PM

Sistema para prestador de serviços de confiança eIDAS

PROJETO DE MESTRADO

Francisco José Figueira Gouveia Caldeira
MESTRADO EM ENGENHARIA INFORMÁTICA

UNIVERSIDADE da MADEIRA
A Nossa Universidade
www.uma.pt

novembro | 2023

Sistema para prestador de serviços de confiança eIDAS

PROJETO DE MESTRADO

Francisco José Figueira Gouveia Caldeira
MESTRADO EM ENGENHARIA INFORMÁTICA

ORIENTAÇÃO
Filipe Magno de Gouveia Quintal

CO-ORIENTAÇÃO
Sofia Catarina Câmara Leme Pessanha de Meneses

Agradecimentos

Em primeiro agradeço ao meu orientador, Excelentíssimo Professor Doutor Engenheiro Filipe Magno Gouveia Quintal, por todo o apoio, orientações prestadas, pela disponibilidade incondicional e conhecimentos transmitido ao longo do desenvolvimento deste projeto de mestrado.

Agradeço também à minha coorientadora, a Sofia Catarina Câmara Leme Pessanha de Meneses, coordenadora de projetos da ACIN iCloud Solutions, por ser possível avançar este projeto na tese de mestrado.

Por último, aos meus familiares, professores e colegas de escola e trabalho que acompanharam ao longo do meu percurso académico na Universidade da Madeira.

Resumo

A Global Trusted Sign é uma entidade certificadora que comercializa serviços de confiança, nomeadamente selos temporais, certificados de assinatura eletrónica e selos eletrónicos que podem ser tanto qualificados como avançados, e ainda certificados de autenticação de websites (TLS/SSL). É proposto o desenvolvimento do backend do portal para a major v3.0.0, utilizando a framework Laravel na construção de uma Application Programming Interface (API) que inclua todos os seguintes requisitos: o registo de novos utilizadores, autenticação OAuth 2.0, aquisição de produtos e serviços por um sistema de carrinho de compras, integração com sistemas para geração de pagamentos e fatura, gestão dos produtos comprados e a aplicação dos certificados digitais em documentos. Como linguagem de consulta de dados à API, foi utilizado o GraphQL, possibilitando o frontend a solicitar apenas os dados necessários numa única chamada à API.

Neste relatório irá ser abordado o processo de desenvolvimento seguido um método formal de Engenharia de Software, desde a arquitetura do sistema, a análise dos requisitos, a modelação da base de dados, até à implementação final. Este projeto visa implementar testes unitários oferecendo confiança do correto funcionamento em cada lançamento, com uma single-page application realizado pelo frontend, com um layout elegante e moderno face à plataforma online atual. Nesse sentido, o objetivo deste projeto de mestrado consiste em melhorar a rapidez nos pedidos, clareza no código concebido de backend e aumentar o número de vendas da plataforma.

Palavras-Chave

API, Laravel, OAuth2.0, GraphQL, Testes HTTP, Engenharia de software.

Abstract

Global Trusted Sign is a certifying entity that sells trusted services, namely timestamps, digital signature certificates, electronic stamps, website authentication (TLS/SSL), certificates that can be qualified as well as advanced. The proposal entails developing the backend of the portal for the major version v3.0.0, using the Laravel framework to build an Application Programming Interface (API), therefore includes all the following requirements: the registration of new users, OAuth 2.0 authentication, purchase of products and services through a shopping cart system, integration with systems for generating payments and invoices, management of the purchased products, and applying digital certificates to documents. The API utilizes GraphQL as the data query language, allowing the frontend to request only the necessary data in a single API call.

This document addresses the development process followed by a formal method of Software Engineering, from system architecture, requirements analysis, database modeling, to the implementation. This project also aims at the implementation of unit tests, offering confidence in the correct functioning in each release, with a single page application conducted by the frontend, with an elegant and modern layout compared to the current online platform. In this context, the objective of this master's project is to improve the of speed in requests, clarity in the backend code and increase the number of platform sales.

Keywords

API, Laravel, OAuth2.0, GraphQL, HTTP Tests, Software Engineering.

Índice

Índice de tabelas

Índice de figuras

Lista de acrónimos e abreviações

2FA - Autenticação de dois fatores

ACID - Atomicidade, Consistência, Isolamento e Durabilidade

AdES - Assinaturas Eletrónicas Avançadas

API - Application Programming Interface

ASiC - Associated Signature Container

BD - Base de Dados

BDOR - Base de dados objeto-relacional

BFSI - Banking, financial services e insurance

BYOK - Bring Your Own Key

CA - Autoridade de Certificação

CAdES - Cryptographic Message Syntax (CMS) Advanced Electronic Signature

CAP - Consistência, Disponibilidade e Tolerância à Partição

CC - Common Criteria for Information Technology Security Evaluation

CI - CodeIgniter

CI/CD - Continuous Integration/Continuous Delivery

CKMS - Crypto Key Management System

CMC - Certificate Management over Cryptographic Message Syntax (CMS)

CMP - Certificate Management Protocol

CMS - Cryptographic Message Syntax

CPU - Central processing unit

CSS - Cascading Style Sheets

CRL - Lista de Revogação de Certificado

CRUD - Create, Read, Update and Delete

DBA - Database Administrator

DDA - Departamento de desenvolvimento aplicacional

DI - Injeção de dependência

EAL4 - Evaluation Assurance level 4+

EDI - Electronic Data Interchange

EID - Identificação eletrónica

ER - Entidade-Relacionamento

ERP - Enterprise Resource Planning

EST - Enrollment over Secure Transport

ETSI - European Telecommunications Standards Institute

GNS - Gabinete Nacional de Segurança

GTS - Global Trusted Sign

GUI - Graphical user interface

HIPPA - Health Insurance Portability and Accountability Act

HSM - Módulos de segurança de hardware

HTML - HyperText Markup Language

HTTP - Hypertext Transfer Protocol

IDE - Integrated Development Environment

IDC - IDentity Connector

IoC - Inversão de controle

ISO - International Organization for Standardization

JDK - Java Development Kit

JS - JavaScript

JSON - JavaScript Object Notation

JSP - JavaServer Pages

MYOK - Manage Your Own Key

MVC - Model-View-Controller

MVT - Model View Template

NoSQL - Not only Structured Query Language

NPM - Node Package Manager

OAUTH - Open Authentication

OCI - Oracle Cloud Infrastructure

ORM - Object-relational mapper

PAdES - PDF Advanced Electronic Signature

PCI DSS - Payment Card Industry Data Security Standard

PDF - Portable Document Format

PHP - Hypertext Preprocessor

PSR - PHP Standard Recommendation

PHP-FIG - PHP Framework Interoperability Group

PKI - Infraestrutura de chave pública

QES - Assinaturas Eletrónicas Qualificadas

QTSP - Prestador de Serviço de Confiança Qualificado

REST - Representational State Transfer

RFC - Request for Comments

RGPD - Regulamento Geral sobre a Proteção de Dados

SCEP - Simple Certificate Enrollment Protocol

SGBD - Sistema de Gestão de Base de Dados

SGS - Société Générale de Surveillance

SGSI - Sistemas de Gestão da Segurança da Informação

SLA - Service Level Agreement

SO - Sistema Operativo

SOAP - Simple Object Access Protocol

SPA - Single Page Application

SQL - Structured Query Language

SSCD - Dispositivo de criação de assinatura segura

TI - Tecnologias de Informação

TL - Trusted List

TLS/SLL EV/OV - Transport Layer Security/Secure Sockets Layer Organization Validation/Extended Validation

UE - União Europeia

UMA - Universidade da Madeira

URI - Uniform Resource Identifier

VM - Máquina virtual

VSC - Virtual Smart Card

WSDL - Web Services Description Language

WSL - Subsistema Windows para Linux

WYSIWYS - What You See Is What You Sign

XAdES - XML Advanced Electronic Signature

XML - Extensible Markup Language

XSS - Cross-site scripting

Glossário

Framework - Conjunto de classes concretas e abstratas, explicitamente projetadas para serem usadas em conjunto de bibliotecas permitindo um rápido desenvolvimento web.

Cross-platform - Abordagem de programação utilizada para dar suporte total em para várias plataformas, funcionando assim em vários sistemas operacionais ou dispositivos.

GraphQL- Linguagem de consulta, podendo consultar exatamente os dados pedidos/necessários evitando o uso de recursos, enviando apenas uma única solicitação ao servidor.

Mutations- Tipo de operação da linguagem de GraphQL, que tem a permissão para alterar os dados no servidor.

Queries - Tipo de operação da linguagem de GraphQL, de consulta à API podendo receber argumentos e retornar um resultado fixo.

Cloud - Ambiente na nuvem, em que os recursos, como armazenamento, processamento e aplicação, são disponibilizados pela Internet e acedidos remotamente.

1. Introdução

Com a evolução dos *e-commerce*, os utilizadores temiam que as suas informações pessoais fossem manipuladas ou utilizadas por pessoas com más intenções. Surgiu então uma necessidade de confiança entre o vendedor e o comprador, usando assim uma estrutura de chaves criptográficas pública e privada, onde uma chave pública é mapeada para uma chave privada [1]. Estas chaves são usadas nas comunicações entre ambas as partes, onde a chave pública encripta a informação e apenas o detentor da chave privada consegue desencriptar esta informação.

Uma infraestrutura de chave pública (PKI), inclui uma Autoridade de Certificação (CA). Esta estrutura é composta por um repositório de certificados que os armazena, emite e assina. Esta Autoridade também pode revogar certificados digitais caso se tornem inválidos, por emissão imprópria ou defeituosa do certificado, erro/alteração/atualização de informações de pessoa física ou coletiva, perda, roubo, acesso indevido, comprometimento ou suspeita de comprometimento da chave privada correspondente entre outros fatores, ficando o seu registo público numa lista de revogação de certificado (CRL). A CA verifica as identidades que procuram armazenar os seus certificados digitais [1].

A comercialização de certificados digitais surge com a evolução das tecnologias. Esta solução é utilizada por diversas empresas na substituição do papel, conforme o processo de assinatura descrito na figura 1, melhorando a segurança e proteção, aumentando a eficiência e reduzindo os custos permitindo automatizar o processo de assinatura eletrónica e carimbo de data e hora, na comunicação por e-mails, nos PDFs, nos documentos do Word e noutros tipos de documentos. Sendo assim são usados em *banking, financial services* e *insurance* (BFSI), autoridades governamentais, área de saúde, educação e outras áreas [2].

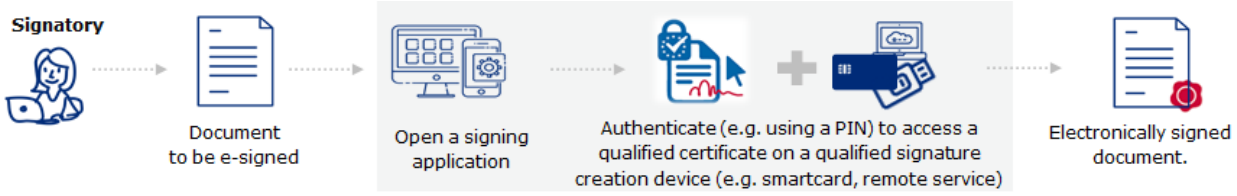

Figura 1 - Processo de assinatura [3].

1.1. Contextualização

Este projeto foi proposto pela empresa ACIN no âmbito pedagógico da unidade curricular de Projeto do Mestrado em Engenharia Informática, na Universidade da Madeira (UMA) no ano letivo de 2022/2023. Nas instalações da empresa realizou-se todo o desenvolvimento e a obtenção dos resultados deste projeto.

1.1.1. Contexto Institucional - Grupo ACIN

O Grupo ACIN iniciou-se em 1999 como uma empresa dedicada à formação na área das Tecnologias de Informação.

Contudo, devido às necessidades do mercado de Tecnologias de Informação (TI) e oportunidades de negócio, o Grupo ACIN investiu e dedicou-se à criação e desenvolvimento de soluções tecnológicas, práticas, económicas e user-*friendly*, que vão ao encontro às diferentes necessidades do mercado e dos clientes.

A firma está sediada no concelho da Ribeira Brava, na Madeira, e possui sucursais em Lisboa, Porto e Açores. Para garantir o seu funcionamento está auditada pela empresa líder mundial em inspeção, verificação, teste e certificação, a *Société Générale de Surveillance* (SGS), cumprindo com as normas ISO 9001, ISO 27001 e a ISO 20000. Ainda possui conformidade eIDAS, uma certificação como prestadores de serviços de confiança, credenciada pelo Gabinete Nacional de Segurança (GNS) e implementada em conformidade com o Regulamento eIDAS (Reg EU N.º 910/2014) [4].

Normas ISO

A *International Organization for Standardization* (ISO) é uma organização internacional não governamental independente, fundada em Londres a 1946. O desenvolvimento destes standards/padrões ISO envolvem um conjunto de processos, que são iniciados numa proposta por especialistas que procuram promover as condições de saúde, segurança, qualidade ou meio ambiente de produtos, serviços, materiais entre outras gamas de atividades [4].

ISO 9001

Esta norma garante os compromissos de qualidade e satisfação dos serviços prestados, potencializando continuamente o desempenho geral com o aumento da fiabilidade dos processos da organização em questão. É então demonstrado o seu compromisso em cumprir os mais altos padrões de qualidade e de satisfação dos clientes onde são documentados processos, procedimentos e responsabilidades envolvidos no cumprimento de políticas e objetivos de qualidade [5].

A ISO 9001 baseia-se então pelos seguintes princípios de focalização no cliente, liderança, envolvimento das pessoas, abordagem por processos, contexto organizacional, melhoria contínua, tomada de decisões baseadas em factos e raciocínio baseado na gestão do risco.

ISO 27001

A certificação ISO 27001 é obtida através de uma auditoria de Sistemas de Gestão da Segurança da Informação (SGSI). Com esta auditoria as organizações são avaliadas no controlo em termos de risco, ameaças e vulnerabilidades à segurança da informação, garantindo a continuidade do negócio e a proteção dos dados. Através desta norma, uma organização destaca-se da concorrência pelo forte controlo da segurança da informação e nos outros fatores como a confidencialidade, integridade e disponibilidade reforçando a sua credibilidade, pelo que toda a informação é sigilosamente utilizada na organização, nunca chegando ao conhecimento público, reduzindo então os riscos de fraude, perda de informação e quebra de confidencialidade [6].

ISO 20000

Esta norma permite que os sistemas de TI prestem serviços de qualidade aceitáveis aos seus clientes (internos ou externos), focando-se nas questões relacionadas com a gestão das TI através de uma abordagem *helpdesk* onde os problemas são classificados, ajudando assim a identificar questões atuais ou interligações. A norma também avalia a capacidade do sistema, os níveis de gestão necessários em termos de mudanças nos sistemas, orçamentação financeira, controlo e distribuição de software.

A ISO 20000 apresenta duas partes principais: a ISO 20000-1 e a ISO 20000-2 [7].

eIDAS

A ACIN sendo um provedor de serviços privados, encontra-se numa *Trusted List* (TL) da Comissão Europeia [8] sendo capaz de prestar serviços qualificados e avançados (não-qualificados) de assinatura

eletrónica e selo eletrónico, mas também outros serviços como selos temporais (*timestamp*) e certificados de SSL conforme a figura 2. Por cumprir todos os standards europeus, é capaz de identificar digitalmente os utilizadores de toda a Europa por meio da assinatura eletrónica, garantindo o seu reconhecimento jurídico e a interoperabilidade transfronteiriça [9].

Figura 2 – Serviços do eIDAS.

1.1.2. Plataformas

Nesta secção vamos apresentar os serviços com integração utilizados na reformulação da Global Trusted Sign (GTS), nomeadamente o iGEST e a PayPay que são geridas e mantidas pela ACIN.

GTS

A plataforma Global Trusted Sign encarrega-se de vender quatro serviços de confiança, nomeadamente os selos temporais, os certificados qualificados de assinatura eletrónica, os certificados qualificados de selos eletrónicos e os certificados de autenticação de servidores na web. O objetivo desta plataforma é tornar-se líder na venda de certificados no mercado a nível nacional e internacional, mas também, dar apoio à crescente evolução do uso de certificados nas plataformas da ACIN, tais como iGEST, iLink e AcinGov [10].

O iGEST e a PayPay são duas soluções que já estavam integradas no antigo projeto GTS e foram novamente utilizadas para este novo projeto.

iGEST

O iGEST é uma plataforma de faturação que permite gerir o negócio de forma fácil e rápida, diretamente de um computador, smartphone, tablet e terminais POS [11].

PayPay

A PayPay ou PAYPAYUE, é uma instituição de pagamentos autorizada e registada no Banco de Portugal sob o nº 8710 que permite facilitar ao máximo a forma de como as empresas emitem e recebem os pagamentos dos seus clientes, sendo possível as seguintes formas de pagamento: por cartão de crédito ou débito, multibanco (através de uma referência) e MB WAY [12].

1.2. Problema

Com este trabalho pretende-se melhorar o atual projeto que beneficiará de maior rapidez e clareza no código concebido, tendo como objetivo principal, rejuvenescer o portal de vendas com uma API que simplifique todo o processo existente, melhorando a experiência do utilizador e futuras manutenções por parte da equipa de desenvolvimento.

Outro fator impulsionador desta mudança a referir será aumentar as vendas face aos concorrentes nacionais de entidades certificadoras, sendo a Multicert e a DigitalSign [8], bem como concorrentes internacionais, tais como a GMO GlobalSign Inc., GoDaddy Group, Cognate, Inc., Verisign Inc., Gemalto NV, Comodo Group Inc., Signix Inc., Ascertia, Secured Signing Ltd., Docusign Inc., Identrust Inc., Kofax Ltd. e Entrust Datacard Corp [2].

1.3. Solução proposta

Uma vez devido à política de boas práticas de programação no contexto do trabalho para o *backend*, ficar aprovado apenas o desenvolvido utilizando a *framework* de Laravel, todo o projeto antigo desenvolvido na *framework* CodeIgniter (CI) necessitava da alteração para esta *framework*. Ainda para o *frontend*, ficou aprovado apenas a utilização do React, em detrimento do HyperText Markup Language (HTML), Cascading Style Sheets (CSS), JavaScript (JS) e jQuery de forma a desenvolver soluções mais elegantes e simples, levando menos tempo nas pequenas alterações de layout e evitando o uso excessivo de media queries para manter a aplicação responsiva nos diversos dispositivos móveis.

Por fim, para evitar atrasos no desenvolvimento do produto e falhas de sincronização dos ficheiros para um servidor, em caso de por exemplo, falhas de internet, utilizou-se o Docker possibilitando também o teste de diferentes ferramentas em ambiente de desenvolvimento na própria máquina de trabalho.

Tendo em vista a abordagem proposta com este projeto, espera-se obter os seguintes resultados:

- Implementação de um método formal de Engenharia de software contínuo e incremental;
- Melhorar a acessibilidade/usabilidade da carteira de clientes existentes face à plataforma anterior;
- Utilização da *framework* Laravel por estar em constante evolução (versões mais recentes em relação ao CI) e com maior uso pela comunidade de programação;
- Utilização da ferramenta React para *frontend*, desenvolvendo assim uma *cross-platform* mais moderna com destino a web e a mobile, facilmente extensível, implementado também o conceito *single page application* (SPA);
- Aplicação de testes unitários evitando o lançamento de versões com falhas para os clientes existentes e evitar os testes manuais;
- Utilização do Docker em ambiente de desenvolvimento;
- Documentação da API e *cookbook* das *queries*/*mutations*/*types*/*inputs* de GraphQL;
- Tratamento de erros em ambiente de produção com códigos específicos;
- Utilização de testes de carga para monitorização de tráfego de utilizadores.

1.4. Planeamento do projeto

Para atingir os objetivos acima identificados, definiu-se que o plano de trabalho englobaria as seguintes etapas: modelação, design, implementação, testes e finalização.

Etapa de modelação - Nesta fase, deverá ser descrito os objetivos dos módulos a implementar, feito um levantamento dos requisitos funcionais e a documentação para a construção da aplicação. Esta etapa não requer validação do cliente, sendo realizado o planeamento dos módulos com uma análise profunda entre o gestor de projeto, o gestor de produto e os membros da equipa de desenvolvimento.

Etapa de design - Nesta etapa, a equipa de desenvolvimento expõe as necessidades do sistema à equipa do departamento de design, para que deste modo seja realizada a construção da aplicação no Adobe XD com base num esboço de alto nível. No decorrer desta fase, foram realizadas reuniões entre departamentos, podendo assim requisitar outros novos layouts conforme o necessário e discutir problemas existentes de acessibilidade ou possíveis dificuldades com os layouts distribuídos.

Etapa de implementação - É nesta etapa que se realiza o desenvolvimento do produto. Caso ocorra a adição/remoção de requisitos, conforme as necessidades, é alterada a versão da fase de modelação/planeamento e por conseguinte, realizando novas versões da etapa de design, paralelamente para que mais tarde seja concebido o código do sistema.

Etapa de testes - Esta etapa sucedesse de imediato após a conclusão do módulo implementado/desenvolvido e destina-se ao processo de testes da aplicação, através dos testes unitários e de browser, verificando o correto funcionamento do sistema.

Etapa de finalização - Esta etapa destina-se à documentação dos pedidos à API.

1.5. Organização do documento

Este documento tem início com o capítulo de introdução, onde define-se a aplicação a ser implementada, abordando a evolução dos certificados até aos dias de hoje. De seguida descrevendo a entidade responsável pelo projeto e ainda os objetivos de trabalho para a plataforma a ser desenvolvida, abordando os problemas existentes e soluções propostas do mesmo. Por fim, toda a organização deste documento. Este capítulo está estruturado da seguinte forma: A secção 1 descreve a história dos certificados. A secção 1.1 clarifica a empresa detentora da aplicação e algumas das suas soluções e a certificação. A secção 1.2 ilustra os problemas do trabalho atual, a secção 1.3 apresenta os objetivos e âmbito da solução proposta, a secção 1.4 apresenta o planeamento da solução. Finalmente, a secção 1.5 descreve toda a organização do documento.

O presente documento está dividido em oito capítulos. Sendo a estrutura da tese durante o desenvolvimento do trabalho a seguinte:

- No primeiro capítulo introdução, são apresentados os tópicos da contextualização, a motivação, os objetivos pretendidos e o planeamento para o desenvolvimento deste projeto de mestrado;
- No segundo capítulo revisão de literatura é apresentado o estado de arte, contextualizando os projetos com bases idênticas ou com objetivos semelhantes, abordando a temática base deste projeto, evidenciando também as caraterísticas específicas do projeto a desenvolver e a demonstração das várias tecnologias utilizadas bem como as ferramentas;
- No terceiro capítulo solução é descrito a visão geral da solução, especificado a arquitetura do sistema, as decisões de desenvolvimento e apresentando os diagramas arquiteturais do sistema da entidade certificadora GTS;
- No quarto capítulo, no método é descrito os requisitos, a modelação da base de dados, a validação dos dados e as especificações dos testes;

- No quinto capítulo implementação, é apresentado o desenvolvimento da plataforma;
- No sexto capítulo os resultados e avaliação são demostrados os testes e os resultados que foram obtidos;
- No sétimo capítulo a discussão, são expostos os objetivos e os problemas encontrados na plataforma original;
- Por fim, no oitavo capítulo, na conclusão é apresentado as conclusões da solução, entre as lições aprendidas e o proposto de trabalho futuro.

2. Revisão de literatura

Neste capítulo é realizado um estudo de soluções comerciais identificando os aspetos positivos e negativos de cada concorrente da plataforma GTS, na secção 2.1. Na secção 2.2 são abordadas as temáticas fundamentais de criptografia, regulamentação e autenticação aplicadas neste projeto. Em seguida, é descrito na secção 2.3 todo o ciclo de desenvolvimento. Na secção 2.4 é descrito a linguagem de programação utilizada na empresa ACIN, a *framework* selecionada através da apresentação de vários estudos, a estrutura de base de dados e a aplicação para o consumo de dados. Para finalizar, é apresentado, o software utilizado durante a concretização do projeto em questão.

2.1. Estudo de soluções comerciais

Neste ponto, identificou-se os concorrentes diretos que têm como consumidor final os mesmos segmentos de mercado que a GTS. Neste estudo de mercado procedeu-se à elaboração de uma análise crítica das principais soluções comerciais que permitem a assinatura de documentos e comercializam certificados digitais ou selos temporais conforme as necessidades do público-alvo, destacando-se assim várias aplicações informatizadas da aplicação a ser preparada. Esta análise é vital na orientação do desenvolvimento da aplicação proposta, com vista na recolha dos pontos fortes e fracos, mas também na identificação dos erros mais relevantes de cada plataforma.

2.1.1. Nexus Group

Os produtos considerados concorrentes designam-se de Nexus Certificate Manager, Nexus Personal Mobile e Nexus IDentity Connector [13].

Alguns pontos a focar nos produtos desta empresa são o Nexus Personal Mobile que possui 3,3 estrelas no google play, com um total de 108 comentários [14]. O Nexus Certificate Manager encontra-se em processo de recertificação para *Evaluation Assurance level* 4+ (EAL4 +) de acordo com o padrão internacional, onde segue o padrão internacional para segurança de computadores *Common Criteria for Information Technology Security Evaluation* (CC), e permite a emissão de certificados para vários domínios do Windows a partir de um único sistema CA, sendo que esta CA pode ser implementada num servidor Windows ou RedHat e oferece suporte a protocolos *Simple Certificate Enrollment Protocol* (SCEP), *Certificate Management over CMS* (CMC), *Certificate Management Protocol* (CMP) e *Enrollment over Secure Transport* (EST) necessários para a certificação [15]. O Nexus IDentity Connector (IDC) facilita a gestão de vários sistemas de identidade por meio de uma interface sendo simples de utilizar o software para a criação de cartões (PAS Card) [16].

2.1.2. Cryptomathic

A Cryptomatthic, possui vários produtos concorrentes ao GTS, a EMV CA e Crypto Key Management System (CKMS) [18].

Nos produtos desta empresa é importante reter que o EMV CA suporta várias CAs, oferece uma solução de *two factor authentication* (2FA), fornece o balanceamento de carga entre servidores e módulos de segurança de hardware (HSM), possui um f*ailover* e *clustering* para disponibilidade e escalabilidade [20]. Já o Signer encontra-se em conformidade com os padrões europeus *European Telecommunications Standards Institute* (ETSI), cumprindo o regulamento eIDAS e a lei suíça ZertES, oferece serviços de Assinaturas Eletrónica Avançadas (AdES) e Assinaturas Eletrónica Qualificadas (QES), possui o mecanismo *What You See Is What You Sign* (WYSIWYS) [21]. O CKMS fornece auditoria à prova de violação de dados e registos de uso para prova de conformidade, onde são suportados diferentes tipos e formatos de chave,

por exemplo, Atalla Key Block, BASE24, IBM CCA, MC OBKM, PKCS # 8, TR-31 e detém *Bring Your Own Key* (BYOK) e *Manage Your Own Key* (MYOK) na *cloud* [22].

2.1.3. ASCERTIA

O produto considerado concorrente da GTS intitula-se por ADSS Server [23]. Este produto apresenta um *graphical user interface* (GUI) intuitivo, suporta formatos de assinatura tais como PAdES, CAdES, XAdES, XML DigSig, PKCS # 7, CMS, PKCS # 1 e S/MIME, é possível assinar documentos em lote/massa e as chaves de assinatura podem ser mantidas localmente num dispositivo de criação de assinatura segura (SSCD), tal como um *smartcard*, *tokens* USB seguros ou arquivos de software [24].

2.1.4. Multicert

Esta empresa é detentora do produto chamado de Public Key Infrastructures Solution, que se baseia numa EJBCA *open-source* CA da PrimeKey, personalizável, fornecendo selos temporais, certificados de assinatura eletrónica e selos eletrónicos, no entanto, não cobre serviços eIDAS [25].

2.1.5. DigitalSign

Esta empresa é especializada em segurança eletrónica, através de certificados de servidor, certificados de email e de assinatura, encriptação de ficheiros, *workflow* com certificação digital, fatura eletrónica, assinatura de documentos eletrónicos, software PKI, hardware criptográfico, deteção de fraude online, *anti-phishing*, autenticação digital e consultoria em segurança eletrónica [26].

2.1.6. Considerações finais

Em suma, o Grupo ACIN pretende com este desenvolvimento, que a GTS se torne líder na venda de produtos eIDAS, comercializando assim serviços de confiança como: assinaturas eletrónicas, selos eletrónicos, selos temporais, autenticação de websites, por um preço mais acessível a nível do mercado nacional e internacional, onde é possível a customização de certificados para diversos perfis de clientes, sejam estes entidades singulares, entidades coletivas, de tipologia profissional, com o certificado do tipo qualificado ou avançado, numa plataforma de vendas moderna, intuitiva*, user-friendly* e rápida.

E este estudo tornou-se uma mais-valia, podendo assim tomar em consideração as vantagens e desvantagens de cada aplicação reforçando a GTS, melhorando os resultados do seu negócio e a ampliar a sua visão de mercado.

De seguida, através da comparação de preços apenas pelo exemplo dos selos temporais entre os rivais nacionais, respetivamente a Multicert e a DigitalSign referidas no ponto 2.1.4 e 2.1.5 respetivamente, versus os preços praticados pela GTS, conclui-se que este estudo é uma mais-valia, podendo assim ter em consideração decisões e estratégias no mercado.

Pela observação da Tabela I, é calculado o preço de cada selo temporal com validade de 1 ano nos diferentes pacotes do produto. Na GTS é de 0.35€ por 100/200 selos, 0.25€ por 600 selos, 0.15€ por 5000 selos, 5€ por selo à medida (customizado), e de 0.11€ por 5000 selos com o pacote de selos ilimitados, na Multicert é de 0.67€ por 50 selos, 0.46€ por 100 selos, e de 0.43€ por 500 selos, na DigitalSign é de 0.44€ por 200 selos e de 0.45€ por 750 selos.

Selos Temporais [a]		
Global Trusted Sign	Multicert	DigitalSign
100 Selos - **35€ + IVA**	50 Selos – **33,83€ + IVA**	200 Selos - **89€ + IVA**
200 Selos - **70€ + IVA**	100 Selos – **46,13€ + IVA**	750 Selos - **315€ + IVA**
600 Selos - **150€ + IVA**	500 Selos – **215,25€ + IVA**	
5000 Selos - **750€ + IVA**		
Selos à medida - **5€ + IVA**		
Selos Ilimitados - **564€ + IVA**		

a. Com validade de 1 ano

Tabela I – Comparação de quantidade e preço de selos temporais.

Mediante a comparação de quantidade/preço dos selos temporais referida na Tabela I, de apenas 1 ano de validade, conseguimos determinar que a GTS apresenta uma maior diversidade de quantidades do produto, e a única que dispõe da customização do produto (à medida) e ilimitada, mas também ao melhor preço por cada selo.

E ainda, sobre a comparação dos rivais nacionais, apesar de ambas apresentarem as traduções para a língua materna do português e o inglês, a GTS além das referidas anteriormente, possui também a tradução para o espanhol, levando assim à expansão e alcance com maior facilidade potenciais clientes do exterior.

2.2. Criptografia e Autenticação

Neste capítulo serão abordadas as considerações de segurança e regulamentação duma CA, como é o caso da empresa ACIN, conceitos gerais da criptografia utilizada e apreciações da tecnologia empregue na autenticação.

2.2.1. Autoridade de certificação (CA)

A criptografia é uma ferramenta de segurança, cujo elemento básico é a confidencialidade pela capacidade de limitar o acesso à informação apenas para as entidades autorizadas, garantindo a integridade contra falsificação, a autenticação do remetente e do destinatário, assegurando o não repúdio confirmando que a mensagem foi realmente originada pelo alegado remetente, não tendo sido forjada nem alterada na transmissão.

Uma CA é uma entidade confiável responsável pela emissão de certificados digitais. Estes certificados contêm informações de identificação, como o nome de uma pessoa ou empresa, bem como a sua chave pública. As informações presentes no certificado são verificadas pela CA, garantindo a autenticidade do certificado e, portanto, da chave pública contida nele [27]. A Autoridade de Certificação (CA) utiliza a sua chave privada para assinar digitalmente os certificados que emite e os utilizadores utilizam a chave pública da CA para verificar a autenticidade do certificado emitido. A chave pública de uma CA é amplamente divulgada e é usada para verificar a assinatura digital presente no certificado.

A técnica da criptografia pode ser dividida em duas categorias principais: criptografia simétrica e criptografia assimétrica. A criptografia simétrica utiliza uma chave secreta compartilhada entre as partes para encriptar e desencriptar as informações. A segurança deste tipo de criptografia depende da proteção da chave secreta entre as partes. A criptografia assimétrica utiliza um par de chaves, uma chave pública e uma chave privada, em que, a chave pública é disponibilizada publicamente sendo utilizada para criptografar as informações, enquanto a chave privada é mantida em segredo pelo proprietário do certificado, sendo utilizada para desencriptar as informações. Como cada indivíduo possui a sua própria chave privada, a criptografia de chave pública permite a integridade e autenticidade dos dados. Portanto, as assinaturas eletrónicas funcionam com a criptografia assimétrica.

2.2.2. Regulamentação eIDAS

A Comissão Europeia criou uma regulamentação (eIDAS 910/2014/EC) única e padronizada, com o intuito de qualquer cidadão da União Europeia (UE) com uma assinatura eletrónica e entidades com selos eletrónicos da sua identidade, possuírem uma estrutura jurídica e de valor legal aceitável no mercado digital no seu país e nos países membros UE.

Na plataforma do eIDAS, é estabelecido um conjunto alargado de serviços de confiança tais como: assinaturas eletrónicas, selos eletrónicos, selos temporais, serviços de envio registado eletrónico, autenticação de websites, onde é feito o reconhecimento mútuo transfronteiriço dos meios de identificação eletrónica (eID) através dos serviços de assinaturas que cumprem os requisitos mais rigorosos de conformidade para verificar a identidade dos signatários e a autenticidade dos documentos assinados. Estas assinaturas podem seguir os seguintes padrões [28], [29].

- XAdES (XML *Advanced Electronic Signature*)
- CAdES (CMS *Advanced Electronic Signature*)
- PAdES (PDF *Advanced Electronic Signature*)
- ASiC (*Associated Signature Container*)

2.2.3. Padrão OAuth 2.0

O *Open Authentication* (OAuth) é uma estrutura de autorização [30]. No OAuth 2.0 está descrito na *Request for Comments* (RFC) 6749 que um serviço de terceiros obtenha acesso limitado a um serviço HTTP, seja em nome de um proprietário de recurso, coordenando uma interação aprovada entre o proprietário do recurso e o serviço HTTP, ou em seu próprio nome. Este padrão substitui o protocolo OAuth 1.0 descrito na RFC 5849 [31].

Podemos observar pela figura 3 um cenário da utilização do fluxo de a*uthorization_code* do RFC. O utilizador acede ao site XYZ que inicia o fluxo de *redirect* (passo 1) para o servidor de autorização do serviço ABC onde inclui no pedido o *cliente_id*, *scope*, *response_type* e o *redirect_uri* (passo (a)).

O servidor de autorização do serviço ABC autentica o utilizador e estabelece caso se este utilizador concede ou nega a solicitação (passo 2).

Ao assumir que o utilizador concede o acesso ao servidor de autorização do serviço ABC (passo 3), este direciona o utilizador de volta para o site XYZ usando o *redirect_uri* fornecido no passo (a). O pedido GET (passo (b)) inclui um *code* e qualquer estado local fornecido pelo site XYZ no passo (a).

Em seguida, esse *code* é trocado por um *token* de acesso quando o site XYZ solicita ao *endpoint* de *token* do servidor de serviço ABC por POST (passo (c)). Ao efetuar o pedido com sucesso, o site XYZ autentica com o servidor de autorização do serviço ABC.

Por fim, o servidor de autorização autentica o site XYZ que utiliza o *token* de acesso para consultar o servidor de recursos do serviço ABC e obter informações sem fazer uma nova autorização durante o tempo de vida do *token*.

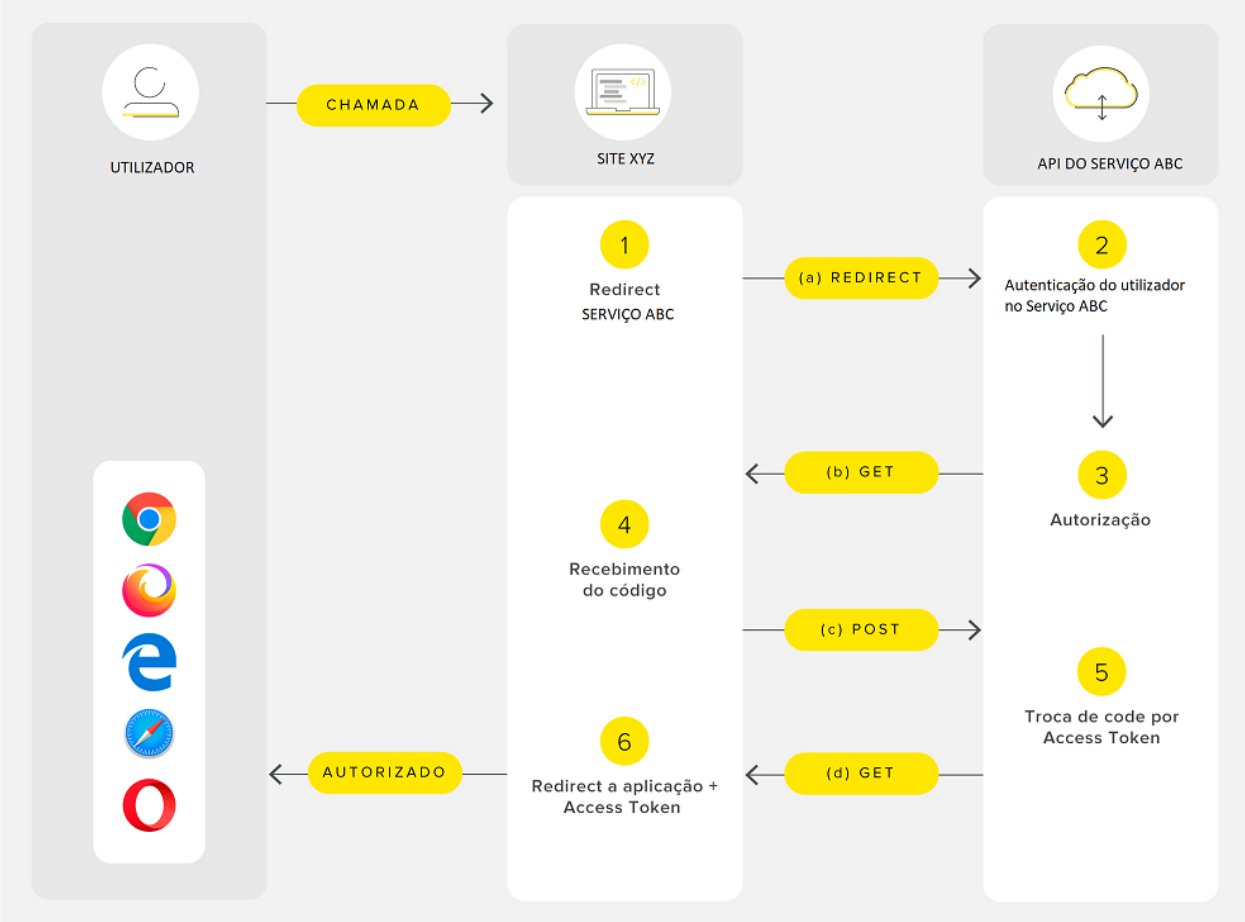

Figura 3 - Authorization Code Grant Flow.

2.2.4. Conclusão

Em síntese, a segurança é um requisito importante na solução GTS, pelo que este prestador de serviços utiliza a criptografia assimétrica nas assinaturas eletrónicas. Atualmente são utilizadas as abordagens de XAdES e PAdES para assinatura de ficheiros do tipo *Extensible Markup Language* (XML) e *Portable Document Format* (PDF) respetivamente, conforme a regulamentação eIDAS válida para qualquer país da UE. Por fim, utilizou-se o protocolo OAuth 2.0 seguindo o fluxo do código de autorização na implementação da autenticação à API.

2.3. Considerações de Desenvolvimento

Neste ponto, identificou-se a técnica adotada para o desenvolvimento, seguindo um método contínuo e incremental utilizado pela empresa ACIN, além disso, será discutido o método ágil de desenvolvimento e a ferramenta utilizada para a gestão de tarefas e acompanhamento do projeto.

2.3.1. Processo da aplicação

O processo de conceção e desenvolvimento de software utilizado pela empresa ACIN tem como base, um modelo iterativo e incremental, descrito na figura 4, apresentando as fases de planeamento, requisitos, análise, design, implementação, testes e avaliação.

A ideia básica por detrás da utilização da abordagem iterativa é desenvolver um sistema de software por fases do ciclo e incrementalmente, que nos permite obter progressivamente versões do produto de software em desenvolvimento cada vez mais refinado.

Figura 4 - Modelo Iterativo e Incremental.

A técnica do método ágil de desenvolvimento utilizado na empresa ACIN, é o *kanban* que permite gestão de tarefas por meio dum *backlog*, que após a utilização de cartões com números é estipulado o tempo necessário para realização de cada tarefa, e posteriormente, cada tarefa segue um fluxo de trabalho (*to do*, *in progress* e *done*). A ferramenta que permitiu a gestão de tarefas e acompanhamento do projeto designa-se de Jira.

Por fim, a escolha do modelo iterativo e do método ágil *kanban* deve-se à natureza do projeto e das exigências de gestão de projetos na ACIN. Neste projeto, foram realizadas melhorias e otimização em processos existentes, exigindo assim, uma abordagem flexível que permitisse a adaptação às mudanças nos requisitos e prioridades do projeto. Em contrapartida, o modelo *waterfall*, por ser uma abordagem sequencial e inflexível, não se adequava às mudanças nos requisitos após o início do projeto.

2.4. Revisão de tecnologia

Na construção de um projeto, é necessário determinar a linguagem a utilizar, o conjunto de ferramentas que sejam mais adequadas para a construção e consumo da implementação com o objetivo de solucionar os problemas atuais. Através de diversas pesquisas na empresa ACIN foi possível determinar o conjunto de tecnologias que facilitam o desenvolvimento, o armazenamento e consumo de dados eficiente, a execução do *deployment* do projeto de uma forma fácil e rápida, e entre outros motivos.

Nas secções seguintes é descrita sucintamente as alternativas e o porquê da escolha das várias ferramentas pretendidas para o projeto, conforme os requisitos. Primeiramente, a ordem de discussão

da *framework* para *backend*, de seguida é exposto a tecnologia a utilizar na base de dados e por fim é abordado qual a linguagem a ser utilizada para o consumo de dados. Em relação ao *frontend*, não será apresentado nenhuma pesquisa, uma vez que o meu trabalho desenvolvido dentro da equipa GTS neste novo projeto é apenas com o *backend*. Assim sendo, ficou definido pela administração juntamente com o departamento de desenvolvimento aplicacional (DDA) a utilização apenas do React, face ao que estava anteriormente implementado em HTML, CSS, jQuery e JS. Para finalizar, é apresentado o software de terceiros que permitiram executar e desenvolver o código, realizar operações na base de dados, executar pedidos à API e efetuar os testes.

2.4.1. Linguagem de programação

O *Hypertext Preprocessor* (PHP) é uma linguagem de programação *open-source*, muito utilizada no desenvolvimento web e pode ser introduzido no código de HTML.

A sua sintaxe requer *tags* de início e fim (`<?php` e `?>`), com o objetivo de executar o bloco de código no lado do servidor, apresentando depois, os resultados no HTML, levando ao desenvolvimento de páginas web dinâmicas [32].

A linguagem de PHP é extremamente simples e oferece muitos recursos avançados, sendo uma ferramenta de ampla utilização, rápida de desenvolver e fácil de aprender. A 26 de novembro de 2020 foi lançado a versão 8.0 do PHP, no entanto, a linguagem empregue na empresa ACIN e no projeto GTS, numa fase inicial, foi a versão PHP 7.4 conforme a figura 5, mantendo-se a mesma versão que o projeto antigo da GTS.

```
root@c6dd4af63e3c:/var/www/html# php -i
phpinfo()
PHP Version => 7.4.9
```

Figura 5 – Verão do PHP.

2.4.2. Frameworks de Backend

Uma *framework* é um conjunto de bibliotecas ou componentes que são usados para criar uma base onde a aplicação será construída. Esta estrutura de código oferece soluções aos problemas recorrentes com uma abordagem genérica, e disponibiliza funcionalidades específicas ao programador ajudando no desenvolvimento rápido e seguro de aplicações. A utilização de uma *framework* é importante por inúmeras razões, tais como [35]:

- Facilidade, rapidez e segurança no processo de desenvolvimento;
- Reutilização, organização e uma melhor manutenção do código;
- Escalabilidade das aplicações.

Segundo o oneclickitconsultancy [36] numa publicação, este considera o Django, Spring e Express como as *frameworks* mais robustas em novembro de 2020.

2.4.2.1. Django

O Django é uma estrutura de aplicação *web* em linguagem Python de alto nível que incentiva o desenvolvimento rápido. É uma solução gratuita e de código aberto. Esta *framework*, foi projetada para

ajudar os desenvolvedores, por ser extremamente fácil de aprender, possuir métodos e ferramentas para a construção de um site seguro, escalável e flexível [37].

O Disqus, o Mozilla, o National Geographic, o Pinterest ou o Instagram, são algumas das aplicações que utilizam esta *framework* [38].

Segundo o LeadBloging [39], estas são as vantagens e desvantagens do Django.

Vantagens:

- É extremamente fácil de aprender, mesmo parecendo difícil inicialmente;
- É bastante popular entre os desenvolvedores e possui uma comunidade enorme;
- É escalável e flexível;
- Possui documentação detalhada;
- Segue o padrão *Model View Template* (MVT).

Desvantagens:

- É monolítico, criando obstáculo na personalização de aplicações;
- Necessita de conhecimento e experiência sobre a linguagem de *phyton*.

2.4.2.2. Spring

O Spring é *open-source* para a plataforma Java podendo ser desenvolvido para seguir as seguintes abordagens de software: micro serviços, sistemas reativos, sistemas orientados a eventos, *cloud* possibilitando o conjunto de serviços necessários para que as aplicações sejam executadas na *cloud* [40], aplicações web, *serverless* e o processamento em lote (*batch*).

Esta ferramenta segue uma estrutura escalável, flexível que leva à melhoria na produtividade e redução de erros, baseando-se nos padrões de inversão de controle (IoC) e injeção de dependência (DI). O Spring possui diversos componentes, um dos quais é o Spring MVC, que permite implementar aplicações web seguindo o padrão de desenho *Model-View-Controller* (MVC) [41].

Sendo as suas vantagens e desvantagens as seguintes:

Vantagens:

- Simples e fácil de aprender;
- Possibilidade de integração com outras *frameworks*;
- Suporta *template engine* com *FreeMarker*, *JavaServer Pages* (JSP), *Velocity*;
- Integração com *hibernate validator* (módulo de validações);
- Utiliza anotações (ex: `@Configuration`, `@Scope`, `@Profile`).

Desvantagens:

- Dependências configuradas em XML;
- Dificuldade no tratamento de erros HTTP;
- Não possui mecanismos para lidar com o *cross-site scripting* (XSS).

2.4.2.3. Express

O Express é uma estrutura de aplicação web em Node.js, escrita em JavaScript sendo *open-source*. Esta possui uma estrutura mínima e flexível, fornecendo um conjunto de ferramentas para *web* e *mobile*.

Dispõe também de métodos para o protocolo *Hypertext Transfer Protocol* (HTTP), *caching* e *midlewares* de modo a preparar uma API robusta com uma ótima performance. É especificamente projetada para a construção de aplicações da *web* híbridos (aplicação nativa nos telemóveis e aplicação *web*) de única página ou de várias páginas [42].

Esta ferramenta utiliza o *Node Package Manager* (npm) para a gestão de pacotes, e permite a utilização de outras linguagens que compilam para JavaScript, tais como o TypeScript, CoffeeScript, ClojureScript, Scala, LiveScript, etc [43]. Na medida em que, o Express tem as seguintes vantagens e desvantagens:

Vantagens:

- Diversos módulos disponibilizados pelo npm;
- *Framework fullstack* (linguagem de *frontend* e *backend*) mais produtiva;
- *Open-source*.

Desvantagens:

- Problemas de perda de memória, pelas incorretas alocações da memória necessária;
- Não há benefício de tarefas ligadas ao CPU, por ter ambiente de execução em Node.js;
- Pouca informação das mensagens de erro da própria *framework*.

2.4.2.4. Laravel

Segundo o oneclickitconsultancy, o Laravel está em 5º lugar na lista das *frameworks* de *backend* [36]. Esta ferramenta surgiu em 2011 e até aos dias de hoje possui uma sintaxe expressiva, capaz de fornecer uma estrutura e um ponto de partida para quem pretende construir aplicações *web*. Os seus recursos poderosos são: a injeção de dependência, a camada de abstração da base de dados, filas e *jobs* agendados, testes unitários e de integração entre muitos mais. Ainda dispõe de uma vasta biblioteca documentada com vídeos tutoriais denominados de Laracasts. Esta estrutura *fullstack*, contém um ecossistema capaz de servir como *backend* de uma API e como *frontend* de uma aplicação em JavaScript de página única ou aplicação móvel [44].

De acordo com o LeadBloging [39] as suas vantagens e desvantagens são:

Vantagens:

- Usa o *template blade* e torna lógica na *view* o mais simples possível;
- Integração com a biblioteca SwitfMailer, para o envio de e-mails.
- É PHP com programação orientada a objetos e baseado em MVC.
- Utiliza o Eloquent que é um *object-relational mapper* (ORM);
- Um suporte ativo e em contínuo desenvolvimento.

Desvantagens:

- Não possui ferramentas integradas na necessidade de integrações de terceiros;
- É necessário experiência na linguagem PHP;
- Complexidade da estrutura face a outras *frameworks*.

2.4.2.5. Conclusão

Através do estudo efetuado e conforme a administração juntamente com o DDA da empresa ACIN, o desenvolvimento aplicacional dos projetos existentes e os futuros projetos, serão implementados

adotando a *framework* Laravel, visto que é uma ferramenta muito popular, que dispõe de uma grande comunidade, possui diversos módulos úteis tornando mais rápido o desenvolvimento dos programadores, onde apresenta uma curva de aprendizagem simples devido ao material de apoio (documentação), suporte e ajudas da comunidade pela *web*, oferecendo também diversas medidas de segurança.

Outro dos motivos pela qual levou à escolha e aprovação do Laravel como ferramenta de *backend* no desenvolvimento e reformulação desta plataforma, foi devido à falta de suporte da *framework* atual CodeIgniter (CI). Desta forma, o CI não será mais utilizado, pelo que a próxima versão de lançamento da GTS, a v.3.0.0 e subsequentes versões serão realizadas com a *framework* Laravel até surgir outra ferramenta que ofereça melhores benefícios e vantagens.

Em relação às outras *frameworks* em estudo foram excluídas de imediato nomeadamente, o Django devido ao facto de possuir sintaxe da linguagem de Python. A *framework* Spring foi excluída por ser desenvolvido para a plataforma Java que engloba a linguagem de Java, aumentando assim drasticamente a curva de aprendizagem tanto de Python ou Java, dado que na empresa ACIN apenas são recrutados programadores com conhecimentos na linguagem de PHP. No que diz respeito ao Express, este foi descartado por ser uma ferramenta menos utilizada na comunidade, com menos recursos e funcionalidades disponíveis ao dispor do programador, requerendo ainda a aprendizagem do JavaScript e Node.js.

Em suma, depois de todas as comparações e análises, ficou contemplado que a *framework* Laravel é a mais indicada para o desenvolvimento na linguagem PHP e será importante a médio e longo prazo no desenvolvimento rápido, consistente e escalável.

2.4.3. Base de dados

Uma base de dados é uma coleção organizada de informações estruturadas ou dados, armazenados eletronicamente num computador. Os dados e o sistemas de gestão de base de dados (SGBD), juntamente com as aplicações que estão associados a estes, são denominados de sistema de base de dados, geralmente abreviado para apenas base de dados (BD) [45].

Um dos maiores desafios no sistema de base de dados é a decisão entre uma estrutura de dados relacional do tipo *Structured Query Language* (SQL) de uma estrutura não relacional do tipo *Not only Structured Query Language* (NoSQL).

As bases de dados relacionais SQL são bases de dados que exigem um esquema mais rígido e pré-definido para dados estruturados, ou seja, todos os registos têm que ter os mesmos atributos e seguir a mesma estrutura. Por outro lado, as bases de dados não relacionais NoSQL são bases de dados que permitem ter um esquema mais dinâmico e flexível para dados não estruturados [46].

Os requisitos básicos que descrevem qualquer sistema distribuído imposto na empresa ACIN, rege-se pelo relacionamento e integridade dos dados, de modo a cumprir com o Regulamento Geral sobre a Proteção de Dados (RGPD) e procuras/inserções na BD com rapidez para o desempenho. Posto isto, é necessário garantir com atomicidade, consistência, isolamento, durabilidade (ACID). Este conceito define que uma transação deve ocorrer com sucesso ou nenhuma alteração é confirmada (atomicidade), os dados somente serão confirmados se verificar-se o cumprimento de todas as regras em vigor na base de dados (consistência), as transações não são afetadas por outras transações (isolamento) e os dados são armazenados de forma segura contra erros (durabilidade), sendo apenas possível no SQL. Por

consequência, no NoSQL só é possível alcançar no máximo duas de três garantias: consistência, disponibilidade e tolerância à partição (CAP), onde os clientes veem os mesmos dados ao mesmo tempo (consistência), qualquer cliente que fizer uma solicitação de dados obterá uma resposta imediata (disponibilidade) e qualquer quebra de comunicações por perda ou lentidão existe um *cluster* para continuar a comunicação (tolerância de partição).

Seguindo então o modelo de bases de dados relacionais e utilizando o Google Trends [47], com as palavras Oracle, MySQL, PostgreSQL, MongoDB e Microsoft SQL Server, durante o ano 2020 a nível mundial, pelas pesquisas da *web* e pela observação da figura 6, o Oracle é o termo mais procurado, estando posicionado no 1º lugar, de seguida em 2º lugar encontra-se o MySQL e no 3º lugar o PostgreSQL, sendo então, o Microsoft SQL Server o menos procurado dos sistema de base de dados SQL.

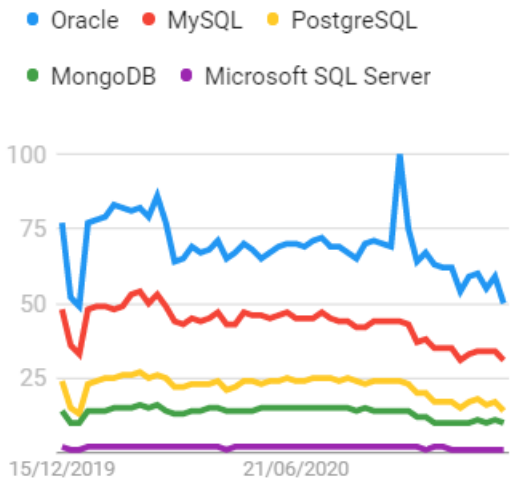

Figura 6 - Google Trends em sistemas de BD SQL.

Com base nas quantidades de pesquisas por palavras, foi realizado a seguinte comparação entre a base de dados Oracle, MySQL e PostgreSQL e as suas vantagens e desvantagens respetivamente [48].

2.4.3.1. Oracle

A base de dados Oracle é uma coleção de dados tratada como uma unidade, capaz de gerir grandes quantidades de dados e permite que vários utilizadores acedam às mesmas informações. Esta ferramenta oferece testes e permite que grandes quantidades de dados sejam movidas rapidamente. A Oracle funciona nas plataformas Windows e Linux, oferecendo um suporte de virtualização.

Esta base de dados proporciona desempenho, escalabilidade, confiabilidade e segurança, tanto em local quanto numa *cloud* pública ou privada. E ainda oferece suporte e correção de bugs, no entanto, o uso desta base de dados implica um custo/preço associado [48].

Sendo as vantagens e desvantagens, as seguintes.

Vantagens:

- Facilidade de configuração e poder de processamento;
- Numerosos recursos e personalizações possíveis;

- Ferramentas gráficas intuitivas.

Desvantagens:

- Complexidade e curva de aprendizagem maior;
- Para um *database administrator* (DBA) sem experiência, a configuração é complicada pela quantidade de detalhes que devem ser considerados;
- Custos/preços associados.

2.4.3.2. MySQL

O MySQL da Oracle Corporation permite implementar na *cloud*, possui um conjunto amplo de recursos avançados, ferramentas de gestão e suporte técnico para atingir os mais altos níveis de escalabilidade, segurança, confiabilidade e tempo de atividade [49]. Para a proteção de dados existem mecanismos de segurança a contra-ataques externos, violações de dados e mau uso interno de informações, atendendo aos requisitos de conformidade regulamentar e da indústria, incluindo RGPD, *Payment Card Industry - Data Security Standard* (PCI DSS) e *Health Insurance Portability and Accountability Act* (HIPPA) com os mais altos níveis de segurança sendo otimizado para *Oracle Cloud Infrastructure* (OCI) [50].

As vantagens e desvantagens são:

Vantagens:

- É *open-source* e gratuito;
- Sintaxe simples e de baixa complexidade;
- Possui uma grande comunidade de utilizadores.

Desvantagens:

- Por ser mantida pela proprietária Oracle, possui restrições;
- Para ter acesso ao suporte, na sua opção gratuita, é necessário pagar;
- Possui desempenho aquém do esperado na execução de algumas tarefas com muitos dados (*backups*, *querys*).

2.4.3.3. PostgreSQL

O PostgreSQL é um sistema *open-source* e gratuito, com muitos anos de desenvolvimento ativo, confiabilidade, robustez de recursos e alto desempenho [51].

Esta BD pode ser executada nos principais sistemas operativos, oferecendo compatibilidade com o ACID. Sendo uma base de dados objeto-relacional (BDOR) é um modelo orientado a objetos, escrito em linguagem de programação C [52].

Algumas das suas vantagens e desvantagens são:

Vantagens:

- É escalável porque vem com muitos recursos para os desenvolvedores construírem aplicações e oferece soluções para os administradores protegerem a integridade dos dados, sendo ainda possível a criação de ambientes tolerantes a falhas.

- É altamente extensível. Podendo ser definido tipos de dados, funções personalizadas e até mesmo escrever código com diferentes linguagens de programação sem recompilar a base de dados.
- É gratuito e *open-source*.

Desvantagens:

- Instalação não é simples.
- Devido à extensão de recursos disponíveis, possui uma maior complexidade.
- Utiliza muita memória e o desempenho torna-se mais lento em inserção/procura de dados.

2.4.3.4. Conclusão

Resumidamente, um requisito a manter seria o relacionamento dos dados, integridade dos dados e desempenho, garantindo a atomicidade, a consistência, o isolamento e durabilidade sendo possível numa estrutura de SQL, mantendo-se assim o sistema de MySQL para a BD desta API, apesar do NoSQL oferecer flexibilidade no seu esquema e mais velocidade de consulta.

2.4.4. Consumo de dados de uma API

As requisições do *frontend* para o *backend* realizam as operações de CRUD (*create*, *read*, *update* e *delete*), isto é, permite criar, ler, atualizar e apagar dados. A linguagem de GraphQL foi utilizado como formato de comunicação entre os dois ambientes, em alternativa ao *Representational State Transfer* (REST). Desta forma teve de ser considerado neste projeto qual seria o repositório de PHP que iria servir a estrutura do GraphQL na realização das comunicações.

2.4.4.1. GraphQL

O GraphQL é uma linguagem de consulta para APIs. Esta fornece uma descrição completa e compreensível dos dados na API, e concede aos consumidores desta linguagem a capacidade de pedir exatamente o necessário, tornando mais fácil evoluir APIs ao longo do tempo.

Na figura 7, podemos observar a estrutura de um pedido de GraphQL. Em primeiro lugar, descrevemos a *operation* (operação), podendo ser *query* (leitura de dados) ou *mutation* (criar, atualizar ou apagar dados), posteriormente é necessário o *operation name* (nome da operação) e de seguida a declaração das *variable definition* (variáveis da operação). Os dados são enviados no *argument* (argumentos), retornando no *nested fields* (campos) os dados pretendidos a receber desta operação.

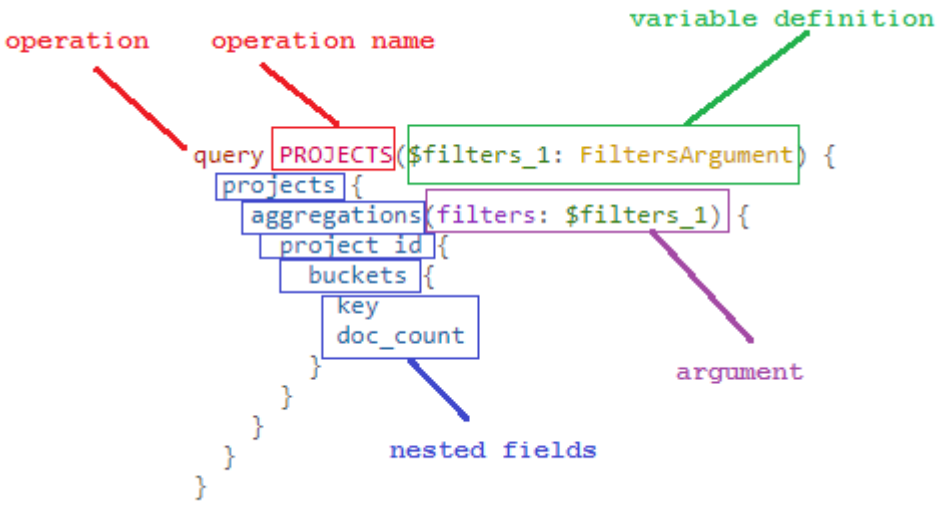

Figura 7 - Estrutura do GraphQL.

A utilização do GraphQL, faz com que as consultas sejam rápidos e estáveis porque são controlados os dados que se obtêm. Ao fazer consultas, também é possível obter todos os dados que a aplicação precisa numa única solicitação. Deste modo, a estrutura é organizada em tipos e campos, que podem ser facilmente ser adicionados ou alterados, sem causar impactos nas consultas existentes, facilitando o crescimento da API. O Facebook, GitHub, Pinterest e Coursera, são alguns dos exemplos que usam esta linguagem [53].

Para utilizar esta linguagem no PHP, existem várias bibliotecas desenvolvidas que permitem servir um *endpoint* de GraphQL, das quais foram estudadas apenas as seguintes [54].

O Lighthouse e o Rebing, são ambos repositórios de PHP que integram-se bem com qualquer versão do projeto Laravel, sendo altamente personalizável, dando controlo total sobre os dados [55]. O Rebing oferece mais recursos e melhorias em relação ao seu pacote original da Folklore [56].

2.4.4.2. Conclusão

Em suma, para o consumo de dados usando a linguagem de GraphQL, ficou definido utilizar o Lighthouse da empresa NuWave, visto que este apresenta uma *code coverage* de 89% demonstrando que o código-fonte é executado em testes, não cobrindo apenas 11% das suas funcionalidades com testes, já o Rebing não usa testes unitários e é um projeto de uma pessoa em singular. Além disso, o Lighthouse é mais utilizado pela comunidade do GitHub, disponibilizando documentação pelo site oficial e dá suporte ativo no Stack Overflow e Slack.

2.4.5. Ferramentas de desenvolvimento

As escolhas de software de desenvolvimento de terceiros potenciam a rapidez no desenvolvimento de aplicações de software, para além de simplificar todo este processo de desenvolvimento, dá auxílio para alcançar os objetivos do projeto de forma focada, acrescentando qualidade e segurança a fim de que o sistema funcione corretamente. Neste projeto, foram utilizados os seguintes softwares durante o percurso de desenvolvimento da aplicação.

2.4.5.1. Docker

Esta aplicação desenvolvida na linguagem de programação Go é utilizada para a configuração de *containers*, sendo que, cabe ao *DevOps* a criação das imagens (*containers*) para a virtualização de um determinado serviço.

Tradicionalmente, uma máquina virtual (VM) trabalha com o sistema operativo emulando determinados sistemas, e havendo múltiplas instâncias de VM ocorre o consumo de mais recursos e espaço da máquina. Em alternativa, o Docker é mais leve e portável porque utiliza o mesmo container para compartilhar as suas bibliotecas e binários na replicação de ambientes isolados, sendo mais rápido e fiável pois permite reduzir o tempo de *deploy* utilizando as mesmas configurações evitando erros ou imprevistos conforme a seguinte figura 8 [33].

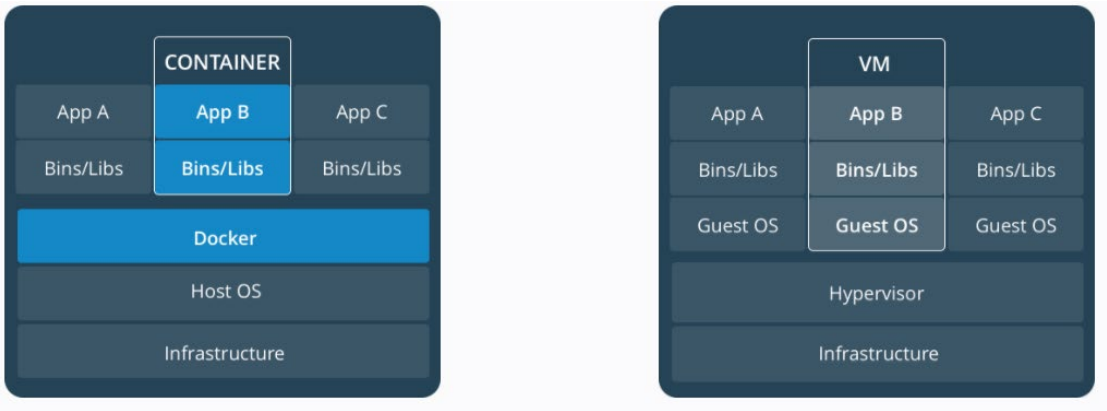

Figura 8 - Suporte e arquitetura do sistema operativo.

O produto Docker Desktop é executado no *Windows Subsystem for Linux* (WSL) 2 e permite que as distribuições do Linux sejam instaladas e acedidas no *Windows,* consumindo apenas os recursos de *central processing unit* (CPU) e memória de forma dinâmica e conforme as necessidades [34].

2.4.5.2. Docker Hub

O Docker Hub é uma ferramenta do Docker que permite pesquisar um conjunto de *containers* de imagens de fornecedores de tecnologia, programas de *open-souce* e de imagens criadas pela comunidade [57]. Esta ferramenta permite aos desenvolvedores partilharem com a sua equipa o mesmo ambiente de implementação para cada projeto, fornecendo estabilidade durante o desenvolvimento.

2.4.5.3. HeidiSQL

O HeidiSQL é um *open-souce,* intuitivo e simples de utilizar, desenvolvido pela Ansgar em 2002 [58]. Este permite visualizar e editar dados do sistema de base de dados MySQL.

2.4.5.4. Visual Studio Code

O Visual Studio Code é um editor de código com suporte para operações de desenvolvimento, depuração de código, controlo de versões, entre muitos mais. O programa é *open-souce*, disponível para os vários sistemas operativos (MacOS, Linux e Windows). Algumas das suas funcionalidades são o IntelliSense que realça a sintaxe e fornece *autocomplete* do código, incluí também o comentário de cada código e o *trace* aos módulos importados. Outra funcionalidade é o *run and debug*, com pontos de interrupção ao longo do código para suporte de depuração. Por fim, ainda tem o controlo de versão Git integrado e a possibilidade de instalar extensões, customizando ao gosto do desenvolvedor [59]. Para

auxiliar o desenvolvimento em equipa neste ambiente de desenvolvimento integrado (IDE), foram ainda utilizadas algumas extensões.

2.4.5.5. SoapUI

O SoapUI possui duas ferramentas, uma *pro* e uma *open-source* [60]. A ferramenta utilizada neste desenvolvimento foi a de *open-source*, com recursos mais limitados em relação à versão *pro*, mas que apresenta uma interface gráfica simples e intuitiva, permitindo a verificação de *Simple Object Access Protocol* (SOAP), REST e *Web Services Description Language* (WSDL). Esta plataforma permitiu testar a comunicação entre a API iGEST para a geração do PDF de faturas.

2.4.5.6. Firecamp

O Firecamp é um software minimalista para realizar funções relacionadas com APIs, desde a possibilidade de realizar pedidos HTTP, de GraphQL, SocketIO e comunicações de WebSocket [61]. Este programa foi utilizado para executar pedidos de *querys* e *mutations* de GraphQL.

2.4.5.7. Postman

O Postman é um software *open-source*, tal como o SoapUI e o Firecamp, que permite criar e executar pedidos REST, SOAP e de GraphQL, porém é mais robusto que as outras ferramentas acima descritas [62]. Este programa foi utilizado para executar os pedidos de HTTP.

2.4.5.8. Url webhook.site

Neste domínio obtém-se instantaneamente um URL para receber *webhooks*, sem a necessidade de um servidor da Web. Ainda apresenta um domínio de e-mail para receber e-mails [63]. Sendo que os pedidos recebidos atravessam um servidor protegido por uma *firewall* ou numa sub-rede. Esta aplicação foi utilizada para receber e testar o *webhook* da API PayPay após o pagamento.

3. Solução

Neste capítulo, é apresentada uma visão detalhada e abrangente de alto nível do proposto para o projeto, abordado os pontos de justificação, finalidade e descrição do sistema, bem como os principais *stakeholders* envolvidos, estimativas de investimento em tempo e recursos financeiros, restrições, critérios de aceitação e riscos do seu desenvolvimento na secção 3.1. Na secção 3.2 é explicada a arquitetura estrutural e a utilização do GraphQL no projeto. Na secção 3.3 são descritos os aspetos do processo de desenvolvimento da plataforma tais como: modularidade, tolerância a falhas, segurança, desempenho, escalabilidade e portabilidade. Na secção 3.3.1 é referenciada a instalação e distribuição da plataforma web, a presença na loja de aplicações dos smartphones e o software da aplicação que a GTS fornece. É ainda abordado o perfil do utilizador alvo da plataforma, onde todos aqueles que desejam adquirir os serviços de segurança da GTS na secção 3.4, assim como é apresentada a estrutura de comunicação entre o cliente, o servidor GraphQL, a API e as integrações na secção 3.5.

3.1. Descrição de alto nível da solução

Em relação às necessidades da solução GTS a ser desenvolvida, foram considerados diversos pontos.

Neste caso particular porque possuímos um nível elevado de complexidade no projeto atual em CI. Além disto, certos formulários de produtos são demasiado pesados e lentos para o cliente, não podendo ser comprado múltiplos produtos do mesmo tipo ou de diversos tipos. Uma nova implementação da plataforma em Laravel, será uma melhor solução de desenvolvimento, tornando o código mais legível melhorando assim a sua manutenção, o seu desenvolvimento e integrações futuras, onde a aplicação de um carrinho de compras permite gerar mais lucros para a organização.

A finalidade de implementar esta *major* servirá para aumentar os lucros, diminuir o tempo de manutenção e respostas dos pedidos. Sendo então o objetivo desta solução a construção da API orientada ao cliente, a fim de que seja rápida, simples, intuitiva face à atual implementação atual da GTS, com a modelação dos módulos de negócios por versões e com documentação dos pedidos para as integrações.

Este sistema (descrição) possibilita a aquisição de diversos produtos em forma de carrinho de compras, gerindo os certificados adquiridos pelo utilizador para a assinatura de documentos PDF/XML na própria plataforma ou por software instalado. Futuramente prevê-se que seja desenvolvido o *backoffice* do projeto, para que os administradores administrem os pedidos dos utilizadores e os *developers* possam executar tarefas mais complexas. Para além disso, pretende-se que no futuro seja desenvolvida a API orientada ao parceiro de forma que um cliente singular ou uma entidade, revenda os produtos GTS.

Os *stakeholders* são os utilizadores da nossa carteira de clientes, um gestor de projeto juntamente com cinco pessoas na equipa de desenvolvimento, dos quais três em *backend* (incluí o gestor de projeto) e duas em *frontend*, uma pessoa como *compliance* e uma pessoa como designer.

No que diz respeito às estimativas de tempo e custo, a modelação iniciou-se a 16 de março de 2020, o desenvolvimento começou a 8 de julho 2020 e a previsão de conclusão são nove meses. Os custos deste projeto, equivalem ao pagamento dos colaboradores que o compõem.

As restrições da solução passam pelo projeto estar em conformidade com a lei portuguesa e a alocação de desenvolvedores do *backend/frontend* de forma a manter a plataforma antiga ou outras plataformas diferentes da GTS.

Em relação aos critérios de aceitação, a plataforma deve albergar todas as necessidades do cliente e as suas respetivas funções testadas após a migração dos dados antigos. Além de que deve passar na auditoria sem não conformidades.

Em relação às restrições, deve-se ter em conta a disponibilidade de recursos humanos.

Por fim, existem riscos associados, nomeadamente a falta de conhecimento sobre as tecnologias utilizadas e consequentemente atraso nas datas de lançamento.

3.2. Arquitetura

Neste tópico é apresentada em detalhe a arquitetura estrutural da solução preparada.

A *framework* Laravel implementa o padrão arquitetural MVC, onde num 1º passo o utilizador faz um pedido para uma rota, de seguida em 2º o diretório *routes* invoca o método do *controller*, em 3º o *controller* interage com a m*odel*, no 4º passo a *model* procura/insere/atualiza os dados à base de dados, em 5º o *controller* invoca a *view* que fornece os dados para o navegador do utilizador no passo 6º. Segue a figura 9 que ilustra todo este processo.

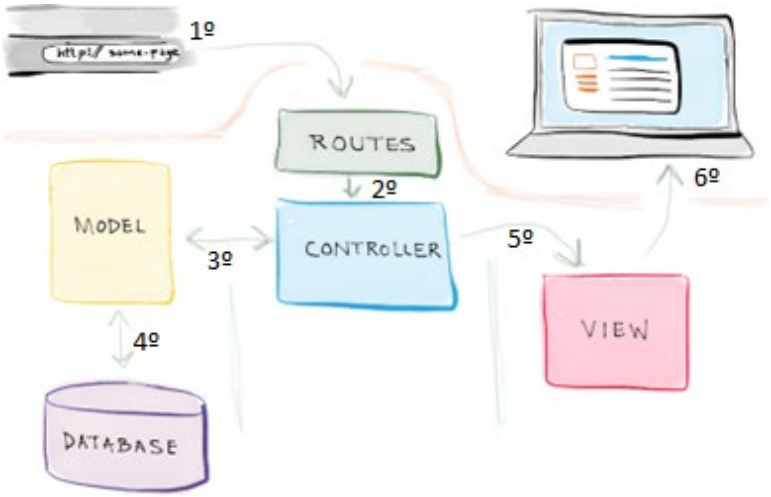

Figura 9 – Padrão de arquitetura implementado pelo Laravel.

No entanto, este modelo arquitetural apresenta falhas de desempenho, conforme o demonstrado na figura 10 pela *view* em *JavaScript Object Notation* (JSON) onde são necessários três pedidos no *routes* para receber informação dos *users*, *posts* e *followers*. Assim sendo, a utilização do GraphQL permite que para o mesmo *endpoint* possam ser solicitado todos os dados necessárias tal como representado na figura 11. Conforme representado pelo hambúrguer na figura 12, verifica-se que uma das vantagens de utilização do GraphQL é receber apenas a informação necessária sem consumir mais recursos ao cliente ou ao serviço.

Figura 10 - Arquiterura RESTul com três endpoints para receber dados.

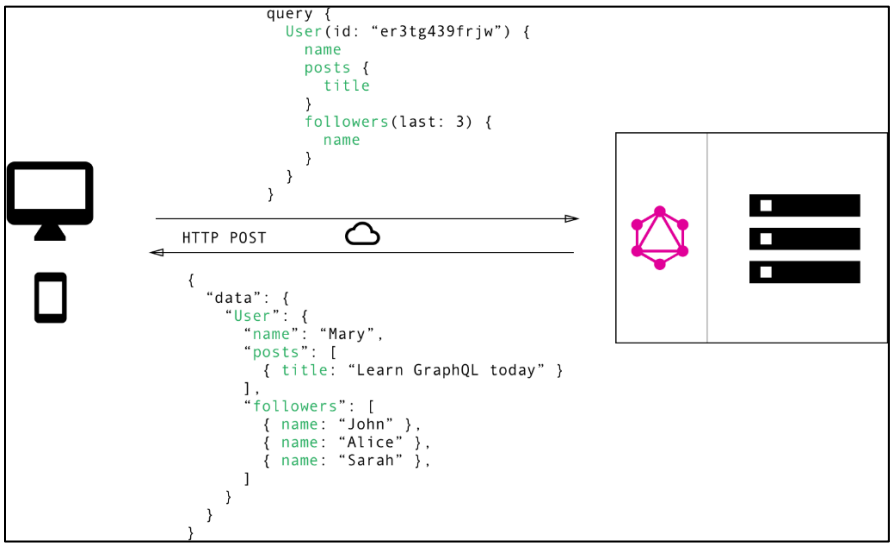

Figura 11 - Uso do GraphQL recolhendo os dados necessários em um pedido.

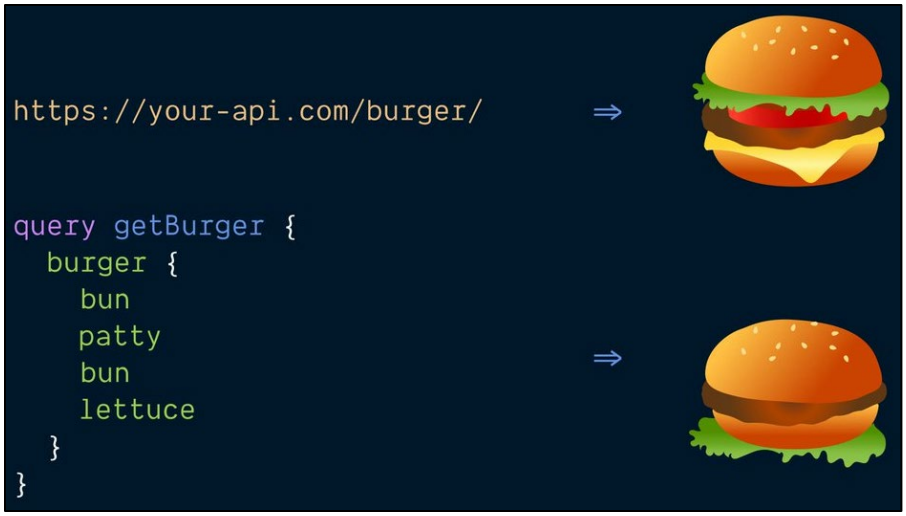

Figura 12 - Uso do GraphQL recolhendo os dados apenas pedidos.

Em termos de representação, o GraphQL é apresentado no formato de JSON [64] onde qualquer tipo de pedido (*web service*, HTTP ou GraphQL) está associado a um código de estado (ou *status code*), composto por três números, em que o primeiro algarismo do código de status define a classe de resposta, e os dois últimos dígitos não possuem qualquer tipo de classificação ou categorização. Pela ordem de classes, os 1xx informam que a solicitação foi recebida e o processo continuado, os 2xx informam que a solicitação foi recebida, compreendida e aceite com sucesso, os 3xx significam que ações adicionais precisam ser tomadas para concluir a solicitação, os 4xx informam que a solicitação contém sintaxe incorreta e os 5xx por sua vez, informam que o servidor falhou em atender uma solicitação aparentemente válida [65]. O GraphQL apresenta apenas o código 200 [65].

Um cliente utiliza o GraphQL *server* para trocar informações com uma API. Em suma, a arquitetura completa final da nova plataforma está representada na figura 13.

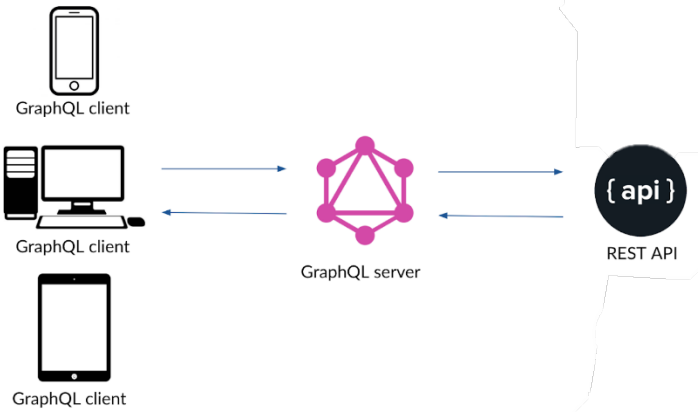

Figura 13 - Arquitetura final do sistema.

3.3. Processo de desenvolvimento

No desenvolvimento deste sistema, foram seguidas as considerações de design:

O sistema final necessita de ter componentes independentes e bem definidos, o que reduz a complexidade, levando a que as futuras manutenções sejam mais simples. Os módulos implementados

têm de ser testados isoladamente antes de serem integrados com os restantes módulos para que o funcionamento seja o desejado (modularidade).

O sistema precisa de ser robusto, resistente e capaz de se recuperar após falhas de componentes, pois é um sistema de alta disponibilidade (tolerância a falhas). Sendo que no *Service Level Agreement* (SLA), o contrato entre o prestador de serviços e o cliente, definem os vários tipos e padrões de serviços a serem oferecidos, onde atualmente está definido para a indisponibilidade de 4 horas e a disponibilidade tem de ser de 99%.

Ainda é necessário que o sistema seja capaz de resistir a ataques do exterior (segurança), oferecendo tempos de resposta aceitáveis para os utilizadores (desempenho), podendo assim aumentar o número de clientes (escalabilidade), para os diversos dispositivos que este venha a utilizar (portabilidade).

3.3.1. Instalação e Distribuição

O website da GTS é utilizado para adquirir produtos, gerar os certificados e ainda assinar documentos PDF/XML.

A GTS está disponível nas lojas da Google Play [66], Apple Store [67] e AppGallery [68]. A aplicação móvel, GTS ID Mobile, permite que o cliente consiga autenticar-se no website sem a necessidade da palavra-passe (*Single Sign-On*), utilizando o *Face ID*, *Touch ID* ou código pin. Esta aplicação móvel permite ainda, validar a assinatura dos documentos no website de forma rápida e simples, seguindo a norma ISO29115 (*Entity Authentication Assurance Framework*). Os requisitos mínimos dos dispositivos para a Google Play é Android 5 ou um sistema operativo superior, para a Apple Store é iOS 8 ou um sistema operativo superior e para a AppGallery está disponível a todos os telemóveis da Huawei com Android 5 ou um sistema operativo superior [69].

Por fim, existe ainda um software instalável, designado de Virtual Smart Card (VSC), disponível para todos os sistemas operativos, onde é possível assinar no Adobe Acrobat ou em plataformas de contratação pública. Esta solução é útil para quando o cliente pretende assinar documentos acima de 30MB ou acima de 40 páginas.

3.4. Possíveis utilizadores

Consoante os produtos fornecidos pela GTS, é possível obter diferentes utilizadores na plataforma.

Os selos temporais, são direcionados a utilizadores (fornecedores) que usam plataformas de compras públicas em Portugal e na União Europeia.

Em relação aos certificados de *Transport Layer Security/Secure Sockets Layer Organization Validation/Extended Validation* (TLS/SLL EV/OV), estes são direcionados aos utilizadores que possuem um site.

No que diz respeito aos certificados de selos eletrónicos, estes são direcionados a entidades coletivas/organizações (pessoas coletivas), em que o selo eletrónico avançado equivale ao carimbo da empresa e um selo eletrónico qualificado equivale ao selo branco de um organismo público.

Os certificados de assinatura eletrónica, são destinados a utilizadores singulares (certificado digital qualificado singular e certificado digital avançado singular), a utilizadores coletivos (certificado digital qualificado coletivo e certificado digital avançado coletivo), a utilizadores profissionais singulares

(certificado digital qualificado profissional singular e certificado digital avançado profissional singular), a utilizadores profissionais coletivos (certificado digital qualificado profissional coletivo e certificado digital avançado profissional coletivo), sendo este utilizado para quando o cliente pretende assinar digitalmente documentos com valor probatório legal. Em termos do certificado digital qualificado profissional médico, este é destinado aos profissionais que utilizem a prescrição eletrónica da plataforma iMed da empresa ACIN.

O selo eletrónico para a faturação eletrónica, é destinado às empresas que possuem *Enterprise Resource Planning* (ERP) ou *Electronic Data Interchange* (EDI) e que pretendem que as faturas enviadas aos clientes por meios eletrónicos sejam assinadas com um certificado.

A solução proposta terá de considerar futuros serviços, entre os quais incluem os utilizadores do *backoffice* que são os colaboradores da empresa ACIN nomeadamente o desenvolvedor, administrador de registo, apoio, contabilidade, marketing etc. Sem esquecer dos clientes que vão beneficiar da API orientada para os pedidos de parceiro, onde estes utilizadores são pessoas singulares e coletivas que queiram revender os produtos GTS.

3.5. Diagrama arquitetural

Neste tópico é apresentada uma explicação em detalhe da estrutura da solução preparada, onde é demonstrado como é que os componentes se relacionam, até porque nos dias atuais, os sistemas estão ligados uns com os outros para partilha e troca de informações.

Este projeto foca-se na infraestrutura assinalada na figura 14, pelo que neste panorama visual, é possível observar-se um *user* independentemente dos diversos dispositivos, acedendo ao link seguro (*https*) do *client application* que por via GraphQL *server* executa pedidos à API do *backend*, do qual se comunica com a base de dados, memória interna, integrações e restantes módulos de negócio, devolvendo assim a informação ao *client application* que apresenta os dados no dispositivo do utilizador.

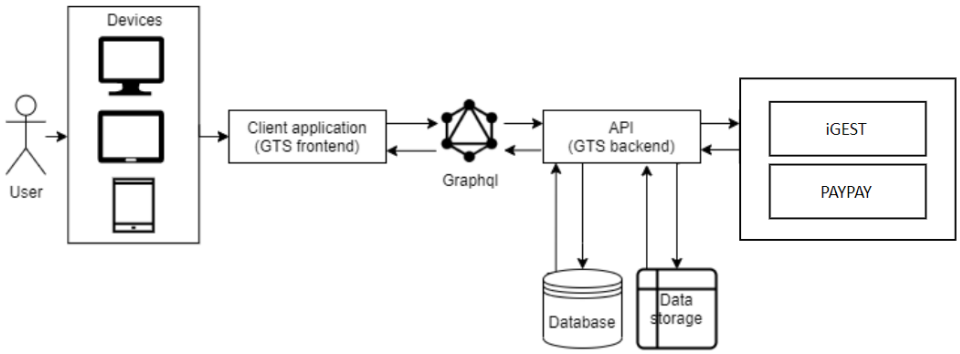

Figura 14 - Arquitetura do sistema.

Esta arquitetura do sistema foi desenhada para que um utilizador pelos diversos dispositivos, realize os vários pedidos a partir dum *frontend,* de forma a comunicar com a API da GTS pelo GraphQL. Os pedidos estabelecidos através de chamadas a uma API de *backend*, permitem que diferentes *client applications* (ex: integrações e *frontend* GTS) comuniquem, utilizando um conjunto de estruturas, operações, protocolos e ferramentas do API GTS, que contém toda a lógica de negócio deste novo sistema (v3.0.0) e permite a gestão com cada um dos módulos de negócio. Estes diversos módulos que compõem a API, permitem a partilha e troca da informação. No futuro, caso seja necessário alterar ou adicionar recursos

à lógica de negócio do sistema GTS, bastará apenas alterar diretamente na API GTS os módulos facilitando assim o seu desenvolvimento. A utilização de módulos de negócios possibilitam uma comunicação isolada entre cada recurso para criar, apagar, obter, ou atualizar dados da base de dados, gerir o armazenamento, ou até comunicar com integrações. Um módulo da API GTS possibilita assim uma comunicação entre sistemas de externos, como por exemplo às plataformas de faturação iGEST e pagamento por PayPay.

A integração com a plataforma iGEST, foi realizada usando a classe de *SoapClient* do PHP seguindo o modo de WSDL [70], e a integração com a plataforma do PayPay foi realizada seguindo as informações disponíveis no repositório do *Git*, utilizando o protocolo SOAP para as comunicações [71].

4. Método de Desenvolvimento

Tendo em conta a visão apresentada na secção anterior, este capítulo descreve o método de engenharia de software através de etapas de levantamento de requisitos do sistema. Em declarações de forma clara sobre o que este sistema deve ser capaz de fazer para satisfazer os requisitos do sistema e utilizadores, originando os requisitos legais, requisitos funcionais e de requisitos não funcionais na secção 4.1. Na secção 4.2, enumera-se os diversos atores que compõem o sistema, bem como são ilustradas as suas interações com o sistema no diagrama de casos de utilização. Por sua vez, na secção 4.3, é exposta a descrição das entidades, atributos e relações do sistema desenvolvido e a sua evolução conforme as obrigações emergentes da base de dados. Ainda na secção 4.4, é abordado as validações dos requisitos impostos no sistema. Por fim, na secção 4.5, são apresentados os diferentes testes aplicada ao sistema.

4.1. Levantamento de requisitos

Um requisito é uma necessidade que o cliente espera do sistema a ser construído. As especificações são normalmente utilizadas como informações fundamentais para a criação de um projeto, definindo as propriedades e as funções necessárias a serem consideradas [72].

Na fase de modelação, o levantamento de requisitos foi realizado sem a aprovação de um *stakeholder*, permitindo ao desenvolvedor, a identificação de funcionalidades e requisitos consoante as necessidades do sistema a desenvolver.

Através da realização de diversas reuniões com os membros do projeto, estas demonstraram ser bastante produtivas, não só para a identificação de problemas críticos já existentes, como também para a identificação de novas funcionalidades pretendidas, adicionando assim a sua modelação nos respetivos documentos.

Após a conclusão do processo, obtiveram-se os requisitos legais, requisitos funcionais e requisitos não funcionais respetivamente.

4.1.1. Requisitos Legais

Um prestador de serviço de confiança qualificado (QTSP) obedece a regras específicas e devem ser supervisionados por uma entidade supervisora nacional, que neste caso é o Gabinete Nacional de Segurança (GNS) [73]. Os prestadores qualificados de serviços de confiança para a emissão de certificados, devem reunir as condições técnicas nomeadamente o cumprimento de requisitos legais, assegurando a condição de integridade, confidencialidade, não-repudiação, e autenticação das informações do titular ou da entidade. Os requisitos a respeitar são os seguintes:

RL1. A GlobaltrustedSign está dotada de capital e meios financeiros adequados;
RL2. A GlobaltrustedSign dispõe de garantias de absoluta integridade e independência no exercício da atividade;
RL3. A GlobaltrustedSign dispõe de recursos técnicos e humanos, que satisfazem os padrões de segurança e de eficácia previstos na regulamentação;
RL4. A GlobaltrustedSign mantém o contrato de seguro válido para cobertura adequada da responsabilidade civil emergente da atividade de certificação;
RL5. A GlobaltrustedSign possui o estatuto de qualificado, publicado nas listas de confiança referidas no artigo 22º, nº 1. do Reg (UE) 910/2014;
RL6. A GlobaltrustedSign precisa de arquivar, em suporte informático, as videoconferências realizadas;

RL7. A empresa ACIN possui as normas ISO 9001, ISO 27001 e a ISO 20000 para operar no mercado;
RL8. A empresa ACIN cumpre com o RGPD;
RL9. A empresa ACIN possui certificação do GNS para operar.

4.1.2. Requisitos Funcionais

Os requisitos funcionais descrevem as operações e atividades que o sistema deve ser capaz de realizar para satisfazer de forma consistente as necessidades dos utilizadores. Durante o levantamento de requisitos foram encontrados requisitos de diferentes tipos, tais como requisitos do sistema, do utilizador e do administrador. Posto isto, obtiveram-se os seguintes requisitos funcionais presentes na lista:

RF1. O sistema deverá permitir o registo de conta;
RF2. O sistema deverá notificar no registo de conta o utilizador com o código OTP;
RF3. O utilizador deverá ser capaz de criar a conta com o código OTP;
RF4. O sistema deverá disponibilizar a subscrição à newsletter;
RF5. O sistema deverá permitir a recuperação da conta;
RF6. O sistema deverá ser capaz de identificar os certificados do utilizador;
RF7. O utilizador deverá ser capaz de criar pedidos de recuperação, recuperando e editando os detalhes desse pedido até decisão do administrador;
RF8. O sistema deverá permitir o upload de documentos do utilizador;
RF9. O sistema deverá notificar o utilizador para recuperação da conta;
RF10. O sistema deverá permitir a alteração da password;
RF11. O sistema deverá ser multilingue;
RF12. O sistema deverá gerir a sessão do utilizador;
RF13. O utilizador deverá ser capaz de criar o pedido de aquisição, recuperando, editando e apagando esse pedido;
RF14. O sistema deverá permitir o download de documentos do utilizador;
RF15. O utilizador deverá ser capaz de ordenar quaisquer pedidos de aquisição;
RF16. O utilizador deverá ser capaz de pesquisar quaisquer pedidos de aquisição;
RF17. O utilizador deverá ser capaz de adquirir diversos produtos, recuperando, editando e apagando esses produtos desse pedido de aquisição;
RF18. O utilizador deverá ser capaz de ordenar quaisquer produtos no pedido de aquisição;
RF19. O utilizador deverá ser capaz de pesquisar quaisquer produtos no pedido de aquisição;
RF20. O utilizador deverá ser capaz de criar dados de faturação, recuperando, editando e apagando esses dados de faturação;
RF21. O sistema deverá permitir vários métodos de pagamento;
RF22. O sistema deverá permitir mudar o atual método de pagamento do utilizador;
RF23. O sistema deverá ser capaz de calcular descontos utilizados pelo utilizador;
RF24. O sistema deverá mudar o estado do pagamento conforme as operações realizadas pelos utilizadores;
RF25. O sistema deve notificar o utilizador quando é efetuado um pagamento;
RF26. O sistema deverá apresentar a fatura;
RF27. O sistema deve ser capaz de enviar por e-mail, anexos de documentos PDF's para o destinatário definido;
RF28. O sistema deverá apresentar o pedido de review, sempre que é efetuado um pagamento com sucesso;
RF29. O utilizador deverá ser capaz de editar os dados da conta;
RF30. O sistema deverá permitir a gestão dos vários certificados;
RF31. O sistema deverá permitir o download dos vários certificados;

RF32. O sistema deve ser capaz de verificar o estado dos certificados do utilizador (revogado, expirado ou suspenso);
RF33. O administrador deverá gerir os pedidos do utilizador;
RF34. O sistema deve guardar e disponibilizar o histórico de eventos ocorridos no sistema;
RF35. O sistema deve ser capaz de comprovar o envio dos e-mails.

4.1.3. Requisitos Não Funcionais

Quanto aos requisitos não funcionais são o conjunto de atributos que o sistema deve ter, relacionados com o uso da aplicação em termos de desempenho, usabilidade, confiabilidade, segurança, disponibilidade, manutenção e tecnologias envolvidas. De seguida, são listados os requisitos não funcionais.

RNF1. O sistema deve ser rápido;
RNF2. O sistema deve ser fácil de usar, independentemente da experiência e conhecimentos do utilizador em termos de tecnologia;
RNF3. O sistema deve apresentar mensagens de erros informativos e orientados ao utilizador, caso ocorra introdução de dados incorretos;
RNF4. O sistema deve ser seguro;
RNF5. O sistema deve ter a documentação necessária disponível e atualizada;
RNF6. O sistema deve ser robusto;
RNF7. O sistema deve cumprir os aspetos legais de RGPD;
RNF8. As passwords dos utilizadores devem ser encriptadas na base de dados;
RNF9. O sistema deve bloquear o utilizador aquando demasiadas tentativas falhadas na palavra-passe;
RNF10. O sistema deve estar disponível;
RNF11. O sistema deve ter flexibilidade;
RNF12. A base de dados deve apoiar qualquer crescimento na recolha de dados;
RNF13. O *backend* do sistema para web deve ser desenvolvido com linguagem PHP e utilizar servidor de GraphQL.

4.2. Casos de utilização

Os diagramas de casos de utilização documentam uma sequência de eventos que ocorrem quando o ator (tipo de utilizador) interage com o sistema para realizar uma determinada tarefa/ação. A descrição das funcionalidades presentes no sistema permitem capturar os requisitos.

4.2.1. Atores

Os atores são os utilizadores ou entidades externas que interagem com o sistema em desenvolvimento. Neste caso em concreto, foram obtidos os seguintes atores:

- Utilizador sem sessão - É o utilizador que tem acesso às funcionalidades públicas do sistema;
- Utilizador com sessão - É um utilizador que tem acesso às funcionalidades privadas do sistema;
- Sistema iGEST - É a entidade externa responsável por gerar as faturas do utilizador;
- Sistema PayPay - É a entidade externa responsável por gerar os pagamentos do utilizador;
- Administrador - É o utilizador que tem acesso às funcionalidades administrativas do sistema.

4.2.2. Diagrama de casos de utilização

No âmbito deste sistema foram identificadas as várias ações para os requisitos, e conforme o modelo de casos de utilização apresentado na figura 15, onde por conseguinte, é exibido a interação entre

os cinco atores apresentados. O ator administrador, com a funcionalidade do grupo de permissões permite obter mais atores no sistema como: auditor, marketing ou customizado.

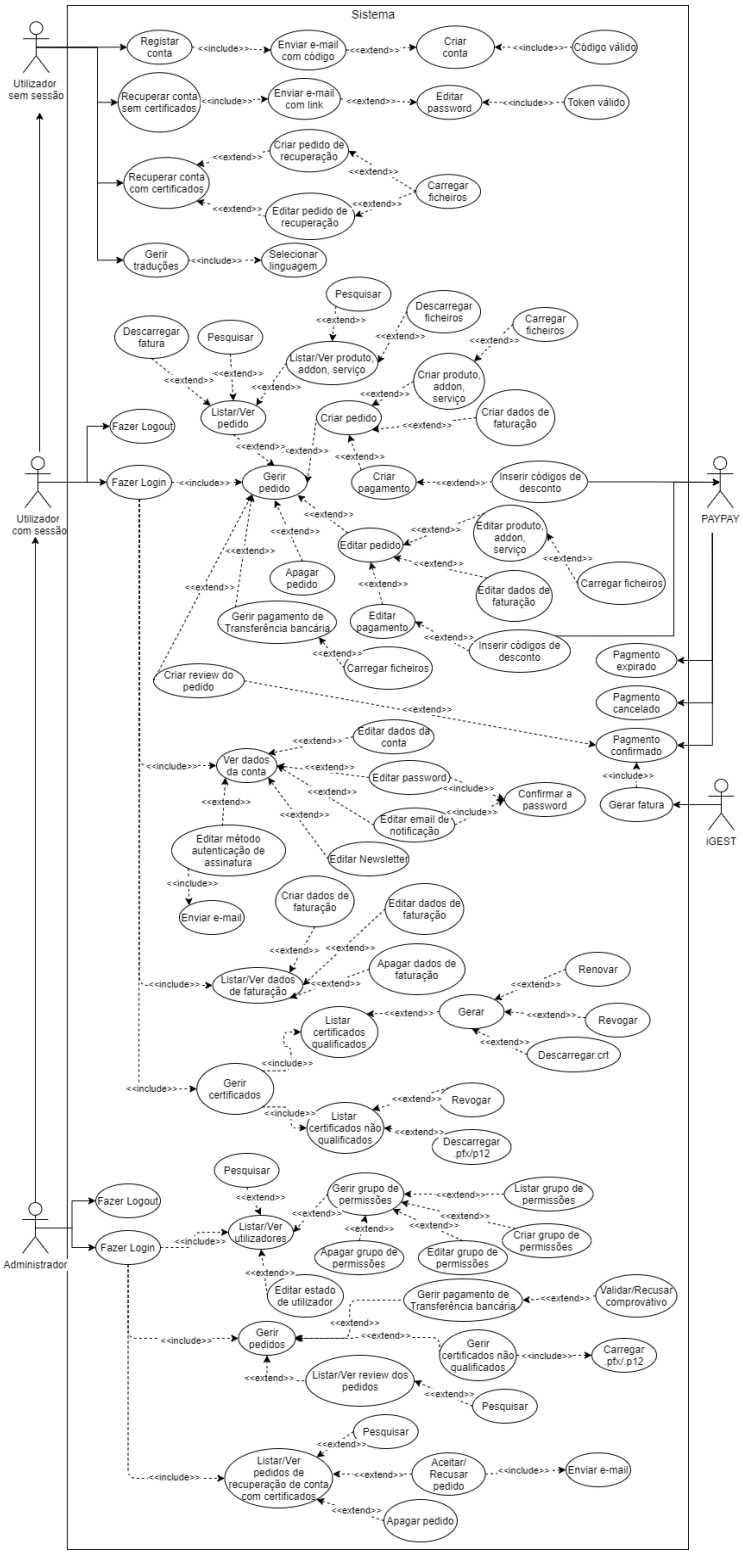

Figura 15 - Diagrama de casos de utilização.

4.3. Modelação

Neste tópico, são referidos os Modelos EER e as respetivas evoluções ao longo do tempo.

4.3.1. Base de dados

Um diagrama de entidade-relacionamento (ER) é uma técnica de modelação para ilustrar a representação das entidades de uma base de dados e as relações entre as restantes entidades. Os diagramas ER são compostos por [74]:

- Entidades: representam pessoas, objetos ou conceitos;
- Atributos: representam propriedades descritivas de uma entidade, e também são conhecidos por elementos de dados;
- Relações: representam as ligações entre as entidades.

Nos diagramas estão dispostos símbolos com os seguintes significados:

- Retângulos: Entidade;
- Informação dentro dos retângulos: Atributos;
- Setas: Relações.

Para o desenvolvimento deste sistema, foi necessário reconstruir a base de dados, de forma a torná-la mais simples e clara que a implementação do sistema anterior, mantendo os dados essenciais que o antigo sistema possuía. De início, utilizou-se a estrutura das tabelas antigas e posteriormente, reorganizou-se melhor a informação, para que o novo sistema de base de dados detivesse os dados armazenados de forma consistente e sem repetições. Mais tarde, foram acrescentados outras tabelas importantes, para o registo de dados.

Na figura 16 encontra-se a versão da base de dados necessária para manter todas as informações nas diversas tabelas deste projeto. Devido à sua dimensão e complexidade do modelo relacional da base de dados final do sistema, achou-se por bem colocar as evoluções que ocorreram durante o desenvolvimento nos diagramas no ANEXO A – BASE DE DADOS.

Figura 16 - Diagrama Entidade-Relação - 1ª versão.

O sistema foi dividido em módulos e submódulos, no diagrama entidade-relação, para uma melhor compreensão das diversas entidades, passando assim a explicar o propósito de cada entidade.

4.3.1.1. Módulo do utilizador

Este módulo permite a gestão dos utilizadores e a tradução da plataforma na API. É composto pelas seguintes entidades da Tabela II.

Nome	Descrição
users	Contém informação dos utilizadores registados na plataforma.
user_types	Contém informação sobre o tipo de utilizadores na plataforma.
languages	Contém informação das linguagens da plataforma.

Tabela II - Descrição das tabelas da base de dados - Módulo de utilizador.

Este módulo do utilizador incluí o submódulo de OAuth necessário para a autorização e a autenticação do utilizador na API. É composto pelas seguintes entidades da Tabela III.

Nome	Descrição
oauth_access_token	Contém informação dos *tokens* de acesso.

oauth_clients	Contém informação sobre o tipo de cliente (*password grant*, *personal access*, *client grant*, *public*).
oauth_auth_codes	Contém informação dos *codes* para depois obter o *token.*
oauth_personal_access_clients	Contém informação dos *tokens* de acesso pessoal.
oauth_refresh_tokens	Contém informação dos *tokens* pelo *refresh_token.*
oauth_temp_tokens	Contém informações dos *tokens* após os *redirects*.

Tabela III - Descrição das tabelas da base de dados - Módulo de Utilizador - Submódulo de oAuth.

Este módulo do utilizador ainda incluí o submódulo de recuperação de password necessário para a gestão dos pedidos de recuperação e a alteração de password, na API. É composto pelas seguintes entidades da Tabela IV.

Nome	Descrição
request_password_resets	Contém informação dos pedidos formais de recuperação de password.
request_password_reset_states	Contém informação dos estados dos pedidos formais de recuperação de password.
password_resets	Contém informação dos *tokens* para mudança da password.

Tabela IV - Descrição das tabelas da base de dados - Módulo de Utilizador – Submódulo da recuperação de password.

Este módulo do utilizador também incluí o submódulo de registo necessário para validação da criação de utilizadores, na API. É composto pela seguinte entidade da Tabela V.

Nome	Descrição
registration_codes	Contém informação dos códigos para registo de utilizador.

Tabela V - Descrição das tabelas da base de dados - Módulo de Utilizador – Submódulo de registo.

4.3.1.2. Módulo de gestão dos pedidos

Este módulo permite a gestão dos pedidos e produtos do sistema, em forma de carrinho de compras, pelo que um pedido possui diversos produtos, quantidades e formulários com diferentes dados conforme o tipo de produto. Na Tabela VI é apresentado uma breve descrição de cada uma das entidades, de forma a elucidar sobre a sua finalidade.

Nome	Descrição
requests	Contém informação dos pedidos de aquisição.
request_states	Contém informação dos estados do pedido de aquisição.
request_products	Contém informação dos produtos nos pedidos de aquisição.
request_product_has_addons	Contém informação dos extras associados ao produto.
request_product_has_files	Contém informação dos ficheiros associados ao produto.
request_file_types	Contém informação dos tipos de ficheiros no produto.
bundle_has_addons	Contém informação dos extras com os conjuntos de produtos realizados pelo administrador.
request_product_bundles	Contém informação do pedido com os conjuntos de produtos realizados pelo administrador.
certificate_states	Contém informação do estado do certificado do produto.
cert_conditions	Contém informação das condições para usar no certificado.
signature_data	Contém informações do pedido de produto de assinatura.
professional_order	Contém informação das ordens profissionais.

ssl_datas	Contém informação do pedido de produto de SSL.
ssl_data_has_domain	Contém informação dos domínios associados ao pedido de produto de SSL.
domains	Contém informação dos domínios.
custom_timestamps	Contém informação dos pedidos de produto de *timestamp* realizados pelo administrador.
custom_signature_packs	Contém informação do número de assinaturas que um produto tem realizados pelo administrador.
products	Contém informação dos produtos disponíveis na plataforma.
product_has_addons	Contém informação dos produtos disponíveis com os extras possíveis.
product_typologies	Contém informação das tipologias dos produtos disponíveis.
certificate_types	Contém informação do tipo de certificado.
product_conditions	Contém informação da condição do produto.
product_types	Contém informação dos tipos de produtos disponíveis.
product_description	Contém informação do produto para a zona pública.
product_modifiers	Contém informação das quantidades dos produtos.
product_modifier_types	Contém informação dos tipos de produtos com quantidades.
package_product_has_products	Contém informação do produto associados ao produto, fazendo um pacote de produtos.

Tabela VI - Descrição das tabelas da base de dados - Módulo de pedidos.

4.3.1.3. Módulo de faturação

Este módulo permite a gestão dos pagamentos e faturação dos pedidos realizados, na API. A Tabela VII apresentada uma breve descrição de cada uma das entidades, de forma a elucidar sobre a finalidade.

Nome	Descrição
billing_addresses	Contém informação dos endereços de faturação.
iva_exemptions	Contém informação sobre a isenção de IVA.
invoices	Contém informação do pedido de faturação.
districts	Contém informação dos distritos de Portugal.
counties	Contém informação dos concelhos de Portugal.
zip_codes	Contém informação das moradas e códigos postais de Portugal.
payments	Contém informação dos pagamentos.
payment_states	Contém informação dos estados de pagamento.
payment_methods	Contém informação dos métodos de pagamento (ref. multibanco, ATM, transferência bancária, MB WAY).
has_payments	Contém informação dos pedidos de pagamento.

Tabela VII - Descrição das tabelas da base de dados - Módulo de faturação.

4.3.1.4. Módulo de serviços

Este módulo permite a gestão de um pedido de serviço que é fornecido na API. Na Tabela VIII é apresentado uma breve descrição de cada uma das entidades, de forma a elucidar a sua finalidade.

Nome	Descrição
request_services	Contém informação dos pedidos de serviços.
services	Contém informação dos tipos de serviços.

service_states	Contém informação dos estados do pedido de serviço.

Tabela VIII - Descrição das tabelas da base de dados - Módulo de serviços.

4.3.1.5. Módulo de descontos

Este módulo permite a gestão dos códigos de desconto, utilizados durante o pagamento do pedido de aquisição. Na Tabela IX é apresentado uma breve descrição de cada uma das entidades, de forma a elucidar sobre a sua finalidade.

Nome	Descrição
discount_codes	Contém informação dos descontos.
has_discount	Contém informação dos pedidos com desconto.
discount_client_types	Contém informação dos tipos de clientes aptos a desconto (novo utilizador, cliente, todos).
discount_types	Contém informação dos tipos de desconto (percentagem, menos X preço, preço fixo).
discount_usage_types	Contém informação da utilização dos descontos (limitada, ilimitada).
discount_applies	Contém informação da aplicação dos descontos aos produtos.
discount_platforms	Contém informação das plataformas que utilizam os descontos.
discount_code_has_payments	Contém informação dos descontos aplicados aos pagamentos.

Tabela IX - Descrição das tabelas da base de dados - Módulo de descontos.

4.3.1.6. Módulo de gestão de certificados

Este módulo permite a gestão dos certificados dos utilizadores no sistema, mantendo registo dos detalhes públicos do certificado qualificado. Na Tabela X é apresentado uma breve descrição da entidade, de forma a elucidar sobre a sua finalidade.

Nome	Descrição
certificate_registry	Contém informação dos certificados gerados.

Tabela X - Descrição das tabelas da base de dados - Módulo de gestão de certificados.

4.3.1.7. Módulo de avaliação do cliente

Este módulo permite a gestão dos *reviews* dos pedidos efetuados pelos utilizadores, de forma a levantar prós e contras do sistema. Na Tabela XI é apresentado uma breve descrição de cada uma das entidades, de forma a elucidar sobre a sua finalidade.

Nome	Descrição
reviews	Contém informação das avaliações de clientes.
review_priorities	Contém informação das prioridades da avaliação na zona pública.
review_form_types	Contém informação dos tipos de formulário de avaliação.
review_form_has_questions	Contém informação das questões a aparecer nos formulários de avaliação.
review_has_awnsers	Contém informação das respostas ao formulário de avaliação.
review_questions	Contém informação das várias questões.

review_question_types	Contém informação dos tipos de questões (estrela, escolha múltipla, opinião pessoal).
review_question_has_review_question_default_values	Contém informação de questões com possíveis respostas.
review_question_default_values	Contém informação de possíveis respostas.

Tabela XI - Descrição das tabelas da base de dados - Módulo de avaliação do cliente.

4.3.1.8. Outros módulos

O conjunto destas restantes entidades, estão diretamente relacionados com os restantes módulos partilhando dados entre si para o bom funcionamento do sistema, bem como o armazenamento e prova do histórico. O submódulo de tabelas comuns é necessário para a identificação de ficheiros e países. Na Tabela XII é apresentado uma breve descrição de cada uma das entidades, de forma a elucidar sobre a sua finalidade.

Nome	Descrição
files	Contém informação dos ficheiros.
file_has_authorization	Contém informação dos *tokens* de autorização para download.
download_file_requests	Contém informação dos ficheiros para download.
countries	Contém informação dos países.

Tabela XII - Descrição das tabelas da base de dados - Tabelas comuns.

O submódulo de *logs* é necessário para preservar os pedidos efetuados e recebidos no sistema. Na Tabela XIII é apresentado uma breve descrição de cada uma das entidades, de forma a elucidar sobre a sua finalidade.

Nome	Descrição
notification_log	Contém informação das notificações de email.
notification_states	Contém informação dos estados de notificações de email.
invoice_log	Contém informação dos pedidos de faturação ao iGEST.
invoice_states	Contém informação dos estados dos pedidos de faturação ao iGEST.
payment_log	Contém informação dos pedidos de pagamento ao PayPay.
payment_log_types	Contém informação dos tipos de pedidos de pagamento ao PayPay.

Tabela XIII - Descrição das tabelas da base de dados - BD Logs.

O submódulo de tabelas do Laravel é necessário para controlar e gerir o próprio sistema. Na Tabela XIV é apresentado uma breve descrição de cada uma das entidades, de forma a elucidar sobre a sua finalidade.

Nome	Descrição
failed_jobs	Contém informação da fila de *queues* falhados (jobs).
jobs	Contém informação da fila de *queues* (jobs).
migrations	Contém informação das tabelas para construir a base de dados.
notifications	Contém informação das notificações criadas.

Tabela XIV - Descrição das tabelas da base de dados - Tabelas existentes do Laravel.

4.3.1.9. Evolução dos módulos

A partir de 26 junho 2021, e no decorrer do tempo, foram realizadas melhorias nas tabelas da base de dados, de acordo com os problemas encontrados durante o desenvolvimento do mesmo, dentro dos quais resultaram na adição e/ou remoção de campos e/ou tabelas, encontrando assim novas entidades, atributos e relações.

No módulo do utilizador foram criadas as tabelas *user_states* e *user_state_update* contendo a informação dos estados de um utilizador e do motivo para alteração do estado do utilizador. Ao submódulo de recuperação de password, foi removido o prefixo *request_* das tabelas *request_password_reset_states* e *request_password_resets* contendo atualmente a informação dos estados do pedido de alteração de password. Ainda foi adicionado as tabelas de *password_recoveries* e *password_recovery_states* contendo atualmente os dados do pedido de recuperação e seu estado. Respetivamente, no submódulo de registo, à tabela existente *registration_codes* foi adicionada a relação com a tabela *registration_code_states* contendo os estados do pedido de registo e adicionado a tabela *pre_registrations* contendo os pedidos de registo externos à plataforma.

No módulo de gestão dos pedidos, foram removidas as tabelas de *bundle_has_addons*, *request_product_budles, custom_signature_packs*, *custom_timestamps*, *signature_data* ficando estes dados registados nas tabelas existentes de *request* e *request_product.* Para melhorar a sintaxe, foi alterada a tabela *cert_conditions* para *certificate_conditions*. A tabela *request_file_types* foi modificada para *file_types,* referenciando-se na tabela de *files* e posteriormente, removido a tabela *request_product_has_files* devido à implementação do conceito de polimorfismo na tabela *files*, acrescentando os identificadores das tabelas referenciadas e o *fileable_type*. A tabela *request_product_reason,* foi adicionado para armazenar o estado do pedido de aquisição e o motivo pelo qual se encontrava naquele estado, fornecendo desta forma mais informações ao próprio pedido.

No módulo de gestão de certificados a tabela *certificate_registry* foi convertida para *certificates* melhorando a sua sintaxe.

Todas as tabelas com a presente coluna *description* e/ou *name*, adicionou-se o sufixo das linguagens, *_pt*, *_es* e *_en* de forma a traduzir a base de dados do sistema. Às tabelas tipos foi acrescentado a coluna *enum,* de modo que o *frontend* e integrações possuam conhecimento da *string* deste tipo de dado do GraphQL.

Nos outros módulos, foi removido as tabelas de *download_file_requests* e *file_has_autorization,* removendo assim necessidade de *tokens* de autorização para efetuar o download e ainda foi adicionado a coluna *updated_at* às tabelas existentes para guardar o seu registo de alteração de dados.

Em suma, as alterações mencionadas fora necessárias durante o desenvolvimento do sistema, permitindo uma melhor compreensão das tabelas na base de dados, a redução de tabelas e relações, levando a uma melhor gestão do sistema, um acesso centralizado e mais rápido, bem como a importância da tradução do sistema de base de dados na obtenção de uma plataforma multilingue.

4.4. Validação

A validação dos requisitos é uma fase importante, dado que minimiza os erros e obriga à qualidade do sistema, uma vez que as alterações em requisitos já consolidados têm um custo muito superior a alterações no código e design do sistema. Este tipo de erro acontece devido ao processo do desenvolvimento do sistema ser iterativo e incremental, traduzindo-se em elevados custos e necessidade de refazer muito do trabalho de que já se julgava concluído.

A nível do sistema, as seguintes validações permitiram que o sistema pretendido fosse eficiente e eficaz.

4.4.1. Psr

O PHP *Framework Interoperability Group* (PHP-FIG) criou vários PHP *Standard Recommendation* (PSR), identificados por um número e um status, que se refere ao conjunto de padrões que promovem

assim as boas práticas na programação seguindo estas recomendações[75]. Neste trabalho foi então aplicado o PSR-4 que promove o carregamento automático a partir de caminhos e o PSR-12 que por sua vez, auxilia na estruturação e uniformização do mesmo estilo na escrita de código PHP.

4.4.2. Depuração de exceções

O lançamento de exceções por parte do sistema implementado no tratamento de erros deve exibir uma mensagem de erro explícita em JSON [76]. Desta forma, neste trabalho foram manipulados os erros da *framework*, podendo em qualquer ação que o utilizador execute, serem lançados erros, obtendo assim um código explícito como por exemplo "*ErrorCode*: xCCC002", auxiliando na localização, tratamento e resolução desse erro no código.

4.4.3. Validações do utilizador e dados

O tratamento das permissões de acesso de um determinado cliente seja pelo *frontend* ou por uma integração, permitem a gestão do acesso ao recurso de um utilizador. A validação dos campos necessários e dados provenientes destes utilizadores, previnem e minimizam os erros no sistema, obrigando o formato correto durante a comunicação dos dados para a API.

4.5. Testes

Nesta secção é descrita brevemente a abordagem utilizada para testar a implementação dos requisitos elicitados, discutindo os diferentes tipos de testes garantindo assim o correto funcionamento do sistema.

4.5.1. Testes HTTP

Para testar a API do sistema, utilizou-se os testes HTTP da *framework* PHPUnit [77]. Esta *framework* é uma estrutura de testes unitários para a linguagem PHP e encontra-se inserida no Laravel [78]. Os diretórios `Feature` e `Unit` são utilizados para armazenar os testes, e o arquivo `phpunit.xml` no *root* contém as configurações para os testes à aplicação.

Estes testes HTTP permitem realizar solicitações HTTP para a API, e posteriormente verificar as *assertions* do código. Ao realizar pedidos HTTP utilizando os verbos de HTTP, é possível verificar a estrutura da resposta do resultado obtido, a interação entre objetos e os registos na base de dados.

4.5.2. Laravel Dusk

O Laravel Dusk [79] fornece um teste e automação de browser. Por padrão, o Dusk não requer a instalação de um *Java Development Kit* (JDK) ou de um software Selenium [80]. Contrariamente, o Dusk utiliza um *ChromeDriver* autônomo que através do Google Chrome executa os testes do navegador, no entanto pode realizar os testes em qualquer navegador com um servidor próprio Selenium. O diretório existente é denominado de `Browser` para armazenar estes testes.

4.5.3. Testes Unitários

O objetivo deste tipo de teste é verificar os blocos mais pequenos de código, de modo isolado, para verificar o funcionamento adequado. Estes testes unitários estão armazenados no diretório `Unit` dos testes HTTP.

4.5.4. Testes de integração

Os testes de integração são mais complexos que os testes unitários, uma vez que os módulos desenvolvidos são combinados e testados. Estes testes de integração estão armazenados no diretório `Feature` dos testes HTTP.

4.5.5. Testes de usabilidade

Este teste é realizado juntamente com os utilizadores para aferir se o produto satisfaz as necessidades dos mesmos. E também analisa de que forma é que o utilizador usa o sistema, verificando assim onde é que este apresenta mais dificuldade. Através das suas opiniões, obtemos um feedback sobre o aspeto visual da plataforma, sucesso ou insucesso na execução de tarefas e nível geral de satisfação.

4.5.6. Testes de carga

Estes testes apresentam um grande valor na avaliação, dado que é uma forma de averiguar a capacidade de resposta do servidor, para vários pedidos de utilizadores em simultâneo.

Com a constante monitorização do hardware do servidor podemos analisar o CPU, a memória RAM, o tempo de execução dos pedidos, o tempo de carregamento de cada página, etc. Numa análise destas dimensões, é possível identificar quais os recursos que devem ser aumentados no servidor, quais os pedidos que necessitam de otimização, entre mais, otimizando e cativando os utilizadores a não abandonar a plataforma no imediato.

4.5.7. Testes de segurança

Para os testes de segurança é utilizado o software Acunetix [81]. Este software permite analisar a aplicação web e verificar a existência de vulnerabilidades na mesma.

Numa análise completa à plataforma usando este software, os resultados indicam problemas por nível: alto, médio, baixo risco e exibem alertas de informações, nomeadamente às configurações no servidor, às bibliotecas que possuem vulnerabilidades, à localização de ficheiros sensíveis ao público, entre outros.

4.6. Sumário

Ao longo deste capítulo do método, descreveu-se detalhadamente todos os requisitos legais, funcionais e não funcionais, bem como os diferentes utilizadores necessários que vão interagir com funcionamento do sistema da GTS.

O levantamento e planeamento da base de dados foram importantes para compreender os diversos módulos do sistema, e ainda modelar as alterações requisitadas ao longo do tempo, cumprindo com o necessário para o correto funcionamento e migração dos dados para este novo sistema.

As validações dos requisitos previnem o mau planeamento do sistema embora tenham sido alterados requisitos já desenvolvidos resultando em mudanças de código atendendo às novas expectativas, enquanto as validações do sistema garantem que o código do produto desenvolvido segue os standards das tecnologias e linguagens, filtrando e validando os dados provenientes dos utilizadores em qualquer ação realizada na API.

Por fim, a prática de diferentes testes à API são fundamentais e cruciais no desenvolvimento de uma API funcional e robusta, pois com este recurso é garantido todo o correto funcionamento da aplicação, realizando a cobertura do código produzido, conforme o esperado além de obter métricas de performance e disponibilidade no sistema.

5. Implementação

Neste capítulo é apresentada a implementação prática do sistema, levando em consideração os requisitos estabelecidos anteriormente. Na seção 5.1, detalhamos a organização de ficheiros da *framework* Laravel com o GraphQL, o fluxo de autenticação da plataforma e a documentação do produto em *yml*.

5.1. Desenvolvimento

Nesta secção é demonstrado o desenvolvimento da plataforma e com o recurso às ferramentas descritas no ponto 2.4, foi utilizado o Laravel como *framework* no desenvolvimento desta aplicação e a biblioteca Lighthouse para consumir a linguagem GraphQL na comunicação. Todas as imagens utilizadas neste capítulo estão presentes no ANEXO B – IMPLEMENTAÇÃO.

5.1.1. Estrutura de ficheiros

A estrutura de ficheiros duma aplicação padrão do Laravel, fornece uma organização clara e livre para customização sem quaisquer restrições de classes para os desenvolvedores desde que o *composer* possa carregar automaticamente a classe [79]. As pastas `app/GraphQL/graphql` e o ficheiro de configuração em `config/lighthouse.php` foram criados após a instalação da dependência do Lighthouse pelo *composer* permitindo assim a utilização do GraphQL. Na figura 33 está representado a estrutura de ficheiros dos seguintes diretórios:

O diretório `app` é a pasta mais importante na estrutura de ficheiros, pois nesta conta o código principal da API. e ainda existem as pastas de `Casts`, `Console`, `Custom`, `Exceptions`, `GraphQL`, `Helpers`, `Http`, `Logging`, `Models`, `Notifications`, `Observers`, `Policies`, `Providers`, `Repositories`, `Schedules` e `Traits`.

A pasta `app/Casts` permitiu registar as classes de conversão personalizáveis, do tipo de dado provenientes de cada coluna da base de dados para o código e vice-versa (do tipo de dado no código para a coluna em questão na base de dados). Por conseguinte, as traduções são armazenadas na base de dados por diversas colunas conforme o país e quando são retornados os dados, apenas é retornada a coluna com a tradução onde é possível observar a sua estrutura pela figura 34.

A pasta `app/Console` possui o código dos comandos que podem ser executados na linha de comandos, o agendamento dos comandos e tarefas a serem executadas. No `app/Console/Kernel.php` foi possível configurar o agendamento das tarefas a serem executadas em intervalos específicos, como a tarefa *expire_registration_codes*, que é executada de minuto a minuto, alterando o estado do código de registo conforme a figura 35.

A pasta `app/Custom` criada pelos desenvolvedores, incluí as classes de lógica do tratamento de dados que o *backend* recebe e possui na base de dados, efetuando as modificações necessárias para o formato obrigatório, para as chamadas ao código do pedido às integrações sendo possível observar a sua estrutura pela figura 36.

A pasta `app/Custom/Facades` inclui a *string* das *facades*, conforme a figura 37, de forma a usar as classes associadas sem criar manualmente um objeto, com recurso ao *service container* do Laravel nos *providers*.

A pasta `app/Exceptions` possui a classe das exceções registadas na aplicação, que quando lançadas são processadas antes de serem enviadas para o utilizador. Conforme a figura 38, foi criada uma exceção para que fossem capturadas todas as exceções e erros retornando a mesma estrutura para o *frontend*.

A pasta `app/GraphQL` possui diversas pastas `Diretives`, `Mutations`, `Queries`, `Resolvers`, `Scalars` e `Validators`. Na `app/GraphQL/Diretives` são criadas classes de diretivas personalizadas para utilizar no *schema* de GraphQL, como por exemplo o `@whereBetweenDates` presente na figura 39. Na `app/GraphQL/Mutations`, encontram-se as classes e funções com toda a lógica de um pedido *mutation*, que executa as ações de *create*, *update* e *delete*, estas funções são responsáveis por receber os argumentos do utilizador, processar e enviar a resposta aos pedidos efetuados pelos utilizadores, segue a sua estrutura de classes presente na figura 40. Na `app/GraphQL/Queries` estão classes com as funções que contêm a lógica de apresentação de um pedido *query*, que executa a ação de *read*, sendo as funçoes responsáveis por receber os argumentos do utilizador caso existam, processar e enviar as respostas aos pedidos efetuados, segue a sua estrutura na presente figura 41. Na `app/GraphQL/Resolvers`, contém as classes com funções que resolvem qualquer consulta de GraphQL, são responsáveis por receber os argumentos do utilizador, processar e enviar a resposta ao pedido efetuado pelos utilizadores, segue a sua estrutura de classes presente na figura 42. Na `app/GraphQL/Scalars`, são armazenados os tipos de dados (*scalar*) personalizados para o *schema* de GraphQL, onde cada tipo de dado que representa um valor, tal como uma *string*, número, *boolean* ou data, mas neste caso não foram criados tipos de dado, visto que o Lighthouse já os fornece. A `app/GraphQL/Validators`, contém as classes com as regras e validações dos dados introduzidos pelo utilizador no GraphQL, segue a sua estrutura de classes presente na figura 43.

A pasta `app/Helpers` possuí a declaração de métodos específicos para a reutilização de código, evitando assim a repetição do código deste método, conforme a figura 44. O seu funcionamento é muito idêntico a uma *trait,* sem a necessidade de declarar o *use* do *namespace* do *helper* na classe em que se pretende utilizar o método.

Na pasta `app/Http` existem as pastas `Controllers`, `Middleware` e `Requests`. Os c*ontrollers* contêm toda a lógica de negócio, nestes agrupam-se a lógica dos pedidos HTTP, visto serem responsáveis por receber, processar e enviar resposta aos pedidos efetuados pelos utilizadores, podendo observar-se a sua estrutura na figura 45. Um m*iddleware* contém os mecanismos necessários para filtrar o início dos pedidos HTTP efetuados na API, permitindo assim que os pedidos à API fossem autenticados por *tokens* de acesso, ou que existisse o *header* para tradução conforme a figura 46. Na pasta `Requests` foram criados os *FormRequest*, para separar a lógica da validação da lógica de negócio do *controller*, conforme a figura 47.

A pasta `app/Logging` possuí uma classe com customização do tratamento de log, permitindo assim obter-se informações importantes como o endereço IP e o *path* do pedido conforme a figura 48. Pela configuração da figura 49, foi inserido na *key* de *channels* um driver personalizado da configuração de *logs*.

A pasta `app/Models` possuí os modelos de Eloquent ORM, responsáveis pelo acesso às tabelas da base de dados, permitindo realizar as operações de leitura, escrita, edição e exclusão de dados. Para cada tabela, como por exemplo a figura 50, foi criado uma *model* que permitiu utilizar o *trait SoftDelete,* uma funcionalidade que não permite remover permanentemente uma *row* da base de dados, mas seja

atualizado a coluna *deleted_at*, excluindo esta *row* da consulta ou podendo reverter a ação de apagar. A variável `$fillable` possuí os campos específicos para a atribuição de valores. A adição de *scopes* permite a reutilização da consulta à base de dados. A descrição das relações permite que sejam consultados, editados e excluídos os dados da tabela com a sua referência, podendo ainda fazer o *cast* de colunas com a tradução pretendida conforme a figura 51.

A pasta `app/Notifications` permite o envio de e-mail, SMS e entre outros métodos, podendo armazenar o registo da notificação na base de dados conforme a figura 52.

A pasta `app/Observers` possuí métodos de eventos, que decorrem após uma ação do *model eloquent*, e a pasta `app/Policies` possuí as políticas de autorização para controlar o acesso dos utilizadores aos recursos específicos da aplicação. Neste caso, não foram adicionados nenhum *observer* e nenhuma *policy*.

A pasta `app/Providers` contém os provedores de serviço, responsáveis por inicializar e configurar os vários componentes da *framework* utilizados para os serviços da aplicação, como serviço de cache, gestão de eventos, provedores de log, etc. No `AppServiceProvider.php` foram adicionados o provedor de log das *querys* e o mapeamento do polimorfismo, conforme a Figura 53. No `AuthServiceProvider.php` foram adicionados as rotas do *passport*, o tempo de vida do *token* e do *refresh_token*, conforme a figura 54. Conforme a figura 55, foi criado o `RepositoryServiceProvider.php` para registar as *facades* às classes responsáveis pela lógica.

A pasta `app/Repositories` foi criada para possuir as classes que encapsulam a manipulação dos dados necessários da aplicação, e a lógica da comunicação com serviços ou integrações internas, o que torna o código mais organizado e fácil de manter.

A pasta `app/Schedules` possuí as classes de tarefas, mas não foram realizadas adições de novas classes neste diretório.

Na pasta `app/Traits` foram criados *traits* com métodos específicos, para reutilizar o código nas diversas classes. Na figura 56 encontra-se a implementação do armazenamento de ficheiros durante o carregamento de ficheiros, nos diversos formulários. O funcionamento de um *trait* é muito idêntico à de um *helper,* mas carece da necessidade de declarar o *use* do *namespace* desse *trait* na classe em que se pretende utilizar o(s) método(s) do *trait*.

A pasta `bootstrap` contém os ficheiros gerados pela estrutura do cache de rotas e serviços, para otimização e desempenho. Sendo que não foram realizadas alterações neste diretório.

A pasta `config` possuí os ficheiros de configuração da API, onde foram adicionadas e editadas as configurações originais consoante o pretendido. Conforme ilustrado na figura 57, foram criados ficheiros de configuração para cada módulo desenvolvido. E para aceder à configuração, basta usar o seguinte código `config('nome_do_ficheiro.key_do_array')`.

O diretório `database` contém as pastas de `factories`,`migrations` e `seeds`. Uma *factory* permite inserir registos nas tabelas da base de dados para efetuar testes, declarando os valores das colunas a popular, podendo ainda usar-se *states* para alterar a estrutura inicialmente criada. Conforme a figura 58 está a definido uma *factory* para a criação de uma fatura. Uma *migration* permite especificar a tabela e os seus campos. Para cada campo é possível definir os tipos de dado, tamanho esperado e chaves estrangeiras. Desta forma é definido o esquema da base de dados, realizando um controlo de versão da

base de dados pelos *timestamps* dos ficheiros, com as funções *up* para criar e *down* para realizar o *rollback* da(s) coluna(s) e ou tabela(s). A figura 59 demonstra como foi criada a tabela *users* e a figura 60 demonstra alterações de colunas à tabela *users*. Uma base de dados é populada com as *seeds*, conforme a figura 61 que exibe uma *seed* para uma tabela tipo (ou de estados). As restantes tabelas que não são de estados, ficarão pendentes da futura migração de dados. Conforme a figura 62, com o comando `php artisan migrate:fresh –seed` é possível recriar a base de dados.

A pasta `public` contém vários ficheiros, tais como o `.htaccess` para configurar todas as solicitações, podendo ainda conter ficheiros de JavaScript e CSS. Nesta pasta, foi posto o logotipo da GTS de forma a ser inserido nos PDF's gerados e enviados ao cliente.

Na pasta `resources` contém as *views* da plataforma, ficheiros de CSS ou JavaScript e também armazena todos os ficheiros do idioma. O conteúdo da pasta `views` são os ficheiros HTML em *blade* (mecanismo de *template* padrão do Laravel), necessários para a geração de PDF's conforme a figura 63. No diretório `lang` foram adicionadas as traduções de inglês na pasta `en`, de espanhol na pasta `es` e português na pasta `pt` conforme a figura 64.

A pasta `routes` contém os ficheiros `web.php` e `api.php`, com as definições das rotas permitidas na aplicação. Cada rota define o método HTTP (*get*, *post*, *put* ou *delete*), o caminho, o controlador e o método que vai resolver a solicitação, e ainda foram definidos os *middlewares* que filtravam os pedidos antes de passarem no controlador. Conforme as figura 65, foi definido uma resposta quando não é encontrada a rota que o utilizador pretende aceder pelo *fallback* no `web.php` e foi adicionado nas rotas `api.php` um prefixo da versão no *Uniform Resource Identifier* (URI) conforme a figura 66.

A pasta `storage` contém os ficheiros de *upload* carregados na plataforma, ficheiros de cache gerados pela *framework* criados durante o processamento da aplicação e os ficheiros de *logs*. Nesta pasta, são carregados os ficheiros encriptados, as *keys* de encriptação do OAuth geradas pela aplicação com o comando `php artisan key:generate` e os ficheiros `.logs` em pastas pela data atual conforme a estrutura da figura 67.

A pasta `tests` contém os testes automatizados da API nos diretórios de `Feature`, `Unit` e `Dusk`. De modo a testar os pedidos de GraphQL nos testes, foram necessárias algumas alterações às classes de testes [82]. Cada teste segue uma determinada estrutura obrigatória, onde uma classe tem o sufixo *Test* e os seus métodos o prefixo *test*. Os métodos avaliam as asserções do pedido conforme a figura 68.

A pasta `vendor` armazena os ficheiros de *autoload* de código referentes a pacotes de terceiros.

O ficheiro `.env` e `.env.example` contém os dados sensíveis da aplicação, como URL da APP, credenciais de acesso à base de dados, servidor de email e integrações.

O ficheiro `composer.json` contém e gere as dependências de pacotes de terceiros pelo *composer*.

O ficheiro `docker-compose.yml` contém e gere a definição dos serviços, redes e volumes do Docker.

5.1.2. Autenticação do utilizador

No processo de login, o utilizador é redirecionado para uma API para colocar as suas credenciais de acesso, conforme a figura 69. Através das ações, Enviar ou Cancelar do utilizador no *callback* do login, este grava o *url* do *frontend* que fez o pedido em sessão, verifica os erros do pedido e o estado do

utilizador que está efetuando o login. No fim deste método, o utilizador é redirecionado para o servidor de autorização, pedindo o *code* conforme a figura 70.

Em seguida, o *code* é trocado por um acess_*token* na seguinte função de *callback*, conforme a figura 71. O resultado desta operação está presente na figura 72, e origina um tempo de vida de utilização do *acess_token*, um *acess_token* e um *refresh_token* para obter novo *acess_token*.

5.1.3. Documentação

Toda a documentação dos pedidos à API está num repositório de GitLab privado, podendo realizar solicitações de HTTP e GraphQL pelo Insomnia conforme a figura 73. Esta documentação está disponível apenas para os desenvolvedores e integrações externas, mantendo-se atualizada conforme as mudanças no código. As funções do código-fonte são documentadas com recurso de extensões do IDE de VSC.

5.1.4. Schema GraphQL

Consoante a figura 74, é seguido a mesma estrutura no diretório de `/graphql` em cada recurso desenvolvido, representando um módulo de negócio, onde é descrito o *schema* de *inputs*, *mutations*, *queries* e *types* do GraphQL.

Na figura 75, encontram-se os *inputs* que ditam os parâmetros de entrada para as *querys* e/ou *mutations* do módulo de *reviews*. A declaração de um *input* começa pela *tag* `input` seguido do nome do *input*, onde são descritos os campos a receber e os seus respetivos tipos de dados impostos, podendo ser *strings*, números, *inputs* de GraphQL ou até receber como *array,* e não podem ser *null* devido ao sinal de exclamação presente no código.

Pela figura 76, consta a implementação de uma *mutation* para o módulo de *reviews*, onde é iniciado com o a *tag* `extended type Mutation`, seguido do nome da *mutation* que recebe os dados de *input* `CreateReviewHasAnswerInput`. Pela diretiva `@spread` aplicada ao tipo *input,* os argumentos são transformados num *array* plano sendo acedidos facilmente. Logo após os dois pontos no código, está o *type* de retorno da *mutation*. A diretiva `@protectedQuery`, com o valor predefinido de `cliente` (podendo ainda ter os valores de `any` ou `admin`) permitem que apenas determinados tipos de utilizador acedam à função. A diretiva `@validator` especifica o caminho das regras de validação do(s) *input(s)*. A diretiva `@field` especifica o caminho para a *mutation*, *query* ou *resolver*, a classe e a função que vai ser executada.

Na figura 77, encontram-se as *querys* para as *reviews*, onde é iniciado com a *tag* `extended type Query`, seguido do nome da *query*, os *inputs* da função nos parênteses, e logo a seguir dos dois pontos o retorno da função e as respetivas diretivas. A diretiva `@in` e `@where` realizam as operações de `where` e `whereIn` do Laravel na consulta à base de dados, nas colunas especificadas em `key`. A diretiva `@all` permite consultar todos os valores mesmo com o *deleted_at* preenchido, enquanto a `@softDelete` remove as *rows* com a coluna *delete_at* preenchida da consulta.

Na figura 78, apresentam-se os *types* para as *reviews,* sendo o retorno da função, este é iniciado com o a *tag* `type`, seguido do nome do tipo, os campos a retornar e a sua tipagem específica.

Por fim, observando a figura 79, estão os *scalars* a serem usados, a importação das pastas que contêm o *schema* de *inputs*, *mutations*, *queries* e *types*, e ainda a declaração dos *enums* presentes no sistema, no `schema.graphql`.

6. Resultados e Avaliação

Neste capítulo, será debatido os resultados realizados na etapa de testes. Esta etapa é fundamental e importante no desenvolvimento de uma API funcional e robusta, pois com este recurso é possível encontrar as falhas durante a etapa de implementação do sistema, fazendo também com que cada desenvolvimento e manutenção forneça uma maior confiança de que todo o sistema esteja funcionando conforme o esperado.

Os resultados dos testes http e Laravel Dusk, referidos nas secções 6.1 e 6.2 respetivamente, realizados à plataforma foram satisfatórios, visto que efetuam-se testes às funções com sucesso e em pouco tempo. No final do desenvolvimento, ficou ainda definido que haverá uma avaliação com os clientes, apoio da plataforma e *beta testers* da entidade ACIN, de forma a avaliar o sistema desenvolvido.

6.1. Testes HTTP

Para alcançar o objetivo de testar todos os pedidos da API e o máximo dos seus possíveis casos, foi desenvolvido um conjunto de 85 (oitenta e cinco) testes e 104 (cento e quatro) *asserts*, que equivalem a um pedido ao método de cada funcionalidade desenvolvida mais o(s) resultado(s) pretendido(s). A *framework* PHPUnit permite gerar um documento em XML com os resultados dos testes realizados, que por sua vez converte-se para HTML para uma melhor interpretação desses resultados, mas neste caso foi utilizado a consola do IDE na geração de dados com o comando `php artisan test`. Na Tabela XV segue-se os resultados dos testes realizados.

Classes testes HTTP	Validações/Métodos	Resultado
Unit		
RegisterCodeTest	testCheckSize	OK (0.15s)
	testMailRegistrationCode	OK (0.18s)
Feature		
AddAddonTest	testAddAddon	OK (1.62s)
	testAddAddonExtended	OK (1.77s)
AddFileTest	testUploadTwoFiles	OK (0.23s)
	testUploadOneFile	OK (0.20s)
AddProductTest	testProductTest	OK (1.56s)
	testProductTestWithData	OK (1.56s)
	testProductTestWithAddon	OK (1.52s)
	testProductTestWithAddonPlusData	OK (1.63s)
BillingAddressTest	testAddBillingAddress	OK (0.30s)
	testEditBillingAddress	OK (0.33s)
	testDeleteBillingAddress	OK (0.25s)
CreatRequestTest	testCreatRequest	OK (1.52s)
	testCreatRequestWithAddon	OK (1.61s)
	testCreatRequestWithPackage	OK (1.62s)
	testCreatRequestWithBundle	OK (1.56s)
	testCreatRequestWithBundleAddon	OK (1.64s)
InvoiceTest	testInvoiceCreation	OK (0.59s)
	testInvoiceUpdate	OK (0.58s)
	testInvoiceFileFieldUpdate	OK (0.67s)
	testStoreInvoiceByBillingAddress	OK (0.60s)

ListDescriptionTest	testListDescription	OK (0.11s)
	testListDescriptionWithData	OK (0.23s)
	testListDescriptionWithDataExtended	OK (0.19s)
ListProductsTest	testListProductsById	OK (1.62s)
	testListProductsByIExtended	OK (1.59s)
	testListProductsByConditionNew	OK (1.58s)
	testListProductsByConditionRenew	OK (1.58s)
	testListProductsByTypePk	OK (1.64s)
	testListProductsByTypeTs	OK (1.77s)
	testListProductsByTypeSsl	OK (1.72s)
	testListProductsByTypeDst	OK (1.77s)
	testListProductsByTypeDsg	OK (1.72s)
	testListProductsByTypeAddon	OK (1.17s)
	testListProductsByTypeExtended	OK (1.95s)
	testListProductsByCertificateTypeNotApplicable	OK (1.96s)
	testListProductsByCertificateTypeQualified	OK (1.95s)
	testListProductsByCertificateTypeNotQualified	OK (2.33s)
	testListProductsByCertificateTypeExtended	OK (2.15s)
	testListProductsByAllOptions	OK (4.15s)
	testListProductsByAllOptionsFail	OK (2.04s)
ListRequestTest	testListRequestByUserId	OK (1.65s)
	testListRequestByRequestNumber	OK (1.66s)
	testListRequestByRequestState	OK (1.03s)
	testListRequestByProductType	OK (1.93s)
	testListRequestByDate	OK (1.22s)
	testListRequestByProductId	OK (1.80s)
	testListRequestByAllOptions	OK (2.17s)
	testListRequestByAllOptionsFail	OK (2.14s)
LoginTest	testRegisterUser	OK (0.66s)
	testLoginUser	OK (0.51s)
	testPasswordRecovery	OK (0.34s)
PasswordResetTest	testGuestUserMakesResetWhithoutCertificates	OK (0.53s)
	testGuestUserMakesResetWhithoutCertificatesAndValidateToken	OK (0.72s)
	testGuestUserMakesResetWhithoutCertificatesAndInsertsNewPassword	OK (0.96s)
	testGuestUserMakesResetWhithCertificates	OK (0.68s)
ProofOfPaymentTest	testUploadProofOfPayment	OK (0.32s)
	testValidateProofOfPaymentAccept	OK (0.39s)
	testValidateProofOfPaymentReject	OK (0.38s)
	testValidateProofOfPaymentFail	OK (0.29s)
PaymentTest	testCreatePaymentRequest	OK (1.76s)
	testCreatePaymentService	OK (1.59s)
	testDeleteOrderPaymentService	OK (1.69s)
RemoveFileTest	testRemoveFile	OK (0.26s)
	testRemoveFileExtended	OK (0.33s)

RemoveProductsTest	testRemoveProducts	OK (1.44s)
	testRemoveProductsAll	OK (1.42s)
	testRemoveProductsPackages	OK (1.41s)
	testRemoveProductsPackagesAll	OK (1.48s)
RemoveRequestTest	testRemoveRequest	OK (1.43s)
ReviewTest	testAddReview	OK (0.86s)
SendRegistrationCodeTest	testSendRegistrationCodeNewUser	OK (0.49s)
	testSendRegistrationCodeExistingUser	OK (0.48s)
	testSendMultipleRegistrationCode	OK (0.41s)
UpdateProductTest	updateProduct	OK (1.32s)
	updateMultipleProducts	OK (1.43s)
	updateProductCustomTs	OK (1.42s)
	updateProductSignatureExtra	OK (1.52s)
	updateProductAddFile	OK (1.58s)
UserTest	testUserCreation	OK (0.64s)
ValidateCodeTest	testValidateCodeForValidCodeAndEmail	OK (0.59s)
	testValidateCodeNotExistingUser	OK (0.58s)
	testValidateCodeSomeUserWrongCode	OK (0.56s)
	testValidateCodeMultipleSendedCodes	OK (0.57s)

Tabela XV - Descrição dos testes de HTTP.

6.2. Laravel Dusk

Para garantir que as ações no browser funcionassem conforme o esperado, foi desenvolvido um conjunto de 3 (três) testes e 4 (quatro) *asserts*, onde foi utilizado o comando `php artisan dusk` no IDE a fim de gerar os resultados aos testes realizados na Tabela XVI.

Dusk	Validações	Resultado
Browser		
PaymentTest	testReturnRedirect	OK (1.3s)
	testCancelRedirect	OK (1.6s)
	testWeb	OK (3.6s)

Tabela XVI - Descrição dos testes de Dusk.

7. Discussão

Neste capítulo será abordado os demais objetivos alcançados e problemas encontrados na plataforma desenvolvida.

Um dos objetivos alcançados com a plataforma desenvolvida foi a capacidade de comprar produtos em maior escala, sendo um grande benefício a nível comercial para a própria empresa, mantendo o preço mais acessível entre os concorrentes nacionais e internacionais.

A linguagem de programação utilizada na plataforma está na versão 7.4 e encontra-se descontinuada dado que esta já não recebe suporte de segurança desde 28 novembro de 2022 [83].

No que diz respeito às tecnologias da plataforma, pelo uso da *framework* Laravel 8.X, esta futuramente necessitará de atualização para o 10.X dado a sua rápida evolução, que levará a melhorias de desempenho ou segurança, correções de bugs e novas funcionalidades. Na plataforma atualmente desenvolvida, continua a ocorrer a mesma falha que na plataforma antiga, no que diz respeito ao tipo de dado *timestamp* para as datas da base de dados do MySQL que terminará em 2038, sendo facilmente colmatado com a troca para o tipo *datetime*. Outro dos problemas futuros, estão relacionado com o consumo de dados com GraphQL, onde o seu *type Int* só consegue suportar 32bits, limitando o número máximo de elementos que podem ser armazenados, pelo que ainda não existe nenhuma solução, mas sendo pouco provável restringir o uso desta aplicação.

Durante o método deste processo formal de Engenharia de software contínuo e incremental, em consequência da elevada complexidade de requisitos da plataforma, comunicação e validação entre responsáveis, até à decisão de novos requisitos juntamente com os requisitos já existentes, houve diversas mudanças na implementação da decisão inicial que levaram ao atraso no desenvolvimento.

A implementação deste sistema foi apropriada dado que é capaz de traduzir a plataforma e a base de dados para os clientes atuais, sendo extensível a mais línguas. Com a utilização de códigos de erros, estes auxiliam a gestão e depuração das exceções, porém existe uma falha na utilização do *schema* GraphQL que poderá existir incompatibilidade com as futuras integrações externas e internas, devido à sua diferença relativamente ao tradicional pedido de HTTP mesmo existindo a documentação de apoio criada pelos *developers*.

Pelos resultados e avaliação dos testes, a utilização de testes HTTP automáticos e do Laravel Dusk, estes permitiram abranger o código de certos módulos, tornando-se eficaz para os próximos novos *deployments* de código e na aceitação do código, além do mais, foram executados em pouco tempo e com sucesso proporcionando uma maior rapidez no sistema. O novo *layout* desenvolvido para esta plataforma permitiu também uma melhor acessibilidade e usabilidade apesar da grande mudança nos módulos de login da plataforma, registo e gestão de pedidos face à plataforma anterior.

8. Conclusão

Esta tese foca-se na criação da nova GTS, resolvendo os problemas amplamente identificados, onde o resultado pretendido deste trabalho era a possibilidade de comprar produtos de forma rápida, fácil e sem limitações de quantidades dos produtos, disponibilizados pela GTS.

O trabalho apresentado cumpre com os objetivos pretendidos, resolvendo assim os problemas existentes incorporando os restantes requisitos da plataforma anterior.

Esta aplicação inclui assim o registo de utilizadores sem o armazenamento prévio dos dados do utilizador, autenticação pelo OAuth, um sistema de compras baseado em carrinho virtual, integração com os sistemas externos (iGEST e PayPay), a opinião dos clientes após aquisição de produtos, mantendo a aplicação de assinaturas em documentos PDF e XML.

A utilização de novas metodologias, ferramentas, estilos e padrões arquiteturais, juntamente com as tecnologias utilizadas, foram uma nova experiência a nível profissional que permitiram o desenvolvimento e implementação desta API da GTS.

O estado do projeto ainda continua em desenvolvimento, devido à quantidade de tempo necessário para o desenvolvimento das *features* que não impedem o negócio atual da plataforma, além disso, qualquer alteração dos requisitos, requer alterações no código, testes e documentação para que o *frontend* depois suporte as novas alterações da API.

8.1. Lições aprendidas

Ao longo deste do Mestrado em Engenharia Informática na UMA, adquiri novos conhecimentos da área de informática bem como, relembrei técnicas lecionadas na Licenciatura em Engenharia Informática na Escola Superior de Tecnologia do Instituto Politécnico de Castelo Branco. No que diz respeito à carreira profissional, trabalhar na empresa ACIN permitiu investir pessoalmente nas diversas ferramentas e linguagens necessárias para elaboração dos projetos. Durante o percurso, foi muito desafiante equilibrar o trabalho com as atividades acadêmicas mas sempre foi possível fazê-lo graças à dedicação demonstrada durante este tempo enquanto trabalhador-estudante. Por fim, consegui obter experiência profissional durante a realização do curso de mestrado relacionado diretamente à minha área, o que contribuiu para ampliar os meus conhecimentos técnicos e teóricos assim como expandir o meu leque técnico de vocabulário, *networking* de profissionais na área e consequentemente aprendido com o *lesson learned* significando que cada erro é uma lição aprendida.

8.2. Trabalho Futuro

Após a conclusão do projeto, podemos traçar algumas perspetivas futuras. Estas incluem-se na modulação e desenvolvimento dos requisitos para os módulos de administrador e parceiro da GTS, que consumirão a API. Além disso, esta abordagem permitirá que possam ser identificadas as necessidades específicas deste projeto para garantir a sua execução bem-sucedida num futuro próximo.

Em relação à migração da base de dados da aplicação atualmente existente para a nova API desenvolvida, será um dos próximos pontos a ser revisto brevemente, pois realizou-se alterações na mesma, em resposta aos problemas anteriores e os requisitos mais recentes, sendo fulcral garantir que o processo de

migração seja feito com sucesso, sem comprometer o funcionamento adequado do sistema mantendo os dados já existentes dos utilizadores.

Por fim, destacam-se as atualizações para o PHP 8 e a *framework* Laravel 10, a integração do PAYPAY usando REST em vez do protocolo SOAP, a utilização do *continuous integration/continuous delivery* (CI/CD) do GitLab e integração com a Salesforce, fornecendo mais segurança, validações à aplicação após lançamento e a gestão de todos os dados de clientes e interações com clientes num único lugar. Estas alterações à API exigem uma manutenção contínua, detalhando o trabalho desenvolvido em *releases* juntamente com a documentação necessária, além disso, é importante a constante revisão dos testes e nos métodos utilizados na aplicação, para seguir as melhores práticas que garantem o código mais otimizado, limpo e legível entregando assim resultados satisfatórios aos clientes.

Referências

[1] Larry Loeb, «The Evolution of Digital Certificates», *Security Intelligence*, 25 de julho de 2017. https://securityintelligence.com/the-evolution-of-digital-certificates/ (acedido 9 de novembro de 2020).

[2] fortune business insights, «Digital Certificates Market Size, Industry Share and Growth Rate 2019-2025». https://www.fortunebusinessinsights.com/industry-reports/digital-certificates-market-100151 (acedido 1 de dezembro de 2020).

[3] cefdigital, «Introduction to e-signature», *CEF Digital*. https://ec.europa.eu/cefdigital/wiki/cefdigital/wiki/display/CEFDIGITAL/Introduction+to+e-signature (acedido 29 de maio de 2021).

[4] «ISO - International Organization for Standardization», *ISO*. https://www.iso.org/home.html (acedido 21 de junho de 2021).

[5] SGS SA, «ISO 9001 - Certificação - Sistemas de gestão da qualidade | SGS Portugal». https://www.sgs.pt/pt-pt/health-safety/quality-health-safety-and-environment/quality/quality-management-systems/iso-9001-certification-quality-management-systems (acedido 15 de dezembro de 2020).

[6] SGS SA, «ISO/IEC 27001:2013 – Sistemas de Gestão de Segurança da Informação | SGS Portugal». https://www.sgs.pt/pt-pt/industrial-manufacturing/quality-health-safety-and-environment/risk-assessment-and-management/security-management/iso-27001-2013-information-security-management-systems (acedido 29 de dezembro de 2020).

[7] SGS SA, «ISO 20000 – Sistema de Gestão de Tecnologias de Informação | SGS Portugal». https://www.sgs.pt/pt-pt/energy/quality-health-safety-and-environment/risk-assessment-and-management/security-management/iso-20000-it-certification (acedido 29 de dezembro de 2020).

[8] cefdigital, «Trusted List Browser». https://webgate.ec.europa.eu/tl-browser/#/tl/PT (acedido 1 de dezembro de 2020).

[9] cefdigital, «eSignature», *CEF Digital*. https://ec.europa.eu/cefdigital/wiki/cefdigital/wiki/display/CEFDIGITAL/eSignature (acedido 30 de dezembro de 2020).

[10] globaltrustedsign, «GTS - Global Trusted Sign https://www.globaltrustedsign.com/». https://www.globaltrustedsign.com/ (acedido 9 de novembro de 2020).

[11] iGEST, «iGEST», *iGEST - Emita faturas e guias de transporte com o seu telemóvel.* http://www.igest.pt/ (acedido 3 de dezembro de 2020).

[12] Paypay, «PayPay - Solução de Pagamentos». https://paypay.pt/paypay/sobreNos (acedido 3 de dezembro de 2020).

[13] Nexus Group, «Identity solutions | Nexus Group», *Nexusgroup*. https://www.nexusgroup.com/ (acedido 26 de janeiro de 2021).

[14] Nexus Group, «Nexus Smart ID - Apps on Google Play». https://play.google.com/store/apps/details?id=com.nexusgroup.personal.mobile&hl=en_US&gl=US (acedido 29 de janeiro de 2021).

[15] Nexus Group, «How to migrate to Nexus certificate authority CA software», *Nexusgroup*, 13 de julho de 2017. https://www.nexusgroup.com/migrate-to-nexus-certificate-authority-ca-software/ (acedido 29 de janeiro de 2021).

[16] «neXus releases IDentity Connector (IDC) 2.3», *Mynewsdesk*, 27 de janeiro de 2014. https://www.mynewsdesk.com/uk/nexus/news/nexus-releases-identity-connector-idc-2-3-88560 (acedido 20 de março de 2023).

[17] Nexus Group, «Nexus Group - YouTube». https://www.youtube.com/channel/UCnHABuq_JBmOyDIset58iFA (acedido 29 de janeiro de 2021).

[18] Cryptomathic, «Cryptomathic - Security Solutions». https://www.cryptomathic.com/ (acedido 26 de janeiro de 2021).

[19] Cryptomathic, «CRYPTOMAThIC - YouTube». https://www.youtube.com/channel/UCUwkfs2SpGELrj7hVdzuR6A (acedido 29 de janeiro de 2021).

[20] Cryptomathic, «EMV CA | Cryptomathic». https://www.cryptomathic.com/products/emv/emv-ca (acedido 29 de janeiro de 2021).

[21] Cryptomathic, «Signer Overview | Cryptomathic». https://www.cryptomathic.com/products/authentication-signing/signer-centralised-digital-signatures (acedido 29 de janeiro de 2021).

[22] Cryptomathic, «CKMS | Overview». https://www.cryptomathic.com/products/key-management/crypto-key-management-system (acedido 29 de janeiro de 2021).

[23] Ascertia, «High Trust PKI & Digital Signature Software Solutions». https://www.ascertia.com/ (acedido 26 de janeiro de 2021).

[24] Ascertia, «Servidor de assinatura ADSS - Solução de software de sinalização digital | Ascertia». https://www.ascertia.com/products/adss-signing-server/ (acedido 29 de janeiro de 2021).

[25] Multicert, «Multicert | Cibersegurança e Certificação Digital». https://www.multicert.com/pt/ (acedido 17 de janeiro de 2021).

[26] DigitalSign, «DigitalSign | Certificadora Digital». https://www.digitalsign.pt/pt (acedido 17 de janeiro de 2021).

[27] kgremban, «Tutorial – Noções básicas sobre a criptografia e os certificados X.509 para o Hub IoT do Azure». https://learn.microsoft.com/pt-br/azure/iot-hub/tutorial-x509-introduction (acedido 15 de janeiro de 2023).

[28] autenticacao.gov, «Autenticação europeia». https://www.autenticacao.gov.pt/outros-meios/autenticacao-europeia (acedido 3 de março de 2022).

[29] Adobe, «O que é a regulamentação eIDAS | Adobe Sign». https://www.adobe.com/pt/sign/compliance/eidas.html (acedido 3 de março de 2022).

[30] pranitbagmar, «What is OAuth (Open Authorization) | How does OAuth works», *Blog - miniOrange*, 20 de fevereiro de 2021. https://blog.miniorange.com/what-is-oauth-2/ (acedido 8 de julho de 2021).

[31] Internet Engineering Task Force e (IETF), «rfc6749». https://datatracker.ietf.org/doc/html/rfc6749 (acedido 29 de maio de 2021).

[32] «PHP: O que é o PHP? - Manual». https://www.php.net/manual/pt_BR/intro-whatis.php (acedido 8 de junho de 2021).

[33] Avi, «Docker vs Virtual Machine (VM) - Understanding the Differences», *Geekflare*, 15 de setembro de 2019. https://geekflare.com/docker-vs-virtual-machine/ (acedido 15 de janeiro de 2023).

[34] Docker, «Docker Desktop WSL 2 backend on Windows», *Docker Documentation*, 13 de janeiro de 2023. https://docs.docker.com/desktop/windows/wsl/ (acedido 15 de janeiro de 2023).

[35] Tableless, «Para iniciantes». https://tableless.github.io/iniciantes/manual/js/o-que-framework.html (acedido 3 de junho de 2021).

[36] O. I. Consultancy, «Best 10 Popular Web Development Frameworks for 2020», *Medium*, 13 de novembro de 2020. https://oneclickitconsultancy.medium.com/best-10-popular-web-development-frameworks-for-2020-5b59c3efac0c (acedido 7 de dezembro de 2020).

[37] Django Software Foundation, «The Web framework for perfectionists with deadlines | Django». https://www.djangoproject.com/ (acedido 28 de janeiro de 2021).

[38] Django Software Foundation, «Visão geral do Django | Django». https://www.djangoproject.com/start/overview/ (acedido 28 de janeiro de 2021).

[39] E. principal de blogs, «Top 3 Backend Frameworks for Web Development in 2020», *LeadBloging*, 13 de julho de 2020. https://www.leadbloging.com/top-3-backend-frameworks-for-web-development-in-2020/ (acedido 7 de dezembro de 2020).

[40] Spring, «The Spring Cloud suite of projects contains many of the services you need to make your applications run well in the cloud.» https://spring.io/cloud (acedido 13 de fevereiro de 2022).

[41] VMware, «Spring makes Java simple.», *Spring*. https://spring.io/ (acedido 28 de janeiro de 2021).

[42] expressjs, «Express - Node.js web application framework». https://expressjs.com/ (acedido 28 de janeiro de 2021).

[43] MDN, «Express/Node introduction - Learn web development | MDN». https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/Introduction (acedido 28 de janeiro de 2021).

[44] «Installation - Laravel - The PHP Framework For Web Artisans». https://laravel.com/docs/8.x (acedido 8 de junho de 2021).

[45] Oracle Corporation, «What is a database?» https://www.oracle.com/database/what-is-database/ (acedido 9 de dezembro de 2020).

[46] Sahiti Kappagantula, «SQL vs NoSQL Key Differences - MySQL vs MongoDB», *Edureka*, 16 de abril de 2019. https://www.edureka.co/blog/sql-vs-nosql-db/ (acedido 18 de janeiro de 2023).

[47] google, «Google Trends», *Google Trends*. https://trends.google.com/trends/explore?q=Oracle,%2Fm%2F04y3k,%2Fm%2F05ynw,%2Fm%2F05z_r2n,Microsoft%20SQL%20Server (acedido 10 de dezembro de 2020).

[48] HAILEY FRIEDMAN, «Oracle vs SAS vs PostgreSQL vs MySQL vs Microsoft Azure». https://improvado.io/blog/oracle-vs-sas-vs-postgresql-vs-mysql-vs-microsoft-azure (acedido 10 de dezembro de 2020).

[49] Oracle Corporation, «MySQL». https://www.mysql.com/ (acedido 29 de janeiro de 2021).

[50] Oracle Corporation, «Quickly Develop and Deploy Cloud Native Applications». https://www.oracle.com/mysql/ (acedido 30 de janeiro de 2021).

[51] P. G. D. Group, «PostgreSQL», *PostgreSQL*, 30 de janeiro de 2021. https://www.postgresql.org/ (acedido 30 de janeiro de 2021).

[52] PostgreSQL Global Development Group, «PostgreSQL: Sobre». https://www.postgresql.org/about/ (acedido 30 de janeiro de 2021).

[53] The GraphQL Foundation, «GraphQL | Uma linguagem de consulta para sua API». https://graphql.org/ (acedido 30 de janeiro de 2021).

[54] The GraphQL Foundation, «GraphQL Code Libraries, Tools and Services». https://graphql.org/code/#php (acedido 30 de janeiro de 2021).

[55] Lighthouse, «Lighthouse». https://lighthouse-php.com/ (acedido 30 de janeiro de 2021).

[56] Mikk Mihkel Nurges, «rebing/graphql-laravel: Laravel wrapper for Facebook's GraphQL». https://github.com/rebing/graphql-laravel (acedido 30 de janeiro de 2021).

[57] Docker, «Docker Hub - Container Image Library | Docker». https://www.docker.com/products/docker-hub (acedido 8 de abril de 2021).

[58] Ansgar Becker, «HeidiSQL - MariaDB, MySQL, MSSQL, PostgreSQL and SQLite made easy». https://www.heidisql.com/ (acedido 31 de março de 2021).

[59] Microsoft, «Visual Studio Code - Code Editing. Redefined». https://code.visualstudio.com/ (acedido 6 de abril de 2021).

[60] SmartBear Software, «The World's Most Popular API Testing Tool | SoapUI». https://www.soapui.org/tools/soapui/ (acedido 6 de abril de 2021).

[61] firecamp, «Firecamp, A campsite for developers», *Firecamp*. https://firecamp.app (acedido 7 de abril de 2021).

[62] Postman, «Postman | The Collaboration Platform for API Development», *Postman*. https://www.postman.com/ (acedido 7 de abril de 2021).

[63] Webhook.site, «About - Webhook.site Docs». https://docs.webhook.site/ (acedido 25 de janeiro de 2023).

[64] The GraphQL Foundation, «Introduction to GraphQL | GraphQL». https://graphql.org/learn/ (acedido 20 de abril de 2021).

[65] Mozilla Developer Network, «HTTP response status codes - HTTP | MDN», 17 de janeiro de 2023. https://developer.mozilla.org/en-US/docs/Web/HTTP/Status (acedido 9 de fevereiro de 2023).

[66] globaltrustedsign, «GTS Mobile ID – Apps no Google Play». https://play.google.com/store/apps/details?id=gts.gtsmobileid&hl=pt_PT&gl=PT (acedido 24 de maio de 2021).

[67] ACIN group, «Connecting to App Store». https://apps.apple.com/us/app/gts-mobile-id/id1410233324?platform=iphone&preserveScrollPosition=true#platform/iphone&platform=iphone (acedido 24 de maio de 2021).

[68] globaltrustedsign, «HUAWEI AppGallery». https://appgallery.huawei.com/#/app/C104046903 (acedido 24 de maio de 2021).

[69] globaltrustedsign, «GTS - Global Trusted Sign public_c/mobile». https://globaltrustedsign.com/public_c/mobile (acedido 9 de fevereiro de 2023).

[70] The PHP Group, «PHP: SoapClient - Manual». https://www.php.net/manual/en/class.soapclient.php (acedido 3 de dezembro de 2020).

[71] Paypay, «paypayue/paypay-soap». PayPay, 20 de novembro de 2020. Acedido: 3 de dezembro de 2020. [Em linha]. Disponível em: https://github.com/paypayue/paypay-soap

[72] «IEEE_SoftwareEngGlossary.pdf». Acedido: 3 de maio de 2021. [Em linha]. Disponível em: http://www.mit.jyu.fi/ope/kurssit/TIES462/Materiaalit/IEEE_SoftwareEngGlossary.pdf

[73] globaltrustedsign, «GTS - Global Trusted Sign public_c/faqs». https://www.globaltrustedsign.com/public_c/faqs (acedido 1 de junho de 2021).

[74] «O que é um diagrama entidade relacionamento?», *Lucidchart*. https://www.lucidchart.com/pages/pt/o-que-e-diagrama-entidade-relacionamento (acedido 4 de junho de 2021).

[75] «PHP-FIG — PHP Framework Interop Group - PHP-FIG». https://www.php-fig.org/ (acedido 9 de junho de 2021).

[76] «Error Handling | Lighthouse». https://lighthouse-php.com/master/digging-deeper/error-handling.html#user-friendly-errors (acedido 9 de junho de 2021).

[77] Sebastian Bergmann, «PHPUnit – The PHP Testing Framework». https://phpunit.de/ (acedido 23 de janeiro de 2021).

[78] Taylor Otwell, «Testando: Primeiros Passos - Laravel - O Framework PHP para Web Artisans». https://laravel.com/docs/8.x/testing (acedido 23 de janeiro de 2021).

[79] Taylor Otwell, «Laravel Dusk - Laravel - The PHP Framework For Web Artisans». https://laravel.com/docs/8.x/dusk (acedido 26 de janeiro de 2021).

[80] Selenium, «SeleniumHQ Browser Automation». https://www.selenium.dev/ (acedido 26 de janeiro de 2021).

[81] Acunetix, «Acunetix | Web Application Security Scanner», *Acunetix*. https://www.acunetix.com/ (acedido 27 de janeiro de 2021).

[82] Lighthouse, «Testing with PHPUnit | Lighthouse». https://lighthouse-php.com/5/testing/phpunit.html#testing-with-phpunit (acedido 18 de maio de 2021).

[83] endoflife, «PHP», *endoflife.date*. https://endoflife.date/php (acedido 7 de dezembro de 2022).

Anexos

Anexo A – Base de dados

Figura 17- Diagrama relacional da base de dados - Módulo Utilizadores 1ª versão.

Figura 18 - Diagrama relacional da base de dados - Módulo Utilizadores 2ª versão.

Figura 19 - Diagrama relacional da base de dados - Módulo Utilizadores 3ª versão.

Figura 20 - Diagrama relacional da base de dados - Gestão de pedidos 1ª versão.

Figura 21 - Diagrama relacional da base de dados - Gestão de pedidos 2ª versão.

Figura 22 - Diagrama relacional da base de dados - Gestão de pedidos 3ª versão.

Figura 23 - Diagrama relacional da base de dados - Gestão de pedidos 4ª versão.

Figura 24 - Diagrama relacional da base de dados - Gestão de pedidos 5ª versão.

Figura 25- Diagrama relacional da base de dados - Módulo Faturação, descontos e serviços - 1ª versão.

Figura 26 - Diagrama relacional da base de dados - Módulo Faturação, descontos e serviços - 2ª versão.

Figura 27 - Diagrama relacional da base de dados - Módulo Faturação, descontos e serviços - 3ª versão.

Figura 28 - Diagrama relacional da base de dados - Módulo Faturação, descontos e serviços - 4ª versão.

Figura 29 - Diagrama relacional da base de dados - Módulo de avaliação do cliente - 1ª versão.

Figura 30 - Diagrama relacional da base de dados - Módulo de avaliação do cliente - 2ª versão.

Figura 31 - Diagrama relacional da base de dados - Módulo de avaliação do cliente - 3ª versão.

Figura 32 - Diagrama relacional da base de dados - Módulo de avaliação do cliente - 4ª versão.

Anexo B – Implementação

Figura 33 - Diretórios da framework Laravel.

Figura 34 - Cast coluna com a tradução.

Figura 35 - Agendamento de tarefas.

Figura 36 - Estrutura de classes que contêm a lógica para as integrações.

Figura 37 - Facade da TSA (integração para os timestamps).

Figura 38 - Exceção do código.

Figura 39 - Diretiva nova de GraphQL.

Figura 40 - Mutations dos módulos para as ações de criar, editar e apagar.

Figura 41 - Query personalizada para mostrar os dados de um ficheiro CSR.

Figura 42 - Resolver com método para a atualização da password do utilizador.

Figura 43 – Validation dos dados para criação de uma fatura.

Figura 44 - Helper com método de encriptação e desencriptação de ficheiros para upload/download.

Figura 45 - Controllers.

Figura 46 – Middleware com o header Accept-Language para tradução da aplicação.

Figura 47 - FormRequest com a validação dos campos para completar um registo externo.

Figura 48 - Logging personalização.

Figura 49 - Configuração do log.

Figura 50 - Model Eloquent da tabela payments.

Figura 51 - Model Eloquent da tabela countries.

Figura 52 - Notification com mensagem de e-mail e registo na BD do bloqueio de conta.

Figura 53 - Provider de serviços.

Figura 54 - Provedor de autenticação.

```php
<?php

namespace App\Providers;

use App\Custom\Objects\Tsa\TsaProvider;
use Illuminate\Support\ServiceProvider;
use App\Custom\Classes\Invoices\Invoice;
use App\Custom\Classes\Payments\Payment;
use App\Repositories\Identity\TrustedXIdentity;

class RepositoryServiceProvider extends ServiceProvider
{
    /**
     * Register services.
     *
     * @return void
     */
    public function register()
    {
        // Identity service provider.
        $this->app->bind('identity provider', TrustedXIdentity::class);

        // Invoice service provider.
        $this->app->bind('invoice', Invoice::class);

        // Payment service provider.
        $this->app->bind('payment', Payment::class);

        // Time Stamp Authority service provider.
        $this->app->bind('tsa provider', TsaProvider::class);
    }

    /**
     * Bootstrap services.
     *
     * @return void
     */
    public function boot()
    {
        //
    }
}
```

Figura 55 - Provider das integrações.

Figura 56 - Implementação do trait com método para armazenamento de ficheiros.

Figura 57 - Ficheiros da pasta config.

```php
<?php

namespace Database\Factories\Invoice;

use App\Models\Invoice\Invoice;
use Illuminate\Database\Eloquent\Factories\Factory;
use App\Models\Invoice\District;
use App\Models\Common\Country;
use App\Models\Invoice\IvaExemption;
use Faker\Factory as FakerFactory;

class InvoiceFactory extends Factory
{
    /**
     * The name of the factory's corresponding model.
     *
     * @var string
     */
    protected $model = Invoiceclass;

    /**
     * Define the model's default state.
     *
     * @return array
     */
    public function definition()
    {
        $fakerPT = FakerFactory::create('pt_PT');

        return [
            'first_name' => $this->faker->name,
            'surname' => $this->faker->name,
            'fiscal_number' => $fakerPT->taxpayerIdentificationNumber,
            'email' => $this->faker->email,
            'phone' => $this->faker->numerify('9########'),
            'address' => $this->faker->streetName,
            'zip_code' => $this->faker->numerify('####-###'),
            'region' => $this->faker->state(),
            'request_id' => 1,
            'district_id' => $this->faker->randomElement(District::all()->pluck('id')),
            'country_id' => $this->faker->randomElement(Country::all()->pluck('id')),
            'iva_exemption_id' => $this->faker->randomElement(IvaExemption::all()->pluck('id')),
            'url' => null,
        ];
    }

    /**
     * Indicate that is a invoice without last name
     *
     * @return \Illuminate\Database\Eloquent\Factories\Factory
     */
    public function noLastName()
    {
        return $this->state(function (array $attributes) {
            return [
                'surname' => null,
            ];
        });
    }
```

Figura 58 -Definição de uma factory.

Figura 59 - Implementação da migration para a criação da tabela Users.

Figura 60 - Migração para alteração da tabela users.

Figura 61 - Seeder para a tabela certificate_types.

Figura 62 - Migração e seed da base de dados.

Figura 63 - View da recuperação de password.

Figura 64 - Ficheiros de traduções dos atributos.

Figura 65 - Ficheiro web.php.

Figura 66 – Ficheiro api.php.

Figura 67 - Diretório storage com os ficheiros carregados encriptados, logs por data (YYYY-MM-DD) e chaves de encriptação.

Figura 68 - Teste com asserções.

Figura 69 - Método de login.

Figura 70 - Callback do login e redireccionamento para o servidor de autorização.

Figura 71 - Troca do code pelo token.

Figura 72 - Bearer access_token.

Figura 73 - Documentação dos pedidos à API pelo Insomnia.

Figura 74 – GraphQL inputs, mutations, queries e types do recurso.

Figura 75 - Inputs do recurso de reviews.

Figura 76 - Mutations do recuso de reviews.

Figura 77 - Querys do recuso de reviews.

Figura 78 - Types do recuso de reviews.

Figura 79 – Ficheiro schema.graphql.