UNIVERSIDADE D
COIMBRA

José Miguel Silva Gomes

PLUGIN DE PAGAMENTOS WEB3 PARA PLATAFORMAS DE E-COMMERCE

Dissertação no âmbito do Mestrado em Engenharia Informática, especialização em Engenharia de Software, orientada pela Professora Doutora Marília Curado e pelos Engenheiros Raul Fonseca e Daniel Oliveira e apresentada ao Departamento de Engenharia Informática da Faculdade de Ciências e Tecnologia da Universidade de Coimbra.

Julho de 2023

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D
COIMBRA

Department of Informatics Engineering

José Miguel Silva Gomes

Web3 payments plugin for E-commerce platforms

Dissertation in the context of the Master in Informatics Engineering, specialization in Software Engineering, advised by Prof. Marília Curado and by Engineers Raul Fonseca and Daniel Oliveira and presented to the Department of Informatics Engineering of the Faculty of Sciences and Technology of the University of Coimbra.

July 2023

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D
COIMBRA

Departamento de Engenharia Informática

José Miguel Silva Gomes

Plugin de pagamentos Web3 para plataformas de E-commerce

Dissertação no âmbito do Mestrado em Engenharia Informática, especialização em Engenharia de Software, orientada pela Professora Doutora Marília Curado e pelos Engenheiros Raul Fonseca e Daniel Oliveira e apresentada ao Departamento de Engenharia Informática da Faculdade de Ciências e Tecnologia da Universidade de Coimbra.

Julho de 2023

Agradecimentos

Após a conclusão do estágio, é importante reconhecer e expressar minha gratidão a todas as pessoas que contribuíram de forma direta ou indireta para o meu crescimento profissional e pessoal ao longo deste período de aprendizagem. Sem a ajuda e apoio de cada uma delas, este estágio não teria sido possível. Assim, gostaria de expressar meu sincero agradecimento a todos os envolvidos.

Em primeiro lugar, gostaria de agradecer à empresa WIT Software pela oportunidade de realizar o estágio e pela valiosa experiência proporcionada. Agradeço especialmente ao meu supervisor Raul Fonseca, pela sua disponibilidade de esclarecer qualquer dúvida que surgia, confiança e apoio ao longo do estágio. Ao meu tutor técnico, Daniel Oliveira pela sua dedicação, suporte e constante monitorização do trabalho realizado durante o estágio. Às *business analysts* Ana Fernandes e Aline Santos pelos conselhos e apoio fornecidos. A todos eles o meu sincero obrigado pelo conhecimento e experiências transmitidas ao longo do estágio curricular.

Expresso minha gratidão à professora Marília Curado, do Departamento de Engenharia Informática, pela sua orientação na elaboração deste relatório e apoio durante todo o período do estágio na sua constante preocupação do desenvolvimento do trabalho através de sucessivas reuniões que se traduziram num *feedback* construtivo e encorajador.

Não posso deixar de mencionar minha família, o meu pai, minha mãe e minha irmã pelo apoio incondicional e pelas condições que me disponibilizaram de modo a realizar o melhor trabalho possível ao longo de todo o meu percurso académico.

Finalmente, gostaria de expressar minha gratidão aos meus amigos e colegas da faculdade. Durante os últimos 5 anos, eles estiveram ao meu lado, transmitindo conhecimentos, partilhando experiências e oferecendo um apoio constante durante esta fase. Não posso deixar de mencionar o impacto positivo que tiveram na minha jornada, tornando-a mais significativa e enriquecedora. A cada um de vós, agradeço pelo apoio incondicional e pela viagem partilhada.

Abstract

The use of Blockchain technology in online transactions, known as Web3 payments, has gained increasing popularity in recent years. This thesis aims to expand upon the capabilities of traditional Web3 payments by developing a plugin that enables the use of cryptocurrency and non-fungible tokens (NFTs) for e-commerce transactions. The emergence of Web3 has opened up new possibilities for online payments, including increased security and transparency.

In addition to the technical development of the plugin, this thesis will also examine the potential benefits and drawbacks of using Web3 payments for merchants and customers. This will include an analysis of the security and reliability of Blockchain technology, as well as its potential impact on the traditional financial system.

One of the main focuses of this research is the integration of vouchers in the form of NFTs (Non-Fungible Tokens) in the payment process. NFTs offer an innovative and secure way to represent digital items in a format that can be easily verified and transferred on the Blockchain. This enables merchants to create unique and personalized vouchers, providing customers with a secure and convenient way to acquire them.

The overall objective of this thesis is to contribute to the growing field of Web3 payments by providing a practical and user-friendly solution for merchants looking to adopt this new technology. The report details the entire research and development process undertaken during this internship, highlighting the advancements in integrating NFTs into e-commerce transactions and the challenges faced during the plugin's development.

Keywords

Blockchain, Smart-Contract, Web3, Cryptocurrency, NFT, ERC-20, ERC-721

Resumo

O uso da tecnologia *Blockchain* em transações online, conhecido como pagamentos Web3, tem ganho cada vez mais popularidade nos últimos anos. Esta tese tem como objetivo expandir as capacidades dos pagamentos Web3 tradicionais, desenvolvendo um *plugin* que possibilite o uso de criptomoedas e NFT para transações de *e-commerce*. A emergência da Web3 tem aberto novas possibilidades para pagamentos online, incluindo aumento da segurança e transparência.

Além do desenvolvimento técnico do *plugin*, esta tese também examinará os possíveis benefícios e desvantagens do uso de pagamentos Web3 para comerciantes e clientes. Isso incluirá uma análise da segurança e confiabilidade da tecnologia *Blockchain*, bem como o seu potencial impacto no sistema financeiro tradicional.

Um dos principais focos desta teste é a integração de vouchers sob a forma de NFT no processo de pagamento. Os NFTs oferecem uma maneira inovadora e segura de representar itens digitais num formato que pode ser facilmente verificado e transferido na *Blockchain*. Isso permite que os comerciantes criem vouchers exclusivos e personalizados, proporcionando aos clientes uma forma segura e conveniente de adquiri-los.

O objetivo geral desta tese é contribuir para o campo em expansão dos pagamentos Web3, fornecendo uma solução prática e amigável para comerciantes que desejam adotar esta nova tecnologia. O relatório detalha todo o processo de pesquisa e desenvolvimento realizado ao longo deste estágio, destacando os avanços na integração de NFT em transações de e-commerce e os desafios enfrentados durante o desenvolvimento do plugin.

Palavras-Chave

Blockchain, Smart-Contract, Web3, *Cryptocurrency*, NFT, ERC-20, ERC-721

Conteúdo

Acrónimos

ABI Application Binary Interface.

API Application Programming Interface.

B2B Business-to-Business.

BTC Bitcoin.

CDN Content delivery network.

dAPPs Decentralized Applications.

DEX Decentralized Exchange.

DPoS Delagated Proof of Stake.

ERC Ethereum Request for Comments.

ETH Ethereum.

IPFS Interplanetary File System.

JWT JSON Web Tokens.

KYC Know Your Customer.

NASA National Aeronautics and Space Administration.

NFT Non-Fungible Tokens.

P2P Peer-to-Peer.

PCI Payment Card Industry Data Security Standard.

PIL Python Imaging Library.

PoS Proof of Stake.

POS Point of Sale.

PoW Proof of Work.

SaaS Software as a Service.

SEO Search Engine Optimization.

SPV Simple Payment Verification.

SSL Secure Sockets Layer.

TRL Technology Readiness Level.

UPS United Parcel Service.

URI Uniform Resource Identifier.

USPS United States Postal Service.

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

O presente relatório tem por base o estágio curricular desenvolvido no âmbito do Mestrado Engenharia Informática segundo o ramo de especialização em Engenharia de Software do Departamento de Engenharia Informática da Faculdade de Ciências e Tecnologia da Universidade de Coimbra.

O trabalho foi elaborado na empresa WIT Software S.A, sob a orientação do Engenheiro Raul Fonseca e do tutor técnico Engenheiro Daniel Oliveira com o acompanhamento da Professora Doutora Marília Pascoal Curado.

1.1 WIT Software

A WIT Software é uma empresa presente no setor da tecnologia, especializada no desenvolvimento de soluções inovadoras para comunicações móveis e sistemas de informação. A empresa destaca-se por fornecer soluções de software de elevada qualidade para empresas em todo o mundo. A sua especialização abrange áreas de comunicação e desenvolvimento de aplicações móveis, integração de sistemas e consultoria em TI.

A empresa possui mais de duas décadas de experiência e o seu software está presente em mais de 40 países ao redor do mundo.

1.2 Contexto

A adoção da tecnologia *Blockchain*, em particular criptomoedas, está a crescer de dia para dia ultrapassando o ritmo de adoção da internet durante os anos 90 [1]. Globalmente, cerca de 300 milhões de pessoas [2] já utilizam criptomoedas e já

efetuam compras com esse método de pagamento.

Empresas de venda de produtos e serviços ao público estão atentas a este crescimento e já anunciaram que passaram a aceitar criptomoedas como meio de pagamento, como por exemplo a Tesla [3]. Outras empresas começam a fazer planos para adotar meios de pagamento em criptomoedas no futuro próximo. Recentemente, empresas como Stripe ou Shopify, passaram também a suportar pagamentos em criptomoedas, proporcionando uma maior abrangência desta tecnologia no segmento de lojas online.

No entanto estas soluções de pagamentos estão limitadas às moedas mais populares e funcionam num modelo *Software as a Service (SaaS)*, aumentando custos e dificultando a entrada a alguns comerciantes.

As criptomoedas e os *Non-Fungible Tokens (NFT)* são duas formas populares de moeda digital. Os NFTs tiveram um aumento de popularidade em 2021 [4] e, com isso, surgiram novas maneiras de os utilizar. Por exemplo, algumas empresas estão a oferecer NFTs aos seus clientes através de *loyalty programs* [5]. Os clientes podem usar esses benefícios para comprar produtos na loja da empresa ou participar em parcerias da marca.

1.3 Objetivos

Esta secção visa descrever os objetivos do estágio e, como tal encontra-se dividida em duas subsecções: a primeira pretende identificar os objetivos pessoais do autor e a segunda tem como objetivo explicar os objetivos do projeto.

Objetivos pessoais

Através deste estágio, o autor pretende consolidar e aperfeiçoar o conhecimento aprendido ao longo destes últimos 5 anos, através da Licenciatura e do Mestrado, bem como a aprendizagem de novos modelos de trabalho em contexto não académico com o propósito de desenvolver uma rotina e uma forma de trabalhar consistente e sólida.

A nível profissional, o autor pretende desenvolver um projeto que traga valor à WIT Software, finalizando assim o Mestrado com sucesso e com um elevado grau de aprendizagem de forma a que exista um desenvolvimento das suas qualidades enquanto *developer* aprendendo processos e metodologias aplicadas ao mundo real.

Objetivos do projeto

O presente estágio tem como objetivo principal prototipar um sistema de pagamentos em criptomoedas para plataformas de *e-commerce*. Esse sistema deve ser capaz de aceitar *tokens* personalizados ERC20 e não depender de entidades externas para funcionar. A solução tem como objetivo ampliar o número de criptomoedas aceites e reduzir os custos de aceitação para os comerciantes.

Este sistema deverá também ser capaz de aceitar *vouchers* digitais sob a forma de NFT na *Blockchain* Polygon com suporte para *Smart Contracts*. Através destes *vouchers*, os clientes dessa loja poderão utilizá-los com o objetivo de apresentarem um desconto no processo de pagamento, desconto esse estipulado na estrutura de metadados de cada NFT.

No final do estágio deverá existir um protótipo funcional onde seja possível executar as funcionalidades propostas de forma a evidenciar o trabalho realizado.

1.4 Estrutura do documento

O presente documento encontra-se dividido em 10 capítulos, em que cada um deles visa identificar cada uma das etapas decorridas ao longo do projeto:

- No Capítulo 1 é realizada uma introdução ao projeto passando pela identificação da empresa onde o estagiário irá desenvolver a tese, uma contextualização ao problema e os seus objetivos, tantos pessoais como profissionais;
- No Capítulo 2 é realizada uma contextualização teórica de conceitos importantes para o desenrolar do tema e necessários para a compreensão da solução. Correspondem maioritariamente a aspetos técnicos como por exemplo o conceito de *Blockchain* e a evolução da *web*;
- De seguida, no Capítulo 3 é elaborado o Estado da Arte, onde são referidos os estudos elaborados para o ponto base da solução, de forma a compreender o problema e alternativas existentes;
- Através do Capítulo 4 é descrita a metodologia seguida ao longo do estágio, bem como o planeamento estipulado para ambos os semestres de trabalho e o processo de identificação e gestão de riscos;
- No Capítulo 5 são analisados os requisitos do problema incluindo diagramas com os respetivos casos de uso. Neste capítulo é identificado o ponto de partida para a solução baseada na pesquisa e requisitos identificados;

- O Capítulo 6 tem como objetivo identificar a arquitetura escolhida para o projeto tendo em conta o ponto de partida da solução descrita no capítulo anterior. Para além da arquitetura são também estipuladas as tecnologias a adotar juntamente com uma análise preliminar fruto do processo de investigação e análise do ponto de partida da solução;

- O Capítulo 7 visa fornecer uma visão detalhada da gestão e organização do projeto durante a fase de desenvolvimento. Neste capítulo, serão apresentadas as ferramentas de gestão utilizadas e a estrutura adotada para a condução do projeto;

- No Capítulo 8, será abordada a fase de desenvolvimento, apresentando uma descrição detalhada do trabalho realizado durante essa etapa, incluindo a análise dos requisitos, as atividades realizadas e problemas enfrentados;

- O Capítulo 9 tem como propósito identificar a fase de testes que ocorreu após a fase de desenvolvimento de modo a testar a aplicação e corrigir problemas que surgiram. Ainda neste capítulo são identificados os testes unitários elaborados e análise dos requisitos não funcionais recolhidos no Capítulo 5;

- Por fim, o Capítulo 10 tem como objetivo resumir o trabalho realizado durante o estágio, fornecendo uma conclusão sobre o progresso alcançado, as lições aprendidas e as direções para trabalhos futuros.

Capítulo 2

Introdução ao problema

Neste capítulo, serão introduzidos os conceitos teóricos fundamentais da *Blockchain* e da Web3, incluindo o que são e como funcionam. Serão também explorados os diferentes métodos de consenso utilizados para garantir a integridade das informações armazenadas na *Blockchain* e discutir algumas das principais características e propriedades que a compõem, como descentralização, transparência e inalterabilidade.

2.1 Evolução da Web

A centralização ajudou milhões de pessoas a entrarem na *World Wide Web* e criou a infraestrutura estável e robusta que dá suporte a milhões de empresas e aos seus utilizadores[6] [7]. Ao mesmo tempo, estas entidades centralizadas têm um grande monopólio no domínio da Internet, decidindo unilateralmente o que deve e não deve ser permitido, bem como o acesso e venda dos nossos dados pessoais.

A Web3 é a resposta a este dilema. Em vez de uma Web monopolizada por grandes empresas tecnológicas, a Web3 abraça a descentralização e está a ser construída, operada, sendo propriedade dos seus utilizadores. A Web3 coloca o poder nas mãos de indivíduos e não de corporações [8].

2.1.1 Web 1.0

Foi no final da década de 90 em que a Internet estava a surgir e com ela surgiu a 1ª era da Web. Era considerada a web apenas de leitura, em que os seus websites eram apenas informativos e englobavam apenas conteúdo estático. Não possuíam qualquer conteúdo interativo ou componentes de design e estavam principalmente ligados através de hiperligações.

Em suma, como se pode verificar pela Figura 2.1 a Web 1.0 era uma *Content delivery network (CDN)* que permitia a exibição de informação em sítios web onde os utilizadores consumiam passivamente materiais sem terem a opção de deixar comentários ou outros tipo de *feedback*.

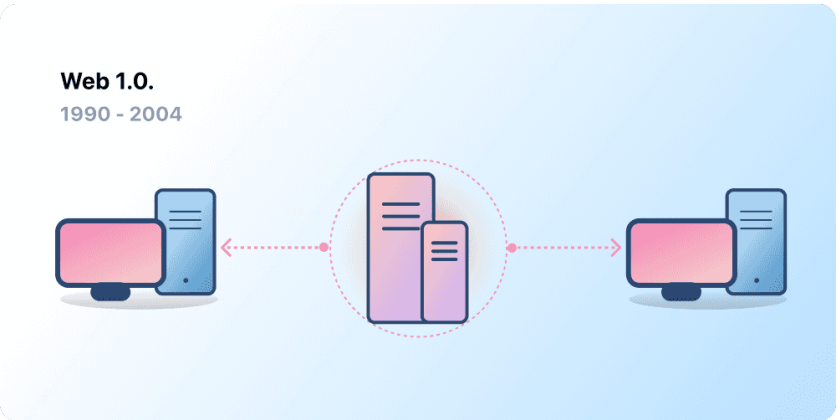


Figura 2.1: Representação do modo de funcionamento da Web 1.0 - Leitura [9]

2.1.2 Web 2.0

O período da Web 2.0 começou em 2004 com o aparecimento de plataformas de comunicação social. Em vez de ser apenas de leitura, a web evoluiu para ser de leitura-escrita. Empresas que fornecem conteúdos aos utilizadores começaram também a fornecer plataformas para partilhar conteúdos gerados pelo utilizador e a envolver-se em interações utilizador a utilizador. À medida que mais pessoas iam ficando *online*, as empresas começaram a ganhar força e controlo sobre os seus utilizadores bem como passaram a controlar uma quantidade desproporcionada do tráfego e do valor gerado na web.

A Web 2.0 também deu origem ao modelo de receitas impulsionado pela publicidade. Enquanto os utilizadores podiam criar conteúdos, não eram proprietários nem beneficiavam da sua monetização.

Corresponde ainda ao período em que vivemos e, os casos de uso da Web2.0 passaram meramente de ser algo de comunicação e interação para *e-comerce* aumentando o número de utilizadores que consumiam conteúdo. É considerada *"web as a platform"* onde aplicações de *software* começaram a ser construídas tal como é representado pela Figura 2.2.

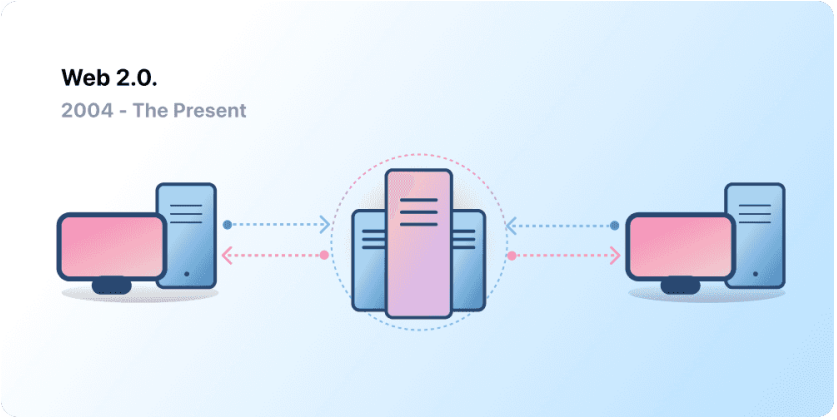


Figura 2.2: Representação do modo de funcionamento da Web 2.0 - Leitura e escrita [9]

2.1.3 Web 3.0

A Web3, também conhecida como Web descentralizada, é uma evolução da Web2 que busca resolver um problema fundamental: a necessidade de confiança e controlo das suas informações. Ao contrário da maior parte da Web atual, que depende da confiança em empresas privadas para agir no melhor interesse do público, a Web3 adota um modelo descentralizado, onde os utilizadores são os proprietários de suas próprias informações e desempenham um papel ativo no desenvolvimento e operação dos serviços [10].

A Web3 tem por base os seguintes princípios: [11][12]

- **Descentralização**: em vez da informação e dos dados serem controlados e pertencentes a entidades centralizadas, a propriedade é distribuída entre seus construtores e utilizadores;
- **Sem permissões**: todos os utilizadores têm acesso igualitário e sem exclusões à participação na Web3, sem quaisquer repercussões negativas;
- **Segurança**: a Web3 opera através de mecanismos económicos ao invés da confiança com terceiros ;

Web3 é uma continuação da Web2 em termos de interatividade, mas nos seus pilares está o protocolo da *Blockchain* de acordo com a Figura 2.3 a comunicação ocorre de forma descentralizada através da tecnologia descentralizada *Peer-to-Peer (P2P)*.

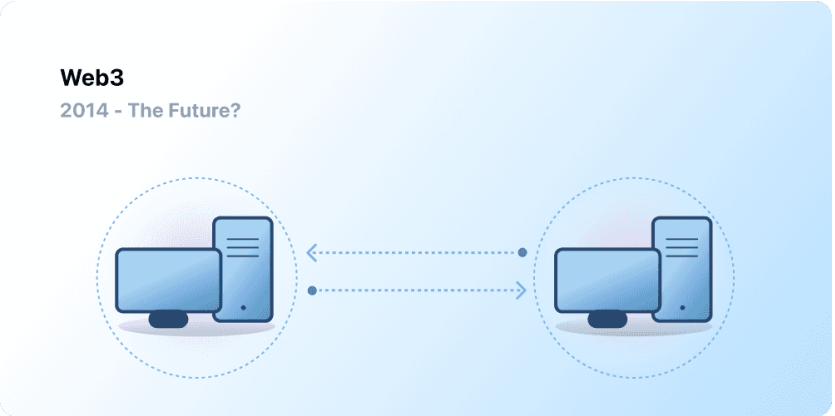


Figura 2.3: Representação do modo de funcionamento da Web 3.0 - Leitura, escrita e propriedade [9]

Através da Figura 2.4, conseguimos dividir a Web3 em 5 camadas [13]:

1. ***Layer 0 - Infrastructure***: Corresponde à camada subjacente sob a qual tudo foi construído e é aquela responsável por fazer a Web3 possível. Inclui todas as especificações técnicas e mecanismos de comunicação na *Blockchain* através de *nodes*.

2. ***Layer 1 - Network***: Esta camada é responsável por incluir os mecanismos de consenso, e podemos incluir nesta camada o termo *Blockchain*. Contém os aspetos necessários para a construção de redes fundamentais como Bitcoin e Ethereum que permitem com que estas redes estejam operacionais, incluindo os seus aspetos de segurança como por exemplo algoritmos de *hashing*, autenticação entre outros *standards*.

3. ***Layer 2 - Protocol***: Também conhecida como *Chain Extention*, esta camada visa introduzir melhorias no modo de operação de uma determinada *Blockchain*. Estas melhorias têm como objetivo aprimorar o modo de funcionamento da rede baseando-se na métrica de escalabilidade, uma vez que redes da camada 1 são geralmente ineficientes gastando uma quantidade significativa de recursos.

4. ***Layer 3 - Services***: Esta camada de tecnologia contém todas as ferramentas necessárias para criar e gerir a camada *dApps*. Geralmente inclui conjunto de dados, computação *off-chain*, canais de estado e *side-chains*. É através desta camada que os utilizadores conseguem fazer operações na *Blockchain* como por exemplo elaborar um *smart-contract* ou transacionar um NFT através de bibliotecas Web3 que ligam *smart-contracts* a interfaces *dApp*: ether.js, web3.js ou web3.py.

5. ***Layer 4 - Application Layer***: Por fim, no topo da Web3 existe a *Application Layer* responsável pelo ponto de partida a todo o tipo de atividades na Web3. Esta permite que os utilizadores finais interajam com a Web3 através de uma componente de frontend. É nesta camada que podemos criar a nossa *crypto wallet*

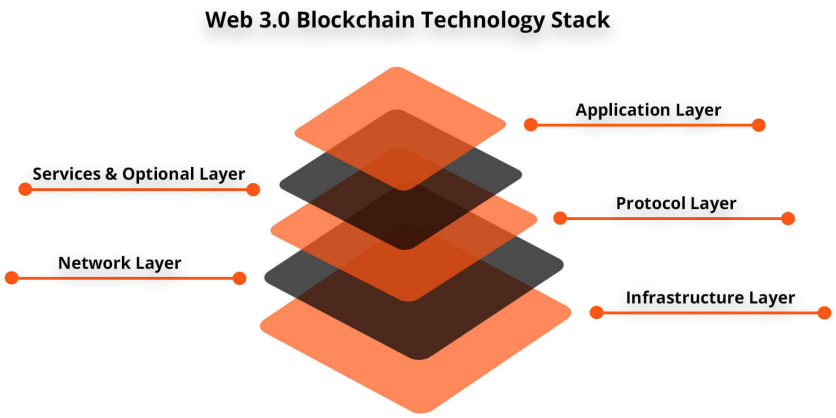


Figura 2.4: Camadas da Web3

Através destas camadas é possível replicar cada plataforma Web2 para uma Web3 cumprindo estes requisitos de descentralização.

Na Tabela 2.1 são apresentadas as vantagens e desvantagens da Web3 assentes nos seus pilares fundamentais.

Tabela 2.1: Vantagens e desvantagens da utilização da Web3

Vantagens da Web3	**Desvantagens da Web3**
Propriedade: dá a posse ao utilizador de todos os seus ativos digitais	**Acessibilidade**: Apesar de algumas funcionalidades serem gratuitas, o custo de transação é algo impeditivo para muitas pessoas
Resistência à censura: quando existe um problema com uma plataforma altamente centralizada, todo o seu conteúdo é perdido pelo caminho enquanto que na Web3 toda a informação é guardada na *Blockchain*	***User Experience***: A barreira de compreensão de entrada é demasiado alta. Os utilizadores necessitam de compreender o modo de funcionamento e critérios de segurança para a sua utilização

Identidade: em vez de criar uma conta para cada plataforma, na Web3 podemos fazer *login* com um identificador único baseado no nosso endereço.	**Infraestrutura centralizada**: atualmente grandes empresas controlam as nossas informações, e a transição para o modelo da Web3 é complexa e demorada.
Pagamentos nativos: elimina a necessidade de bancos como intermediários e o dinheiro vai de imediato para a carteira do utilizador de uma forma mais rápida.	

2.2 O que é a *Blockchain*?

A *Blockchain* [14] é uma estrutura de dados descentralizada que armazena uma cadeia de registos imutáveis chamados blocos. Cada bloco contém uma lista de transações e uma *hash* que corresponde a um resumo único dos dados do bloco anterior. Isso cria uma ligação criptográfica entre os blocos, tornando-os dependentes uns dos outros.

Através da Figura 2.5 é demonstrado o modo de funcionamento da *Blockchain*, desde o processo de criação da transação até ao seu término incluindo a sua verificação perante a cadeia distribuída. A cadeia de blocos é mantida por uma rede de computadores distribuídos, chamados de nós (*nodes*). Cada nó mantém uma cópia da cadeia de blocos e é responsável por validar as novas transações e adicioná-las à cadeia. Quando um novo bloco é adicionado à cadeia, também conhecida como *chain*, todos os nós da rede recebem e validam-o de forma a garantir a integridade dos dados.

O objetivo da *Blockchain* é permitir que a informação digital seja registada e distribuída, mas não editada. Assim corresponde a uma base para *ledgers* imutáveis, ou registos de transações que não podem ser alterados, apagados, ou destruídos. Atualmente a *Blockchain* é utilizada para a criação de várias criptomoedas, construção de aplicações, *NFTs* e *smart contracts*.

O que a *Blockchain* faz é permitir que os dados mantidos na sua rede sejam espalhados entre vários nós (*nodes*) em vários locais. Isto não só cria redundância, como também mantém a fidelidade dos dados aí armazenados, o que permite restabelecer a ordem exata e transparente de acontecimentos.

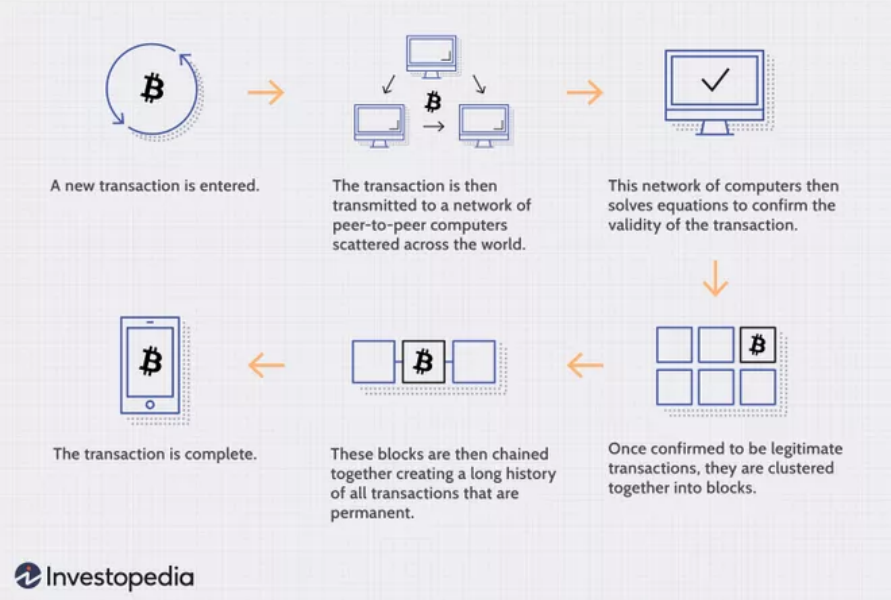


Figura 2.5: Representação do modo de funcionamento da *Blockchain* [14]

De forma a validar novas entradas ou registos para um bloco, a maioria do poder computacional da rede descentralizada tem de concordar com a mesma. Para evitar que maus atores validem más transações ou despesas duplas, as cadeias de bloqueio são asseguradas por um mecanismo de consenso como *Proof of Work (PoW)* ou *Proof of Stake (PoS)*. Estes mecanismos permitem o acordo mesmo quando não há um único nó no comando.

2.3 Mecanismos de Consenso

Os mecanismos de consenso PoW e PoS [15] são os dois principais mecanismos de consenso que as criptomoedas utilizam para verificar novas transações, adicioná-las à *Blockchain*, e criar novos *tokens*. O mecanismo PoW, pioneiro da Bitcoin, utiliza a mineração para atingir esses objetivos. O PoS - que é utilizado por Cardano, a *Blockchain* ETH2, e outros - utiliza este método para alcançar as mesmas verificações com um custo bastante inferior.

Estes algoritmos são importantes de forma a garantir que não existe o problema de *Double-Spending* [16] de forma a cumprir com a integridade e validade das transações.

- ***Proof of Work***: *Proof of Work* e a mineração são ideias intimamente relacionadas. A razão pela qual se chama *Proof of Work* é devido ao facto da rede requerer uma enorme quantidade de poder de processamento uma vez

que os nós da rede competem entre si para resolver um problema matemático complexo. O primeiro nó a resolver o problema é recompensado com uma pequena quantidade de moeda digital e adiciona-se o novo bloco de transações à cadeia.

O objetivo do mecanismo de consenso PoW é garantir que todos os nós da rede estejam a trabalhar para validar as transações e manter a integridade da cadeia. Isso é feito exigindo que os nós gastem um certo esforço computacional para resolver o problema matemático.

- ***Proof of Stake***: *Proof of Stake* reduz a quantidade de trabalho computacional necessária para verificar blocos e transações, uma vez que a criação de um novo bloco não depende da resolução de problemas matemáticos complexos como no caso do mecanismo PoW.

  Os proprietários oferecem as suas moedas como garantia colateral (*staking*) para a validação dos blocos e depois tornam-se *validators*. A rede seleciona um vencedor com base na quantidade de criptomoedas que cada *validator* tem na *pool* e no período de tempo que esteve presente - recompensando os participantes com maior investimento na criptomoeda nativa.

Tabela 2.2: Diferenças entre os métodos de consenso PoW e PoS

Proof of Work	***Proof of Stake***
Os criadores dos blocos são chamados de *miners*	Os criadores dos blocos são chamados de *validators*
Os participantes necessitam de comprar equipamento e utilizar uma grande quantidade de energia para se tornarem um *miner*	Os participantes têm de dispor moedas/-*tokens* para se tornarem *validators*
Segurança robusta devido ao requisito inicial caro dos equipamentos	Segurança através do controlo da comunidade
Miners recebem recompensa por bloco	*Validators* recebem os custos de transação como recompensa
Transações mais lentas devido ao tempo necessário de resolução de problemas matemáticos complexos	Transações mais rápidas e eficientes através do mecanismo de *staking*

A Tabela 2.2 visa mostrar as principais diferenças entre estes dois mecanismos de consenso utilizados para validar as transações e garantir o bom funcionamento da rede. Cada um destes mecanismos tem as suas vantagens e desvantagens, não sendo possível determinar qual dos dois deve ser adotado na construção futuras redes.

Embora os mecanismos de consenso PoW e PoS sejam os mais prevalentes na *Blockchain*, existem outros algoritmos e alternativas que derivam destes dois métodos principais como é o caso do mecanismo conhecido como *Delagated Proof of Stake (DPoS)* que corresponde a uma iteração do conceito do mecanismo PoS. No entanto, utiliza um sistema de votação de modo a eleger delegados que são responsáveis por validar as transações e produzir novos blocos. Os delegados eleitos para um bloco, não podem ser os mesmos para o próximo, aumentando assim o processo democrático de eleição para a validação dos mesmos. No final de cada bloco validado os delegados eleitos recebem as suas taxas de transação que depois serão compartilhadas com os utilizadores que agruparam o seu voto sendo estas repartidas com base na aposta no delegado [17].

Cada método possui características únicas e é utilizado para atender diferentes necessidades, como segurança, escalabilidade e eficiência. A variedade de métodos reflete a diversidade de requisitos e objetivos das *Blockchains*.

2.4 *Smart Contracts*

Um *smart contract* é um contrato que se executa quando certas condições estão estabelecidas, sendo os termos do acordo entre comprador e vendedor escritos diretamente em linhas de código. O código e os acordos nele contidos existem através de uma rede distribuída e descentralizada na *Blockchain*. O código controla a execução, e as transações são rastreáveis e irreversíveis.

Contratos inteligentes permitem a realização de transações e acordos de confiança entre partes díspares e anónimas, sem necessidade de uma autoridade central ou mecanismo de execução externo [18].

De seguida, são apresentados os benefícios dos *smart contracts* adaptados do seguinte *website* [19]:

- **Rapidez, eficiência e precisão:** Uma vez satisfeita uma condição, o contrato é executado imediatamente. Dado que os contratos inteligentes são digitais e automatizados, não há burocracia a processar nem tempo gasto a reconciliar erros que muitas vezes resultam do preenchimento manual de documentos.
- **Confiança e transparência:** Como não há terceiros envolvidos, e porque os registos encriptados das transações são partilhados entre todos os participantes, não existe a necessidade de questionar se a informação foi alterada para benefício pessoal.
- **Segurança:** Os registos das transações são encriptados, o que os torna muito

difíceis de piratear. Além disso, porque cada registo está ligado aos registos anteriores e subsequentes num livro de registos distribuído (*ledger*), os *hackers* teriam de alterar toda a cadeia para alterar um único registo.

- **Poupança:** Contratos inteligentes eliminam a necessidade de intermediários para tratar de transações e, por extensão, os atrasos e taxas e custos associados.

2.5 *Ethereum standards*

De acordo com o fundador da *Ethereum*, Vitalik Bueterin, esta *network* foi construída para combater as limitações das redes *Blockchain* de primeira geração, como a Bitcoin [20] uma vez que este tipo de *Blockchain* foi criado com o propósito de alterar o sistema monetário através da eliminação do intermediário nas transações efetuadas. Em contrapartida a *Blockchain* Ethereum fornece mecanismos de criação de *Decentralized Applications (dAPPs)* e *smart-contracts* utilizados na *Blockchain* utilizando a sua moeda nativa *Ether* que revolucionam a forma como aplicações e transações são efetuadas a um custo substancialmente menor com uma velocidade bastante superior à *Blockchain* de primeira geração [21].

Esta fornece também um serviço de desenvolvimento de *tokens* assentes na sua rede, no entanto, existe a necessidade de estipular algumas regras de forma a que os diferentes *tokens* possam ser usados por todos de forma compatível entre os diversos ecossistemas existentes. Daí surge o termo de Ethereum Request for Comments (ERC) [22].

Este termo corresponde a um conjunto de diretrizes e regras em que são definidos os métodos e comportamentos que os desenvolvedores têm de seguir de forma a utilizarem o ecossistema Ethereum e também para que outros sistemas (aplicações, *smart-contracts*) consigam interagir com os seus métodos e funcionalidades.

2.5.1 ERC-20

Uma vez que a *Blockchain* Ethereum permite o desenvolvimento de *tokens* na sua rede, o padrão ERC-20 introduz o mecanismo de criar *fungible tokens* na sua *Blockchain*, seguindo um conjunto de regras. O seu objetivo é simplificar o processo de criação de *tokens* idênticos de forma a que o valor de uma moeda seja sempre igual ao valor de outra e intercambiável entre si.

De acordo com o artigo divulgado pela bdtask.com [23] este padrão permite aumentar a segurança da *Ethereum Blockchain Network* na medida em que introduz um

standard facilitando a interoperabilidade entre outras aplicações e sistemas reduzindo deste modo erros de implementação. Os tokens ERC-20 são criados através de um *smart-contract* que define as regras para a criação, distribuição e transferência dos *tokens*. Eles são compatíveis com outros contratos e dApps assentes na Ethereum Virtual Machine.

2.5.2 ERC-721

Antes do *standard* ERC-721, a maioria dos *tokens* presentes nas diversas *Blockchains* funcionavam como moeda, reserva de valor como o ouro ou prata, stock ou até mesmo património. Contudo através deste standard, essa filosofia mudou. De acordo com o blog erc721.org [24] o conceito de ERC-721 é definido por:

> *"ERC-721 is a free, open standard that describes how to build non-fungible or unique tokens on the Ethereum Blockchain. While most tokens are fungible (every token is the same as every other token), ERC-721 tokens are all unique. Think of them like rare, one-of-a-kind collectables."*

E, foi através deste *standard* que surgiram os Non-Fungible Tokens (NFT). NFTs correspondem a ativos criptográficos que estão assentes na *Blockchain* com códigos de identificação únicos e metadados que permitem distinguir e serem distinguidos dos restantes.

Ao contrário das criptomoedas, os NFT não podem ser negociados ou trocados em equivalência. Isto difere dos *fungible tokens* como as criptomoedas, que são idênticas umas às outras e, portanto, podem servir como meio para transações comerciais. De seguida apresentamos alguns dos benefícios dos NFT [25]:

- A conversão de um ativo físico para um ativo digital é elaborada eliminando a necessidade de terceiros para que esta operação ocorra;
- Aumenta a segurança, uma vez que todos os ativos são únicos com propriedades distintas que podem ser sempre verificáveis e identificáveis através da *Blockchain* como por exemplo passaportes dado que é algo único;
- É possível visualizar todo o processo pelo qual aquele NFT passou inclusive donos uma vez que está na *Blockchain*;
- O processo de divisão de um ativo digital é mais fácil do que um ativo físico. Como exemplo um quadro pode ter sido feito recorrendo a vários artistas e

pode não ter só um proprietário, pode ter múltiplos - a este processo chama-se *tokenization*.

Atualmente os NFTs são utilizados maioritariamente como forma de colecionáveis [26]. Os utilizadores compram este tipo de arte por um elevado número de razões: suportar o artista que gostam, como forma de investimento na esperança que o preço suba com o decorrer do tempo fazendo assim algum lucro sobre o investimento, como forma de *trading* entre outras razões.

Contudo, os NFTs existem muito para além da arte digital tendo também a seguinte utilidade: [27]

- Utilizados como personagens/*skins* num determinado jogo assente na *Blockchain*;
- Eventos exclusivos para os detentores do NFT;
- Como forma de representar a propriedade de ativos elevados como por exemplo um relógio ou um quadro, semelhante ao da Figura 2.6 que corresponde a um dos NFTs mais caro alguma vez transacionado.

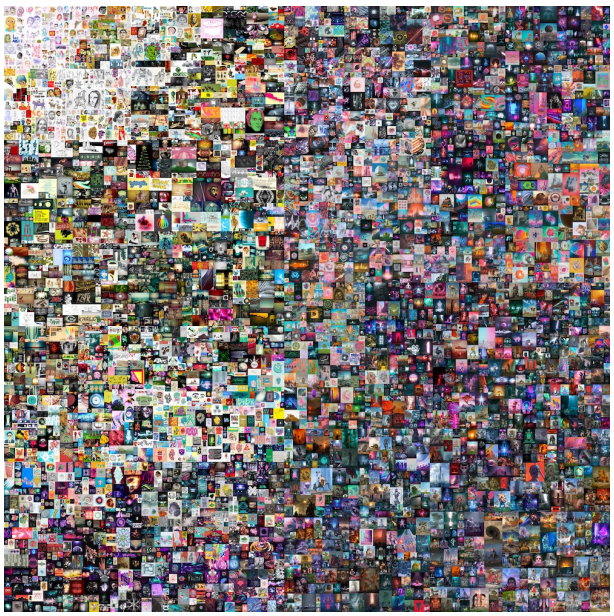


Figura 2.6: *NFT: Beeple, Everydays—The First 5000 Days $69 million, March 2021, Christie's* [28]

2.5.3 ERC-1155

O padrão ERC-1155 [29] ao contrário dos referidos acima permite a execução de *smart contracts* em que sejam utilizados quer *fungible tokens* (como por exemplo Bitcoin (BTC) e Ethereum (ETH)), como também *non fungible* como é o caso dos NFTs.

Antes do surgimento deste novo protocolo, na eventualidade de se querer transferir um *token* ERC-20 [30] juntamente com um NFT assente no *standard* ERC-721 como por exemplo um CryptoKittie [31], simplesmente não era possível fazer estas operações de uma só vez o que era algo bastante ineficiente e traria custos adicionais devido à execução de múltiplas transações.

No entanto o padrão ERC-1155 resolve este problema na medida em que aglomera estes dois *standards* (ERC 20 e 721) num só permitindo efetuar uma única transação contento estes dois tipos de ativos digitais.

Para além de permitir a transferência de vários tipos de *tokens* de uma só vez, tem ganhos de performance substancialmente superiores aos antigos padrões e ainda: [32]

- Suporta um número infinito de *tokens*, em contraste com ERC-20 e ERC-721, que requerem um novo *smart contract* para cada tipo de *token*;
- Suporta não só *tokens* fungíveis e não fungíveis, mas também *tokens* semi-fungíveis. Os tokens semi-fungíveis são como bilhetes para concertos. São intercambiáveis (podemos trocar por outro) e podem ser vendidos por dinheiro antes do espetáculo (fungíveis). Mas depois do seu término perdem o seu valor e tornam-se colecionáveis (não-fungíveis);
- Tem uma função de transferência segura que permite recuperar os *tokens* se forem enviados para o endereço errado, ao contrário do ERC-20 [33] [34] [35];
- Elimina a necessidade de "aprovar" contratos simbólicos individuais separadamente, o que significa assinar menos transações e consequente existe um gasto inferior em taxas por cada transação.

2.6 *Cryptocurrency Wallets*

Uma *cryptocurrency wallet* é um dispositivo ou programa que armazena as chaves criptográficas e permite o acesso aos nossos fundos [36]. As carteiras dispõem de

uma chave pública (o endereço da carteira) e de chaves privadas necessárias para assinar transações criptográficas. Qualquer pessoa que conheça a chave privada pode controlar as moedas associadas a esse endereço. Existem vários tipos diferentes de carteiras, cada uma com suas próprias características e níveis de segurança e estão divididas em duas grandes categorias de carteiras: *custodial wallets* e *non-custodial wallets*, também conhecidas como *self-custody wallets*.

2.6.1 *Custodial Wallet*

Uma *custodial wallet* é uma carteira em que as chaves privadas são mantidas por terceiros, sob custódia, para gerir a sua conta em seu nome. Usufruir de uma *custodial wallet* normalmente significa que é necessário passar por um processo de *Know Your Customer (KYC)* e enviar vários documentos para concluir o processo. O KYC é uma maneira de identificar e confirmar que um cliente é quem ele diz ser, através do estabelecimento da sua identidade, com o objetivo de compreender a natureza das suas atividades e certificar-se de que a origem dos fundos é legítima de modo a avaliar os riscos de lavagem de dinheiro associados aos clientes. Este processo visa proteger as instituições financeiras contra fraude, corrupção, lavagem de dinheiro e financiamento do terrorismo [37].

2.6.2 *Self-custody Wallet*

Uma carteira de criptomoedas não custodial é uma carteira em que apenas o titular possui e controla as chaves privadas. Normalmente, possui uma frase de recuperação de doze palavras que permite aceder e gerir os fundos e a carteira. Isso deve ser feito com segurança e nunca mostrado a ninguém, uma vez que caso essa cifra seja descoberta, a conta poderá ficar em risco [37].

2.6.3 *Shared Custodial Wallets*

Este tipo de carteira corresponde a uma evolução da carteira custodial, na medida em que ao invés da chave privada ser exclusivamente guardada por entidades terceiras, esta é compartilhada entre diversas entidades ou indivíduos eliminando deste modo a centralização da responsabilidade pela chave privada da carteira.

A principal característica das *Shared Custodial Wallets* [38] é que são necessárias as várias chaves privadas para realizar transações. Isso significa que, para autorizar uma transação, é necessário o consenso de múltiplas partes envolvidas. Através desta abordagem é aumentada a segurança e proteção dos ativos criptográficos uma vez

que nenhuma entidade tem o controlo total dos fundos. No entanto, este tipo de carteiras acarreta complexidades operacionais e dependência com as partes envolvidas [39].

Capítulo 3

Estado da Arte

Este capítulo, visa explorar diversas plataformas de *e-commerce* existentes no mercado incluindo os seus meios de pagamento. Serão também analisados os diferentes provedores de pagamento envolvendo criptomoedas que correspondem a novas alternativas de pagamentos para as plataformas de *e-commerce*.

Este capítulo está dividido em duas secções: a primeira secção é composta pela secção de plataformas de *e-commerce* com o propósito de elaborar uma comparação entre os diversos métodos aceites por estas plataformas. E, na segunda iremos abordar os diferentes mecanismos de pagamento existentes para as plataformas de *e-commerce*.

3.1 Estudo de plataformas de *e-commerce*

Nesta secção será feita uma análise de diferentes plataformas de *e-commerce* tendo em conta alguns fatores nomeadamente a sua performance Search Engine Optimization (SEO), número de *plugins* aceites, planos de subscrição, capacidade de aceitar criptomoedas, facilidade de uso entre outros. As plataformas a analisar serão as seguintes: BigCommerce, Shopify, WIX, Shopwired, Zyro, Shift4shop, WooCommerce, SquareSpace e Magento.

De acordo com o Shopify, *e-commerce* pode ser definido da seguinte forma: [40]

> *"Ecommerce, also known as electronic commerce or internet commerce, refers to the buying and selling of goods or services using the internet, and the transfer of money and data to execute these transactions. Ecommerce is often used to refer to the sale of physical products online, but it can also describe any kind of commercial transaction that is facilitated through*

the internet."

correspondendo a uma maneira simples de vender produtos pela internet sem a preocupação da infraestrutura e das funcionalidades necessárias para o funcionamento da loja.

Existem três tipos de plataformas de *e-commerce* [41]:

- ***Open-source*** [42]: correspondem a soluções nas quais podemos modificar completamente todos os aspetos do código de acordo com as nossas preferências e objetivos. É popular nas organizações tecnológicas onde estas querem um controlo total sobre o seu ambiente *e-commerce*. No entanto, temos também de obedecer a regras e a marca também é responsável por certos critérios e obrigações nomeadamente:
  - *Payment Card Industry Data Security Standard (PCI)*;
  - Hospedagem web;
  - *Updates* e *patches* da plataforma;
  - Integração com outros sistemas;

  Todas estas obrigações fazem com que todo este processo seja demasiado caro e com elevados custos técnicos;
- **SaaS**: estas soluções removem grande parte da complexidade de gerir um negócio online. Em vez de construir e desenvolver uma solução personalizada ou uma solução *open-source*, aluga-se a plataforma.
  As atualizações de produtos, segurança, alojamento, conformidade com PCI e todas as outras tarefas que vêm com a gestão do próprio *software* são geridas pelo fornecedor de SaaS;
- ***Headless Commerce***: solução que armazena, gere, e entrega conteúdos sem a parte de *frontend*. Contém apenas as funcionalidades na parte de *backend*, sem qualquer interface visual;

Este estudo apenas irá consistir na análise de soluções *open-source* e SaaS, uma vez que correspondem à maioria de soluções existentes no mercado e acessíveis à maioria dos comerciantes[43].

3.1.1 BigCommerce

Esta plataforma tem um motor de pesquisa de produtos tornando-o ideal para grandes marcas de retalho, inclui ainda opções de design e personalização do website.

A sua interface permite personalizar a sua loja *online* fazendo uso dos templates personalizáveis para desenhar, vender e comercializar os seus produtos [44]. Fornece mais de 65 soluções de pagamento [45], mais de 600 aplicações com uma banda larga ilimitada e zero taxas de transação [46][47].

3.1.2 Shopify

Corresponde uma das maiores plataformas em termos de vendas [48]. Esta dispõe de funcionalidades de *post-purchase*, *1-click upsells* opções extra de pagamentos e recursos de marketing. Possui uma interface de *drag and drop* oferecendo aos seus utilizadores a capacidade de personalização do seu *website*. Dispõe de um *marketplace* onde existem muitas aplicações que podemos adicionar ao nosso *website* com o objetivo de o melhorar e aumentar as vendas, contudo muitas dessas aplicações não são grátis [49].

Dispõe de modelos profissionais, integrações com *gateways* de pagamento, ferramentas de SEO e gestão de *stock*, esta ferramenta é uma escolha popular para lojas de todos os tamanhos. No entanto, Shopify pode ser mais caro do que algumas outras opções de plataformas de e-commerce, especialmente para lojas de *e-commerce* de menor escala.

3.1.3 WIX

Wix é uma plataforma de e-commerce estabelecida, mas comparativamente com os seus concorrentes ainda está um pouco atrás em termos de funcionalidades. Embora ofereça 20 GB de armazenamento, esse espaço pode ser limitado para lojas de e-commerce que estão a crescer rapidamente. À medida que a loja expande seus produtos e aumenta o número de visitas ao seu site, 20 GB podem não ser suficientes para gerir todos os recursos da loja.

Existe também a falta de funcionalidades de *upselling* e automação que são fundamentais no papel da plataforma de *e-commerce*. Contudo, esta plataforma dispõe de temas únicos e de funcionalidades de *drag and drop* para a personalização dos templates que oferecem [50].

3.1.4 ShopWired

Esta empresa fornece um serviço cujo preço e performance são semelhantes ao Shopify e BigCommerce com uma gama de funcionalidades competitiva perante estas duas alternativas. Uma das funcionalidades que é destacada logo desde o início são

as funcionalidades Business-to-Business (B2B) que oferece como por exemplo gestão de fornecedores, preços de contratos, faturação, direitos de produtos, gestão de contratos, entre outros [51].

3.1.5 Zyro

É uma plataforma SaaS direcionada para negócios de baixa dimensão. Apesar de ser relativamente nova neste mercado, corresponde a um produto derivado da empresa de hospedagem Hostinger. Para além do seu *website builder* a plataforma oferece uma ampla gama de funcionalidades usando Inteligência Artificial de forma a ajudar os seus clientes. Entre estas incluem-se: a) criação de logótipos e ícones; b) remoção do fundo de imagens; c) criação do nome da empresa e *slogan*; d) sugestão de títulos de *post* e o seu conteúdo;

3.1.6 Shift4Shop

Esta plataforma tem funcionalidades práticas como armazenamento ilimitado e sem taxas de transação [52]. É possível utilizar o seu Point of Sale (POS) e suportam vendas *multi-channel*. É uma solução com uma curva de aprendizagem mais acentuada devido à existência da necessidade de um conhecimento técnico sobre o seu modo de funcionamento. Um dos seus pontos fortes é o envio, uma vez que fornecem soluções avançadas sem ser necessário a utilização de nenhuma API [53].

3.1.7 WooCommerce

Plataforma que tenta competir diretamente com BigCommerce e Shopify. Funciona através do WordPress de forma a tornar um website numa loja de *e-commerce*. Suporta também extensões que permitem integrar novos *gateways* de pagamento, redes sociais, *email marketing*, *1-click selling* e envio. Esta plataforma possui problemas de escalabilidade, uma vez que muitos utilizadores notam que com o aumentar da loja e dos seus produtos, a performance do seu website também diminui [54].

Caso o utilizador tenha conhecimento sobre o funcionamento do WordPress, esta plataforma funciona de forma semelhante, caso contrário ainda tem uma curva de aprendizagem que pode ser uma limitação na escolha desta plataforma para hospedar a sua loja.

3.1.8 SquareSpace

Plataforma que oferece *drag and drop* para a personalização do website através de um editor online melhorando deste modo a simplicidade de criação de um website. Existe uma taxa aplicada a cada transação efetuada na plataforma. No caso da escolha de um plano mais caro esta taxa é eliminada. No entanto apenas o plano mais caro vem com funcionalidades de recuperação do carrinho abandonado, venda de subscrições e descontos flexíveis. Apesar de ser uma plataforma em que é possível fazer *drag and drop*, a personalização do website não é algo trivial. Suporta a integração de um serviço de envio com as maiores transportadoras existentes (United Parcel Service (UPS), FedEx, United States Postal Service (USPS)). Uma das grandes desvantagens desta plataforma são os processadores de pagamento e aplicações de terceiros em que apenas suportam Stripe e Paypal e existem uma gama bastante limitada de aplicações aceites pela plataforma.

3.1.9 Magento

Corresponde a uma plataforma *open-source self-hosted* que dispõe de diversos *plugins* com o propósito de aumentar as soluções e as funcionalidades da loja. Recentemente adicionaram uma ferramenta que permite fazer *drag and drop* de novos elementos para o website de modo a personalizar e editar de forma mais simples o website [55]. Como se trata de uma solução open-source, é possível personalizar todo o conteúdo do nosso website, ficando apenas restringido à capacidade de desenvolvimento do programador.

3.1.10 Análise comparativa entre as plataformas

Esta secção apresenta uma análise comparativa entre as diversas plataformas de *e-commerce* enumeradas anteriormente tendo em conta alguns aspetos na ótica do utilizador, como por exemplo a sua facilidade de uso e o suporte ao cliente, mas também aspetos mais técnicos como por exemplo a capacidade de aceitar criptomoedas e os planos de subscrição oferecidos. Alguns dos dados apresentados foram adaptados do seguinte website (Facilidade de uso, Performance SEO, Número de Plugins, Temas grátis) [56].

Tabela 3.1: Comparação entre as diferentes plataformas de e-commerce

Plataforma	Facilidade de uso	Performance SEO	Número de Plugins	Temas grátis	Suporte Crypto	Planos
BigCommerce	4.8/5	4.5/5	1000	12	✅	**Standard**-29.95$/mês **Plus**-79.95$/mês **Pro**-299.95$/mês **Custom/Enterprise**
Shopify	4.9/5	3.9/5	5000	9	✅	**Básico**-24€/mês **Shopify**-69€/mês **Avançado**-289€/mês **Shopify Plus**-2000€/mês
Wix	4.2/5	3.9/5	700	72	✅	**Basic**-20€/mês **Ilimitado**-30€/mês **VIP**-44€/mês
ShopWired	4.5/5	4.3/5	72	20	✅	**Pro**-30$/mês **Avançado**-70$/mês **Premium**-130$/mês **Enterprise**-250$/mês
Zyro	3.7/5	3.3/5	30	50	❌	**Basic**-$2.90/mês. **Unleashed**-$3.90/mês **eCommerce**-$9.90/mês **eCommerce Plus**-$14.90/mês
Shift4Shop	4.3/5	3.0/5	250	50	✅	**Grátis** **Basic**-29$/mês **Plus**-79$/mês **Pro**-229$/mês **Enterprise**-1999$/mês
WooCommerce	3.3/5	3.1/5	250	1000	✅	Custos *Open-Source*
SquareSpace	3.8/5	3.5/5	10	14	❌	**Personal**-15€/mês **Business**-24€/mês **Commerce**-28€/mês **Commerce Plus**-42€/mês
Magento	2.2/5	2.8/5	3000	1	✅	Funciona sob escalões de receita em que o mais baixo custa 22000€ /ano

Como se pode verificar pela Tabela 3.1, uma grande parte das soluções evidenciadas, suporta pagamentos em criptomoedas, quer através de *plugins* ou até mesmo de

soluções embutidas nas plataformas.

Com o desenvolvimento do ecossistema das criptomoedas, muitas destas plataformas de *e-commerce* têm parcerias com alguns processadores de pagamento de forma a evidenciar e dar a conhecer a sua solução aos seus comerciantes de um modo mais acessível. Além disso, as plataformas de *e-commerce* têm investido no desenvolvimento da sua própria infraestrutura de modo a aceitar criptomoedas, além de oferecer suporte para integrações com diferentes processadores de pagamento. Isso permite que as lojas online aceitem pagamentos em diferentes criptomoedas e ofereçam novas opções aos seus clientes [57].

Após a análise das soluções, a escolha para o projeto incide sob a plataforma Shopify uma vez que dispõe de uma ampla gama de recursos para construir, gerir e integrar novas soluções na plataforma de *e-commerce* bem como fornece toda a documentação necessária para a construção desse tipo de soluções.

3.2 Estudo de métodos de pagamento em plataformas de *e-commerce*

Atualmente, nas plataformas de *e-commerce*, quando procedemos ao pagamento existe uma ampla escolha, derivado das soluções existentes no mercado mas também das parcerias acordadas com a plataforma.

Entre os modos de pagamento mais tradicionais, como por exemplo através de cartão de crédito ou débito, transferência bancária ou depósito direto, muitas das plataformas evidenciadas previamente suportam novos tipos de pagamento, como por exemplo:

- **Carteiras eletrónicas** (Amazon Pay [58], Apple Pay [59]): este tipo de carteira permite aos utilizadores pagarem de uma forma mais rápida, simples sem sair do serviço ao qual estão associados e sem terem de preencher os seus dados sempre que querem fazer uma compra. É um tipo de conveniência que melhora a experiência do utilizador.
- **Buy Now, Pay Later** [60]: correspondem a sistemas que oferecem empréstimos de curto prazo aos seus clientes no momento da compra. Caso o montante não for pago até à data imposta pelo provedor, são aplicadas taxas que acrescem ao valor total da compra.

Além destes novos tipos de pagamentos, existem outras formas, nomeadamente através de *Payment gateways* que correspondem a interfaces orientadas para o consu-

midor usadas para reunir informações de pagamento, como se pode verificar pela Figura 3.1. Este tipo de alternativas, permite realizar transações mais seguras, rápidas evitando que o comerciante se preocupe com questões de segurança e de burocracia [61].

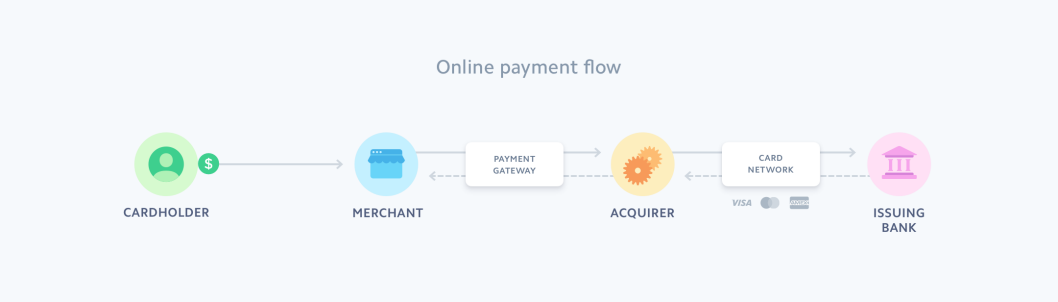


Figura 3.1: Representação do modo de funcionamento de *Gateways* de pagamento [62]

Na próxima secção, iremos abordar algumas destas soluções e características que diferem dos seus competidores, como por exemplo as funcionalidades oferecidas e o preço praticado.

3.2.1 Stripe

Stripe [63] é uma das plataformas de pagamento mais conhecida com uma grande versatilidade e ampla gama de funcionalidades. Permite a aceitação de vários tipos de pagamentos. Relativamente ao preço, de acordo com o plano integrado, a empresa cobra cerca de $2.9\% + 0.25€$ por cada compra bem sucedida, não havendo um modelo de subscrição mensal. No entanto esta solução não é aconselhável para negócios de alto risco, uma vez que a plataforma dispõe de mecanismos de bloqueio de transações [64].

A Stripe tem a possibilidade de funcionar como uma *wallet* em que é possível comprar e vender criptomoedas. Existe também a possibilidade de processar pagamentos para moedas FIAT globalmente através de uma integração no processo [65].

3.2.2 Square

A ferramenta Square [66] é utilizada no mercado de processamento de pagamentos não só para lojas *e-commerce* mas também para lojas físicas fazendo uso do seu sistema de POS.

Este serviço oferece também a possibilidade de criar uma loja online de forma a

integrar os seus métodos de pagamento. Relativamente aos custos, existem 3 planos diferentes: **grátis**, **plus** e **premium**. O modelo grátis não funciona como subscrição e apenas cobra as taxas por cada transação efetuada. O modelo plus tem uma mensalidade de 29€/mês com a adição ainda das taxas de cada transação. Por fim o modelo *premium* para além dos custos transacionais é adequado tendo em conta as necessidades da empresa, ou seja, é personalizado. As taxas transacionais são fixas havendo as seguintes categorias [67]:

- Cartão presente: 2.6% + 0.10€ /transação
- Cartão não presente: 2.9% + 0.30€ /transação

Para além dos pagamentos na loja *online*, existe a possibilidade de criar um terminal virtual de forma a receber pagamentos e criação de *links* como forma de pagar por um produto. É também suportado o mecanismo de *Buy Now, Pay Later* denominado AfterPay em que os utilizadores pagam o valor estipulado por 4 vezes sem juros durante um período de 6 semanas. Este tipo de transação tem um custo de 6% + 0.30€ por transação efetuada [68].

Esta empresa para além destes serviços enunciados, tem ao dispor uma ampla gama de funcionalidades estipulada no seu website [67]. Contudo, uma desvantagem desta plataforma é o facto de não fornecer nenhum serviço com criptomoedas, e também as taxas fixas praticadas podem ser demasiado altas para os negócios de larga escala.

3.2.3 PayPal

O PayPal é o provedor de pagamentos mais conhecido em todo o mundo com mais de 430 milhões de utilizadores [69]. Dado que é utilizado por muita gente, pode ser benéfico para pequenos negócios devido ao conhecimento e segurança dada pela ferramenta. No entanto as suas taxas de transação não são das mais baixas, rondando cerca de 3.49% + 0.49€ por cada transação, o que pode ser algo desafiante para pequenos negócios. É suportada pelas maiores empresas de cartões de crédito como Visa, Mastercard, Discover e American Express [70].

Em relação à ótica do comerciante, o Paypal oferece a possibilidade de efetuar pagamentos via QR-Code que mais uma vez está sujeito a taxas: para valores acima de 10€: 1.9% + 0.10€ e para valores inferiores a 10€: 2.4% + 0.05€ por transação efetuada.

Atualmente o Paypal permite comprar, transferir e vender criptomoedas nomeadamente: BTC, BTC Cash, Ethereum e Litecoin. Contudo esta funcionalidade apenas está disponível em território Americano e apenas para contas pessoais, ou seja, caso

tenhamos uma conta de negócio não seremos elegíveis para usufruir desta funcionalidade [71].

Outro aspeto fornecido pelo PayPal é o facto de garantir reembolsos a 100%, algo benéfico para o cliente e comerciante, e também oferecem um nível extra de segurança e prevenção de fraude, o que também ajuda o comerciante a sentir seguro caso adote esta solução.

3.2.4 Amazon Pay

Para além de ser considerada uma *digital wallet*, a Amazon Pay é uma *gateway* de pagamentos projetada para comerciantes e compradores da Amazon. Tem uma integração com as maiores plataformas de *e-commerce* como BigCommerce, Magento ou Zuora.

A Amazon Pay dispõe de recursos e ferramentas que atraem clientes e os incentivam a comprar mais, simplificando o processo de compra e melhorando o envolvimento e a fidelidade do cliente por meio de detalhes de pagamento salvos para *checkouts* mais rápidos. Na aplicação mobile existe uma taxa fixa de 2.9% + 0.30€ por cada transação. Em transações internacionais, pode-se aplicar uma taxa diferente de cerca de 3.9% + 0.30€.

Esta solução, como vem da empresa Amazon, traz o seu nome como forma de confiança e segurança para os clientes que a usam e é algo conveniente para as pessoas que utilizam o serviço da Amazon, uma vez que poderiam utilizar este modo de pagamento em outras plataformas de *e-commerce* sem a necessidade de criar conta em outros provedores. Como desvantagem, o utilizador é obrigado a ter uma conta na plataforma da Amazon e, caso as regras estipuladas pela empresa sejam quebradas, este poderá ficar com o acesso restringido à mesma e, é uma solução que não aceita criptomoedas como meio de pagamento [72].

3.2.5 checkout.com

A *gateway* de pagamentos checkout.com corresponde a uma *fintech* internacional que processa diferentes modos de pagamento *online* em várias moedas. Fácil de integrar ao seu site, o checkout.com é uma ferramenta a considerar se desejamos expandir os negócios para novos mercados e regiões, uma vez que ajudam a adaptar sua solução para cada mercado com suas perceções e conhecimentos locais. Esta solução oferece um novo meio de pagamento através de *stablecoins* que são uma parte fundamental do mercado de criptomoedas, ajudando os investidores a negociar dentro e fora de moedas digitais rapidamente sem ter que passar por bancos.

3.2.6 Revolut

Através da plataforma, temos acesso à conta Revolut Merchant que nos permite instalar um *plugin* da ferramenta nas diversas plataformas de *e-commerce*. Oferecem também a possibilidade de criar um *link* de forma a receber o pagamento, ou até mesmo a criação de um QR-CODE. Existe também a possibilidade de adquirir um leitor de cartões Revolut, onde se podem efetuar pagamentos por 39€. Os fundos ficam disponíveis dentro de 24 horas na conta *business* Revolut [73].

Relativamente ás taxas estas diferem entre *online* e de forma presencial. Caso o pagamento ocorra online, é cobrado 1% + 0.20£ caso o cartão seja do UK, caso contrário cobram 2.8% + 0.20£. No caso da compra ser presencial, é taxado 0.8% + 0.02£ no caso de ser um cartão do UK e nos restantes cobram cerca de 2.6% + 0.02£. Existem também planos em que estas taxas podem variar de acordo com o plano escolhido (quando mais alto o plano mais baixo serão as taxas e mais benefícios traz) [74].

Infelizmente, neste momento não é possível efetuar pagamentos com criptomoedas diretamente com o cartão Revolut. No entanto, é possível transferir criptomoedas na plataforma Revolut para os seus amigos e transferir Bitcoin para carteiras externas.

3.2.7 *Cryptocurrency Gateways*

Aquando da decisão sobre a receção de pagamentos em criptomoedas, é necessário decidir se os queremos receber em criptomoedas ou apenas pretendemos que os clientes consigam pagar em criptomoedas e proceder à sua conversão para moeda fiduciária (FIAT).

No caso de pretendermos guardar e usar as criptomoedas provenientes do negócio ou uso pessoal, então necessitamos de uma *crypto wallet*. No caso de querer aceitar pagamentos em criptomoedas e convertê-los para moeda FIAT então iremos necessitar de uma *crypto payment gateway*. Algumas soluções são, por exemplo, Coinbase e BitPay que têm ambas as funcionalidades: carteira e *gateway*.

As *crypto gateways* funcionam de forma semelhante às *crypto wallets*, no entanto têm a funcionalidade de conversão (que as carteiras não têm). Geralmente vêm com uma carteira associada de forma a guardar os fundos até que o utilizador escolha convertê-los para uma moeda FIAT à sua escolha e proceda à transferência da mesma para a sua conta bancária.

Por outro lado, a utilização deste tipo de *gateways* trás ao negócio uma terceira pessoa, algo que o processo das criptomoedas tenta eliminar. Este novo participante

cobra uma taxa de conversão de *crypto* para moeda fiduciária e geralmente esta taxa pode chegar aos 5% (ao contrário das transações em *crypto* que rondam os 1% por transação).

3.2.7.1 BitPay

BitPay é um processador de pagamentos que foi criado com o objetivo dos negócios aceitarem Bitcoin como meio de pagamento. Através da Figura 3.2, conseguimos verificar o modo de funcionamento desta solução em que os clientes fazem o pagamento em Bitcoin ou outra criptomoeda suportada pelo BitPay e, de seguida, converte-se esse valor para moeda fiduciária ou o montante é guardado na *crypto wallet* do negócio.

Ou seja o BitPay recolhe e deposita todos os pagamentos processados desde o dia útil anterior diretamente no seu banco ou carteira criptográfica, de acordo com as suas preferências de liquidação [75].

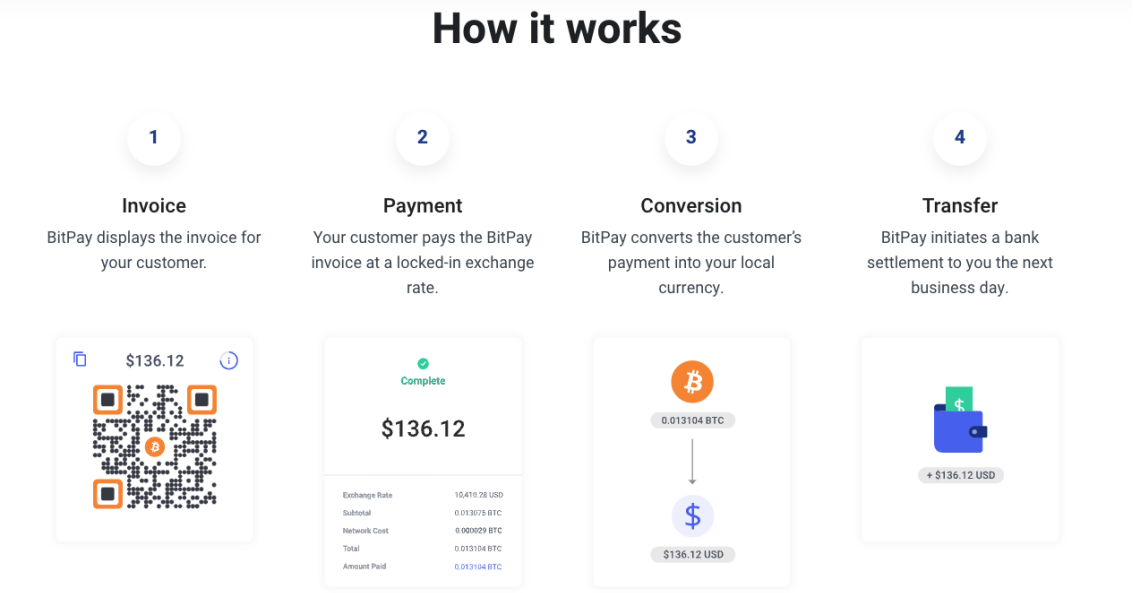


Figura 3.2: Representação do modo de funcionamento da solução Bitpay [76]

A empresa suporta uma ampla gama de criptomoedas. Para além destas pode suportar também tokens ERC20 presentes na *Blockchain* Ethereum. Relativamente aos custos, é cobrado 1% por cada transação[77]. No caso de se proceder à troca da criptomoeda para uma moeda FIAT, a taxa de câmbio é fixada no momento da transação de forma a proceder à sua troca no dia seguinte.

Em termos de integrações, este processador de pagamentos possui *plugins* para al-

gumas das mais conhecidas plataformas de *e-commerce open-source* como é o caso do WooCommerce e também possui soluções integradas para as plataformas de *e-commerce* como por exemplo o Shopify e Wix.

3.2.7.2 Coinbase Commerce

O Coinbase Commerce é uma plataforma que permite que estabelecimentos comerciais de qualquer lugar no mundo aceitem pagamentos com criptomoedas de forma totalmente descentralizada. Isso significa que todos os fundos recebidos são única e totalmente controlados pelo comerciante. O Coinbase Commerce não tem acesso aos fundos que o comerciante guarda. Infelizmente, isso também significa que se houver a perda da chave *master* de 12 palavras, não há como a Coinbase ajudar a recuperar a conta. Esta solução cobra uma taxa de processamento de pagamento fixa de 1% pelas transações. [78]

Atualmente oferecem as seguintes criptomoedas como forma de pagamento: Bitcoin, Bitcoin Cash, DAI, Ethereum, Litecoin, Dogecoin e USD Coin.

Esta plataforma fornece integração para 2 das maiores plataformas de *e-commerce* nomeadamente Shopify e WooCommerce, contudo também fornece instrumentos e documentação para incluir as suas funcionalidades na nossa loja através de uma API [79].

3.2.7.3 NowPayments

Ferramenta que permite aceitar pagamentos em criptomoedas quer nas plataformas de *e-commerce*, quer na loja física através de um POS. Fornece integração com diversas plataforma de *e-commerce* como: Shopify, WooCommerce, openCart, Magento, ShopWare, entre outras e através da documentação e API é possível proceder à sua implementação em outras ferramentas.

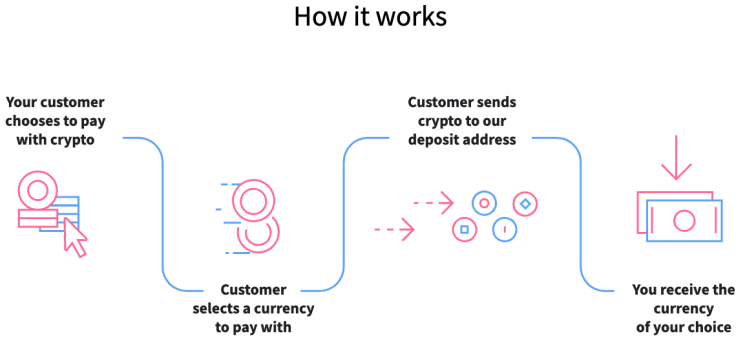


Figura 3.3: Representação do modo de funcionamento da solução NowPayments [80]

Aceitam uma gama de mais de 50 moedas, incluindo *stablecoins*, as mais populares e outros tokens [80]. Em relação às taxas cobradas pela empresa, se o comerciante aceitar pagamentos na mesma criptomoeda que o cliente quer pagar (por exemplo ETH para ambos), é cobrada uma taxa de 0.4% por transação. No entanto, se a moeda que aceitamos como comerciantes for diferente daquela que o cliente pagar, é cobrado um extra de 0.5% de forma a proceder a esse câmbio de criptomoeda. Esse câmbio é efetuado de imediato e o custo da transação pode ter um valor máximo de 1% [81]. Através da Figura 3.3, é possível observar o fluxo transacional desde o momento de pagamento até ao momento de receção do valor por parte do comerciante.

3.2.8 Comparação entre aos diversos métodos de pagamento

Nesta secção através da Tabela 3.2 é realizada uma comparação entre os diversos meios de pagamento referenciados tendo em conta os custos de cada transação para o comerciante, o número de moedas aceites, as taxas mensais de renovação. São também analisadas as capacidades da ferramenta permitir a troca de criptomoedas e se através das mesmas podemos efetuar pagamentos.

Tabela 3.2: Comparação entre as diferentes soluções de pagamento

Método de pagamento	Custos de transação	Moedas FIAT aceites	Transação com cripotmoedas	Compra com criptomoedas
Stripe	2.9% + 0.25€	135+	✅	❌
Square	2.9% + 0.30€	24	❌	❌
PayPal	3.49% + 0.49€	25	✅	❌
AmazonPay	3.9% + 0.30€	12	❌	❌
Checkout	2.3-2.9% + 0.30€	154	✅	✅
Revolut	2.8% + £0.20	30+	✅	❌
BitPay	1-2% + 0.25€	-	✅	✅
Coinbase Commerce	1%	-	✅	✅
NowPayments	0.5-1%	-	✅	✅

Através da Tabela 3.2, é possível verificar que soluções como a Stripe e Square facilitam o processo de *checkout* uma vez que são ferramentas que permitem a aceitação

de uma ampla gama de moedas FIAT e podem ser integradas de forma simples. No entanto a adoção de criptomoedas por parte das empresas é algo que apenas chegou aos seus utilizadores individuais que podem comprar e vender criptomoedas no entanto não conseguem efetuar pagamentos para comerciantes.

Por outro lado existem as soluções como o BitPay, Coinbase Commerce e NowPayments que correspondem a *Cryptocurrency Gateways* o que significa que apenas processam transações na *Blockchain* e, como tal são incapazes de processar moedas FIAT.

3.3 *Self-hosted Crypto Payment Gateways*

De forma a proceder a uma transação com uma determinada criptomoeda é necessário dispor de uma *crypto wallet* de forma a guardar os nossos fundos e é necessário algum tipo de *software* que nos permita proceder à elaboração da transação usando a *Blockchain* adequada ao tipo de moeda que dispomos.

Estas plataformas essencialmente fornecem aos seus utilizadores o *software* necessário e redirecionam o utilizador para a página de pagamento. A este tipo de solução são chamadas de 3^{rd} *party platforms* (CoinGate, BitPay, CoinbaseCommerce). Todas estas plataformas concentram-se em fornecer uma melhor experiência ao utilizador, utilizando interfaces especialmente projetadas que tornam as transações de criptomoedas realmente fáceis de realizar, mesmo para quem não tem experiência de utilização.

O conceito de plataformas *self-hosted* surge como alternativa às *Cryptocurrency Gateways* que correspondem a soluções gratuitas e *open-source*. São manualmente implementadas e hospedadas nas nossas plataformas que dispomos, sem recorrer a serviços de terceiros que nos cobram uma mensalidade/taxa por utilizarmos os seus serviços.

No entanto, através da Tabela 3.3 é possível identificar as vantagens e as desvantagens que este tipo de plataforma traz para o comerciante, evidenciando as grandes diferenças que existem perante as soluções comerciais existentes no mercado.

Tabela 3.3: Vantagens e desvantagens da utilização de plataformas *self-hosted*

Vantagens de plataformas *self-hosted*	**Desvantagens de plataformas *self-hosted***
Implicação direta na forma como a informação é guardada, que tipo de informação é recolhida e como é que esta é usada	É necessário algum conhecimento técnico/informático de forma a configurar este tipo de sistema
A informação é guardada pelo *host*. Permite tomar medidas adicionais (que tenhamos conhecimento) de forma a garantir que não exista fugas de informação	Muitas destas ferramentas vêm com uma interface simples ao contrário das outras que vêm com uma UI complexa e centrada no utilizador;
Aumento de segurança dado que podemos tomar as nossas próprias medidas	Aceitam um determinado número fixo de criptomoedas e a variedade pode não agradar a todos os clientes
Menos *delay* dado que não existe nenhuma *3rd party* a ser utilizada	Documentação e suporte podem constituir um problema aquando da necessidade de resolução de adversidades que surgem

De seguida, iremos apresentar alguns exemplos destas soluções, bem como algumas das suas funcionalidades oferecidas aos comerciantes.

3.3.1 BTCPay

A solução BTCPay corresponde a um processador de pagamentos *self-hosted*, *open-source*, privado e grátis para a *Blockchain* Bitcoin. Pode ser usada das seguintes maneiras: hospedado na *cloud*, hospedado no nosso próprio *hardware* ou então em alguma plataforma de *hosting*.

É possível conectar a algumas plataforma de e-commerce mais conhecidas (WooCommerce, Shopify, Drupal, Magento, PrestaShop)[82] e os fundos serão automaticamente depositados na nossa carteira. Esta solução não possui nenhumas taxas de transação, não suporta conversão para moedas FIAT, no entanto, possui uma API para integrarmos no nosso website.[83].

Através da seguinte Figura 3.4, conseguimos visualizar que a solução é composta por dois grandes elementos:

1. Dashboard onde o comerciante consegue visualizar algumas métricas sobre a sua loja, como por exemplo o número de pagamentos recebidos, efetuados, o montante disponível entre várias outras métricas. É possível também fazer a gestão das suas *wallets*, bem como gerir e/ou habilitar a rede *lightning* para a *Blockchain* da Bitcoin;

2. Modal onde o cliente (consumidor final) poderá efetuar o pagamento através de um QR-Code destinado a uma transação específica;

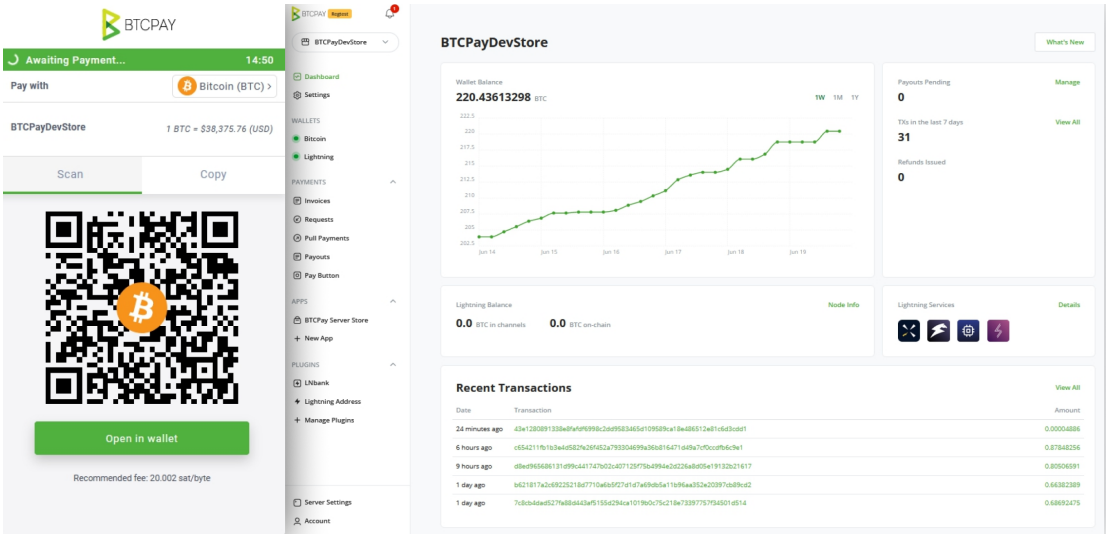


Figura 3.4: Representação visual da plataforma BTCPay [83]

Para além destas duas grandes funcionalidades, esta ferramenta dispõe de um mecanismo de POS, onde é possível criar a nossa própria loja com os nossos produtos, usufruindo do modo de pagamento oferecido pelo BTCPay [84]. É uma solução assente sob a *Blockchain* do Bitcoin, suportando as seguintes *altcoins*: Bitcoin Gold, Bitcoin Plus, Dash, DogeCoin, FeatherCoin, Litecoin, Moenro, Polis, Visacoin.

No entanto, como a solução é focada em pagamentos na rede Bitcoin, esta não suporta *tokens* ERC-20 uma vez que estes apenas são suportados por *Blockchain* que suportem *Ethereum Virtual Machine*.

3.3.2 BitCartCC

Tal como o BTCPay, o BitCartCC corresponde a uma solução completa de criptomoedas de forma *open-source* e *self-hosted*. Através desta ferramenta, é possível receber pagamentos de criptomoedas e *tokens* sem taxas ou envolvimento de terceiros. O comerciante é seu próprio banco [85].

Os fundos vão diretamente para a carteira do comerciante e, ao contrário da solução acima identificada, esta ferramenta para além de suportar a *Blockchain* do Bitcoin, suporta também a rede Ethereum, Bincance Smart Chain (BNB), Tron bem como os seus *tokens* respetivos ERC20, BEP20 e TRC20.

De forma similar, esta solução pode ser hospedada na *cloud*, em plataformas de terceiros ou até mesmo localmente através de um *container* Docker.

Como é possível observar pela seguinte Figura 3.5, esta solução é bastante semelhante à anterior, na medida em que dispõe de uma interface onde é possível visualizar algumas métricas, nomeadamente o número de carteiras disponíveis (em que cada carteira representa um meio de pagamento distinto da criptomoeda aceite), o número de faturas criadas para uma determinada loja, o montante disponível na nossa conta entre outras funcionalidades.

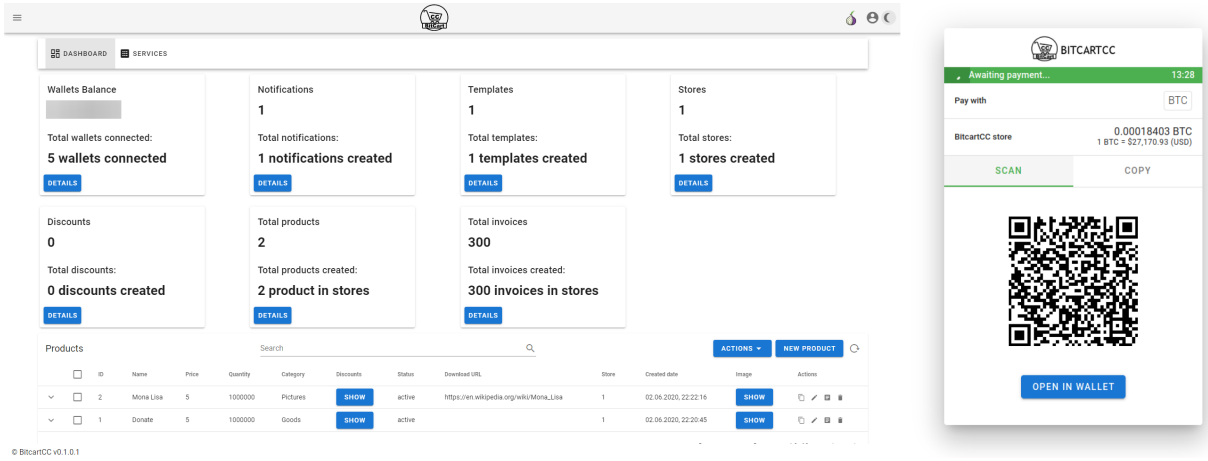


Figura 3.5: Representação visual da plataforma BitcartCC [85]

Para além dessas funcionalidades, esta ferramenta dispõe também de uma loja POS onde é possível criar, personalizar os nossos produtos e vendê-los fazendo uso dos sistemas referenciados anteriormente. Dispõe também de um mecanismo de descontos, em que é possível para uma determinada loja aplicar um desconto durante um determinado período de tempo.

Esta solução como é uma ferramenta *self-hosted* não dependente de terceiros, não dispõe da funcionalidade de conversão para moeda FIAT, algo que as soluções comerciais apresentam como benefício.

3.3.3 Hub20

A solução Hub20 [86], corresponde também a uma plataforma *self-hosted open-source*, capaz de processar pagamentos na *Blockchain* Ethereum onde é possível proceder à sua integração com projetos *layer-2* de forma a permitir pagamentos instantâneos na rede.

É possível aceitar pagamentos de qualquer ERC-20 token. Aceita também *stablecoins* de forma a contornar a volatilidade das moedas. É uma alternativa que permite realizar pagamentos sem a existência de qualquer tipo de taxas (apenas as taxas intrínsecas à *Blockchain*). Pode ser integrado com o nosso próprio website ou uma plataforma de *e-commerce*. Uma desvantagem desta alternativa é o facto do seu projeto não ser bem documentado e, não ter uma constante atualização pelos seus membros, o que o torna uma alternativa pouco viável tendo em conta as demais apresentadas.

3.3.4 Análise comparativa entre as soluções encontradas

Na Tabela 3.4 é feita uma comparação entre as diferentes crypto *gateways*. É possível observar de acordo com o seu tipo de (*self-custody, custodial*), os *tokens* suportados e os custos praticados por cada uma das opções listadas.

Para a realização desta tabela foi contactado o suporte de cada uma das plataformas via email com o propósito de verificar se era possível proceder à submissão de *tokens* personalizados. Contudo através das respostas obtidas, o *token* a introduzir tem cumprir com certos critérios estipulados pela empresa nomeadamente o número de *tokens* em circulação e o seu volume de transação. No entanto na eventualidade destas condições serem cumpridas, a plataforma não garantia que a implementação do *token* fosse realizada.

Tabela 3.4: Comparação entre as diferentes soluções de pagamento

Processador	**Tipo**	**Suporte *Lightning***	**Conversão para FIAT**	**Tokens suportados**	**Custos**
Bitpay	*Self-Custody*	✅	✅	Não suporta custom Tokens ERC20	1% / transação

Coinbase Commerce	*Self-Custody*	❌	✅	Não suporta tokens personalizados. Apenas os estipulados no website	1% / transação
Now Payments	*Custodial*	✅	✅	+50 tokens incluindo ERC20	0.5% / transação
BTCPay	*Self-Hosted*	✅	❌	Não suporta custom tokens ERC20, apenas a *Blockchain* BTC	❌
BitcartCC	*Self-Hosted*	✅	❌	Tokens na rede BTC, BNB, ETH, incluindo ERC20	❌
Hub20	*Self-Hosted*	✅	❌	Apenas tokens presentes na rede ETH	❌

Apesar das soluções apresentadas corresponderem a ferramentas muito idênticas, em termos do seu funcionamento e das funcionalidades que oferecem aos comerciantes, algumas diferem no modo de operação. Ou seja, enquanto que a solução Coinbase Commerce é do tipo *self-custody*, a solução NowPayments é do tipo *custodial* isto significando que enquanto na primeira solução referida, o utilizador tem o controlo total dos seus fundos (associado à sua chave privada da sua *wallet*), na solução proferida pela NowPayments, é a empresa que tem o controlo total dos fundos do utilizador, ou seja têm acesso à sua chave privada.

3.4 NFT como meio de pagamento

Atualmente, este tipo de ativos digitais é utilizado como forma de *trading*, apreciação do trabalho do autor ou então como colecionáveis, algo único em que o seu detentor tem os diretos daquele ativo. No entanto a sua utilização tem muitas outras aplicações e, através deste estágio pretende-se utilizar NFT como cupão/*voucher* numa determindada compra dentro de uma plataforma de *e-commerce*.

Os *vouchers* são fáceis de manipular e os proprietários são facilmente enganados. Os *scams* de *gift-cards* são um dos principais golpes da Internet [87]. NFTs poderiam

substituir o algoritmo tradicional de atribuição de direitos de propriedade aos *vouchers*. Utilizar *vouchers* NFT é tão fácil quanto resgatar um NFT por um prémio. Isso também poupa a instituição de produzir novos *vouchers* regularmente uma vez que os cupões NFT podem ser reutilizados repetidamente, tanto quanto possível.

De seguida será apresentada uma solução existente no mercado (ainda em período beta) onde um cliente pode comprar um cupão sob a forma de NFT e utilizá-lo na loja em questão.

GetKupon.io

Através desta plataforma [88], o utilizador pode comprar o cupão (sob a forma de NFT) e, após algum tempo, obter o seu benefício.

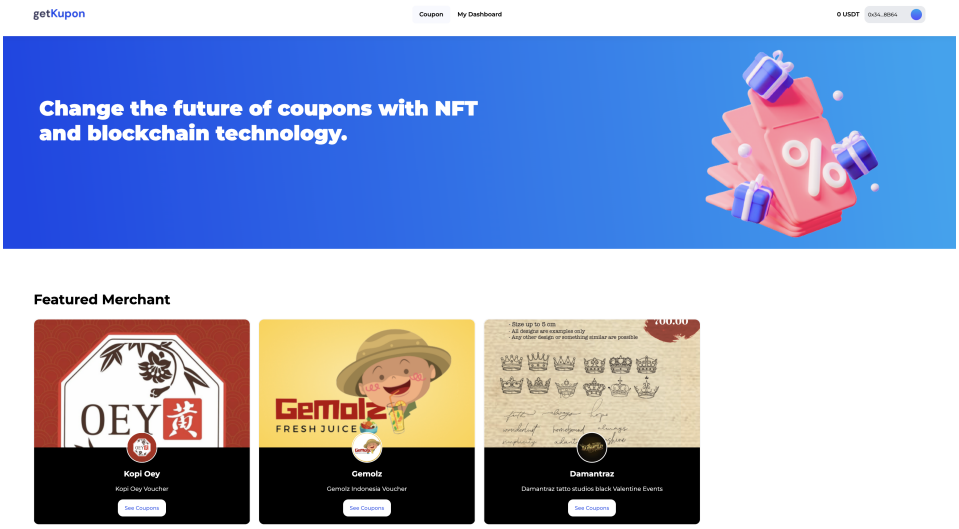


Figura 3.6: Interface da plataforma getKupon.io [88]

Através desta ferramenta, de acordo com a Figura 3.6 os comerciantes conseguem colocar na plataforma NFT específicos da sua loja de forma a que os seus clientes os possam comprar e usufruir aquando do processo de compra. Estes *vouchers* podem ser reutilizados tantas vezes quanto o comerciante desejar, uma vez que estão disponibilizados na *Blockchain*.

De forma a comprar estes *vouchers*, o cliente tem de iniciar sessão nesta plataforma com a sua *digital wallet* e, comprar o cupão com a criptomoeda USDT. De seguida, este cupão fica disponível quer na sua *wallet*, quer na aba "My Dashboard" onde o pode utilizar na loja em questão desde que o seu período de validade ainda esteja em vigor.

Capítulo 4

Metodologia e Planeamento

Esta secção visa analisar a abordagem à metodologia seguida para a realização deste projeto, bem como o planeamento estimado e a análise de riscos.

Este capítulo divide-se em cinco secções. Na primeira será identificada a metodologia a seguir ao longo do projeto incluindo as suas características e detalhes. De seguida, será apresentado o planeamento estipulado para ambos os semestres de trabalho de acordo com um diagrama de *Gantt*. A terceira secção estará destinada para a análise de riscos, onde serão identificados a sua descrição bem como o processo de constituição de um risco. Através da quarta secção será possível identificar os critérios de sucesso do estágio e, na última secção está presente a descrição sobre o nível de desenvolvimento e maturidade da aplicação.

4.1 Metodologia

A metodologia de trabalho é o conjunto de técnicas, ferramentas e processos que uma pessoa ou uma equipa utilizam para realizar uma tarefa ou projeto de forma eficiente e eficaz, maximizando o uso dos recursos disponíveis e garantindo que todos os objetivos propostos sejam atingidos.

Para este trabalho foi adotada uma variação da metodologia *Scrum*, alinhada com a abordagem utilizada internamente na empresa.

Na próxima subsecção, será apresentado o modo de funcionamento desta metodologia ágil e a sua consequente adaptação ao modelo de funcionamento do presente estágio tendo em conta alguns fatores chave como por exemplo o período de trabalho durante o primeiro semestre.

4.1.1 *Scrum*

O *Scrum* é uma *framework* de gestão de projetos baseada em ciclos de desenvolvimento iterativos e incrementais [89]. Atualmente, o modelo agile continua como preferência dos seus utilizadores, de acordo com o State of Agile Report (2022) como se pode verificar também pela seguinte Figura 4.1, retirada do mesmo artigo [90].

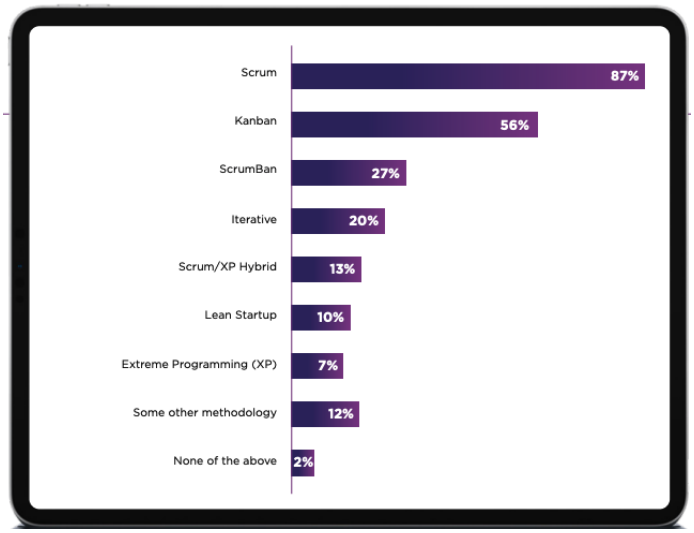


Figura 4.1: Metodologias ágeis mais utilizadas em 2022 [90]

O *Scrum* baseia-se em três pilares: transparência, inspeção e adaptação significando que todas as informações relevantes sobre o projeto devem ser de fácil acesso para todos os membros da equipa, que devem rever constantemente o progresso e ajustar o planeamento de acordo com o que foi aprendido.

Esta metodologia também tem um conjunto de papéis específicos, cada um com suas próprias responsabilidades:

- ***Scrum Master***: é o responsável por garantir que o *Scrum* seja implementado corretamente e ajuda a equipa a seguir as práticas recomendadas. Remove também obstáculos que possam impedir o progresso da equipa procurando melhorar as interações e o processo comunicativo da mesma;
- ***Product Owner***: é o responsável por definir o que deve ser desenvolvido e priorizar as funcionalidades a serem implementadas. Corresponde ao principal ponto de contacto com o cliente ou utilizador final. Visa cumprir e manter o produto focado nas características do cliente;
- **Equipa de desenvolvimento**: é composta pelos membros que realizam o trabalho de desenvolvimento e entregam as funcionalidades solicitadas estipuladas no *backlog* do projeto.

O *Scrum* também é baseado em eventos ou ciclos, que ocorrem em períodos regulares e têm um propósito específico entre os quais se distinguem [91]:

- ***Sprint Planning***: ocorre no início de cada *sprint* e é responsável por definir o que será feito durante o ciclo de trabalho;
- ***Sprint***: equivalem a ciclos curtos (cerca de 2 semanas), durante as quais o trabalho é realizado tendo em conta as tarefas planeadas;
- ***Daily Scrum***: corresponde a uma reunião curta diária (cerca de 10-15 minutos) de forma a integrar e informar a equipa sobre o progresso realizado e identificar obstáculos;
- ***Sprint Review***: ocorre no final de cada *sprint* e é uma oportunidade para a equipa demonstrar o que foi desenvolvido e receber *feedback*;
- ***Sprint Retrospective***: estipula um mecanismo de reflexão, onde a equipa pode observar os aspetos positivos e debater o que pode ser melhorado no próximo *sprint*.

Desvios do Scrum

A escolha da metodologia recaiu no Scrum uma vez que também é a metodologia utilizada internamente. No entanto, esta não foi cumprida à risca por diversos fatores nomeadamente o período de trabalho e tamanho da equipa pelo que foi utilizada uma variação desta metodologia ágil.

Durante o primeiro semestre do presente estágio, o estagiário estava em regime parcial (correspondendo a 20 horas semanais) o que fez com que cada *sprint* tivesse uma duração de 4 semanas. Durante esse período, existiam *daily scrums* realizadas 2 vezes por semana com o orientador do estágio, orientador técnico e *business analyst*.

No final do primeiro *sprint*, (final de outubro) o autor realizou uma apresentação na empresa onde foram debatidos os avanços relativos ao projeto e o planeamento para os próximos *sprints*. Esta apresentação teve um papel vital no processo de aprendizagem e de interação perante outros elementos da empresa bem como no processo de *feedback* recebido de modo a melhorar as futuras apresentações.

Ao longo do segundo semestre, a carga horária consistia em 40 horas semanais, divididos em *sprints* de 15 dias, totalizando 10 sprints. Nesse período, foram realizadas reuniões três vezes por semana com o orientador técnico com o objetivo de acompanhar o progresso realizado e esclarecer quaisquer questões que pudessem impedir o processo de desenvolvimento.

Ainda durante o segundo semestre foram realizadas um conjunto de apresentações internas de modo expor o trabalho realizado aos demais elementos da empresa e recolher *feedback* do desenvolvimento da aplicação.

A grande variação ao Scrum é a composição da equipa, ou seja, como para este projeto a equipa de desenvolvimento é apenas constituída pelo estagiário, questões de discussão, planeamento e priorização foram apenas tido em conta perante o estagiário que corresponde à equipa de desenvolvimento.

4.2 Planeamento

Nesta secção será apresentado o planeamento estipulado para o primeiro e segundo semestres através de um diagrama de *Gantt* que contém a delineação de tarefas necessárias para cumprir os objetivos estipulados.

4.2.1 Primeiro semestre

Ao logo do primeiro semestre, era expectável que o autor cumprisse com as seguintes metas:

- **Contextualização**: o primeiro objetivo consistia em compreender o motivo do estágio, que objetivos eram necessários para que este fosse concluído e principalmente o seu propósito, ou seja, qual seria a sua utilidade;
- **Estado da arte**: de seguida era necessário fazer uma pesquisa sobre alguns conceitos nomeadamente *Blockchain*, criptomoedas e o seu modo de funcionamento. Essa pesquisa auxiliou o modo de compreensão do tema e serviu como ponte para os estudos seguintes nomeadamente de plataformas de *e-commerce* e das suas soluções envolvendo criptomoedas;
- **Metodologia e planeamento**: após o estudo era necessário estipular o próximo passo a tomar e as tarefas que eram necessário realizar para o desenrolar do estágio e da solução;
- **Requisitos**: esta tarefa consistiu na definição dos requisitos funcionais, não funcionais e casos de uso identificados no capítulo 5;
- **Análise da solução e arquitetura**: após a identificação dos requisitos, o aluno tinha debateu com a equipa o aspeto da solução final e que arquitetura deveria seguir de forma a prever potenciais problemas que pudessem surgir no processo de desenvolvimento;

- **Documentação intermédia de estágio**: finalmente, o aluno ao longo do semestre tem de desenvolver o relatório intermédio em paralelo com as tarefas anteriores com o objetivo de documentar o progresso e descobertas realizadas;

4.2.2 Segundo semestre

O segundo semestre é dedicado à fase de desenvolvimento, com foco na implementação dos requisitos identificados durante o primeiro semestre. Assim como mencionado anteriormente, esse período foi dividido em *sprints* de duas semanas, nos quais foi elaborada a documentação no final de cada *sprint* de modo a registar decisões, processos ou alterações que pudessem impactar o âmbito do estágio.

Durante o 2º semestre era esperado a realização das seguintes tarefas descritas no diagrama de Gantt da Figura 4.2:

- **Revisão dos requisitos e casos de uso**: esta tarefa consiste na revisão detalhada dos requisitos levantados e análise dos casos de uso para garantir a compreensão adequada dos objetivos e funcionalidades esperadas;
- **Elaboração da Prototipagem da UI/UX**: para esta tarefa é necessário conceptualizar de forma visual o aspeto da solução de modo a validar as funcionalidades do sistema;
- **Setup do projeto**: corresponde à configuração do ambiente de desenvolvimento, incluindo a instalação de ferramentas e *frameworks* necessárias;
- **Desenvolvimento**: esta tarefa corresponde à fase central do segundo semestre, uma vez que inclui o processo de implementação da solução e dos requisitos necessários;
- **Testes e validações**: corresponde à fase de *testing* onde será avaliado o protótipo através de testes funcionais manuais e automatizados;
- **Documentação final de estágio**: como tarefa final temos a documentação final do estágio que será realizada ao longo do segundo semestre onde para além dos tópicos acima descritos deve incluir todas as decisões tomadas, alterações efetuadas caso existem e impactem o rumo do projeto incluindo também o trabalho futuro;

4.2.3 Visão do planeamento

Na Figura 4.2, também presente em formato PDF no Apêndice A, é ilustrado o planeamento seguido durante o primeiro e o segundo semestres incluindo a calenda-

rização das tarefas para a fase de desenvolvimento.

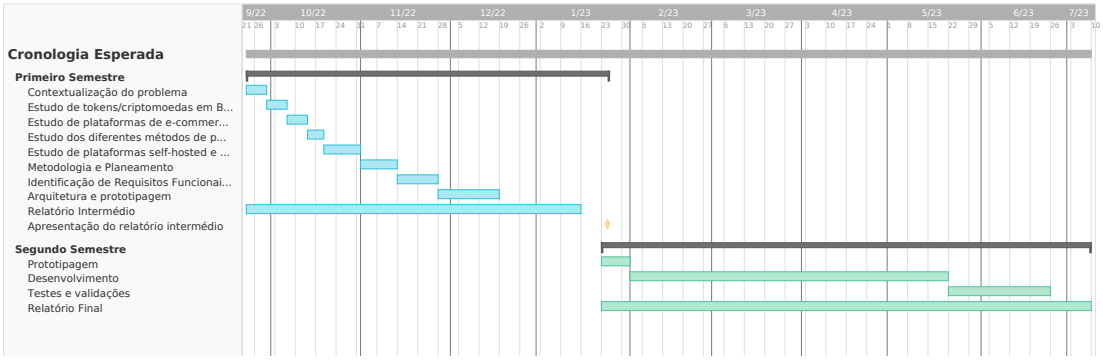


Figura 4.2: Diagrama de *Gantt* com a cronologia esperada para ambos os semestres

Com o objetivo de fornecer uma visão mais detalhada das divisões em *sprints*, a Figura 4.3 exibe o planeamento específico do segundo semestre. Além disso, no Apêndice A, é possível encontrar o planeamento em maior detalhe, incluindo as tarefas atribuídas a cada *sprint* e sua duração correspondente.

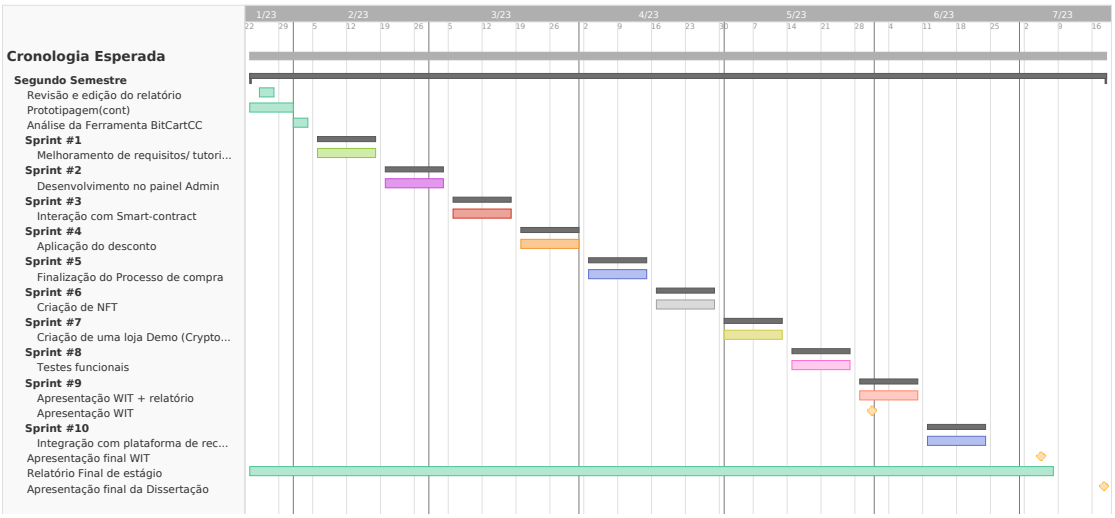


Figura 4.3: Diagrama de *Gantt* com a cronologia real do 2 semestre

Durante o primeiro semestre, o estágio envolve pesquisa e contextualização de ferramentas que seriam importantes no decorrer do mesmo pelo que não houveram grandes desvios do plano inicial estipulado no início do estágio.

Já durante a segunda metade do estágio, em termos temporais não ocorreram alterações significativas que comprometessem a duração do mesmo, no entanto, foi necessário ajustar alguns casos de uso e adicionar alguns requisitos fruto das decisões que foram tomadas com os orientadores. Essas decisões estão identificadas no Capítulo 8.

4.3 Gestão de riscos

Esta secção visa identificar o processo de gestão de riscos ao longo do projeto bem como proceder à sua identificação. Como tal está dividida em duas subsecções em que na primeira é relatado o processo de obtenção dos riscos através de uma matriz de riscos (Figura 4.4) e, na segunda parte será efetuada uma enumeração dos mesmos tendo por base as características estipuladas bem como o seu plano de mitigação.

4.3.1 Análise de riscos do projeto

Uma análise de riscos é importante num projeto de *software* uma vez que ajuda a identificar possíveis problemas ou ameaças que podem afetar o sucesso do mesmo. Esta análise pode ajudar a equipa a tomar medidas preventivas para evitar ou minimizar o impacto de tais riscos.

Como tal, é necessário estabelecer alguns processos de identificação de riscos que visem proceder à sua identificação e análise segundo certos parâmetros: [92]

- **Identificação do risco**: corresponde ao primeiro passo de forma a identificar todos os eventos que podem afetar negativamente (ou de forma positiva) os objetivos do projeto, nomeadamente colocar em risco a sua entrega ou o seu tempo, alteração de tema/âmbito do projeto ou em casos mais reais o custo/trajetória financeira do trabalho em curso [93];

- **Análise do risco**: existem dois tipos de avaliações de riscos/oportunidade: qualitativa e quantitativa. Uma avaliação qualitativa analisa o nível crítico do risco com base na probabilidade e no impacto do evento. Uma avaliação quantitativa analisa o impacto financeiro ou benefício do evento. Ambos são necessários para uma avaliação abrangente de riscos e oportunidades [93].

  A análise qualitativa tem por base 2 fatores representada pela Figura 4.4 que correspondem à probabilidade de ocorrência * impacto do risco e, estes podem ser divididos na seguinte escala:
  De acordo com a probabilidade

  - **Alta**: se o evento em questão tiver uma probabilidade de acontecer maior do que 70%
  - **Média**: se a probabilidade do acontecimento se compreender entre os 40 e 70%
  - **Baixa**: se a sua probabilidade for inferior a 40%

De acordo com o seu impacto

- **Alto**: quando existe um impacto bastante substancial que coloca em causa o orçamento/cronograma do projeto. O orçamento não será totalmente executado e o procedimento de negociação deve ser iniciado;
- **Médio**: existe um impacto médio no projeto em que pode ser necessário priorizar algumas tarefas mas, os critérios de sucesso ainda podem ser atingidos;
- **Baixo**: quando não existe um impacto no projeto, não havendo desvios significativos no âmbito nem nos entregáveis propostos em que os critérios de sucesso podem ser facilmente atingidos;

- **Planeamento**: este é o próximo passo, onde se irá desenvolver um plano e estratégias com o objetivo de reduzir a probabilidade de ocorrência do risco e/ou o seu impacto. Como tal, é necessário estabelecer um plano de mitigação para o risco em questão [93];

- **Controlo e gestão de riscos**: riscos e os seus planos de tratamento precisam ser monitorados e relatados com o decorrer do projeto. A frequência dessa operação dependerá do quão crítico o risco for. Ao desenvolver uma estrutura de monitorização de riscos, garantirá a existência de canais apropriados para a resposta aos riscos [93].

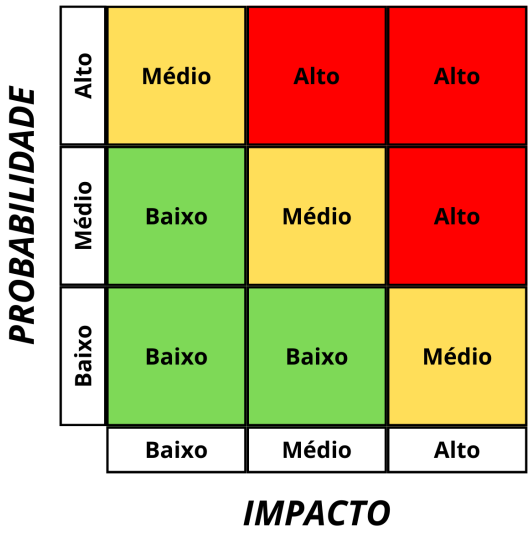


Figura 4.4: Matriz de riscos [94]

4.3.2 Identificação de riscos

Nesta secção serão enumerados os riscos encontrados até à data, procedendo à sua classificação tendo em conta a Figura 4.4.

Risco 1 - Mudança nos requisitos

Probabilidade: Média;
Impacto: Alto;
Consequências: É necessário alterar documentação e será essencial gastar mais tempo na pesquisa o que pode condicionar a duração do projeto. Uma vez que o prazo de entrega não pode ser alterado, será necessário mais tempo de modo a cumprir com os novos requisitos o que pode colocar em causa fatores como por exemplo a qualidade do produto;
Plano de mitigação: Realização de mais reuniões com a equipa (orientador, tutor técnico) de forma a esclarecer e identificar com mais detalhe todos os requisitos necessários de forma a não colocar em causa atrasos no projeto e os seus entregáveis.

Risco 2 - Falta de conhecimento em alguma tecnologia/tema

Probabilidade: Alta;
Impacto: Médio;
Consequências: Impossibilidade de desenvolver código nessa tecnologia. Ou é necessário um tempo extra de forma a se integrar com os conceitos sobre a tecnologia/tema em questão;
Plano de mitigação: Treinar, atempadamente, ao resolver tutoriais *online* sobre essas tecnologias. Além disso, comunicar com os orientadores, que poderão ter conhecimento sobre as mesmas e poderão auxiliar em qualquer ponto do desenvolvimento.

Risco 3 - Estimativas erradas

Probabilidade: Médio;
Impacto: Alto;
Consequências:A má identificação dos tempos para as tarefas pode colocar em causa as tarefas seguintes uma vez o seu cumprimento pode ficar em causa;
Plano de mitigação: Identificação dos requisitos segundo uma escala de MoSCoW de forma a estabelecer a identificação dos riscos mais importantes para o projeto. Comunicar com os orientadores sobre a nossa estimativa de forma a obter *feedback* e aconselhamento sobre o *timeline* estipulado;

Risco 4 - Mudanças na integração com serviços externos

Probabilidade: Média;

Impacto: Alto;
Consequências: Alterações inesperadas na forma como a integração com serviços externos, como o Shopify, é realizada podem comprometer o trabalho desenvolvido e o âmbito do estágio. Isso pode resultar em atrasos devido à alteração da aplicação, possíveis falhas na funcionalidade esperada ou no pior dos casos a completa eliminação da funcionalidade com o serviço externo;
Plano de mitigação: Manter um acompanhamento regular das documentações e atualizações dos serviços de modo a antecipar-se a possíveis mudanças. Em casos de mudanças significativas, buscar suporte e orientação dos supervisores e especialistas técnicos para avaliar o impacto e adaptar o trabalho em conformidade de modo a não comprometer o estágio e as funcionalidades previstas;

Risco 5 - Limitações das plataformas de integração

Probabilidade: Média;
Impacto: Alto;
Consequências: As limitações das plataformas a serem integradas podem apresentar obstáculos e desafios que podem inviabilizar a realização dos objetivos do projeto. Isso pode incluir restrições de funcionalidade, incompatibilidade de sistemas, dificuldades de integração ou falta de suporte adequado;
Plano de mitigação: Realizar uma avaliação detalhada das plataformas de integração considerando as limitações técnicas e requisitos do projeto. Considerar a possibilidade de realizar ajustes na arquitetura ou buscar alternativas de plataformas, ou então alterar objetivos de modo a prosseguir com a plataforma pretendida;

Apesar dos riscos terem sido previamente identificados, é essencial que o estagiário reconheça sua existência neste estágio, a fim de minimizar possíveis problemas que possam surgir. Durante o decorrer do estágio, foram enfrentados alguns desafios nomeadamente questões relativas à falta de conhecimento do tema *Blockchain* o que fez com que houvesse a necessidade de realizar tutoriais com as tecnologias a utilizar de modo a promover o processo de auto-aprendizagem e de integração com a mesma. Através da elaboração dos casos de uso e suas tarefas, foram recolhidas informações sobre a plataforma de *e-commerce* Shopify onde chegámos à conclusão que o caso de uso recolhido não era possível realizar devido a problemas intrínsecos à plataforma e, de modo a contornar o problema, foi pensada uma nova funcionalidade que, de forma similar, cumprisse com o requisito pretendido e no Capítulo 8 são referidos mais detalhes sobre os desafios de implementação. Já no final do estágio, a plataforma Shopify lançou uma nova atualização à forma como as lojas em ambiente de desenvolvimento integravam e testavam as suas soluções, pelo que houve a necessidade de alterar a forma como a aplicação era testada e após a conversação com o suporte e orientadores, a forma de resolver o problema passou por criar um plano pago de modo a conseguir testar a aplicação na nova atualização.

4.4 Critérios de Sucesso

De modo a determinar o sucesso do estágio é necessário definir alguns critérios de modo a estes sejam cumpridos até à conclusão do mesmo. Esses critérios incluem:

1. Segurança e confiabilidade: Mediremos a segurança do *plugin* através de uma combinação de testes manuais e automatizados;

2. No término do estágio, espera-se a conclusão de um protótipo funcional, que seja facilmente demonstrável e represente de forma sólida os conceitos explorados ao longo do período do estágio.

Para além destes critérios de modo a que se possa determinar o sucesso e o término do projeto, todos os requisitos funcionais e não-funcionais classificados com a prioridade *"Must Have"*, deverão ser devidamente completos e testados até à sua data de término.

Em resumo, estes critérios de sucesso vão permitir medir e avaliar o desempenho do nosso *plugin* de pagamento Web3 para plataformas de *e-commerce*, bem como identificar áreas de oportunidade para melhoria contínua.

4.5 *Technology Readiness Level*

Technology Readiness Level (TRL) corresponde a um método de modo a determinar a maturidade técnica de uma tecnologia durante a sua fase de aquisição. Originalmente desenvolvido pela National Aeronautics and Space Administration (NASA)[95] na década de 70 para tecnologias de exploração espacial, os TRLs medem o nível de maturidade de uma tecnologia ao longo da sua pesquisa, desenvolvimento e progressão da fase de implementação. Os TRLs são baseados em uma escala de 1 a 9, sendo 9 a tecnologia mais madura como se pode verificar pela Figura 4.5.

Aplicando esta escala ao nosso projeto, no final do seu desenvolvimento é esperado um protótipo funcional contento todas as funcionalidades e cumprindo com todos os requisitos funcionais e não-funcionais descritos. Uma vez que se trata de um protótipo funcional de acordo com a escala de TRL descrita na Figura 4.5, este incide no nível 4 descrito como um "protótipo de demonstração em ambiente controlado".

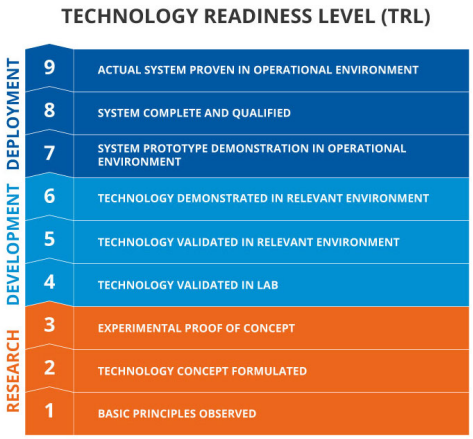


Figura 4.5: Escala de *Technology Readiness Level* [96]

Em suma, o objetivo deste estágio é criar uma prova de conceito de modo a avaliar a exequibilidade da aplicação e se é uma ideia que irá trazer benefícios para os seus utilizadores.

Capítulo 5

Requisitos

A análise de requisitos é um dos passos mais importantes e iniciais num projeto de *software* uma vez que ajuda a entender o que o sistema deve fazer estipulando as suas funcionalidades.

Esta secção encontra-se dividida em cinco subsecções em que na primeira é introduzido o contexto do problema de forma a proceder à sua compreensão e introdução aos requisitos. De seguida, são apresentados os requisitos funcionais juntamente com os diagramas de casos de uso nas subsecções dois e três respetivamente.

Na secção quatro, serão apresentados os requisitos não funcionais e, na subsecção cinco será assinalado o ponto de partida da solução.

5.1 Contexto

Tal como identificado no Capítulo 1, este estágio tem como objetivo a criação de um *plugin* de pagamentos em criptomoedas para plataformas de *e-commerce* onde o utilizador final possa usufruir deste modo de pagamento juntamente com NFT como *vouchers* para efetuar as suas compras.

O objetivo é a construção de uma ferramenta que disponha da funcionalidade acima enumerada e que não tenha dependência de processadores de pagamentos externos com suporte para *tokens* personalizados implementados em *Blockchains* compatíveis com Ethereum.

A análise dos requisitos por parte do estagiário teve por base a descrição do estágio para além de várias reuniões com a equipa de forma a transmitir a informação identificada e proceder ao seu aperfeiçoamento.

5.2 Requisitos funcionais

Os requisitos funcionais são as características ou funcionalidades que um sistema deve possuir de forma a atender às necessidades de um utilizador ou cliente. Estes descrevem o que o sistema deve fazer, mas não como ele deve fazê-lo.

De forma a proceder à sua classificação, foi utilizada a classificação de MoSCoW com o objetivo de priorizar as necessidades de um projeto. A escala é composta pelas seguintes categorias [97]:

- ***Must Have (M)***: requisitos considerados críticos para o sucesso do projeto e que devem ser implementados no ciclo de desenvolvimento;
- ***Should Have (S)***: requisitos importantes, mas que podem ser adiados para um ciclo de desenvolvimento posterior se necessário;
- ***Could have (C)***: requisitos desejáveis, mas que podem ser descartados se necessário devido a restrições de tempo ou orçamento;
- ***Won't have this time (W)***: requisitos que não são considerados prioritários e que podem ser adiados para uma versão futura do projeto;

A solução pretendida tem por base 3 atores:

- **Comerciante**: Que dispõe de uma interface onde consegue visualizar o histórico de transações, o seu saldo e gerir as suas *wallets*;
- **Consumidor**: Corresponde à pessoa que vai pagar pelos produtos em que lhe é apresentada a fatura e, este pode escolher com que criptmoeda deseja pagar (dentro das disponíveis) através de um QR-Code, ou pode também efetuar *login* com a ferramenta *Metamask* de forma a pagar a fatura;
- **Suporte**: O suporte corresponde ao ator que fica responsável pela gestão da aplicação ao comerciante, onde pode fazer *updates* à plataforma, dar suporte aos comerciantes, adicionar novos *tokens* à plataforma de modo a que estes fiquem disponíveis como meio de pagamento. Ou seja, é o ator que trata da parte de suporte e de *backend* de forma a ajudar o comerciante em qualquer situação.

No entanto, para a enumeração dos requisitos funcionais, foi adotada a abordagem de criação de *User Stories* como ponto de partida, as quais foram posteriormente refinadas em requisitos específicos e tarefas necessárias para sua implementação.

Conforme é comum em projetos de engenharia de software, houve a necessidade de atualizar as *User Stories* existentes já na fase de desenvolvimento, resultando em ajustes nos requisitos da aplicação. Essas adições são identificadas pelo símbolo (+) na Tabela 5.1 e as *user stories* estão definidas no Apêndice B.

Tabela 5.1: Requisitos Funcionais

ID	Requisito Funcional	Prioridade
Suporte		
RF-1	Aceitar tokens personalizados ERC20 e *vouchers* NFT (ERC 721)	**(M)**
RF-2	Integração com a plataforma de *e-commerce* Shopify	**(M)**
RF-3	Integração com a *Torus Wallet*	**(C)**
RF-4	Integração da aplicação num website personalizado	**(M)**
RF-5	Criação de um *smart contract* para utilização de NFTs	**(M)**
RF-6	Criação dos *vouchers* NFT no painel de administrador (+)	**(M)**
RF-7	Desenvolvimento de uma aplicação Shopify para a criação de *vouchers* (+)	**(M)**
RF-8	Integração com o estágio *Blockchain Loyalty Platform* (+)	**(C)**
RF-9	Integração com plataforma de recompensas interna (+)	**(C)**
Comerciante		
RF-9	Dashboard com o histórico de transações (crypto e NFT)	**(M)**
RF-10	Visualização do saldo da loja	**(M)**
RF-11	Gerir as suas *wallets*	**(M)**
RF-12	Visualização dos detalhes do voucher	**(M)**
RF-13	Transferir o voucher para outro endereço na Blockchain	**(M)**
RF-14	Visualizar as interações de um voucher num explorador de blocos	**(M)**
RF-15	Eliminar um determinado voucher	**(M)**
Consumidor		
RF-16	Utilização de um QR-Code como meio de pagamento	**(M)**
RF-17	Integração com wallets externas (**Metamask**, **WalletConnect**)	**(M)**

RF-18	Visualização dos vouchers disponíveis no ato de pagamento	**(M)**
RF-19	Escolher e aplicar um determinado voucher	**(M)**
RF-20	Alterar a escolha de um voucher no ato de pagamento	**(M)**
RF-21	Pagar através de criptomoedas (tokens ERC20)	**(M)**

5.3 Diagramas de casos de uso

Com o objetivo de clarificar e identificar de forma visual os requisitos estipulados anteriormente, foram criados diagramas para cada ator envolvido e para os requisitos funcionais estipulados na Tabela 5.1.

Através da visualização da Figura 5.1 conseguimos identificar as funcionalidades a que o consumidor terá acesso.

Este conseguirá escolher com que *wallet* irá proceder ao pagamento e, de seguida ser-lhe-á apresentado o devido QR-Code de forma a que de uma forma simples este possa pagar o devido montante.

Antes de efetuar o pagamento final, este poderá também submeter cupões de desconto (sob a forma de NFT) de forma a que consiga usufruir de um desconto através desse ativo digital.

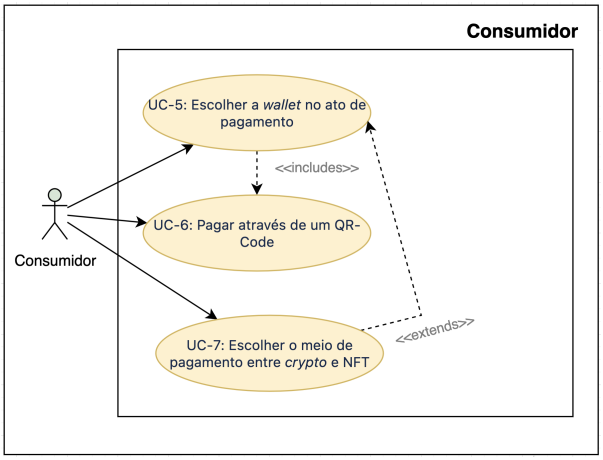


Figura 5.1: Casos de uso - Consumidor

Na Figura 5.2 estão representadas as funcionalidades a que o Comerciante e Suporte terão acesso.

O Comerciante terá acesso a uma *dashboard* onde poderá visualizar o seu saldo (correspondente às suas vendas e disponível nas suas *wallets*), inclusive poderá adicionar

novas *wallets* de forma a aceitar novas criptomoedas. Este também poderá criar faturas com um valor específico onde poderá confirmar se deseja que os clientes usem *vouchers*/NFT como meio de pagamento.

Por fim, como se trata de uma *dashboard*, este também será capaz de visualizar as transações realizadas, incluindo o seu montante, data e *hash* de forma a garantir que o montante foi pago. Este histórico de transações aplica-se para os dois meios de pagamento, tanto com criptomoedas como para NFT.

O Suporte, como ator surge tal como o nome indica, como a ferramenta de ajuda do comerciante que fornece e garante a gestão da infraestrutura. Como tal, este será capaz de adicionar novos *tokens* à plataforma de modo a que o Comerciante os consiga aceitar como meio de pagamento.

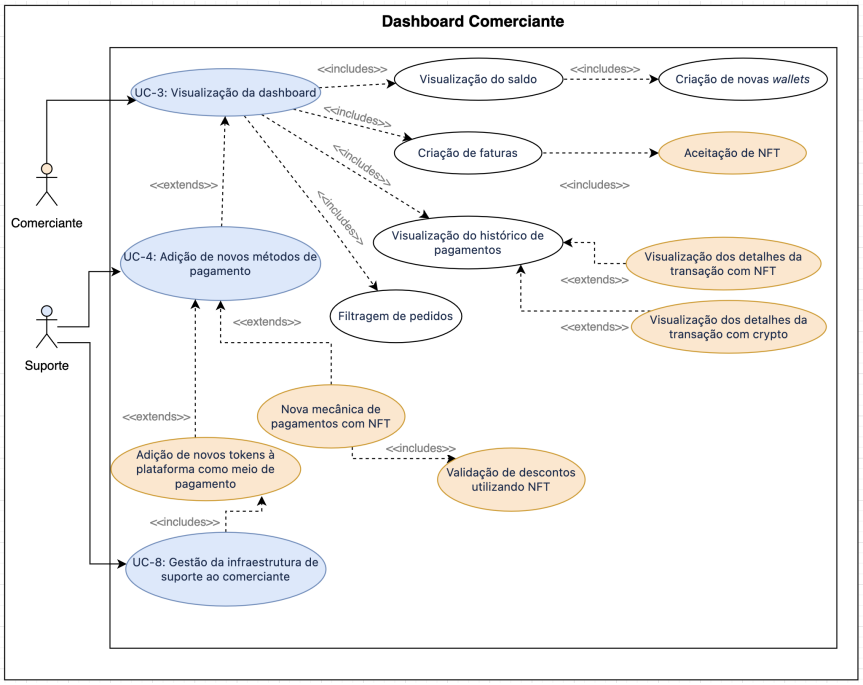


Figura 5.2: Caso de uso - Comerciante e Suporte

Através da Figura 5.3 o comerciante conseguirá utilizar esta solução na sua plataforma de *e-commerce*, de modo a que na sua loja consiga aceitar criptomoedas como meio de pagamento. Como tal, a plataforma Shopify (plataforma de *e-commerce* escolhida para este requisito), deverá comunicar com a nossa solução de modo a que o consumidor final consiga usufruir deste modo de pagamento através da *Merchants API* que irá invocar um *modal* onde o cliente consegue proceder ao pagamento da fatura.

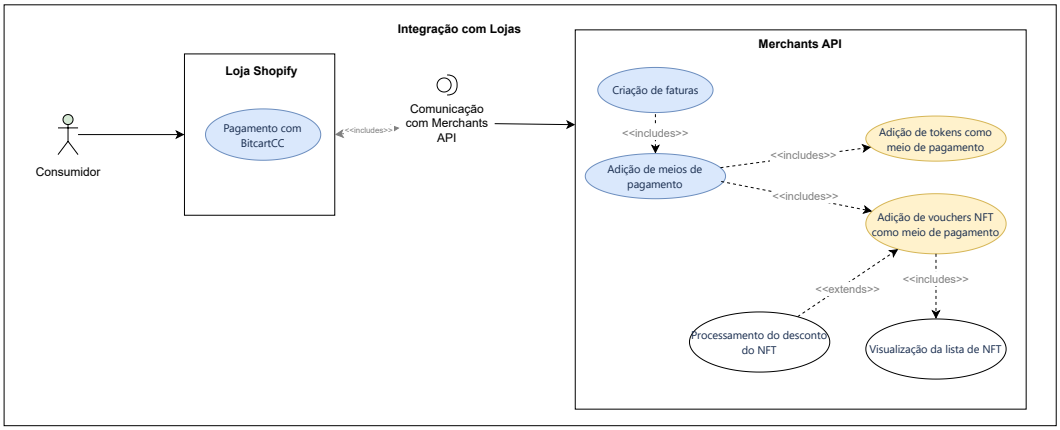


Figura 5.3: Caso de uso - Integração com plataforma de *e-commerce*

No entanto, caso a integração seja feito num website personalizado, ao invés de haver uma comunicação com os serviços do Shopify, existe a necessidade de adaptar a comunicação, passando sempre pela comunicação com a *Merchants API*.

5.4 Requisitos não funcionais

Os requisitos não funcionais correspondem às especificações de um sistema, produto ou serviço que descrevem as características que não são diretamente relacionadas às suas funcionalidades, mas que são importantes para o seu funcionamento e qualidade.

É importante que os requisitos não funcionais sejam bem definidos e compreendidos por todos os envolvidos no projeto, de forma a garantir que o produto final atende às expectativas propostas e seja entregue dentro do prazo e orçamento estabelecidos.

Além disso, é importante levar em consideração os requisitos não funcionais desde o início do projeto, uma vez que a sua implementação pode exigir muito tempo e esforço e, se não forem considerados, podem levar a problemas no futuro.

Na Tabela 5.2 estão definidos os requisitos não funcionais deste projeto.

Tabela 5.2: Requisitos Não Funcionais

ID	Requisito Não Funcional
RNF-1	Desempenho
RNF-2	Escalabilidade
RNF-3	Resiliência
RNF-4	Segurança

5.4.1 Desempenho

Através do presente atributo de qualidade, pretende-se que o sistema seja capaz de processar e responder a todos os pedidos num curto período de tempo.

Para tal existem 2 cenários: um envolvendo a comunicação com a *Blockchain* e outro para as operações normais da plataforma à Application Programming Interface (API) sem efetuar este tipo de comunicação.

Tabela 5.3: Cenário 1: RNF-1 Desempenho

Cenário 1 - Comunicação com a *Blockchain*	
Fonte	Utilizador
Estímulo	Procede a um pedido que envolve a comunicação e alteração de dados na *Blockchain* (transferência de um *voucher*)
Artefacto	*Blockchain Gateway*
Ambiente	Operação normal do sistema
Resposta	O sistema devolve a resposta sobre o estado da transação
Métrica da resposta	Este deve responder num tempo máximo inferior a 1 minuto (1)

(1): Este valor foi recolhido através da análise de diversos documentos que retratam o desempenho da *Blockchain* Polygon [98] [99].

Tabela 5.4: Cenário 2: RNF-1 Desempenho

Cenário 2 - Comunicação a aplicação	
Fonte	Comerciante
Estímulo	Procede a um pedido que envolve a comunicação com a API
Artefacto	Painel de Administrador
Ambiente	Operação normal do sistema
Resposta	O sistema devolve uma resposta
Métrica da resposta	Este deve responder num tempo máximo inferior a 1 segundo (2)

(2): Este valor foi retirado da análise de um livro de *Usability Engineering* que retrata os tempos de resposta para os utilizadores seguindo a regra de 0.1, 1 e 10 segundos. O livro explora a perceção do utilizador e o tempo máximo que ele está disposto a esperar antes de perder a sua atenção. [100].

5.4.2 Escalabilidade

É uma característica desejável em todo o sistema, rede ou processo, que indica a capacidade de manipular uma porção crescente de trabalho de forma uniforme, ou estar preparado para crescer.

Ou seja para este RNF, pretende-se que a plataforma em casos de grande fluxo de pedidos, se mantenha disponível, mantendo sempre a performance de forma a não colocar em causa a experiência do utilizador. Como tal, na Tabela 5.5 é definido um cenário para a sua aplicação.

Tabela 5.5: Cenário 1: RNF-1 Escalabilidade

Cenário 1 - Comunicação com a Blockchain	
Fonte	Aumento do número de utilizadores da plataforma de comércio eletrónico e do volume de transações realizadas através do plugin de pagamentos Web3
Estímulo	A capacidade do plugin de pagamentos web3 em lidar com o crescimento do uso da plataforma
Artefacto	Plugin Pagamentos Web3
Ambiente	Operação de sobrecarga de pedidos do sistema
Resposta	Desempenho rápido e estável do plugin de pagamentos web3, capacidade de lidar com o crescimento do uso da plataforma
Métrica da resposta	Tempo de resposta do sistema e a taxa de falhas de transações.

Através do conjunto destas métricas, é possível analisar a variação do tempo de resposta da plataforma face ao crescimento dos seus pedidos e a decisão de escalar os serviços horizontalmente através da adição de mais Docker *containers* à plataforma e de um *load balancer* de modo a distribuir a carga pelas diversas instâncias a correr.

5.4.3 Resiliência

Esta plataforma, como solução existente é composta por vários módulos que comunicam entre si quando necessário, de forma a que caso aconteça algum erro num dos módulos, este não afete o desempenho e o seu impacto seja mínimo nos restantes.

Como tal, foi definido o cenário da Tabela 5.6 de modo a demonstrar a sua aplicação.

Tabela 5.6: Cenário 1: RNF-3 Resiliência

Cenário 1 - Ocorrência de um erro no módulo da *Blockchain*	
Fonte	Sistema e os seus diversos módulos
Estímulo	Ocorrência de um erro num dos módulos da plataforma devido a fatores externos
Artefacto	Plugin Pagamentos Web3
Ambiente	Operação normal
Resposta	Os demais módulos continuam a sua operação normal fornecendo um desempenho aceitável e o módulo com o erro é reiniciado automaticamente
Métrica da resposta	Tempo de reiniciar o módulo do sistema

No entanto, caso algum dos serviços esteja inoperacional por um período prolongado e o reinício dos serviços não solucione o problema, será possível recorrer a um mecanismo de balanceamento de carga, no qual os pedidos serão redirecionados para outro servidor funcional.

5.4.4 Segurança

Uma vez que através desta ferramenta se irá proceder a transferências financeiras, o sistema deve ter medidas de segurança robustas para proteger informações financeiras confidenciais.

No entanto, a solução assenta apenas sobre *Non-Custodial wallets* o que significa que é apenas o titular que tem acesso à chave privada da sua carteira pelo que a solução apenas necessita da sua chave pública de modo a efetuar transações, ou seja, a plataforma não guarda qualquer dados confidenciais da sua carteira.

Para além disso como estamos a efetuar transações na *Blockchain*, esta dispõe de mecanismos de integridade de imutabilidade de transações garantindo que estas são seguras e confidenciais. A combinação destas tecnologias fornece um alto nível de segurança e privacidade em transações financeiras e outros tipos de interações digitais [14]. No entanto, é importante notar que as vulnerabilidades ainda existem, e que as ameaças de segurança estão sempre a evoluir, exigindo que sejam constantemente revistas.

Como tal na Tabela 5.7 foi elaborado um cenário para o presente requisito não funcional de modo a demonstrar a sua utilização.

Tabela 5.7: Cenário 1: RNF-4 Segurança

Cenário 1 - Acesso a informação privilegiada	
Fonte	Utilizador não autenticado
Estímulo	Tenta aceder a informação pessoal sem autorização
Artefacto	Plataforma
Ambiente	Operação normal do sistema
Resposta	O sistema bloqueia o acesso, negando o pedido ao utilizador, uma vez que este não tem acesso à funcionalidade pretendida
Métrica da resposta	O atacante não conseguirá efetuar a operação pretendida

O bloqueio do pedido é feito através da utilização de autenticação através de JWT, onde cada pedido verifica se para aceder aquele recurso este dispõe das permissões necessárias para obter o seu valor. Para além desse método de autenticação na API, é também utilizado o protocolo HTTPS de modo a evitar interceção de comunicações por entidades externas.

5.5 Escolha da solução

Após a identificação dos requisitos, o próximo passo consiste na identificação da solução e do seu ponto de partida. Para isso, foi elaborada uma tabela com os pontos fortes e pontos fracos das soluções *open-source* existentes, uma vez que este é um dos requisitos chave propostos na Tabela 5.1.

Como tal, o autor procedeu a uma análise com mais detalhe sobre as soluções *open-source* encontradas, identificadas na Subseção 3.2.7 e realizou a Tabela 5.8 tendo em conta a documentação das plataformas. Os pontos fortes e fracos identificados tiveram também por base o processo de instalação das ferramentas BTCPay e BitcartCC onde se avaliaram as suas funcionalidades.

Tabela 5.8: Tabela comparativas de soluções *self-hosted*

Pontos fortes ✅	**Pontos Fracos ❌**
Plataforma: BTCPay	
possibilidade de criar várias lojas dentro do mesmo servidor	o foco é na rede BTC e a gestão de outras *altcoins* não é fidedigna
criação de faturas personalizadas com o montante específico bem como mecanismos de reembolso	não existe suporta para token assentes na **Blockchain** ETH
existência de uma aplicação POS	não existe mecanismos de conversão para FIAT
dispõem de mecanismos de suporte via email	necessário algum conhecimento técnico para o processo de configuração
suporte para a rede *lightning* BTC	
Plataforma: BitCartCC	
possui uma *API* e um *SDK* de forma a utilizar algumas das suas ferramentas	a sua implementação pode ser desafiadora devido à complexidade do projeto
possibilidade de criar várias *wallets* ("*1 wallet per currency*")[85]	suporte reduzido realizado através de canais de Discord ou Telegram
para cada fatura gerada, existe um mecanismos onde é gerado um novo endereço pertencente à chave pública introduzida de forma a melhorar a privacidade	dificuldade em dar *setup* uma vez que é uma solução composta por vários módulos
possibilidade de aceitar tokens ERC20	
Plataforma: Hub20.io	
suporta a *Blockchain* ETH e ERC20 *tokens*	suporte apenas para *fungible tokens*
	os tokens disponíveis vêm através de uma lista de uma Decentralized Exchange (DEX) [101][102]
	corresponde a uma *custodial wallet*("*Yes, users of any instance do not have access to the keys and have to trust the hub operator*")[103]
	solução com escassez de informação sobre o seu modo de funcionamento e instalação.

Após a análise das vantagens e desvantagens da Tabela 5.8, o autor decidiu que o melhor ponto de partida seria a solução oferecida pelo BitCartCC, uma vez que correspondia a uma solução com alguns requisitos já estipulados, nomeadamente a existência de uma forma de pagamentos com criptomoedas assentes na rede ETH e, o utilizador através desta solução consegue já pagar utilizando a *wallet* externa *Metamask* ou *Wallet Connect*.

Este mecanismo dispõe também de um QR-Code no processo de pagamento onde podemos pagar fazendo uso do mesmo de modo a que seja uma experiência mais simples para o consumidor final.

Capítulo 6

Arquitetura de software

Neste capítulo será feita a análise da arquitetura do ponto de partida da solução escolhida e identificada na Seção 5.5, que corresponde à plataforma BitCartCC. E serão também analisadas as tecnologias a utilizar no decorrer deste projeto.

6.1 Arquitetura

Uma vez que se trata de uma plataforma *open-source*, a arquitetura da mesma já se encontra estipulada pelos seus desenvolvedores e, na Figura 6.1 é apresentada com maior detalhe a sua implementação. No entanto através do decorrer da fase de desenvolvimento foi necessário adicionar novos elementos à arquitetura de modo a realizar os requisitos estipulados.

A aplicação é composta por um conjunto de módulos cada um com o seu objetivo. Estes módulos de forma conjunta permitem à aplicação executar as inúmeras funcionalidades que esta possui. Através da Figura 6.1 é possível identificar a divisão da arquitetura de acordo com os atores intervenientes:

1. **Cliente**: ator responsável por aceder à plataforma de e-commerce e usufruir do método de pagamento e dos vouchers;

2. **Comerciante**: corresponde ao ator que dispõe de uma loja e tem acesso ao seu painel de administrador onde consegue fazer a gestão da sua loja e dos seus *vouchers* NFT;

3. **Suporte**: este ator fica responsável pela gestão da infraestrutura onde será capaz de adicionar novas funcionalidades e fornecer ajuda ao comerciante perante a gestão da sua loja.

Ainda na Figura 6.1 são representadas as interações que a aplicação tem com sistemas externos:

1. ***Metamask Wallet***: utilizada para efetuar transações na *Blockchain* de modo a pagar faturas e/ou criar *vouchers* NFT;
2. **Alchemy API**: API utilizada para obter detalhes e informações sobre os *vouchers*;
3. **Pinata Cloud**: ferramenta utilizada para guardar dados no IPFS;
4. **Shopify API**: API utilizada pelo comerciante de modo a obter os produtos da sua loja ou criar faturas na plataforma de *e-commerce*;
5. **Plataforma de *e-commerce***: local onde o cliente procede à compra de itens na loja do comerciante;

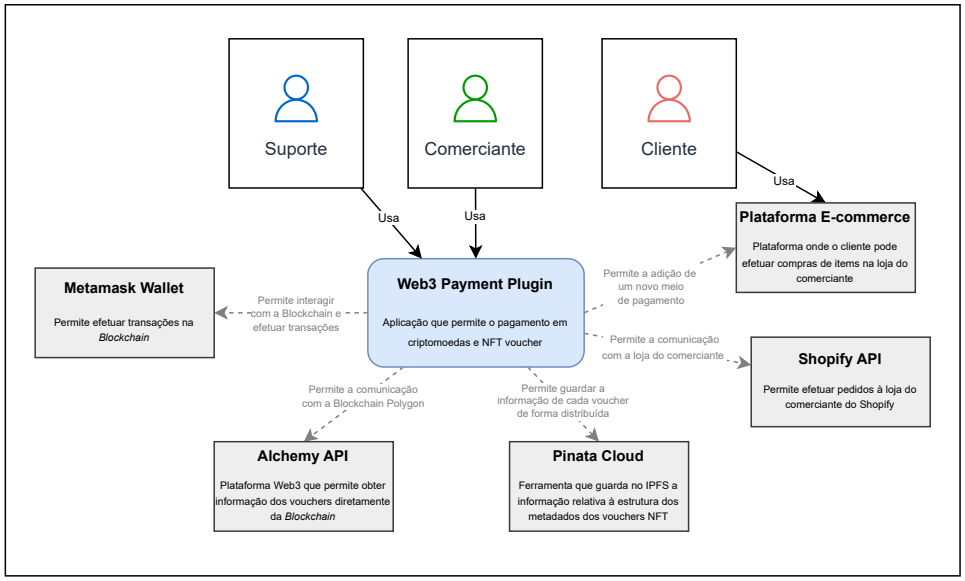


Figura 6.1: Diagrama de contexto da arquitetura

Tal como referido anteriormente houve a necessidade de acrescentar módulos à aplicação de modo a cumprir com os requisitos estipulados. Um dos elementos a adicionar é uma ferramenta de comunicação com a *Blockchain Polygon* de modo a efetuar pedidos e obter informação sobre os vouchers NFT.

Antes do processo de desenvolvimento foram realizados inúmeros tutorias de aprendizagem e integração ao tema pelo que a ferramenta utilizada nesses tutoriais recaiu no Alchemy API [104] ao invés de alternativas como por exemplo a ferramenta Moralis [105] ou Quicknode [106] apresentando também uma boa documentação para os recursos necessários.

Foi necessário também adicionar uma ferramenta que pudesse guardar informação dos metadados dos vouchers de forma descentralizada no IPFS e, para tal, foi escolhida a plataforma Pinata Cloud [107] uma vez que oferece um serviço *Fremium* e a sua integração é algo simples o que foi um fator que contribuiu bastante para a sua utilização em comparação com alternativas como Filebase.

De seguida, na Figura 6.2 bem como no Apêndice C é analisada a composição do sistema em diversos *containers* com o propósito de analisar as interações e as comunicações existentes entre a aplicação e os diversos sistemas externos que a compõem.

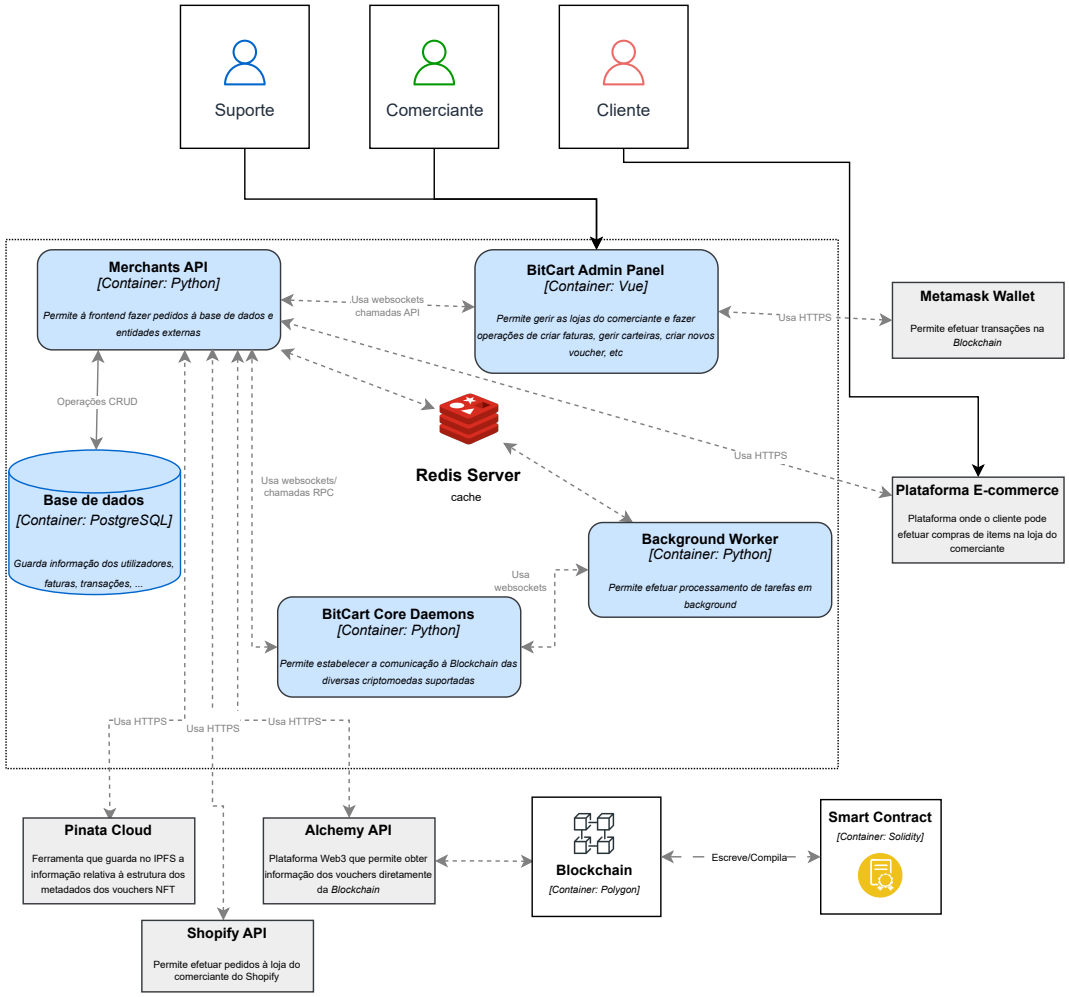


Figura 6.2: Diagrama de componentes da arquitetura

Com o intuito de aprofundar o conhecimento sobre o funcionamento de cada módulo mencionado anteriormente, as próximas subsecções apresentarão uma explicação detalhada sobre cada um deles.

BitcartCC Core Daemons

O *daemon* BitcartCC é apenas um *wrapper* em torno da *wallet eletrum* [108], estendendo a funcionalidade *daemon* existente e fornecendo mais métodos, carregamento da *wallet* em tempo real, melhor entrega de eventos e uma especificação para melhores exceções para melhorar o tempo de desenvolvimento.

O *electrum* é uma carteira *Simple Payment Verification (SPV)*, conectando-se a vários servidores públicos e verificando-os, selecionando apenas os válidos. Este mecanismo permite verificar transações sem a necessidade de sincronizar com toda a rede *Blockchain*, focando apenas nos nós confiáveis que fornecem informações relevantes.

Ou seja, este módulo é responsável por efetuar a comunicação à *Blockchain* das moedas aceites na loja de modo a validar transações, obter confirmações ou até mesmo obter informações na *Blockchain* que sejam necessárias para a aplicação como por exemplo o saldo de uma determinada carteira.

Merchants API

Este módulo é o responsável pela gestão da API onde fornece um conjunto de recursos de modo a gerir dados comuns da plataforma, como por exemplo consultar saldo, gerir pedidos, visualizar histórico e dispõe de todos os *endpoints* necessários para o bom funcionamento da aplicação.

Para garantir o funcionamento correto da API, são necessárias algumas instalações adicionais. Primeiramente, o **PostgreSQL** é utilizado como meio de armazenamento de dados, fornecendo um ambiente seguro e confiável para armazenar informações importantes da plataforma. Além disso, a ferramenta **Redis Server** é instalada como meio de *caching* e comunicação eficiente entre os processos, garantindo um desempenho otimizado da aplicação.

Outro componente essencial é o **Background Worker**, responsável por executar tarefas em segundo plano. Essas tarefas podem incluir a verificação de atualizações, a disponibilidade de serviços, a consulta do estado de pagamentos e outras atividades que não precisam ser realizadas imediatamente ou em tempo real. A instância do **Background Worker** permite que essas tarefas sejam executadas de forma assíncrona, sem interromper a funcionalidade principal da plataforma.

Alchemy API

Para a utilização de *vouchers* NFT foi utilizada a plataforma Alchemy que corresponde a uma plataforma de desenvolvimento Web3 que dispõe de um conjunto de ferramentas que permite construir aplicações que interagem com a *Blockchain* sem a necessidade de gerir a infraestrutura da *Blockchain*. Deste modo através da sua API, esta plataforma permite a construção de aplicações Web3 de forma mais rápida acelerando o processo de desenvolvimento. [104]

Esta ferramenta através da sua API permite aceder a informações na *Blockchain* e será importante para obter informações como a lista de vouchers de um determinado endereço bem como os detalhes de cada *voucher* presente numa determinada carteira.

Pinata Cloud

A ferramenta Pinata Cloud desempenha um papel essencial na arquitetura, sendo responsável por armazenar as informações de cada *voucher* no *IPFS (InterPlanetary File System)*. Essa integração traz diversas vantagens significativas para o projeto.

O IPFS oferece um sistema de armazenamento descentralizado, o que significa que os dados são distribuídos numa rede global de nós, eliminando a dependência de servidores centralizados. Isso resulta numa maior robustez e disponibilidade dos dados, uma vez que não há um único ponto de falha.

Esta informação presente no IPFS será importante no processo de criação de um *voucher* de modo a agregar toda a informação sobre o seu valor para que esta informação fique disponível para o utilizador. [107]

Admin *Panel*

Existe ainda um painel de administrador, que corresponde a uma interface onde o comerciante consegue ter acesso a todas as métricas, definições, produtos da sua loja, criação de novas *wallets* de forma a aceitar novas moedas, entre outras funcionalidades.

Esta foi construída sob a *MerchantsAPI*, logo requer a sua comunicação de forma a ter acesso às demais funcionalidades.

Dentro desta ferramenta, o comerciante consegue fazer a gestão da sua loja, inclusive fazer a gestão das suas carteiras, faturas e *vouchers*.

6.2 Tecnologias a adotar

Nesta secção, serão descritas as tecnologias a adotar para cada uma das camadas descritas anteriormente.

Como se trata de uma solução pré-existente, as escolhas para as tecnologias tiveram um grande impacto nas tecnologias já existentes da solução, com base a facilitar futuras integrações na plataforma.

Backend

Esta plataforma foi construída utilizando com linguagem de programação principal *Python*, mais concretamente a versão 3.9 [109] que foi utilizada no desenvolvimento dos componentes BitCartCC Core Daemons e Background Worker.

Merchants API

De forma a proceder à construção da API, foi utilizada uma *framework* baseada em *Python* chamada FastAPI [110]. O FastAPI é uma framework Python que oferece um conjunto abrangente de ferramentas para a implementação de interfaces REST.

Uma das vantagens notáveis do FastAPI é a sua integração com o Swagger [111], que permite visualizar e interagir com os endpoints da API de forma natural e documentada. Essa integração simplifica o desenvolvimento e teste da API, além de fornecer uma documentação detalhada dos endpoints, parâmetros e respostas disponíveis.

Base de Dados

Para a construção da base de dados, foi utilizada a solução já existente que recaía sobre a utilização de PostgreSQL, uma base de dados relacional [112].

Frontend

Quando à tecnologia estipulada para a *frontend*, a escolha recaiu no *NUXT.js* [113] que representa uma *framework* de VUE.js utilizada já no ponto de partida da solução encontrada que foi utilizada para desenvolver as funcionalidades do *BitCartCC Admin Dashboard*.

6.3 Desenvolvimento Preliminar

Este capítulo é dedicado a uma análise preliminar do ponto de partida da solução escolhida que corresponde à ferramenta *open-source* BitCartCC, de modo a analisar com mais detalhe as suas funcionalidades.

6.3.1 Análise do processo de pagamento

Através da Figura 6.4 é possível representar o processo pelo qual uma fatura é gerada até ao momento do cliente efetuar o pagamento representado pelo *modal* da Figura 6.3. Este processo consiste nos seguintes passos:

1. O cliente no ecrã de *checkout* seleciona a opção de pagar com criptomoedas;
2. De seguida, a *frontend* carrega o *script* do *modal* e fica à espera de informação;
3. É enviado um pedido à API de forma a criar uma fatura com o montante estipulado e com o identificador da loja (*store_id*);
4. Após a criação do pedido é retornado o identificador da fatura criada (*invoice_id*);
5. O *modal* representado pela Figura 6.3 é apresentado ao cliente e este pode pagá-lo com as moedas suportadas pela loja.

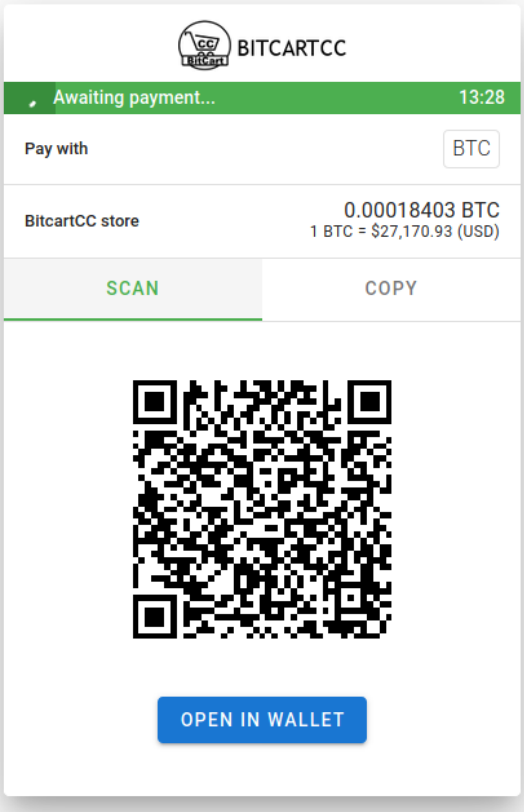


Figura 6.3: *Modal* com o pedido de pagamento para o utilizador

Através do *modal* representado pela Figura 6.3, o utilizador consegue escolher a criptomoeda com que deseja pagar (caso a loja suporte mais do que um tipo de criptomoeda) e, dispõem de um QR-Code de modo a que este processo seja mais simples. Caso a *wallet* do cliente não disponha de mecanismos de *scan*, este também consegue obter o identificador da transação na aba *Copy*, onde consegue visualizar o endereço de envio e o montante a enviar.

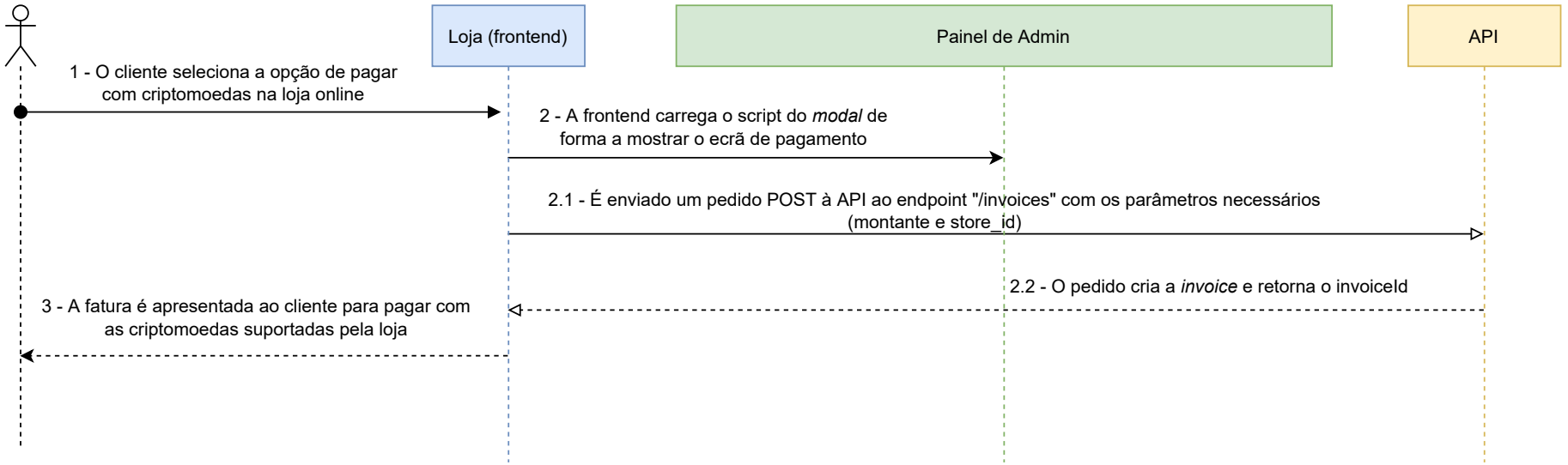


Figura 6.4: Modo de funcionamento do processo de pagamento

6.3.2 Gestão do progresso da transação

Esta solução, dispõem de mecanismos de gestão da transação, de forma a determinar quando é que um montante foi pago. Como tal, uma transação pode corresponder aos seguintes estados: *Pending* (em progresso), *Complete* (fatura paga), *invalid* (erro inesperado) ou *expired* (fatura expirada). E, de modo a analisar a mudança de estados das transções foi analisado o seu processo desde o momento de pagamento até à sua conclusão através do diagrama de sequência representado pela Figura 6.5.

Este processo corresponde a uma especificação do diagrama de sequência de análise do processo de pagamento representado pela Figura 6.4. Consiste nas seguintes etapas:

1. Confirmação do pagamento por parte do utilizador;
2. De seguida, é feita uma comunicação à *Backend* de modo a criar a fatura na base de dados e adicionar as criptomoedas suportadas pela loja. Para além de obter os métodos aceites, este cria também o método de pagamento para cada criptmoeda aceite obtendo para tal o seu símbolo (ETH, BTC) a sua divisibilidade e a sua taxa atual através da comunicação com uma API externa.
3. Após esta informação, o *modal* é apresentado ao cliente onde este pode escolher com que criptomoeda irá proceder ao pagamento;
4. Quando o cliente pagou, é criada uma *websocket* que irá ser responsável pela comunicação com a *Blockchain* e verifica se nas transações recebidas existe alguma cujo parâmetro "*:to*" corresponda ao endereço do comerciante. Caso essa ação seja verdadeira, significa que a transação já se encontra na *Blockchain* e é originado um novo evento denominado *new_transaction*;
5. Este evento notifica uma função que vai ser responsável por alterar o estado da transação na base de dados de *pending* para *complete* ou *expired*.

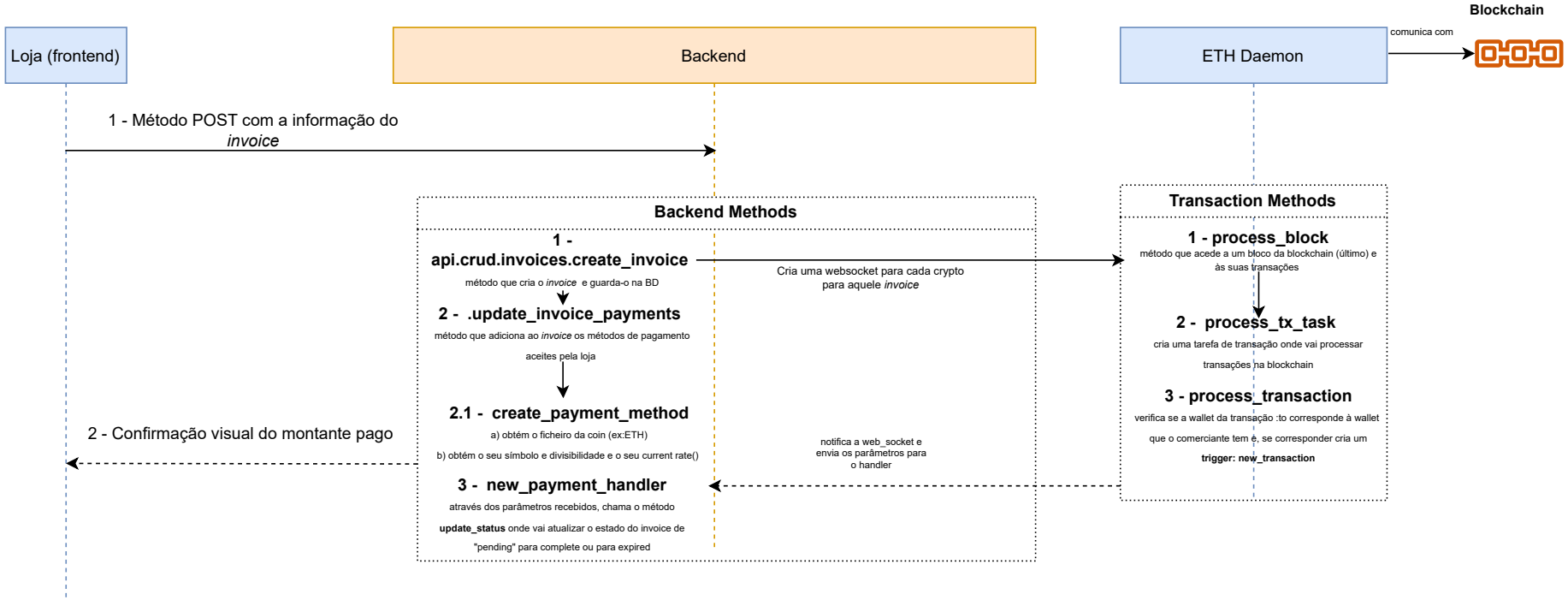


Figura 6.5: Processo de gestão do estado de uma transação desde a sua criação à sua conclusão

6.3.3 Integração com a plataforma de *e-commerce* Shopify

Através do BitCartCC, ferramenta escolhida para o ponto de partida da solução, já é possível integrá-la na loja *Shopify* de cada utilizador. No entanto existem alguns pré-requisitos necessários de modo a que esta integração seja possível.

Como se trata de uma integração na plataforma Shopify, antes de mais é necessário uma conta na plataforma de *e-commerce* de modo a proceder à integração da ferramenta. De seguida, é necessário uma instância do BitCartCC com uma loja criada e pelo menos uma *wallet* de modo a que seja possível efetuar pagamentos.

Ainda dentro do *Admin Panel*, é necessário elaborar umas configurações identificadas na documentação do BitCartCC [114].

Após estas condições, a Figura 6.6 demonstra o processo seguido pela aplicação de modo a comunicar com os serviços do Shopify.

- Este processo começa com o utilizador a aceder uma página web hospedada nos serviços do Shopify. De seguida, no processo de *checkout*, é criado um novo botão simbolizando um novo meio de pagamento em adição aos já existentes e suportado pela loja do comerciante;
- Após clicar no botão de pagar, é feito um pedido à *Backend* de modo a criar uma fatura com o montante específico e são validadas também as credencias do comerciante, nomeadamente o seu nome da loja, *APIKey* e *API Acess Token*, de modo a confirmar que o comerciante consegue utilizar os serviços e API do Shopify com o propósito de criar as faturas e estas ficarem registadas;
- Após esta validação, é devolvido o identificador da compra e aparece o *modal* para o utilizador pagar.
- De seguida o utilizador dispõem da opção de pagar identificada de modo semelhante pela Figura 6.3 e o processo de comunicação e atualização de estados é igual ao definido na Subseção 6.3.2

Após o término da transação, o comerciante nos seus pedidos, consegue visualizar o tipo de pagamento e o estado do mesmo realizados com esta nova solução.

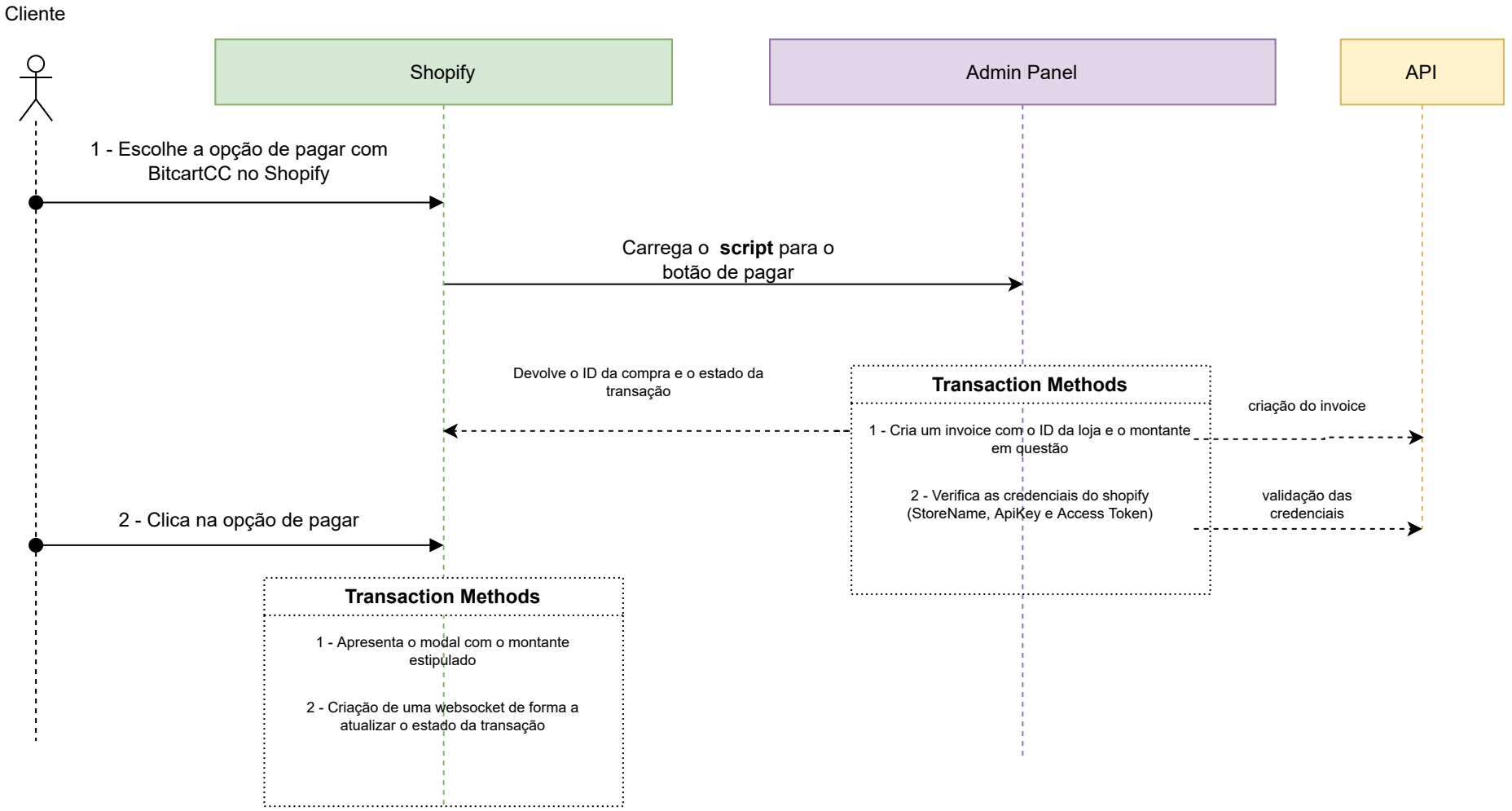


Figura 6.6: Processo de gestão do estado de uma transação desde a sua criação à sua conclusão

Com os diagramas de sequência elaborados, foi possível compreender o modo de funcionamento da plataforma através da identificação dos diferentes processos e módulos por onde passam.

Esta análise corresponde ao ponto de partida para o mecanismo de desenvolvimento das restantes funcionalidades que irão ser desenvolvidas na presente solução.

Capítulo 7

Organização e Estrutura do Projeto

Neste capítulo, serão abordados os aspetos relativos à organização e estrutura do projeto. Para garantir uma gestão eficiente e organizada, foram utilizadas ferramentas como o JIRA Software [115] para o acompanhamento e atribuição de tarefas e o *workflow* do GitLab para o controlo de versões e colaboração. Será realizada a identificação dos repositórios e as suas estruturas de modo a compreender a função de cada pasta dentro do projeto e a estrutura adotada.

7.1 Gestão do Projeto

De modo a eleger o procedimento utilizado durante o segundo semestre, a equipa tomou uma decisão conjunta, tendo em conta a sugestão do estagiário que consistiu na utilização da ferramenta JIRA Software [115].

Para isso, foi configurado um projeto no JIRA Software no início do desenvolvimento de modo a escalonar as tarefas tendo em conta *sprints* com a duração de 15 dias cada um. Esta ferramenta possibilita a manutenção de um registo de cada *sprint*, assim como a avaliação de seu desempenho, de modo a determinar o esforço aplicado tendo em conta a estimação das tarefas.

Além disso, o JIRA permite a documentação de informações relevantes, as quais são acessíveis a todos os membros da equipa através da sua ferramenta Confluence [116]. A Figura 7.1 representa a organização do Quadro JIRA em um dos *sprints* finais em que se encontra dividido em quatro colunas para cada *sprint*:

- ***To do***: coluna que representa as tarefas que ainda não foram iniciadas;

- ***In progress***: coluna que indica as tarefas que já foram iniciadas, mas ainda não foram concluídas;

- ***Revision***: tarefas que já foram concluídas, mas que ainda necessitam de revisão;

- ***Done***: representa as tarefas que foram concluídas e aprovadas.

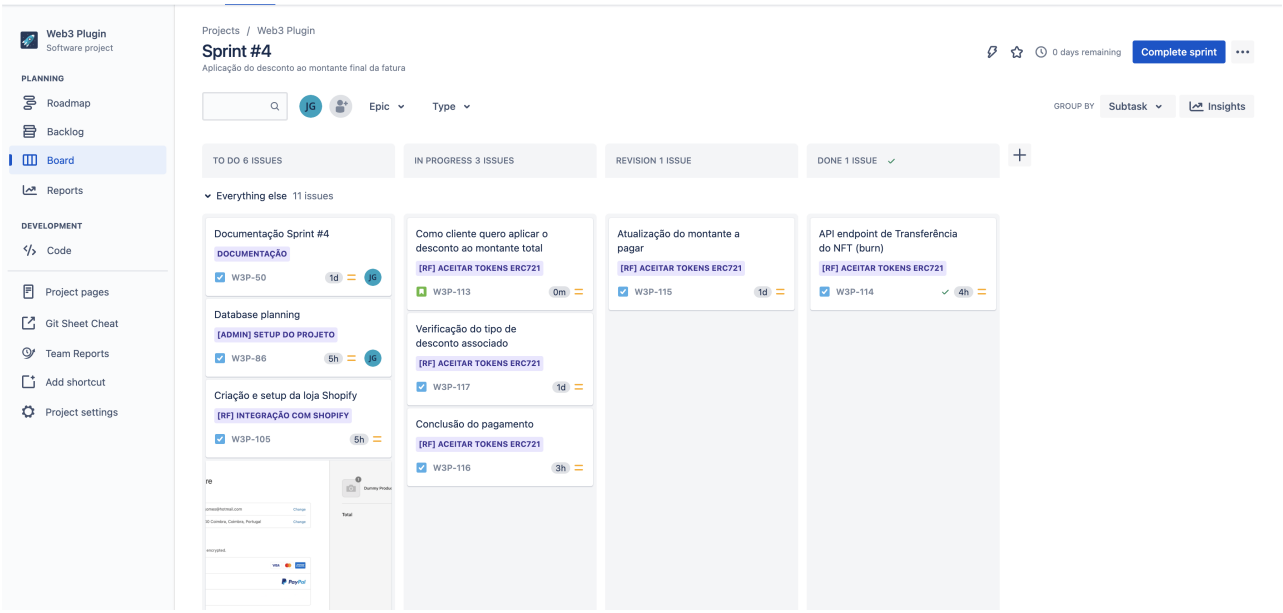


Figura 7.1: Quadro Jira Software

Para que as tarefas fossem aplicadas, foi necessário criar sprints com a duração de 15 dias cada um e definir *user Stories* que representam as funcionalidades desejadas para a aplicação. Essas *user Stories*, por sua vez, deram origem a tarefas que foram criadas, estimadas e realizadas, de modo a que estas fossem concluídas dentro dos prazos estipulados para cada *sprint*.

Através da Figura 7.2 é possível verificar o fluxo por onde as tarefas podem passar. Começando pela sua criação, as tarefas encontram-se no *backlog* de cada *sprint*, onde após o seu começo ficam disponíveis no quadro Jira, representado pela Figura 7.1.

No entanto caso exista algum problema com a tarefa em revisão, esta passa para a coluna "TO DO" e serão efetuadas as alterações necessárias de modo a que a tarefa consiga acabar o fluxo de execução na coluna "DONE". No final de cada *sprint* na eventualidade de existirem tarefas não terminadas, estas passam para o *sprint* seguinte onde serão as primeiras a serem realizadas.

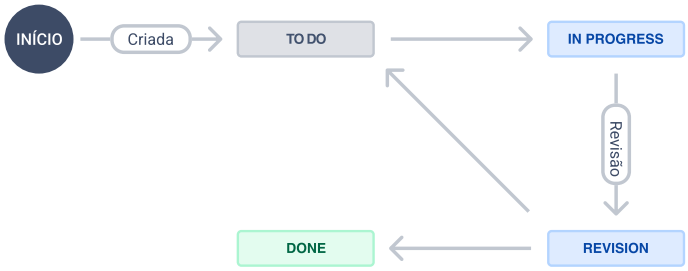


Figura 7.2: Jira *workflow*

Após a criação das tarefas a antes do começo do desenvolvimento, o *Scrum Master* procedeu à criação de um repositório GitLab para o desenvolvimento da solução que foi utilizado como ferramenta de controlo de versão e colaboração no desenvolvimento do código. Com esta ferramenta, foi possível manter um registo do histórico de alterações no código, bem como gerir os *branches* de desenvolvimento de cada funcionalidade que seguiam um modelo de trabalho semelhante ao representado pela Figura 7.3.

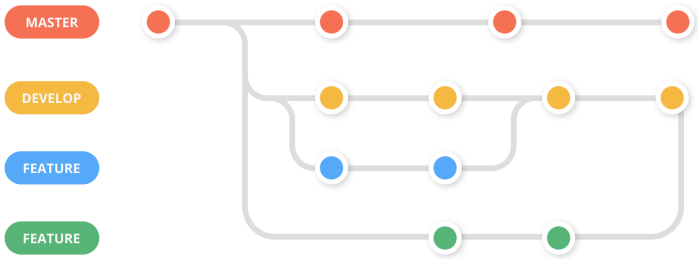


Figura 7.3: GitLab *workflow* [117]

Uma vez que a equipa de desenvolvimento era apenas constituída por um elemento, não havia a necessidade de proceder à criação de um novo *branch* sempre que fosse desenvolvida uma nova funcionalidade, no entanto, é considerado boa prática este modo de funcionamento pelo que foi também adotado durante a realização deste projeto.

7.2 Comunicação

De forma a que a equipa se mantivesse em contacto durante todo o processo de desenvolvimento e pesquisa estabeleceram-se os seguintes meios de comunicação:

- **Zoom**: plataforma utilizada para a elaboração de reuniões de forma remota;

- **RocketChat e Discord**: aplicações utilizadas para comunicar entre elementos da empresa e/ou da equipa;

- **Email**: utilizado para a troca de documentos ou marcação de reuniões.

7.3 Estrutura do projeto

As próximas subsecções visam expor a estrutura dos repositórios de *frontend* e de *backend*.

7.3.1 Frontend

A estrutura do repositório de *frontend* adota a solução fornecida pelo BitCartCC, em consonância com as funcionalidades que estão a ser desenvolvidas com base nesta solução. A Figura 7.4 ilustra a estrutura seguida pelo projeto.

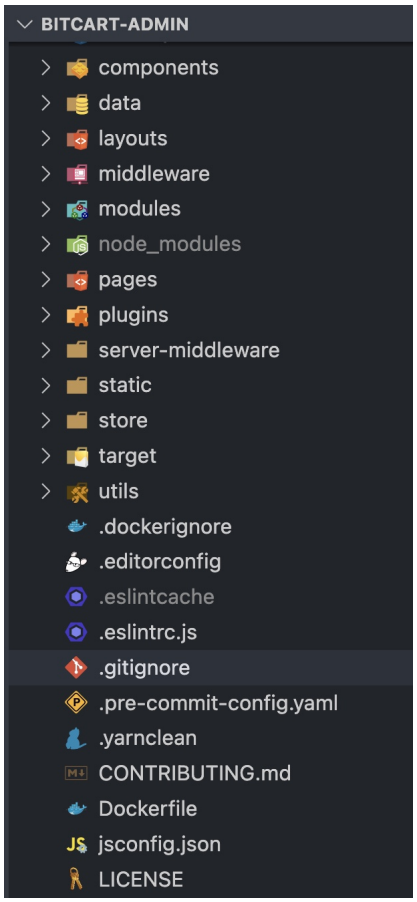


Figura 7.4: Estrutura Frontend

/components: esta pasta contém todos os componentes que foram criados para as diversas funcionalidades do projeto;

/layouts: a pasta "layouts" é utilizada para definir a estrutura de uma determinada página, permitindo assim a criação de uma estrutura uniforme ao longo de toda a aplicação;

/pages: esta pasta é responsável por definir o conteúdo das diferentes páginas da aplicação através da utilização de diversos componentes existentes na diretoria "/components";

/server-middleware: na *framework* NUXT, "server-middleware" são funções que são executadas no servidor antes que a página seja renderizada. Na nossa aplicação, é utilizado para fornecer um *script* personalizado para a comunicação com a plataforma de *e-commerce* Shopify;

/static: esta diretoria tem os ficheiros necessários para a integração do método de pagamento em outro tipo de soluções, como por exemplo o Shopify;

/utils: a diretoria é responsável por conter uma série de métodos que são amplamente utilizados em diferentes secções da aplicação. Esta abordagem visa tornar a manutenção do código mais simples e eficiente, permitindo que os métodos sejam reutilizados em várias partes da aplicação, o que leva a um desenvolvimento mais consistente e escalável.

Com o propósito de cumprir com as regras e *coding standards* do repositório existente, existe o ficheiro ".eslintrc.js" que contém as configurações do ESLint. Essas configurações definem as regras para o projeto em relação à sintaxe do JavaScript, estilo de código, boas práticas, uso inadequado de variáveis e potenciais problemas de segurança. O objetivo final do LINT é produzir um código mais legível, confiável e fácil de manter, além de promover a consistência e a qualidade do código ao longo do tempo.

7.3.2 Backend

Esta subsecção tem como objetivo apresentar a estrutura do repositório de *backend*, que foi baseada no repositório principal da aplicação BitCartCC.

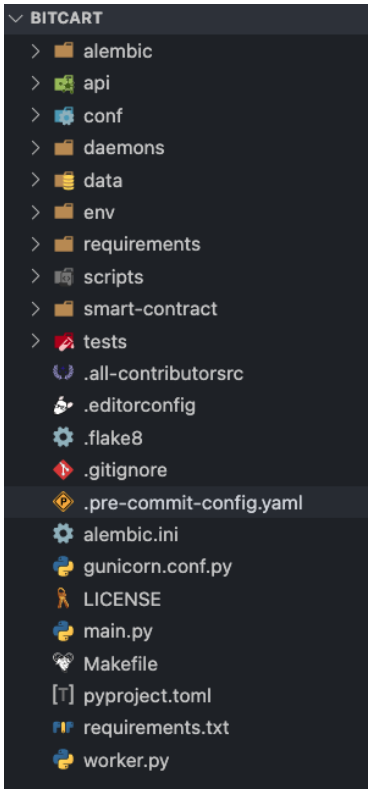


Figura 7.5: Estrutura Backend

Através da Figura 7.5 é possível visualizar a divisão efetuada no repositório de *backend*. Este repositório é responsável por gerir e processar toda a informação presente na nossa aplicação através da comunicação com a API e dos seus diversos *endpoints*.

/api: esta diretoria é a responsável por conter todos os *endpoints* necessários para o funcionamento da nossa aplicação, construídos recorrendo à *framework* python FastAPI. Está dividida de acordo com as seguintes diretorias:

- **/crud:** responsável pela existência dos métodos CRUD das diversas classes existentes;
- **/ext:** contém diversos métodos que são

utilizados frequentemente em toda a lógica de backend;

- **/views:** esta diretoria é a responsável por definir os *endpoints* da aplicação e a lógica de processamento da informação tendo em conta os pedidos HTTP recebidos;

/conf: esta pasta é a responsável pela existência de um ficheiro ".env" que contém todas as variáveis de ambiente necessárias para a execução do projeto sem as expor no *source-code*;

/daemons: através desta diretoria é possível efetuar a comunicação com um determinado *token*, de modo a conectar à sua *Blockchain* e processar pedidos em tempo real;

/requirements: esta diretoria é a responsável por identificar as bibliotecas necessárias para o funcionamento da aplicação. Através da instalação de todos os recursos necessários estes serão inseridos na diretoria "/env" que contém informação sobre o ambiente virtual python criado para a presente aplicação;

/smart-contract: é nesta diretoria que estão definidos of ficheiros relativos ao *smart-contract* criado para a utilização dos vouchers, bem como o seu *Application Binary Interface (ABI)*, que descreve a interface pública do contrato e permite que outras aplicações possam interagir com ele garantindo a interoperabilidade entre diferentes contratos bem como a transmissão de dados forma correta e segura. Na Seção 8.2 são referidos aspetos mais técnicos em relação aos métodos e propriedades do *smart-contract*.

De forma similar ao repositório de *frontend*, também neste foi inserido um conjunto de regras de modo a que todo o código produzido siga um padrão e seja consistente, que estão presentes no ficheiro ".editorconfig".

Capítulo 8

Desenvolvimento

Este capítulo descreve a fase de desenvolvimento ocorrida durante a segunda metade do estágio, dedicada à execução das tarefas planeadas com base nos requisitos recolhidos no primeiro semestre e melhorados já no segundo semestre. Além disso, apresenta uma lista das tarefas realizadas durante esse período de acordo com a sua posição na arquitetura, seguindo a estrutura apresentada na Figura 6.1, as quais foram estabelecidas a partir dos requisitos previamente identificados.

8.1 Escolha de Tecnologias Adicionais

Como foi referido no início do Capítulo 7 a primeira tarefa consistiu na realização de *user stories* de modo a determinar as tarefas a realizar e as calendarizar tendo em conta *sprints* com a duração de 15 dias. A tarefa seguinte consistiu na integração e aprendizagem de algumas tecnologias nomeadamente *Blockchain*, *smart contracts* a biblioteca Web3.js e VUE onde foram realizados alguns tutoriais e tomadas algumas decisões entre as quais:

- **Adoção da plataforma de desenvolvimento Web3 - Alchemy**: corresponde uma plataforma Web3 que oferece uma API que comunica com a *Blockchain* de forma simples, permitindo um desenvolvimento mais ágil e eficiente. Ao simplificar a comunicação com a *Blockchain*, a ferramenta Alchemy [104] elimina a necessidade de gerir uma infraestrutura complexa, oferecendo uma velocidade de desenvolvimento superior;

- **Escolha do sistema de armazenamento IPFS**: com o propósito de guardar a informação relativa aos metadados dos *vouchers*, foi escolhida a ferramenta Pinata[107] que permite o armazenamento e gestão de arquivos no

Interplanetary File System (IPFS), uma rede distribuída *peer-to-peer* que permite a partilha de informação de forma descentralizada e que não pode ser alterada;

Através da realização de diversos tutoriais, o autor construiu as bases necessárias para proceder à fase de desenvolvimento já dentro da ferramenta BitCartCC incorporando as ferramentas Alchemy e Pinata nos tutoriais sempre que possível.

8.2 *Smart-contract* - detalhes e funcionalidades

Nesta secção, será abordada a implementação do padrão ERC-721 no nosso *smart contract* através da ferramenta Remix - Ethereum IDE [118] utilizando a linguagem de programação Solidity [119] que contém os seguintes métodos:

- ***safeMint***: cria um novo *token* e atribui-o a um endereço específico. O método recebe como parâmetros o endereço de destino para o novo *token* e um *Uniform Resource Identifier (URI)* para a sua representação que corresponde a um endereço onde está presente a informação relativa aos metadados do voucher;
- ***safeTransfer***: transfere um *token* da posse do remetente para o destinatário especificado, desde que o remetente seja o proprietário atual do *token*. O método recebe como parâmetros o endereço de origem, o endereço de destino e o identificador do *token*;
- ***validateOwnership***: verifica se o endereço de chamada do método é o proprietário atual do *token* especificado. O método recebe como parâmetro o identificador do *token* e retorna um valor binário indicando se o endereço de chamada é o proprietário do respetivo *token*;
- ***getMetadata***: retorna a URI que representa os metadados do *token* especificado, desde que o endereço de chamada seja o proprietário atual do *token*. O método recebe como parâmetro o ID do *token*;
- ***burnToken***: este método queima o *token* especificado, removendo-o do registo de tokens, desde que o endereço de chamada seja o proprietário. O método recebe como parâmetro o ID do *token*.

Com o propósito de testar cada funcionalidade do nosso *smart contract* ERC721, utilizamos a plataforma Remix Ethereum IDE. Através dessa plataforma, foram criados casos de teste para cada um dos métodos desenvolvidos e analisados se

funcionam corretamente de acordo com as regras definidas. Todos os testes passaram com sucesso como é possível verificar pela Figura 8.1.

Após o teste do *smart contract*, o próximo passo consistiu no *deployment* na rede Polygon Mumbai, um ambiente de testes para a rede principal Polygon. Através da compilação do contrato, é gerado um ABI que, tal como dito anteriormente, corresponde a um ficheiro JSON que apresenta todos os métodos presentes no *smart contract* de forma a que o utilizador possa invocar os seus métodos, transmitir e comunicar informação de forma correta.

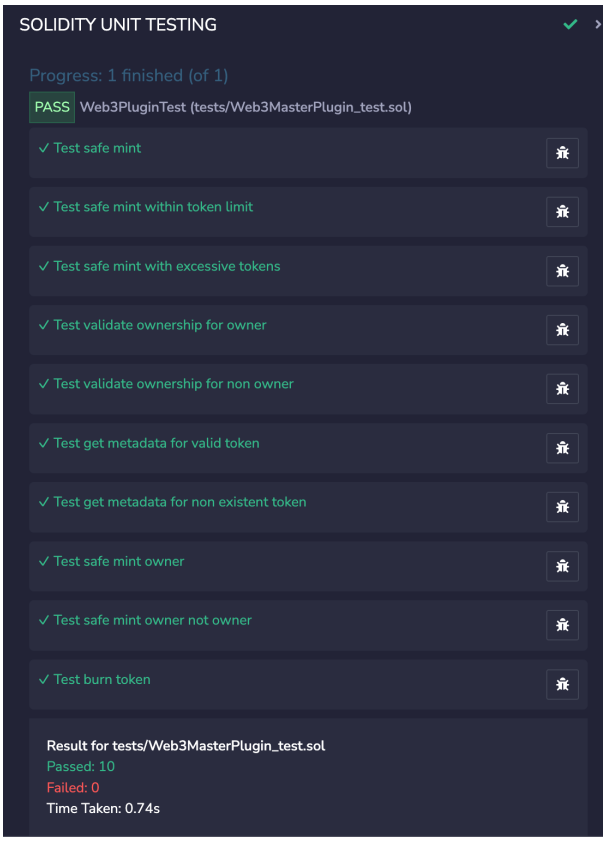


Figura 8.1: Plano de testes do *smart contract*

Após o pagamento da taxa de *deployment* do nosso contracto, este fica disponível na plataforma Mumbai Polygonscan [120] que corresponde a um explorador de blocos da rede Polygon Mumbai, permitindo a visualização e verificação das transações, contratos e endereços na rede.

8.3 Estrutura do desconto NFT

Uma vez que o contracto se encontra acessível na *Blockchain*, houve a necessidade de identificar o tipo de descontos a suportar e como é que estes seriam validados. A solução passou pela criação de uma estrutura de metadados compatível com a

estrutura oficial do standard ERC721 criada pelo *markeplace* OpenSea[121], ilustrada pela Figura 8.2 que é um padrão que é utilizado pelos *developers* de tokens ERC-721.

Como é possível observar pela Figura 8.2 os NFT seguem uma estrutura que contém as seguintes propriedades: [121]

- ***description***: corresponde à descrição do NFT onde é possível usar a linguagem markdown;
- ***external_url***: este é o URL que aparecerá abaixo da imagem do item na plataforma OpenSea e permitirá que os utilizadores saiam do OpenSea e vejam o item no seu site;
- ***image***: este parâmetro corresponde ao URL da imagem do item. Pode ser qualquer tipo de imagem;
- ***name***: corresponde ao nome do NFT;
- ***attributes***: estes são os atributos do item, que aparecerão na página do OpenSea para o item.

Figura 8.2: Metadados standard ERC721

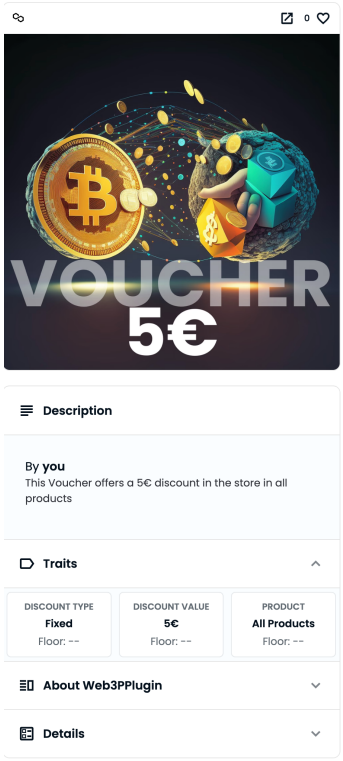
Esta é a estrutura base que qualquer tipo de *token* ERC721 tem de seguir. No entanto, de modo a proceder à criação da estrutura dos vouchers, é necessário identificar que tipo de descontos iremos suportar na nossa loja e como tal foram identificados 3 tipos:

- **Descontos fixos**: tem uma dedução de um montante fixo, como por exemplo 5€;

- **Descontos absolutos**: a sua aplicação resulta na redução de uma percentagem ao montante total da compra como por exemplo um desconto de 10%;
- **Descontos baseados em produto**: este tipo de desconto é aplicado em certo produto, sendo o seu desconto total, fixo ou absoluto como por exemplo um desconto de 50% num determinado artigo da loja.

```
"attributes": [
        {
            "trait_type": "Discount Type",
            "value": "Product-based"
        },
        {
            "trait_type": "Discount Value",
            "value": "Free"
        },
        {
            "trait_type": "Product",
            "value": "Sneakers"
        },
        {
            "trait_type": "Store",
            "value": [ "STORE_ID" ]
        },
        {
            "trait_type": "Product ID",
            "value": [ "SKU_ID" ]
        }
    ]
```

(a) Lista de atributos do voucher NFT



(b) Exemplo de um voucher com desconto fixo na plataforma OpenSea

Após esta especificação, na estrutura dos metadados foram adicionadas as seguintes propriedades, representadas pela Figura 8.3a, à lista de *attributes* presentes nos metadados do voucher, onde *trait_type* é o nome do atributo e *value* é o valor do atributo:

- ***Discount Type***: corresponde ao tipo de desconto que se trata podendo ter o valor de *Product-based*, *Absolute* ou *Fixed*;
- ***Discount Value***: remete para o valor que o desconto tem e, está relacionado com o tipo de desconto;
- ***Product***: este atributo é algo informativo que faz referência ao tipo de desconto que o voucher está associado;

- ***Store***: este parâmetro indica em que loja(s) é que o voucher pode ser utilizado, sendo referenciado pelo identificador da loja proveniente da ferramenta BitCartCC;

- ***Product ID***: caso se trate de um desconto do tipo *"Product-based"*, este dispõem de um atributo extra onde é possível identificar a que produto(s) se destina o presente voucher através da identificação do identificador do produto.

Através da utilização dos metadados descritos anteriormente, foi criado um voucher na plataforma OpenSea[122] cujos atributos, interações e detalhes sobre a coleção presente no NFT podem ser visualizados na Figura 8.3b. Esta figura ilustra de forma clara um exemplo prático da aplicação dos metadados na criação de um NFT e como a plataforma OpenSea apresenta todas as informações relevantes para o utilizador.

8.4 BitCartCC Merchants API

Conforme a representação da Figura 6.1, o módulo central da aplicação é a API REST em Python denominada *BitCartCC Merchants API*, construída sobre a *framework* web FastAPI. Esta API é responsável pelo processamento de todos os pedidos provenientes do painel de administração, da *Blockchain* e da plataforma de *e-commerce*. Além das funcionalidades já mencionadas, é importante destacar os seguintes aspectos da API:

- **Autenticação JWT**: A API utiliza autenticação baseada em JSON Web Tokens, proporcionando um esquema seguro para autenticar e autorizar os pedidos. Isso garante que apenas utilizadores autenticados e autorizados tenham acesso aos recursos e funcionalidades da API;

- **Controlo de permissões**: A API oferece diferentes níveis de permissões para os os seus utilizadores. Desde permissões mais restritas, que permitem apenas a consulta e manipulação das carteiras, por exemplo, até permissões de administrador (*full_controll*) com acesso completo a todas as funcionalidades. Isso permite uma gestão granular de acesso e segurança na utilização da API;

- **Documentação automática**: A API é acompanhada de documentação automática, disponibilizada por meio da interface Swagger UI. Isso facilita o entendimento e utilização da API, fornecendo detalhes sobre os endpoints disponíveis, os parâmetros necessários e as respostas esperadas. A documentação automática agiliza o processo de integração e desenvolvimento de aplicações que utilizam a API [111].

Dessa forma, a presente secção visa expor os diversos *endpoints* que foram criados expondo o seu propósito e comportamento.

8.4.1 GET /vouchers/getNFTS

Este *endpoint* é responsável por retornar uma lista de NFT presentes no endereço do utilizador e que pertençam ao endereço do *smart contract* dos *vouchers*. Para tal, recebe como argumentos o endereço público da carteira do utilizador e a *chain* a que se encontra conectado. De seguida, comunica com a API da ferramenta Alchemy e em caso de sucesso retorna um código HTTP 200 (OK) juntamente com a lista de NFT. Em caso de erro retorna o código HTTP 422 (Unprocessable Entity) juntamente com uma mensagem de erro.

8.4.2 GET /vouchers/nftClient

Este *endpoint* tem como objetivo obter a lista de NFTs que o cliente pode usar durante o processo de pagamento. Para obter a lista de NFTs, é necessário fornecer alguns parâmetros, incluindo o endereço público do utilizador, a *chain*, o ID da loja e os itens do carrinho de compras. De seguida, é feita uma comunicação com a Alchemy API e são obtidos os *vouchers* que o cliente poderá utilizar no processo de pagamento através da filtração dos itens presentes na lista.

Caso exista algum erro no processo, o *endpoint* retornará um erro HTTP 400 (Bad Request) com a uma mensagem de erro. Caso contrário é devolvida a lista de NFT juntamente com um código HTTP 200 (OK).

8.4.3 GET /vouchers/getABI

O *endpoint* "vouchers/getABI" disponibiliza informações relevantes para a interação com o *smart contract* dos vouchers. Em particular, é possível obter o endereço desse *smart contract* e seu ABI, que define a sua interface e permite a invocação dos seus métodos. Essa funcionalidade é crucial, uma vez que permite a execução de transações que envolvem o *smart contract* dos *vouchers*, como a criação e transferência de NFTs.

8.4.4 POST /vouchers/submit

O *endpoint* "POST /vouchers/submit" é utilizado para processar o desconto dos *vouchers* submetidos pelo cliente. Este *endpoint* recebe como parâmetros o ID do *voucher* (voucherID), o ID da fatura (invoiceID), a cadeia de blocos a que o utilizador está conectado (chainID) e o ID da criptomoeda escolhida para o pagamento (id). Após a submissão do NFT, este método analisa os metadados do *voucher* para determinar o desconto a ser aplicado. Em seguida, com base no tipo de desconto, é retornado o valor a ser descontado na fatura na criptomoeda escolhida pelo cliente. O *voucher* fica depois selecionado e o cliente pode, então, efetuar o pagamento do valor restante em criptomoeda selecionada.

8.4.5 POST /vouchers/create

Um dos requisitos que não estava inicialmente previsto era a capacidade do comerciante criar os seus próprios descontos/vouchers. No entanto, após analisar as funcionalidades e o valor que estas trariam ao estágio, ficou decidido que seria implementada a funcionalidade acima descrita e, como tal é através deste *endpoint* que a informação será passada e o *voucher* será criado. Este *endpoint* recebe como parâmetros informação relativa aos metadados do *voucher* que foram obtidos através do preenchimento de um formulário por parte do comerciante no seu painel de administração.

De seguida, irá ser criada a imagem do *voucher* que difere de desconto para desconto uma vez que é gerada de forma dinâmica, ou seja, tendo em conta o tipo de desconto e o valor do desconto será criada uma imagem tendo em conta essa informação através do uso da biblioteca *Python Imaging Library (PIL)*[123] onde será posteriormente publicada para a rede IPFS. Após a criação da imagem será criado os metadados do *voucher* utilizando o endereço IPFS da imagem no atributo "image", semelhante ao representado pela Figura 8.2.

De forma análoga ao processo de upload de imagem, os dados do *voucher* são publicados na rede IPFS por meio do serviço Pinata[107], que retorna um endereço correspondente à localização dessas informações. Com base nesses dados, é possível invocar o *smart contract* e o método ***safeMint*** utilizando o endereço da carteira do comerciante e o URL fornecido pelo IPFS. Esse procedimento desencadeia uma interação com o *smart contract*, que resulta na criação do *voucher* e na sua disponibilização na carteira do comerciante.

8.4.6 GET /vouchers/getShopifyProducts

O endpoint "/shopify-products" é responsável por realizar a comunicação com a API do Shopify, tendo em conta o ID da loja (storeID) fornecido. Através desta comunicação, a aplicação obtém todos os produtos presentes na loja e devolve-os ao utilizador para que sejam utilizados na *frontend* da aplicação.

Caso ocorra algum erro no pedido efetuado à API do Shopify ou no processamento da aplicação, será devolvida uma mensagem de erro acompanhada do respetivo código. Por outro lado, se o pedido for processado com sucesso, serão devolvidos todos os produtos da loja em questão, juntamente com o código HTTP 200 para indicar que a operação foi bem sucedida.

8.4.7 GET /vouchers/stats

Este vai ser o responsável por apresentar no ecrã do painel de administrador informação sobre o número de vouchers, como se pode verificar pela Figura 8.4, que o comerciante dispõe na sua carteira e este valor será atualizado sempre que se procede à criação, transferência ou eliminação de um determinado voucher.

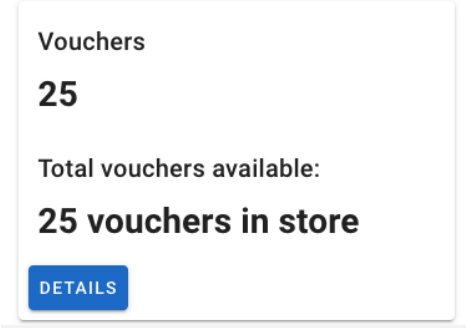


Figura 8.4: Painel de Administrador - Vouchers

A Figura 8.4 está incluída no painel de controlo da ferramenta BitCartCC cujas funcionalidades serão descritas com maior detalhe na Seção 8.5.

8.5 BitCartCC Admin Dashboard

Esta seção destina-se a apresentar o módulo do Painel de Administrador da ferramenta, que foi desenvolvido em torno da *Merchants API*. Esta interface oferece recursos avançados de edição, sendo muito poderosa graças à sua construção com as ferramentas de Material Design[124]. Adicionalmente, o painel de administração

fornece uma página de *checkout* universal e um *script* que pode ser utilizado para lançar a ferramenta de pagamento em qualquer website.

Aqui serão apresentadas em detalhe as funcionalidades desenvolvidas de acordo com os requisitos recolhidos.

8.5.1 Dashboard

A página inicial da nossa aplicação, representada pela Figura 8.5, na qual é possível visualizar detalhes e informações sobre as funcionalidades já suportadas. Entre elas, destaca-se a possibilidade de adicionar *crypto wallets*, visualizar o saldo das carteiras, gerir lojas e criar faturas, sendo que também foi incluída a funcionalidade de gerir e visualizar *vouchers* NFT.

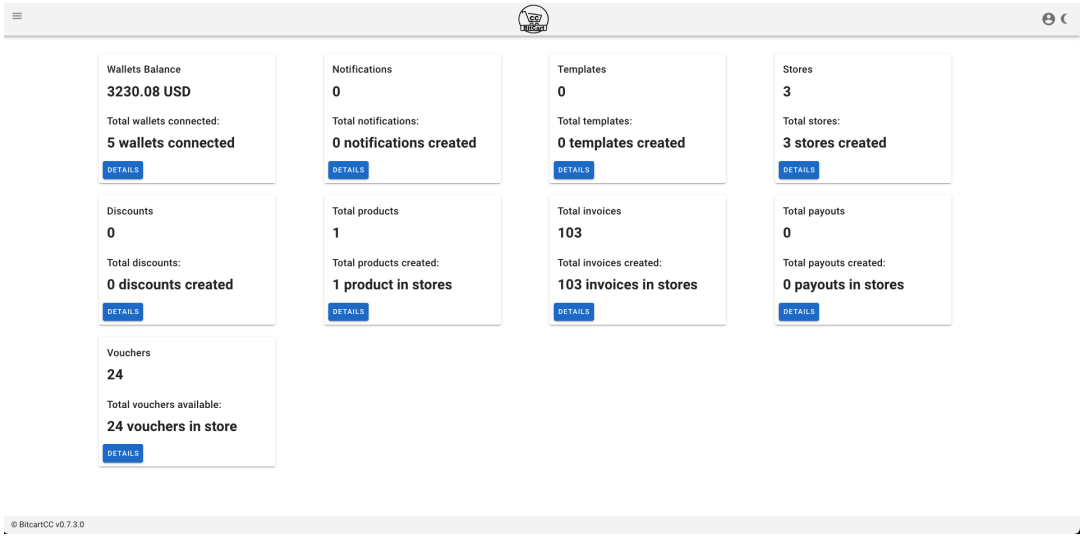


Figura 8.5: BitCartCC Dashboard

8.5.2 Gestão de vouchers

Para a funcionalidade de gestão dos *vouchers*, foi criada uma nova página "/vouchers" representada pela Figura 8.6 onde o comerciante após se autenticar com a sua carteira Metamask consegue visualizar todos os *vouchers* que possui na sua carteira e consegue inclusive obter algumas informações sobre os mesmos como por exemplo o seu título, descrição, tipo de desconto e preço associado.

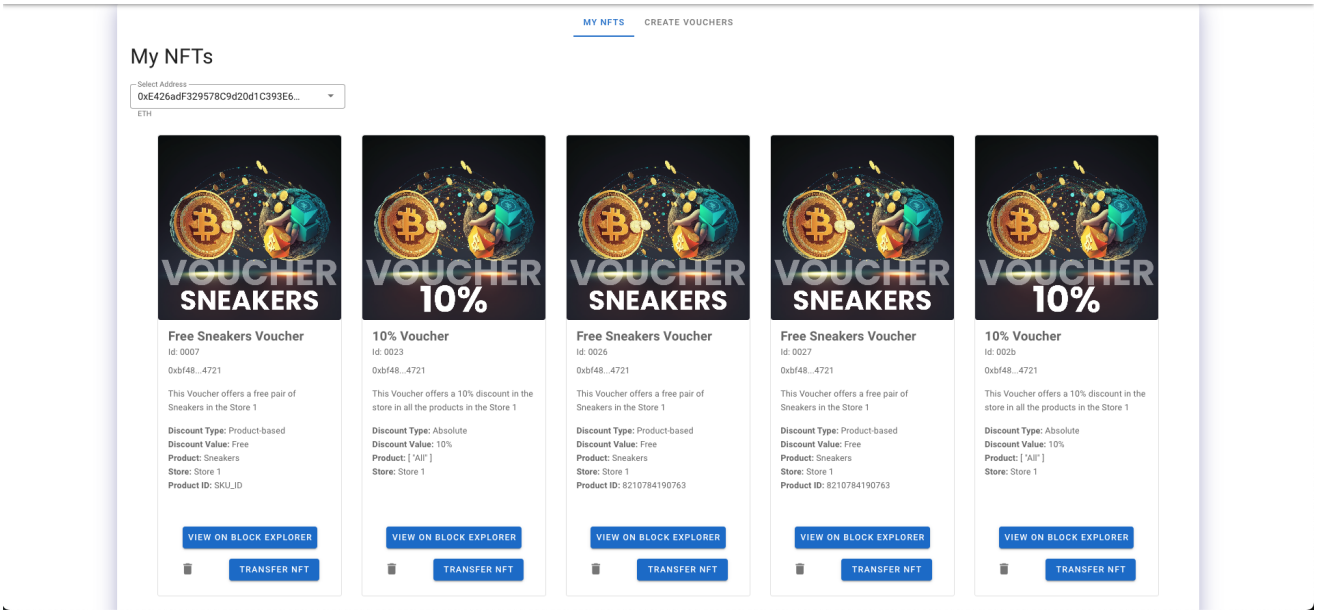


Figura 8.6: BitCartCC Dashboard

Para além desta informação ainda consegue realizar três ações dentro da página:

1. **Visualizar o voucher num explorador de blocos**: o comerciante consegue visualizar o histórico do *voucher* incluindo transações passadas no explorador de blocos Mumbai PolygonScan[120];

2. **Eliminar o voucher**: uma ação adicional disponível na página é a possibilidade de eliminar o *voucher*. Ao clicar no ícone correspondente, o comerciante pode interagir com o *smart contract* e executar o método *burn*, que irá remover permanentemente o *voucher* identificado pelo seu id;

3. **Transferir o voucher**: após pressionar o botão *"Transfer NFT"*, o comerciante será levado para uma nova página "vouchers/:nft_id" representada pela Figura 8.7 onde após inserir um endereço válido de destino é invocado o método ***safeTransfer*** do *smart-contract* e o *voucher* é transferido para esse endereço.

8.5.3 Criação de vouchers

A página "/vouchers" possui duas abas distintas. A primeira é responsável por apresentar os cupões disponíveis e realizar as operações já mencionadas. Já a segunda tem como objetivo permitir que o comerciante crie *vouchers* personalizados, através

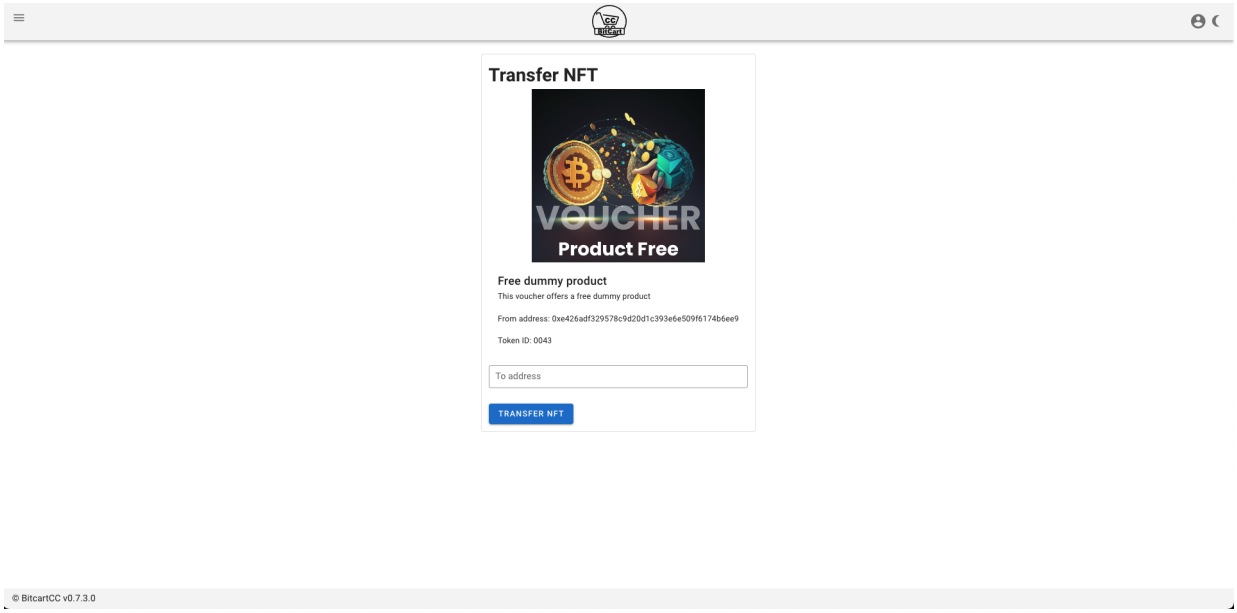


Figura 8.7: Página de transferência de *voucher* do BitCartCC

de um formulário que segue a estrutura de metadados descrita na Seção 8.3, como é possível observar pela Figura 8.8.

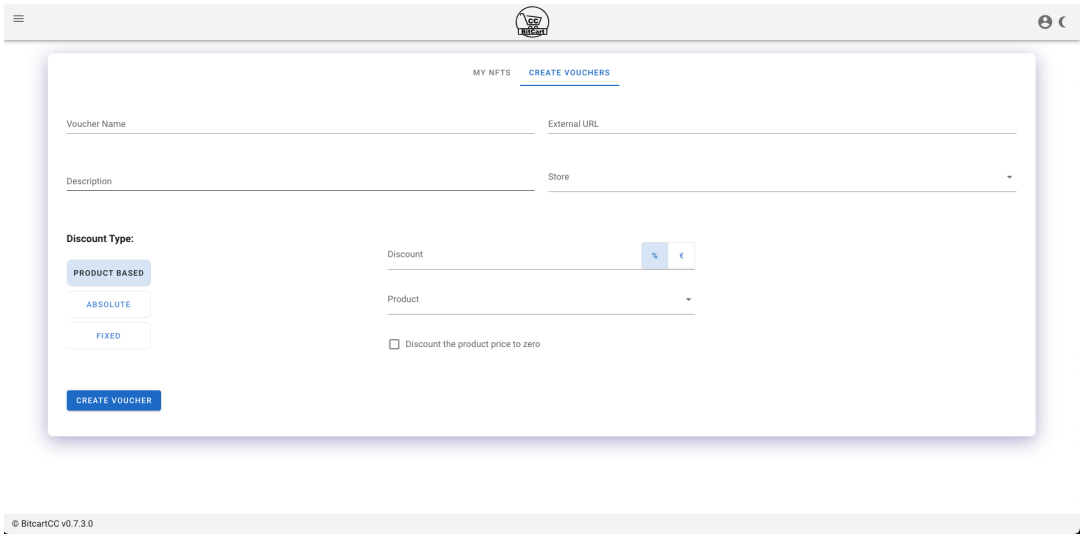


Figura 8.8: Página de criação de *vouchers*

No formulário de criação de *vouchers* personalizados, o comerciante pode selecionar a loja ou lojas para as quais pretende adicionar o cupão através de um menu *dropdown* denominado *Store*. Se o cupão for baseado em produtos, a lista de produtos de cada loja será obtida através de uma comunicação com a loja Shopify, caso esta esteja configurada, e carregada no menu *dropdown* - *Products*, permitindo ao comerciante escolher os produtos aos quais pretende aplicar o desconto. Se o cupão

for baseado em outros tipos de desconto, como descontos absolutos ou fixos, apenas será solicitado ao comerciante que indique o valor em percentagem ou monetário que pretende atribuir.

No fim de preencher todos os campos, é invocado o endpoint "/vouchers/create" que irá proceder à criação do voucher através da comunicação com o *smart contract*.

8.5.4 Ecrã de Pagamento

De modo a que o cliente pudesse usufruir dos seus descontos no ecrã de pagamento representado pela Figura 6.3, foi necessário implementar novas funcionalidades representadas pela Figura 8.9a de modo a que o requisito fosse cumprido.

As funcionalidades elaboradas foram as seguintes:

- **Aba Vouchers:** Foi adicionada uma nova aba à estrutura existente que permite ao cliente autenticar-se com a sua *crypto wallet* e, posteriormente, escolher o desconto a aplicar, como ilustrado na Figura 8.9b.

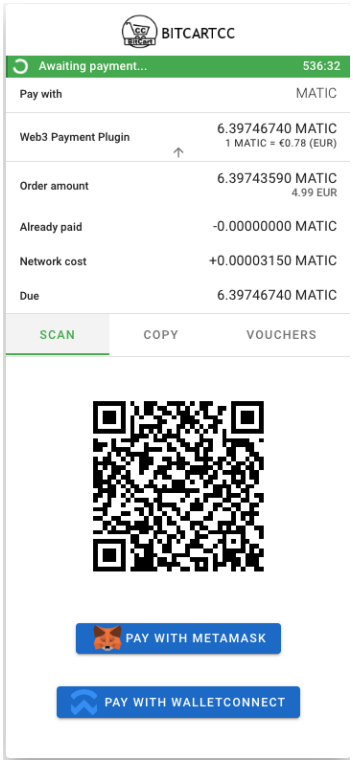
- **Criação de pagamentos parciais:** foi criado um algoritmo para diferenciar pagamentos completos e parciais em criptomoedas e NFTs, permitindo a submissão de *vouchers* como pagamento parcial. Quando o valor do voucher é inferior ao montante da fatura, é considerado um pagamento parcial e, após o pagamento do restante, a fatura é paga integralmente.

  Adicionalmente, quando o pagamento submetido é parcial, é apresentada uma confirmação visual na interface do cliente indicando que se trata de um pagamento parcial e que ainda existe um valor em aberto a ser pago. Isso ajuda a evitar confusão e erros por parte do utilizador, garantindo uma melhor experiência de utilização.

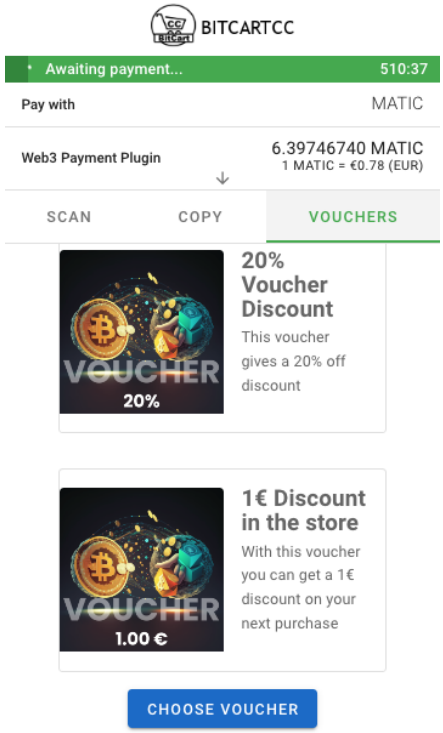
- **Fluxo de pagamentos:** Considerando a imutabilidade das transações na *Blockchain*, é crucial garantir a precisão e a intenção de cada transação realizada, uma vez que a sua reversão não é possível devido aos princípios fundamentais da tecnologia. Com o objetivo de estabelecer um processo seguro, foi elaborado um fluxo de pagamentos, ilustrado pelo diagrama da Figura 8.10 que descreve as etapas envolvidas na seleção de um cupão específico:

  1. Utilizador autentica-se com a sua carteira Metamask na aba Vouchers do ecrã de pagamento:

     1.1. Autenticação bem sucedida: apresenta a lista de *vouchers* que o cliente poderá utilizar no ato da compra;



(a) Novo ecrã de pagamento



(b) Ecrã de pagamento - aba Vouchers

1.2. Erro de autenticação: apresenta uma mensagem de erro e não será possível utilizar os *vouchers*;

2. O utilizador consegue selecionar um determinado *voucher* através do clique em "*Choose Voucher*", é aplicado ao montante final o desconto proveniente do *voucher* através de um pagamento parcial;

3. O cliente dispõe de duas opções: mudar de *voucher* ou prosseguir com o pagamento:

   3.1. Mudar de *voucher*: a lista de todos os cupões é novamente apresentada, onde este pode selecionar um novo cupão a utilizar;

   3.2. Prosseguir pagamento: nesta fase o cliente paga o montante em falta através da sua carteira criptográfica com criptomoeda escolhida e o cupão escolhido fica bloqueado;

4. É criado um novo pagamento parcial e a informação relativa à transação é guardada;

5. De seguida, o cliente poderá confirmar a utilização do *voucher* através do clique no botão "*Confirm*" na aba Vouchers que irá dar origem a uma interação com a carteira Metamask e com o *smart-contract* do *voucher*;

6. Após o sucesso da transação, a informação relativa à transação do *voucher* é guardada e o cliente obtém a conformação visual que a fatura se encontra totalmente paga.

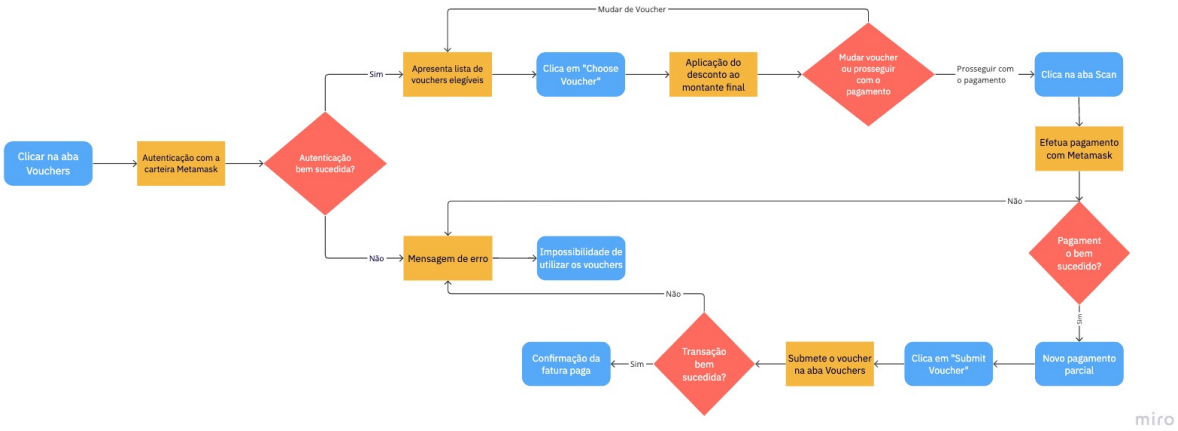


Figura 8.10: Fluxo de pagamentos

8.6 Loja de E-commerce Shopify

Nesta secção, será apresentada a integração do BitCartCC com a plataforma de e-commerce Shopify. Esta integração foi identificada como um dos requisitos necessários para permitir que os clientes pudessem efetuar compras diretamente na loja online, tornando o processo de compra mais fácil e conveniente facilitando o processo de adoção de criptomoedas.

Durante a implementação da integração, foram identificados alguns problemas que puseram em causa um dos casos de uso inicialmente recolhidos e foi necessário adaptar e encontrar uma solução para o problema que cumprisse com o objetivo e com o requisito pretendido. Na Subseção 8.6.1, serão descritos esses problemas em detalhe e as soluções encontradas para superá-los.

8.6.1 Desafios de Implementação

Com base na análise dos requisitos, foi identificado um caso de uso que visa proporcionar aos clientes a possibilidade de utilizar *vouchers* no formato de NFT e, em seguida, selecionar o método de pagamento de modo a pagar o valor remanescente da fatura. Essa funcionalidade permite que os clientes optem por utilizar métodos de pagamento convencionais já suportados pela loja ou realizar o pagamento utilizando criptomoedas.

Este caso de uso segue o exemplo ilustrado na Figura 8.9, referente à Figura 8.11, onde o cliente tem a possibilidade de autenticar-se utilizando sua carteira Metamask. Após a autenticação, serão apresentados ao cliente os cupões disponíveis

para utilização durante o processo de compra.

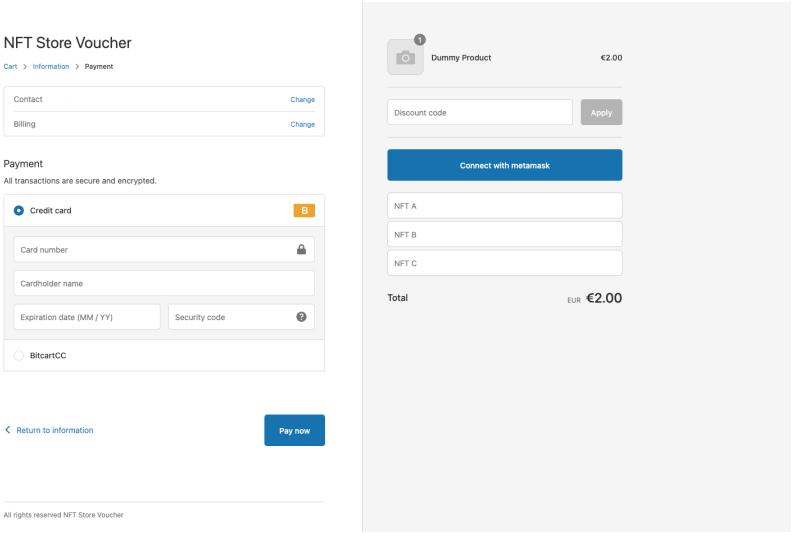


Figura 8.11: Caso de uso de utilização de *vouchers* dentro da ferramenta Shopify

No entanto, ao tentar personalizar o processo de pagamento, foram identificados certos desafios inerentes ao Shopify que limitavam essa personalização. Ou seja, para elaborar estas funcionalidades na página de *checkout* do Shopify, é necessário utilizar uma ferramenta do Shopify denominada "*Checkout UI Extensions*"[125] a qual permite aprimorar e introduzir novas funcionalidades personalizadas no processo de pagamento, proporcionando uma nova experiência aos clientes. Mas, essa ferramenta apresenta algumas limitações confirmadas pela equipa de desenvolvedores do Shopify, incluindo:

- Encontra-se apenas disponível para assinantes do plano ShopifyPlus que tem um custo para o comerciante bastante elevando (superior a 2000$/mês)[126];
- Por se tratar de uma página segura, o Shopify restringe as operações que os desenvolvedores podem realizar nessa página, criando um ambiente fechado conhecido como "*sandbox environment*" que não permite interações externas;
- Mesmo que fosse possível construir essa funcionalidade, o Shopify, por sua natureza, não permite a execução de pagamentos parciais.

Uma solução para esse problema foi a integração dos vouchers no método de pagamento com criptomoedas. Isso significa que, caso o cliente deseje utilizar os vouchers NFT, ele é obrigado a efetuar o pagamento utilizando criptomoedas. Como resultado, o caso de uso anterior, que permitia ao cliente escolher entre utilizar o voucher com os métodos de pagamento convencionais suportados pela loja, não é mais válido.

8.6.2 Aplicação Shopify

Com o propósito de melhorar o processo de criação e de gestão de vouchers por parte do comerciante foi desenvolvida uma aplicação dentro da plataforma de e-commerce Shopify. Através das ferramentas fornecidas pela plataforma [127] esta aplicação permite ao comerciante criar os seus próprios descontos dentro do seu painel de administrador da sua loja Shopify.

Esta solução foi criada com o intuito de substituir o caso de uso anteriormente identificado uma vez que ficou invalidado, garantindo deste modo a manutenção da complexidade do projeto.

Através da Figura 8.12 é possível visualizar o aspeto da aplicação que se encontra embutida dentro do painel de administrador da loja do comerciante. A sua construção teve por base as ferramentas disponibilizadas pelo Shopify incluindo a utilização da biblioteca *Shopify Polaris Components* [128] que nos permite integrar as nossas funcionalidades de forma consistente com as diretrizes de design da plataforma. Isso resultou numa aplicação que se alinha perfeitamente ao design nativo do Shopify, proporcionando uma *user experience* coerente e familiar para o comerciante.

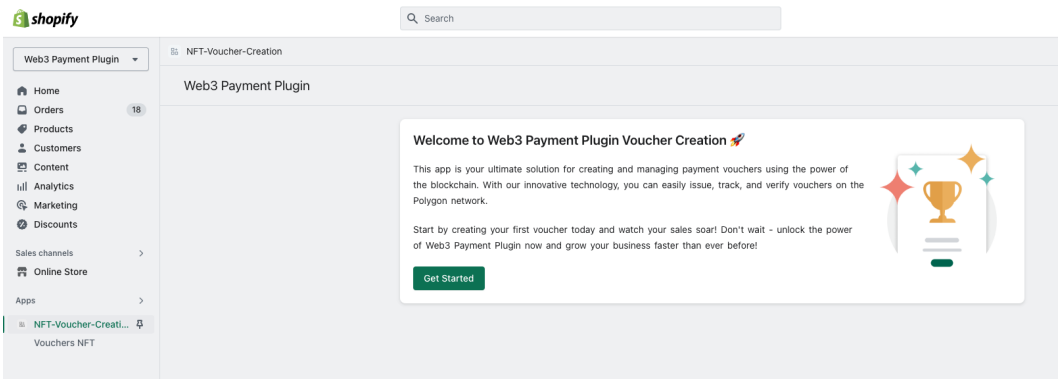
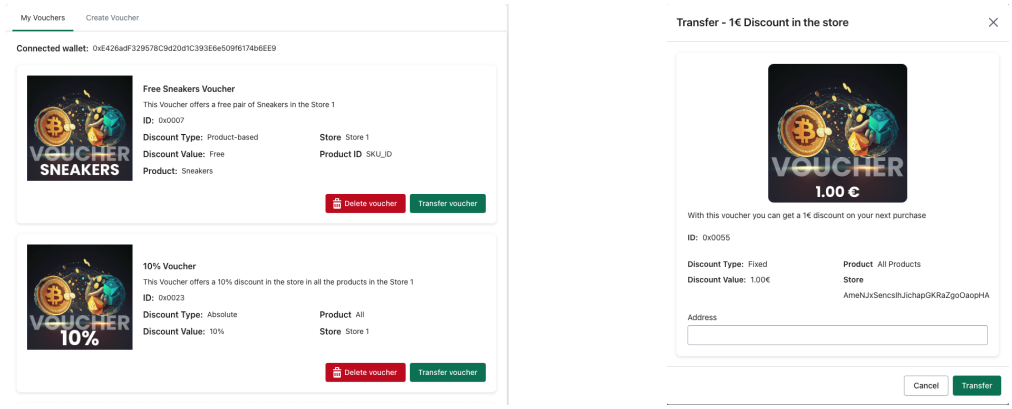


Figura 8.12: Ecrã inicial da aplicação Shopify

Ainda dentro da aplicação Shopify na aba "My Vouchers", como é possível verificar pela Figura 8.13 o comerciante de forma similar tem a opção de gerir e visualizar informação sobre os seus vouchers, incluindo a possibilidade de eliminar ou transferir um determinado voucher para um endereço em específico.



(a) Ecrã de listar os vouchers da aplicação Shopify

(b) Ecrã de transferir os vouchers na aplicação Shopify

Figura 8.13: Páginas da aplicação Shopify

De forma similar à funcionalidade descrita na Subseção 8.5.3, a Figura 8.14 ilustra o processo de criação de vouchers, seguindo o padrão de metadados previamente definido. Esta funcionalidade permite que o comerciante crie os seus próprios vouchers sem a necessidade de sair do seu painel de administração da loja.

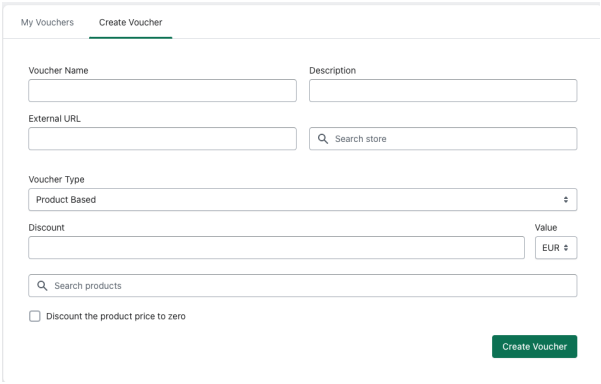


Figura 8.14: Ecrã de criação de vouchers na aplicação Shopify

8.7 Integração em Website Personalizado

Através da Tabela 5.1, é possível identificar um requisito que consiste na integração da aplicação num website genérico e, como tal foi construído um website *storefront* denominado "*Cryptoccino*", como é possível visualizar pela Figura 8.15 utilizando as tecnologias Vite [129], TypeScript [130], React Redux [131] e a *framework* TailwindCSS [132].

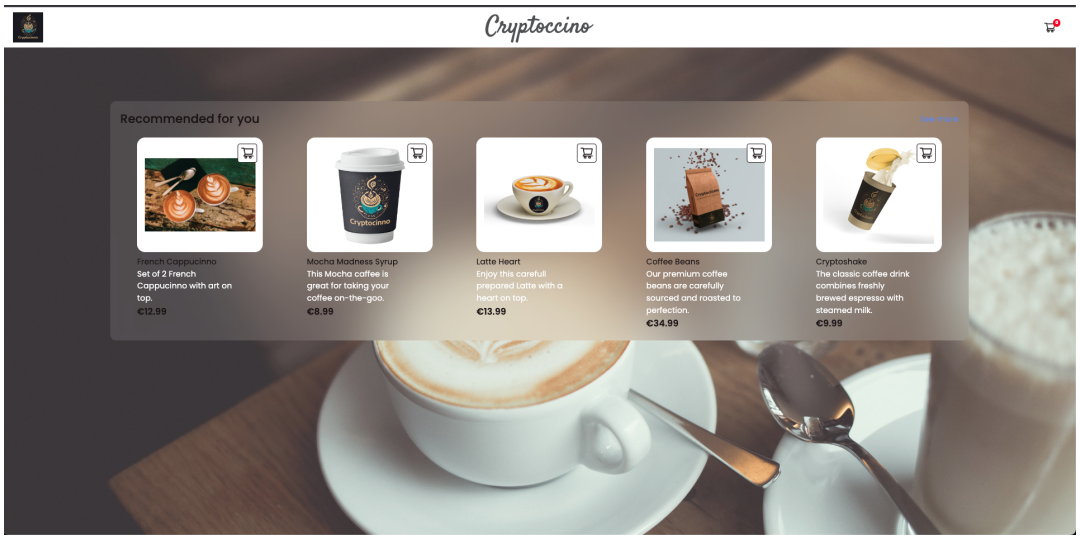


Figura 8.15: Ecrã principal do website Cryptoccino

Este website é constituído pela página principal onde o utilizador consegue visualizar os artigos disponíveis na loja e pelo ecrã de pagamento como é possível visualizar pela Figura 8.16 onde o cliente tem informação sobre os itens presentes no seu carrinho e o valor total a pagar. Ainda no ecrã de pagamento, quando o utilizador pressiona o botão "*Confirm*" é apresentada a interface de pagamento que foi integrada neste website apresentando assim outra forma de integração mais personalizada para o comerciante similar à interface representada pela Figura 8.9a.

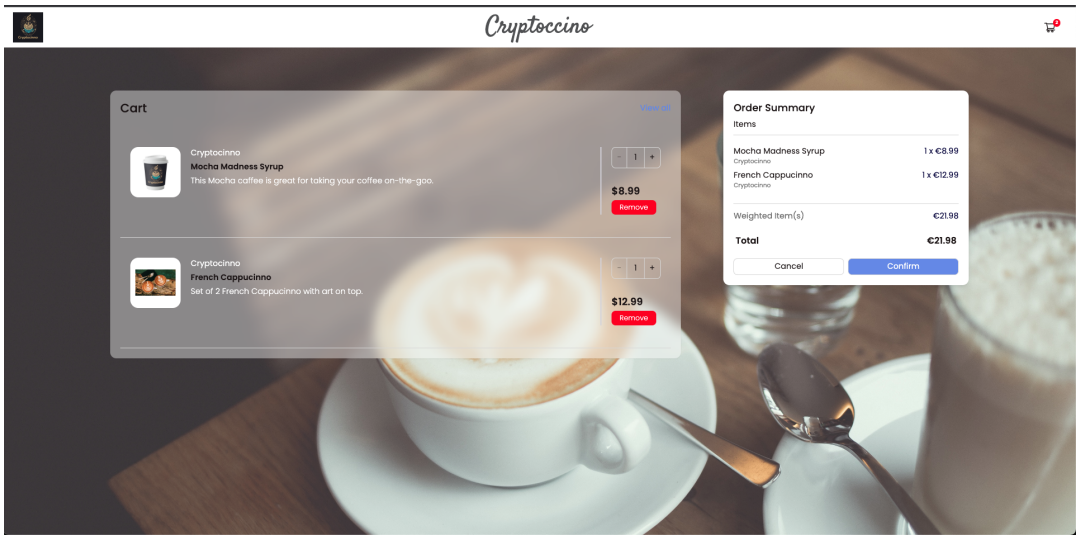


Figura 8.16: Ecrã de pagamento no website Cryptoccino

O objetivo deste caso de uso era fornecer uma integração personalizada, para além da plataforma de e-commerce, através da construção de um website. Neste contexto, o website atua apenas como uma loja online e não inclui uma camada de

backend responsável por executar a lógica da loja, como a pesquisa de produtos ou o cálculo dos valores do carrinho. Esta abordagem foi adotada considerando que o processamento de dados não era uma parte essencial para a realização do caso de uso em questão. Assim, durante a criação deste website, não se deu prioridade à implementação dessa arquitetura.

8.8 Integração *Blockchain Loyalty Platform*

Durante o segundo semestre, por meio das apresentações realizadas na empresa, surgiu a proposta de integrar o estágio "Blockchain Loyalty Platform" [133], conduzido pelo estagiário Nuno Silva, com o sistema de plugin de pagamentos Web3. O objetivo desse estágio é criar um programa de fidelidade baseado na *Blockchain* para comerciantes, permitindo oferecer benefícios aos seus clientes. Através do *feedback* recebido nas apresentações, surgiu a sugestão de integrar os dois estágios, permitindo que os descontos gerados pelo programa de fidelidade pudessem ser utilizados como *vouchers* no plugin de pagamentos Web3.

Essa integração visa proporcionar uma experiência mais abrangente e vantajosa para os clientes finais, potencializando os benefícios oferecidos pelo programa de fidelidade por meio da utilização dos *vouchers* no processo de pagamento, adicionando deste modo uma nova aplicação aos *vouchers* NFT e ampliando as possibilidades do estágio.

Com o intuito de viabilizar essa integração, foram compartilhados com o estagiário Nuno Silva os detalhes de implementação da estrutura de metadados seguidos pelo plugin de pagamentos web3. Dessa forma, ao criar os descontos no programa de fidelidade, era necessário seguir a estrutura definida, permitindo que os *vouchers* fossem utilizados nas lojas correspondentes. Além disso, foi disponibilizado o *smart contract* utilizado para a criação dos *vouchers*, garantindo que eles fossem criados no contexto adequado.

8.9 Integração com a plataforma de recompensas interna

Na fase de desenvolvimento, foi proposta uma nova integração com um produto que está em desenvolvimento na empresa. No entanto, como se trata de um produto de desenvolvimento interno e não pertencendo ao estágio, encontra-se abrangido pelo acordo de confidencialidade logo, não serão revelados nomes nem detalhes específicos

sobre o modo de funcionamento do produto em questão.

No entanto de modo a cumprir com o caso de uso US-7 foram realizadas tarefas específicas que através do NFT e dos seus atributos este pudesse ser utilizado na plataforma de e-commerce Shopify como troca por bens físicos.

Capítulo 9

Testes e validações

Os testes desempenham um papel crítico na garantia da qualidade do software, assegurando que o sistema funciona conforme o esperado e cumpre com os requisitos estabelecidos. Eles desempenham um papel crucial na relevação de erros e falhas, permitindo que sejam identificados e corrigidos antes da entrega do produto final [134] melhorando deste modo a confiabilidade no software [135].

Este capítulo tem como objetivo descrever os testes realizados de modo avaliar a qualidade e conformidade dos requisitos funcionais e não funcionais estabelecidos no Capítulo 5. A primeira secção aborda os testes unitários aplicados à API da aplicação, fornecendo detalhes sobre sua implementação e resultados obtidos.

Em seguida, serão apresentados os testes realizados para avaliar os requisitos não funcionais, destacando as principais conclusões e observações derivadas dessas avaliações.

9.1 Testes Unitários

Os testes unitários correspondem a uma prática fundamental que visam validar componentes de uma aplicação de forma isolada. Foram realizados os seguintes testes à *Merchants API*:

- Manuais: Para este tipo de teste foi considerado a interação com a aplicação de modo a verificar se através da exploração das funcionalidades dentro da aplicação surgia algum problema;
- Automatizados: Através da biblioteca PyTest [136], foram elaborados testes específicos aos endpoints da *Merchants API* de modo a verificar o comportamento da API em determinados casos específicos com o objetivo de garantir o

bom funcionamento mesmo na presença de potenciais erros.

Uma vez que o software se encontra *deployed* num *container* Docker [137] este foi também alojado num servidor da WIT que possui as seguintes especificações:

- **Sistema Operativo:** CentOS Linux 8
- **Número de CPUs:** 2
- **Total memória RAM:** 8 GB
- **Memória do servidor:** 20GB

Para a realização dos testes foi utilizada a ferramenta Postman [138] onde cada teste é representado pela seguinte estrutura:

- **ID:** ID do teste
- **Descrição:** Corresponde ao propósito do teste
- **Resultado esperado:** Valor que o teste deverá retornar
- **Resultado:** Corresponde ao valor do teste: Passou (**P**) ou Falhou (**F**)

Através da Tabela 9.1, é possível identificar os diversos testes unitários desenvolvidos de modo a validar os diferentes métodos da *Merchants* API criados. Estes testes visaram comunicar com a API em diferentes cenários de utilização de modo a observar o seu comportamento diante de situações inesperadas.

Método	**ID**	**Descrição**	**Resultado Esperado**	**Resultado (P/F)**
getNFTUser	1	Execução em condições normais com os parâmetros corretos	200 OK	**P**
	2	Execução com o parâmetro *chainID* inválido	422 *Unprocessable Entity*	**P**
	3	Execução com o parâmetro *address* inválido	422 *Unprocessable Entity*	**P**

getNFTCheckout	4	Execução em condições normais com os parâmetros corretos	200 OK	**P**
	5	Execução com parâmetros inválidos (fabricados)	422 *Unprocessable Entity*	**P**
getTokensABI	6	Execução em condições normais com os parâmetros corretos	200 OK	**P**
submitVoucher	7	Execução em condições normais com os parâmetros corretos	200 OK	**P**
	8	Execução com o parâmetro voucherID inválido	400 *Bad Request*	**P**
	9	Execução com o parâmetro invoiceID inválido	404 *Not Found*	**P**
	10	Execução com o parâmetro paymentID inválido	404 *Not Found*	**P**
createVoucher	11	Execução em condições normais com os parâmetros corretos	200 OK	**P**
	12	Execução com o parâmetro voucherType inválido	404 *Not Found*	**P**
getShopifyProducts	13	Execução em condições normais com os parâmetros corretos	200 OK	**P**
	14	Execução com o parâmetro storeID inválido (não existente)	400 *Bad Request*	**P**
	15	Execução com uma storeID sem permissões do Shopify	403 *Forbidden*	**P**
	16	Execução com uma storeID sem permissões do Shopify	400 *Bad Request*	**P**
getNFTCreated	17	Execução em condições normais com os parâmetros corretos	200 OK	**P**
	18	Execução com o parâmetro cid inválido não existente	422 *Unprocessable Entity*	**P**

getNFTCreated	19	Execução em condições normais com os parâmetros corretos	200 OK	**P**
	20	Execução sem autenticação	401 *Unauthorized*	**P**
getTokenID	21	Execução em condições normais com os parâmetros corretos	200 OK	**P**
	22	Execução com o parâmetro tokenID inválido	400 *Bad Request*	**P**
	23	Execução com o parâmetro tokenID inexistente	422 *Unprocessable Entity*	**P**

Tabela 9.1: Testes unitários à Merchants API

Com base nos resultados da Tabela 9.1, é possível observar que todos os testes realizados foram bem sucedidos garantindo assim o bom funcionamento da API em diversas situações inusitadas.

9.2 Requisitos Não Funcionais

Esta secção será responsável por identificar os testes realizados aos requisitos não funcionais recolhidos durante o 1º semestre presentes na Seção 5.4.

9.2.1 Desempenho

No processo de levantamento dos requisitos não funcionais, presentes no Capítulo 5, foram levantados dois cenários para o requisito não funcional de desempenho: um envolvendo uma comunicação com a *Blockchain* e outro focado na interação e comunicação com a API desenvolvida.

No entanto para o cenário 1, a realização de testes de desempenho em sistemas que envolvem a comunicação com a *Blockchain* apresenta desafios específicos. A natureza descentralizada da *Blockchain* implica fatores como a latência da rede, a taxa de transação paga, o tráfego da rede entre outros. A obtenção de métricas precisas e confiáveis, como tempos de resposta e taxa de transações, pode ser uma tarefa desafiadora devido às características intrínsecas da *Blockchain*, pelo apenas

foram identificados os tempos médios de transação de um NFT, correspondendo a um valor de 13.3 segundos.

No entanto como foi referido anteriormente, este valor é altamente variável e não garante que seja sempre constante uma vez que varia tendo em conta fatores externos que não são possíveis de controlar e/ou evitar.

Para o cenário 2 que envolve a comunicação com a API desenvolvida, foram também elaborados testes de performance recorrendo à ferramenta Postman que nos permite realizar testes de performance estipulando parâmetros como:

- **Utilizadores Virtuais**: corresponde ao número clientes que irão realizar execuções em paralelo e repetidamente durante o período estipulado;
- **Duração do teste**: tempo máximo de execução do teste;
- **Perfil de carga**: determina como o número de utilizadores varia durante o teste. Este pode ser **fixo** - o número de utilizadores é constante durante o período de teste, ou **incremental** - durante este período, o número de utilizadores virtuais aumenta gradualmente para o número estipulado.

Para a realização dos testes foram simulados 10 utilizadores virtuais, com um tempo de teste de 2 minutos ao qual os utilizadores foram sendo adicionados de forma incremental, ou seja o teste começou com 1 cliente e este foi aumentando até 10 com o decorrer do teste.

Através da Figura 9.1 é possível visualizar o gráfico que contém informação acerca dos pedidos realizados e do seu tempo de resposta à medida que o número de utilizadores em simultâneo aumenta.

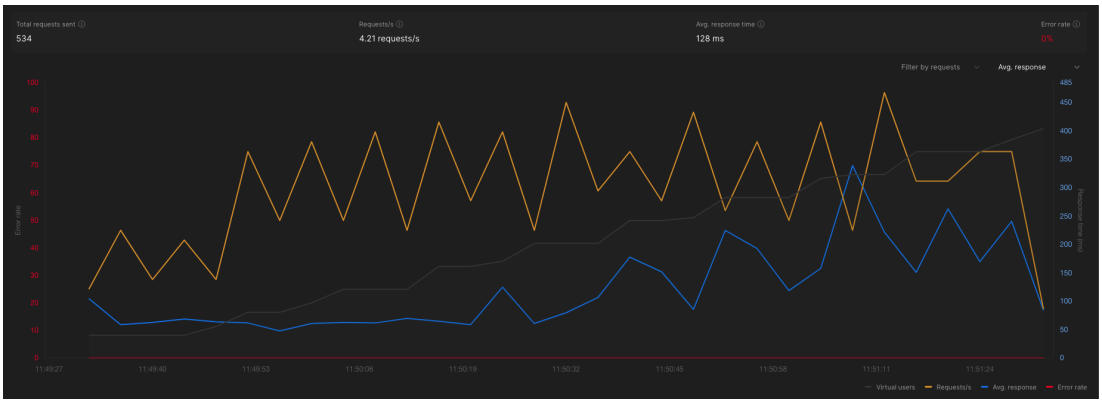


Figura 9.1: Teste de desempenho ao *endpoint* "getStores"

Este teste destina-se apenas à comunicação com a *Merchants* API onde é feito um simples pedido à base de dados de modo a devolver uma lista de objetos. Também

na Tabela 9.2 são enumeradas mais métricas de modo a verificar o comportamento deste teste em específico.

Tabela 9.2: Cenário 2: Comunicação com Merchants API (*getStores*)

Cenário 2a - Comunicação com Merchants API (*getStores*)	
Número total pedidos realizados	534
Número de pedidos por segundo	4.31/s
Tempo média de resposta (ms)	128 ms
Tempo mínimo de resposta (ms)	30 ms
Tempo máximo de resposta (ms)	1856 ms
Taxa de erro	0%

Através da análise da Tabela 9.2 é possível verificar que todos os 534 pedidos efetuados ao longo do teste foram bem sucedidos através da análise da taxa de erro. O tempo médio de resposta foi de 128 milissegundos, indicando uma resposta relativamente rápida por parte da API. No entanto, é importante notar que houve variações significativas nos tempos de resposta, com um tempo mínimo de 30 milissegundos e um tempo máximo de 1856 milissegundos. Mas, através da análise da Figura 9.1 podemos verificar que essa oscilação em média não é significativa podendo ter ocorrido através de um pico no servidor, latência da rede, entre outros fatores.

Foi ainda explorado outro caso de uso que consistia na análise do desempenho quando era feito um pedido à Merchants API que comunicava com uma API externa, como por exemplo com o Shopify. Este é um caso que ocorre no nosso sistema quando, por exemplo, pretendemos obter a lista de produtos que uma loja tem na sua plataforma Shopify.

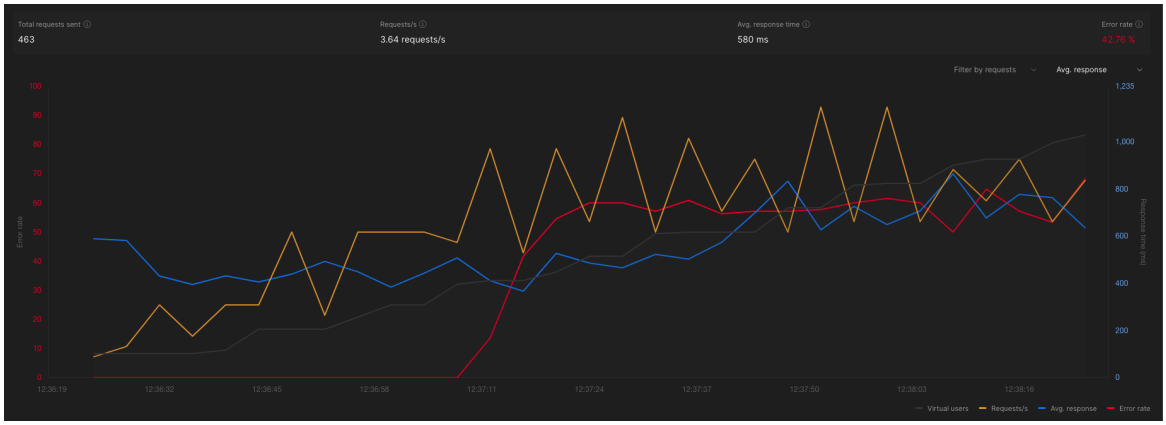


Figura 9.2: Teste de desempenho ao *endpoint* "getShopifyItems"

A Figura 9.2 demonstra os resultados obtidos para o teste do endpoint "getShopifyI-

tems" que comunica tanto com a MerchantsAPI como a Shopify API. No entanto com a ajuda da Tabela 9.3 é possível verificar que este pedido comparado ao teste anterior, apresenta tempos de resposta médios bastante superiores (cerca de 4.5 vezes) e, à medida que o número de utilizadores em simultâneo aumenta, a taxa de erro também aumenta.

Ao analisar o pedido feito à API que se comunica com a API do Shopify, é importante considerar que estamos a consultar uma fonte externa de dados. Nesse contexto, é normal que o tempo de resposta do pedido seja superior em comparação com um pedido que não envolve a comunicação com uma API externa.

O mesmo se aplica à taxa de erro, ou seja, o Shopify de modo a proteger os seus serviços e infraestrutura tem um mecanismo de limite de consulta à sua API, o qual varia conforme o tipo de plano utilizado. Utilizadores com o plano básico, por exemplo, estão sujeitos a um limite de 2 pedidos por segundo [139]. Quando esse limite é excedido, o Shopify descarta os pedidos, resultando num aumento na taxa de erro dos pedidos.

Tabela 9.3: Cenário 2: Comunicação com Merchants API (*getShopifyItems*)

Cenário 2a - Comunicação com Merchants API (*getShopifyItems*)	
Número total pedidos realizados	463
Número de pedidos por segundo	3.64/s
Tempo média de resposta (ms)	580 ms
Tempo mínimo de resposta (ms)	313 ms
Tempo máximo de resposta (ms)	1986 ms
Taxa de erro	42.76%

Após analisar todos os resultados das tabelas mencionadas anteriormente, constatamos que eles foram extremamente satisfatórios, pois atendem ao valor estabelecido de ser inferior a 1 segundo. Sob circunstâncias normais, o requisito não funcional estabelecido é cumprido de forma consistente.

9.2.2 Segurança

No âmbito deste projeto, foram tomadas medidas específicas para garantir a segurança das informações e do sistema. Essas medidas incluem:

- Autenticação com API Key: a aplicação adota o padrão de autenticação com JSON Web Tokens (JWT). Isso significa que cada solicitação à API deve conter uma chave de autenticação válida, o que ajuda a controlar o acesso aos recursos da aplicação e evita acessos não autorizados;

- Encriptação de dados sensíveis: de modo a proteger dados sensíveis, como palavras passe armazenadas na base de dados, estes dados são encriptados de modo a garantir que não fiquem comprometidos ou utilizados indevidamente;
- Protocolo SSL: o sistema utiliza o protocolo Secure Sockets Layer (SSL), garantindo uma comunicação segura por meio de HTTPS. Essa camada adicional de segurança permite a encriptação das informações transmitidas entre o cliente e o servidor, mitigando a possibilidade de interseção e manipulação dos dados durante a transmissão;
- Integração com a *Blockchain*: a utilização da *Blockchain* traz um alto nível de segurança. No caso deste projeto, apenas o endereço público da carteira é necessário para interagir com a *Blockchain*. Isso significa que informações sensíveis, como chaves privadas, não são guardadas ou expostas na aplicação, reduzindo significativamente os riscos de roubo ou acesso não autorizado.

Essas medidas são essenciais para garantir a confidencialidade, a integridade e a disponibilidade das informações, bem como a proteção contra ameaças externas. Ao adotar essas práticas de segurança, o sistema proporciona um ambiente seguro e confiável para os seus utilizadores, promovendo a confiança e a tranquilidade no uso da aplicação.

9.3 Considerações sobre os resultados obtidos

Os testes unitários desempenharam um papel importantíssimo na fase final do desenvolvimento, através do teste de todas as funcionalidades principais da API e validadas com sucesso. Além disso, os testes aos requisitos não funcionais revelaram que a aplicação atendeu aos critérios de desempenho, segurança e escalabilidade estabelecidos. Foi também importante para corrigir problemas que haviam tanto na *backend* no processamento de informação como na *frontend* na visualização de dados.

Com base nos resultados dos testes, recomenda-se a implementação de uma estratégia de *caching* para consultas frequentes e melhoramentos no design da API para garantir uma escalabilidade adequada em cenários de alta demanda.

Capítulo 10

Conclusão

Este capítulo é responsável pela apresentação das reflexões finais do estágio ao longo de toda a dissertação incluindo os desafios encontrados e as aprendizagens realizadas.

Por fim, é enunciado o trabalho futuro que poderá ser implementado de modo a acrescentar valor e aumentar as funcionalidades da aplicação com o propósito a que sejam atendidas mais necessidades e exploradas novas situações.

10.1 Reflexões finais

O projeto apresentado pela WIT Software consiste no desenvolvimento de um *plugin* de pagamentos Web3. O objetivo principal é desenvolver um meio de pagamentos para plataformas de *e-commerce*, permitindo que os clientes realizem transações utilizando *tokens* ERC20 e aproveitem *vouchers* na forma de NFT. Durante o estágio, foram executadas diversas etapas cruciais, incluindo uma análise detalhada da tecnologia *Blockchain*, pesquisa de soluções existentes no mercado e estudo aprofundado das plataformas de *e-commerce* para a implementação do *plugin*. A análise desse processo permitiu escolher a ferramenta BitCartCC como ponto de partida para o desenvolvimento.

A metodologia utilizada que corresponde a uma variação do Scrum, foi adotada para garantir a organização e o planeamento adequados do projeto. Ferramentas como GitLab e Jira foram utilizadas para apoiar a equipa e o estagiário no acompanhamento e controlo do desenvolvimento e principalmente que o desenvolvimento acontecia de acordo com os tempos delineados.

A fase de desenvolvimento foi crucial, com a implementação dos recursos planeados. No entanto, antes do início da fase de desenvolvimento houve a necessidade de avaliar os requisitos, tarefas e casos de uso de modo a que o planeamento das tarefas

fosse o mais direto possível. Durante esta fase, foram realizadas várias apresentações na empresa, envolvendo os demais estagiários e colaboradores. Essas apresentações proporcionaram um *feedback* importante sobre o trabalho realizado, permitindo-nos identificar áreas de melhoria tanto no projeto como nas apresentação. Essas experiências contribuíram para aprimorar o nosso trabalho e preparar melhores apresentações no futuro.

Finalmente os testes desempenharam um papel importante no estágio, permitindo a identificação e correção de erros nos principais componentes da aplicação.

Em consideração ao objetivo do estágio, questiona-se se a solução desenvolvida é plausível para os utilizadores-alvo. Através da análise dos critérios de sucesso definidos no Capítulo 4 podemos considerar que o estágio foi um sucesso. Deste a realização de todos os requisitos funcionais atribuídos como *Must* à adição de novas funcionalidades passando pela incorporação de novos casos de uso da aplicação podemos considerar que este tipo de pagamento pode ser bastante utilizado para plataformas de *e-commerce*.

No entanto um dos grandes problemas encontrados inicialmente foi o facto do ponto de partida da aplicação corresponder a uma solução *open-source* o que levou à exaustiva análise das funcionalidades já suportadas e do seu modo de funcionamento de modo a conseguir integrar os requisitos pretendidos. Outro desafio já mencionado foi o desconhecimento ao tema o que levou a uma fase de pesquisa e aprendizagem que também contribuíram para o sucesso do estágio e do estagiário como profissional através do desenvolvimento de um conjunto de competências e capacidades sobre as práticas, técnicas e tecnologias utilizadas pela empresa e pela indústria no mercado de trabalho.

10.2 Trabalho futuro

Este documento é responsável por retratar o processo desde a introdução ao problema, passando pela fase de recolha de requisitos e acabando na fase de desenvolvimento onde são retratadas todas as funcionalidades elaboradas que permitiram concluir todos os requisitos identificados. No entanto, neste capítulo, são exploradas algumas direções promissoras para o desenvolvimento e melhoramento contínuo da aplicação de vouchers sob a forma de NFT.

Uma das principais tarefas a realizar no futuro é a adição do *standard* ERC-1155 na criação e gestão dos vouchers, uma vez que este *standard* trás benefícios na gestão dos *smart-contracts* bem como na criação de vários *tokens* com diferentes atributos o que pode ser relevante no processo de adição de novos tipos de descontos

à ferramenta.

Outra funcionalidade corresponde a melhorias na experiência do utilizador onde o processo de *checkout* seria melhorado de modo a que o cliente de forma natural conseguisse usufruir dos seus vouchers sem qualquer tipo de problema.

Através destas novas funcionalidades e melhorias poderíamos expandir a funcionalidade da aplicação e introduzir novos casos de uso e aplicações para a ferramenta de pagamentos em plataformas de *e-commerce*.

Referências

[1] Daily Hodl Staff. Crypto winter unlikely as 'astonishing' user growth dwarfs internet adoption rate: Macro guru raoul pal, 2022. URL dailyhodl.com/2022/05/04/crypto-winter-unlikely-as-astonishing-user-growth-dwarfs-internet-adoption-rate-macro-guru-raoul-pal/. Consultado a 23 de dezembro de 2022.

[2] Jordan Tuwiner. 63+ cryptocurrency statistics, facts & trends, 2022. URL https://buybitcoinworldwide.com/cryptocurrency-statistics/. Consultado a 18 de novembro de 2022.

[3] Tesla. Dogecoin | tesla support, 2022. URL https://www.tesla.com/support/dogecoin. Consultado a 23 de dezembro de 2022.

[4] The Block. Nft overview, 2022. URL https://www.theblock.co/data/nft-non-fungible-tokens/nft-overview. Consultado a 29 de novembro de 2022.

[5] SOPHIE HARES. These blockchain experts are transforming loyalty, one nft at a time, 2022. URL https://www.mastercard.com/news/perspectives/2022/uptop-startup-story/. Consultado a 7 de janeiro de 2023.

[6] IBISWorld. Percentage of business conducted online, 2022. URL https://www.ibisworld.com/us/bed/percentage-of-business-conducted-online/88090/. Consultado a 10 de janeiro de 2023.

[7] DataReportal. Digital around the world, 2022. URL https://datareportal.com/global-digital-overview. Consultado a 10 de janeiro de 2023.

[8] Pragati Verma. Evolution of web, 2021. URL https://dev.to/pragativerma18/evolution-of-web-42eh. Consultado a 9 de novembro de 2022.

[9] Ethereum.org. What is web3?, 2022. URL https://ethereum.org/en/web3/. Consultado a 26 de setembro de 2022.

[10] Ethereum.org. Introduction to web3, 2023. URL https://ethereum.org/en/web3/. Consultado a 21 de junho de 2023.

[11] David Baily. Can web 3.0 solve the problems of web 2.0?, 2022. URL `https://supplain.io/news/web3-solve-web2-problems`. Consultado a 21 de junho de 2023.

[12] Unstoppable Domains. The story of web3 - reimagining the web, 2021. URL `https://unstoppabledomains.com/blog/categories/web3-101/article/the-story-of-web3`. Consultado a 21 de junho de 2023.

[13] Vitalii Shevchuk. Top web3 architecture layers explained: Frontend, backend, and data, 2022. URL `https://itnext.io/top-3-web-3-0-architecture-layers-explained-frontend-backend-and-data-e10200f7fc76`. Consultado a 26 de setembro de 2022.

[14] ADAM HAYES. Blockchain facts: What is it, how it works, and how it can be used, 2022. URL `https://www.investopedia.com/terms/b/blockchain.asp`. Consultado a 22 de setembro de 2022.

[15] River Finantial. Proof-of-work (pow) vs proof-of-stake (pos), 2022. URL `https://river.com/learn/proof-of-work-pow-vs-pos-proof-of-stake/`. Consultado a 3 de outubro de 2022.

[16] CFI Team. Double-spending, 2022. URL `https://corporatefinanceinstitute.com/resources/cryptocurrency/double-spending/`. Consultado a 1 de outubro de 2022.

[17] Cryptopedia by Gemini. What are proof of stake and delegated proof of stake?, 2021. URL `https://www.gemini.com/cryptopedia/proof-of-stake-delegated-pos-dpos`. Consultado a 21 de junho de 2023.

[18] Jake Frankenfield. What are smart contracts on the blockchain and how they work, 2022. URL `https://www.investopedia.com/terms/s/smart-contracts.asp`. Consultado a 24 de setembro de 2022.

[19] Ashish Sinha | Vanessa Purwin | Mohamed Damak | Erkan Erturk. Smart contracts could improve efficiency and transparency in financial transaction, 2022. URL `https://www.spglobal.com/en/research-insights/featured/special-editorial/smart-contracts-could-improve-efficiency-and-transparency-in-financial-transactions`. Consultado a 24 de setembro de 2022.

[20] Kirsty Moreland | Ledger. The blockchain generations, 2023. URL `https://www.ledger.com/academy/blockchain/web-3-the-three-blockchain-generations`. Consultado a 21 de junho de 2023.

[21] The Nation. Evolution of blockchain technology, 2022. URL https://www.nationthailand.com/business/corporate/40021641. Consultado a 21 de junho de 2023.

[22] Ethereum.org. Erc-20 token standard, 2022. URL https://ethereum.org/en/developers/docs/standards/tokens/erc-20/. Consultado a 25 de setembro de 2022.

[23] Mahmuda Akter Isha. 7 core benefits of erc20 tokens in the business world, 2022. URL https://www.bdtask.com/blog/benefits-of-erc20-tokens. Consultado a 26 de setembro de 2022.

[24] ERC-721. What is erc-721, 2022. URL https://erc721.org. Consultado a 26 de setembro de 2022.

[25] Rakesh Sharma. Non-fungible token (nft): What it means and how it works, 2022. URL https://www.investopedia.com/non-fungible-tokens-nft-5115211. Consultado a 23 de setembro de 2022.

[26] CryptoStars. Exploring nfts' use-cases [part 1], 2022. URL https://blog.cryptostars.is/exploring-nfts-use-cases-part-1-74b015c56d0f. Consultado a 21 de junho de 2023.

[27] Griffin Mcshane. What are utility nfts?, 2022. URL https://www.coindesk.com/web3/2022/11/16/what-are-utility-nfts/. Consultado a 26 de setembro de 2022.

[28] CHRISTIE'S. Beeple's opus, 2021. URL https://www.christies.com/features/Monumental-collage-by-Beeple-is-first-purely-digital-artwork-NFT-to-come-to-auction-11510-7.aspx. Consultado a 9 de janeiro de 2023.

[29] Ethereum.org. Erc-1155 multi-token standard | ethereum.org, 2022. URL https://ethereum.org/en/developers/docs/standards/tokens/erc-1155/. Consultado a 23 de setembro de 2022.

[30] Circle.com. Usdc - crypto that's held to a higher standard, 2022. URL https://www.circle.com/en/usdc. Consultado a 3 de janeiro de 2023.

[31] CryptoKitties. Cryptokitties | collect and breed furrever friends, 2022. URL https://www.cryptokitties.co/. Consultado a 9 de novembro de 2022.

[32] moreReese. What is erc-1155? ethereum's flexible token standard, 2022. URL https://decrypt.co/resources/what-is-erc-1155-ethereums-flexible-token-standard. Consultado a 23 de setembro de 2022.

[33] Georgia Weston. Ethereum tokens: Erc1155 tutorial, 2022. URL `https://101blockchains.com/erc1155-tutorial/`. Consultado a 21 de junho de 2023.

[34] Ledger. Erc-1155 meaning, 2022. URL `https://www.ledger.com/academy/glossary/erc-1155`. Consultado a 21 de junho de 2023.

[35] Third Web. What is erc-1155? the ethereum semi-fungible token standard, 2022. URL `https://blog.thirdweb.com/what-is-erc-1155-nft/`. Consultado a 21 de junho de 2023.

[36] Jake Frankenfield. Cryptocurrency wallet: What it is, how it works, types, security, 2022. URL `https://www.investopedia.com/terms/b/bitcoin-wallet.asp`. Consultado a 10 de novembro de 2022.

[37] Cryptopedia Staff. Custodial vs. non-custodial wallets, 2021. URL `https://www.gemini.com/cryptopedia/crypto-wallets-custodial-vs-noncustodial`. Consultado a 10 de novembro de 2022.

[38] Sepior. Shared-custody wallet application, 2022. URL `https://sepior.com/solutions/shared-custody-wallet`. Consultado a 21 de junho de 2023.

[39] BitPanda. What are multi-signature wallets and how do they work?, 2022. URL `https://www.bitpanda.com/academy/en/lessons/what-are-multi-signature-wallets-and-how-do-they-work/`. Consultado a 21 de junho de 2023.

[40] Shopify. What is ecommerce?, 2022. URL `https://www.shopify.com/encyclopedia/what-is-ecommerce`. Consultado a 30 de novembro de 2022.

[41] The Ecomm Manager Team. What is an ecommerce platform? types and features, 2022. URL `https://theecommmanager.com/general/what-is-ecommerce-platform/`. Consultado a 30 de novembro de 2022.

[42] Open Source Initiative. Licenses & standards, 2022. URL `https://opensource.org/licenses`. Consultado a 30 de novembro de 2022.

[43] Pipecandy. ecommerce platform market share 2022, 2022. URL `https://pipecandy.com/ecommerce-market-share/platforms`. Consultado a 26 de setembro de 2022.

[44] BigCommerce. 15 best ecommerce platforms to consider, 2022. URL `https://www.bigcommerce.com/articles/ecommerce/ecommerce-platforms/#-best-ecommerce-platforms-to-consider`. Consultado a 29 de setembro de 2022.

[45] BigCommerce. Available payment gateways, 2022. URL https://support.bigcommerce.com/s/article/Available-Payment-Gateways?language=en_US. Consultado a 29 de setembro de 2022.

[46] Ngoc Minh. Bigcommerce apps marketplace: All you need to know, 2022. URL https://www.beehexa.com/blog/bigcommerce-apps-marketplace. Consultado a 29 de setembro de 2022.

[47] BigCommerce. Bigcommerce pricing and plans, 2022. URL https://www.bigcommerce.com/essentials/pricing/. Consultado a 29 de setembro de 2022.

[48] Emil Kristensen. Shopify statistics you need to know (2022), 2022. URL https://www.drip.com/blog/shopify-statistics. Consultado a 30 de novembro de 2022.

[49] Shopify. App store shopify, 2022. URL https://apps.shopify.com/. Consultado a 29 de setembro de 2022.

[50] WIX. Templates wix, 2022. URL https://pt.wix.com/website/templates/html/most-popular. Consultado a 1 de dezembro de 2022.

[51] ShopWired. B2b, trade & wholesale functionality - shopwired, 2022. URL https://www.shopwired.co.uk/b2b. Consultado a 26 de setembro de 2022.

[52] Shift4Shop. A premium ecommerce solution - and it could be $0, 2022. URL https://www.shift4shop.com/end-to-end-plan.html?country=notUS. Consultado a 26 de setembro de 2022.

[53] Shift4Shop. Ecommerce shipping guide, 2022. URL https://www.shift4shop.com/shopping-cart-software/shipping-features.html. Consultado a 26 de setembro de 2022.

[54] WooCommerce. Woocommerce scaling faqs, 2022. URL https://woocommerce.com/document/woocommerce-scaling-faqs/. Consultado a 26 de setembro de 2022.

[55] Adobe Experienced Cloud. Page builder, 2023. URL https://business.adobe.com/pt/products/magento/ecommerce-cms.html. Consultado a 21 de junho de 2023.

[56] Darren DeMatas. 11 best ecommerce platforms compared & rated for 2022, 2022. URL https://www.ecommerceceo.com/ecommerce-platforms/. Consultado a 1 de dezembro de 2022.

[57] Shopify Help Center. Cryptocurrency, 2022. URL https://help.shopify.com/en/manual/payments/additional-payment-methods/cryptocurrency. Consultado a 26 de setembro de 2022.

[58] Amazon. Online payment service | amazon, 2022. URL https://pay.amazon.com/. Consultado a 19 de dezembro de 2022.

[59] Apple. Apple pay, 2022. URL https://www.apple.com/pt/apple-pay/. Consultado a 19 de dezembro de 2022.

[60] Rebecca Lake. Buy now, pay later (bnpl): What it is, how it works, pros & cons, 2022. URL https://www.investopedia.com/buy-now-pay-later-5182291. Consultado a 18 de dezembro de 2022.

[61] GoCardless. 6 benefits of payment gateway apis, 2022. URL https://gocardless.com/en-us/guides/posts/six-benefits-of-payment-gateway-apis/. Consultado a 6 de fevereiro de 2023.

[62] Stripe. Introdução aos pagamentos online, 2022. URL https://stripe.com/pt-br-us/guides/introduction-to-online-payments. Consultado a 28 de setembro de 2022.

[63] Stripe. Payments infrastructure for the internet, 2022. URL https://stripe.com/en-pt. Consultado a 30 de setembro de 2022.

[64] Stripe. Risk evaluation, 2022. URL https://stripe.com/docs/radar/risk-evaluation. Consultado a 19 de dezembro de 2022.

[65] Stripe. Build your crypto business with stripe, 2022. URL https://stripe.com/en-pt/use-cases/crypto. Consultado a 30 de setembro de 2022.

[66] Square. Power your business with square., 2022. URL https://squareup.com/us/en. Consultado a 10 de outubro de 2022.

[67] Square. Priced to help you grow and thrive., 2022. URL https://squareup.com/us/en/pricing. Consultado a 29 de setembro de 2022.

[68] Square. Bring in more business with buy now, pay later., 2022. URL https://squareup.com/us/en/buy-now-pay-later. Consultado a 29 de setembro de 2022.

[69] Raynor de Best. Global user number of paypal from 1st quarter 2010 to 3rd quarter 2022, 2022. URL https://www.statista.com/statistics/218493/paypals-total-active-registered-accounts-from-2010/. Consultado a 22 de dezembro de 2022.

[70] Paypal. Paypal merchant fees, 2022. URL https://www.paypal.com/us/webapps/mpp/merchant-fees. Consultado a 29 de setembro de 2022.

[71] Paypal. How to use crypto at checkout?, 2022. URL https://www.paypal.com/us/cshelp/article/how-to-use-crypto-at-checkout-help571. Consultado a 11 de outubro de 2022.

[72] Reuters. Amazon ceo says not adding cryptocurrency as payment option anytime soon, 2022. URL www.reuters.com/technology/amazon-ceo-says-not-adding-cryptocurrency-payment-option-anytime-soon-cnbc-2022-04-14/. Consultado a 22 de dezembro de 2022.

[73] Revolut. Accept payments easily from anywhe, 2022. URL https://www.revolut.com/en-PT/business/accept-payments/. Consultado a 12 de outubro de 2022.

[74] Revolut. Personal fees (standard), 2022. URL https://www.revolut.com/en-PT/legal/standard-fees/. Consultado a 29 de setembro de 2022.

[75] BitPay. Banking & settlements, 2022. URL https://bitpay.com/docs/settlement. Consultado a 14 de outubro de 2022.

[76] BitPay. Accept cryptocurrency payments from all over the world, 2022. URL https://bitpay.com/online-payments. Consultado a 30 de setembro de 2022.

[77] BitPay. Bitpay pricing - simple pricing for all businesses, 2022. URL https://bitpay.com/pricing/. Consultado a 14 de outubro de 2022.

[78] Coinbase Commerce. Coinbase commerce fees, 2022. URL https://help.coinbase.com/en/commerce/getting-started/fees. Consultado a 17 de outubro de 2022.

[79] Coinbase Commerce. Coinbase commerce integrations, 2022. URL https://www.coinbase.com/pt-PT/commerce/integrations. Consultado a 17 de outubro de 2022.

[80] NowPayments. Supported coins, 2022. URL https://nowpayments.io/supported-coins. Consultado a 17 de outubro de 2022.

[81] NowPayments. Pricing on crypto payments - nowpayments, 2022. URL https://nowpayments.io/pricing. Consultado a 17 de outubro de 2022.

[82] BTCPay. Integrations faq, 2022. URL https://docs.btcpayserver.org/FAQ/Integrations/. Consultado a 3 de outubro de 2022.

[83] BTCPay Server. Btcpay server documentation - what is btcpay server?, 2022. URL https://docs.btcpayserver.org/Guide/. Consultado a 21 de outubro de 2022.

[84] BTCPay Server. Btcpay server documentation - btcpay server apps, 2022. URL https://docs.btcpayserver.org/Apps/. Consultado a 22 de outubro de 2022.

[85] BitCartCC. Walkthrough, 2022. URL https://docs.bitcartcc.com/bitcartcc-basics/walkthrough#wallets. Consultado a 8 de dezembro de 2022.

[86] Hub20. Hub20 - self-hosted, open source payment gateway for ethereum-compatible blockchains, 2022. URL https://hub20.io/. Consultado a 3 de novembro de 2022.

[87] David Nadelle. Aarp study finds 1 in 3 us adults have been targeted by gift card scams, 2022. URL https://www.gobankingrates.com/saving-money/savings-advice/aarp-study-finds-1-in-3-us-adults-have-been-targeted-by-gift-card-scams/. Consultado a 22 de dezembro de 2022.

[88] GetKupon.io. getkupon.io the first - backed up nft in indonesia, 2022. URL https://getkupon.io/. Consultado a 22 de dezembro de 2022.

[89] Digite. What is scrum methodology? & scrum project management, 2022. URL https://www.digite.com/agile/scrum-methodology/. Consultado a 23 de dezembro de 2022.

[90] Digital.ai. State of agile report, 2022. URL https://info.digital.ai/rs/981-LQX-968/images/AR-SA-2022-16th-Annual-State-Of-Agile-Report.pdf. Consultado a 23 de dezembro de 2022.

[91] Coursera. The 3 scrum roles and responsibilities explained, 2022. URL https://www.coursera.org/articles/scrum-roles-and-responsibilities. Consultado a 20 de dezembro de 2022.

[92] Marie BELGODERE. The risk management process:4 essential steps, 2021. URL https://www.migso-pcubed.com/blog/pmo-project-delivery/four-step-risk-management-process/. Consultado a 7 de novembro de 2022.

[93] ADAM HAYES. Risk analysis: Definition, types, limitations, and examples, 2022. URL https://www.investopedia.com/terms/r/risk-analysis.asp. Consultado a 7 de novembro de 2022.

[94] Kat Boogaard. What is a risk matrix?, 2022. URL https://www.wrike.com/blog/what-is-risk-matrix/#How-do-you-create-a-risk-matrix-in-Excel. Consultado a 7 de novembro de 2022.

[95] NASA. Technology readiness level, 2012. URL https://www.nasa.gov/directorates/heo/scan/engineering/technology/technology_readiness_level. Consultado a 6 de fevereiro de 2023.

[96] TWI Global. What are technology readiness levels (trl)?, 2016. URL https://www.twi-global.com/technical-knowledge/faqs/technology-readiness-levels. Consultado a 6 de fevereiro de 2023.

[97] ProductPlan. Moscow prioritization, 2022. URL https://www.productplan.com/glossary/moscow-prioritization/. Consultado a 5 de dezembro de 2022.

[98] Klaytn. A comparison of blockchain network latencies, 2022. URL https://klaytn.foundation/a-comparison-of-blockchain-network-latencies/. Consultado a 6 de junho de 2023.

[99] Skale Network. Best blockchain performance: Results from dartmouth blockchain study, 2023. URL https://skale.space/blog/the-quest-for-the-best-blockchain-performance-results-from-dartmouth-blockchain-study. Consultado a 6 de junho de 2023.

[100] Jakob Nielsen. Usability engineering, 1993. URL https://www.nngroup.com/books/usability-engineering/. Consultado a 6 de junho de 2023.

[101] Filip Dimkovski. Dex vs. dapp: What's the difference?, 2022. URL https://defipedia.com/answers/dexs/dex-vs-dapp-what-s-the-difference. Consultado a 26 de dezembro de 2022.

[102] Filip Dimkovski. Uniswap protocol, 2022. URL https://uniswap.org/. Consultado a 26 de dezembro de 2022.

[103] Hub20. Faq, 2022. URL https://docs.hub20.io/faq/. Consultado a 17 de outubro de 2022.

[104] Alchemy. Alchemy - the web3 development platform, 2023. URL https://www.alchemy.com/. Consultado a 1 de março de 2023.

[105] Moralis. Moralis web3 - enterprise-grade web3 apis, 2023. URL https://moralis.io/. Consultado a 22 de junho de 2023.

[106] Quicknode. The blockchain development platform. learn. build. scale., 2023. URL https://www.quicknode.com/. Consultado a 22 de junho de 2023.

[107] Pinata. Pinata - powering the future of media distribution, 2023. URL https://www.pinata.cloud/. Consultado a 4 de março de 2023.

[108] Electrum. Electrum bitcoin wallet, 2023. URL `https://electrum.org/`. Consultado a 6 de junho de 2023.

[109] Python.org. Welcome to python.org, 2022. URL `https://www.python.org/`. Consultado a 26 de dezembro de 2022.

[110] FastAPI. Fastapi, 2022. URL `https://fastapi.tiangolo.com/`. Consultado a 22 de dezembro de 2022.

[111] SmartBear Software. Swagger ui | rest api documentation tool, 2023. URL `https://swagger.io/tools/swagger-ui/`. Consultado a 20 de junho de 2023.

[112] PostgreSQL. Postgresql: The world's most advanced open source relational database, 2022. URL `https://www.postgresql.org/`. Consultado a 21 de novembro de 2022.

[113] NUXT.js. Nuxt: A abstração intuitiva de vue, 2022. URL `https://nuxtjs.org/pt/`. Consultado a 28 de novembro de 2022.

[114] BitcartCC. Shopify, 2022. URL `https://docs.bitcartcc.com/integrations/shopify`. Consultado a 15 de dezembro de 2022.

[115] Jira | Atlassian. Jira | issue & project tracking, 2023. URL `https://www.atlassian.com/software/jira`. Consultado a 31 de março de 2023.

[116] Confluence | Atlassian. Confluence | your remote-friendly team workspace | atlassian, 2023. URL `https://www.atlassian.com/software/confluence`. Consultado a 31 de março de 2023.

[117] Vikash Koushik. 5 git workflows and branching strategy you can use to improve your development process, 2022. URL `https://rovitpm.com/5-git-workflows-to-improve-development/`. Consultado a 31 de março de 2023.

[118] Ethereum Foundation. Remix ethereum, 2022. URL `https://remix.ethereum.org/`. Consultado a 24 de fevereiro de 2023.

[119] Ethereum Foundation. Solidity documentation, 2022. URL `https://soliditylang.org/`. Consultado a 15 de março de 2023.

[120] polygonscan.com. polygonscan.com, 2022. URL `https://mumbai.polygonscan.com/`. Consultado a 19 de março de 2023.

[121] Metadata Standards. Opensea.io, 2023. URL `https://docs.opensea.io/docs/metadata-standards`. Consultado a 26 de fevereiro de 2023.

[122] OpenSea. Opensea.io, 2023. URL `https://opensea.io`. Consultado a 28 de fevereiro de 2023.

[123] Fredrik Lundh. Pillow, 2023. URL `https://pillow.readthedocs.io/en/stable/`. Consultado a 5 de maio de 2023.

[124] Google. Material design 2, 2023. URL `https://m2.material.io/`. Consultado a 5 de maio de 2023.

[125] Shopify Dev Docs. Checkout ui extensions, 2023. URL `https://shopify.dev/docs/api/checkout-ui-extensions`. Consultado a 19 de maio de 2023.

[126] Shopify. Shopify plus pricing, 2023. URL `https://www.shopify.com/plus/pricing`. Consultado a 19 de maio de 2023.

[127] Shopify. Build the best commerce apps, 2023. URL `https://shopify.dev/docs/apps`. Consultado a 19 de maio de 2023.

[128] Shopify. Shopify polaris - build. contribute. evolve., 2023. URL `https://polaris.shopify.com/`. Consultado a 19 de maio de 2023.

[129] Vite. Vite | next generation frontend tooling, 2023. URL `https://vitejs.dev/`. Consultado a 30 de maio de 2023.

[130] TypeScript. Typescript is javascript with syntax for types., 2023. URL `https://www.typescriptlang.org/`. Consultado a 30 de maio de 2023.

[131] React Readux. React redux official react bindings for redux, 2023. URL `https://react-redux.js.org/`. Consultado a 30 de maio de 2023.

[132] React Readux. Rapidly build modern websites without ever leaving your html., 2023. URL `https://tailwindcss.com/`. Consultado a 30 de maio de 2023.

[133] Estágio DEI - FCTUC. Prototype blockchain loyalty platform, 2022. URL `http://estagios.dei.uc.pt/cursos/mei/ano-lectivo-2022-2023/atribuidos-2022-2023/?idestagio=4885`. Consultado a 6 de junho de 2023.

[134] IBM. How does software testing work?, 2023. URL `https://www.ibm.com/topics/software-testing`. Consultado a 2 de junho de 2023.

[135] Indium. 7 reasons why software testing is important, 2021. URL `https://www.indiumsoftware.com/blog/why-software-testing/`. Consultado a 2 de junho de 2023.

[136] Pytest-dev team. pytest: helps you write better programs, 2015. URL `https://docs.pytest.org/en/7.3.x/`. Consultado a 2 de junho de 2023.

[137] Docker Inc. Docker: Develop faster. run anywhere., 2015. URL `https://www.docker.com/`. Consultado a 2 de junho de 2023.

[138] Postman. Postman api platform, 2023. URL `https://www.postman.com/`. Consultado a 2 de junho de 2023.

[139] Shopify Dev Docs. Shopify api rate limits, 2023. URL `https://shopify.dev/docs/api/usage/rate-limits`. Consultado a 5 de junho de 2023.

[140] ProductPlan. User story, 2023. URL `https://www.productplan.com/glossary/user-story/`. Consultado a 25 de maio de 2023.

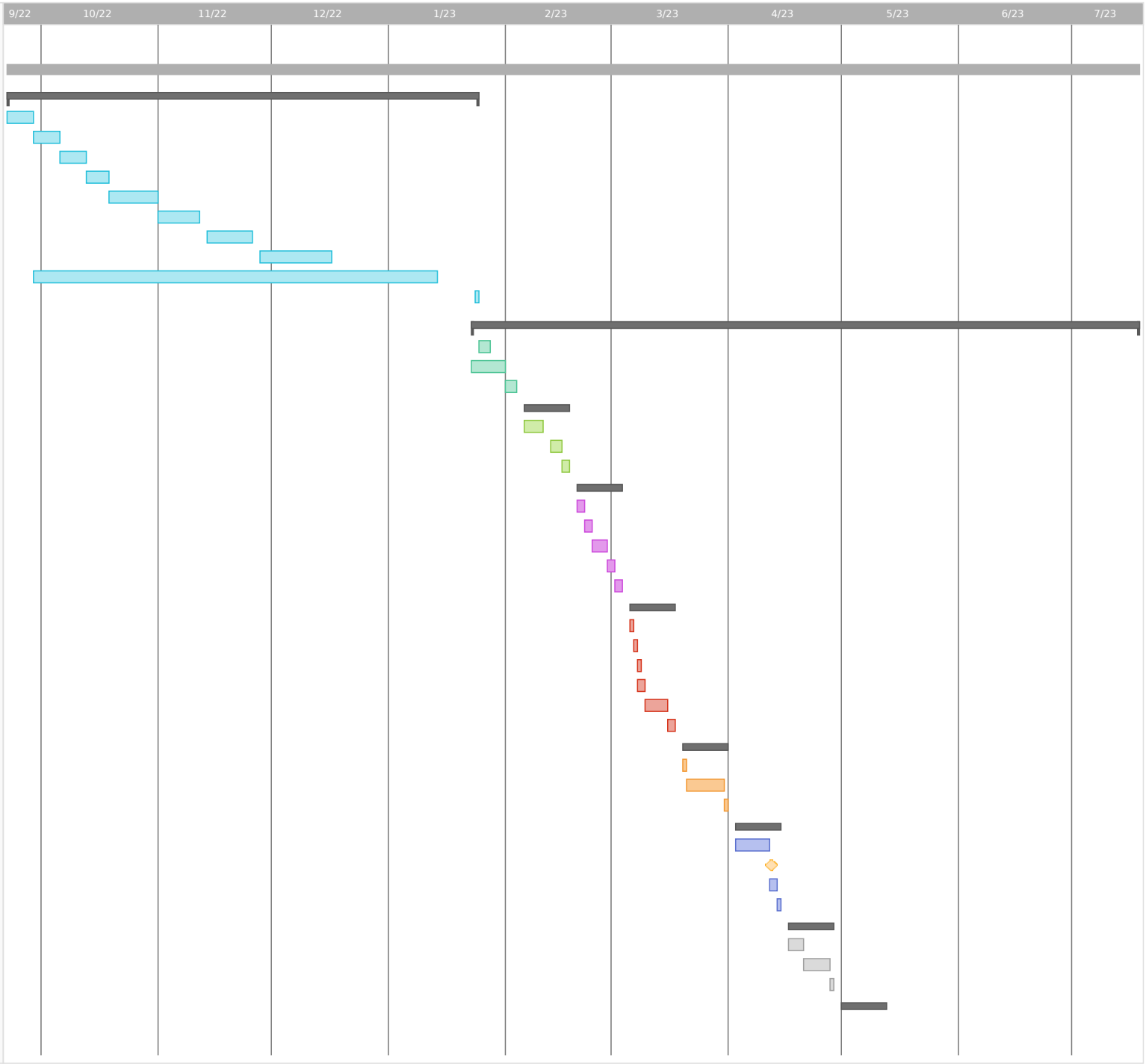
Appendices

Apêndice A

Planeamento

Nesta secção, encontra-se disponível o planeamento do estágio para ambos os semestres.

Cronologia Esperada



Primeiro Semestre
- Contextualização do problema
- Estudo de tokens/criptomoedas em B...
- Estudo de plataformas de e-commer...
- Estudo dos diferentes métodos de p...
- Estudo de plataformas self-hosted e ...
- Metodologia e Planeamento
- Identificação de Requisitos Funcionai...
- Arquitetura e prototipagem
- Relatório Intermédio
- Apresentação do relatório intermédio

Segundo Semestre
- Revisão e edição do relatório
- Prototipagem(cont)
- Análise da Ferramenta BitCartCC
- **Sprint #1**
  - Planeamento e levantamento de r...
  - Elaboração de tutorais
  - Documentação Sprint #1
- **Sprint #2**
  - Suporte para a rede Polygon
  - Implementação de Tokens ERC20 ...
  - Criação e deployment do Smart Co...
  - Listar vouchers do comerciante
  - Visualizar detalhes do voucher
- **Sprint #3**
  - Transferência de voucher
  - Eliminar o voucher
  - Apresentação WIT
  - Listar vouchers disponíveis no clien...
  - Estudo da ferramenta Shopify para ...
  - Documentação Sprint #3
- **Sprint #4**
  - Integração com shopify
  - Aplicação do desconto no processo...
  - Documentação Sprint #4
- **Sprint #5**
  - Finalização do Processo de compra
  - Apresentação WIT
  - Criação de NFT
  - Documentação Sprint #5
- **Sprint #6**
  - Criação de NFT
  - Desenvolvimento de uma Aplicaçã...
  - Documentação Sprint #6
- **Sprint #7**

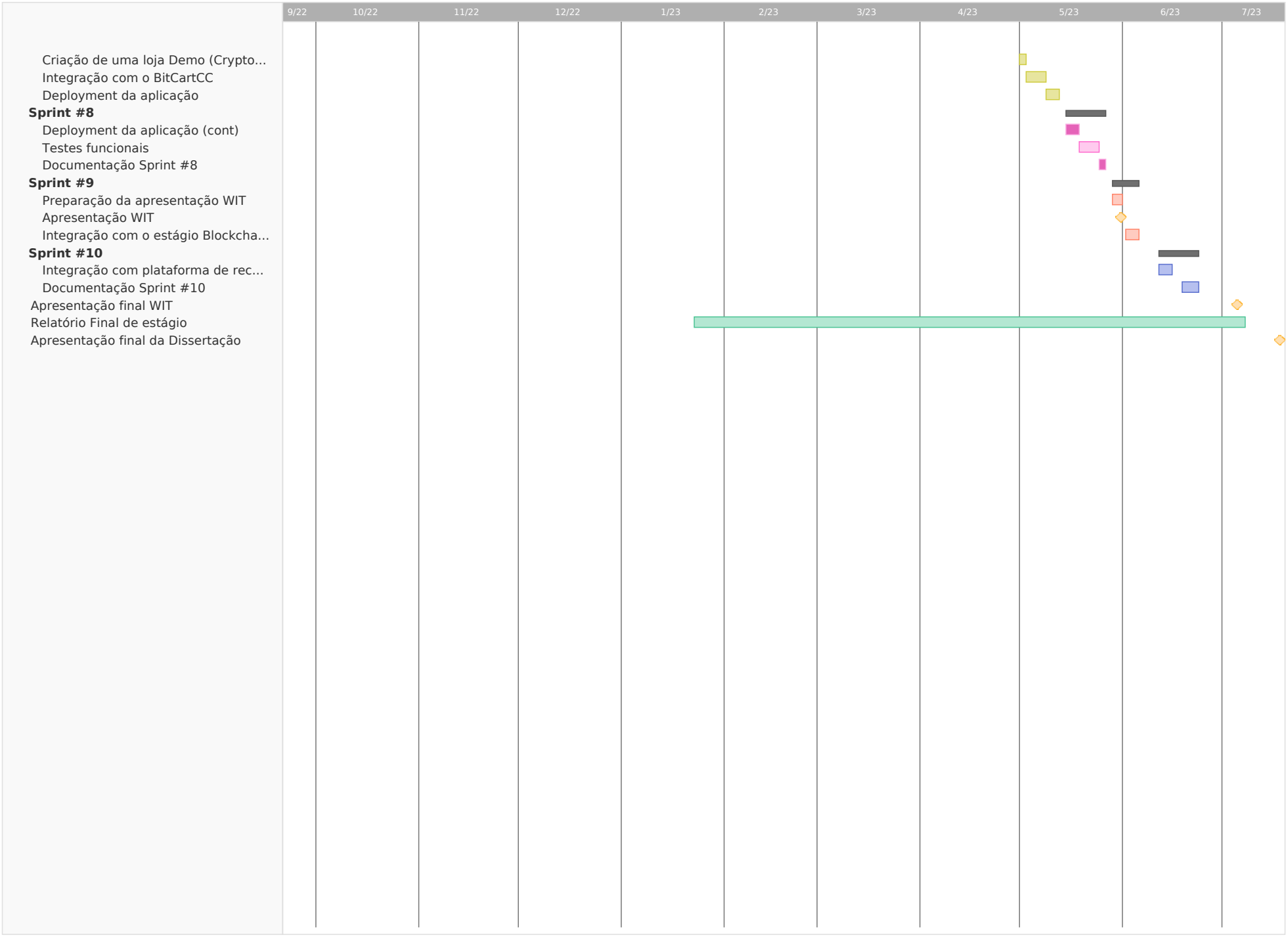


Figura A.1: Diagrama de Gantt com a cronologia esperada para ambos os semestres

Apêndice B

User Stories

A secção a seguir descreve a abordagem utilizada para recolher as *user stories*, com o objetivo de apresentar de forma geral a funcionalidade do sistema a partir da perspetiva do utilizador final. Essa abordagem segue o modelo padrão [140]:

Como [**utilizador**], eu quero [**funcionalidade**] para que [**benefício**]

Este formato é amplamente reconhecido na indústria de desenvolvimento de software como uma maneira eficaz de descrever as necessidades e expectativas do utilizador de forma concisa. Cada *user stories* é composta por uma identificação do utilizador, uma descrição da funcionalidade desejada e o benefício associado à sua realização.

Suporte

US-1: Criar um smart-contract para utilização de NFT

- **Descrição:** Como suporte, eu quero criar um smart contract, para facilitar a criação, gestão e utilização de vouchers em formato de NFT.
- **Critérios de aceitação:** O smart-contract deve ser capaz de criar e gerir NFTs que serão utilizados como vouchers de pagamento, incluindo a atribuição de metadados relevantes e os métodos de transferência, criação e exclusão dos vouchers.
- **Prioridade**: Must

US-2: Guardar informação dos vouchers nos metadados do NFT

- **Descrição:** Como suporte, quero que as informações dos vouchers sejam armazenadas nos metadados associados aos NFTs correspondentes para que tenha toda a informação sobre um determinado voucher.
- **Critérios de aceitação:** Os metadados do NFT devem ser capazes de armazenar informações como nome, imagem, descrição, tipo de desconto e outros detalhes relevantes do voucher.
- **Prioridade**: Must

US-3: Integrar a solução na plataforma de e-commerce Shopify e num website construído à medida

- **Descrição:** Como suporte, quero integrar a solução de pagamento na plataforma de e-commerce Shopify, permitindo que os comerciantes disponibilizem a funcionalidade de vouchers e criptomoedas aos seus clientes.
- **Critérios de aceitação:** A solução de pagamento deve ser integrada de forma completa e funcional na plataforma de e-commerce Shopify, permitindo que os comerciantes aceitem vouchers e criptomoedas como opções de pagamento.
- **Prioridade**: Must

US-4: Integração com Torus Wallet

- **Descrição:** Como suporte, eu quero garantir a integração da solução com a Torus Wallet, para oferecer aos utilizadores uma experiência de pagamento simplificada e segura combinando os métodos de autenticação Web2 com a segurança Web3.
- **Critérios de aceitação:** Os utilizadores podem selecionar a Torus Wallet como meio de pagamento no processo de *checkout* de forma a pagar a fatura em questão.
- **Prioridade**: Could

US-5: Aceitar Tokens ERC20 e NFT

- **Descrição:** Como suporte, eu quero aceitar tokens ERC20 e NFT, para oferecer uma variedade de opções de pagamento aos clientes e ampliar a adoção de criptomoedas na plataforma.
- **Critérios de aceitação:** O comerciante adiciona uma carteira da criptomoeda que pretende aceitar e esta fica disponível como meio de pagamento.

- **Prioridade**: Must

US-6: Integrar a aplicação com o estágio *Blockchain Loyalty Platform*

- **Descrição:** Como suporte, eu quero integrar a solução com o estágio *Blockchain Loyalty Platform* para que as recompensas emitidas pelo programa de *loyalty* para o cliente possam ser utilizadas na loja do comerciante.
- **Critérios de aceitação:** O processo de criação das recompensas no programa de *loyalty* deve seguir a estrutura de metadados criados de modo a que o desconto criado possa ser utilizado pela aplicação.
- **Prioridade**: Could

US-7: Integrar a aplicação com o estágio a plataforma interna de recompensas

- **Descrição:** Como suporte, eu quero integrar a solução a plataforma de recompensas interna de modo a trocar essas recompensas por bens físicos utilizando a nossa aplicação.
- **Critérios de aceitação:** O cliente o processo de *checkout* consegue usufruir da sua recompensa em NFT proveniente da plataforma de recompensas,
- **Prioridade**: Could

Comerciante

US-8: Visualizar os detalhes do voucher

- **Descrição:** Como comerciante, eu quero visualizar os detalhes do meu voucher, de modo a ter acesso às informações relevantes sobre o mesmo.
- **Critérios de aceitação:** O comerciante deve ser capaz de visualizar todas as informações relevantes do voucher, incluindo o valor, tipo de desconto, nome descrição e quaisquer outras informações específicas relacionadas ao NFT.
- **Prioridade**: Must

US-9: Transferir o voucher para outro endereço

- **Descrição:** Como comerciante, quero ter a opção de transferir a posse do meu voucher para outro endereço de modo a que seja utilizado por um cliente.

- **Critérios de aceitação:** O comerciante deve ser capaz de iniciar uma transferência do voucher para outro endereço, garantindo que a posse seja transferida corretamente.
- **Prioridade**: Must

US-10: Visualizar as interações do voucher num explorador de blocos

- **Descrição:** Como comerciante, quero ter a capacidade de visualizar as transações e interações relacionadas ao meu voucher através de um explorador de blocos.
- **Critérios de aceitação:** O comerciante deve ser capaz de rastrear e visualizar todas as transações e interações associadas ao voucher em um explorador de blocos, garantindo a transparência e a validade das operações.
- **Prioridade**: Must

US-11: Eliminar o voucher

- **Descrição:** Como comerciante, desejo ter a opção de eliminar permanentemente o meu voucher, caso não seja mais necessário ou válido.
- **Critérios de aceitação:** O comerciante deve ser capaz de excluir o voucher de forma permanente, garantindo que não seja mais utilizado.
- **Prioridade**: Must

US-12: Visualizar o histórico de transações efetuadas

- **Descrição:** Como comerciante, quero ter acesso ao histórico de transações efetuadas na minha loja, permitindo monitorizar e analisar as transações realizadas.
- **Critérios de aceitação:** O comerciante deve ter a capacidade de visualizar um registo detalhado das transações efetuadas, incluindo informações a *hash* de cada transação.
- **Prioridade**: Must

US-13: Criar os meus próprios vouchers

- **Descrição:** Como comerciante, quero ter a capacidade de criar e gerir meus próprios vouchers, com valores e características personalizadas presentes nos metadados do NFT.

- **Critérios de aceitação:** O comerciante deve ter a capacidade de criar vouchers personalizados, definindo valores, validades e outros detalhes conforme suas necessidades.

- **Prioridade**: Must

US-14: Criar os meus próprios vouchers dentro da ferramenta Shopify

- **Descrição:** Como comerciante, quero ter a facilidade de criar meus próprios vouchers diretamente na plataforma Shopify, sem a necessidade de usar interfaces ou ferramentas externas.

- **Critérios de aceitação:** O comerciante deve ter a capacidade de criar vouchers personalizados diretamente na plataforma Shopify, com opções para definir valores, validades e outras características, sem a necessidade de acessar interfaces externas.

- **Prioridade**: Must

US-15: Visualizar o saldo das minhas wallets

- **Descrição:** Como comerciante, eu quero visualizar o saldo da minha loja, de modo a visualizar o montante que tenho disponível nas minhas carteiras.

- **Critérios de aceitação:** O comerciante deve conseguir visualizar o montante disponível das suas carteiras dentro do painel de administrador.

- **Prioridade**: Must

US-16: Gerir as minhas carteiras

- **Descrição:** Como comerciante, gerir as minhas carteiras de modo a adicionar ou remover novas criptomoedas aceites na minha loja.

- **Critérios de aceitação:** O comerciante deve conseguir adicionar novas carteiras à aplicação estipulando valores como nome, chave pública e contrato da criptomoeda a adicionar.

- **Prioridade**: Must

Consumidor

US-17: Visualizar vouchers no ato de pagamento

- **Descrição:** Como consumidor, quero ter a capacidade de visualizar os vouchers disponíveis para utilização no momento do pagamento de modo a que os consiga utilizar.
- **Critérios de aceitação:** O consumidor deve ser capaz de visualizar uma lista dos vouchers disponíveis para seleção no momento do pagamento, apresentando as informações relevantes, como valor e nome.
- **Prioridade**: Must

US-18: Alterar o voucher escolhido

- **Descrição:** Como consumidor, quero ter a opção de alterar o voucher selecionado durante o processo de pagamento, caso necessário.
- **Critérios de aceitação:** O consumidor deve ter a capacidade de visualizar e modificar o voucher selecionado antes de finalizar o pagamento, garantindo a flexibilidade na escolha do voucher adequado e a sua alteração refletir-se-à no montante pago.
- **Prioridade**: Must

US-19: Efetuar o pagamento com criptomoedas

- **Descrição:** Como consumidor, quero poder realizar o pagamento utilizando criptomoedas, conforme selecionado durante o processo de pagamento.
- **Critérios de aceitação:** O consumidor deve ter a capacidade de realizar o pagamento utilizando a criptomoeda selecionada, com o valor correspondente ao voucher escolhido.
- **Prioridade**: Must

US-20: Pagar utilizando carteiras externas como Metamask e Wallet Connect

- **Descrição:** Como consumidor, quero ter a opção de efetuar o pagamento utilizando carteiras externas, como Metamask e Wallet Connect, para maior conveniência e segurança.

- **Critérios de aceitação:** O consumidor deve ser capaz de conectar e utilizar carteiras externas, como Metamask e Wallet Connect, para autorizar e efetuar o pagamento com criptomoedas.
- **Prioridade**: Must

US-21: Escolher o método de pagamento(*)

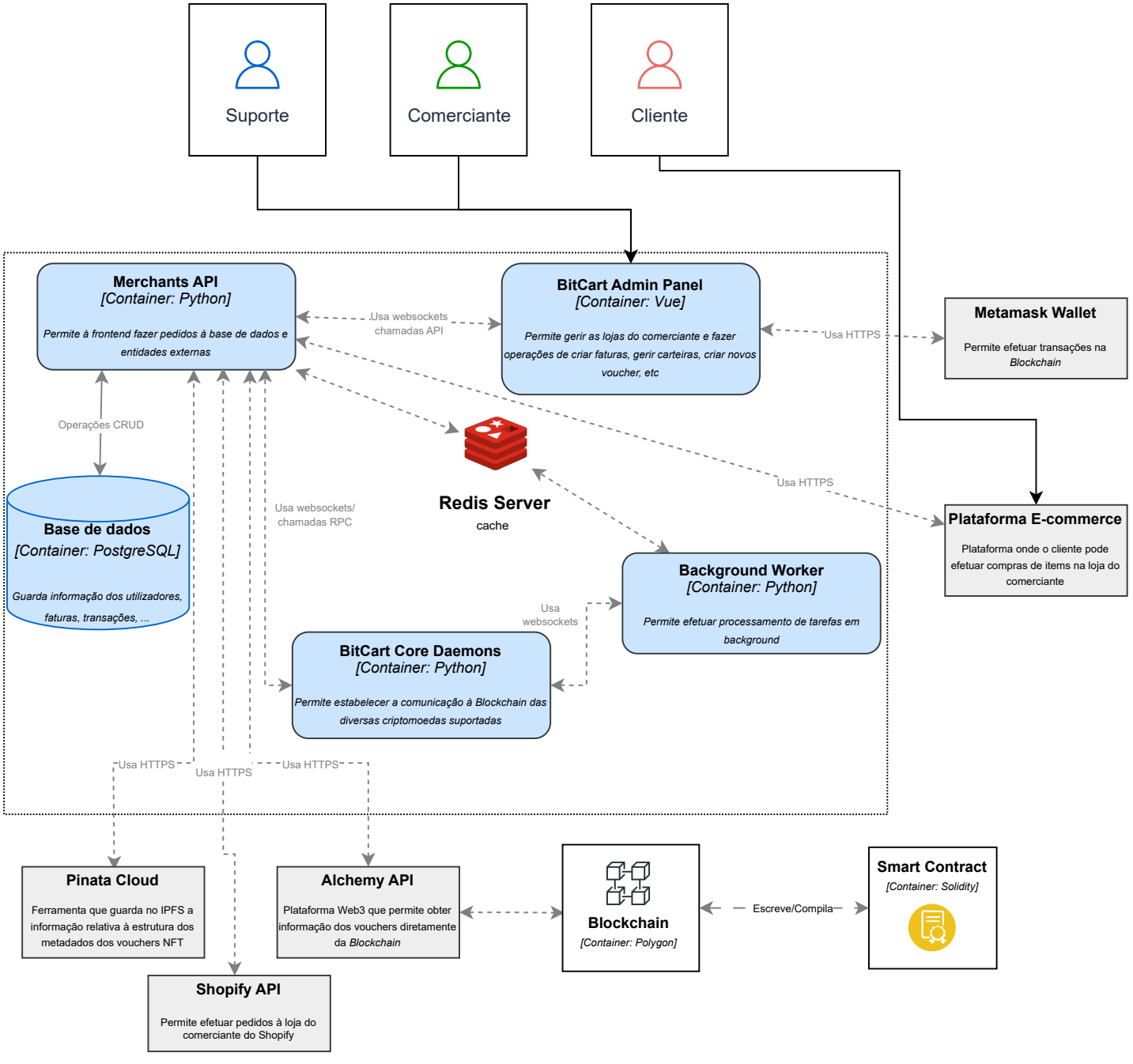
- **Descrição:** Como consumidor, desejo ter a opção de utilizar vouchers NFT no ecrã de pagamento do Shopify. Além disso, desejo ter a flexibilidade de escolher o método de pagamento para o valor restante, seja por cartão de crédito ou criptomoedas.
- **Critérios de aceitação:** O consumidor deve ser capaz de usufruir do desconto no ecrã de *checkout* e ter as opções de pagamento de modo a escolher qual das opções é que irá usufruir: as já suportadas pela loja ou criptomoedas.
- **Prioridade**: Must

* O caso de uso identificado com (*), US-21, representa uma situação que inicialmente estava prevista, no entanto após uma deliberação e estudo da plataforma Shopify, a equipa chegou à conclusão que não era possível tal execução tal como foi referido na Subseção 8.6.1

Apêndice C

Arquitetura

Nesta secção, encontra-se disponível a visão da Arquitetura da solução do presente estágio.


Suporte
Comerciante
Cliente
Merchants API
[Container: Python]
Permite à frontend fazer pedidos à base de dados e entidades externas
Usa websockets chamadas API
BitCart Admin Panel
[Container: Vue]
Permite gerir as lojas do comerciante e fazer operações de criar faturas, gerir carteiras, criar novos voucher, etc
Usa HTTPS
Metamask Wallet
Permite efetuar transações na Blockchain
Operações CRUD
Redis Server
cache
Usa HTTPS
Base de dados
[Container: PostgreSQL]
Guarda informação dos utilizadores, faturas, transações, ...
Usa websockets/ chamadas RPC
Plataforma E-commerce
Plataforma onde o cliente pode efetuar compras de items na loja do comerciante
Background Worker
[Container: Python]
Permite efetuar processamento de tarefas em background
Usa websockets
BitCart Core Daemons
[Container: Python]
Permite estabelecer a comunicação à Blockchain das diversas criptomoedas suportadas
Usa HTTPS
Usa HTTPS
Usa HTTPS
Pinata Cloud
Ferramenta que guarda no IPFS a informação relativa à estrutura dos metadados dos vouchers NFT
Alchemy API
Plataforma Web3 que permite obter informação dos vouchers diretamente da Blockchain
Blockchain
[Container: Polygon]
Escreve/Compila
Smart Contract
[Container: Solidity]
Shopify API
Permite efetuar pedidos à loja do comerciante do Shopify