-Dez-2017. [Em linha]. Disponível em: https://internationalbanker.com/banking/psd2-ready-strong-customer-authentication-sca/. [Acedido: 11-Fev-2018].

[45] «PSD2: The EBA’s standards on strong customer authentication – where are we now?» [Em linha]. Disponível em: http://www.osborneclarke.com/insights/psd2-the-ebas-standards-on-strong-customer-authentication-where-are-we-now/. [Acedido: 11-Fev-2018].

[46] MikeWasson, «Microservices architecture style». [Em linha]. Disponível em: https://docs.microsoft.com/en-us/azure/architecture/guide/architecture-styles/microservices. [Acedido: 17-Dez-2017].

[47] «Open Bank Project | Home». [Em linha]. Disponível em: https://openbankproject.com/. [Acedido: 16-Dez-2017].

[48] «Open Bank Project | For Developers». [Em linha]. Disponível em: https://openbankproject.com/for-developers/. [Acedido: 16-Dez-2017].

[49] «Open Bank Project: Get API Key». [Em linha]. Disponível em: https://apisandbox.openbankproject.com/consumer-registration. [Acedido: 16-Dez-2017].

[50] «API Explorer». [Em linha]. Disponível em: https://apiexplorersandbox.openbankproject.com/?version=2.0.0&psd2=true. [Acedido: 16-Dez-2017].

[51] «Open Bank Project: Home». [Em linha]. Disponível em: https://sofisandbox.openbankproject.com/. [Acedido: 16-Dez-2017].

[52] «Nordea | Open Banking Developer Portal». [Em linha]. Disponível em: https://developer.nordeaopenbanking.com/app/documentation?api=Payment%20Initiation%20Services%20API. [Acedido: 16-Dez-2017].

[53] «Open Bank API portal | API Catalogue». [Em linha]. Disponível em: https://openbank.apigee.io/api-catalogue. [Acedido: 16-Dez-2017].

[54] «HTML», *Mozilla Developer Network*. [Em linha]. Disponível em: https://developer.mozilla.org/en-US/docs/Web/HTML. [Acedido: 16-Dez-2017].

[55] «What is HTML? | HyperText Markup Language explained». [Em linha]. Disponível em: http://www.yourhtmlsource.com/starthere/whatishtml.html. [Acedido: 16-Dez-2017].

[56] «CSS», *Mozilla Developer Network*. [Em linha]. Disponível em: https://developer.mozilla.org/pt-BR/docs/Web/CSS. [Acedido: 16-Dez-2017].

[57] «Para iniciantes». [Em linha]. Disponível em: http://tableless.github.io/iniciantes/manual/css/. [Acedido: 16-Dez-2017].

[57] jQuery F.- jquery.org, «jQuery». [Em linha]. Disponível em: https://jquery.com/

[59] «What is jQuery and How to Start using jQuery? - CodeProject». [Em linha]. Disponível em: https://www.codeproject.com/Articles/157446/What-is-jQuery-and-How-to-Start-using-jQuery. [Acedido: 16-Dez-2017].

[60] «Welcome | Flask (A Python Microframework)». [Em linha]. Disponível em: http://flask.pocoo.org/. [Acedido: 21-Fev-2018].

[61] «What Is MongoDB?», *MongoDB*. [Em linha]. Disponível em: https://www.mongodb.com/what-is-mongodb. [Acedido: 17-Dez-2017].

[61] M. Solutions, «Advantages and Disadvantages of Python Programming Language», *Medium*, 24-Abr-2017. [Em linha]. Disponível em: https://medium.com/@mindfiresolutions.usa/advantages-and-disadvantages-of-python-programming-language-fd0b394f2121

[63] «Flask Python Framework». [Em linha]. Disponível em: http://quintagroup.com/cms/python/flask. [Acedido: 21-Fev-2018].

[64] «What is MongoDB? - Definition from WhatIs.com», *SearchDataManagement*. [Em linha]. Disponível em: http://searchdatamanagement.techtarget.com/definition/MongoDB. [Acedido: 17-Dez-2017].

[65] «What is Docker?», *Docker*, 14-Mai-2015. [Em linha]. Disponível em: https://www.docker.com/what-docker. [Acedido: 17-Dez-2017].

[66] CESARDELATORRE, «What is Docker?» [Em linha]. Disponível em: https://docs.microsoft.com/en-us/dotnet/standard/microservices-architecture/container-docker-introduction/docker-defined. [Acedido: 17-Dez-2017].

[67] «Getting Started with Swagger», *Swagger*. [Em linha]. Disponível em: https://swagger.io/getting-started/. [Acedido: 17-Dez-2017].

[68] T. H. | Intern, «Top 4 Software Development Methodologies», *Black Duck Software Blog*. [Em linha]. Disponível em: https://blog.blackducksoftware.com/top-4-software-development-methodologies. [Acedido: 13-Mar-2018].

[69] «What is Prototype model- advantages, disadvantages and when to use it?» [Em linha]. Disponível em: http://istqbexamcertification.com/what-is-prototype-model-advantages-disadvantages-and-when-to-use-it/. [Acedido: 13-Mar-2018].

[69] «Prototyping process models: streamline software development». [Em linha]. Disponível em: https://www.justinmind.com/blog/4-prototyping-process-models-to-streamline-software-development/

[71] «Software Prototyping - IXO Software». [Em linha]. Disponível em: https://ixosoftware.com/2017/11/16/software-prototyping. [Acedido: 10-Jul-2018].

[72] C. Doig, «Why software requirements development matters», *CIO*, 17-Mai-2016. [Em linha]. Disponível em: https://www.cio.com/article/3071333/software/why-software-requirements-development-matters.html. [Acedido: 12-Mar-2018].

[73] «Business Rules: An Agile Introduction». [Em linha]. Disponível em: http://agilemodeling.com/artifacts/businessRule.htm. [Acedido: 06-Abr-2018].

[74] «Functional Requirements vs Non Functional Requirements», *ReQtest*, 05-Abr-2012. .

[75] «Why RESTful communication between microservices can be perfectly fine». [Em linha]. Disponível em: https://www.innoq.com/en/blog/why-restful-communication-between-microservices-can-be-perfectly-fine/. [Acedido: 10-Abr-2018].

[76] «Open Bank Project: Home». [Em linha]. Disponível em: https://psd2-api.openbankproject.com/. [Acedido: 09-Out-2017].

[77] «View Decorators — Flask 0.12.4 documentation». [Em linha]. Disponível em: http://flask.pocoo.org/docs/0.12/patterns/viewdecorators/. [Acedido: 12-Mai-2018].

[78] «Laravel - The PHP Framework For Web Artisans». [Em linha]. Disponível em: https://laravel.com/. [Acedido: 21-Jun-2018].

[78] «Usability Testing», *Software Testing Fundamentals*, 28-Jun-2014. [Em linha]. Disponível em: http://softwaretestingfundamentals.com/usability-testing/

[80] «Thinking Aloud: The #1 Usability Tool», *Nielsen Norman Group*. [Em linha]. Disponível em: https://www.nngroup.com/articles/thinking-aloud-the-1-usability-tool/. [Acedido: 09-Jun-2018].

[81] «Beyond the NPS: Measuring Perceived Usability with the SUS, NASA-TLX, and the Single Ease Question After Tasks and Usability Tests», *Nielsen Norman Group*. [Em linha]. Disponível em: https://www.nngroup.com/articles/measuring-perceived-usability/. [Acedido: 25-Mai-2018].

[82] «How To Use The System Usability Scale (SUS) To Evaluate The Usability Of Your Website», *Usability Geek*, 13-Jul-2015. [Em linha]. Disponível em: https://usabilitygeek.com/how-to-use-the-system-usability-scale-sus-to-evaluate-the-usability-of-your-website/. [Acedido: 10-Jun-2018].

[82] «Proteção de Dados - RGPD (Regulamento Geral Proteção de Dados)», *Proteção de Dados - RGPD*. [Em linha]. Disponível em: https://protecao-dados.pt/o-regulamento/

[84] «Art. 17 GDPR – Right to erasure (‘right to be forgotten’) | General Data Protection Regulation (GDPR)», *General Data Protection Regulation (GDPR)*. .

[85] «What is Scrum?», *Scrum.org*. [Em linha]. Disponível em: http://www.scrum.org/resources/what-is-scrum. [Acedido: 12-Jun-2018].

8. Anexo

8.1. Anexo A - Testes de Usabilidade

Estes testes têm como principal objetivo verificar a usabilidade da plataforma *NearSoft Payment Provaider* e do seu botão de pagamento.

Para a realização do teste, foi criada uma conta na plataforma *Open Bank Project*, a conta tem com ***username****:* "user_test_x" e ***password***: "aB-1234567". Dentro dessa conta, estão disponíveis 2 contas bancárias em 2 bancos diferentes ambas com 120€ depositados.

Algumas das tarefas dependem de outras, dessa forma as tarefas essenciais serão realizadas primeiro. Pede-se que ao realizar as tarefas "pense alto", de forma a que seja retirado o máximo de informação.

1ª Tarefa: Aceda a plataforma NPP e faça um registo da mesma.

2ª Tarefa: Associe na plataforma NPP, a conta OBP fornecida acima.

3ª Tarefa: Escolha das duas contas cridas para você, como acima referido, uma conta de pagamentos predefinida (*default*).

4ª Tarefa: Aceda a Umazon e realize uma compra de um produto que custe mais de 100€.

5ª Tarefa: Faça uma nova compra de qualquer produto.

6ª Tarefa: Consulte o histórico das compras.

Tarefa	Tempo de início	Tempo de fim	Nº Tentativas	Avaliação da Interface (1 a 10)	Avaliação da facilidade (1 a 10)
1					
2					
3					
4					
5					

6					

Tarefa	**Nº *Bugs***	**Informação do *bug***
1		
2		
3		
4		
5		
6		

Tarefa	**Anotações importantes**
1	
2	
3	
4	
5	
6	

1. I think that I would like to use this system frequently.

1. Strongly Disagree	2.	3.	4.	5. Strongly Agree
○	○	○	○	○

2. I found the system unnecessarily complex.

1. Strongly Disagree	2.	3.	4.	5. Strongly Agree
○	○	○	○	○

3. I thought the system was easy to use.

1. Strongly Disagree	2.	3.	4.	5. Strongly Agree
○	○	○	○	○

4. I think that I would need the support of a technical person to be able to use this system.

1. Strongly Disagree	2.	3.	4.	5. Strongly Agree
○	○	○	○	○

5. I found the various functions in this system were well integrated.

1. Strongly Disagree	2.	3.	4.	5. Strongly Agree
○	○	○	○	○

6. I thought there was too much inconsistency in this system.

1. Strongly Disagree	2.	3.	4.	5. Strongly Agree
○	○	○	○	○

7. I would imagine that most people would learn to use this system very quickly.

1. Strongly Disagree	2.	3.	4.	5. Strongly Agree
○	○	○	○	○

8. I found the system very cumbersome to use.

1. Strongly Disagree	2.	3.	4.	5. Strongly Agree
○	○	○	○	○

9. I felt very confident using the system.

1. Strongly Disagree	2.	3.	4.	5. Strongly Agree
○	○	○	○	○

10. I needed to learn a lot of things before I could get going with this system.

1. Strongly Disagree	2.	3.	4.	5. Strongly Agree
○	○	○	○	○

8.2. Anexo B - Testes de Facilidade de Integração

Neste teste é pretendido verificar a facilidade de integração de um botão de pagamentos num site de *e-commerce*, e testar o seu funcionamento.

Para este teste são fornecidos um website de *e-commerce* e o botão de pagamento que será integrado no website. O website encontra-se na diretoria ***/test/e-commerce_website***, o botão por sua vez encontra-se na diretoria ***/test/button***. O botão é composto por 2 pastas, uma delas ***/test/button/js*** contém os ficheiros JavaScript, a outra, ***/test/button/css*** corresponde aos ficheiros CSS.

Preparação para o teste:

Antes da realização do teste, será feita uma breve explicação do código do site de *e-commerce*, uma vez que num ambiente real os desenvolvedores que irá integrar o botão estão dentro do código do website.

Tarefa a realizar:

É pretendido neste teste integrar o componente botão no *website* e testar o seu funcionamento. Para tal devem ser seguidas as seguintes instruções:

- Os scripts (***/test/button/js/buy_buttons.js*** e ***/test/button/js/config.js***) devem ser adicionado no *header* de todas as páginas que necessitem de conter um dos botões embebidos.
- Os ficheiros **.css** são adicionados a página através do script **buy_buttons.js,** no entanto estes ficheiros têm de fazer parte do projeto, o caminho que deverá ser comum a todos os ficheiros **.css**. Para tal, tem de ser adicionado no ficheiro de configuração **config.js** no parâmetro **”css_path”.**
- Cada produto deverá conter um formulário com *inputs* com os dados dos seguintes parâmetros*: amount, currency, product_name, product_id e quantity.*
- A classe do formulário dos produtos que deve ser comum ao parâmetro **”product_form”** do ficheiro de configuração (***/test/button/js/config.js***).
- Para o botão “*Buy All*”, é necessário que seja definido a classe da onde é pretendido embeber o botão, no parâmetro **”buy_all”** do ficheiro de configuração.

Questionário:

Data e hora de início:

Data e hora de fim:

Foram encontradas inconsistências? Quais?

Foram encontrados *bugs*? Quais?

Existem dependências desnecessárias? Quais?

Maior dificuldade?

Notas importantes:

8.3. Anexo C - Imagens

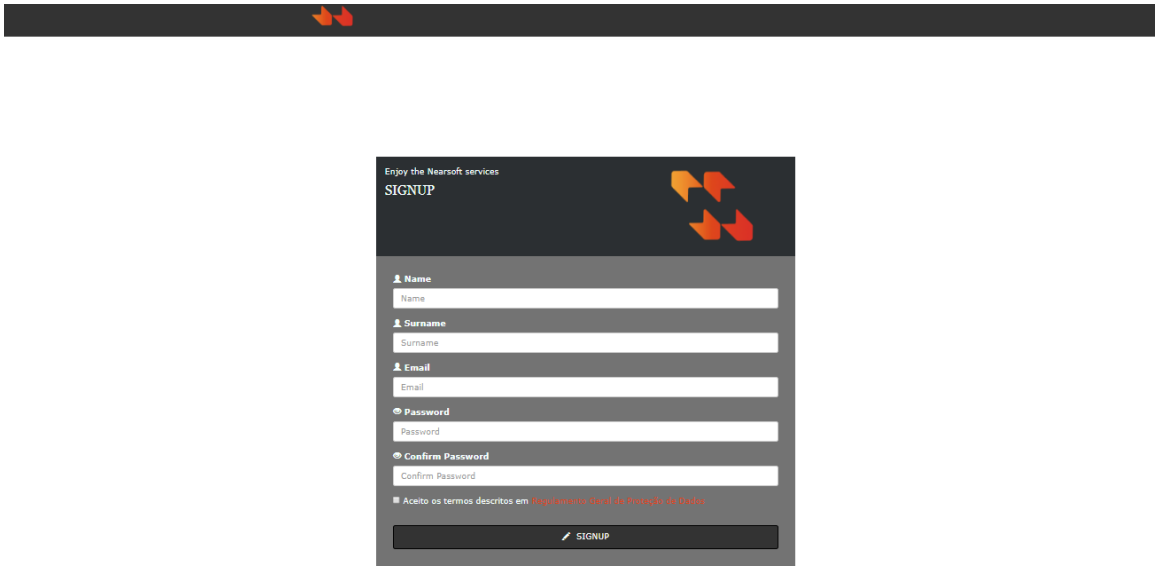

Figura 35 Formulário de registo

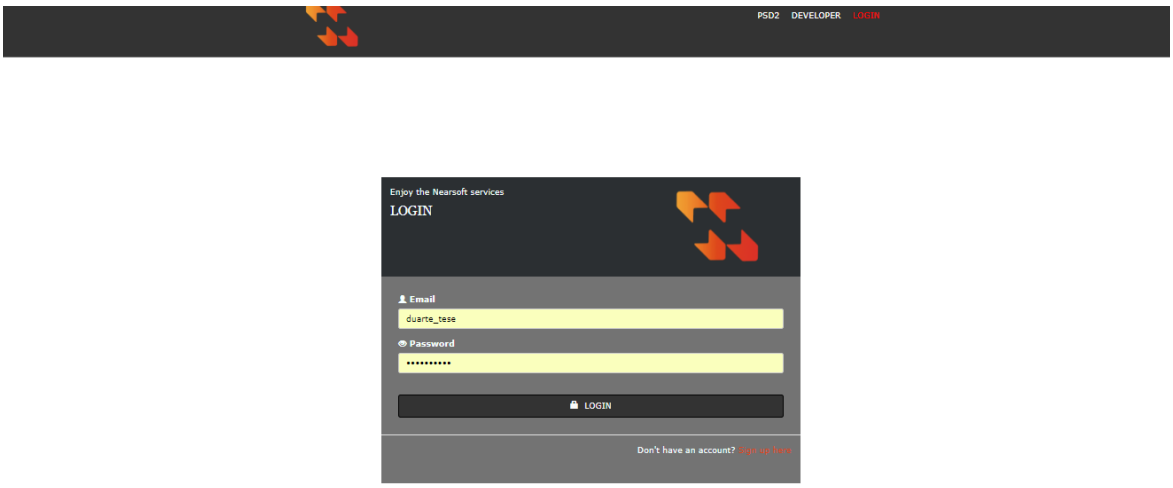

Figura 36 Formulário de Login

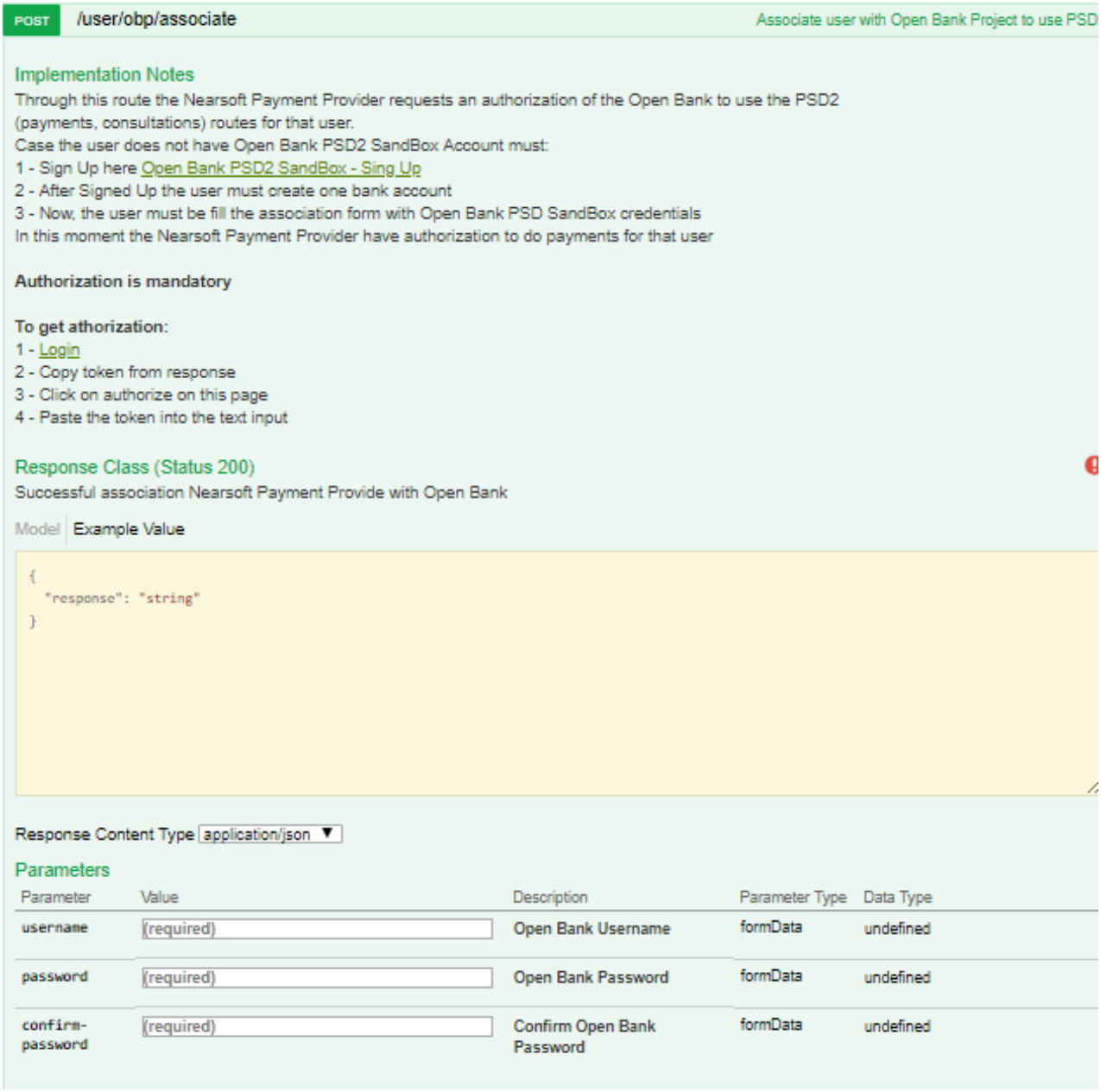

Figura 37 Documentação do pedido de associação no OBP API

```
sp/bank/<bank_id>/account/<account_id>/answer-challenge: User 5aef2d32e5a28c1688cc1342 answer challenge of transaction 160d23ed-bbe5-45d9-b3
sp/bank/<bank_id>/account/<account_id>/charge: User 5aef147de5a28c1688cc133c get the charge for BANK_ID=psd201-bank-x--uk and ACCOUNT_ID 20
sp/bank/<bank_id>/account/<account_id>/initiate-transaction-request: User 5aef147de5a28c1688cc133c initiate a transaction request on BANK_I
sp/bank/<bank_id>/account/<account_id>/answer-challenge: User 5aef147de5a28c1688cc133c answer challenge of transaction 12ba9720-b16a-4aa3-a
sp/bank/<bank_id>/account/<account_id>/charge: User 5aef147de5a28c1688cc133c get the charge for BANK_ID=psd201-bank-x--uk and ACCOUNT_ID 20
sp/bank/<bank_id>/account/<account_id>/initiate-transaction-request: User 5aef147de5a28c1688cc133c initiate a transaction request on BANK_I
sp/bank/<bank_id>/account/<account_id>/charge: User 5aef147de5a28c1688cc133c get the charge for BANK_ID=psd201-bank-x--uk and ACCOUNT_ID 20
sp/bank/<bank_id>/account/<account_id>/initiate-transaction-request: User 5aef147de5a28c1688cc133c initiate a transaction request on BANK_I
sp/bank/<bank_id>/account/<account_id>/charge: User 5aef147de5a28c1688cc133c get the charge for BANK_ID=psd201-bank-x--uk and ACCOUNT_ID 20
sp/bank/<bank_id>/account/<account_id>/initiate-transaction-request: User 5aef147de5a28c1688cc133c initiate a transaction request on BANK_I
sp/bank/<bank_id>/account/<account_id>/charge: User 5aef147de5a28c1688cc133c get the charge for BANK_ID=psd201-bank-x--uk and ACCOUNT_ID 20
sp/bank/<bank_id>/account/<account_id>/initiate-transaction-request: User 5aef147de5a28c1688cc133c initiate a transaction request on BANK_I
sp/bank/<bank_id>/account/<account_id>/charge: User 5aff47dbe5a28c2b60d6d90e get the charge for BANK_ID=psd201-bank-x--uk and ACCOUNT_ID fa
sp/bank/<bank_id>/account/<account_id>/initiate-transaction-request: User 5aff47dbe5a28c2b60d6d90e initiate a transaction request on BANK_I
sp/bank/<bank_id>/account/<account_id>/answer-challenge: User 5aff47dbe5a28c2b60d6d90e answer challenge of transaction 06b39ac1-a86e-4e5a-b5
sp/bank/<bank_id>/account/<account_id>/charge: User 5aff47dbe5a28c2b60d6d90e get the charge for BANK_ID=at02-bank-x--01 and ACCOUNT_ID fabi
sp/bank/<bank_id>/account/<account_id>/initiate-transaction-request: User 5aff47dbe5a28c2b60d6d90e initiate a transaction request on BANK_I
sp/bank/<bank_id>/account/<account_id>/charge: User 5aff47dbe5a28c2b60d6d90e get the charge for BANK_ID=at02-bank-x--01 and ACCOUNT_ID fabi
sp/bank/<bank_id>/account/<account_id>/initiate-transaction-request: User 5aff47dbe5a28c2b60d6d90e initiate a transaction request on BANK_I
sp/bank/<bank_id>/account/<account_id>/charge: User 5aef147de5a28c1688cc133c get the charge for BANK_ID=psd201-bank-x--uk and ACCOUNT_ID 20
sp/bank/<bank_id>/account/<account_id>/initiate-transaction-request: User 5aef147de5a28c1688cc133c initiate a transaction request on BANK_I
sp/bank/<bank_id>/account/<account_id>/charge: User 5aef147de5a28c1688cc133c get the charge for BANK_ID=psd201-bank-x--uk and ACCOUNT_ID 20
sp/bank/<bank_id>/account/<account_id>/initiate-transaction-request: User 5aef147de5a28c1688cc133c initiate a transaction request on BANK_I
sp/bank/<bank_id>/account/<account_id>/answer-challenge: User 5aef147de5a28c1688cc133c answer challenge of transaction 955cd208-f385-45c7-98
```

Figura 38 Ficheiro Log do micro serviço PISP