PM

GTS Front-end

PROJETO DE MESTRADO

José Maurício Teixeira Gouveia
MESTRADO EM ENGENHARIA INFORMÁTICA

UNIVERSIDADE da MADEIRA
A Nossa Universidade
www.uma.pt

novembro | 2022

GTS Front-end

PROJETO DE MESTRADO

José Maurício Teixeira Gouveia
MESTRADO EM ENGENHARIA INFORMÁTICA

ORIENTAÇÃO
Filipe Magno de Gouveia Quintal

COORIENTAÇÃO
Sofia Catarina Câmara Leme Pessanha de Meneses

Resumo

O presente relatório foi concebido no âmbito do projeto realizado na empresa Grupo ACIN, nomeadamente na plataforma GTS (Global Trusted Sign) que é prestadora de serviços de confiança, para a conclusão do curso de Mestrado em Engenharia Informática, da Universidade da Madeira.

Este projeto tem com principal objetivo a reestruturação da plataforma GTS, de modo a utilizar tecnologias mais recentes, adicionar novos requisitos, tal como a compra de diversos produtos em simultâneo, nomeadamente através de um carrinho de compras. Tem também o intuito de separar o front-end do back-end, uma vez que a plataforma antiga foi implementada seguindo uma estrutura MVC, o que irá melhorar o método de desenvolvimento.

Neste relatório irá ser abordado o processo de desenvolvimento seguido, desde a análise de requisitos, fluxogramas e protótipos, até à implementação das diversas áreas, assim como a elaboração dos testes automatizados da plataforma.

Será também discutido o produto final implementado, assim como os constrangimentos encontrados durante o desenvolvimento do mesmo e os requisitos que não foram possíveis alcançar atempadamente. O resultado, foi uma plataforma que permite a compra e geração de certificados digitais, que serão posteriormente utilizados para assinar digitalmente documentos.

Palavras-chave: Aplicação Web, Back-end, Front-end, RestAPI, React, Processo de Desenvolvimento de Software, Engenharia de Software, Métodos Ágeis, Testes unitários, Testes de integração

Abstract

This report was conceived within the project carried out in the company Grupo ACIN, namely in the GTS (Global Trusted Sign) platform, which is a provider of reliable services, for the completion of the master's course in Computer Engineering, at the University of Madeira.

This project has as main objective of restructuring the GTS platform, to use the latest technologies, add new requirements, such as the purchase of several products at the same time, namely through a shopping cart. It is also intended to separate the frontend from the backend, since the old platform was implemented following an MVC structure, which will improve the development method.

This report will address the development process followed, from the analysis of requirements, flowcharts, and prototypes to the implementation of the various areas, as well as the elaboration of the automated tests of the platform.

The final product implemented will also be discussed, as well as the constraints encountered during its development and the requirements that were not possible to achieve in time. The result was a platform that allows to purchase and generate digital certificates, which will later be used to digitally sign documents.

Keywords: Web Application, Backend, Frontend, RestAPI, React, Software Development Process, Software Engineering, Agile Methods, Unit Tests, Integration Tests

Acrónimos

AES – Advanced Encryption Standard

API – Application Programming Interface

CA – Certificate Authority

CRL – Certificate Revocation List

CSR – Certificate Signing Request

DES – Data Encryption Standard

DSA – Digital Signature Algorithm

ECC – Elliptic Curve Cryptography

GTS – Global Trusted Sign

MD – Message Digest

MVC – Model-View-Controller

OCSP – Online Certificate Status Protocol

OTP – One Time Password

PKI – Public Key Infrastructure

RC – Rivest Cypher

RSA – Rivest-Shamir-Adleman

SHA – Secure Hash Algorithm

SPA – Single Page Application

SSL – Secure Sockets Layer

TX – TrustedX

VA – Validation Authority

“Your work is going to fill a large part of your life, and the only way to be truly satisfied is to do what you believe is great work. And the only way to do great work is to love what you do. If you haven't found it yet, keep looking. Don't settle. As with all matters of the heart, you'll know when you find it.”

– Steve Jobs

Aos meus pais e irmãs que sempre me apoiaram nesta jornada

Agradecimentos

Gostaria de agradecer ao meu orientador Filipe Quintal por toda a paciência, apoio e disponibilidade demonstrada e pelos conselhos que orientaram na conclusão desta tese.

Quero agradecer também ao Grupo ACIN e à minha coorientadora Sofia Meneses pela oportunidade proporcionada na elaboração deste projeto.

Quero também agradecer aos colegas da plataforma GTS por toda a ajuda e disponibilidade, na da partilha de experiências, momentos e discussões, contribuindo para este projeto.

Por último, mas não menos importante quero agradecer aos meus pais e irmãs, restantes familiares e amigos por todo o apoio incondicional, incentivo e compreensão, e por proporcionarem um ambiente de motivação que me levou ao desfecho desta etapa.

Índice

Lista de Tabelas

Lista de Figuras

1. Introdução

Durante milhares de anos, desde o início da civilização, que as pessoas com um enorme poder na sociedade, tais como reis, rainhas, generais, tiveram a necessidade de efetuar comunicações eficientes de modo a governar com segurança as suas propriedades e exércitos. Era fundamental que esta informação não caísse nas mãos erradas, uma vez que poderia colocar em perigo o seu património e povo. Foi o receio desta ameaça constante, que motivou o impedimento de terceiros conseguirem visualizar de forma lógica e percetível o conteúdo da informação. Por sua vez, levou ao desenvolvimento de técnicas para disfarçar e esconder os dados de carácter confidencial, de modo que apenas o emissor e o destinatário tivessem acesso à informação.

Uma das primeiras abordagens utilizadas para "esconder" a informação, teve origem no século 5 aC, sendo utilizada por Histiaeus para enviar uma mensagem de ataque do exército persa, rapou a cabeça de um servo e tatuou a mensagem no couro cabeludo do mesmo. Aguardou que o cabelo voltasse a crescer e enviou-o então para o destinatário da informação, com a instrução de que para ter acesso a esta, deveria voltar a rapar o couro cabeludo do servo para conseguir ler a mensagem. [1]

Alguns anos mais tarde, ocorreu outro exemplo, no ano de 480 aC quando o rei espartano Demarato precisava enviar informações para a sua terra Esparta, pois tinha sido exilado da Grécia. De modo a conseguir fazer chegar a informação sobre um ataque persa, e para precaver que a mensagem fosse intercetada pelos guardas persas, Demerato rapou a cera de umas tábuas de madeira, escreveu a mensagem pretendida e voltou a cobrir as tábuas de madeira com cera. Desta forma, as tábuas de madeira conseguiram passar pelos guardas persas ao longo da estrada sem causar qualquer problema. Quando chegou ao destino, a rainha Gorgo mandou rapar a cera e conseguiu ler a mensagem enviada. [1]

Os exemplos anteriormente mencionados utilizam uma técnica denominada por esteganografia. [2] Esta deriva de duas palavras gregas: *steganos* que significa coberto e *graphien* que significa escrever. Tal como estas palavras indicam, a estenografia tem como objetivo ocultar uma determinada informação, tal como a existência de uma mensagem.

Outra técnica semelhante é a criptografia, sendo que ambas têm o objetivo de "esconder" a informação, contudo são realizadas de maneiras distintas. A criptografia oculta o significado da mensagem, dando enfâse à privacidade, enquanto a esteganografia como já referido oculta a existência de qualquer informação. Este trabalho foca-se então na criptografia e na sua utilização nos dias que correm.

Denomina-se de criptografia a ciência que estuda a arte de comunicar confidencialmente. [1] A criptografia também deriva de duas palavras gregas: *kryptos*, que significa oculto e *graphein* que significa escrever, ou seja, foi criada para proteger informações sensíveis do acesso não autorizado de terceiros, isto é, de pessoas que não são o remetente nem o destinatário dessas informações e dados. Esta serve não apenas para assegurar a confidencialidade do conteúdo de mensagens, dados e informações, mas também a integridade (contra alterações do conteúdo) e autenticidade. [3] Neste processo temos presente a encriptação da informação, que é a transformação de forma reversível dos dados de forma a torná-los ilegíveis, e a desencriptação que é o processo inverso para voltar a colocar os dados legíveis.

Após a descrição do conceito referido anteriormente, é possível identificar que a mesma pode ser divida em dois tipos, criptografia simétrica e criptografia assimétrica, das quais irão ser posteriormente exemplificadas. Dentro da criptografia simétrica, é efetuada a diferenciação entre cifra de substituição e cifra de transposição. [4]

Os sistemas mais simples de criptografia, eram elaborados através da reorganização do alfabeto, sendo este a chave que possibilitava a descodificação da informação presente na mensagem. Para desencriptar o conteúdo seria necessário verificar o padrão utilizado na repetição dos caracteres e equiparar com a frequência das palavras de um certo idioma. [5]

1.1 Criptografia Antiga

1.1.1 Hieróglifos

O exemplo mais antigo conhecido da criptografia tem perto de 4.000 anos, tendo acontecido por volta do ano 1900 aC, segundo Nicholas McDonald, do Departamento de Engenharia Elétrica e de Computação da Universidade de Utah. [6] Este ocorreu na cidade egípcia de Beni Hasan no túmulo de Khnumhotep II, quando o seu escriva fazia a decoração do mesmo, ao desenhar hieróglifos que representavam a vida do seu mestre. Os hieróglifos eram o estilo de escrita utilizado pelos egípcios, onde os símbolos, denominados por glifos representavam as diferentes ideias, palavras ou letras. [7] Em uma parte específica da decoração, o escriva substituiu os símbolos hieróglifos normais por outros novos que pareciam não fazer qualquer sentido.

Não se sabe ao certo o motivo pelo qual este procedimento foi realizado, contudo existem várias teorias para esta ação. Uma delas tem a ver com o fato de que apenas as classes altas do Egipto sabiam ler e escrever, assim ao tornar o processo mais difícil de entender, seria possível utilizar este ato como um mecanismo de proteção, de modo a esconder as mensagens e rituais sagrados das pessoas “comuns” que poderiam ter aprendido o que um determinado glifo poderia simbolizar, uma vez que muitos dos textos poderiam ser religiosos, assim como dos ladrões de túmulos. [8] Existe também a teoria que isto poderia ser um divertido enigma para as pessoas resolverem, de modo a instigá-las a ler os hieróglifos. [7]

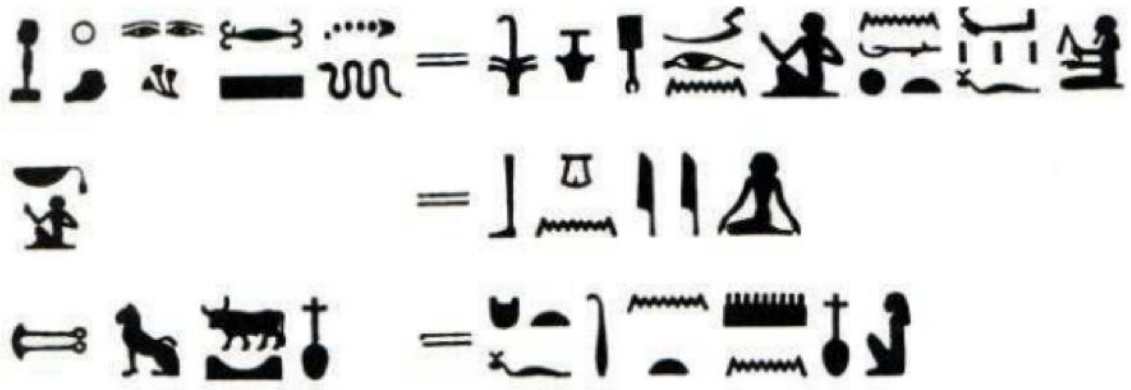


Figura 1 - A mensagem oculta de MenetKhufu [6]

1.1.2 Scytale ou Bastão de Licurgo

No ano 500 aC temos outro registo de criptografia na cidade grega de Esparta, onde os gregos não se limitaram apenas a substituir alguns símbolos aqui ou ali, mas desenvolveram um dispositivo para efetuar a criptografia. Uma vez que os espartanos estavam continuamente em guerras de modo a conquistar novas terras e proteger o território atual, era necessário um meio para efetuar e receber comunicações de forma segura e confidencial. Foi desta necessidade que surgiu então o scytale ou bastão de Licurgo, que era bastão de madeira com um pergaminho de papel fino enrolado em forma espiral. [7]

O primeiro passo seria enrolar o pergaminho à volta do scytale e, em seguida, proceder à escrita longitudinal da informação que iria ser transmitida. Após a conclusão da mensagem, o pergaminho era desenrolado e a informação ficava incompreensível. Assim, caso fosse intercetado por outra pessoa que não o destinatário, esta não saberia o significado do que estava escrito no pergaminho. Para o destinatário conseguir interpretar a mensagem teria de possuir um bastão com o mesmo comprimento e diâmetro, de modo a alinhar todas as letras, decifrando assim o seu conteúdo.

O scytale forneceu aos espartanos um meio seguro de comunicação e foi então, um dos primeiros exemplos da denominada cifra de transposição. Neste tipo de cifras, a ordem das letras ou símbolos é trocada, sendo possível decifrar o procedimento invertendo o processo [7]

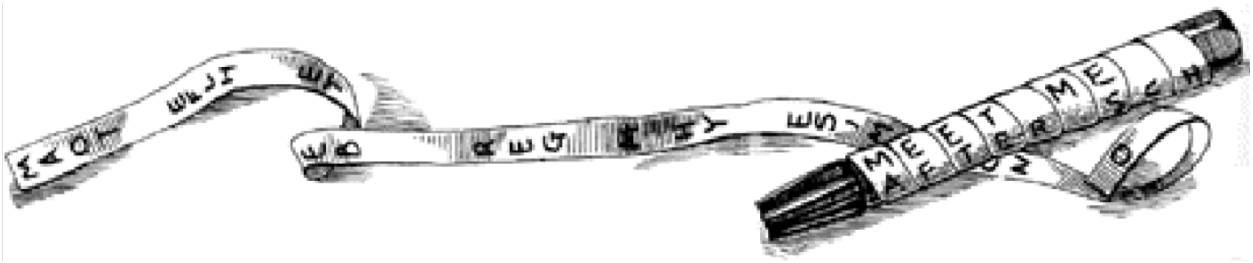


Figura 2 - Scytale ou Bastão de Licurgo [9]

1.1.3 Cifra de César

Uma vez que não eram apenas os espartanos que estavam constantemente em guerra, também Júlio César, o imperador romano, precisou de arranjar uma alternativa para comunicar com os seus oficiais até ao seu assassinato em 44 a.C. Assim, entre 60 e 50 aC, as tropas de César comandadas por Cícero estavam em apuros e perto de se render, contudo César precisava avisá-los que chegaria em breve e eles deveriam resistir. Com receio que as tropas inimigas intercetassem a sua mensagem e soubessem que os ia ajudar, eliminando o fator surpresa, César criou então uma técnica de substituição.

Esta consistia na substituição de cada letra pela terceira letra que a acompanha no alfabeto, tal como exemplificado na tabela seguinte.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

Tabela 1 - Cifra de César [10]

Susan Meyer [7] indica o procedimento a seguir para resolver uma frase criptografada com a cifra de César, sendo que o principal objetivo é descobrir o número de lugares que as letras foram deslocadas. O primeiro passo é escrever a mensagem, e de seguida escreve-la novamente substituindo cada letra pela letra correspondente duas casas à frente. Caso a frase ainda não faça sentido, o processo deve ser repetido, avançando uma casa em cada letra até as palavras começarem a ser familiares. Depois de ser possível ler corretamente uma palavra, é possível descobrir então o número de lugares que as letras foram deslocadas e aplicar a toda a mensagem. É possível ver abaixo o exemplo elaborado por Susan.

Número de lugares deslocados	**Mensagem**
0	LIPPS XLIVI
1	KHOOR WKHUH
2	JGNNO VJGTG
3	IFMMP UIFSF
4	HELLO THERE

Tabela 2 - Resolução da Cifra de César [7]

Este é assim um exemplo de uma cifra de substituição, uma vez que a partir da informação original, cada caractere é trocado por outro caractere, que pode ser de diferentes alfabetos, desde os conhecidos aos inventados. [7]

1.2 Criptografia Moderna

1.2.1 Código Morse

Em 1835 foi desenvolvido por Samuel Morse, o inventor do telégrafo, o código Morse que é um alfabeto cifrado em sons, sinais elétricos, ondas sonoras ou sinais visuais curtos e longos. Esta técnica pode então ser utilizada com outro método criptográfico para aumentar a segurança. Embora atualmente possa ser um sistema obsoleto, foi amplamente utilizado durante o século XX.

A	·–	J	·–––	S	···	2	··–––
B	–···	K	–·–	T	–	3	···––
C	–·–·	L	·–··	U	··–	4	····–
D	–··	M	––	V	···–	5	·····
E	·	N	–·	W	·––	6	–····
F	··–·	O	–––	X	–··–	7	––···
G	––·	P	·––·	Y	–·––	8	–––··
H	····	Q	––·–	Z	––··	9	––––·
I	··	R	·–·	1	·––––	0	–––––

Figura 3 - Código Morse [58]

1.2.2 Enigma

Até à revolução industrial, a maioria dos sistemas criptográficos eram limitados a cálculos feitos à mão ou com dispositivos simples. Após este período e com a chegada das grandes guerras mundiais, houve a necessidade de desenvolver sistemas mais eficazes de comunicação segura.

Um dos exemplos com mais êxito e mais conhecido foi a máquina Enigma. Esta foi desenvolvida por Arthur Scherbius em 1918 e utilizada pelo exército alemão durante a Segunda Guerra Mundial, dada a necessidade de comunicarem com as tropas e a marinha.

Esta máquina era composta por cinco cilindros numerados, cada um com 26 posições que representavam cada letra do alfabeto. [11] Dentro da máquina apenas era inserido três dos cinco cilindros. Os soldados tinham um livro, onde tinham as configurações da máquina para cada dia do mês, para todos os dias a maneira de desencriptar o código ser diferente e todos os meses recebiam um novo livro com as configurações de cada dia desse mês. Apesar de todos os esforços, em 1933 Alan Turing conseguiu desencriptar a Enigma. [12]

Figura 4 - Máquina Enigma [59]

1.3 Criptografia Digital

Até à Segunda Guerra Mundial a maior parte das técnicas criptográficas eram apenas utilizadas com fins militares. Contudo após a guerra, as empresas ficaram "atraídas" pela criptografia, de modo a conseguirem proteger os seus dados das empresas rivais. Foi aí que no início dos anos 70, a IBM juntou um grupo de pessoas que desenvolveram o DES – *Data Encryption Standard* – como algoritmo padrão criptográfico.

No presente, a essência da criptografia mantem-se. Contudo de uma forma mais sofisticada e complexa com fórmulas matemáticas, os algoritmos, aplicados na codificação da informação.

Segundo Simon Singh, "Já foi dito que a Primeira Guerra Mundial foi a guerra dos químicos, porque o gás mostarda e o cloro foram empregues pela primeira vez, e que a Segunda Guerra Mundial foi a guerra dos físicos, porque a bomba atômica foi detonada. Da mesma forma, tem sido argumentado que a Terceira Guerra Mundial seria a guerra dos matemáticos, porque os matemáticos terão controle sobre a próxima grande arma de guerra - a informação". [13]

A criptografia é muito mais do que apenas "dificultar" a leitura da informação, tornou-se um enorme negócio no século XXI. Há medida que vamos cada vez mais utilizando a "cloud" e que muitos serviços estão lá alojados, assim como as comunicações digitais que efetuamos, é necessário criptografar toda a informação de modo a não ser acessível. Os exemplos mais simples, mas também os que mais utilizamos, é por exemplo os emails, ou as redes sociais como o *Telegram* ou *WhatsApp*.

Um dos casos mais mediáticos da atualidade, envolvendo a criptografia, aconteceu a 2 de Dezembro de 2015 num dos piores tiroteios nos Estados Unidos da América. O casal Syed Rizwan Farook e a sua esposa Tashfeen Malik atiraram e mataram 14 pessoas na cidade californiana de San Bernardino, e deixaram aproximadamente outras 21 pessoas feridas. [14]

O casal colocou-se em fuga após esses atos, contudo foram mais tarde intercetados e abatidos pela polícia. No decorrer da investigação, o FBI pretendia obter mais informações sobre o que levou a estes atos, assim como possíveis futuros acontecimentos semelhantes caso existissem cúmplices. O FBI tentou proceder ao desbloqueio do iPhone 5C que era propriedade do Farook, contudo sem sucesso. Como alternativa, solicitaram à Agência de Segurança Nacional (NSA) que tentasse invadir o smartphone, mas também não obtiveram êxito devido aos recursos de segurança avançados implementados pela Apple no equipamento.

Assim, o FBI achava que a única solução seria a colaboração da Apple para desenvolver uma nova versão do sistema operativo iOS presente no equipamento, que após a sua instalação desativasse os recursos de segurança de modo a permitir o acesso ao smartphone pelo FBI e assim conseguir obter toda a informação presente no mesmo.

Contudo a Apple recusou tal pedido, nomeadamente o seu CEO Tim Cook, uma vez que isto iria criar um grande precedente e seria uma enorme violação da privacidade dos seus clientes e subscrevendo que "Os mesmos engenheiros que criaram uma forte criptografia no iPhone para proteger nossos utilizares seriam, ironicamente, ordenados a enfraquecer essas proteções e tornar nossos utilizares menos seguros."

Com esta decisão por porte da Apple, o FBI não teve outra solução senão procurar outra alternativa. Esta recaiu sobre a contratação de hackers que conseguiram encontrar vulnerabilidades

na criptografia do sistema operativo iOS e aceder então a todos os dados presentes no equipamento, contudo não haveria nada que fosse importante para a investigação.

Este caso mostra-nos que por mais que um sistema de criptografia seja aprimorado, existe sempre a possibilidade do mesmo ser inspecionado ao pormenor, de modo a verificar onde pode existir uma falha que permita conceder o acesso à informação. Tanto os criadores de um mecanismo criptográfico como quem tenta decifrar o sistema, recorrem a várias técnicas para ajudar no processo, desde a matemática à tecnologia, nomeadamente com a ajuda do computador. Este esforço por parte de ambos, seja para de um lado ocultar a informação e por outro de tentar descobrir as informações que a mesma contém, são um grande enriquecimento para o desenvolvimento tecnológico.

E uma vez que atualmente os negócios são cada vez mais efetuados pela Internet, principalmente neste momento que enfrentamos uma grande pandemia, as medidas de segurança implementadas para proteger as empresas e os clientes devem ser cada vez mais aperfeiçoadas.

A criptografia é então a maneira mais segura de proteger os nossos dados e este trabalho tem como objetivo desenvolver uma plataforma de venda de certificados digitais. Estes não ocultam a informação, como foi demonstrado na maioria dos casos até agora, mas permitem autenticar os dados que um emissor pretende transmitir ao destinatário, garantindo que não haverá nenhum intermediário que efetua alterações na informação.

1.4 ACIN

Este projeto faz parte de uma das plataformas criadas pelo Grupo ACIN, nomeadamente a GTS (Global Trusted Sign). Sendo assim, daqui em diante, iremos nos referir à GTS como a entidade na qual este trabalho está englobado.

O Grupo ACIN conta com mais de 20 anos de experiência na área das Tecnologias da Informação, tendo como objetivo a criação e manutenção de plataformas na *cloud* e foi criado em 1999. Fazem parte do Grupo ACIN as seguintes plataformas:

- GTS – prestadora de serviços de confiança, disponível em globaltrustedsign.com
- acinGov – plataforma eletrónica de compras públicas, disponível em acingov.pt
- iParque – software de gestão de estacionamentos, disponível em iparque.pt
- iMed – software de gestão clínica, disponível em imed.pt
- iGest – software de faturação, disponível em igest.pt
- Compras do Estado – oportunidades de negócio, disponível em comprasdoestado.pt
- iFreg – software de gestão de freguesias, disponível em ifreg.pt
- iDok – software de gestão documental, disponível em idok.pt
- iRH – software de gestão de assiduidade, disponível em irh.pt
- PayPay – solução de pagamentos, disponível em paypay.pt
- Ilink – EDI (Electronic Data Interchange) e faturação eletrónica, disponível em ilink.pt
- omeutáxi – software de faturação para motoristas de táxi, disponível em omeutaxi.pt

1.4.1 Serviços de criptografia na ACIN

A GTS é "uma prestadora de serviços de confiança, credenciada em Portugal pelo Gabinete Nacional de Segurança e implementada em conformidade com o Regulamento EU N.º 910/2014, que irá colocar ao seu dispor 4 serviços de confiança, nomeadamente Selos Temporais, Certificados Qualificados de Assinatura Digital, Certificados Qualificados de Selos Eletrónicos e Certificados de Autenticação de sítios web". [15]

Ou seja, a GTS como plataforma pertencente ao Grupo ACIN, está certificada "como prestadores de serviços de confiança, credenciada pelo Gabinete Nacional de Segurança e implementada em conformidade com o Regulamento eIDAS". [16]

Relativamente aos produtos disponibilizados pela GTS, os Selos Temporais garantem a data e hora de um evento, tal como a criação, envio ou receção de um documento digital, garantindo que o documento existia naquele determinado momento e provando que não foi modificado.

Os certificados SSL, autenticam um domínio web, agregando-o a uma pessoa singular ou coletiva, sendo esta a detentora do certificado emitido. De uma forma simplificada, os certificados SSL são o motivo de quando vamos a um website e verificamos que ao lado do endereço aparece um "cadeado verde". É utilizado principalmente por empresas e utilizadores que pretendem legitimar os seus domínios web, transmitindo uma maior confiança a quem navega nos seus websites. Aumentam também a segurança dos domínios, o que ajuda na prevenção de ataques phishing.

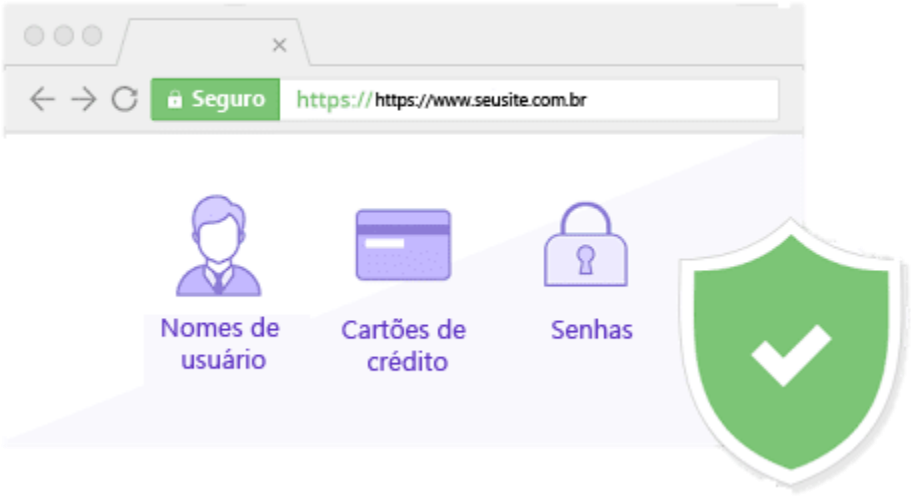


Figura 5 – Certificado SSL [17]

Por sua vez os Selos Eletrónicos destinam-se a pessoas coletivas. Estes podem ser equiparados ao carimbo de uma empresa ou o selo branco de um organismo público. Este certificado é semelhante à assinatura eletrónica, contudo apenas pode ser emitido para empresas e

não por pessoas singulares. Pode ser utilizada em documentos digitais, assinaturas de email, declarações eletrónicas, etc.

Por fim temos a Assinatura Eletrónica, já referida no ponto anterior que confirmam a identidade do seu proprietário, seja pessoa singular (designando-se por Certificado Digital Qualificado Singular) ou coletiva (Certificado Digital Qualificado Coletivo). Para utilizadores pertencentes a uma ordem profissional existem os Certificado Digital Qualificado Profissional que também pode ser singular ou coletivo. A Assinatura Eletrónica pode ser utilizada para submeter propostas de compras públicas, submeter documentos em portais do Estado, representar a sua organização com valor probatório, assinar emails como representante de uma organização, etc.

1.5 Problema

Sendo a GTS uma aplicação de venda centralizada de certificados digitais, precisa de estar constantemente atualiza com as novas tecnologias e produtos que fazem parte da criptografia.

Quando a plataforma da GTS foi desenvolvida, nem todos os requisitos e funcionalidades foram devidamente explícitos. Isto levou a que com o passar do tempo fosse necessário estar constantemente a adaptar a plataforma com as novas necessidades, o que limita o seu desenvolvimento.

O desenvolvimento anterior foi realizado segundo o padrão de arquitetura de software MVC com a framework CodeIgniter. O padrão MVC é um acrónimo de Model-View-Controller ou Modelo-Vista-Controlador e foi criado por Trygve Reenskaug, que descreveu como deve ser efetuado a sua implementação no seu artigo “Applications Programming in Smalltalk-80: How to use Model-View-Controller”. [18] Utilizando este padrão é possível dividir o projeto em responsabilidades bem distintas.

O modelo é o responsável por receber os pedidos do controlador e responder ao mesmo, é responsável pela parte lógica e por efetuar o CRUD (*create, read, updade, delete* – em português criar, consultar, atualizar e apagar) à informação presente na base de dados. Por sua vez a vista é a responsável pela apresentação dos dados ao utilizador, tais como os resultados do modelo, imagens, tabelas, formulários, etc. É esta que vai interagir com o utilizador e também recolher os dados por ele inseridos e enviar para o controlador. O controlador é então quem define o comportamento da aplicação, gerindo o modelo e a vista, assim como a transição entre ambos, e verificando as ações efetuadas pelo utilizador.

Uma das desvantagens deste padrão relaciona-se com o fato de ser necessário mais tempo para modelar a aplicação e por consequente mais tempo para desenvolver uma nova funcionalidade. Para o programador não existe distinção de *back-end* (programador responsável por implementar a parte lógica e comunicações da aplicação) e *front-end* (programador responsável por implementar a parte visual da aplicação), sendo que quem desenvolve é *full-stack* (programador que implementa tanto a parte lógica como a parte visual da aplicação). Este é assim um dos motivos para renovar a aplicação em outra linguagem.

A plataforma atual permite um modelo de negócio de revendedores que bonifica quem publicitar com sucesso o produto. Contudo, o desenvolvimento atual obriga a que todo o processo seja efetuado diretamente no site.

Um dos maiores entraves da plataforma neste momento, deve-se ao facto de não ser possível efetuar a compras de vários produtos em simultâneo. Assim sendo, se o utilizador pretender adquirir mais que um produto, terá de preencher a documentação várias vezes, na maioria dos casos com a mesma informação, assim como efetuar pagamentos separados.

Além disso, os testes automatizados não foram implementados na plataforma anterior, o que aumenta a possibilidade de ocorrerem erros inesperados, que poderiam ser detetados antecipadamente.

Por último, a plataforma existente possui credenciais de autenticação para o PKI de assinatura de documentos e outras credenciais de autenticação para a parte privada da própria plataforma, o que confunde o utilizador sobre os seus próprios dados.

1.6 Solução

De modo a resolver os problemas apontados no ponto anterior e após alguma ponderação, chegou-se à conclusão de que o mais assertivo seria o planeamento de uma nova plataforma que colmatasse as fragilidades da plataforma anterior.

De salientar que durante a elaboração deste projeto, será implementado o máximo de funcionalidades possíveis previstas, contudo é importante realçar que é pretendido no mínimo deixar o processo de compra dos produtos completamente funcional.

Foram elaboradas diversas reuniões com todos os setores envolvidos, desde o departamento de contabilidade, o departamento de apoio ao cliente, o departamento de marketing, o departamento de design, o gestor de produto e a administração para elaborar os requisitos necessários a um *update* da plataforma, de acordo com a experiência adquirida pela plataforma atual e do feedback do departamento de apoio ao cliente, consoante o feedback dos mesmos.

A solução passa então pela construção de uma nova plataforma recorrendo às novas tecnologias, de modo a obter uma aplicação rápida, eficaz, de fácil manutenção e que seja prática a adição constante de novas funcionalidades. Para tal procedimento, irá ser feito o estudo de diversas ferramentas e tecnologias de modo a averiguar a mais adequada para a elaboração da nova GTS, de modo a colmatar as falhas anteriormente referidas.

A sua construção seguirá um rigoroso processo de desenvolvimento e modelação de software. Este deverá incluir o levantamento e análise dos requisitos, levantamento das tecnologias a utilizar (como já referido anteriormente), arquitetura utilizada, processo de design (desde os protótipos de baixa fidelidade aos *mockups*), planeamento das tarefas, implementação, testes unitários, de integração e de usabilidade, implantação, documentação, etc.

Após todos estes procedimentos, será expectável que a plataforma elaborada seja de fácil adição de novas funcionalidades, que todas as componentes tenham testes unitários e de integração de modo a prevenir o maior número de “bugs” possíveis, que seja possível uma melhor gestão dos revendedores e que seja possível efetuar a compra de vários produtos em simultâneo. Espera-se também que apenas exista umas credenciais para acesso a toda a plataforma, de modo a ser mais prático para o utilizador final.

1.7 Estrutura da tese

Esta tese está organizada em sete capítulos que visam demonstrar o trabalho efetuado ao longo do desenvolvimento deste projeto.

No **capítulo um**, temos presente a introdução da tese, é apresentada a entidade ACIN para a qual este projeto foi desenvolvido, são referidos e expostos os problemas que a entidade identificou e é apresentada uma possível solução que pretende dar resposta a esses problemas.

No **capítulo dois** temos a revisão de literatura que engloba os algoritmos de encriptação, explica o que é uma infraestrutura de chaves públicas e é efetuada uma análise da concorrência da GTS. É explicado também o processo de engenharia de software e é feito um estudo sobre quais as tecnologias a serem utilizadas neste projeto, onde no fim é descrita qual a opção escolhida e que bibliotecas serão utilizadas.

No **capítulo três** damos início à solução e é demonstrado todo o processo da engenharia de software seguido, desde os requisitos à prototipagem.

No **capítulo quatro** é demonstrado como foi efetuada a implementação do projeto, passando pela forma como a aplicação comunica com a API e efetua a autenticação, o formulário de compra, a implementação do carrinho de compras, o pagamento dos produtos e a geração dos certificados digitais.

No **capítulo cinco** é apresentado os testes implementados na aplicação, sejam eles unitários ou de integração, o *code coverage* (percentagem de código que possui testes) e os testes manuais.

No **capítulo seis** é apresentada a discussão do trabalho efetuado, passando pelos problemas apresentados inicialmente e tentando dar resposta aos mesmos. São também enunciados os requisitos que por algum motivo ficaram em falta e os problemas encontrados no decorrer da elaboração do projeto.

Por fim, no **capítulo sete** é apresentada a conclusão deste projeto, a experiência que foi a elaboração do mesmo e as perspetivas futuras.

No final do documento é possível encontrar os anexos que contêm informação adicional para complementar o trabalho apresentado nos diversos capítulos desta tese.

2. Revisão de Literatura

Nesta secção vamos abordar os algoritmos e falar sobre que tipos de algoritmos existem, qual o seu propósito e como são utilizados nas assinaturas digitais. Seguidamente será explicado o que é uma infraestrutura de chaves públicas e como estão os certificados estão os certificados ligados a estas.

Será também apresentada uma análise da concorrência, analisando as tecnologias utilizadas por cada uma, assim como os produtos comercializados. Posteriormente será falado sobre a Engenharia de Software e descrito o processo e as metodologias existentes.

Por fim, será feito um estudo sobre as tecnologias possíveis de serem utilizadas neste projeto, assim como as suas vantagens e desvantagens, e que bibliotecas irão ser utilizadas de modo a agilizar a implementação do projeto.

2.1 Algoritmos de encriptação

Atualmente quem pretende desencriptar uma certa informação, não consegue efetuar esta tarefa apenas observando os padrões ou sem a ajuda de "supercomputadores". Ou seja, para efetuar este processo é necessário conhecimentos avançados em matemática para compreender os algoritmos utilizados e uma grande potência computacional.

Um algoritmo é uma função matemática complexa que executa um conjunto de procedimentos para efetuar um cálculo, ou seja, uma sequência de instruções, até que uma determinada condição se verifique. [19]

Existem vários tipos de algoritmos relevantes para a criptografia, contudo será apresentado abaixo os que consideramos mais importantes no âmbito desta tese.

2.1.1 Algoritmos simétricos

Na criptografia existem dois algoritmos principais, os assimétricos e os simétricos [6][7]. Os algoritmos simétricos, também conhecidos como criptografia de chave secreta, utilizam a mesma chave, também designada de chave mestra, para encriptar e desencriptar a mensagem, sendo então necessário que todas as partes tenham conhecimento da chave. A manipulação dos dados é mais rápida que nos algoritmos assimétricos pelo facto de que normalmente utilizam chaves mais curtas, contudo, devido ao facto da chave ser partilhada, não são tão seguros como os algoritmos assimétricos. Os algoritmos que utilizam criptografia simétrica mais conhecidos são: DES (Data Encryption Standard), 3DES, AES (Advanced Encryption Standard) e RC4 (Rivest Cypher).

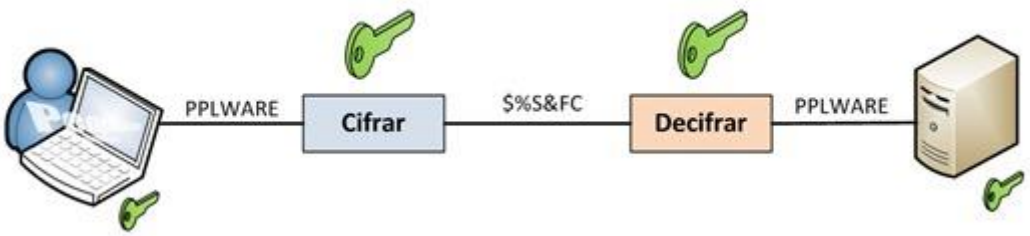


Figura 6 - Algoritmo simétrico [20]

2.1.2 Algoritmos assimétricos

Por sua vez, os algoritmos assimétricos, também conhecidos por criptografia de chave pública, utilizam uma chave pública para encriptar uma mensagem e uma chave privada para efetuar a desencriptação. A operação executada por uma das chaves, só pode ser revertida pela outra. Enquanto a chave privada é mantida em sigilo, só sendo conhecida pelo seu dono, a chave pública é de acesso geral, disponibilizada e tornada acessível a qualquer indivíduo que pretenda efetuar comunicações com o proprietário da chave privada correspondente.

Os algoritmos assimétricos comparativamente com os algoritmos simétricos, são mais lentos e envolvem maior poder computacional, contudo são extremamente seguros. [8] Os algoritmos assimétricos mais conhecidos são o RSA (Rivest-Shamir-Adleman), o DSA (Digital Signature Algorithm), o ElGamal e o ECC (Elliptic Curve Cryptography).

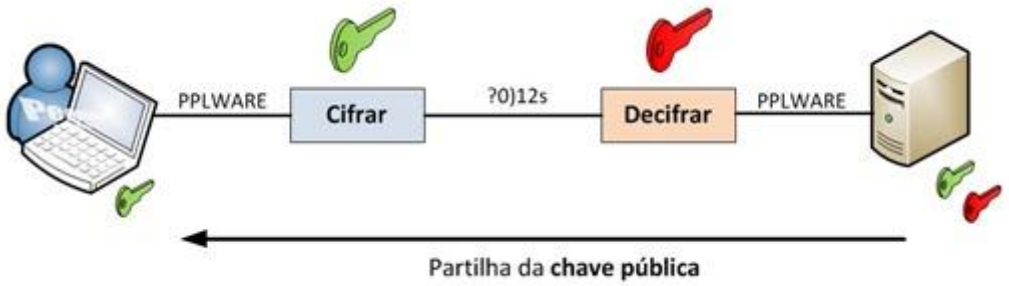


Figura 7 - Algoritmo assimétrico [20]

2.1.3 Função hash

Função hash, também conhecida como função resumo, é uma função unilateral que manipula uma dada entrada de dados, de comprimento variável, e retorna uma sequência de bits de comprimento fixo. Este resultado é também conhecido por valor hash da entrada de dados e representa de forma concisa a mensagem original. [21]

Sendo uma função unilateral, significa que após o cálculo do valor hash, não é possível calcular a informação de entrada. Esta deve também garantir que diferentes entradas de dados não geram valores hash iguais. Atualmente, existem duas funções hash amplamente utilizadas: MD5 (Message Digest) e SHA-1 (Secure Hash Algorithm). [22]

Figura 8 - Função hash [23]

2.1.4 Conclusão

Neste tópico apresentamos os algoritmos de encriptação e que tipos de algoritmos de encriptação existem. Na GTS é utilizado o algoritmo assimétricos RSA para efetuar a encriptação dos dados.

Falamos também da função hash, que é utilizada para assinar o documento (este processo será explicado mais à frente no ponto 2.2.6.1) e garantir a integridade dos dados. Esta é implementada na plataforma com recurso ao algoritmo SHA-256, que é uma variante do SHA-1. Outra das utilizações da função hash na GTS, é para "esconder" as passwords definidas pelos utilizadores como meio de autenticação, uma vez que assim não é possível saber a que valor se refere. Quando é feita a autenticação, apenas é feito um *matching* do valor inserido com a hash, a ver se ambas correspondem.

2.2 Infraestrutura de chaves pública

Uma Infraestrutura de chaves pública (Public Key Infrastructure – PKI), é uma estrutura pública ou privada que possui um vasto leque de ferramentas necessárias para a criação, gestão, distribuição, uso, armazenamento e revogação de certificados digitais, assim como pela gestão das chaves públicas. [24]

Um PKI deve garantir os princípios básicos da segurança da informação, que segundo os padrões internacionais presentes na ISO/IEC 17799:2005, sendo estes:

- Autenticação – garantir a autenticidade dos dados e de quem é proveniente a informação. Garante também o não repúdio, uma vez que o emissor é o único a possuir a chave privada;
- Confidencialidade – garantir que a informação é apenas acedida por quem tenha autorização;
- Integridade – garantir que a informação não foi alterada, salvaguardo a mesma durante o processamento dos dados e protegendo de acessos indevidos ou alterações acidentais;
- Disponibilidade – garantir que a informação está sempre que necessário disponível, sendo necessário backup e redundância.

2.2.1 Autoridade Certificadora

Numa típica infraestrutura de chave pública (PKI), o emissor dos certificados é a Autoridade Certificadora (CA), sendo esta geralmente uma empresa que emite certificados digitais e que garante a autenticidade e validade dos dados presentes no mesmo.

Uma CA é constituída pelo conjunto do hardware, software e pelas pessoas que trabalham na emissão e revogação dos certificados digitais, assim como na emissão e publicação de CRL (Certificate Revocation List ou Lista de Certificados Revogados). [25] Apenas as autoridades certificadoras podem gerar certificados digitais válidos, e são estas as únicas a terem acesso às chaves privadas dos certificados. Se ocorrer algum problema com a CA, todos os certificados por esta emitidos podem ficar comprometidos e invalidados.

No caso deste projeto, a autoridade certificadora é a entidade patronal ACIN. [26] É possível verificar que o website anterior faz menção à eIDAS. Segundo a Adobe [27], "A Regulamentação de Identificação Eletrônica e Serviços Confiáveis (eIDAS 910/2014/EC) é um

regulamento único e padronizado aplicável em todos os países-membros da UE, que fornece uma estrutura jurídica consistente para aceitar identidades e assinaturas eletrônicas". É possível saber mais sobre a eIDAS no website do Gabinete Nacional de Segurança [28].

Esta CA está estruturada de uma forma hierárquica como ilustrado na Figura 9.

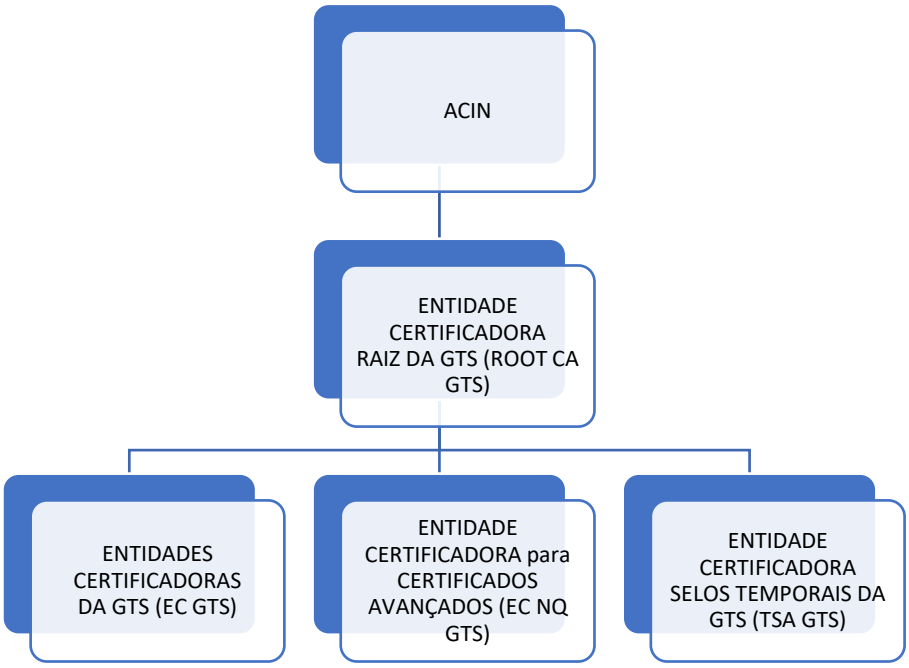


Figura 9 - Arquitetura da Infraestrutura de Chaves Públicas da ACIN

2.2.2 Certificados Digitais

O conceito de certificado digital foi introduzido por Loren Kohnfelder na sua tese sobre como facilitar a gestão de chaves públicas. [29] Em criptografia, um certificado digital, é de forma simplificada, um documento eletrónico que comprova que uma determinada chave pública pertence a uma determina pessoa presente nesse certificado, bem como a realização das transações eletrónicas de forma segura.

Além da chave pública, o certificado também incluí como já mencionado em cima, as informações sobre a identidade do proprietário e a assinatura digital da entidade que emitiu o certificado. O certificado é assinado pela chave privada da Autoridade Certificadora (CA – Certificate Authority), de modo a garantir a validade da informação do certificado digital.

2.2.2.1 X.509

O formato mais comum para a criação da estrutura de dados dos certificados digitais é determinado pelo X.509, sendo este é definido pelo RFC 5280. [30] Um certificado com a estrutura padrão do X.509 é constituído pelos seguintes campos:

- Versão: versão do certificado X.509, atualmente versão 3;
- Número de série: todos os certificados possuem um número de série que não é necessariamente único do ponto de vista de todos os certificados existentes, mas é único nos certificados gerados por uma CA, uma vez que é o indicador utilizado para a revogação do mesmo;
- Tipo de algoritmo: identificador do algoritmo utilizado pela CA para assinar o certificado, juntamente com o tipo de função hash utilizada no certificado;
- Nome do titular: entidade para o qual o certificado foi emitido;
- Nome do emissor: CA que emitiu/assinou o certificado;
- Período de validade: validade do certificado
- Informações da chave pública da entidade:
- Assinatura da CA: a garantia da CA sobre a veracidade das informações contidas no certificado;
- "Centro de Revogação": pode ser o local de consulta da CRL – Certificate Revocation List – Lista de Certificados Revogados, ou o local de consulta do OCSP – Online Certificate Status Protocol – Protocolo de Status de Certificado Online);
- Identificador da chave do titular: extensão do X.509 que possui um identificador numérico para a chave pública contida neste certificado;
- Identificador da chave do emissor: a mesma ideia mencionada anteriormente, só que com relação à chave pública da AC que emitiu o certificado;
- Atributos ou extensões: informações extras, como títulos do titular, ordem profissional a que pertence, etc.

Através da utilização da estrutura anterior, é possível identificar o titular do certificado digital, assim como as informações mais relevantes, sem haver a necessidade de consultar a infraestrutura de chave pública.

Na imagem seguinte, é possível verificar um exemplo de um certificado digital e os elementos que o compõem.

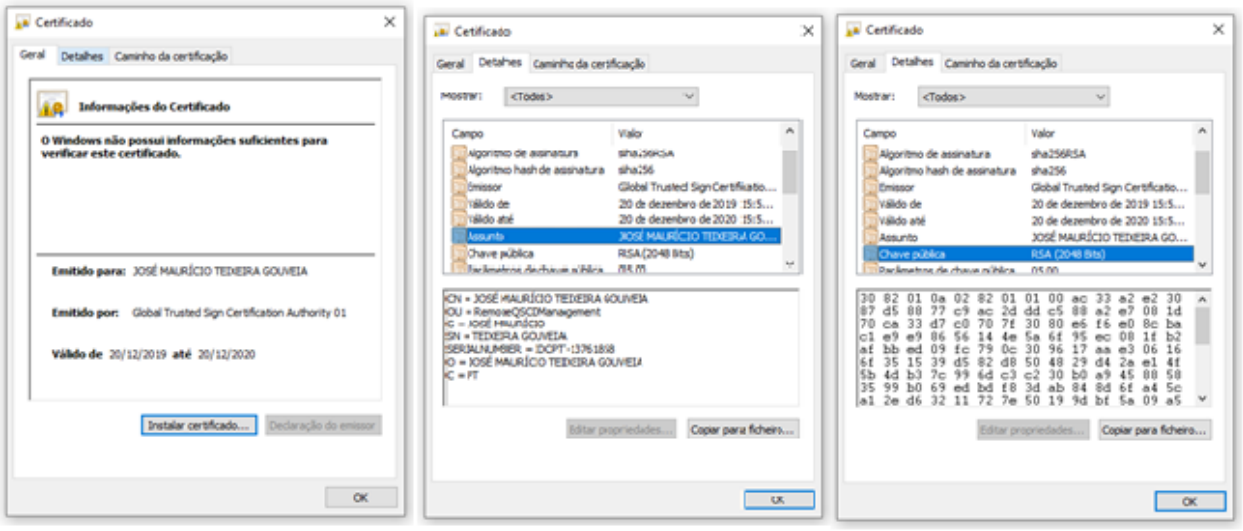


Figura 10 – Exemplo de um certificado

2.2.3 Autoridades de Registo

Quando é efetuada a compra de um certificado digital, existe a necessidade de autenticar o titular que pretende adquirir o certificado. Esta função é então da responsabilidade das Autoridades de Registo, ou como designado na ACIN, os Administradores de Registo.

Tal como as Autoridades Certificadoras, também as Autoridades de Registo são constituídas pelo conjunto do hardware, software e pelas pessoas que trabalham na realização destas funções. [25]

No caso dos Certificados Digitais emitidos pela empresa ACIN, a validade dos titulares podes ser comprovada através de três formas. A primeira é o preenchimento e assinatura do formulário de adesão e envio da documentação original para a morada da empresa. A segunda alternativa é através de uma videoconferência com o titular para comprovar a sua entidade. Por fim e a opção mais recente, é a validação do titular através de uma integração com aplicação do governo português, Autenticação.gov. [31] Nesta última opção, o utilizar tem duas alternativas: pode efetuar a validação com um leitor de cartão de cidadão, ou através da chave móvel digital. Assim, teremos mais garantias que a informação submetida pelo utilizador corresponde à informação real transmitida pelo cartão de cidadão.

Contudo, em qualquer uma das três opções disponíveis de validação, os administradores de registo efetuam sempre uma validação final da informação submetida com a informação presente no cartão de cidadão, de modo a garantir que o certificado digital que irá ser gerado contém informações fidedignas.

Após a definição do papel das autoridades de registo, é possível então definir o fluxo do processo de aquisição de um certificado digital, tal como exemplificado na imagem seguinte.

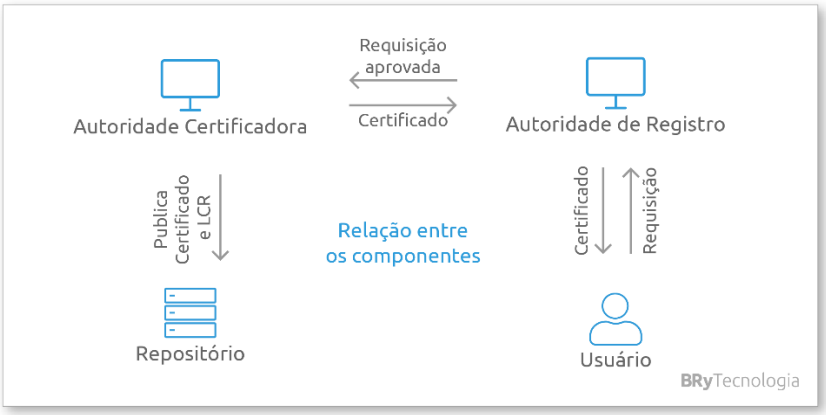


Figura 11 - Fluxo do processo de aquisição de um certificado digital

O processo é então o seguinte: um utilizador pretende efetuar a aquisição de um certificado digital, a autoridade de registo efetua a validação dos dados do requerente, após os dados serem validados a autoridade certificadora efetua a emissão do certificado e o mesmo é guardado no repositório da autoridade certificadora.

2.2.4 Ciclo de vida de um Certificado Digital

Um certificado digital é assente nos estados seguinte: emitido, revogado e expirado. Aquando da criação do certificado digital pela CA, este passa a ter o estado emitido. Quando ocorre algum problema com o certificado e o mesmo tem de ser "destruído", este passa a ter o estado revogado. Como já mencionado anteriormente, os certificados digitais têm um período de validade, quando esta chega ao fim, o certificado passa a ter o estado de expirado. É da responsabilidade das CA efetuar a gestão dos estados dos certificado.

Figura 12 - Ciclo de vida de um certificado digital

2.2.5 Revogação de um Certificado Digital

Após a emissão de um certificado digital, é expectável que o mesmo seja utilizado durante o período completo da sua validade. Contudo, pelas mais diversas razões, como o comprometimento da chave privada ou por mera decisão do seu proprietário, este pode ser invalidado em um período anterior à sua data de validade. Se tal situação ocorrer, a CA efetua a revogação do certificado e comunica esse procedimento.

Para a comunicação deste procedimento, existem duas alternativas normalmente utilizadas e já mencionadas anteriormente no "centro de revogação". O forma mais simples é através da CRL (Certificate Revocation List ou Lista de Certificados Revogados), publicando periodicamente em um repositório a lista dos certificados revogadas. Nesta lista deverá constar a data que a mesma foi tornada pública, o número de série de todos os certificados revogados pela CA, assim como o motivo pelo qual o certificado foi revogado. Seguidamente a lista é assinada pela CA garantindo a autenticidade e integridade da mesma.

Outra solução seria a implementação do OCSP (Protocolo de Status de Certificado Online). Este faria os pedidos para a VA (Validation Authority), que recebe o número de série do certificado e vai à base de dados verificar se o mesmo está revogado. A resposta emitida é assinada por um certificado de modo a garantir a sua veracidade.

A desvantagem da CRL é que esta é atualizada uma vez por dia, sendo que um certificado revogado hoje, só seria atualizado amanhã. Com o OCSP isso não acontece, como o pedido é efetuado diretamente à base de dados, a resposta está atualizada.

Quando é finalizado o processo de revogação do certificado ou quando um certificado está expirado, qualquer assinatura que seja efetuado com este certificado não é válida, contudo as assinaturas que foram efetuadas anteriormente permanecem válidas.

2.2.6 Assinatura Digital

Segundo o Decreto-Lei n.º 290-D/99, artigo 2 alínea c) presente no Diário da República, a assinatura digital é um "processo de assinatura eletrónica baseado em sistema criptográfico assimétrico composto de um algoritmo ou série de algoritmos, mediante o qual é gerado um par de chaves assimétricas exclusivas e interdependentes, uma das quais privada e outra pública, e que permite ao titular usar a chave privada para declarar a autoria do documento eletrónico ao qual a assinatura é aposta e concordância com o seu conteúdo, e ao declaratário usar a chave pública para verificar se a assinatura foi criada mediante o uso da correspondente chave privada e se o documento eletrónico foi alterado depois de aposta a assinatura".

Na assinatura tradicional, é exigido o papel e a caneta para impedir que sejam utilizadas falsificações ou roubo da entidade com recurso a fotocópias da assinatura original. As rasuras ou emendas são facilmente detetáveis, uma vez que a confirmação da assinatura é efetuada meramente por inspeção visual.

No caso da assinatura digital esta inspeção visual não é possível. Para confirmar a assinatura é necessário inverter o cálculo da mesma e possuir o par de chaves: chave pública e chave privada.

A assinatura digital assim como a assinatura convencional, pretende garantir inequivocamente a identidade de quem assina um documento digital, possuindo certificados digitais. Segundo o Autenticação.gov, que é responsável pela assinatura com o cartão de cidadão, "A assinatura digital tem a mesma validade legal que uma assinatura à mão. Uma assinatura digital possui certificados digitais associados que asseguram inequivocamente a identidade de quem assina um documento digital".

Mas afinal o que distingue uma assinatura digital de um certificado digital? Pode existir alguma confusão entre os dois termos, contudo são bem distintos apesar de um complementar o outro. Um certificado digital pode ser comparado a um documento de identificação de uma pessoa, como o cartão de cidadão, ou a identificação de uma empresa. Com esse documento de identificação é possível assinar digitalmente documentos.

Por sua vez, a assinatura digital é um método criptográfico que garante a autenticidade dos ficheiros assinados e adiciona valor jurídico aos mesmos. Apenas os documentos assinados por certificados digitais emitidos por entidades certificadores reconhecidas, possuem valor jurídico.

Com um certificado digital é então possível assinar digitalmente documentos e enviá-los para qualquer pessoa em qualquer parte do mundo. É também possível aceder a ambientes digitais de forma segura.

Uma das últimas adições na empresa ACIN, foi a assinatura de emails. Ou seja, após a redação de um email, é possível assinar digitalmente o mesmo, de modo que o destinatário tenha a garantia que não ocorreram alterações naquele email, e que garante também a entidade do remetente. Outra adição foi a faturação eletrónica, onde é possível por exemplo, assinar digitalmente um lote de faturas.

Podemos então concluir que as assinaturas digitais têm imensas vantagens, desde a validade jurídica dos documentos assinados, à simplificação do processo de validação de assinaturas pessoais ou empresarias nos cartórios, a poupança de tempos nestes processos, assinar digitalmente contratos, efetuar este processo em qualquer sítio com acesso à internet (como por exemplo o smartphone), etc.

2.2.6.1 Utilização da função hash nas assinaturas digitais

A assinatura de um documento pode ser efetuada essencialmente de duas formas: *document signing* ou *hash signing*. Enquanto no *document signing*, o documento é enviado na totalidade para ser assinado, no caso da *hash signing*, é feita uma preparação do documento que resulta no *digest* do mesmo, ou seja, uma representação do documento e efetuado o *hashing* deste. O resultado é enviado para ser então assinado, sendo a *hash* assinada depois incluída no documento junto com a componente pública do certificado, de modo a validar a assinatura do mesmo.

A GTS utiliza o método de *hash signing* devido a este ser um método mais rápido, uma vez que no caso de um documento possuir 10MB por exemplo, em vez de ser necessário enviar todo o documento para ser assinado, a *hash* enviada para assinar que representa o mesmo, terá apenas alguns KB. Também desta forma não é necessário partilhar a informação do documento para esta ser assinada. O processo de *hash signing* pode ser visualizado na Figura 13 - Processo de hash signing baixo.

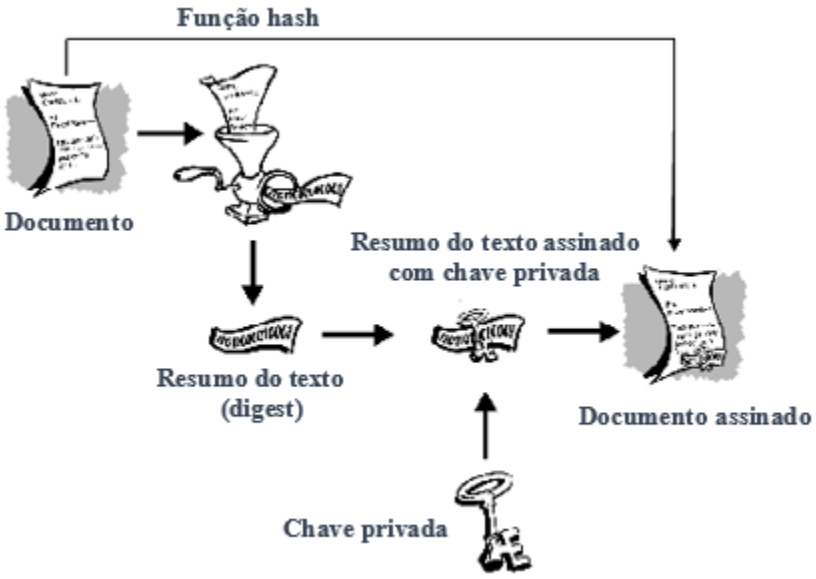


Figura 13 - Processo de hash signing [23]

No início deste capítulo foram referidos os princípios básicos da segurança. Segundo Guimarães [32], é então possível efetuar a correspondência do processo anterior com os princípios básicos da segurança da seguinte forma:

- Autenticação – pelo processo mencionado anteriormente é possível confirmar se o documento original está intacto ou se sofreu alterações, devido à comparação dos valores hash que apenas é gerado com recurso à chave privada, conseguindo garantir a autenticidade do documento;
- Integridade – confirmação que o documento enviado pelo emissor é o mesmo que chega ao recetor, uma vez que se em algum ponto intermédio do processo o mesmo for modificado, os valores hash não vão coincidir;

- Não repúdio – é possível garantir o emissor do documento, pelo facto de apenas este último ser o único a possuir a chave privada.

2.2.7 Conclusão

Neste projeto a infraestrutura de chaves pública (PKI), será denominada daqui em diante por TX. Esta é então responsável por emitir e gerir os certificados utilizados nas assinaturas digitais. Neste tópico foi possível observar como é o processo de aquisição de um certificado digital, desde a validação dos dados até à sua emissão, e também a importância das funções hash para validarem que o documento não foi modificado após a sua assinatura.

2.3 Análise da concorrência

Com recurso ao website *wappalyzer* foi possível obter algumas das tecnologias utilizadas pelos dois principais concorrentes da GTS, assim como a da própria GTS e efetuar uma comparação entre estas que pode ser analisada no exemplo da tabela seguinte. As restantes tabelas podem ser consultadas no anexo A, ponto 9.1.1.

Global Trusted Sign – GTS	
Servidor Web	Apache
Linguagem de programação	PHP
Web Framework	CodeIgniter
Bibliotecas de Javascript	jQuery jQuery UI core-js OWL Carousel PDF.js
Frameworks UI	Bootstrap animate.css
Segurança	reCAPTCHA
Biblioteca de Fontes	Font Awesome Google Font API
Tag Manager	Google Tag Manager

Dados analíticos	Google Analytics Facebook Pixel Google Ads Conversion Tracking Linkedin Insight Tag
Markting	Google Ads Google Remarketing Tag

Tabela 3 - Tecnologias utilizadas pela GTS

Após uma breve visita aos sites dos principais concorrentes, foram analisados os produtos comercializados e efetuada uma comparação de acordo com o mesmo tipo de produto. A mesma pode ser encontrada também no anexo A, ponto 9.1.2.

2.4 Engenharia de Software

Segundo Marco Tulio Valente [33], a "Engenharia de Software trata da aplicação de abordagens sistemáticas, disciplinadas e quantificáveis para desenvolver, operar, manter e evoluir software." Já o *Guide to the Software Engineering Body of Knowledge* (SWEBOK) [34], que é um documento elaborado pela *IEEE Computer Society* (uma sociedade científica internacional) e que tem como objetivo documentar as áreas de conhecimento da Engenharia de Software, indica que estas estão divididas nas seguintes categorias:

- Requisitos de software
- Design de software
- Construção de software
- Teste de software
- Manutenção de software
- Gestão da configuração de software
- Gestão de Engenharia de Software
- Processo de Engenharia de Software
- Modelos e métodos de engenharia de software
- Qualidade de Software
- Prática Profissional de Engenharia de Software
- Economia da Engenharia de Software
- Fundações de computação
- Fundamentos Matemáticos
- Fundações de Engenharia

Neste capítulo iremos então especificar algumas das áreas de conhecimento da Engenharia de Software utilizadas, nomeadamente o processo de desenvolvimento de software e as metodologias existentes.

2.4.1 Processo de Engenharia de Software

A engenharia de software pode ser organizada em camadas e tem como principal foco a qualidade do produto final. As restantes camadas são os processos, os métodos e as ferramentas.

Figura 14 - Camadas da Engenharia de Software [35]

De modo a dar ênfase à qualidade, o software deve possuir características tais como: funcionalidade solicitadas, produto confiável, boa usabilidade, fácil manutenção, eficiente, fácil integração, etc. Na figura seguinte é possível visualizar as características da qualidade segundo a norma ISO 9126. [36]

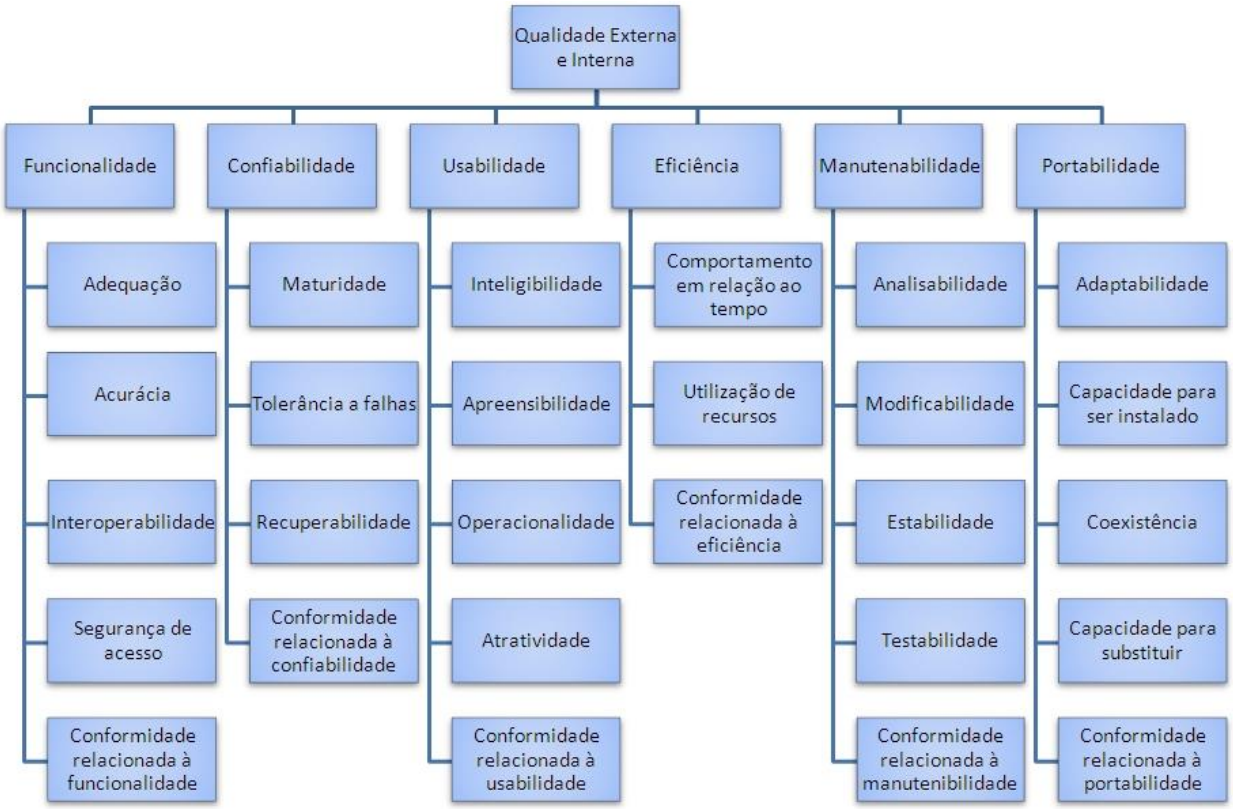


Figura 15 - Características de qualidade segundo a norma ISO 9126 [36]

Segundo Boehm, Brown e Lipow [37], os atributos de qualidade podem ser bem definidos e distintos segundo um esquema em àrvore, na figura seguinte, onde as setas assinalam as pressuposições cada atributo. Ou seja, um software que detenha uma boa usabilidade, irá ser eficiente e de fácil acesso ou confiável e preciso. Outro exemplo, é se for de fácil manutenção, também será fácil efetuar testes e fácil de ser entendido.

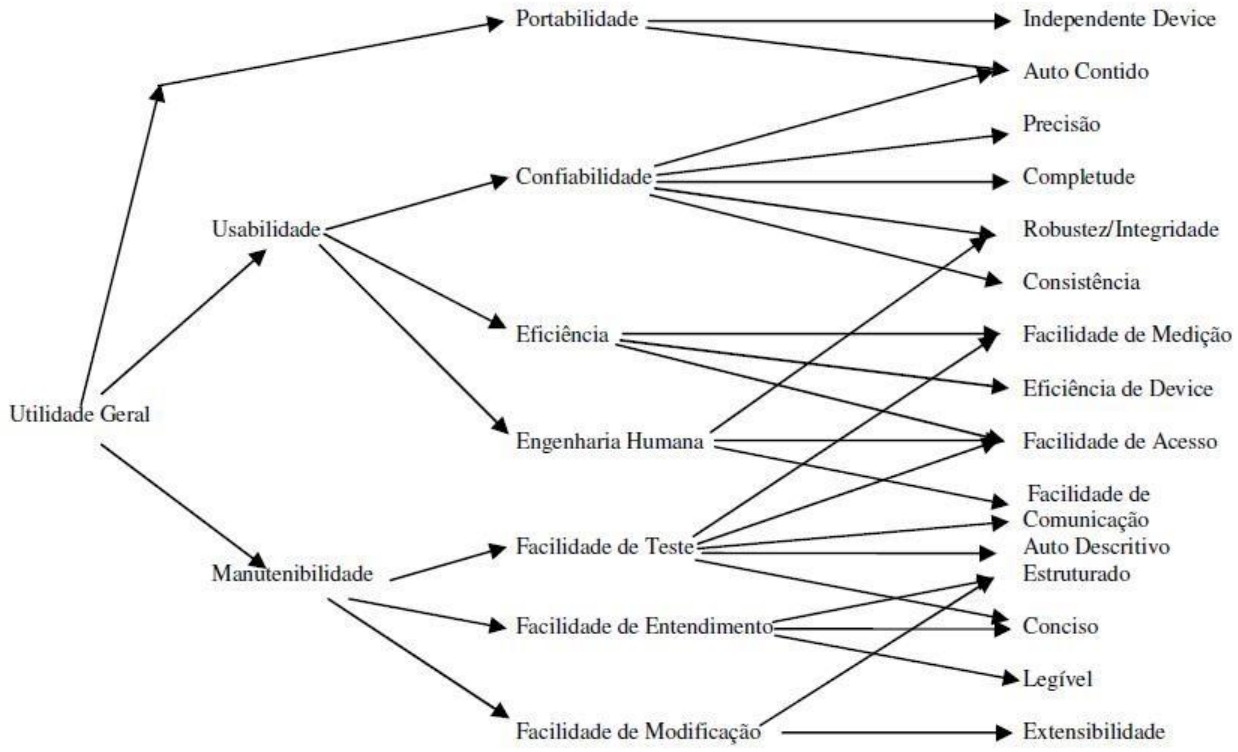


Figura 16 - Árvore de Características de Qualidade de Software (Boehm,Brown e Lipow, 1976)

Foram criadas várias normas que pretentem ajudar no processo de desenvolvimento, contudo a intenção não é fornecer uma fórmula a ser seguida à risca, mas identificar os pontos principais que um bom processo deve ter. Um exemplo é a ISO 9001 que tem como princípios melhorar o desempenho e a capacidade de fornecer produtos e serviços que satisfaçam os requisitos e exigências. A GTS cumpre toda a regulamentação necessária para ser portadora desta norma.

Na organização em camadas, a parte principal é a camada dos processos, é esta que estabelece a metadologia de desenvolvimento, efetua a gestão do projeto e controla os prazos de entrega. Este processo deve ter em conta pormenores como o tipo aplicação a ser implementada, o tamanho que esta vai possuir, a complexidade existente, as características da equipa de desenvolvimento, etc.

Um processo de desenvolvimento de software é então segundo Macoratti, “Um processo de software pode ser entendido como um conjunto estruturado de atividades exigidas para desenvolver um sistema de software”. [38] A implementação deve seguir um conjunto de procedimentos que determinam o seu ciclo de vida.

2.4.2 Ciclo de vida

Segunda a norma ISO 12207:1998 o ciclo de vida do desenvolvimento de software é uma "estrutura contendo processos, atividades e tarefas envolvidas no desenvolvimento, operação e manutenção de um produto de software, abrangendo a vida do sistema, desde a definição de seus requisitos até o término de seu uso."

O processo de desenvolvimento de software está divido em 5 fases [38]:

1) **Fase de requisitos** – onde são levantados os requisitos necessários para o projeto, efetuado um estudo da exequibilidade do mesmo;
2) **Fase do projeto** – é efetuado um planeamento de todas as tarefas necessárias para a conclusão do projeto, assim como a escolha da metadologia a utilizar e os diagramas e cronogramas necessários, a prototipagem e o design;
3) **Fase de implementação** – é implementado todas as funcionalidades existentes nos requisitos na linguagem de programação escolhida;
4) **Fase de testes** – são construidos testes automatizados, assim como são executados testes manuais por beta testers de modo a garantir que tudo funciona como esperado na integração dos diversos módulos;
5) **Fase de produção e manutenção** – é avaliado o produto final e implantado em produção, onde posteriormente será efetuada a manutenção e evolução, entrando num novo loop das fases anterior.

2.4.3 Metodologias

Para desenvolver o processo anteriomente descrito, existem vários modelos já desenvolvidos que podem ser adotados, contudo os mais utilizados são: o modelo cascata, o modelo espiral, o modelo incremental. Posteriormente a estes modelos foi adicionado as metadologias ágeis. [35]

O modelo cascata, também denominado por ciclo de vida clássico, define uma abordagem sequencial do processo, ou seja, o desenvolvimento é efetuado apenas em um sentido e as tarefas não podem ser repetidas. Uma das vantagens deste modelo é que o desenvolvimento apenas transita para a tarefa seguinte depois do cliente validar que todos os requisitos estão cumpridos. Contudo, também é facil verificar que a desvantagem na sua utilização é quando os requisitos não estão completos, são vagos ou prováveis de sofrerem alterações. [35]

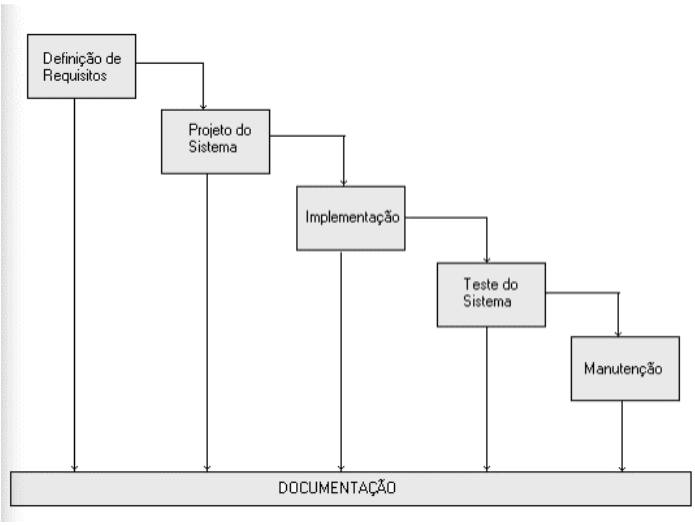


Figura 17 - Modelo cascata [38]

No modelo em espiral temos um processo de desenvolvimento em ciclos, sendo que cada volta é uma das fases do processo. Cada ciclo é divido em quatro etapas: análise dos objetivos e restrições, análise de risco, desenvolvimento e por fim avaliação das etapas anteriores e planeamento da seguinte. Como desvantagem, este é um desenvolvimento complexo que consome mais tempo e recursos. Contudo, temos um planeamento sistemático que permite lidar melhor com alterações no desenvolvimento e onde os problemas mais importantes são encontrados atempadamente. [38]

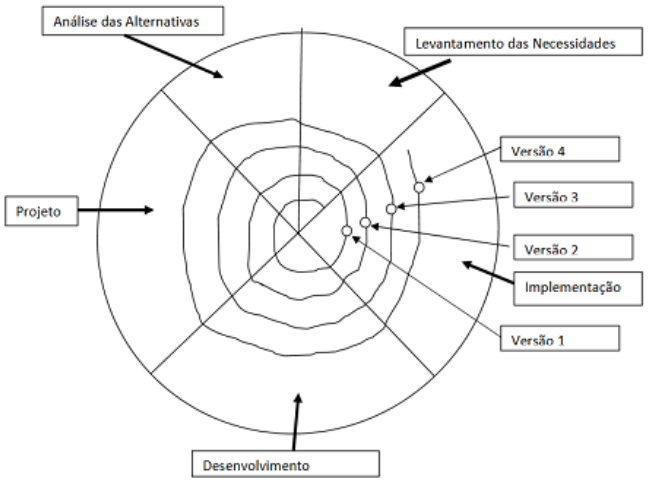


Figura 18 - Modelo espiral [39]

Já o modelo incremental pode ser visto como uma extensão do modelo espiral, contudo utiliza um processo mais exigente. Neste modelo é possível desenvolver diversos requisitos de forma paralela, e efetuar a integração quando estes estiverem finalizados. Em cada uma das integrações é efetuado todo o ciclo de desenvolvimento de software, de modo a obter um resultado funcional. Estas integrações devem ser de dimensões reduzidas e genéricas de modo a possibilitar a sua

expansão e de maneira que seja possível demonstrar a nova funcionalidade introduzida. Este modelo também possui uma grande gestão de custos devido ao elevado número de iterações. Por outro lado, permite que vários membros da equipa possam trabalhar em tarefas distintas e que se uma integração correr mal, apenas seja necessário corrigir a mesma. [38]

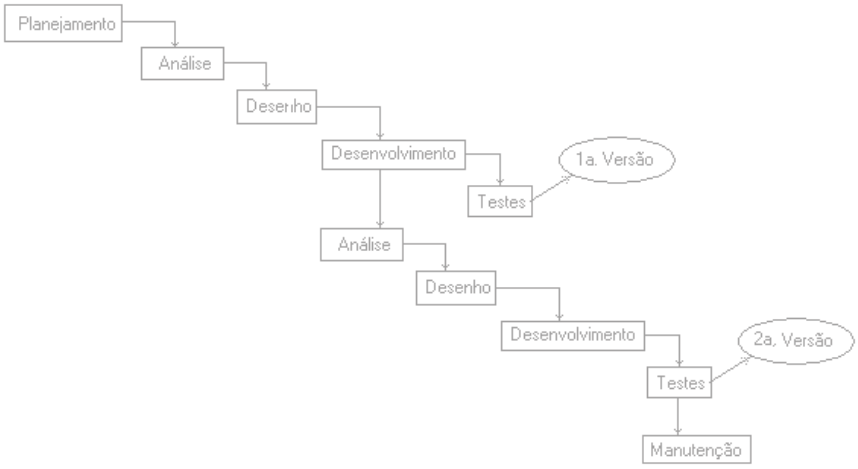


Figura 19 - Modelo incremental [38]

Por último temos as metodologias ágeis, que nasceram da experiência de diversos profissionais na área da engenharia de software. Segundo esta experiência, a abordagem utilizada deveria ser mais focada nas pessoas e nos clientes, sendo adaptáveis a recorrentes alterações e que tivessem prazos de entrega mais curtos. Estas metodologias tornaram-se mais conhecidas em 2001 quando um grupo de dezassete programadores se reuniram em um workshop na cidade de Snowbird no Utah. Assim nasceu o Manifesto para o Desenvolvimento Ágil de Software. [40] Este manifesto tem definido quatro valores:

- Indivíduos e interações mais que processos e ferramentas
- Software em funcionamento mais que documentação abrangente
- Colaboração com o cliente mais que negociação de contratos
- Responder a mudanças mais que seguir um plano

A metodologia ágil pode ser equiparada ao modelo incremental, contudo os períodos de tempos são medidos em semanas, em vez de meses. Um dos principais exemplos da metodologia ágil é o *Scrum*. No *Scrum* o desenvolvimento é efetuado por *sprints*, que são ciclos de trabalho normalmente com uma duração média de um mês. Todos os membros do desenvolvimento do projeto sabem os requisitos do mesmo, o estado atual do projeto. O *Scrum Master* é o responsável pela equipa de desenvolvimento que reúne com o cliente (*product owner*) para saber os requisitos prioritários (*product backlog*). Em cada *sprint* o *Scrum Master* efetua uma lista das maiores prioridades do *backlog* a serem implementadas e atribui a cada programador.

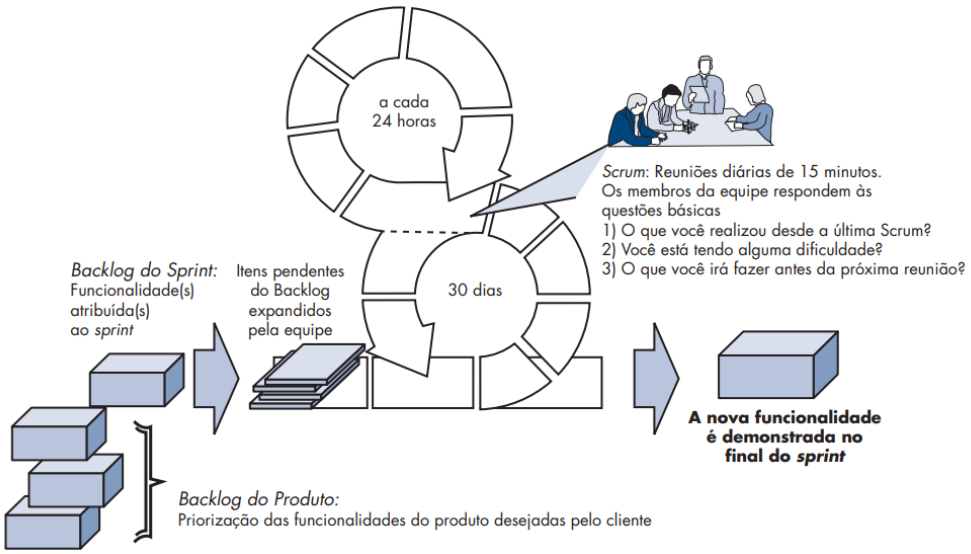


Figura 20 – Metodologia ágil Scrum [35]

Na aplicação em questão, será utilizado o software *JIRA*[1], desenvolvido pela *Atlassian* para ser utilizado em projetos ágeis, e tem como finalidade a gestão e monotorização das tarefas que serão implementadas. Esta ferramenta funciona em sintonia com o *Scrum*, uma vez que tem uma área para serem inseridos os requisitos no backlog.

Antes de um sprint começar, quando o *Scrum Master* se reúne com a equipa, este irá utilizar o *JIRA* durante a reunião para definir as tarefas com a equipa. O *Scrum Master*, tal como já mencionado anteriormente, vai ao *backlog* e verifica os requisitos mais prioritários para serem implementados durante esse *sprint*. Na figura seguinte é possível verificar um exemplo de um *backlog* no *JIRA*, onde as setas à direita simbolizam a prioridade que a tarefa tem, sendo que a cor vermelha indica uma grande prioridade, a cor laranja prioridade média e a cor verde indica baixa prioridade.

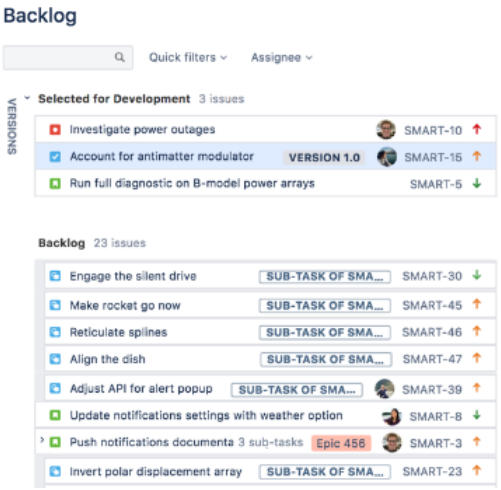


Figura 21 - Exemplo do Backlog do JIRA [41]

[1] Disponível em https://www.atlassian.com/software/jira

Durante a reunião, o *Scrum Master* após verificar as tarefas com indicação vermelha, cria um novo *sprint* no *JIRA*, indicando as datas de início e fim do mesmo, e arrasta do *backlog* para cima, onde está o novo sprint que irá começar, e atribui a um programador, ficando na coluna "TO DO". Quando o programador começar a implementar a tarefa, arrasta a mesma da coluna "TO DO" para a coluna "IN PROGRESS", simbolizando que já está a iniciar a tarefa correspondente. Quando termina a tarefa e esta já se encontra testada, o "código" é verificado pelo *Scrum Master*, de modo a garantir que tudo se encontra organizado e implementado da melhor forma, sendo que a isto chamamos de "Code Review" e a tarefa é colocada nesta coluna no *JIRA*. Por fim após validação esta é movida para a coluna "DONE" e o programador passa à tarefa seguinte.

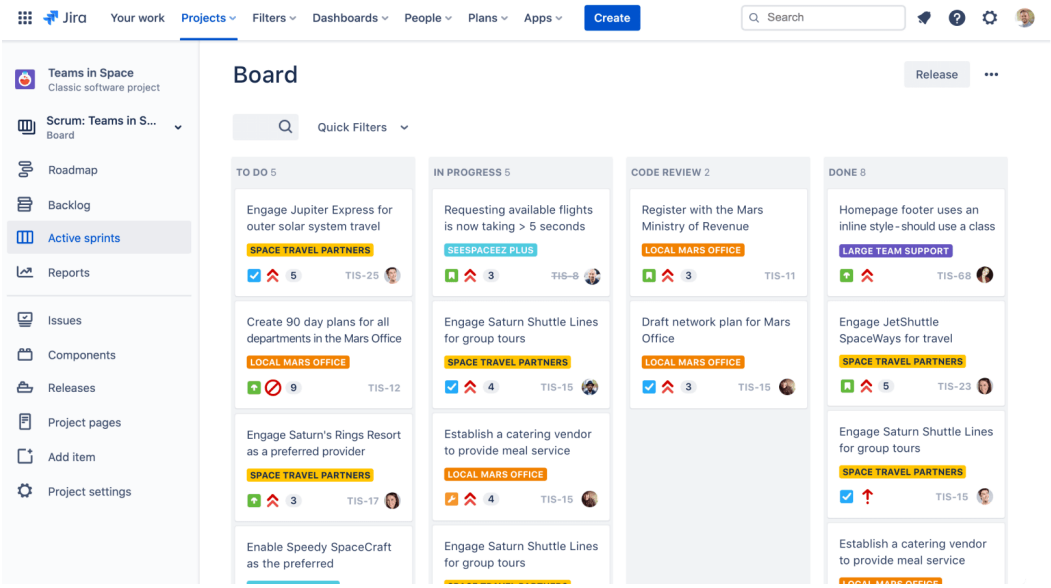
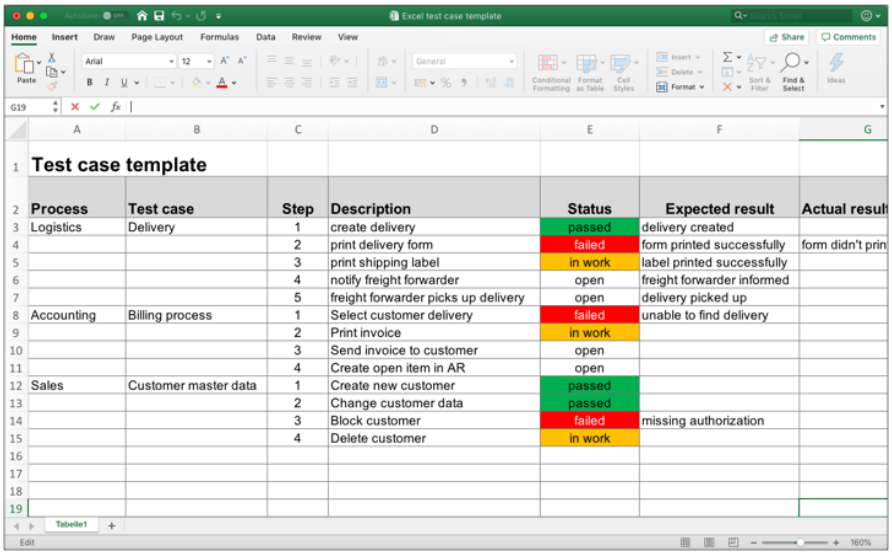


Figura 22 - Exemplo de um sprint no JIRA [41]

Com a utilização desta ferramenta, é fácil o *Scrum Master* e a equipa terem sempre presentes os requisitos que são necessários implementar, estando estes no *backlog*, os programadores sabem sempre que tarefas precisam de realizar e o *Scrum Master* consegue diariamente acompanhar a evolução do desenvolvimento através dos estados em que estão cada tarefa.

Durante o desenvolvimento da tarefa atribuída, serão também implementados testes unitários e de integração, recorrendo à ferramenta Jest que será descrita no ponto 2.6.15. Estes testes automatizados têm como finalidade, garantir que um determinado módulo irá executar uma ação e produzir uma determinada resposta, assim como garantir também que esse módulo se "encaixa" perfeitamente com o restante projeto. Estes testes duram segundos ou menos e garantem que novas implementações não criaram conflitos com o sistema já existente. Os programadores são também responsáveis por elaborarem planos de testes manuais, para antes do sprint terminar, os *beta testers* executarem esses procedimentos e confirmarem se encontraram algum "*bug*" que precise de correção, ou se tudo decorreu como previsto.



Test case template

Process	Test case	Step	Description	Status	Expected result	Actual result
Logistics	Delivery	1	create delivery	passed	delivery created	
		2	print delivery form	failed	form printed successfully	form didn't prin
		3	print shipping label	in work	label printed successfully	
		4	notify freight forwarder	open	freight forwarder informed	
		5	freight forwarder picks up delivery	open	delivery picked up	
Accounting	Billing process	1	Select customer delivery	failed	unable to find delivery	
		2	Print invoice	in work		
		3	Send invoice to customer	open		
		4	Create open item in AR	open		
Sales	Customer master data	1	Create new customer	passed		
		2	Change customer data	passed		
		3	Block customer	failed	missing authorization	
		4	Delete customer	in work		

Figura 23 - Exemplo de um plano de testes utilizado na ACIN

Por fim, quando termina o prazo do *sprint*, a equipa reúne-se com *Scrum Master* e o cliente para fazer a análise de risco. Esta consiste em um levantamento de todos os problemas que podem acontecer quando os requisitos implementados forem entregues ao cliente. Cada possibilidade que possa ser encontrada, tem de ter atribuído uma probabilidade de acontecer e o impacto que pode causar no produto final.

Se a conjugação da probabilidade com o impacto ocorrer em um risco de nível grave, quer dizer que algo não foi bem planeado/implementado e que precisa de ser rapidamente corrigido. Os níveis da matriz são então os seguintes: no nível 1 temos uma atuação não prioritária, no nível 2 temos uma intervenção a médio prazo, no nível 3 temos uma intervenção a curto prazo, no nível 4 temos uma atuação urgente e por último no nível 5 temos uma atuação muito urgente que requer medidas imediatas.

Além desta avaliação, também é necessário atribuir um responsável ao risco encontrado, que caso esta possibilidade suceda, será esta a pessoa quem deve resolver o problema, logo é também necessário averiguar antecipadamente a solução para o problema que pode acontecer.

		PROBABILIDADE			
		Improvavel	Raro	Ocasional	Frequente
GRAVIDADE	Sem Incapacidade	1	2	2	3
	incapacidade temporária absoluta (=< 30 dias)	2	2	3	4
	incapacidade temporária absoluta (> 30 dias)	2	3	4	5
	Incapacidade permanente ou morte	3	4	5	5

Figura 24 - Exemplo de uma matriz de análise de risco utilizada na ACIN

2.4.4 Conclusão

Para o projeto em questão a metodologia seguida será a metodologia ágil, nomeadamente o *Scrum*, sendo que o mesmo é também utilizado nos restantes projetos da empresa ACIN.

Na realização das análises de risco, a ocorrência de riscos de nível 4 ou superior implica que não seja possível implementar as funcionalidades seguintes, sem que os mesmos sejam resolvidos.

2.5 Escolha das tecnologias a utilizar

Tal como mencionado anteriormente, este trabalho irá ser desenvolvido dentro da empresa ACIN, que possui algumas restrições nas ferramentas a utilizar, de modo a obter um maior nível de segurança e qualidade para os seus clientes. Apesar de muitas das ferramentas estarem definidas à partida, efetuou-se um levantamento das tecnologias/frameworks/bibliotecas mais utilizadas atualmente de *front-end*, de modo a encontrar a melhor ferramenta para a solução proposta. Chegou-se à conclusão de que a escolha cairia sobre as seguintes: React, Angular e Vue.

Nos gráficos abaixo é possível visualizar o número de downloads das três bibliotecas por dia (gráfico 1), assim como por ano (gráfico 2). É possível então verificar, que das três bibliotecas mencionadas anteriormente, o React é o que teve um maior número de downloads diários e anuais ao longo do tempo, obtendo um crescimento exponencial e levando uma larga vantagem sobre as bibliotecas. Seguidamente temos o Vue e por fim o Angular.

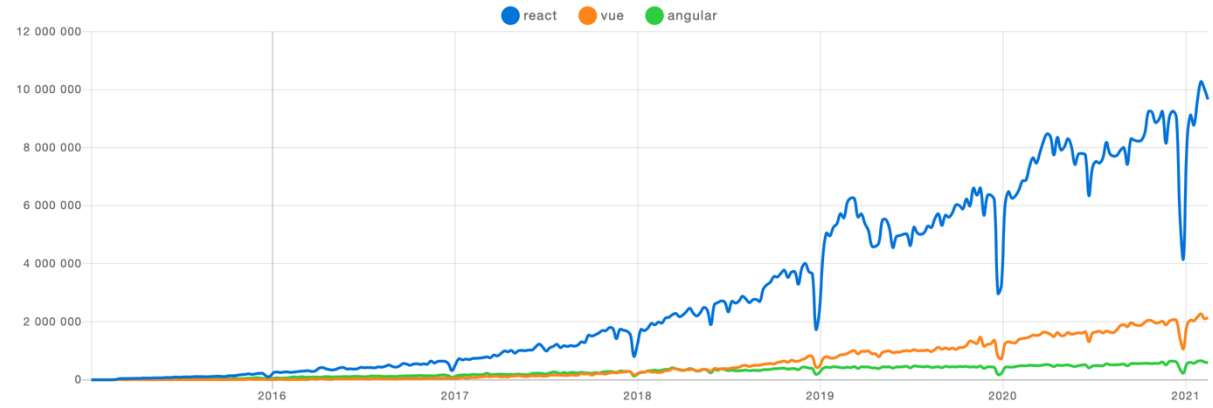


Gráfico 1 - Número de downloads diário [42] [43]

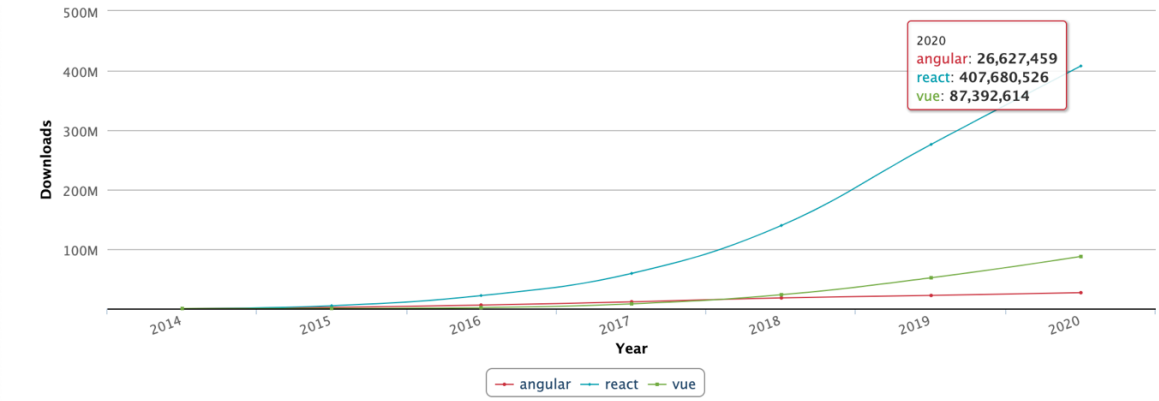


Gráfico 2 - Número de downloads anuais [43, 42]

Na tabela seguinte temos o número total de downloads das três bibliotecas entre os períodos de 12/12/2014 e 31/12/2020, sendo este o mesmo período do gráfico anterior.

Biblioteca	Número de downloads
React	908,916,130
Vue	172,000,056
Angular	85,596,578

Tabela 4 - Número total de downloads das três bibliotecas [44]

Dentro das bibliotecas selecionadas, são apresentados na tabela abaixo as principais estatísticas de cada uma, tendo sido as mesmas consultadas a 23/03/2021 e estando estas presentes no maior repositório de software open-source e de controlo de versões, o Github. É possível verificar desde as estrelas (ranking utilizado na plataforma), o número de problemas encontrados por resolver, a data do último update, a data de criação e o tamanho comprimido de cada uma das bibliotecas.

	Estrelas	Problemas	Data do último update	Data de criação	Tamanho
React	164 400	719	Mar 22, 2021	May 24, 2013	2.8kb
Vue	180 028	564	Mar 1, 2021	Jul 29, 2013	22.9kb
Angular	59 581	467	Mar 17, 2021	Jan 6, 2010	62.3kb

Tabela 5 - Comparação React, Vue e Angular

De realçar que toda a informação apresentada anteriormente provém dos dados estatísticos armazenados no maior gestor de pacotes de JavaScript, o NPM.[2]

2.5.1 React

O React é uma biblioteca de JavaScript, que é composta por componentes que são pequenos módulos da aplicação. Estes componentes têm estados, que vão sendo alterados consoante as ações do utilizador, e podem ser reutilizados por toda a aplicação.

Esta é uma biblioteca de pequenas dimensões, pois é composta essencialmente pelo seu core. Todas as funcionalidades extra, podem ser adicionadas através de bibliotecas externas. Contudo, se não houver cuidado, o React pode ter várias pequenas dependências, ou seja, pequenas bibliotecas incluídas que podem deixar a aplicação final pesada e lenta. Isso acontece, pois, cada uma dessas bibliotecas são implementadas por programadores diferentes, e nem sempre as fazem com o melhor desempenho e da maneira correta.

O React em si é uma ferramenta simples e fácil de se programar, mas à medida que vão sendo adicionadas funcionalidades e bibliotecas, sua complexidade também aumenta.

As funções são declaradas em camelCase e os componentes em PascalCase. No React podemos também integrar o TypeScript.

2.5.1.1 Vantagens

O React é flexível e livre para definir um padrão e também fácil de migrar entre versões. Permite utilizar o javascript, ES6 (javascript moderno com novas funcionalidades), TypeScript, entre outros, utiliza também o Virtual DOM, o que melhora a performance. Utiliza o JSX que é uma das características do React, onde é possível escrever HTML e CSS num ficheiro JavaScript, e além disso possui todos os recursos da linguagem como loops, funções de formatação, validações e muito mais.

Possui a extensão para Chrome e Firefox “React Developer Tool”, que mostra os dados dos estados e os dados da store utilizada, um grande auxiliar como ferramenta de debug e tem a possibilidade de adicionar mais bibliotecas consoante a necessidade. É mantido pelo Facebook, o que dá uma maior segurança ao seu suporte a longo prazo. É popular e tem uma extensa comunidade, que em caso de dúvidas ou problemas, ajuda na sua resolução.

Pode ser totalmente convertido numa aplicação mobile com o React Native e as aplicações React.js são fáceis de serem testadas. É possível usar por exemplo o Jest (ferramenta para testes unitários desenvolvida pelo Facebook) e testar assertions, report de coverage e snapshots.

E a maior vantagem é o facto de ser implementada por componentes o que permite o reaproveitamento do código, facilita a sua manutenção, assim como a evolução e expansão do mesmo.

[2] Disponível em https://www.npmjs.com

2.5.1.2 Desvantagens

O React tem com desvantagem a grande curva de aprendizagem, onde para iniciantes o fluxo entre componentes pode ficar confuso. Se o mesmo não for pensado com antecedência, pode-se não conseguir efetuar a comunicação correta entre componentes.

Se o layout está em constante mudança é necessário criar novos componentes, não sendo possível reutilizar os existentes. O uso de JSX opõe-se ao princípio da separação de linguagens, o que sendo uma vantagem desta biblioteca, pode ser um pesadelo para os novos programadores, ou até mesmo para quem prefere trabalhar com HTML, CSS e JavaScript separado.

Tal como dito inicialmente, o React é uma biblioteca, não uma *framework*. Quer isto dizer que poderá ser necessário utilizar mais bibliotecas ou *frameworks* no projeto. Pelo ponto anterior, é provável haver várias pequenas dependências espalhadas por toda a aplicação, que no *bundle* final, deixa a aplicação mais pesada e lenta. Isso acontece porque cada uma dessas bibliotecas são feitas por programadores diferentes, e nem sempre é feito com o melhor desempenho e da maneira correta.

Este apenas oferece soluções para o *front-end* e a vantagem de ser livre para definir um padrão, pode gerar projetos pouco escaláveis, caso não seja bem estruturado e planificado. E por fim a principal ferramenta de gerenciamento global de estado, o Redux, que é usada na maioria dos projetos, torna a aplicação muito complexa, onde para fazer algo simples é necessário seguir uma série de padrões e normas que se não forem feitas estritamente dentro do padrão podem gerar grandes erros.

2.5.2 Angular

O Angular é uma *framework* de código aberto baseada em TypeScript, que utiliza HTML, CSS e, principalmente, JavaScript. Foi criado pelos programadores da Google e possui uma grande comunidade.

Este é conhecido por utilizar a arquitetura MVC, a forte modularização do código e por utilizar o conceito de *two-way binding*, onde associa um dado na *view* com o respetivo *controller* e vice-versa, possibilitando a alteração simultânea dos elementos ligados.

2.5.2.1 Vantagens

O Angular segue a estrutura MVC, que também pode ser substituído pelo Model-View-ViewModel (MVVM), onde o Controller é trocado pelo ViewModel. Já a preparação do ambiente é feita via CLI (código “pré feito” para o projeto).

Muitos websites utilizam o Angular e este tem uma extensa comunidade. É também mantido pelo Google, o que oferece muita segurança.

2.5.2.2 Desvantagens

O Angular tem também uma grande curva de aprendizagem, não só sua pela sua estrutura, como também pela utilização do TypeScript. Caso a SPA (Single Page Application) cresça imenso,

pode haver problemas de performance. Esta também força a organização, o que pode tirar a liberdade do programador;

2.5.3 Vue.js

O Vue foi criado pelo chinês Evan You, que trabalhou para a Google em diversos projetos utilizando o Angular. Este teve a ideia de criar algo semelhante ao Angular, apenas retirando as funcionalidades que realmente gostava de modo a criar uma *framework* leve. É então uma alternativa às complexidades do Angular.

Segundo a página oficial, o Vue "é um *framework* progressivo para a construção de interfaces de usuário. Ao contrário de outros *frameworks* monolíticos, Vue foi projetado desde sua conceção para ser adotável incrementalmente. A biblioteca principal é focada exclusivamente na camada visual (*view layer*), sendo fácil adotar e integrar com outras bibliotecas ou projetos existentes. Por outro lado, Vue também é perfeitamente capaz de dar poder a sofisticadas Single-Page Applications quando usado em conjunto com ferramentas modernas e bibliotecas de apoio".

2.5.3.1 Vantagens

A sintaxe do Vue é simples de utilizar, comparativamente ao Angular e React e possui também uma extensão *devTool* para *debug* e desenvolvimento. Comparativamente ao Angular, não força uma estrutura pré-definida, dando flexibilidade ao programador e possui documentação extensa e detalhada.

2.5.3.2 Desvantagens

Comparativamente às duas soluções anteriormente apresentadas, o Vue não é mantido de momento por uma "grande" empresa, inferindo menos confiança em relação a futuros *updates*. A comunidade é relativamente pequena, o que diminui o número de recursos disponíveis, comparando com os exemplos anteriores. Alguns dos plug-ins e componentes escritos estão apenas disponíveis em chinês.

2.5.4 Tecnologia Escolhida

Após uma análise das escolhas apresentadas anteriormente, a empresa ACIN escolheu como tecnologia a adotar o React, também conhecido como React.js ou ReactJS. Este não é uma *framework* mas sim uma biblioteca de JavaScript, *open source*, usada para construir *user interfaces* nomeadamente para aplicações de página única. Tem como principal objetivo ser rápido, escalável e simples, podendo ainda ser usada em combinação com outras bibliotecas ou *frameworks* de JavaScript.

O React foi criado em 2011 para o *feed* do Facebook pelo engenheiro de software Jordan Walke, que trabalhava na empresa. Em 2013 o código foi tornado público para programadores e tornou-se muito popular. Esta biblioteca é utilizada nos sites de grandes empresas tais como o

Facebook, Instagram, Netflix, Spotify, Uber, Paypal, Imgur, Feedly, Airbnb, SeatGeek, HelloSign, Walmart, entre outros.

A vantagem de ser utilizada uma biblioteca de *front-end* é a possibilidade de atualizar o estado da interface do utilizador, o reaproveitamento do código, maior velocidade de desenvolvimento e a criação de estruturas mais complexas.

2.5.4.1 Hooks

A funcionalidade dos *hooks* foi lançada na versão 16.8.0 do React, que basicamente oferece formas de trabalhar com estado, o ciclo de vida e outras funcionalidades sem a necessidade de ter uma classe.

A grande vantagem dos *hooks* é a reutilização da lógica dos estados entre componentes, assim como a possibilidade de criar *custom hooks* para facilitar esta partilha. Outra vantagem é evitar a confusão com classes ao ter que utilizar "*this*". Ao utilizar *hooks*, é fornecido aos componentes funcionais um estado interno e acesso ao próprio ciclo de vida através de uma forma mais direta, limpa e sendo necessário muito menos código.

Para utilizar os *hooks* no React existem apenas algumas regras específicas:

- Não devem ser utilizados *hooks* dentro de *loops*, condições ou *callbacks*;
- Os *hooks* devem ser chamados apenas em componentes ou funções de React, não em funções normais de JavaScript e sempre no início, ou seja, no nível mais alto. Isto porque o React identifica os *hooks* de um componente pela ordem em que são chamados, garantindo que são renderizados na ordem correta, permitindo ao React preservar o estado dos *hooks* apesar das múltiplas chamadas do *useState* e *useEffect*;
- As variáveis de estado devem ser sempre declaradas aos pares. ex: const [age, setAge];
- Os *custom hooks* devem ter um nome sempre iniciado por "use" e podem conter outros *hooks*.

2.5.4.2 Context API

No React existem várias abordagens para a partilha de informação entre componentes. A forma mais comum e utilizada são as *props*, que são a maneira de um componente pai transmitir informação para o filho. Contudo, quando queremos transmitir informação para componentes que estão vários nós abaixo, não é viável transmitir essa informação por cada nó até porque pode ser necessário um componente pai necessitar de informação do filho.

Deste modo, é necessário utilizar um gestor de estados. O React oferece 3 alternativas, além das *props* já referenciadas: o Flux, o Redux e o Context API.

Na metodologia Flux, existem várias Stores enquanto o Redux armazena o conteúdo em apenas uma Store. A grande questão destas metodologias, tal como referido nas desvantagens do React, é terem uma implementação mais complexa e que requer diversos procedimentos para guardar algo simples.

Desta forma, o método preferencial e escolhido para este projeto foi o Context API. O Context API sempre existiu, mas era instável e com diversos problemas relacionados à performance. Com

as novas atualizações na versão 16.3 este procedimento melhorou e facilmente é possível partilhar informações entre os vários componentes do React. Para a implementação deste método apenas são necessárias três coisas: React.createContext, Provider e Consumer. Contudo, para facilitar a partilha de métodos, é possível também juntar um *reducer* (semelhante ao utilizado no Redux) para verificar as ações que pretendemos executar.

2.5.4.3 TypeScript

No React para garantir um correto fluxo de dados são utilizadas PropTypes, ou seja, são utilizados tipos para funções e variáveis de modo que os dados que estas recebem, realmente correspondam ao que foi definido e é suposto receberem. A grande desvantagem de apenas utilizar PropTypes, é que estas apenas são verificadas em *runtime*, ou seja, os erros apenas vão acontecer ao executar a aplicação. Como alternativa ou complemento existe o TypeScript.

O TypeScript é um superconjunto de JavaScript desenvolvido pela Microsoft que adiciona tipagem a esta linguagem. A grande vantagem desta ferramenta é a possibilidade de descobrir erros durante o desenvolvimento e incrementar a inteligência (IntelliSense) do ambiente de desenvolvimento utilizado. Deste modo, é possível evitar erros que iriam acontecer apenas em runtime. A magia por detrás do TypeScript é que este é compilado pelo Babel para JavaScript, fazendo a correspondência com a biblioteca de tipos que vem armazenada no TypeScript.

Contudo, existe uma grande desvantagem que poderá ter enorme impacto na produtividade inicial, uma vez que no início da aprendizagem é necessário verificar imensas vezes a documentação para perceber alguns conceitos e verificar os tipos das funções ou métodos definidos.

2.5.4.4 CSS

Para a atribuição de estilos CSS e um ficheiro HTML, normalmente é feita em ficheiros CSS separados ou *inline*, contudo no React temos as seguintes hipóteses: CSS Inline, Styled Components, CSS Módulos e Folhas de estilo.

Existe um grande debate sobre o uso de estilos *inline*, são normalmente utilizados quando são feitas esporádicas alterações ou quando é executada uma alteração dinâmica de uma propriedade CSS via JavaScript.

Um dos problemas ao utilizar CSS puro *inline*, é o facto de não ter acesso a determinados recursos nativos do CSS, como *pseudo-selectors*, *pseudo-elements*, *media queries*, *keyframe animations*, entre outros. A outra questão deve-se ao facto de o navegador levar mais tempo com a renderização do CSS nos scripts, pois precisa mapear todas as regras de estilos do componente, sendo a renderização com regras ‘normais’ do CSS muito mais rápida.

```
const divStyle = {
  margin: '30px 0',
  padding: '15px',
  border: '1px solid #ccc'
};
const pStyle = {
  fontSize: '16px',
  textAlign: 'center'
};

<div style={divStyle}>
  <p style={pStyle}>Inline styles</p>
</div>
```

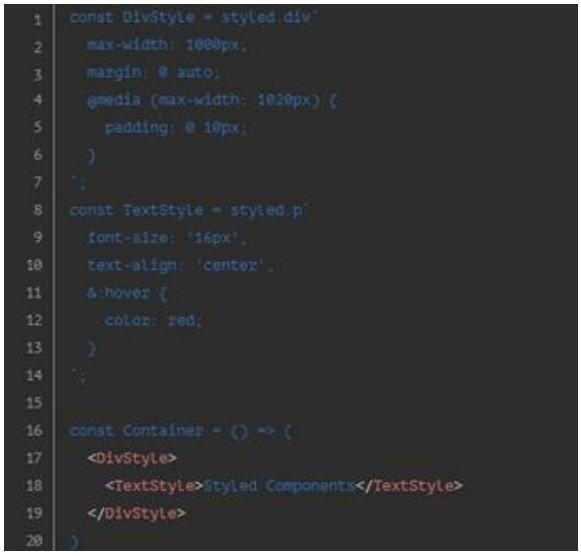
Figura 25 - Exemplo de CSS inline

Styled-components é uma biblioteca para React e React Native que permite a utilização de estilos de CSS como componentes na aplicação. Esses estilos são escritos com uma mistura de JavaScript e CSS.

Ao contrário do CSS Inline, com *styled-components* é possível ultrapassar as restrições do CSS inline, utilizando os recursos nativos do CSS, como *pseudo-selectors*, *pseudo-elements*, *media queries*, *keyframe animations*, entre outros, de forma semelhante à que utilizamos com pré-processadores.

Assim como no CSS Inline, é necessário colocar o código CSS no mesmo ficheiro do componente, tornando-o independente da restante aplicação, o que pode ser uma excelente vantagem.

Contudo, para a utilização de *styled-components*, é necessário realizar sua instalação, seja via npm ou yarn. Uma vez instalados, a importação da biblioteca é realizada no componente onde será utilizada.



```
const DivStyle = styled.div`
  max-width: 1000px;
  margin: 0 auto;
  @media (max-width: 1020px) {
    padding: 0 10px;
  }
`;
const TextStyle = styled.p`
  font-size: '16px',
  text-align: 'center',
  &:hover {
    color: red;
  }
`;

const Container = () => (
  <DivStyle>
    <TextStyle>Styled Components</TextStyle>
  </DivStyle>
)
```

Figura 26 - Exemplo de styled-components

Os CSS Modules são ficheiros CSS onde os nomes das classes são definidos no escopo local, ou seja, apenas estão disponíveis localmente. Estes não são uma biblioteca ou ferramenta externa, mas sim um processo que se utiliza na arquitetura de componentes. Desta forma, os nomes das classes CSS tornam-se semelhantes a variáveis locais em JavaScript, o que facilita a sua utilização.

A chave para este processo, é que cada componente React possui seu próprio ficheiro CSS, que está no escopo do componente. Na compilação (*build*), os nomes das classes locais são mapeados, gerados automaticamente e exportados como um objeto JS para ser utilizado no React. Para funcionar corretamente, essa técnica requer uma configuração específica via Webpack. Nesse caso, o Webpack será o responsável por executar toda a magia entre o HTML, o JavaScript e o CSS.

```
/* style.css */

.button {
  background: #333;
}

// button.js

import styles from './style.css';

const Button = () => (
  `<button class='${ styles.button }'>Click!</button>`
);
```

Figura 27 - Exemplo de CSS modules

A última forma de usar o CSS dentro de componentes React é o tradicional *import* de um arquivo externo CSS onde se concentram todas as regras de estilo. Essa é a forma mais comum de utilização em qualquer aplicação, seja em React ou outro tipo de *framework*, e que mantém os ficheiros CSS independentes do código HTML e JavaScript, além de não ser necessário nenhuma instalação ou configuração extra. Esta será a escolha maioritária neste projeto, tal como utilizado no anterior.

```
1  import React from 'react';
2  import './Box.css';
3
4  const Box = () => (
5    <div className="Box">
6      <p className="Box_text">Folhas de Estilo</p>
7    </div>
8  );
9
10 export default Box;
```

Figura 28 - Exemplo de CSS com folha de estilo

2.5.4.5 GraphQL

A API com a qual a aplicação React irá comunicar, foi implementada usando Laravel e GraphQL, o que significa que os pedidos efetuados terão que ter a estrutura em GraphQL também.

O GraphQL foi criado em 2012 pelo Facebook, contudo apenas foi lançado publicamente em 2015. [45] Resultou da necessidade de melhorar o *feed* de notícias do Facebook, uma vez que este se tornava cada vez mais complexo e com um mau desempenho que provocava bloqueios com frequência. Surgiu então o GraphQL como linguagem de consulta alternativa às APIs comuns e modelo de dados hierárquico, onde os clientes é que decidem os dados que são retornadas da API. [46]

Este foi concebido com o objetivo de tornar as APIs mais rápidas, flexíveis e intuitivas para os programadores. Enquanto um pedido normal a uma API irá ter uma resposta standard, onde o objeto JSON da mesma pode conter por exemplo 30 campos, num pedido GraphQL a resposta é um também um objeto JSON, mas que contém apenas os campos solicitados. Um pedido apenas com os campos necessários, será um pedido com menos informação, que por consequente será um pedido de menor tamanho (pudendo haver uma redução de MB para KB) que por sua vez será um pedido mais rápido de ser executado.

Segundo o RedHat [47], o GraphQL tem imensas vantagens, mas também algumas desvantagens. Como vantagens temos que, os clientes recebem exatamente o que solicitam, sem mais dados do que o necessário. Os tipos de dados são bem definidos, o que reduz as falhas de comunicação entre o cliente e o servidor. Um cliente pode solicitar uma lista de tipos de dados disponíveis, o que é ideal para gerar documentação automaticamente. O GraphQL não determina a arquitetura especifica da aplicação, podendo ser introduzido em uma API REST existente sem influenciar o que já existe.

Já como desvantagens, temos que os programadores terão que enfrentar uma grande curva de aprendizagem. O GraphQL direciona muito do trabalho de consulta de dados para o servidor, o que aumenta a complexidade para os programadores. Já o armazenamento em cache é mais complexo do que na arquitetura REST.

Apesar da complexidade inicial, é possível verificar que as vantagens são enormes na utilização do GraphQL, e podemos afirmar que a decisão na escolha recaiu principalmente na redução da sobrecarga de transferência de dados em relação às APIs habituais, em termos de quantidade de dados transferidos desnecessariamente e do número de consultas separadas necessárias para fazê-lo.

```
{
  human(id: "1000") {
    name
    height
  }
}
```

```
{
  "data": {
    "human": {
      "name": "Luke Skywalker",
      "height": 1.72
    }
  }
}
```

Figura 29 - Exemplo de um pedido GraphQL [60]

2.5.5 Ferramentas de apoio

Durante a elaboração do projeto, foram utilizadas diversas ferramentas de apoio. Uma destas ferramentas foi já mencionada anteriormente no ponto 2.4.3, sendo ela o JIRA, utilizado para a gestão das tarefas a serem implementadas.

Outra ferramenta utilizada e essencial para qualquer programador é um sistema de controlo de versões. Este é um sistema que rastreia e gerência as alterações efetuadas no código de software, assim como em qualquer ficheiro do projeto, mantendo um registo numa espécie de base de dados. Será utilizado como sistema de controlo de versões o *GIT*[3], sendo este talvez um dos mais utilizados. Uma vez que o *GIT* é apenas acessível pela linha de comandos, será utilizado em conjunto com outro software que atribui ao GIT uma interface gráfica, sendo este o *Sourcetree*[4].

De modo a auxiliar na visualização dos pedidos GraphQL, foi utilizada a aplicação *Firecamp*[5], que após indicar o URL do servidor, assim como as credenciais necessárias, possibilita o envio dos pedidos GraphQL e a visualização da respetiva resposta.

Devido à pandemia do COVID-19, grande parte do projeto foi elaborada remotamente, desta forma e para manter a comunicação com a restante equipa, foram utilizadas ferramentas. O *Slack*[6] foi utilizado como ferramenta de chat entre os funcionários da empresa, de forma a haver uma constante comunicação. Como serviço de email, foi utilizado o *Thunderbird*[7], uma vez que o mesmo permite a utilização dos servidores privados de email da empresa, permitindo um aumento da segurança nos mesmos. Já para as reuniões remotas, quer de *Scrum*, quer de análises de risco, foi utilizado o *Webex*[8] que pertence à empresa Cisco, que por esse motivo já tem implícito o tipo de segurança que a mesma possui.

2.5.6 Conclusão

Neste capítulo verificamos que a tecnologia que será utilizada no projeto, será então o React, quais as suas vantagens e algumas das suas características e funcionalidades. Relativamente às opções de estilo CSS, neste projeto será utilizada as folhas de estilo, uma vez que separa o estilo da parte visual da restante estrutura do código, por uma questão de gosto pessoal. Com o React, será também utilizado o GraphQL para comunicar com o *back-end*.

Por fim, foram também enunciadas as restantes ferramentas utilizadas durante o projeto e que foram de extrema utilidade para que ocorresse uma boa comunicação entre a equipa e a elaboração de uma "cópia de segurança" do projeto graças ao sistema de controlo de versões.

2.6 Bibliotecas Utilizadas

Tal como mencionado anteriormente, o React por ser uma biblioteca de JavaScript, pode ser complementado com diversas outras pequenas bibliotecas que complementam ou adicionam

[3] Disponível em https://git-scm.com
[4] Disponível em https://sourcetreeapp.com
[5] Disponível em https://firecamp.io
[6] Disponível em https://slack.com
[7] Disponível em https://thunderbird.net
[8] Disponível em https://www.webex.com

funcionalidades pretendidas. Abaixo serão então enunciadas e descritas as bibliotecas que foram adicionadas ao projeto da GTS.

2.6.1 Eslint e Prettier

No desenvolvimento do *front-end* com ReactJS serão utilizados o Eslint[9] e o Prettier[10]. O Eslint é responsável por identificar padrões de código em desacordo com as regras pré-estabelecidas. Já o Prettier é responsável por formatar o código de acordo com essas regras.

As vantagens de implementar estas ferramentas no projeto, são que além de facilitar a inserção de novos desenvolvedores no projeto, facilita também os *code reviews* e a manutenção do código. Se estas ferramentas forem configuradas corretamente, podem poupar diversas horas procurando por variáveis com nomes incorretos ou até mesmo os parênteses em falta.

Estas duas ferramentas vão ser instaladas no VSCode e por sua vez o projeto terá um ficheiro de configuração para cada uma destas. Desta forma o código terá um estilo predefinido tal como a escolha da indentação (por tabs ou espaços), o número de espaços por cada tab, o número máximo de caracteres por linha, etc.

2.6.2 Axios

Quando usamos serviços REST precisamos efetuar pedidos HTTP como GET, PUT, POST ou DELETE. Para tal irá ser utilizado o Axios[11] que é um cliente HTTP e irá lidar com envio e receção de dados entre a aplicação React e a API. A biblioteca é basicamente uma API que interage com os XMLHttpRequest, onde os pedidos feitos através da mesma retornam uma *promise*, sendo esta compatível com a nova versão do JavaScript - ES6. A vantagem desta biblioteca é ser pequena, leve e com uma implementação muito simples.

2.6.3 Material-UI

O Material-UI[12] é uma *framework* de *front-end open source* que no GitHub conta com 56K estrelas e 15.2K *forks*. Esta biblioteca provém do Material Design que é uma linguagem de design desenvolvida pela Google em 2014.

O Material-UI tem como lema: componentes de React para um desenvolvimento web rápido e fácil. Ou seja, é possível utilizar componentes já existentes na biblioteca poupando tempo de implementação.

Esta biblioteca irá substituir o Bootstrap usado atualmente no CodeIgniter, de modo a termos uma página responsiva tanto em computadores como smartphones, poupando a implementação de CSS para melhorar o visual da aplicação.

[9] Disponível em https://eslint.org
[10] Disponível em https://prettier.io
[11] Disponível em https://github.com/axios/axios
[12] Disponível em https://github.com/mui-org/material-ui

2.6.4 React Hook Form

No React quando é criado um formulário, é necessário indicar o comportamento do mesmo. A biblioteca React Hook Form[13] surge como meio de simplificar o processo e reduzir a quantidade de código necessário para utilizar um formulário.

A biblioteca tem já implementada os métodos de retorno dos valores inseridos no formulário assim como um maior controlo sobre a validação dos erros. Tem como vantagens ser uma biblioteca leve, sem outras dependências e que minimiza o número de renderizações para mostrar e atualizar um formulário.

2.6.5 React-jsonschema-form

A biblioteca React-jsonschema-form[14] tem como objetivo gerar automaticamente um formulário React com base em um esquema JSON. Esta biblioteca tem tal como a anterior, métodos já específicos para o retorno dos valores inseridos no formulário e a validação dos erros do mesmo.

A grande vantagem desta biblioteca é o fato de com um simples esquema JSON proveniente da API, ser possível gerar um formulário automaticamente em qualquer parte do projeto, assim como a alteração dos formulários ser maioritariamente sem alterações de código no *front-end*.

2.6.6 Json-server

Uma vez que o *front-end* e o *back-end* da nova versão da GTS estão sendo contruídos em simultâneo, em diversas situações vai ser necessário o *front-end* receber/enviar dados para conceber novas funcionalidades ou realizar testes. Informação essa que pode ainda não estar disponível na implementação da API. Deste modo irá ser utilizada a biblioteca Json-server[15] para simular esses recursos que posteriormente irão ser consumidos pela API.

A grande vantagem desta biblioteca em relação aos *mocks* é que com ela é possível: criar uma API REST com POST, PUT, PATCH ou DELETE; criar suas próprias rotas, além das predefinidas; criar filtros; criar paginação, gerar dados aleatórios e muito mais.

2.6.7 JS-Cookie

Embora o React possua os estados, stores, etc, toda a informação presente nestes meios volta ao estado inicial quando a página é recarregada. Para fazer face a esta situação, é possível utilizar três abordagens:

[13] Disponível em https://github.com/react-hook-form/react-hook-form
[14] Disponível em https://github.com/rjsf-team/react-jsonschema-form
[15] Disponível em https://github.com/typicode/json-server

- LocalStorage: a informação não tem data de expiração, é limpa pelo JavaScript ou quando limpamos a informação/cache do browser. Tem 5MB de espaço disponível para armazenar informação;
- SessionStorage: é semelhante ao LocalStorage, mas informação apenas permanece durante o tempo de sessão, ou seja, até que a tab ou o browser sejam encerrados. O volume de dados é muito superior ao da Cookie;
- Cookie: podem ser definidas tanto do lado do cliente como no lado do servidor, enquanto o LocalStorage e o SessionStorage são apenas são definidos no lado do cliente. A cookie por sua vez, tem como parâmetro a data de expiração.

Para a utilização das cookies no React, de modo a guardas os dados de autenticação, irá então ser utilizada a biblioteca JS-Cookie[16] para gestão das mesmas.

2.6.8 JS-Base64

A biblioteca JS-Base64[17] será utilizada para codificar e descodificar a informação com base64 de modo a não ficar tão exposta e de fácil visualização. Após a instalação da mesma, é possível por exemplo utilizar as funções do JavaScript como atob() e btoa() ou decode() e encode() para descodificar e codificar a informação com recurso ao base64.

2.6.9 React-toastify

Como sistema de notificações e alertas, irá ser utilizado a biblioteca React-toastify[18]. Foram verificadas várias bibliotecas para este efeito, contudo esta destacou-se por várias razões. O primeiro ponto foi a grande diferença de downloads em relação às restantes, seguidamente o tempo de manutenção assim como a facilidade de uso da mesma e por fim a vasta personalização possível.

2.6.10 React-multi-carousel

Na nova plataforma iremos ter presente um *slider* na página inicial, que irá apresentar os produtos assim como as promoções ativas na presente data. Este será dinâmico, sendo o seu conteúdo armazenada na base de dados, de forma que quando aconteça uma nova campanha não seja necessário grande desenvolvimento e essencialmente adicionar uma nova entrada na base de dados. Para facilitar este processo, foi utilizada a biblioteca react-multi-carousel[19]. Esta escolha incidiu na facilidade de uso da mesma, assim como a diversa personalização, entre elas o número de elementos visíveis ou que é possível mover em simultâneo, a personalização das setas de controlo, *autoplay*, responsividade, etc.

[16] Disponível em https://github.com/js-cookie/js-cookie
[17] Disponível em https://github.com/dankogai/js-base64
[18] Disponível em https://github.com/fkhadra/react-toastify
[19] Disponível em https://github.com/YIZHUANG/react-multi-carousel

2.6.11 React-scroll

As animações de *scroll* automático na plataforma despoletadas por algum evento, vão ser realizadas com recurso à biblioteca react-scroll[20], sendo assim possível conjugar com os componentes de React. Nas configurações da mesma é possível fazer *scroll* para o fundo, topo, para uma posição especifica, ou um valor abaixo dessa posição, etc. Também é posição especificar o atraso da animação, o estado da mesma, o tipo de animação, entre outras.

2.6.12 React-device-detect

A biblioteca React-device-detect[21] será utilizada para a deteção do tipo de dispositivo usado para aceder à plataforma. Com a mesma é possível se o dispositivo é mobile, tablet ou PC, o tipo de browser, o tipo de dispositivo, o nome e versão do sistema operativo, o "*engine name*" e a versão, o "*user agent*", entre outros.

2.6.13 i18next

Para as traduções da nova plataforma irá ser utilizado a biblioteca i18next[22]. A vantagem de utilizar esta biblioteca, é que a mesma está disponível na maioria das plataformas, fazendo com que seja uma das mais utilizadas e por sua vez exista manutenção de forma mais frequente.

Esta utiliza ficheiros diferentes para cada língua que seja necessário suportar, onde cada ficheiro contém uma estrutura de objeto. Nas configurações ao alterar o idioma, permite verificar qual o que está selecionado e se novo idioma pretendido não fizer parte dos possíveis de serem selecionados, escolhe o que está por defeito.

É possível também instalar a aplicação BabelEdit para extrair os ficheiros a traduzir do código, e completar as traduções inexistentes.

2.6.14 i18next-browser-languagedetector

A biblioteca i18next-browser-languagedetector[23] é um plugin que complementa a biblioteca anterior. A sua utilização é simples e permite alterar/armazenar o idioma selecionado de várias formas, entre as quais: *path*, sub-domínio, *querystring, cookie, session, localstorage*, idioma do navegador, etc. É também possível selecionar uma ordem pelo qual deve verificar o idioma a selecionar.

[20] Disponível em https://github.com/fisshy/react-scroll
[21] Disponível em https://github.com/duskload/react-device-detect
[22] Disponível em https://github.com/i18next/i18next
[23] Disponível em https://github.com/i18next/i18next-browser-languageDetector

2.6.15 Jest, Enzyme e React Testing Library

A biblioteca Jest[24] foi criada pelo Facebook e é utilizada para os testes unitários e de integração, assim como os *snapshot* que são uma espécie de foto do código, de modo a comparar a mesma, com a próxima execução do *snapshot*, para garantir que o resultado é exatamente o mesmo. Além da biblioteca anterior, também o React Testing Library[25] e a Enzyme[26], criada pelo Airbnb, são bibliotecas utilizadas na realização dos testes para a plataforma.

Estas têm como vantagens a simplicidade e velocidade da execução de testes. São utilizadas as três bibliotecas, uma vez que cada uma tem mais facilidade em executar certos tipos de testes. O Jest será usado para os *snapshots* dos componentes, onde também será utilizado o Enzyme, para a simulação do componente a ser feito o *snapshot*. Já o React Testing Library será utilizado para as asserções, ou seja, a verificação das condições dos testes.

2.7 Conclusão

Neste capítulo abordamos o que são as assinaturas digitais, os algoritmos mais utilizados para a implementação das mesmas e qual o formato padrão de um certificado. Foi também abordado o que é uma infraestrutura de chaves públicas e demonstrado como funciona a mesma na empresa ACIN.

De modo a ter um panorama geral dos produtos e tecnologias utilizadas pela principal concorrência da GTS, foi feito uma análise sobre o que é utilizado e comercializado pelos mesmos e é possível chegar à conclusão que a GTS possui uma maior diversidade de produtos e utiliza tecnologias semelhantes em relação aos seus opositores.

Em relação à metodologia de trabalho, foram analisadas as mais utilizadas pelas empresas da mesma área. Tal como mencionado no ponto 2.5, a empresa possui algumas restrições em relação às tecnologias a serem utilizadas, contudo foi feita uma análise sobre as mais utilizadas atualmente de forma a verificar se as mesmas correspondiam às utilizadas pela empresa. Com base na revisão, chegou-se à conclusão de que uma das mais vantajosas era o React, tendo sido a escolhida para este projeto e uma das mais utilizadas dentro da empresa, pelo que o resultado foi de encontro às expectativas. Além do React, foi também escolhido o GraphQL, devido às vantagens já enunciadas anteriormente.

Já no que diz respeito às ferramentas de apoio e às bibliotecas utilizadas, foi efetuado um levantamento de todas as utilizadas na implementação deste projeto, assim como os benefícios e os motivos para as mesmas serem utilizadas.

[24] Disponível em https://github.com/facebook/jest
[25] Disponível em https://github.com/testing-library
[26] Disponível em https://github.com/enzymejs/enzyme

3. Solução

Tendo em conta o que foi discutido no ponto 1.5 e o que foi sugerido na solução do ponto seguinte (1.6), avançamos com a revisão de literatura apresentada nos capítulos anteriores. O projeto idealizado pela empresa para a GTS, envolve um desenvolvimento a longo prazo, de modo a ter uma área de cliente, área de compra, área de notícias, área de assinatura, *backoffice* para departamento de apoio ao cliente e para o design.

Contudo, devido às restrições de tempo, nesta tese iremos apenas contemplar a área de compra dos produtos comercializados pela GTS, nomeadamente o desenvolvimento em *front-end* da plataforma. Este será implementado em React tal como mencionado no ponto 2.5.4 e por sua vez a API será implementada pela restante equipa com a *framework* Laravel. Por sua vez, a comunicação intermédia do *front-end* em React com o *back-end* em Laravel será feito com pedidos de GraphQL, tal como já mencionado no ponto 2.5.4.5.

De acordo com a informação apresentada no capítulo 2.4 sobre a Engenharia de Software, iremos de seguida apresentar o processo que foi seguido.

3.1 Metodologia seguida

No capítulo 2.4.3 foram apresentadas as principais metodologias. Na empresa ACIN, e tal como apresentado no ponto 2.4.4, todos os projetos por esta elaborados seguem a metodologia *Scrum*, por esta se adequar melhor a grandes projetos, havendo um melhor fluxo na equipa através das reuniões, e pudendo ter uma aplicação executável no fim de cada *sprint*. Por norma na ACIN, antes de cada *sprint* temos uma reunião para saber as tarefas de cada membro da equipa, assim como atribuir os *stories points* (grau de dificuldade da tarefa).

Em média, cada *sprint* dura 4 semanas, sendo que reservamos sempre a última semana para os testes aplicacionais e consequentemente correção dos *bugs* encontrados. É também dada uma grande importância pela empresa a todas as análises de riscos necessárias, de modo a existir um bom planeamento em caso de possíveis falhas. Nas reuniões, temos por vezes presente o *produt owner*, de modo a obter feedback do mesmo sobre as implementações realizadas nos *sprints* anteriores e as implementações planeadas para os *sprints* seguintes.

3.2 Requisitos de software

Nos requisitos de software temos presente as necessidades e limitações que a aplicação a ser implementada vai possuir. Estes dividem-se em duas categorias, requisitos funcionais e requisitos não funcionais.

É importante efetuar uma profunda e correta análise dos requisitos, uma vez que estes são a base do resultado final do projeto. De modo a ilustrar esta importância, temos abaixo uma conhecida imagem na Engenharia de Software, denominada de *"tree swing analogy"* (análise do balanço na árvore), que surgiu na década de 1970 e que descreve a maneira como cada departamento interpreta e implementa um requisito que não está claro e preciso, tornando o produto final diferente do esperado.

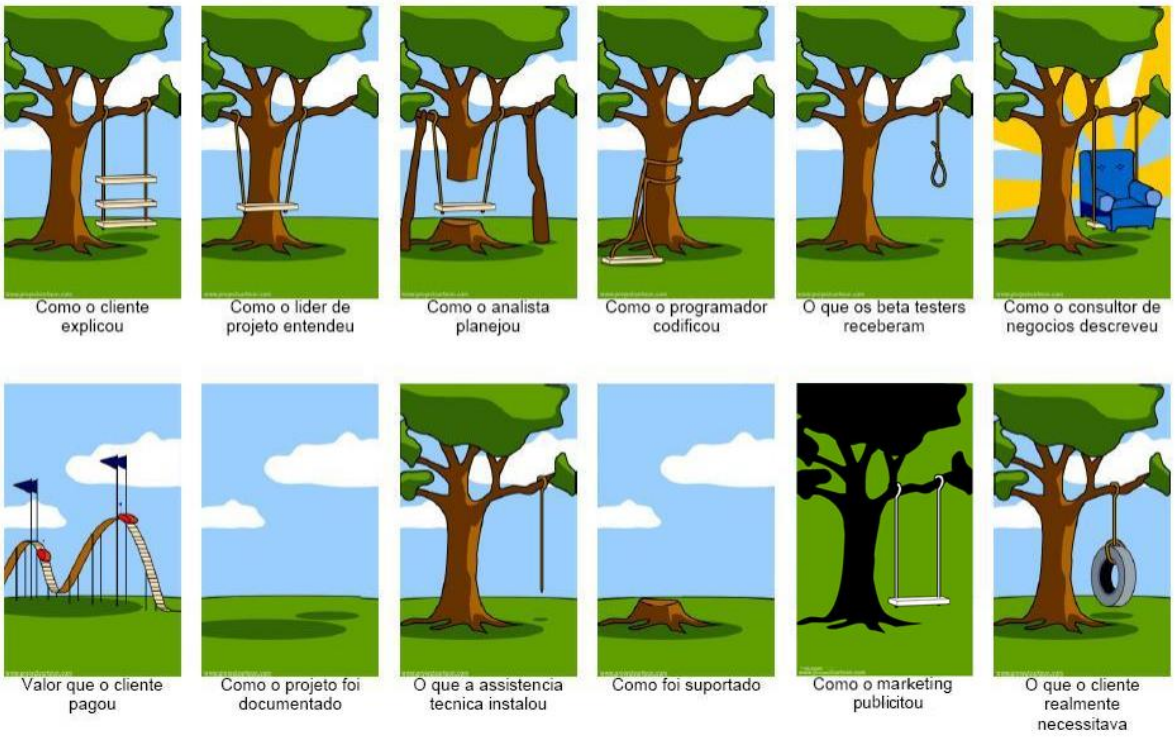


Figura 30 - Tree Swing Story [48]

3.2.1 Requisitos funcionais

Segundo o SWEBOK, os requisitos funcionais indicam os serviços que a aplicação deve executar e suportar. [34] Estes descrevem pormenorizadamente as funcionalidades da aplicação, assim como documentar como o sistema reage a situações especificas e que comportamentos deve possuir, assim como indicar o que a aplicação não deve fazer. [49]

Tal como já mencionado anterior no ponto 1.6, do decorrer das diversas reuniões entre a equipa de desenvolvimento, a equipa de design, a equipa de marketing, a equipa de apoio ao cliente, o departamento de contabilidade, o gestor de produto e a administração, emergiram os seguintes requisitos funcionais que a nova plataforma deveria cumprir:

RF1	O sistema deverá registar todas as operações realizadas pelo utilizador
RF2	O sistema deverá uniformizar os métodos de comunicação
RF3	O sistema deverá permitir o registo novos utilizadores
RF4	O sistema deverá permitir a alteração de dados pessoais
RF5	O sistema deverá permitir a recuperação das credenciais de acesso
RF6	O sistema deverá permitir o envio de email de confirmação da conta
RF7	O sistema deverá permitir o envio de email para recuperação de dados
RF8	O sistema deverá permitir a criação de um novo perfil de utilizador identificado como entidade

RF9	O sistema deverá a criação de utilizadores na TX através da API
RF10	O sistema deverá permitir a remoção de contas que não confirmem o email e que não demonstrem "interesse" na plataforma durante um período de 6 meses (RGPD)
RF11	O sistema deverá permitir a eliminação de contas (RGPD)
RF12	O sistema deverá permitir a extração de dados de conta (RGPD)
RF13	O sistema deverá permitir a aquisição de produtos por parte de utilizadores
RF14	O sistema deverá permitir a aquisição de produtos por parte de entidade
RF15	O sistema deverá permitir a atribuição de certificados a outros utilizadores
RF16	O sistema deverá permitir a suspensão de certificados
RF17	O sistema deverá permitir a revogação de certificados
RF18	O sistema deverá permitir a validação dos pedidos
RF19	O sistema deverá permitir a inativação de selos temporais
RF20	O sistema deverá gerir conflitos entre carrinhos de compras
RF21	O sistema deverá permitir a aquisição de produtos através de pacotes
RF22	O sistema deverá permitir a aquisição de produtos através de campanha promocional
RF23	O sistema deverá permitir a validação dos dados preenchidos no formulário de pagamento
RF24	O sistema deverá gerar dados de pagamento através da Paypay
RF25	O sistema deverá gerar faturas através do iGEST
RF26	O sistema deverá permitir a gestão dos códigos de descontos
RF27	O sistema deverá permitir a gestão dos dados de faturação do cliente
RF28	O sistema deverá associar um certificado a um pedido
RF29	O sistema deverá permitir a extração dos dados da componente pública do certificado
RF30	O sistema deverá permitir a renovação de um certificado
RF31	O sistema deverá permitir assinar documentos PDF
RF32	O sistema deverá permitir assinar documentos XML
RF33	O sistema deverá permitir o registo de parceiros
RF34	O sistema deverá possuir uma área para dados estatísticos
RF35	O sistema deverá permitir formulários dinâmicos para a integração com outras plataformas
RF36	O sistema deverá associar pedidos a parceiros
RF37	O sistema deverá atribuir comissões aos parceiros
RF38	O sistema deverá permitir o levantamento da comissão
RF39	O sistema deverá permitir a utilização da comissão na plataforma da GTS
RF40	O sistema deverá permitir que os utilizadores sem autenticação possam visualizar a informação dos produtos, assim como o preçário
RF41	O sistema deverá permitir que os utilizadores sem autenticação possam adicionar/remover produtos do carrinho de compras

Tabela 6 - Tabela de requisitos funcionais

3.2.2 Requisitos não funcionais

Já os requisitos não funcionais, também conhecidos como requisitos de qualidade, definem as restrições do sistema e como este irá proceder. [34] Por norma, estes são colocados pelo cliente, de forma a garantir que o produto final seja de boa qualidade e estão agrupados em algumas das seguintes categorias: segurança, desenvolvimento, desempenho, usabilidade, disponibilidade, portabilidade, fiabilidade, manutenção, flexibilidade e expansibilidade. A descrição dos requisitos não funcionais deve possuir atributos quantitativos para que possam ser posteriormente testados e garantir que os requisitos são cumpridos.

RNF1 Segurança	O sistema deverá garantir apenas um acesso autorizado à plataforma por cada utilizador
RNF2 Segurança	O sistema deverá encriptar a palava passe
RNF3 Segurança	O sistema deverá possuir identificação para diferentes tipos de utilizador: administrador, super administrador, utilizador, entidade, etc…
RNF4 Segurança	O sistema deverá ter um tempo limite máximo de sessão de 20 minutos
RNF5 Desenvolvimento	O sistema deverá integrar com o Passport package
RNF6 Desenvolvimento	O sistema deverá combinar o Passport + CSRF
RNF7 Desenvolvimento	O sistema deverá integrar com o GraphQL
RNF8 Desenvolvimento	O sistema deverá ter o back-end implementado em Laravel
RNF9 Desenvolvimento	O sistema deverá ter o front-end implementado em React
RNF10 Desenvolvimento	O sistema deverá ter implementado a autenticação de client application
RNF11 Desempenho	O sistema deverá ter um tempo limite máximo de espera pela resposta da API de 5 segundos
RNF12 Desempenho	O sistema deverá ter um tempo limite máximo de espera pelo carregamento de páginas de 5 segundos
RNF13 Usabilidade	O sistema deverá ser simples e intuitivo
RNF14 Disponibilidade	O sistema deverá possuir redundâncias para estar sempre disponível
RNF15 Disponibilidade	O sistema deverá possuir memória suficiente para armazenar os dados dos clientes
RNF16 Portabilidade	O sistema deverá ser responsivo
RNF17 Portabilidade	O sistema deverá ser compatível com os browsers mais utilizados

RNF18 Fiabilidade	O sistema deverá ter capacidade para recuperar os dados perdidos na última operação caso exista uma falha
RNF19 Manutenção	O sistema deverá apresentar um aviso aos utilizadores quando estiver agendada qualquer tipo de manutenção
RNF20 Manutenção	O sistema deverá estar funcional no máximo 4 horas após um período de manutenção
RNF21 Flexibilidade	O sistema deverá ser implementado de modo a permitir alterações sempre que necessárias
RNF22 Expansibilidade	O sistema deverá ser implementado de modo a permitir a sua expansão

Tabela 7- Tabela de requisitos não funcionais

3.3 Processos

Um fluxograma é um tipo de diagrama (representação visual de um determinado conceito ou esquema), que descreve um processo ou algoritmo. Este representa uma sequência de eventos, os passos de processamento e as decisões que são preciso tomar durante o processo e tem como objetivo tornar esse fluxo fácil de entender, assim como saber todas as ações possíveis. [50]

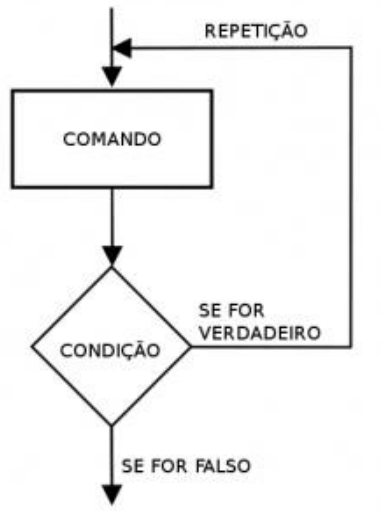


Figura 31 - Exemplo básico de um fluxograma [51]

Os fluxogramas são compostos por figuras geométricas e também por setas que indicam o fluxo a ser seguido. As formas mais ovais nas pontas, representam o início e fim do fluxo, os quadrados simbolizam as ações que precisam ser executadas e os losangos identificam os pontos onde é necessário haver uma decisão. Por fim temos o paralelograma que representa as entradas e saídas de informação.

Figura 32 - Componentes de um fluxograma

Para a elaboração dos fluxogramas seguintes, a equipa de desenvolvimento efetuou várias reuniões, juntando a equipa de *front-end* e a equipa de *back-end*, de modo a definir bem os fluxos de comunicação, assim como todas as ações que o utilizador deverá tomar e de que forma o sistema se deverá reagir.

3.3.1 Comunicação entre *front-end* e *back-end*

No diagrama seguinte, é possível visualizar como será implementada a comunicação entre o *front-end* e o *back-end*. A aplicação cliente, ou seja, o *front-end* em React, irá comunicar maioritariamente com a API da GTS, sendo esta o intermediário de todo o processo. Por sua vez a API vai comunicar com todos os módulos, com a base de dados, com o sistema de armazenamento e devolver a resposta para o *front-end.*

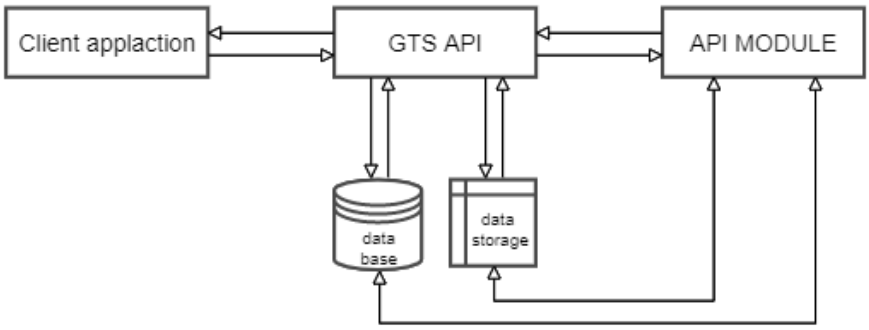


Figura 33 - Diagrama genérico de comunicação entre front-end e back-end

Para a comunicação entre o *front-end* e o *back-end*, irá também ser necessário distinguir um utilizador autenticado que possui acesso a informação "privada", de um utilizador que não está autenticado e apenas pode ver informações públicas. Neste processo, descrito na Figura 34, o utilizador acede à zona de autenticação e coloca as credencias de acesso. Caso estas estejam corretas, irá receber um "*token*" de acesso, que não é mais nem menos que um conjunto de letras

e números único, que tem uma validade. Assim, após a autenticação, cada pedido que seja efetuado à API deve levar junto o "*token*" para identificar o utilizador em sessão e confirmar que o mesmo tem acesso à informação solicitada.

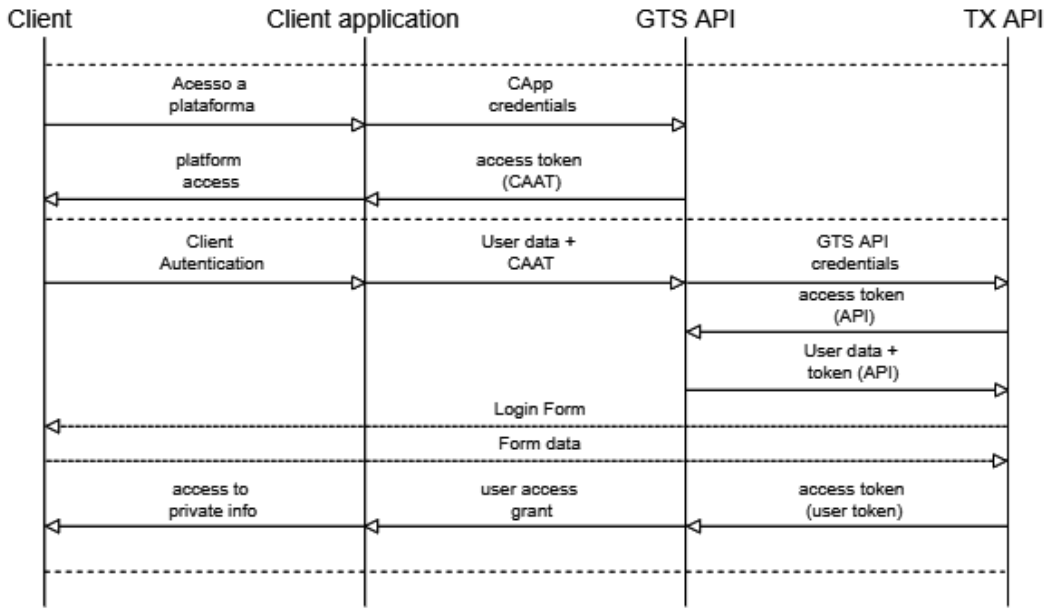


Figura 34 - Exemplo da comunicação para autenticar um utilizador

3.3.2 Processo de registo, autenticação e recuperação de conta

No diagrama abaixo (Figura 35), temos o fluxo do processo inicial de registo de um novo cliente na plataforma, onde é necessário confirmar que aquele email pertence à pessoa em questão. Para tal, o utilizador deve colocar o seu endereço de email e seguidamente irá receber um código na sua caixa de correio. Esse código tem um tempo de validade de 15 minutos, e antes de ser enviado, é validado se aquele email já está registado. Caso o utilizador não receba o código no seu email, deve aguardar 1 minuto para conseguir enviar outro, de maneira a não gerar *spam*. Se o código estiver correto, o utilizador é encaminhado para o passo seguinte do registo, onde coloca as suas informações pessoais.

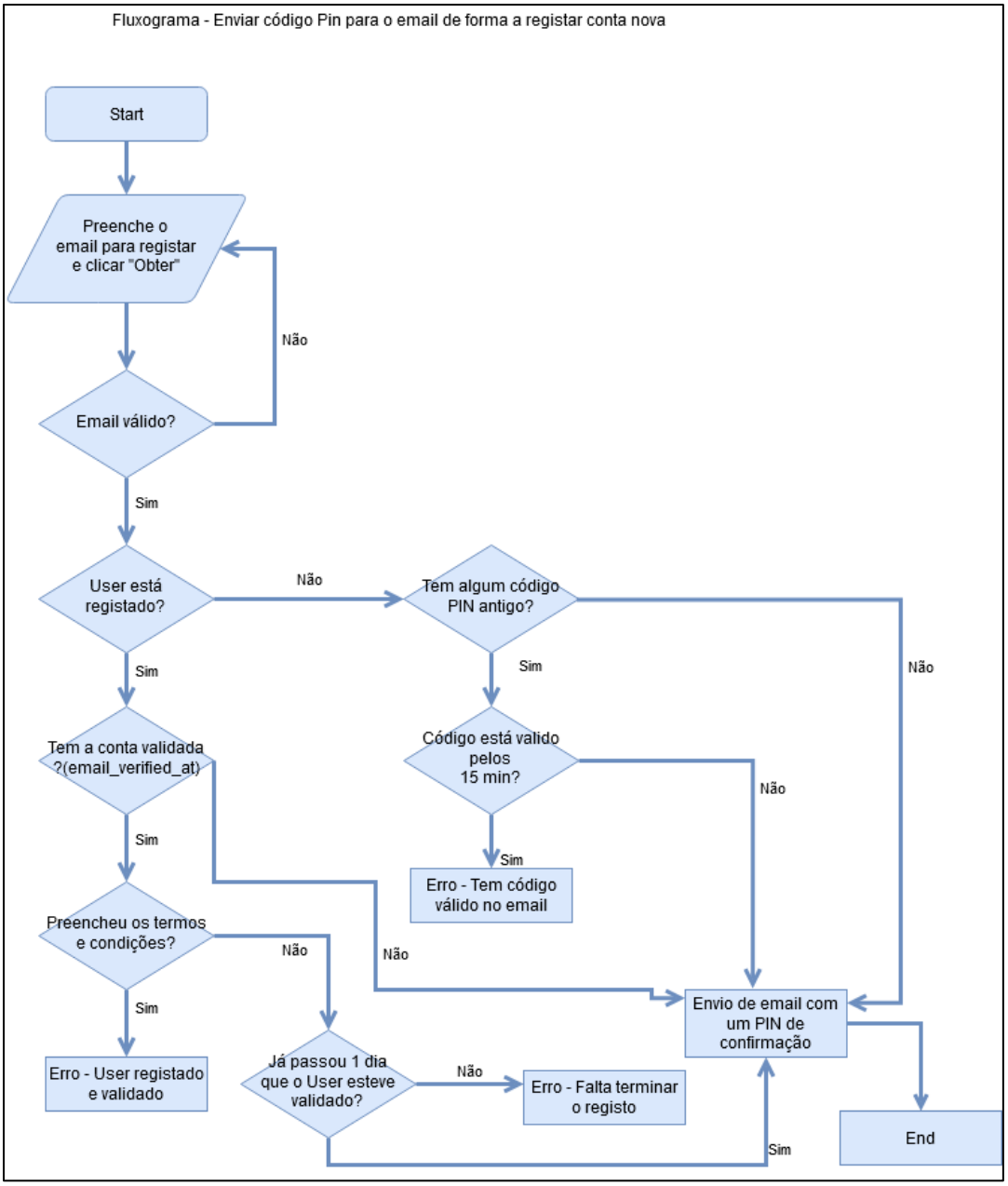


Figura 35 - Fluxograma inicial de um novo registo

No passo seguinte após o email estar validado, é apresentado ao utilizador um formulário onde deve preencher os dados pessoais necessários para a criação de um novo registo. Nomeadamente o tipo de utilizador (singular ou coletivo), o nome do utilizador em questão e a palavra-passe pretendida, assim como a confirmação da mesma. Se algum dos dados não preencher os requisitos necessários, será apresentada uma mensagem de erro, caso contrário, é verificado se o utilizador pretende receber newsletters para efetuar a comunicação com o sistema destas, e é então criado um novo registo do utilizador na base de dados.

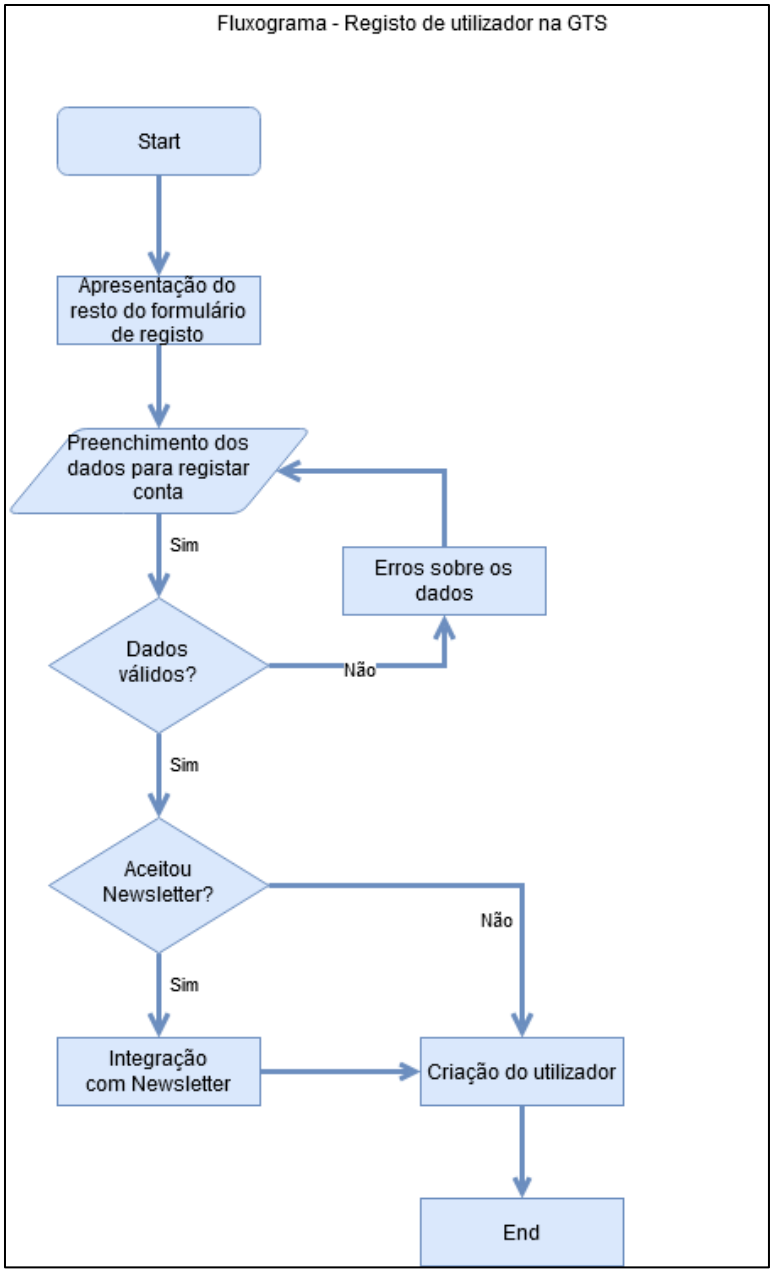


Figura 36 - Processo final da criação de um novo registo

No processo de autenticação do utilizador, este ao selecionar a opção de login é encaminhado para o PKI (Infraestrutura de chaves públicas) da GTS. É apresentado um formulário e o cliente deve introduzir os dados da sua conta, nomeadamente o email e password. Caso as credenciais estejam incorretas é apresentado uma mensagem de erro. Se estas forem válidas, é devolvido um *token*, tal como já mencionado anteriormente na comunicação do *front-end* com o *back-end*. Este *token* é enviado para a API da GTS, sendo que esta com recurso ao mesmo, gera

os dados de OAuth, nomeadamente o "*access token*" e o "*refresh token*", de modo a ser possível uma comunicação na zona "privada" da GTS.

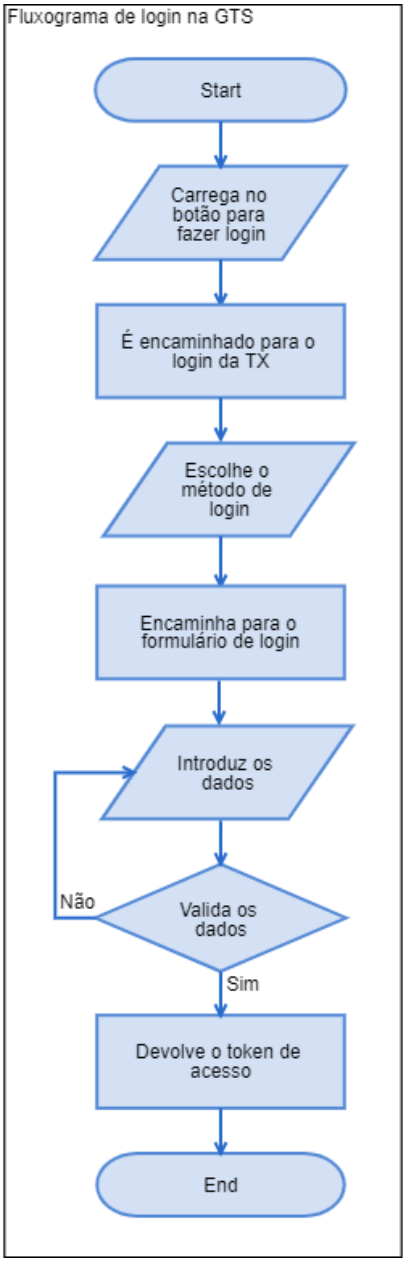


Figura 37 - Fluxo de login

Já na recuperação de uma conta na GTS, esta pode ocorrer em duas situações, um utilizador possui já certificados na sua conta, ou então a conta não contém certificados. No primeiro caso e de forma a ser possível validar que é mesmo o titular da conta a fazer o pedido, é necessário preencher um formulário com os dados pessoais e anexar o cartão de cidadão do titular de modo que os administradores de registo possam validar a sua entidade. Esta medida tenta aumentar a

segurança, de modo a evitar que a conta seja recuperada por alguém que não o próprio utilizador, uma vez que apenas este deverá ter em sua posse o seu cartão de cidadão. É importante que apenas o próprio utilizador tenha acesso a uma conta com certificados, uma vez que desta forma consegue "assinar" documentos com valor probatório em seu nome. O fluxograma abaixo apresentado, simboliza a segunda opção onde o utilizador não tem certificados associados à sua conta. Neste caso o utilizador preenche um formulário onde introduz o seu endereço eletrónico, momentos depois irá receber na sua caixa de correio um email com um link. Ao ser acedido, é validado a hash contida no link, caso a mesma seja inválida é apresentado uma mensagem de erro, caso contrário, o cliente deve introduzir a nova palavra-passe assim como a confirmação da mesma. Esta palavra-passe deverá corresponder a certos requisitos para ser válida, sendo eles os seguintes:

- conter pelo menos 10 caracteres;
- conter 3 destes elementos: Letras maiúsculas; Letras minúsculas; Números; Caracteres especiais (! ' ; , : _ - / ( ) + ? * .);

Após terminar um processo, o utilizador recebe um novo email a informar que a palavra-passe foi modificada e que caso não tenha sido o próprio a fazer este pedido, deve pressionar o link enviado para bloquear os acessos à conta, prevenindo acessos indevidos à mesma. Posteriormente seria necessário contatar os administradores de registo para validar a identidade do cliente e desbloquear a conta, devolvendo o acesso ao utilizador devido.

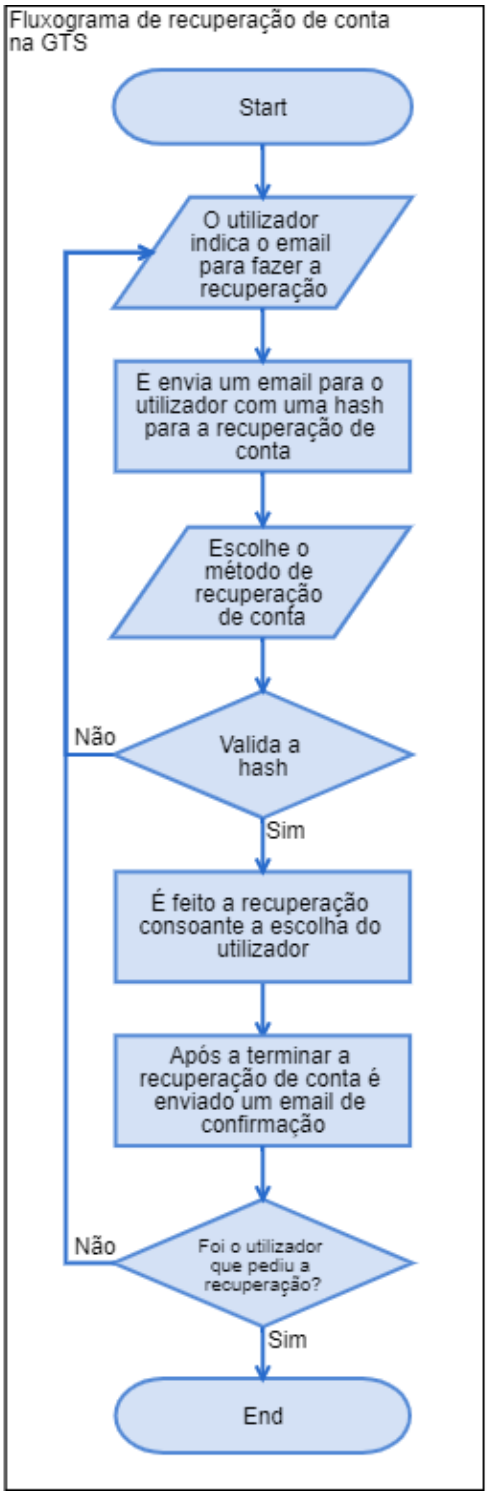


Figura 38 - Fluxo de recuperação de conta

3.3.3 Fluxo de aquisição de um produto

Para a aquisição de um produto, é necessário primeiro que tudo autenticar o utilizador, de modo a ser possível confirmar se o cliente em questão possui uma conta do tipo normal ou de entidade. Este pormenor é importante para o processo de atribuição do certificado no fim do procedimento.

Caso a conta seja do tipo utilizador normal, é apresentado um formulário para o preenchimento das informações relevantes. Caso os dados sejam inválidos, será apresentada uma mensagem de erro, coso contrário é verificado se o utilizador selecionou no formulário se os produtos em questão são para o próprio, ou se está efetuando a compra para outro utilizador. Se forem para o próprio, as informações são guardadas na base de dados, se forem para outro utilizador, primeiro é verificado se o mesmo possui uma conta no email introduzido no formulário. Se o utilizador já estiver registado, recebe um email a notificar a compra dos produtos, caso contrário, é criada uma nova conta no email introduzido, e o utilizador receberá um email para criar uma palavra-passe para esta nova conta. Tal como no procedimento anterior, os dados são depois armazenados na base de dados.

Se o tipo de conta for por sua vez uma entidade, é novamente preenchido o formulário e validados os dados introduzidos, daqui para a frente o processo é semelhante ao do utilizador que efetua uma compra para outra pessoa, tal como descrito anteriormente.

Qualquer um dos três casos descritos anteriormente, chegam ao mesmo ponto, é verificado se o produto comprado é um certificado qualificado, pois em caso afirmativo, tem de ser confirmada a validade do utilizador, e o mesmo recebe no email o contrato que deve ser assinado e devolvido aos administradores de registo. Para a validação do utilizador, a mesma pode ser feita de três formas:

- agendamento presencial nas instalações da ACIN, onde o utilizador deverá assinar os termos e condições da compra de produtos, assim como fornecer o seu cartão de cidadão de modo a comprovar a sua identidade;
- agendamento de uma videochamada entre o cliente e os administradores de registo para validar a identidade, sendo que durante a mesma e tal como no procedimento anterior, o utilizador deve fornecer o seu cartão de cidadão para comprovar que a pessoa em videochamada é a mesma que a titular do cartão de cidadão;
- através da plataforma Autenticação.gov, onde através do cartão de cidadão ou da chave móvel digital, é confirmada a identidade do utilizador. Neste procedimento, a GTS recebe as informações presentes no cartão de cidadão por parte da plataforma Autenticação.gov e compara os mesmos com os fornecidos pelo utilizador no ato da compra. Se corresponderem, o pedido fica validado, caso contrário poderá corrigir os dados da compra de modo a corresponderem aos dados do cartão de cidadão.

Caso alguns destes três processos falhe, é agendada uma nova validação de identidade. Só após a mesma estar válida é que o utilizador pode gerar o certificado e finalizar o pedido de compra.

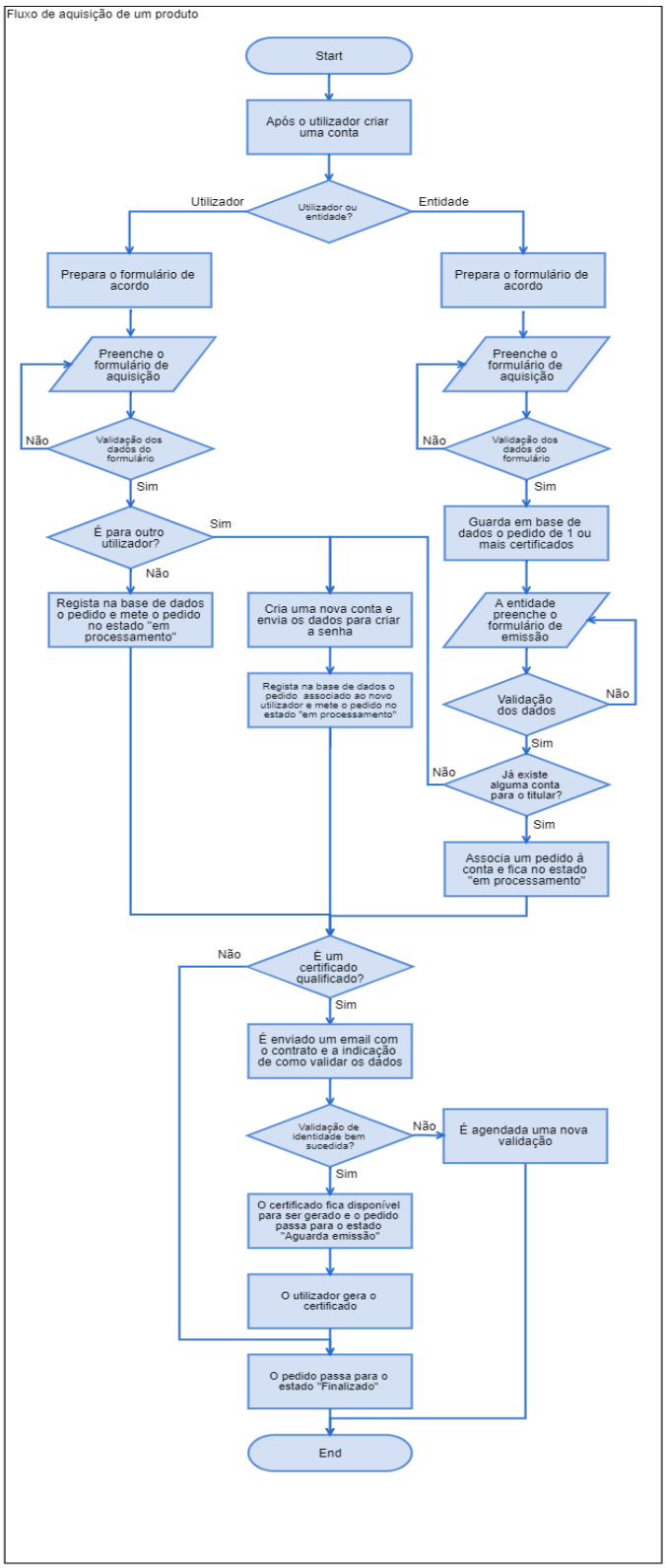


Figura 39 - Fluxo de aquisição de um produto

3.3.4 Gestão de conflitos na aquisição de produtos

Durante o processo de compra, se o utilizador iniciar o processo sem estar autenticado, pode acontecer que após a autenticação bem-sucedida, este já possua artigos no carrinho adicionados anteriormente e armazenados na base de dados. Caso isto suceda e de forma a mitigar a situação será pedido ao utilizador para optar por uma das três opções possíveis:

- A primeira escolha é a opção de juntar os novos produtos do carrinho de compras, ao carrinho que já tenha sido armazenado anteriormente;
- A segunda escolha é a opção de descartar o pedido feito anteriormente e prosseguir com o pedido atual;
- A terceira e última opção é descartar o novo pedido e continuar com o pedido anterior.

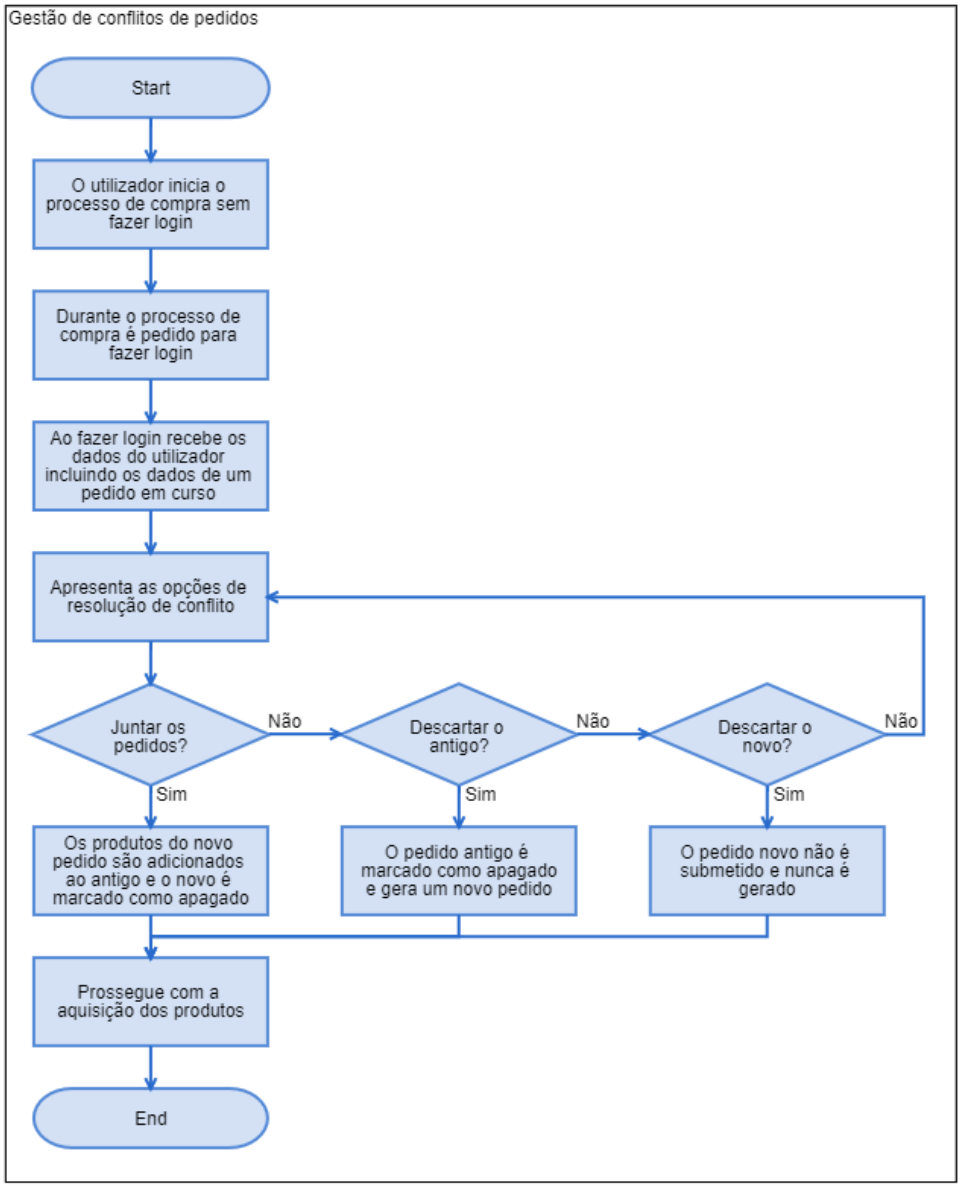


Figura 40 - Gestão de conflitos do carrinho de compras

3.4 Casos de Uso

Segundo Barros [52], "o objetivo dos casos de uso é a identificação das funcionalidades requeridas para o sistema. Assim, os casos de uso incluem-se na fase de análise de requisitos, análise de requisitos a fase em que procuramos identificar, da melhor forma possível, o que é que o nosso sistema (frequentemente um ou mais programas) deve realmente ser capaz de fazer".

Os casos de uso são compostos por atores, que são os "bonecos" que representam o utilizador ou o sistema, os casos de uso que são as funcionalidades do sistema representados pelas elipses e os relacionamentos que são as ligações entre atores e os casos de uso.

Com a criação dos casos de uso, é possível melhorar a robustez da aplicação, uma vez que podem ser implementados caminhos alternativos para um determinado caso de uso. Com recurso a estes, é também mais fácil descrever os requisitos e dar um contexto, uma vez que é feito sob a forma de um cenário ou história. Os casos de uso estão relacionados com a interação da aplicação, o que permite aos designers identificarem todas as ações que precisam criar.

Através da análise dos requisitos e dos processos, foi possível chegar aos casos de uso definidos abaixo.

3.4.1 Acesso à zona privada

No diagrama abaixo, temos o caso de uso do acesso à zona privada. Este começa pela validação do email a ser registado que seguidamente passa para o restante processo de registo e por consequente para a zona de login, que após validação das credenciais por parte da TX (tal como enunciado no ponto 3.8 é a infraestrutura de chaves públicas utilizada neste projeto), é concedido o acesso à zona privada. O outro caso de uso, é a recuperação da password por parte do utilizador, que requer a posterior validação dos dados por parte dos administradores de registo, para assim ser possível o login e consequentemente o acesso à zona privada.

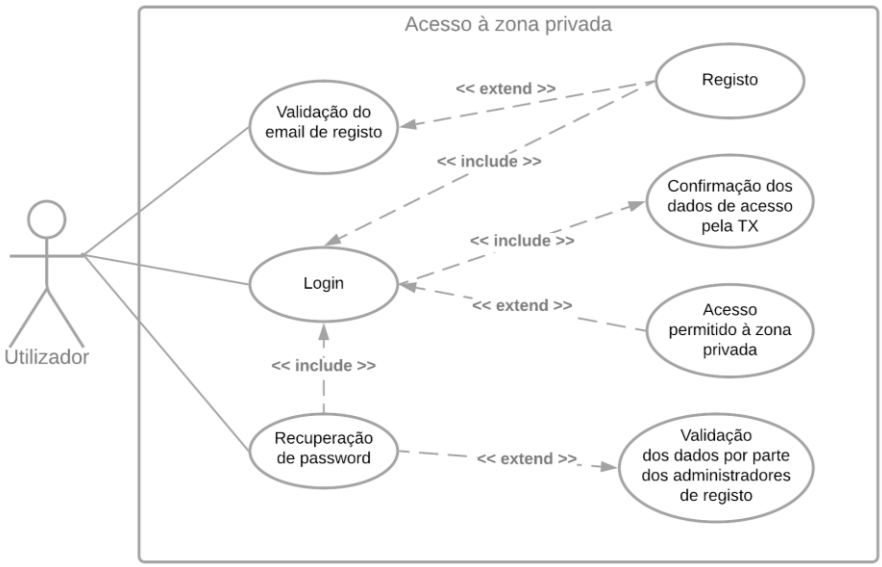


Figura 41 - Caso de uso do acesso à zona privada

3.4.2 Processo de compra

No esquema seguinte, é descrito o caso de utilização do processo de compra de um novo produto. No mesmo, é possível observar o carrinho de compras que contém os produtos a serem adquiridos, o preenchimento do formulário do processo de compra com os dados do cliente e a finalização do pedido. Seguidamente temos os que estendem dos anteriores, tal como a adição/remoção dos produtos do carrinho, que inclui a alteração de quantidades e a seleção do meio de pagamento.

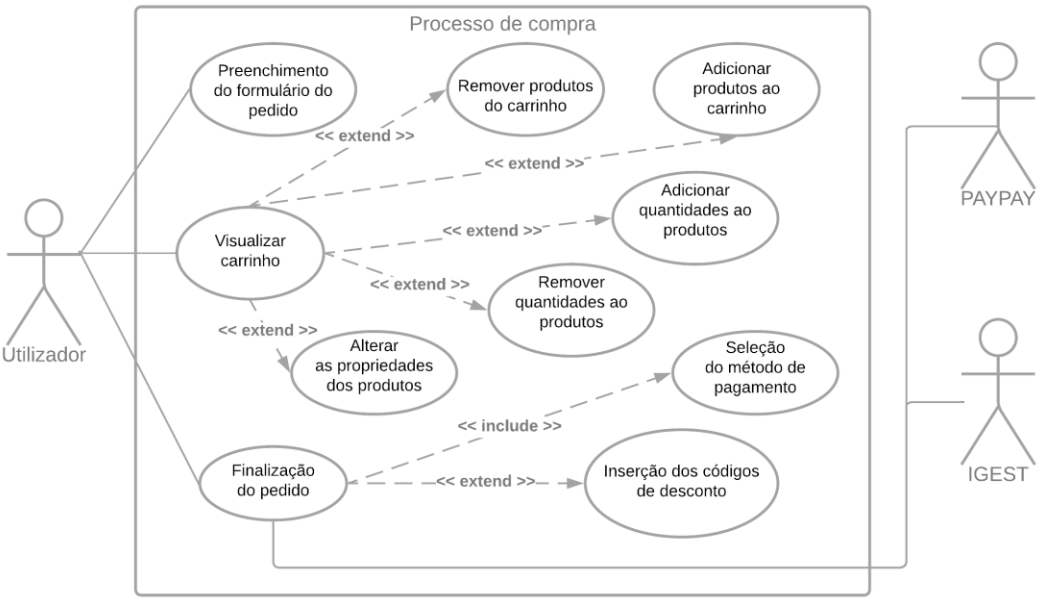


Figura 42 - Caso de uso do processo de compra

3.5 Prototipagem

Os *mockups* são um esboço de uma possível interface do utilizador da aplicação que queremos implementar. Estes permitem representar os componentes da interface e a navegação nas diferentes páginas da aplicação.

A grande vantagem dos *mockups* é a facilidade com que é possível simular a interface gráfica do produto final, antes de ser feito qualquer desenvolvimento aplicacional. Assim, o cliente pode ter uma noção de como será a interação com a plataforma.

Com recurso aos casos de uso anteriormente elaborados e após uma análise da antiga plataforma por parte da equipa de desenvolvimento, foi possível verificar as áreas que poderiam ser melhoradas. Como resultado foram elaborados os *mockups* seguidamente apresentados.

Após a criação dos mesmos, estes foram entregues à equipa de design para conceber os protótipos de alta-fidelidade. Estes são já uma representação próxima do resultado final pretendido. Com recurso a diversas ferramentas, como por exemplo o caso do Adobe XD, é

possível simular o fluxo completo de todas as funcionalidades pretendidas, de modo a ser possível testar com utilizadores, como se fosse o produto final.

Seguidamente será apresentado um exemplo do *mockup* elaborado, assim como o protótipo de alta-fidelidade correspondente, os restantes podem ser consultados no ponto 9.2. Cada *mockup* irá possuir um identificador (ID), de modo a ser facilmente identificado no *Dialog Map* presente no ponto 3.6.

3.5.1 Página inicial

Na página inicial (ID I), iremos ter presente um *carousel* com os produtos comercializados pela plataforma, sendo eles os selos eletrónicos, as assinaturas digitais, os selos temporais e os certificados SSL. Cada produto do *carousel* irá possuir dois botões com duas ações distintas, onde o primeiro botão será o redireccionamento para a página informativa do produto e o para a página de compra do mesmo.

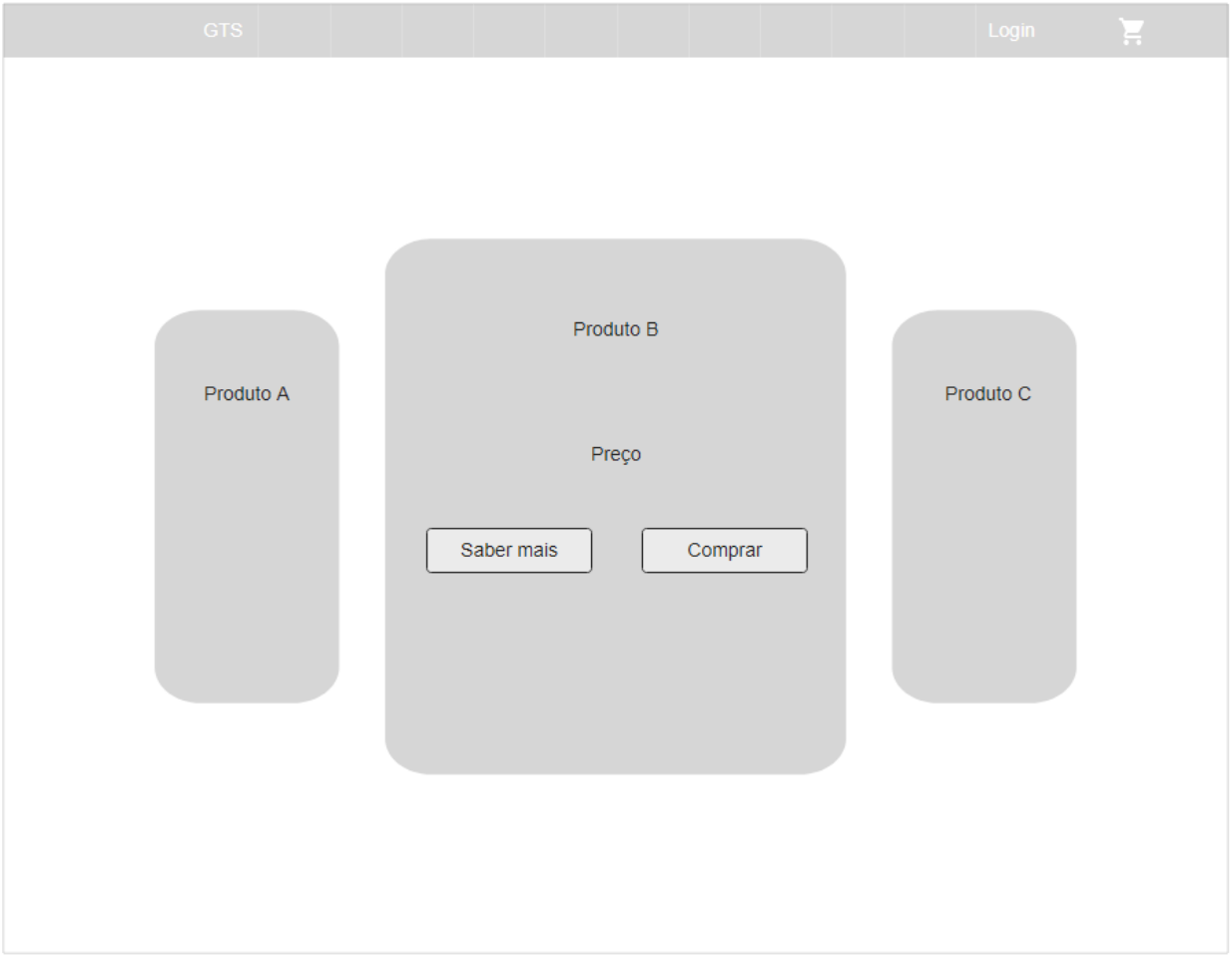


Figura 43 - Mockup da página inicial

Cada componente do *carousel* irá conter a seguinte estrutura:

1. Cor de fundo;
2. Cor de fundo2 (cor mostrada ao ser selecionado);
3. Link da imagem (mostrada ao ser selecionado);
4. Título;
5. Subtítulo;
6. Título2 (texto mostrado ao ser selecionado);
7. Subtítulo2 (texto mostrado ao ser selecionado);
8. Valor;
9. Texto do valor;
10. Link para informações do produto;
11. Link para comprar produto;

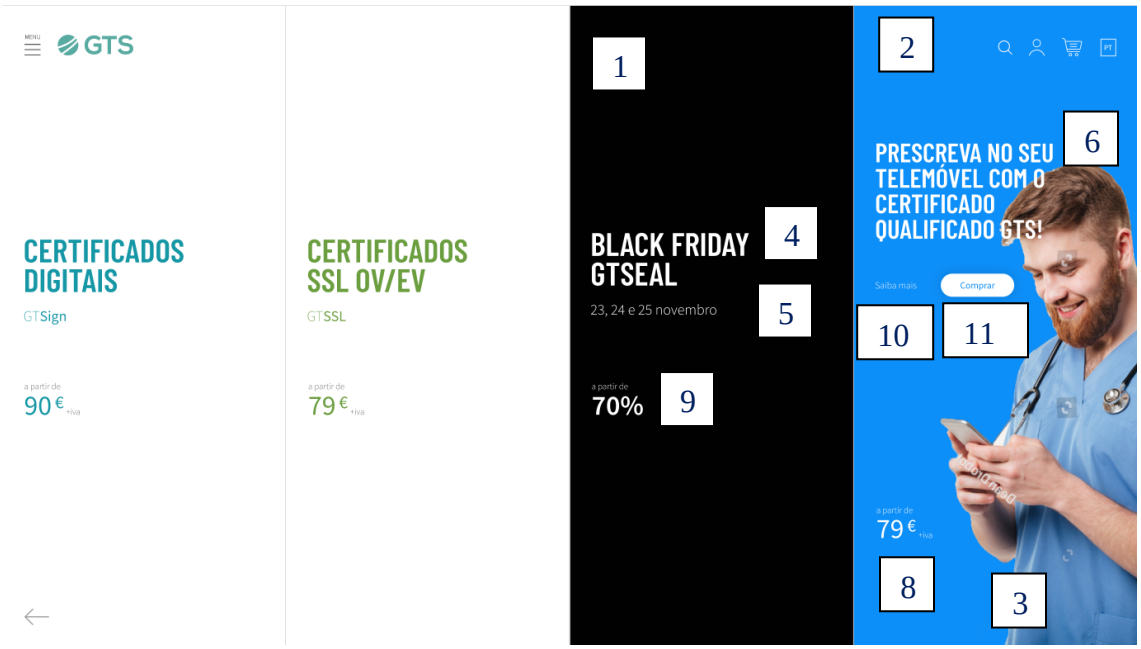


Figura 44 - Protótipo da página inicial

3.6 Mapa de diálogo

O mapa de diálogo ou "*dialog map*", simboliza uma interface abstrata do utilizador, sendo que cada opção é representada por uma ligação que muda o estado da interface. Este mapa é semelhante a um diagrama de transição de estados, onde cada ligação representa a possível navegação, sendo que as ações seguintes dependem sempre da ação atual. Em aplicações complexas irá existir uma grande quantidade de "caminhos", contudo será sempre um número finito de possibilidades, onde estas serão todas conhecidas.

Para construir um mapa de diálogo, devemos pensar nas páginas que a aplicação deve possuir, assim como a navegação entre as mesmas. Cada transição pode ser através do uso de uma tecla, um *click* ou um gesto em um dispositivo *touchscreen*.

A vantagem de utilizar estes mapas, é que podemos verificar todas a navegação e interação necessária, sem especificar ecrãs detalhadamente e sem grande esforço. Desta forma, é possível verificar se existe falta de interações, navegações incorretas ou desnecessárias, assim como analisar requisitos incorretos, em falta ou também desnecessários.

Para descomplicar um mapa de diálogo, podemos não mostrar interações gerais, como por exemplo um menu de ajuda que esteja sempre presente ou outro link qualquer. Assim, pelo facto de a barra de navegação conter uma ligação à página principal, ao carrinho de compras, à página de login (ID II) e ao menu de alteração de idioma, as ligações para estes quatro elementos não serão representadas no mapa de diálogo, uma vez que estão presentes em todas as páginas.

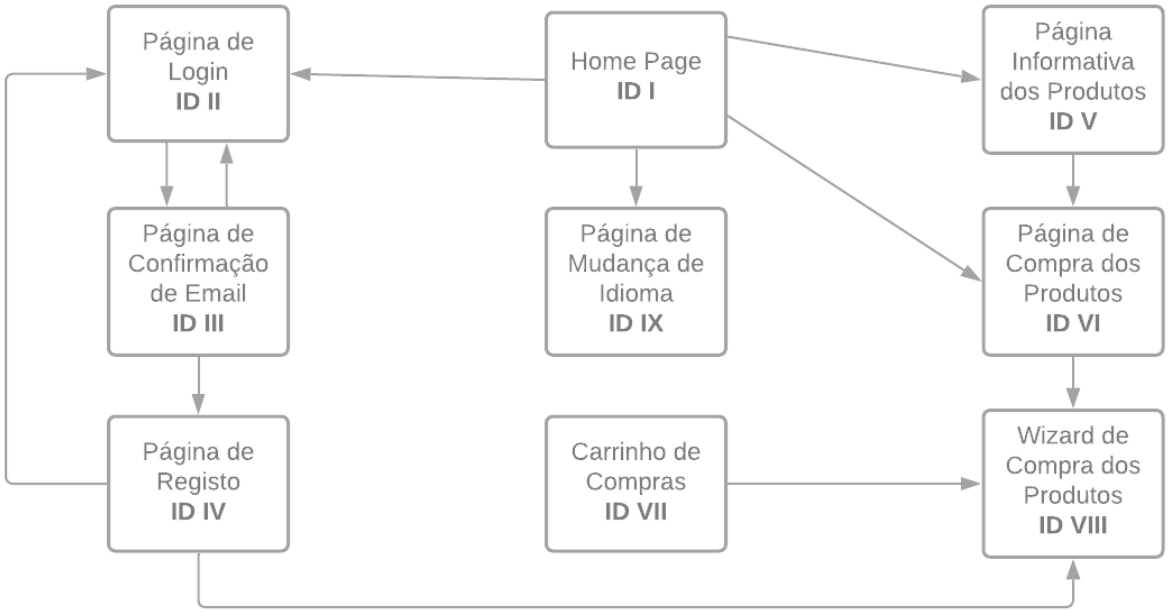


Figura 45 - Mapa de Diálogo

3.7 Conclusão

Após a revisão de literatura, neste capítulo foi enunciado a metodologia que a empresa ACIN segue em todos os seus projetos, sendo ela o *Scrum*. Foram também enumerados os requisitos necessários, após diversas reuniões com o *product owner* e todas as partes envolvidas neste projeto, nomeadamente os departamentos da empresa.

Neste capítulo foi também modelado a fluxo de comunicação entre o *front-end* e o *back-end*, assim como o fluxo de autenticação e de aquisição de produtos. Foi também previsto o

fluxo da gestão de conflitos que pode ocorrer caso sejam adicionados produtos ao carrinho sem que o utilizador esteja autenticado e que o mesmo já possua produtos no carrinho da sua conta, sendo neste caso possível juntar os produtos ou descartar, tanto os atuais como os guardados na conta.

Por fim, foram elaborados todos os passos necessários para a elaboração dos protótipos, onde com os mesmos é possível ter noção de como será o resultado final da aplicação, tanto a nível visual como a nível de navegação.

4. Implementação

Tendo em conta a modelação da solução apresentada no ponto 3, onde foram enumerados os requisitos funcionais e não funcionais, os fluxos e também os protótipos, neste capítulo iremos demonstrar como foi realizada a implementação do mesmo, seguindo o método da engenharia de software selecionado anteriormente.

Neste capítulo iremos demonstrar o processo de comunicação e autenticação com a API, o processo de compra e pagamento, e por fim a geração de um certificado.

4.1 Implementação de segurança na plataforma

Dentro da empresa do grupo ACIN, existem dois departamentos que se completam na responsabilidade de gerir todos os equipamentos da organização e que trabalham em conjunto de modo a garantir que as soluções implementadas são compatíveis com as necessidades da organização e com as melhores práticas do setor. Enquanto um dos departamentos está responsável pela instalação, configuração, manutenção e atualização de hardware e software, o outro departamento, o GAS (gabinete de administração de sistemas), é responsável por toda a gestão da rede da empresa. Os administradores de sistemas também são responsáveis por garantir a segurança dos sistemas de TI, protegendo-os contra ameaças e gerindo as possíveis falhas do sistema, de forma a estar sempre disponível e seguro.

De forma a manter a rede segura, os administradores de sistemas são responsáveis por configurarem as *firewalls* de modo a filtrar o acesso aos dados e controlar quem pode aceder aos sistemas, monitorizando todo o tráfego da rede, evitando acessos sem permissão e por consequentemente o alojamento de *malware* no servidor. Configuram também *VPNs* e redes privadas que permitem ter acesso seguro a servidores de forma remota. Este recurso permite que os funcionários que não estão na empresa, consigam visualizar os dados e iniciar sessão nos sistemas da empresa de uma forma protegida, como se estivessem em uma rede privada.

Uma das formas de garantir que não existe perda de dados em caso de falhas ou ameaças, é a realização regular e automática dos servidores da empresa. São também realizadas frequentemente as auditorias que avaliam os processos e as políticas, de modo a identificar falhas e proporcionar melhores resultados. Esta é uma forma de analisar o que está sendo feito e como a proteção dos sistemas está ocorrendo.

Os administradores de sistemas são também responsáveis pela gestão de acessos e permissões de todos os colaboradores da empresa. Estes dados, assim como as configurações de acesso aos servidores, são encriptadas, de modo a garantir o máximo sigilo e segurança. Os servidores web possuem também certificados *SSL*, e são acedidos remotamente por *FTP* com passwords robustas e que existem a troca das mesmas regularmente. Além do protocolo *FTP*, o acesso aos servidores, pode também ser feito por *SSH* (*Secure Shell*), de modo a ser feito em segurança. Qualquer acesso ou tentativa do mesmo, é registado em *logs*, de forma que seja sempre possível verificar se ocorreu alguma falha ou tentativa de ataque dentro da rede da empresa.

De modo a melhorar a velocidade e disponibilidade do servidor, os administradores de sistema utilização a replicação de servidores e o *load balancing* (ou balanceamento de carga). O *load balancing* é uma técnica que distribui o tráfego de rede entre vários servidores para melhorar a eficiência, desempenho e disponibilidade do sistema. Pode ser implementado de várias maneiras,

sejam elas a geolocalização, onde os utilizadores são direcionados para o servidor mais próximo geograficamente, baseada na carga de cada servidor ou distribuir à vez para cada servidor. As vantagens incluem um melhor desempenho e eficiência, assim como o aumento da disponibilidade, mas também há complexidade e custo associados à configuração e gestão de vários servidores.

Por outro lado, a replicação de servidores envolve a criação de cópias idênticas de um servidor em locais geográficos diferentes para garantir a disponibilidade do sistema mesmo em caso de falha de um servidor. Isso permite a recuperação rápida de dados e melhora o desempenho ao direcionar os utilizadores para o servidor mais próximo geograficamente. No entanto, também há custo e complexidade associados à aquisição e manutenção de várias cópias do servidor, além de possíveis problemas de sincronização de dados em sistemas ativos e com muitas atualizações.

4.2 Comunicação da aplicação com a API e autenticação

Tal como enunciado anteriormente no ponto 2.5.4, o *front-end* desta aplicação será implementado em ReactJS que irá comunicar com uma API implementada em Laravel. Esta comunicação irá ser feita com recurso a pedidos graphQL, como já falado no ponto 6.5.5. Por sua vez, os pedidos graphQL serão efetuados através da biblioteca Axios, referida no ponto 7.2 que tem como base o método *fetch* do *javascript*, contudo, permite adicionar mais configurações ao mesmo.

Para demonstrar como é realizado todo este procedimento, desde o tipo de pedido enviado, à resposta esperada, será apresentado abaixo o pedido feito à API para ser construído o *carousel* de produtos que é apresentado na página inicial.

Começamos por ter uma configuração padrão no Axios, onde é colocado o tempo máximo que o pedido vai aguardar por uma resposta, o URL ao qual vai fazer o pedido (neste caso o URL da API), o tipo de pedido (por exemplo *GET* ou *POST*) e os dados do pedido, neste caso a *query* do graphQL.

Na Figura 46 abaixo podemos verificar o *template* de um pedido com recurso à biblioteca Axios, com as configurações, gestão de resposta e gestão de erros.

```
export async function axiosRequest(givenQuery: string) {
    let result_request;
    const axios = axiosWithInterceptors();
    try {
        await axios({
            timeout: AppConstants.API_MAX_TIMEOUT,
            url: process.env.REACT_APP_API_GRAPHQL_SERVER_ENDPOINT,
            method: 'post',
            data: {
                query: givenQuery,
            },
        })
            .then((response) => {
                result_request = handleResponse(response);
            })
            .catch((error) => {
                return handleError(error);
            });
    } catch (e) {
        return handleError(e);
    }
    return result_request;
}
```

Figura 46 – Configuração do Axios

Com estas configurações, já temos o Axios pronto a realizar pedidos, consoante a *query* que for passada. Contudo, se realizar um pedido a uma área privada, será necessário verificar se o utilizador está autenticado, tal como mencionado no ponto 8.3.1. Para estar autenticado no graphQL, junto com o pedido, deve ser enviado um *header* de autorização, este não é nada mais, nada menos, que um *access token* que identifica que o utilizador está autenticado.

Uma vez que o *back-end* utiliza OAuth2 para o processo de autenticação, este *token* é guardado em uma *cookie* no *front-end* após o processo de login, assim como o *refresh token*, que é utilizado para gerar um novo *access token*, quando o *token* anterior deixa de estar válido. Resumindo, quando o utilizador efetua o login, caso as credenciais estejam corretas, este recebe o um *access token* e o *refresh token* (tal como é possível observar nos pontos 1 e 2 da Figura 47, onde *Authorization Server* simboliza a rota de autenticação), ambos com validades diferentes, sendo que a do *access token* é inferior.

Quando é então efetuado um pedido com o Axios, se este for um pedido privado (rotas privadas simbolizam o *Resource Server* na Figura 47), é verificado se existe a *cookie* com os *tokens* de acesso, caso exista, é então adicionado o *header* de autorização com o *access token*. Se o *access token* for válido, é recebida a resposta esperado ao pedido. Caso o *access token* seja inválido, ou seja, é recebida uma resposta de pedido não autorizado, é feito primeiro um novo pedido de renovação dos *tokens* de acesso através do *refresh token*. Se for bem-sucedido, são recebidos dois novos *tokens*, o *access token* e o *refresh token*, sendo estes novamente armazenados na *cookie*, e é novamente efetuado o pedido inicial, já com o novo *access token*.

Se por acaso o *refresh token* também for inválido, ou se não existir a *cookie* com os *tokens*, o utilizador é redirecionado para a página de login, onde se deve autenticar novamente para receber os *tokens* de acesso e estes serem armazenados na cookie. No caso dos pedidos públicos, não é necessário enviar nenhum *token* nos *headers* do pedido, e será sempre recebida uma resposta.

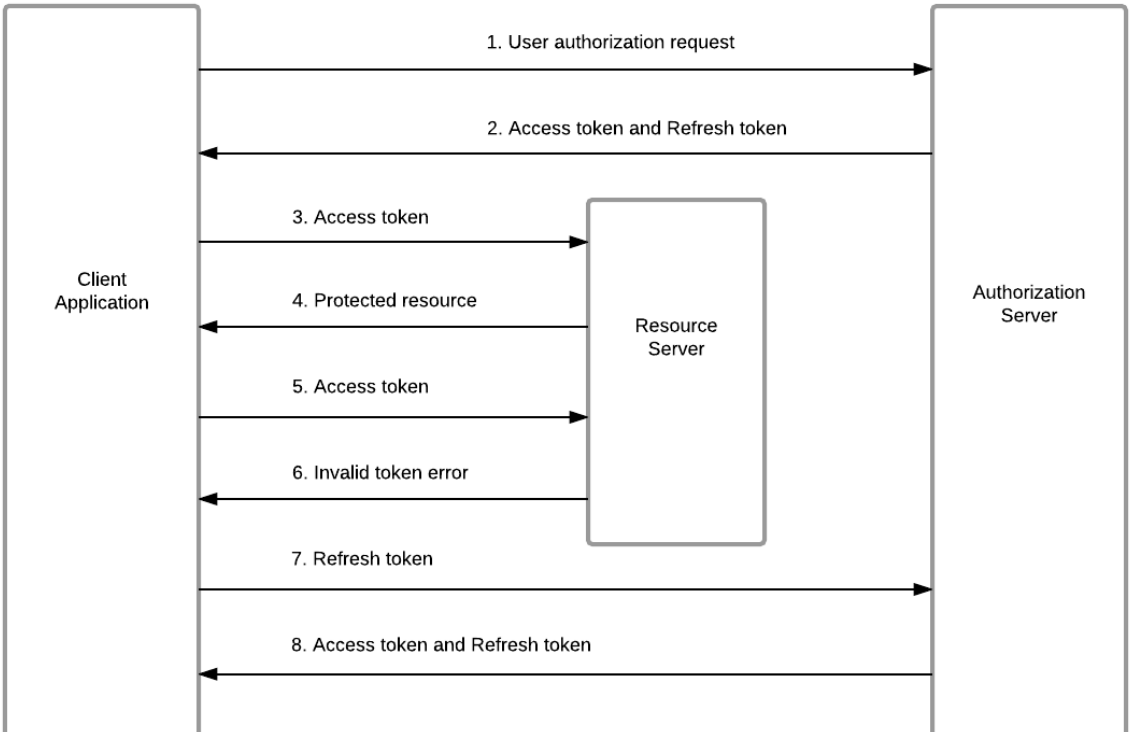


Figura 47 - Gestão de pedidos e tokens

Na Figura 48 seguinte, é possível verificar o pedido POST efetuado a um URL, neste caso o da API de pedidos graphQL, e este obteve um código de estado 200, ou seja, o pedido foi enviado corretamente e retornou uma resposta.

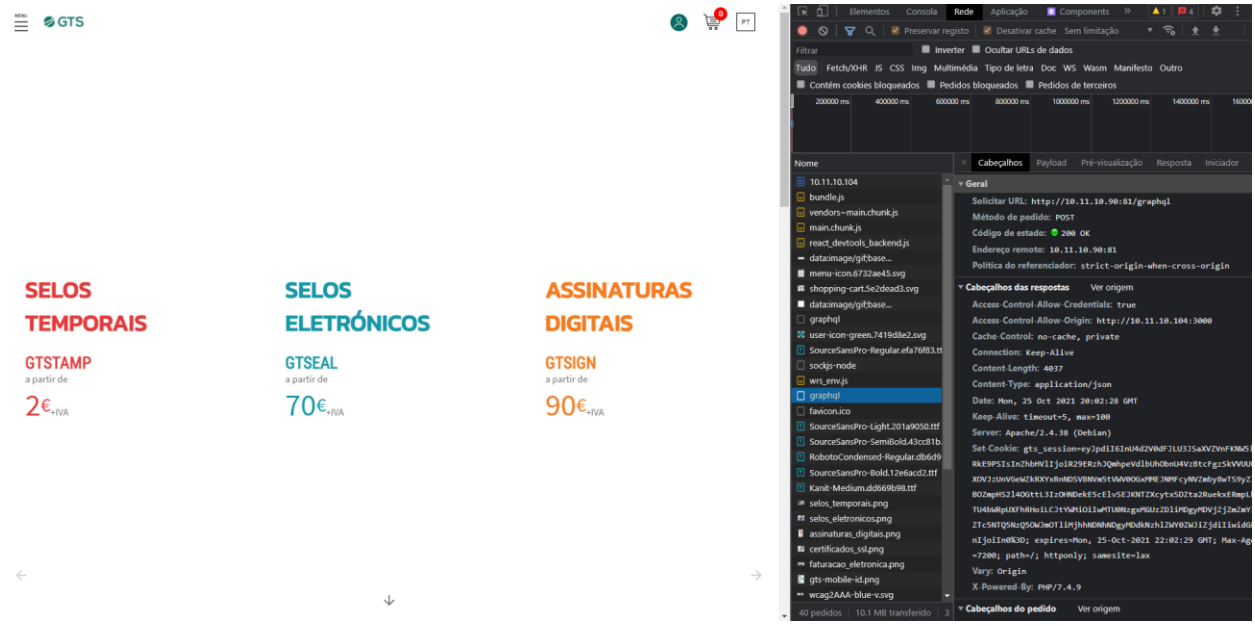


Figura 48 - Cabeçalho de um pedido

Na Figura 49 abaixo temos a *query* enviada para a API com os parâmetros de resposta pretendidos.

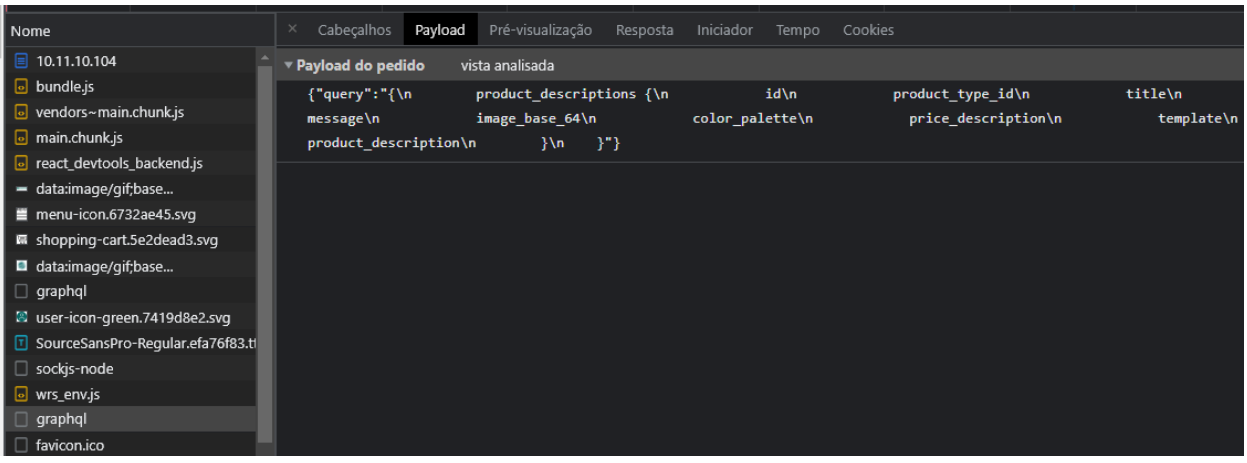


Figura 49 - Query de um pedido GraphQL

A Figura 50 seguinte mostra a resposta recebida, enviada pela API, onde podemos ver que recebemos o objeto producto_description, que recebe um *array* de cinco elementos, cada um com os parâmetros requisitados na *query*. Estes elementos são então os produtos que irão constar no *carousel* e nos parâmetros temos as informações necessárias para a restante construção do mesmo.

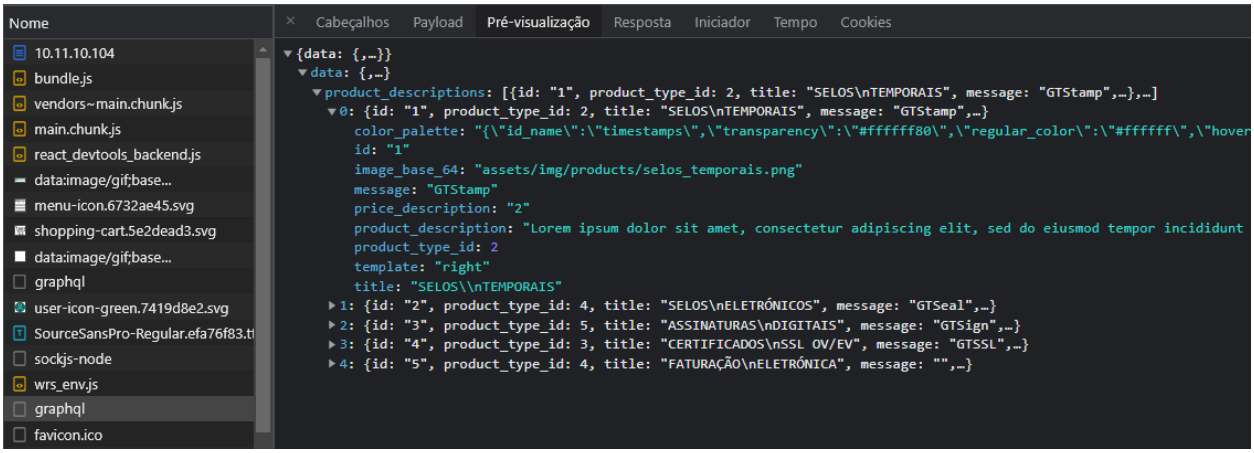


Figura 50 - Resposta de um pedido

Por sua vez, com recurso à aplicação *Firecamp* mencionada no ponto 2.5.5, é possível validar que os dados enviados na *query* da Figura 49, realmente correspondem à resposta esperada, tal como demonstrado na Figura 51 - Pedido graphQL no firecamp.

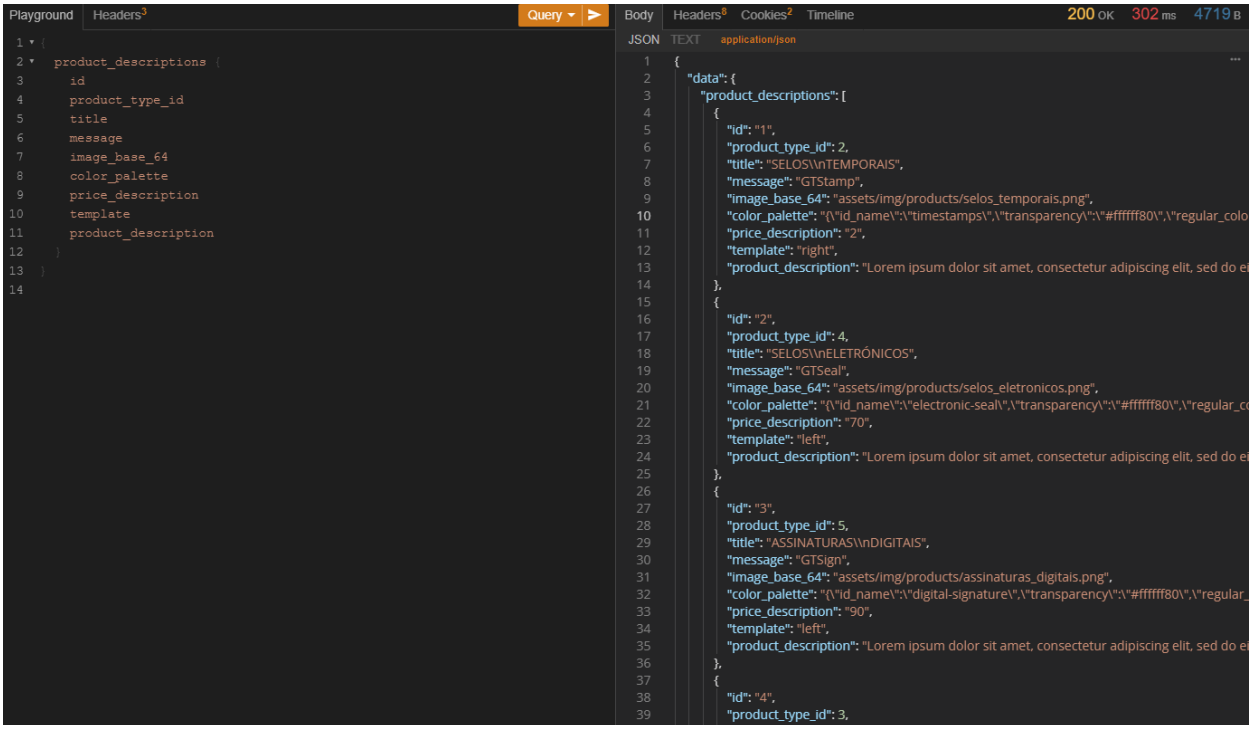


Figura 51 - Pedido graphQL no firecamp

4.3 Formulários de compra

Um dos problemas da atual plataforma, mencionados no ponto 1.5, foi o facto de a mesma possuir revendedores e estes terem de efetuar todo o processo no site da GTS. De forma a tentar melhorar este procedimento, foi decidido que a GTS poderia partilhar o seu formulário de compra diretamente com os revendedores, de forma que estes os pudessem colocar nos seus websites e partilhar na forma desejada. Estes formulários poderiam ser partilhados por *iframe* ou algo semelhante.

Desde modo, tanto para a partilha com os revendedores, como para a própria venda na nova plataforma, foi armazenado na base de dados um ficheiro JSON para cada produto, que possui a estrutura base a construção do formulário e que será posteriormente complementado com os dados dos produtos existentes na mesma base de dados.

Para a construção do formulário de compra, o *front-end* efetua uma *query* graqhQl, onde indica que o tipo de produto pretendido e recebe como resposta o JSON pretendido, semelhante à estrutura de dados abaixo representada na parte esquerda da figura seguinte. Com recurso à biblioteca React-jsonschema-form mencionada no ponto 2.6.5, o JSON anterior é enviado para a mesma como parâmetro e é automaticamente gerado o formulário representado na parte direita da Figura 52.

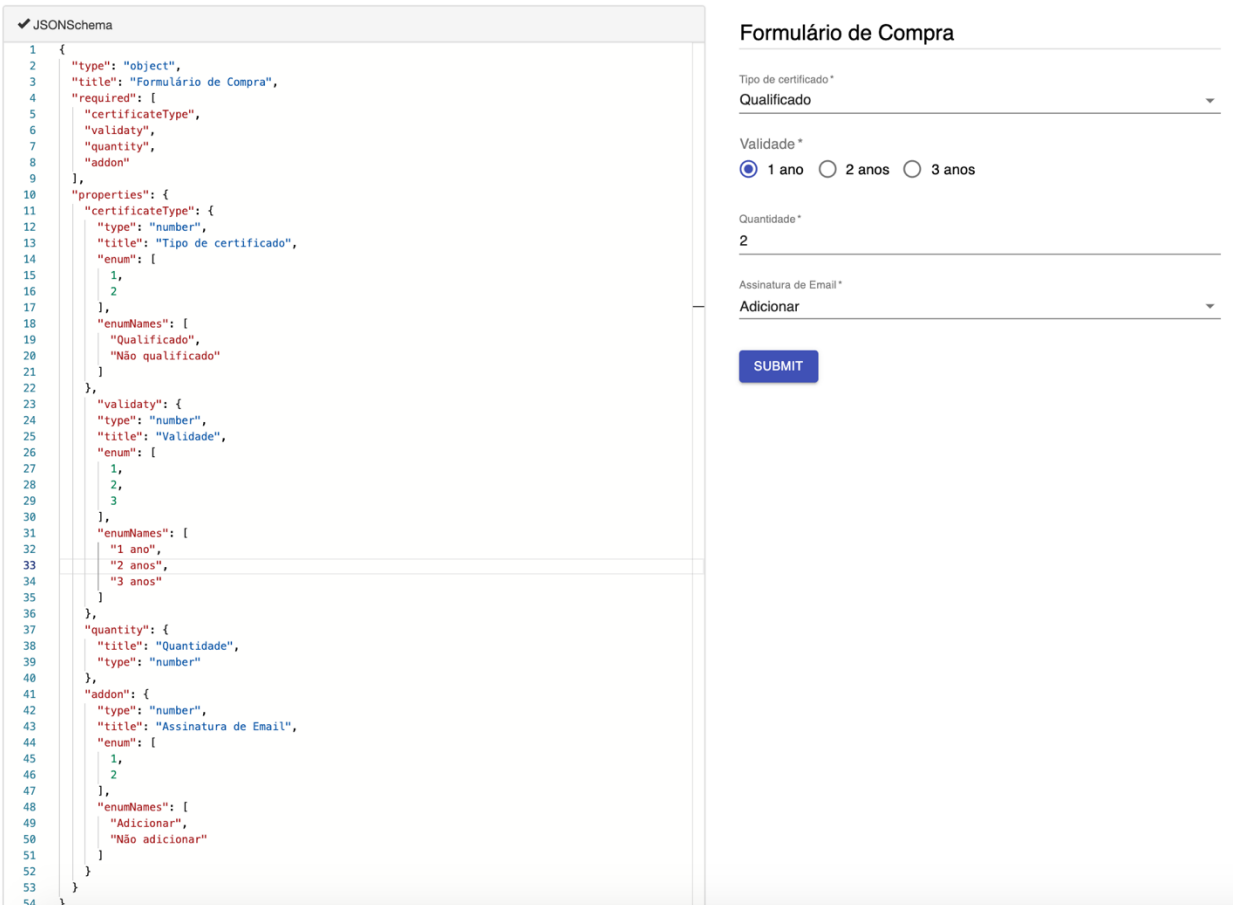


Figura 52 - Exemplo de um formulário gerado com objeto JSON

Desta forma, sempre o objeto JSON armazenado na base de dados, será sempre o ponto central da construção do formulário de compra da aplicação e dos revendedores. Sempre que for necessário modificar o formulário, apenas é necessário modificar o objeto JSON, a biblioteca irá gerar o novo formulário e não será necessário efetuar qualquer alteração no *front-end*, automatizando o formulário de compra.

Este método foi utilizado para implementar os formulários de compra mostrados nas figuras Figura 76, Figura 77, Figura 78 e Figura 79. Os restantes formulários do processo de compra e de autenticação, nomeadamente os dados de faturação, a customização dos produtos, etc, foram inicialmente implementados com recurso à biblioteca React Hook Form apresentada no ponto 2.6.4, contudo, foi posteriormente desenvolvido pela empresa ACIN uma biblioteca interna de formulários, de modo a uniformizar o aspeto dos mesmos em todos os projetos que a empresa está presente. Assim, foi necessário reformular todo o trabalho já desenvolvido na elaboração dos formulários, de modo a substituir a biblioteca React Hook Form pela biblioteca interna.

Tal como os formulários, também a gestão de notificações através de *popups* foi inicialmente implementada através da biblioteca React-toastify apresentada no ponto 2.6.9, sendo posteriormente substituída pela biblioteca interna de gestão de notificações, pelo mesmo motivo de manter o aspeto visual em todas as plataformas do grupo ACIN.

4.4 Carrinho de compras

O módulo do carrinho de compras é o grande destaque da remodelação da plataforma. Desde modo, foi planeado que o mesmo deveria ser simples, mas com funcionalidades que fossem úteis para o utilizador. Assim, foi implementado um carrinho de compras que permitisse ao utilizador adicionar itens ao mesmo, estando ou não autenticado, assim como alterar o seu conteúdo e quantidades, verificar o valor de cada produto e o total do carrinho em qualquer parte da aplicação.

De modo a apresentar toda esta informação, o conteúdo do carrinho está estrutura em um objeto JSON que posteriormente é armazenado em uma cookie. Desta forma, mesmo que o utilizador não se autentique e adicione produtos ao carrinho de compras, este estará sempre disponível quando o utilizador voltar à aplicação.

O primeiro passo desta implementação, é ter sempre disponível a lista dos produtos, de modo a não ser necessário estar constantemente a fazer pedidos à API sobre informações dos mesmos. Quando o utilizador entra na aplicação, é efetuado um pedido de todos os produtos e as suas informações e é armazenado em sessão. Assim, sempre que forem necessárias informações do pedido, é primeiro verificado se existe a variável de sessão com os mesmos, se houver utiliza essa informação, caso contrário efetua um novo pedido e armazena novamente em sessão.

A criação do objeto do carrinho de compras, possuí quatro "chaves" principais: o identificador do pedido armazenado no carrinho de compras (sendo null enquanto não está autenticado e possuindo um valor após a autenticação e envio para a API), o número total de produtos contidos no carrinho, o valor total do carrinho e a lista de produtos que o mesmo contém. Esta lista possui todas as informações necessárias para a construção do carrinho, desde o nome do produto, o valor, o identificador na base de dados, os "*addons*" que possui, etc. Um exemplo da estrutura deste objeto, pode ser visualizado na Figura 52.

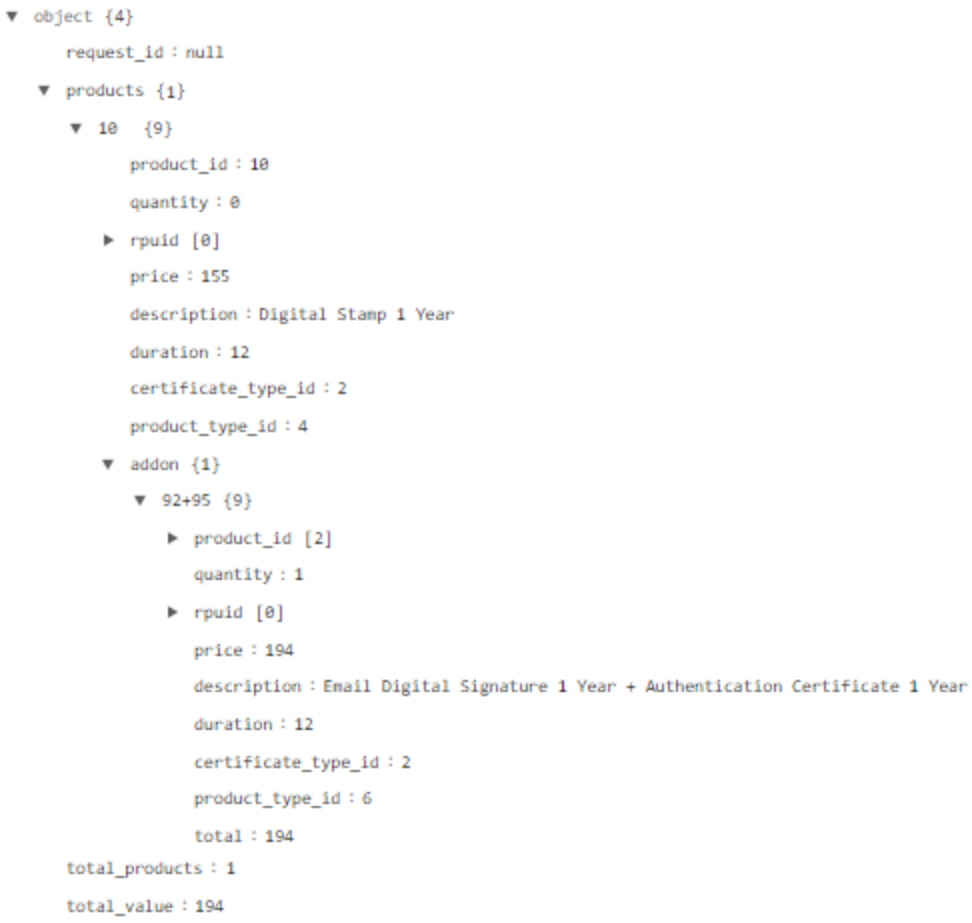


Figura 53 - Estrutura do objeto do carrinho de compras

Tal como referido no ponto 3.3.4, no caso em que um utilizador adicione produtos ao carrinho sem estar autenticado e que já tenha adicionado anteriormente outros produtos ao carrinho estando autenticado, irá existir conflitos no carrinho de compras. Nesta situação irá ser apresentado ao utilizador uma lista dos produtos adicionados anteriormente, assim como dos atuais, e solicitado que escolha se quer manter os produtos antigos, os produtos atuais ou se pretender juntar ambos.

Se o utilizador selecionar o caso onde pretende manter os antigos ou juntar ambos, o objeto armazenado na cookie do carrinho de compras, irá ser substituído pelo novo objeto, de acordo com a opção selecionada. Caso pretenda manter os atuais, os dados já correspondem aos existentes na cookie e não será necessário efetuar qualquer alteração.

4.5 Pagamento

O pagamento dos produtos adquiridos na plataforma implementada, poderia ser efetuado por quatro formas diferentes, tal como ilustra a Figura 54 abaixo: MBWay, Visa, Multibanco e Transferência bancária.

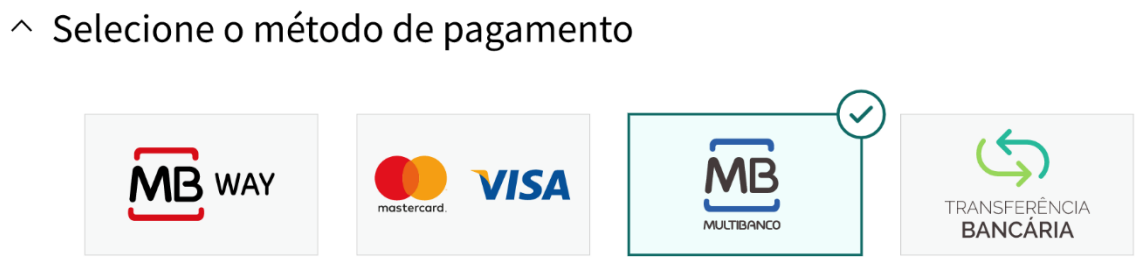


Figura 54 - Meios de pagamento

Caso seja selecionado a opção MBWay ou Visa, essa informação é enviada para o *back-end*, que retorna como resposta o URL para o qual o utilizador deve ser redirecionado, de modo a efetuar o pagamento, na plataforma Paypay que pertence também ao grupo ACIN. Após o redireccionamento, o sistema ficará a aguardar a resposta enviado pelo Paypay, de erro ou sucesso. Se correr tudo bem e a resposta for afirmativa, o utilizador passa ao próximo passo, caso contrário é avisado que algo correu mal e que deve efetuar o pagamento novamente.

Se for selecionado a opção multibanco, são enviados para o utilizador os dados para os quais deve efetuar o pagamento (entidade e referência multibanco). Por último, se escolher transferência bancária, será apresentado o NIB para o qual o utilizador deve efetuar a transferência e enviar por email o comprovativo da mesma.

4.6 Gerar e descarregar certificado

Após a conclusão do processo de compra de um certificado, assim como da validação dos dados do mesmo pelas Autoridades de Registo, como referido no ponto 3.4, o utilizador já pode efetuar a geração do seu certificado.

De maneira a aumentar a segurança, para efetuar este procedimento, o utilizador tem de estar autenticado na plataforma. Depois da autenticação, ainda será necessário um segundo passo de autenticação, através do denominado OTP (*One Time Password*). Este não é mais que um código "extra" que o utilizador recebe no seu email, de modo a confirmar que o certificado é gerado pelo seu proprietário.

Assim, caso o código esteja correto, é enviada uma notificação para o *back-end*, informando que o certificado pode ser gerado o certificado. O *back-end* ao receber esta informação, verifica no pedido de compra os dados necessários, tais como o nome, o país, o email, o número de identificação, etc. Como exemplo, temos o recente caso dos certificados de COVID que para serem gerados é necessário o país, o nome completo e a data de nascimento.

Estes dados são enviados para o PKI que inicialmente gera um certificado CSR (Certificate Signing Request) com uma estrutura semelhante à imagem seguinte, que irá definir as "instruções" para a criação do certificado final. Consequente o CSR é assinado pela chave privada, tornando-se assim um certificado válido para assinaturas digitais. [53]

Figura 55 - Exemplo de um ficheiro CSR

Após este procedimento é gerada a componente pública do certificado, que será a informação exportada do PKI, tendo um formato semelhante à imagem seguinte.

Figura 56 - Exemplo de um certificado

O PKI responde então para o *back-end* com a componente pública que é posteriormente armazenada na base de dados, ficando assim disponível para o utilizador efetuar o download

quando assim o entender. A componente privada do certificado gerado anteriormente será utilizada para a assinatura digital de documentos por parte do utilizador, e nunca será acessível fora do PKI. Se o *back-end* tiver a necessidade de ler o conteúdo do certificado público, irá precisar de usar a biblioteca do X.509 de modo a conseguir interpretar o seu conteúdo e extrair os dados nele contidos.

Por fim, o *back-end* envia a resposta para o *front-end*, informando se a geração do certificado ocorreu com sucesso ou não. Após este procedimento, o *front-end* efetua um novo pedido de modo a obter os dados mais recentes dos certificados do utilizador, de modo a atualizar o conteúdo mostrado ao cliente.

Caso o certificado seja gerado com sucesso, o cliente irá ter então a possibilidade de descarregar a componente pública, visualizar a data de validade do certificado e também a opção de revogação do certificado, caso o mesmo possua algum dado inválido, se o utilizador suspeitar que alguém possua acesso indevido ao mesmo, etc.

4.7 Conclusão

Neste capítulo foram abordados os componentes principais implementados na aplicação. O ponto inicial e mais importante foi averiguar a melhor forma de gerir a comunicação entre o *front-end* e o *back-end*. Este processo foi então realizado com recurso ao Axios, OAuth e graphQL, de modo a tornar seguras as comunicações.

Com recurso à biblioteca React-jsonschema-form, foram contruídos os formulários de compra dos produtos assim como os dos revendedores. Já os pagamentos, passam sempre de alguma forma pelo sistema do Paypay, que posteriormente comunica ao *back-end* o resultado do procedimento.

Por último, foi também implementado os pedidos de gerar e descarregar os certificados que o utilizador possui, sendo os mesmos solicitados ao *back-end*, que por sua vez efetua o pedido dos mesmos ao PKI, sendo que o PKI da empresa ACIN, é também denominado por TX, que foi já referido anteriormente.

De notar que para todas estas componentes foram construídos testes automatizados, de forma a garantir que com o decorrer da implementação dos requisitos solicitados, o sistema tivesse em todos os momentos o comportamento esperado.

5. Avaliação

Nos anos de 1960 e 1970 apareceram os testes de software, contudo, apenas nos anos 80 é que passaram a ser um processo formal, e atualmente são de extrema importância. Os testes de software têm como objetivo diminuir o risco de algo falhar. [54]

Quando compramos um produto, assumimos que este foi rigorosamente submetido a diversos testes de qualidade e resistência, de modo a ser algo que dure muito tempo e tenha qualidade. O mesmo ocorre com o desenvolvimento de software, onde o utilizador final espera que o fluxo de dados ocorra normalmente sem que qualquer erro ocorra até o fim do processo. Ao elaborar um teste de software, deve ser tido em conta os requisitos propostos pelo cliente, de modo a garantir que o resultado final é o pretendido.

Neste capítulo iremos abordar os testes automatizados realizados, assim como os testes manuais realizados pelos *beta testers*, tal como mencionado no ponto 2.4.3.

5.1 Testes unitários e de integração

Na construção de aplicações complexas, é necessário a separação das componentes, de modo a evitar dependências e para garantir que essa componente efetua o esperado, sendo que caso ocorra algum erro, o mesmo possa ser corrigido facilmente. Os testes unitários têm como objetivo testar uma pequena parte do código, antes do mesmo ser integrado com a restante aplicação. [55]

Os testes unitários têm várias vantagens como ajudar na fácil perceção e manutenção do código e documentar o mesmo. Outra vantagem, é garantir que em todas as adições de novas funcionalidades, as funcionalidades anteriores tenham o mesmo comportamento. Caso os testes anteriores falhem, pode ter ocorrido algum erro na nova funcionalidade que possa ter criado um conflito com as anteriores, ou uma alteração que não estava prevista nos testes anteriores. Algumas das desvantagens são a manutenção dos testes já desenvolvidos e a complexidade que pode trazer no *refactoring* do código.

Já os testes de integração, é a junção de dois ou mais testes unitários, de modo a comprovar que os mesmos são executados sem problemas, tal como nos testes unitários. Podemos comparar com um puzzle, onde a juntar várias vezes temos de verificar que as mesmas coincidem, para no fim tudo bater certo.

Para a realização dos testes, foram utilizadas as ferramentas já enunciadas nos pontos 2.6.15. De modo a garantir que a aplicação execute as funções necessárias sem ocorrer nenhum erro, foram criados testes unitários e de integração para os componentes principais e mais utlizados. Entre estes, podemos destacar então os componentes do menu principal, a barra de navegação, o menu de idioma, a componente de notificações, o carrinho de compras, o *carousel* da página principal, e o formulário de compra dos produtos.

A elaboração dos testes consistiu primeiramente em verificar se os componentes eram compilados corretamente, efetuando um *snapshot* dos mesmos, tal como exemplifica a Figura 57. Outro tipo de teste, consistiu em simular os toques/ações que o utilizador poderia despoletar e verificar se o comportamento seria o esperado, tal como mostra a Figura 58. Também foram efetuados testes para verificar se o conteúdo compilado era o esperado, por exemplo no

componente do menu, de modo a garantir que os links para as diversas páginas estão corretos, como por exemplo a Figura 59.

```
it('renders component without crashing', () => {
    var component = (
        <MainCarousel
            products={products}
            t={t}
            updateHover={updateHover}
            setCurrentSlide={setCurrentSlide}
            ButtonGroup={ButtonGroup}
            backgroundColourState={backgroundColourState}
            primaryProductColourState={primaryProductColourState}
            secundaryProductColourState={secundaryProductColourState}
        />
    );

    const document = render(component, dom_container);
    expect.hasAssertions();
    expect(document).toMatchSnapshot();
});
```

Figura 57 – Teste unitário snapshot

```
it('digital signature form non-qualified 3 years without signature', () => {
    render(component, dom_container);

    const radio_certificate_type = screen.getByLabelText('Não-qualificado');
    fireEvent.click(radio_certificate_type);

    const radio_validaty = screen.getByLabelText('3 anos');
    fireEvent.click(radio_validaty);

    const radio_signature = screen.getByLabelText('Não adicionar');
    fireEvent.click(radio_signature);

    fireEvent.click(screen.getByText('Submit'));

    expect.hasAssertions();
    expect(screen.getByTestId('FormResult')).toHaveTextContent('certificate_type=3&validity_type=36&typology_type=2&email_signature=false');
    submit_div.remove();
});
```

Figura 58 - Teste unitário de ações

```
it('get correct sub menus links', () => {
    var json = require('../../translations/pt/main_menu');
    render(component, dom_container);
    expect.hasAssertions();

    expect(screen.getByTestId('menu-news')).toBeDefined();
    expect(screen.getByTestId('menu-news').getAttribute('href')).toBe('/' + json.url.news);

    expect(screen.getByTestId('menu-simulator')).toBeDefined();
    expect(screen.getByTestId('menu-simulator').getAttribute('href')).toBe('/' + json.url.simulator);

    expect(screen.getByTestId('menu-onlineShop')).toBeDefined();
    expect(screen.getByTestId('menu-onlineShop').getAttribute('href')).toBe('/' + json.url.onlineShop);

    expect(screen.getByTestId('menu-timestamps')).toBeDefined();
    expect(screen.getByTestId('menu-timestamps').getAttribute('href')).toBe('/' + json.url.timestamps);

    expect(screen.getByTestId('menu-electronicSeal')).toBeDefined();
    expect(screen.getByTestId('menu-electronicSeal').getAttribute('href')).toBe('/' + json.url.electronicSeal);

    expect(screen.getByTestId('menu-digitalSignature')).toBeDefined();
    expect(screen.getByTestId('menu-digitalSignature').getAttribute('href')).toBe('/' + json.url.digitalSignature);

    expect(screen.getByTestId('menu-sslCertificates')).toBeDefined();
    expect(screen.getByTestId('menu-sslCertificates').getAttribute('href')).toBe('/' + json.url.sslCertificates);

    expect(screen.getByTestId('menu-registration')).toBeDefined();
    expect(screen.getByTestId('menu-registration').getAttribute('href')).toBe('/' + json.url.registration);

    expect(screen.getByTestId('menu-partners')).toBeDefined();
    expect(screen.getByTestId('menu-partners').getAttribute('href')).toBe('/' + json.url.partners);

    expect(screen.getByTestId('menu-about')).toBeDefined();
    expect(screen.getByTestId('menu-about').getAttribute('href')).toBe('/' + json.url.about);
});
```

Figura 59 - Teste unitário de asserções

Por fim, foram também efetuados os testes unitários e de integração, para garantir que os componentes, não só funcionam bem individualmente, como também em conjunto com os restantes. Estes foram feitos por exemplo, na integração do formulário de compra de produtos com a componente de resumo da compra, onde ambas devem estar sincronizadas de forma a mostrar a mesma informação. Na tabela abaixo é possível visualizar um exemplo dos testes elaborados, os restantes podem ser encontrados em 9.3.

Assinaturas Digitais	
✔	library renders correctly (174 ms)
✔	send correct default digital signature form values (56 ms)
✔	change quantity in digital signature form (88 ms)

✓	digital signature form - singular professional qualified 2 years with signature (52 ms)
✓	digital signature form non-qualified 3 years without signature (54 ms)
✓	digital signature resume with default mock store (65 ms)
✓	digital signatures resume with mock store without email signature, collective non-qualified 2 years, quantity 2 (15 ms)
✓	component renders correctly (110 ms)
✓	component integration digital signature form and resume (78 ms)
✓	component integration digital signatures form and resume, change store values (85 ms)
✓	get correct digital signatures products (382 ms)
Testes	**11 corretos**, 11 total
Snapshots	**2 escritos**, 2 total
Tempo	**15.275 s**

Tabela 8 – Testes às assinaturas digitais

5.2 Code Coverage

O *code coverage* é uma métrica que auxilia no cálculo da relação de testes efetuados vs código fonte do projeto de modo a prevenir que não ocorram bugs. [56] Esta métrica pode ser útil para analisar a qualidade dos testes implementados. As métricas mais comuns utilizadas no *code coverage* são a cobertura por função, por instrução, por condição, por linha, etc.

Obter uma boa cobertura do código não indica necessariamente que os testes foram bem realizados, uma vez que os mesmos devem ser robustos. Não existe o valor fixo a ser alcançado pelo *code coverage*, é geralmente aceite 80%. Tentar obrigar esta percentagem a ser próxima dos 100%, pode levar à realização de testes menos bons. É importante ir atualizando o *code coverage* no decorrer da implementação e da criação de novos testes.

Para a realização do *code coverage* deste projeto, foi também utilizada a ferramenta Jest, a qual gerou os resultados apresentados na Figura 60 abaixo. Nos componentes mais importantes o resultado foi sempre acima dos 80%, logo não houve uma necessidade de melhorar os restantes.

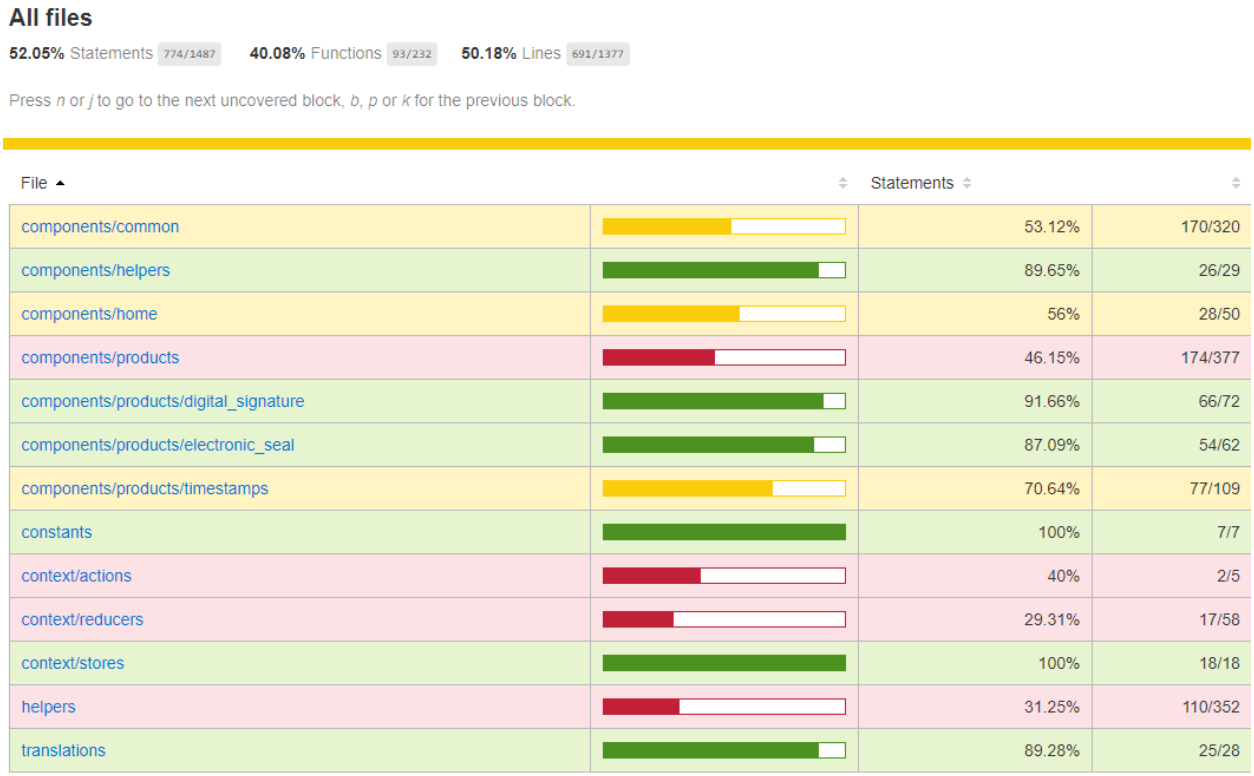
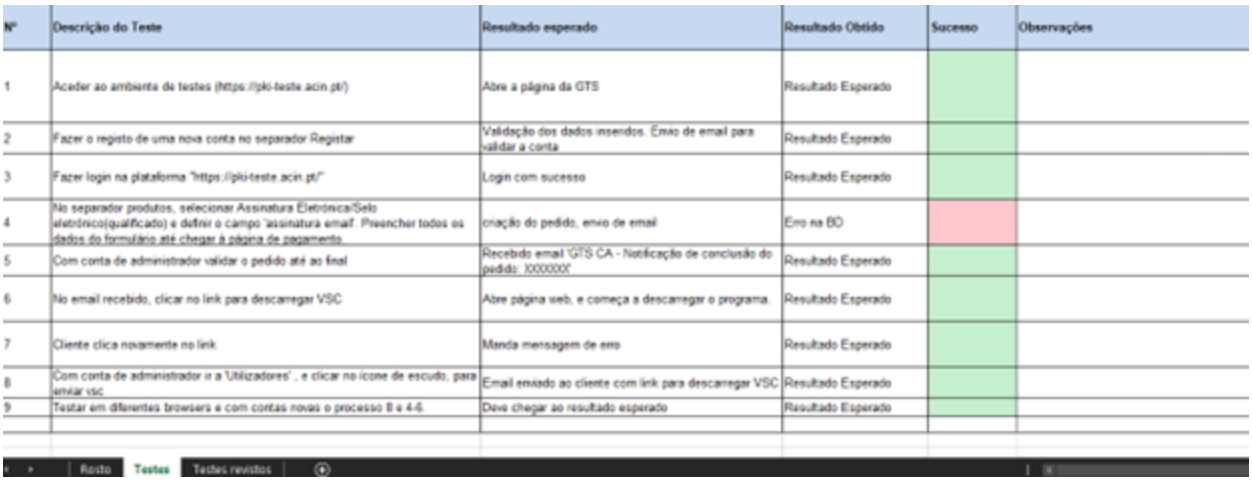


Figura 60 - Code Coverage

5.3 Testes Manuais

Os testes manuais são elaborados pelo programador quando termina a tarefa implementada. Posteriormente, o teste é realizado pelos *beta testers* da empresa, de modo a tentarem encontrar possíveis bugs. Os testes manuais podem não ser tão precisos como os testes automatizados devido ao erro humano, contudo podem encontrar erros não previstos nos testes automatizados, assim como são importantes para testar a usabilidade da aplicação. Nas figuras seguintes, é possível observar um dos testes manuais elaborados, assim como a sua avaliação pelos *beta testers.*

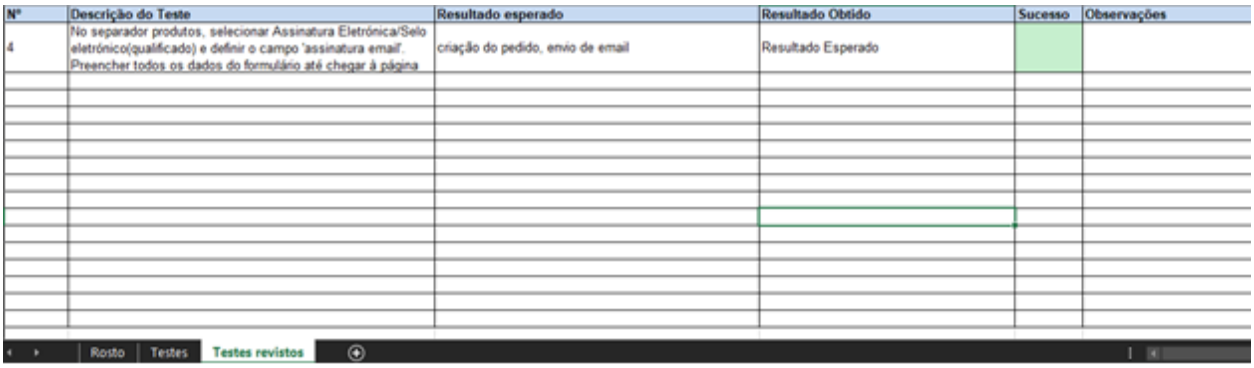
Na **Erro! A origem da referência não foi encontrada.** é possível observar o teste manual realizado pelos *beta tester*, onde a verde indica os testes com resultado esperado e a vermelho os testes onde foram encontrados erros. Já na **Erro! A origem da referência não foi encontrada.** temos a segunda execução dos testes após serem corrigidos e onde é possível observar que todos os testes obtiveram o resultado esperado.



Nº	Descrição do Teste	Resultado esperado	Resultado Obtido	Sucesso	Observações
1	Aceder ao ambiente de testes (https://pki-teste.acin.pt/)	Abre a página da GTS	Resultado Esperado		
2	Fazer o registo de uma nova conta no separador Registar	Validação dos dados inseridos. Envio de email para validar a conta	Resultado Esperado		
3	Fazer login na plataforma "https://pki-teste.acin.pt/"	Login com sucesso	Resultado Esperado		
4	No separador produtos, selecionar Assinatura Eletrónica/Selo eletrónico(qualificado) e definir o campo 'assinatura email'. Preencher todos os dados do formulário até chegar à página de pagamento	criação do pedido, envio de email	Erro na BD		
5	Com conta de administrador validar o pedido até ao final	Recebido email 'GTS CA - Notificação de conclusão do pedido: XXXXXXX'	Resultado Esperado		
6	No email recebido, clicar no link para descarregar VSC	Abre página web, e começa a descarregar o programa.	Resultado Esperado		
7	Cliente clica novamente no link	Manda mensagem de erro	Resultado Esperado		
8	Com conta de administrador ir a 'Utilizadores' , e clicar no ícone de escudo, para enviar vsc	Email enviado ao cliente com link para descarregar VSC	Resultado Esperado		
9	Testar em diferentes browsers e com contas novas o processo 8 e 4-6.	Deve chegar ao resultado esperado	Resultado Esperado		

Rosto | Testes | Testes revistos

Figura 61 - Testes manuais executados inicialmente



Nº	Descrição do Teste	Resultado esperado	Resultado Obtido	Sucesso	Observações
4	No separador produtos, selecionar Assinatura Eletrónica/Selo eletrónico(qualificado) e definir o campo 'assinatura email'. Preencher todos os dados do formulário até chegar à página	criação do pedido, envio de email	Resultado Esperado		

Rosto | Testes | Testes revistos

Figura 62 - Testes manuais corrigidos

5.4 Conclusão

Os testes unitários foram muito importantes na elaboração do projeto em questão, foi possível ver a sua importância quando por exemplo foram alterados dados de produtos na base de dados, e os testes dos produtos começaram a falhar, pois não estavam preparados para essa alteração. Isto apenas demonstra o valor que os testes unitários têm, e que os mesmos devem ser atualizados sempre que necessário, de modo que no final da adição de cada nova funcionalidade, todos os testes sejam executados sem erros.

6. Discussão

Nesta secção vamos discutir os resultados da implementação, tendo em conta a solução proposta na secção 3 e todas as decisões tomadas no desenvolvimento.

A discussão irá estar assente nos problemas encontrados, que deram início à elaboração deste projeto e na intenção de reformular a plataforma com tecnologias mais recentes. Será discutida a implementação do carrinho de compras e alguns motivos apontados para os atrasos e constrangimentos na implementação deste projeto.

Por fim, serão enunciados os requisitos que por algum motivo não puderam ser implementados neste projeto, ficando planeados para logo que possível.

6.1 Reformulação da plataforma antiga

No ponto 1.5 foram mencionados alguns dos problemas identificados na plataforma atual da GTS, que seriam corrigidos na implementação deste projeto. O primeiro problema identificado foi um planeamento menos correto do projeto anterior, uma vez que por ser algo novo, não havia grande experiência no tema, e grande parte dos requisitos apenas apareceram à posteriori e com prazos extremamente curtos, levando a que a sua implementação não fosse realizada da melhor maneira possível. Contudo, devido à alteração das pessoas responsáveis, voltaram a aparecer alterações pelo meio, algumas dúvidas sobre procedimentos e coisas por decidir ou feitas "em cima do joelho", fazendo com que a implementação não fosse tão linear como inicialmente planeada.

6.2 Atualização para tecnologias mais recentes

Com a elaboração deste novo projeto, tendo em mente os requisitos anteriores, assim como sugestões de melhoria, foi possível uma melhor organização do mesmo, de modo a já prever futuras implementações, deixando o projeto o mais escalável possível.

A plataforma antiga foi implementada com a arquitetura MVC, e um dos objetivos era utilizar linguagens de programação mais recentes e onde existisse maior divisão de responsabilidades. Dessa forma, foi implementada uma API REST em *back-end* e o *front-end* em React, tal como descrito no ponto 2.5.4. Assim, foi possível ter uma plataforma mais rápida, que permite inúmeras integrações mais facilmente e onde a lógica está completamente separada da parte visual, permitindo implementações de forma independente.

Já numa fase posterior ao início do projeto, um dos funcionários da empresa ficou responsável pela construção de componentes genéricos em React, para serem utilizados em futuros projetos, de modo que a implementação fosse mais rápida e que o aspeto das plataformas fosse mais uniforme. Uma vez que o projeto já estava numa fase avançada, não foi possível já incluir os componentes da empresa no mesmo, contudo futuramente será feita esta alteração.

Esta é então uma das grandes vantagens do React, que permite a reutilização de componentes previamente implementados. A empresa já possui por exemplo componentes genéricos para alertas, barras de pesquisa dinâmicas, botões, seleção de datas, formulário (*inputs* normais, *checkbox*, *selectbox*), tabelas, *modals* e *sidepanes*, e até um componente base de aplicação, que é ponto central da aplicação. Por este passam todos os pedidos, comunicações, rotas de aplicação, configurações, etc. Ao criar um novo projeto utilizando este componente base de aplicações e todos os restantes componentes genéricos da empresa, é extremamente mais rápido a

sua implementação, e deixa a "imagem" da empresa, uma vez que estes componentes têm por defeito um aspeto visual previamente definido pela equipa de design.

Não existe qualquer dúvida que se estes componentes genéricos estivessem disponíveis antes da implementação deste novo projeto, grande parte da implementação levaria metade do esforço, sendo que seria possível neste momento ter mais funcionalidades disponíveis.

6.3 Área de revendedores e duplo método de autenticação

Outro dos problemas identificados na plataforma anterior, era a área de revendedoras, que não era muito prática. Umas das ideias, seria implementar formulários dos campos necessários ao registo, para que depois pudessem ser customizados e partilhados para os revendedores (fosse por meio de *iframes* ou outro tipo). Devido à escassez de tempo e às prioridades definidas, esta área será implementada numa segunda fase. Contudo, já existiu alguma análise deste problema, e uma das soluções passará pela utilização da biblioteca React-jsonschema-form, presente no ponto 2.6.5. A mesma foi também utilizada no formulário de compra, de forma a guardar o objeto JSON em base de dados, que ao ser chamada em um *iframe* por exemplo, iria gerar o formulário de registo de revendedores automaticamente.

O fato de na versão anterior da GTS, existir um meio de autenticação na plataforma e outra autenticação para ter acesso aos dados do PKI, fazia com que os utilizadores ficassem confusos sobre estes procedimentos. Assim, esta nova implementação foi realizada de forma que a autenticação fosse sempre realizada no PKI (pois é obrigatório que o utilizador se autentique para ter acesso aos seus certificados e dados lá armazenados), sendo que a informação de sessão é posteriormente transmitida para a plataforma, de modo a garantir que o utilizador está autenticado. Desta forma é realizada uma autenticação que é utilizada em ambos os sítios, simplificando o procedimento para os utilizadores. A resolução deste problema é exemplificada no ponto 3.3.1, mais precisamente com recurso à Figura 34.

6.4 Carrinho de compras

Talvez a maior questão que este projeto visava resolver, era a possibilidade de comprar vários produtos em simultâneo. Inicialmente foi necessário analisar a melhor opção para armazenar os dados do carrinho de compras, uma vez que seria possível adicionar produtos ao carrinho sem autenticação, desta forma os dados do carrinho não são enviados para a API. Foi então implementada esta funcionalidade com recurso a *cookies*, onde os dados eram armazenados nas mesmas.

Após a implementação desta funcionalidade foram executados testes automatizados e os resultados obtidos foram todos satisfatórios, sendo que partimos do pressuposto que tudo estava bem. Contudo, quando passamos aos testes manuais executados pelos *beta testers*, estes verificaram que dependendo da quantidade de produtos adicionados ao carrinho, poderiam ocorrer erros devido ao tamanho do objeto JSON armazenado na cookie.

Foi analisado o problema e a solução passaria por duas alternativas. A primeira seria limitar o número de produtos que poderia estar presente no carrinho, contudo esta traria problemas ao requisito RF14 (o sistema deverá permitir a aquisição de produtos por parte de entidade), uma vez que dependendo do tamanho da entidade, a mesma poderia não conseguir comprar em conjunto todos os produtos necessários. A segunda solução e a escolhida, passou então por modificar o local

de armazenamento dos dados do carrinho para o *localStorage*. Enquanto para as cookies o browser dispõe de 4096 bytes de capacidade, no caso do *localStorage* já é possível ter um máximo de 5MB, o que é uma grande diferença.

Inicialmente foi escolhida a *cookie* devido a ser possível controlar o tempo de vida da mesma, enquanto o *localStorage* mantém os dados até os mesmos serem apagados "manualmente". Devido à escassez de tempo ainda não foi possível implementar, contudo numa próxima fase a ideia será voltar a modificar esta parte, de modo a ter uma *cookie* apenas para verificar se existe dados do carrinho em *localStorage*, e caso o mesmo aconteça estes vão ter um tempo de vida igual ao da *cookie*, sendo que o sistema irá verificar que quando a cookie deixar de existir, deve apagar os dados do carrinho em *localStorage*.

Após estas alterações, os testes automatizados foram novamente refeitos de modo a ocorrem com sucesso na implementação em *localStorage* em vez de cookie. Podemos concluir que desta forma, a implementação do carrinho de compras foi bem-sucedida e a mesma funciona da maneira esperada.

6.5 Testes

Tal como mencionado no ponto 1.5, a plataforma anterior foi desenvolvida sem a implementação de qualquer tipo de teste automatizado. Desta forma, qualquer desenvolvimento na plataforma anterior, poderia trazer "consequências" não planeadas e afetar inesperadamente a plataforma. No ponto 5.1 foi então demonstrado a implementação dos testes automatizados neste novo projeto e podemos concluir que os mesmo ocorreram da forma esperada, uma vez que cada alteração ocorrida sem ser planeada, fez com que os testes falhassem ao serem executados.

Com isto não quer dizer que não possam ocorrer erros na plataforma, assim como não dispensa a realização dos testes manuais pelos *beta testers*, contudo se a implementação dos testes automatizados for realizada tendo em conta a maioria dos casos possíveis, a probabilidade de ocorrerem erros é menor.

6.6 Atrasos e constrangimentos na implementação

Um dos acontecimentos que mais prejudicou a elaboração deste projeto, foi o fato de ter aparecido na empresa um novo projeto com maior prioridade, tendo levado à pausa da renovação da GTS até o mesmo estar concluído. Além deste atraso, voltar ao desenvolvimento deste projeto implicou despender mais algum tempo, dada a necessidade de rever o que já tinha sido elaborado, o ponto de situação atual e o desenvolvimento em falta. Ocorreu a saída do responsável de produto e ainda a alteração no responsável da equipa de desenvolvimento, o que também contribuiu para o atraso do projeto.

Aquando da sua iniciação, não havia qualquer experiência na elaboração de um projeto desde a sua fase inicial, na estrutura e organização que o mesmo deveria ter, nem tinha experiência com a utilização do React. Estes fatores levaram a que durante a implementação algumas das decisões tomadas não tivessem sido as melhores, como por exemplo a organização dos ficheiros, a maneira como os pedidos à API são realizados, a implementação do CSS (utilização de folhas de estilo em vez dos estilos utilizados no Material UI), etc. O fato de ter aparecido outro projeto no "meio" deste, foi benéfico para a aquisição de experiência nos pontos anteriormente

enunciados, e ao olhar para o trabalho realizado, é possível concluir que o mesmo poderia ter sido melhorado.

Outro ponto que poderia traria grandes vantagens para este projeto, que estava inicialmente planeado, mas que devido à escassez de tempo, não foi possível realizar, foram os testes de usabilidade. Estes seriam de extrema importância para verificar se o design escolhido era simples e intuitivo para a grande maioria dos utilizadores da plataforma. Contudo é algo que ainda existe esperança de ser realizado antes das novas implementações, de modo a ver se será necessário corrigir alguma coisa.

Um grande entrave no processo de implementação, foram os servidores. Foi realizado um pedido ao departamento técnico da empresa, para a configuração de servidores de desenvolvimento para as equipas de programação, tanto *front-end* como *back-end*. Contudo, devido à quantidade de trabalho que os mesmos tinham, auditórias das diversas plataformas da empresa e outros motivos alheios, os servidores apenas ficaram disponíveis quase no fim da elaboração desta tese.

Isto trouxe vários entraves e constrangimentos no desenvolvimento, uma vez que a solução mais rápida e prática passou pelo *front-end* utilizar diretamente o servidor local do *back-end*. Ou seja, o *back-end* para testar as novas funcionalidades desenvolvidas, criava um servidor local para fazer pedidos e verificar se a implementação estava correta. O *front-end* utilizava esse mesmo servidor para fazer os pedidos necessários também para o seu desenvolvimento. O grande problema desta solução, é que os erros de desenvolvimento do *back-end*, afetavam diretamente os pedidos do *front-end*, havendo a necessidade de inspecionar se o erro ocorrido era de *front-end* ou *back-end*, que por consequência atrasava ainda mais a implementação do mesmo.

6.7 Requisitos em falta

Dos requisitos inicialmente previstos a serem implementados, é apresentado abaixo na Tabela 9 os que não foram possíveis de realizar devido à escassez de tempo. Estes estão relacionados com os revendedores, códigos de desconto, registo de entidades, processo de RGPD e processo de assinatura.

RF8	O sistema deverá permitir a criação de um novo perfil de utilizador identificado como entidade
RF10	O sistema deverá permitir a remoção de contas que não confirmem o email e que não demonstrem “interesse” na plataforma durante um período de 6 meses (RGPD)
RF12	O sistema deverá permitir a extração de dados de conta (RGPD)
RF14	O sistema deverá permitir a aquisição de produtos por parte de entidade
RF22	O sistema deverá permitir a aquisição de produtos através de campanha promocional
RF31	O sistema deverá permitir assinar documentos PDF
RF32	O sistema deverá permitir assinar documentos XML
RF33	O sistema deverá permitir o registo de parceiros
RF34	O sistema deverá possuir uma área para dados estatísticos
RF36	O sistema deverá associar pedidos a parceiros
RF37	O sistema deverá atribuir comissões aos parceiros

RF38	O sistema deverá permitir o levantamento da comissão
RF39	O sistema deverá permitir a utilização da comissão na plataforma da GTS

Tabela 9 - Requisitos em falta

Além dos requisitos anteriores em falta na implementação, será efetuada posteriormente uma análise dos pontos a serem corrigidos, de modo que os mesmos sejam melhorados junto com o desenvolvimento em falta.

Podemos concluir que seguindo todos os passos de um processo de engenharia de software, desde a análise de requisitos, reuniões de *scrum* constantes, *sprints* planeados, implementação de testes automatizados, temos um projeto bem organizado e com uma taxa de sucesso maior. Contudo, nem mesmo seguindo o processo à risca e tendo sido identificadas as falhas da plataforma atual, foi possível terminar o projeto na altura pretendida, seja por alterações e dúvidas nos requisitos ou pelo grande atraso com a introdução de outro projeto prioritário pelo meio.

7. Conclusão

O projeto desenvolvido teve como objetivo a "remodelação" de uma plataforma de venda de certificados digitais de forma que fosse possível comprar diversos produtos ao mesmo tempo e onde os clientes pudessem assinarem digitalmente os seus documentos, tal como sucedia na plataforma antiga.

Tal como mencionado no ponto 6.7, aquando da conclusão desta tese de mestrado, ficaram em falta implementar alguns requisitos devido aos motivos também anteriormente enunciados. Contudo, podemos indicar que se não fosse necessário pausar este projeto devido ao aparecimento de outro, não tivessem saído da empresa algumas pessoas com grandes responsabilidades no mesmo e num contexto normal sem a pandemia, seria expectável que neste momento houvesse um maior avanço e já estivesse implementado o módulo de assinatura de documentos.

Uma vez que a plataforma antiga continua a funcionar sem problemas, foi decidido pela administração da empresa, manter a mesma em funcionamento até a nova plataforma possuir pelo menos a área de assinatura de documentos, de modo a esta deter no mínimo todas as funcionalidades da plataforma anterior.

Apesar de o projeto não ficar totalmente implementado de modo a ser possível colocar em produção e disponibilizado a todos os clientes, o objetivo principal era a implementação da autenticação e do processo completo de compra dos produtos utilizando tecnologias mais recentes, o que sucedeu. Deste modo, podemos afirmar que o objetivo foi alcançado com sucesso.

O fato deste projeto ter sido realizado dentro de um âmbito profissional, foi sem dúvida uma grande vantagem para ganhar imensa experiência, uma vez que no âmbito académico apenas adquirimos uma vertente muito teórica. Nos projetos realizados academicamente onde colocamos em prática a engenharia de software, é aplicado todo o processo "à risca", enquanto no meio profissional, muitas vezes isto não acontece, por diversos motivos. Entre estes temos a falta de tempo, o aparecimento de projetos prioritários, os recursos finitos ou os novos requisitos que aparecem a meio do projeto devido a novos pedidos ou novos clientes.

Este foi um projeto extremamente interessante de ser realizado, uma vez que não é todos os dias que existe a possibilidade de fazermos parte de um novo projeto profissional, elaborado dentro de uma grande empresa, e que é realizado por nós, do início ao fim. É algo extremamente emocionante, uma vez que podemos dar o nosso contributo na solução esperada, mas foi também complicado criar uma base estável sem possuir experiência e com pouco conhecimento nas tecnologias utilizadas.

Assim, neste momento é possível olhar para o projeto desenvolvido, e de acordo com a experiência e conhecimentos adquiridos e os requisitos finais sem alterações pelo meio, verificar que tanto a organização do mesmo como alguns métodos implementados, poderiam ter sido elaborados de outra forma.

Se o projeto fosse iniciado neste momento, uma vez que os componentes genéricos da empresa já estão concluídos, a primeira alteração seria incluir inicialmente o componente base da empresa, que já tem integrado os meios de comunicação do *front-end* com o *back-end*. Esta alteração já iria poupar imenso trabalho despendido a realizar esta tarefa manualmente. Outra alteração, seria a organização dos pedidos desta comunicação, assim como dos componentes reutilizáveis em zonas distintas e de fácil visualização. Seriam também utilizados desde início os componentes de notificações da empresa, em vez de ser necessário criar de raiz os inicialmente utilizados no projeto.

Como trabalho futuro pretende-se maioritariamente implementar os requisitos em falta listados no ponto 6.7 e outros, sendo os mais relevantes os seguintes:

- Implementar a área de assinatura de documentos (PDF e XML), sendo esta realizada de modo a puder ser integrada em outros projetos da empresa que utilizem as assinaturas da GTS, nomeadamente a acinGov e o Ilink;
- Implementar a área de parceiros, para os mesmo puderem gerir todos os seus movimentos, estatísticas e levantamento de comissões;
- Implementar um *backoffice* interno, para os responsáveis do apoio, administradores de registo, marketing e administração de sistemas com as seguintes áreas:
    - Gestão de clientes da plataforma;
    - Gestão de utilizadores do *backoffice*;
    - Gestão de códigos de promocionais utilizados na GTS;
    - Gestão de avisos na plataforma GTS (avisos de manutenção, avisos de cookies, avisos de informação como o COVID);
    - Área de estatísticas relevantes aos utilizadores, produtos e assinaturas da plataforma;
    - Área de marketing, para gestão do *carousel* presente na página inicial da GTS, assim como da criação de futuras campanhas;
    - Área de visualização de acessos à plataforma e *logs* de erros;
    - Área de gestão de documentação, tanto para os clientes como para os utilizadores do *backoffice*;
    - Área de *faqs* interna, respondendo às dúvidas mais frequentes apresentadas pelos responsáveis do apoio e administradores de registo;
    - Área de integrações da GTS com as mais diversas plataformas.

É possível verificar pela lista acima que ainda existe imenso trabalho a ser desenvolvido na plataforma, e que este irá ter uma contínua adição de novos requisitos com o passar do tempo. Desta forma é possível afirmar que os testes unitários foram uma importante adição a este projeto, uma vez que devem garantir que apesar das novas *features*, as anteriores devem continuar a funcionar sem problema, desde que constantemente atualizados. A primeira versão com a área de assinatura de documentos está prevista ser lançada em meados de Março de 2023. Já a área de parceiros e o *backoffice*, tem data prevista para o fim do ano de 2023.

Durante a elaboração do projeto, o ambiente na empresa foi muito agradável, havendo entreajuda na equipa GTS de modo a ultrapassar todos os obstáculos encontrados. Caso necessário os membros das restantes equipas da empresa estavam também sempre disponíveis a ajudar em tudo o que necessário.

As condições de trabalho são muito boas, onde cada colaborador dispõe de um portátil para trabalhar, assim como dois monitores externos, rato e teclado externo, telefone fixo, cadeiras muito confortáveis ar-condicionado, zona de descanso com sofás e máquina de café, zona de refeições com almoço oferecido pela empresa, micro-ondas, frigorifico e máquina com água purificada.

A base teórica captada durante todo o curso, foi fulcral no desenvolvimento deste projeto, de modo a puder implementar todo o processo de software necessário, desde a análise de requisitos, fluxogramas e processos, prototipagem e implementação. Apesar deste ter sido o primeiro projeto executado a nível profissional, a elaboração dos diversos trabalhos universitários ajudou a ganhar alguma experiência nesta parte teórica. O projeto ajudou a perceber a dinâmica existente no ambiente empresarial e a estar constantemente disponível para as constantes alterações que vão surgindo. Foi também importante para perceber como é realizada toda a documentação e

modelação necessária para puder implementar um projeto deste tamanho. As reuniões de *scrum* ajudaram a melhorar a minha comunicação com os colegas de trabalho, assim como a apresentar o trabalho realizado até ao momento.

Apesar de todos os constrangimentos e contratempos ocorridos durante a realização deste projeto, posso afirmar que a experiência foi muito boa, contribuindo para um grande crescimento pessoal e profissional.

8. Referências

[1] G. Rigolin e A. A. Rieznik, “Introdução à criptografia quântica,” 2005. [Online]. Available: https://www.scielo.br/scielo.php?pid=S0102-47442005000400004&script=sci_arttext.

[2] J. Lake, “What is steganography and how does it differ from cryptography?,” [Online]. Available: https://www.crime-research.org/articles/Stegano26.

[3] J. d. S. Abreu, “Passado, Presente E Futuro Da Criptografia Forte: Desenvolvimento Tecnológico E Regulação,” [Online]. Available: https://www.publicacoesacademicas.uniceub.br/RBPP/article/view/4869.

[4] V. M. C. Fiarresga, “Criptografia e Matemática,” [Online]. Available: https://repositorio.ul.pt/bitstream/10451/3647/1/ulfc055857_tm_Victor_Fiarresga.pdf.

[5] S. Singh, The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography, New York: Anchor Books, 1999.

[6] N. G. McDonald, “Past, Present, And Future Methods Of Cryptography And Data Encryption,” [Online]. Available: https://my.eng.utah.edu/~nmcdonal/Tutorials/EncryptionResearchReview.pdf. [Acedido em 21 Janeiro 2021].

[7] S. Meyer, “The History of Cryptography,” Rosen Young Adult, 2017. [Online].

[8] M. Whitman e H. Mattord, “Principles of information security,” 4 Maio 2009. [Online].

[9] L. Mota, “Criptografia, Notas de estudo de Eletrônica,” [Online]. Available: https://www.docsity.com/pt/criptografia-33/4816598/.

[10] A. Sweigart, “Invent Your Own Computer Games with Python,” [Online]. Available: https://inventwithpython.com/invent4thed/chapter14.html.

[11] C. Smith , “Cracking the Enigma code: How Turing’s Bombe turned the tide of WWII,” 2 Novembro 2017. [Online]. Available: http://home.bt.com/tech-gadgets/cracking-the-enigma-code-how-turings-bombe-turned-the-tide-of-wwii-11363990654704. [Acedido em 21 Janeiro 2021].

[12] F. W. Winterbotham, “Enigma, o segredo de Hitler,” BIBLIEX - Biblioteca do Exército, 1978. [Online].

[13] A. A. Rieznik e G. Rigolin, “Introdução à criptografia quântica,” [Online]. Available: https://www.scielo.br/scielo.php?pid=S0102-47442005000400004&script=sci_arttext.

[14] M. Stevens, “San Bernardino shooting updates,” [Online]. Available: https://www.latimes.com/local/lanow/la-me-ln-san-bernardino-shooting-live-updates-htmlstory.html. [Acedido em 26 Janeiro 2021].

[15] “Global Trusted Sign,” [Online]. Available: https://globaltrustedsign.com/.

[16] “ACIN,” [Online]. Available: https://www.acin.pt/pt/about.

[17] “Hostinger,” [Online]. Available: https://www.hostinger.com.br/tutoriais/o-que-e-ssl-tls-https. [Acedido em 24 Março 2021].

[18] T. Reenskaug, “How to Use Model-View-Controller,” [Online]. Available: https://folk.universitetetioslo.no/trygver/1979/sysreq/SysReq.pdf.

[19] T. H. Cormen, Introduction to algorithms, MIT press, 2022.

[20] P. Pinto, “Criptografia simétrica e assimétrica. Sabe a diferença?,” [Online]. Available: https://pplware.sapo.pt/tutoriais/networking/criptografia-simetrica-e-assimetrica-sabe-a-diferenca/. [Acedido em 2 Fevereiro 2021].

[21] V. da Silveira Serafim, “Introdução à Criptografia: Funções Criptográficas de Hash,” [Online]. Available: http://www.serafim.eti.br/academia/recursos/Roteiro_08-Funcoes_de_Hash.pdf. [Acedido em 8 Janeiro 2021].

[22] M. Stevens, “On Collisions for MD5,” 2007.

[23] “An Introduction to Cryptography,” [Online]. Available: https://www.cs.stonybrook.edu/sites/default/files/PGP70IntroToCrypto.pdf.

[24] J. Weise, “Public Key Infrastructure,” 2001. [Online]. Available: http://highsecu.free.fr/db/outils_de_securite/cryptographie/pki/publickey.pdf. [Acedido em 10 Março 2021].

[25] P. Housley, “Planning for PKI: Best Practices Guide for Deploying Public Key Infrastructure,” 2001. [Online].

[26] “Autoridade Certificadora ACIN,” [Online]. Available: https://webgate.ec.europa.eu/tl-browser/#/tl/PT/7.

[27] “Eidas,” [Online]. Available: https://acrobat.adobe.com/pt/pt/sign/compliance/eidas.html.

[28] “Assinatura Electrónica,” Gabinete Nacional de Segurança , [Online]. Available: https://www.gns.gov.pt/assinatura-eletr%C3%B3nica.aspx.

[29] L. M. Kohnfelder, “Towards a Practical Public-key Cryptosystem,” Maio 1978. [Online]. Available: https://simson.net/ref/1978/kohnfelder78.pdf.

[30] “RFC 5280,” [Online]. Available: https://tools.ietf.org/html/rfc5280.

[31] “Autenticação.gov,” [Online]. Available: www.autenticacao.gov.pt.

[32] R. Guimarães Sakurai, “Funções de hash,” [Online]. Available: http://www.universidadejava.com.br/outros/criptografia-funcao-hash.

[33] M. T. Valente , Engenharia de Software Moderna”, 2020.

[34] P. Bourque e R. E. Fairley, Edits., {SWEBOK}: Guide to the Software Engineering Body of Knowledge, Version 3.0 ed., Los Alamitos, CA: IEEE Computer Society, 2014.

[35] R. Pressman, Engenharia De Software, AMGH, 2011.

[36] A. Koscianski e M. d. S. Soares, Qualidade de Software - Aprenda as metodologias e técnicas mais modernas para o desenvolvimento de software, Novatec Editora, 2007.

[37] B. W. Boehm , J. R. Brown e M. Lipow, “Quantitative evaluation of software quality,” 1976.

[38] J. C. Macoratti, “O Processo de Software,” [Online]. Available: www.macoratti.net/proc_sw1.htm. [Acedido em 14 05 2021].

[39] “Desenvolvimento de Sistemas,” [Online]. Available: https://sites.google.com/site/estacioadssi/home.

[40] “Agile Manifesto,” [Online]. Available: https://agilemanifesto.org/iso/ptpt/manifesto.html. [Acedido em 02 Junho 2021].

[41] “JIRA Software,” Atlassian, [Online]. Available: https://www.atlassian.com/software/jira. [Acedido em 3 Junho 2021].

[42] [Online]. Available: https://www.npmtrends.com. [Acedido em 01 Março 2021].

[43] [Online]. Available: https://npm-stat.com. [Acedido em 01 Março 2021].

[44] [Online]. Available: https://www.npmjs.com. [Acedido em 01 Março 2021].

[45] L. Byron, *GraphQL: A data query language*, 14 09 2015.

[46] G. Brito, T. Mombach e M. T. Valente, *Migrating to GraphQL: A Practical Assessment*, 02 2019.

[47] RedHat, "O que é GraphQL?," 19 01 2019. [Online]. Available: https://www.redhat.com/pt-br/topics/api/what-is-graphql.

[48] "A Tree Swing Story," [Online]. Available: https://www.zentao.pm/agile-knowledge-share/tree-swing-project-management-cartoon-97.html.

[49] E. Figueiredo, "Requisitos Funcionais e Requisitos Não Funcionais," [Online]. Available: https://homepages.dcc.ufmg.br/~figueiredo/disciplinas/aulas/req-funcional-rnf_v01.pdf. [Acedido em 21 Junho 2021].

[50] "O que é um fluxograma?".

[51] P. Ventura, "Entendendo o Diagrama de Atividades da UML," 29 Outubro 2016.

[52] J. P. Barros, "Casos de Uso e Respectivos Diagramas," 29 Maio 2009.

[53] IBM, "Generating a private key and certificate request," [Online]. Available: https://www.ibm.com/docs/en/sia?topic=msccu-generating-private-key-certificate-request-1.

[54] C. Costa, "Introdução ao Teste de Software," [Online]. Available: http://pt.slideshare.net/x25treinamento/introduo-ao-teste-de-software.

[55] L. Copeland, "A practitioner's guide to software test design," 2004.

[56] S. Pittet, "An introduction to code coverage," [Online]. Available: https://www.atlassian.com/continuous-delivery/software-testing/code-coverage.

[57] S. Singh, The Code Book: The Secret History of Codes and Code-Breaking Cryptography, Fourth Estate, 2000.

[58] F. Wagner de Cerqueira, "Código Morse," [Online]. Available: https://brasilescola.uol.com.br/geografia/codigo-morse.htm.

[59] A. Nassiri, "Museo della Scienza e della Tecnologia," [Online]. Available: https://commons.wikimedia.org/w/index.php?curid=47910919.

[60] "GraphQL," [Online]. Available: https://graphql.org.

9. Anexos

9.1 Anexo A – Análise da concorrência

9.1.1 Análise das tecnologias

No ponto XXX falamos sobre a análise dos concorrentes da GTS, assim nas tabelas seguintes, é possível analisar as tecnologias utilizadas pelos mesmos.

Multicert	
Linguagem de programação	Node.js
Framework de Javascript	Socket.io
Bibliotecas de Javascript	jQuery Isotope Modernizr
Biblioteca de Fontes	Google Font API
Tag Manager	Google Tag Manager
Markting	BySide
Dados analíticos	Google Analytics Hotjar

Tabela 10 - Tecnologias utilizadas pela Multicert

DigitalSign	
Linguagem de programação	PHP
Servidor Web	Apache
Sistema Operativo do servidor	CentOS
Bibliotecas de Javascript	jQuery jQuery UI core-js

Frameworks UI	Bootstrap
Biblioteca de Fontes	Font Awesome
Tag Manager	Google Tag Manager
Dados analíticos	Google Analytics Facebook Pixel
E-commerce	Funcionalidade carrinho de compras

Tabela 11 - Tecnologias utilizadas pela DigitalSign

9.1.2 Análise dos produtos

9.1.2.1 Selos temporais

Global Trusted Sign	Multicert	DigitalSign
100 Selos (Validade de 1 ano) - 35€ + IVA	50 Selos (Validade de 1 ano) - 27,5€ + IVA	----------
200 Selos (Validade de 1 ano) - 70€ + IVA	100 Selos (Validade de 1 ano) - 37,5€ + IVA	200 Selos (Validade de 1 ano) - 84€ + IVA
600 Selos (Validade de 1 ano) - 150€ + IVA	500 Selos (Validade de 1 ano) - 175€ + IVA	750 Selos (Validade de 1 ano) - 299€ + IVA
5000 Selos (Validade de 1 ano) - 750€ + IVA	----------	----------
Selos à medida (Validade de 1 ano) - 5€ + IVA / 1 Selo	----------	----------

Tabela 12 - Comparação dos selos temporais

9.1.2.2 Certificados Digitais Qualificados (Assinaturas Digitais)

Produto / Empresa	Global Trusted Sign	Multicert	DigitalSign
Certificado Digital Qualificado **Coletivo** (1 ano)	**145€ + IVA** (Renovação **115€ + IVA**)	**145€ + IVA** (Renovação **107€ + IVA**)	**150€ + IVA**
Certificado Digital Qualificado **Coletivo** (2 anos)	**235€ + IVA** (Renovação **210€ + IVA**)	**252,15€ + IVA** (Renovação **202€ + IVA**)	----------
Certificado Digital Qualificado **Coletivo** (3 anos)	**320€ + IVA** (Renovação **276€ + IVA**)	**360,02€ + IVA** (Renovação **283€ + IVA**)	----------
Certificado Digital Qualificado **Singular** (1 ano)	**120€ + IVA** (Renovação **90€ + IVA**)	**135€ + IVA** (Renovação **82€ + IVA**)	----------
Certificado Digital Qualificado **Singular** (2 anos)	**190€ + IVA** (Renovação **160€ + IVA**)	**224,64€ + IVA** (Renovação **152€ + IVA**)	----------
Certificado Digital Qualificado **Singular** (3 anos)	**255€ + IVA** (Renovação **215€ + IVA**)	**322,26€ + IVA** (Renovação **217€ + IVA**)	----------
Certificado Digital Qualificado **Profissional Coletivo** (1 ano)	155€ + IVA (Renovação 120€ + IVA)	135€ + IVA (Renovação 82€ + IVA)	----------
Certificado Digital Qualificado **Profissional Coletivo** (2 anos)	248€ + IVA (Renovação 215€ + IVA)	224,64€ + IVA (Renovação 152€ + IVA)	----------

Certificado Digital Qualificado **Profissional Coletivo** (3 anos)	355€ + IVA (Renovação 288€ + IVA)	322,26€ + IVA) (Renovação 217€ + IVA)	----------
Certificado Digital Qualificado **Profissional Singular** (1 ano)	130€ + IVA (Renovação 100€ + IVA)	135€ + IVA (Renovação 82€ + IVA)	----------
Certificado Digital Qualificado **Profissional Singular** (2 anos)	210€ + IVA (Renovação 180€ + IVA)	224,64€ + IVA (Renovação 152€ + IVA)	----------
Certificado Digital Qualificado **Profissional Singular** (3 anos)	280€ + IVA (Renovação 240€ + IVA)	322,26€ + IVA (Renovação 217€ + IVA)	----------
Certificado Digital Qualificado **Médico** (1 ano)	130€ + IVA (Renovação 100€ + IVA)	135€ + IVA (Renovação 82€ + IVA)	----------
Certificado Digital Qualificado **Médico** (2 anos)	210€ + IVA (Renovação 180€ + IVA)	224,64€ + IVA (Renovação 152€ + IVA)	----------
Certificado Digital Qualificado **Médico** (3 anos)	280€ + IVA (Renovação 240€ + IVA)	322,26€ + IVA (Renovação 217€ + IVA	----------

Tabela 13 - Comparação dos certificados digitais qualificados

9.1.2.3 Selos Eletrónicos

Produto / Empresa	Global Trusted Sign	Multicert	DigitalSign
Selo Eletrónico (1 ano)	**155€ + IVA** (Renovação **120€ + IVA**)	**165€ + IVA** (Renovação **160,05€ + IVA**)	**155€ + IVA** (Renovação **120€ + IVA**)
Selo Eletrónico (2 anos)	**248€ + IVA** (Renovação **215€ + IVA**)	**288,75€ + IVA** (Renovação **274,31€ + IVA**)	----------
Selo Eletrónico (3 anos)	**355€ + IVA** (Renovação **288€ + IVA**)	**405,90€ + IVA** (Renovação **377,49€ + IVA**)	----------

Tabela 14 - Comparação dos selos eletrónicos

9.2 Anexo B – Prototipagem

9.2.1 Página de login

Ao pressionar o botão de login na barra de navegação, somos redirecionados para a página de login (ID II). Esta página irá estar dividida em duas partes, onde na zona lateral direita irá constar um carousel com informações da plaforma e na zona lateral esquerda irá estar o formulário de login para aceder à zona privada da plataforma.

Este formulário também irá possuir ligações para a página de recuperações de password, assim como também para a página de registo.

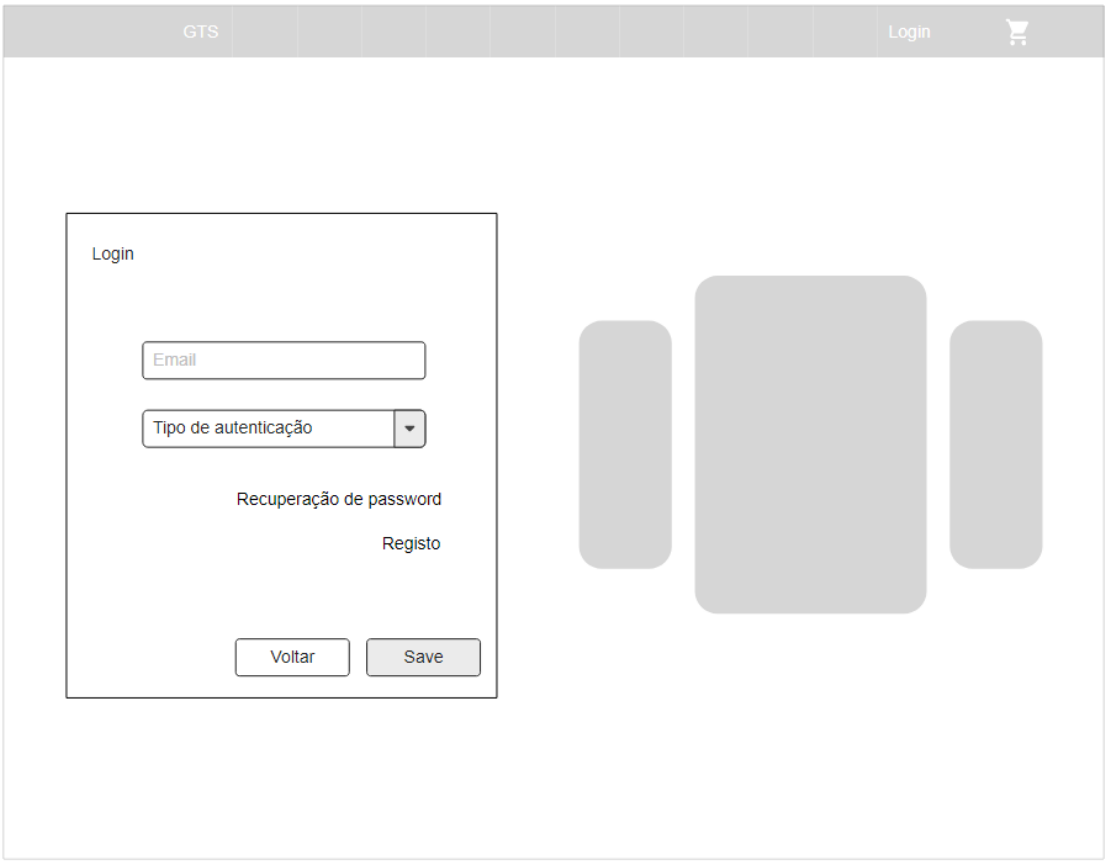


Figura 63 - Protótipo do login

Figura 64 - Mockup da página de login

9.2.2 Página de confirmação do email de um novo registo

Se na página anterior for pressionado o botão de Registo, somos redirecionados para a página onde devemos confirmar o novo email a ser registado (ID III). Este procedimento visa confirmar que a conta realmente pertence a quem está tentando fazer o registo, uma vez que a pessoa terá que colocar o código recebido por email para avançar para o passo seguinte. Esta verificação também valida se o email a registar já está associado a uma conta existente.

Figura 65 - Mockup da confirmação de email de um novo registo

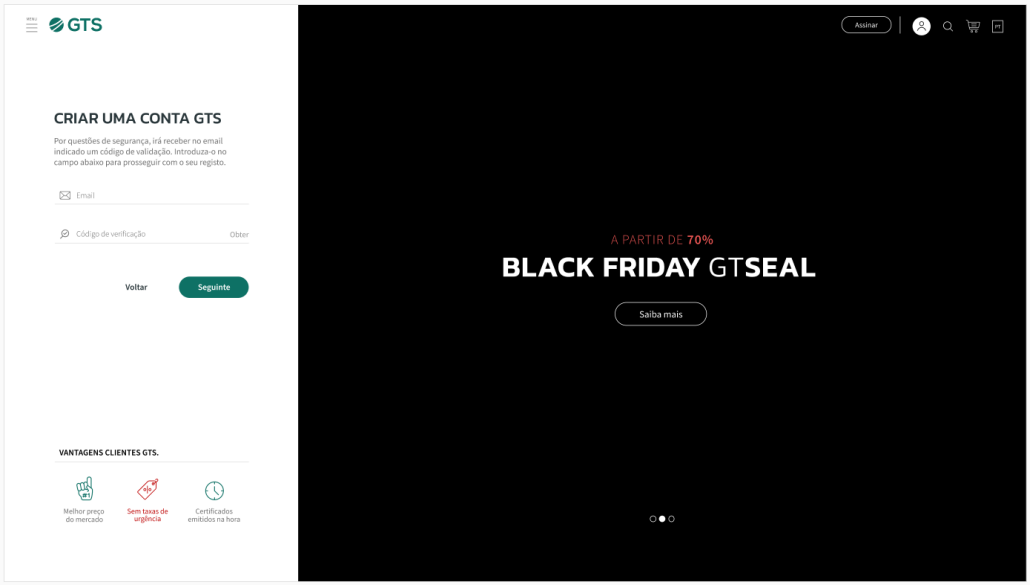


Figura 66 - Protótipo da página de confirmação de email

9.2.3 Página de registo

Após a validação do endereço de email efetuado na página anterior, somos redirecionados para a página de registo onde são introduzidos os dados do novo utilizador (ID IV). Aqui será também validada a força da password introduzida, de modo a respeitar os requisitos mínimos de segurança necessários para a criação de uma password segura.

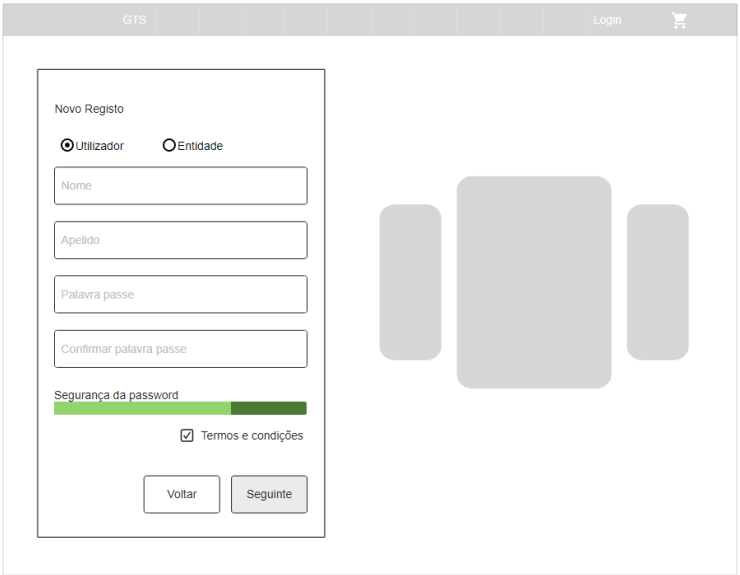


Figura 67 - Mockup da página de registo

Figura 68 - Protótipo da página de registo

9.2.4 Página de informação do produto

No carousel da página inicial, caso o utilizador selecione o botão "Saber mais", será redirecionado para a página informativa do produto em questão (ID V). Esta irá conter o título do produto, assim como uma imagem ilustrativa, seguidamente irá conter um texto explicando o produto selecionado e poderá também conter algum vídeo ou preçário do produto.

Figura 69 - Mockup da página de informação do produto

Figura 70 - Protótipo da página de informação do produto

Figura 71 - Protótipo do preçário dos selos temporais

Figura 72 - Protótipo do preçário dos selos eletrónicos

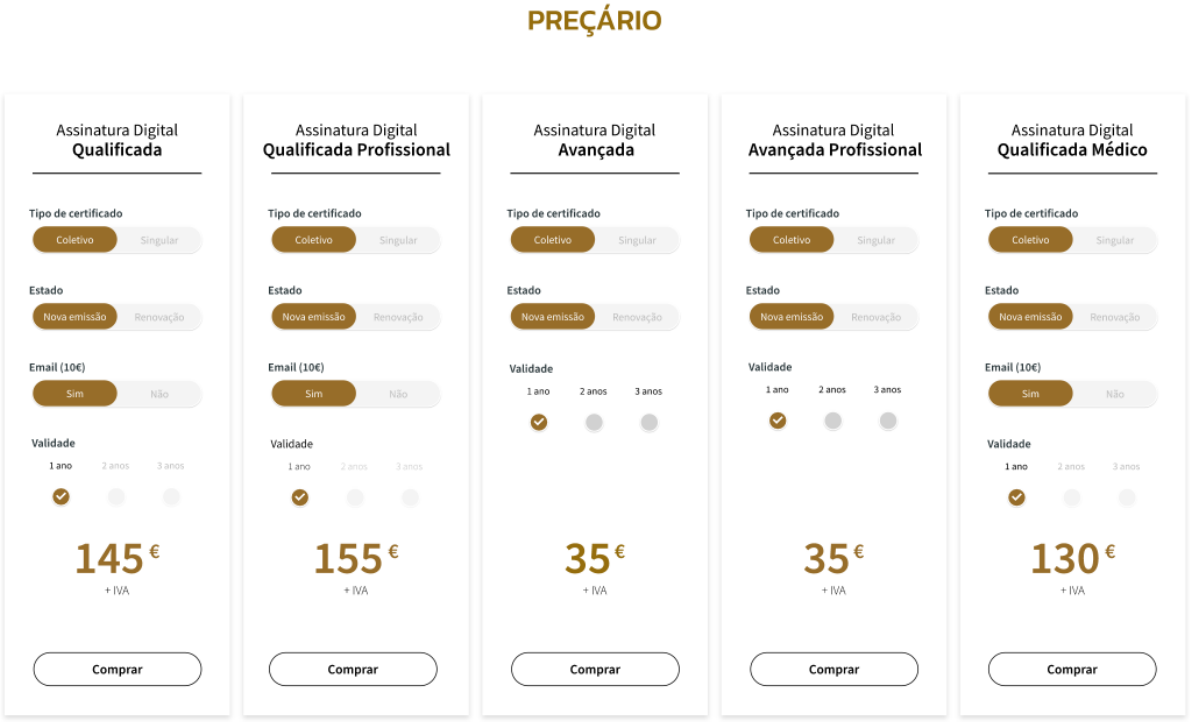


Figura 73 - Protótipo do preçário das assinaturas digitais

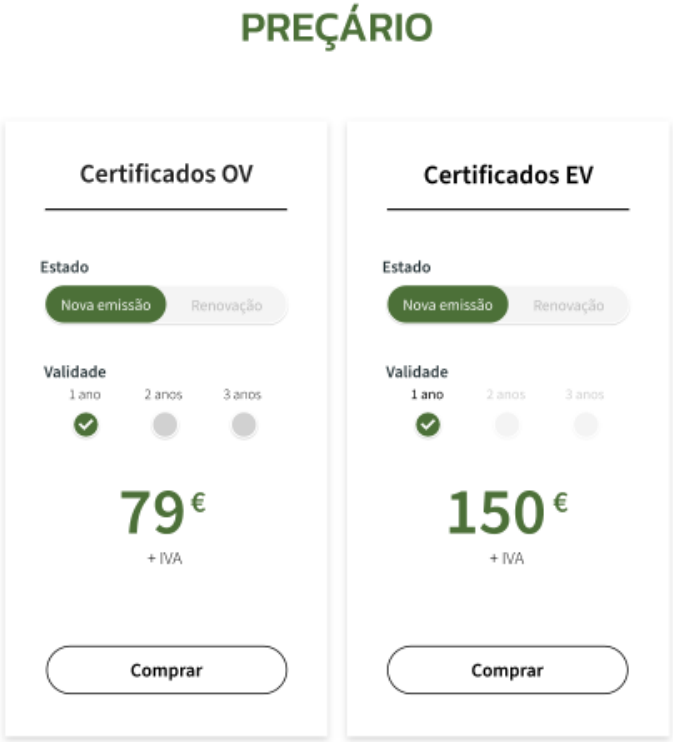


Figura 74 - Protótipo do preçário dos SSL

9.2.5 Página de compra dos produtos

Também no carousel da página inicial, caso o utilizador selecione o botão "Comprar", será redirecionado para a página de compra do produto (ID VI). Aqui teremos um formulário com as opções que o produto em questão pode conter, as quais o utilizador pode adicionar ou remover. Também será possível indicar a quantidade pretendida e visualizar o preço unitário, assim como o preço de acordo com as quantidades selecionadas e os "extras" pretendidos.

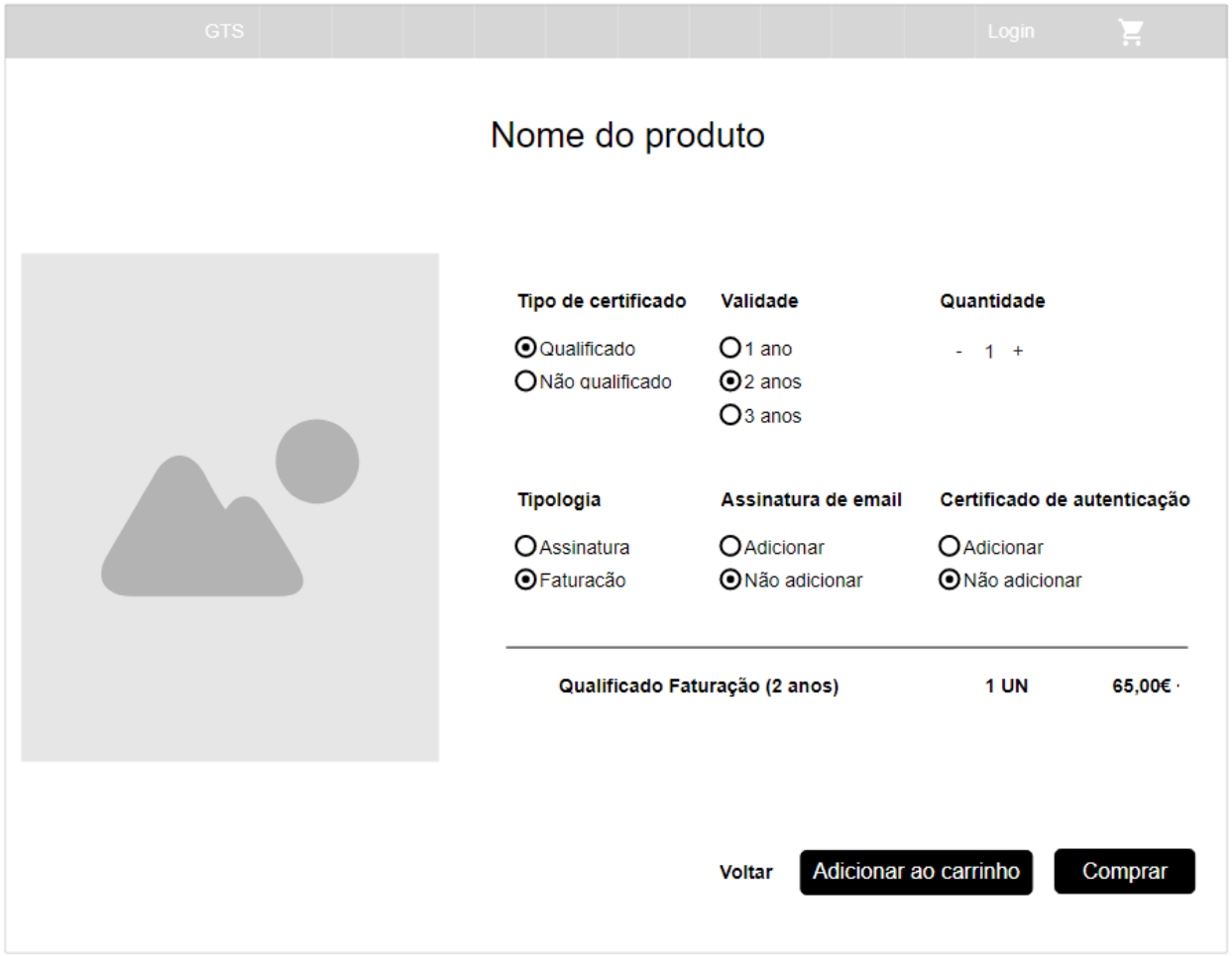


Figura 75 - Mockup da página de compra dos produtos

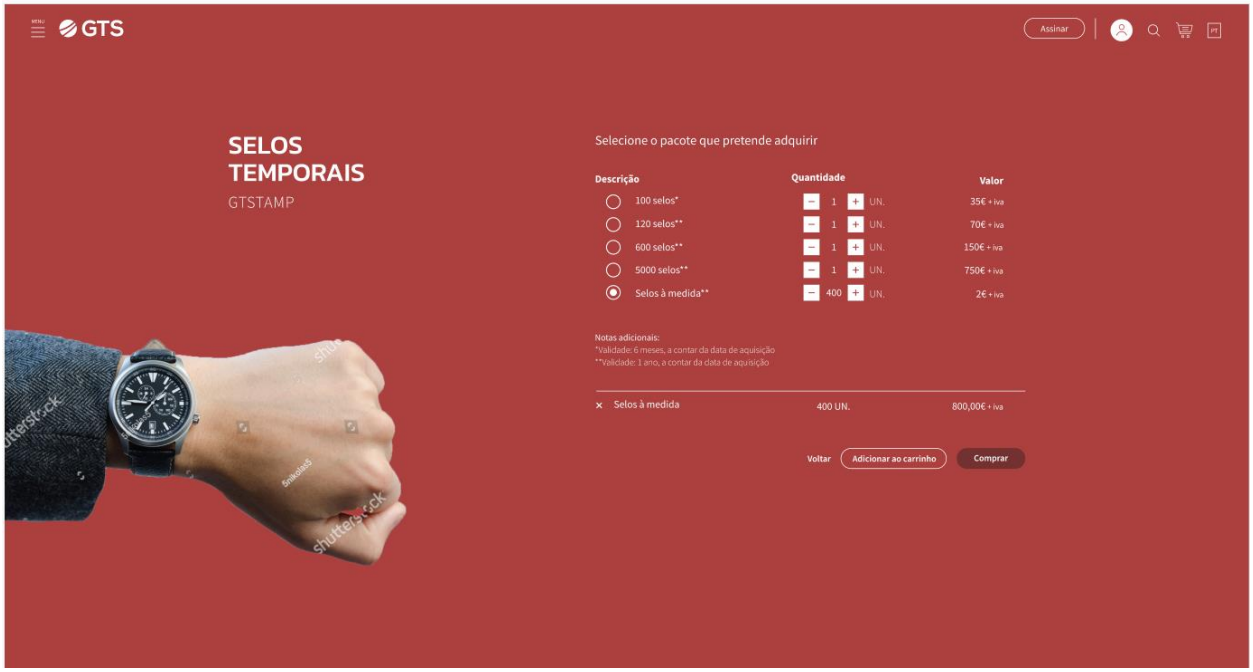


Figura 76 - Protótipo da página de compra dos selos temporais

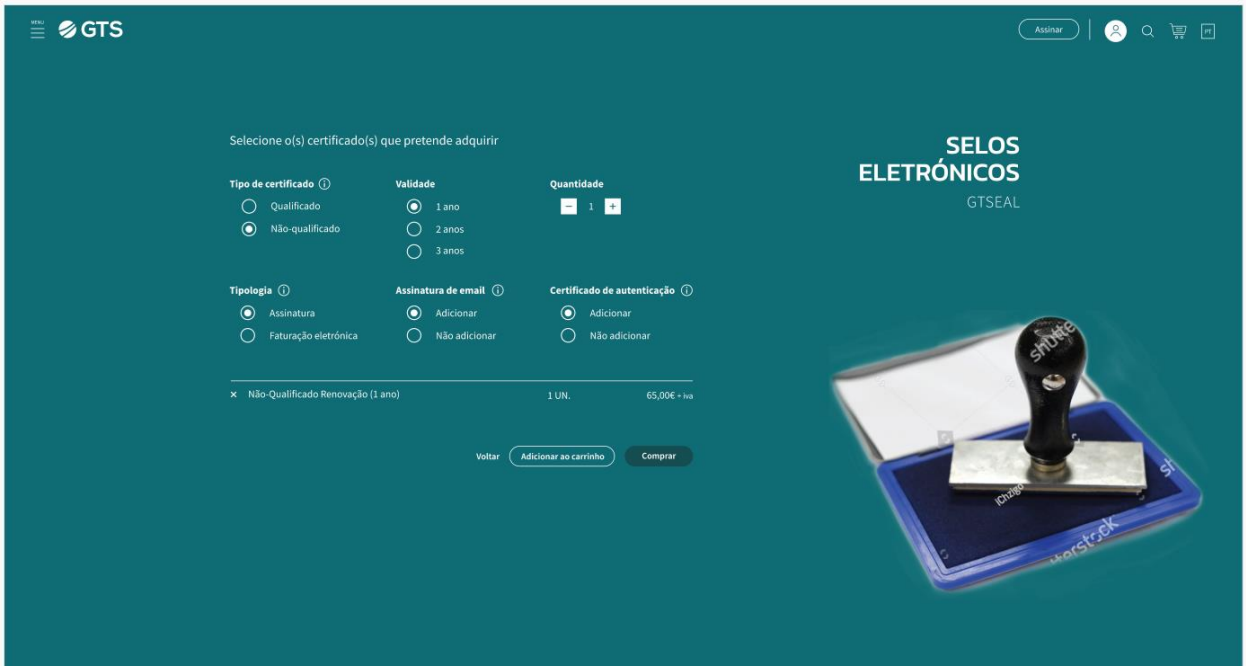


Figura 77 - Protótipo da página de compra dos selos eletrónicos

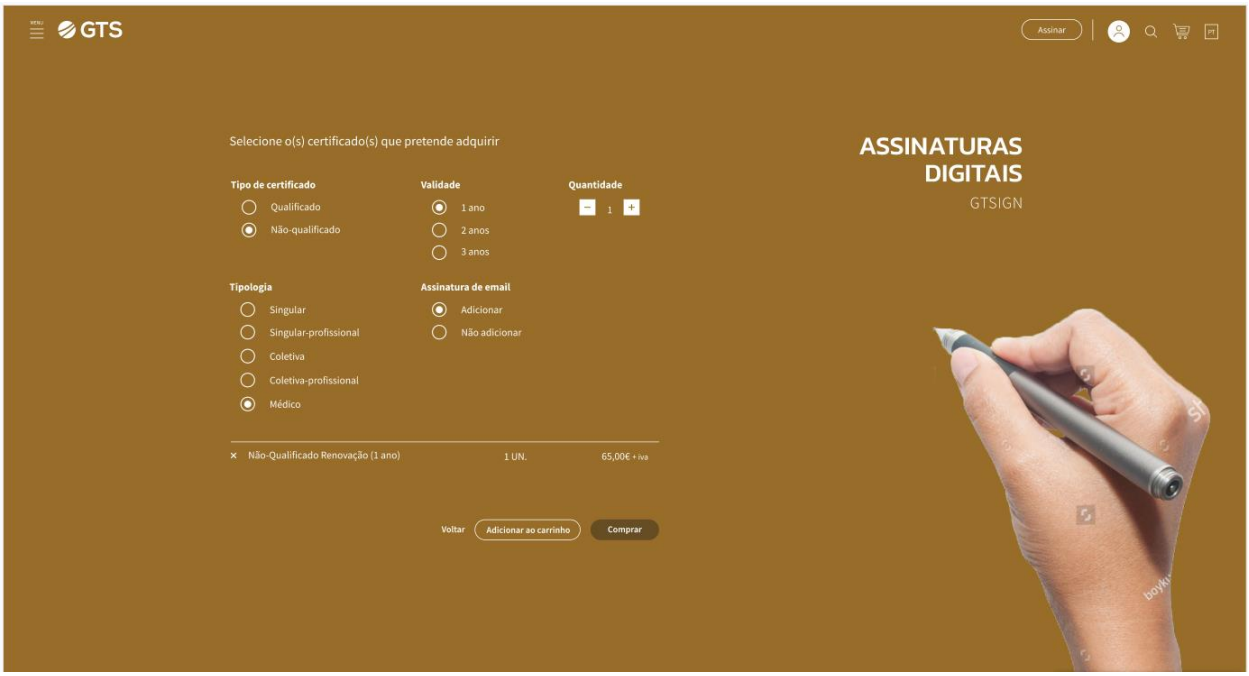


Figura 78 - Protótipo da página de compra das assinaturas digitais

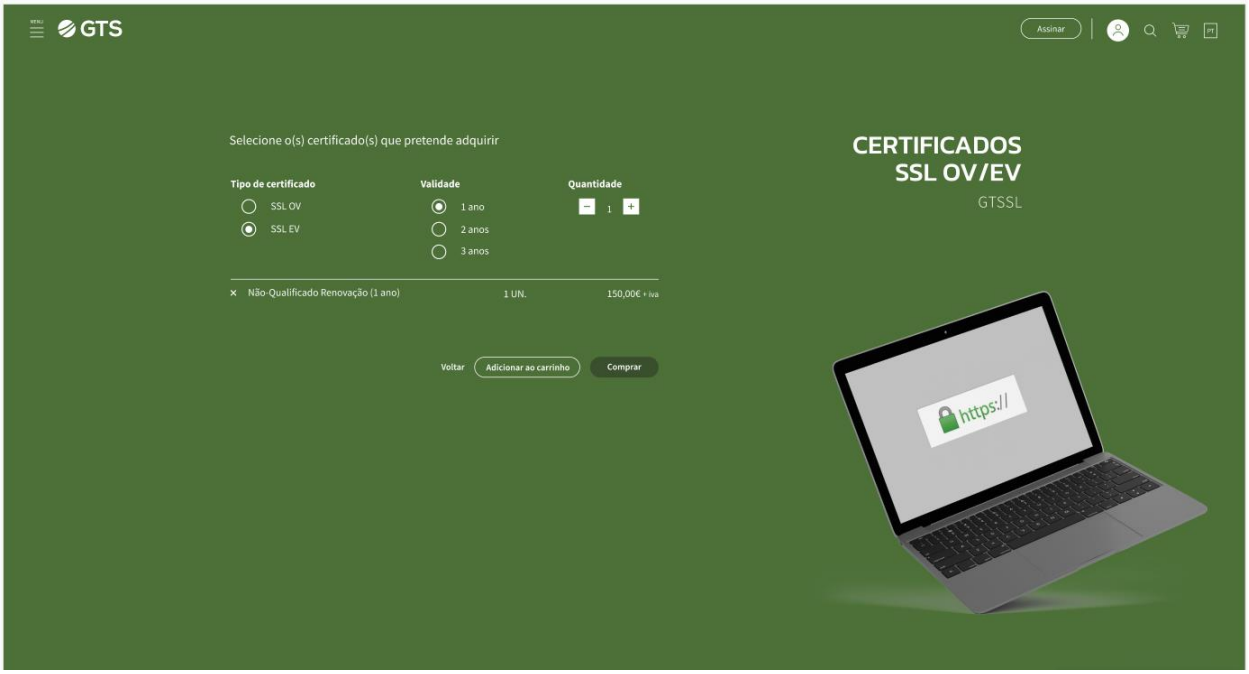


Figura 79 - Protótipo da página de compra dos SSL

9.2.6 Carrinho de compras

Em qualquer uma das páginas da aplicação, será possível no menu de navegação visualizar o carrinho de compras (ID VII). Ao pressionar o ícone deste, irá aparecer mais informações sobre os produtos adicionados ao carrinho, nomeadamente o número total de produtos, a quantidade de cada produto, o preço total de cada produto e o preço total do carrinho. Também será possível aumentar e diminuir a quantidade de cada produto, assim como eliminar apenas um tipo de produto, ou eliminar o carrinho de compras na totalidade. Por último, também irá possuir um botão no fundo para redirecionar para o processo de pagamento.

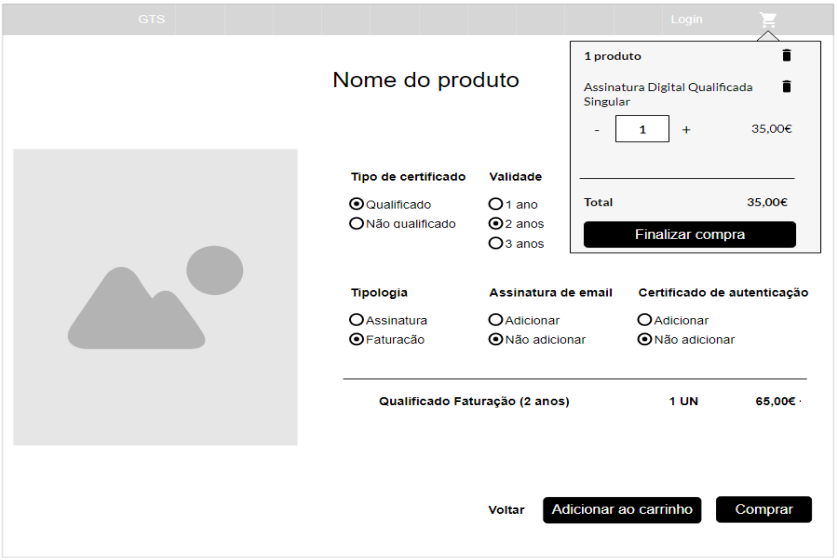


Figura 80 - Mockup do carrinho de compras

Figura 81 - Protótipo do carrinho de compras

9.2.7 Wizard de finalização da compra

Se o utilizador no passo anterior, no carrinho de compras, pressionar o botão de "Finalizar compra" ou se na página de compra de produtos, pressionar no fundo o botão "Comprar", será redirecionado para a página do wizard de finalização de compra (ID VIII).

No primeiro passo do wizard, é apresentado um resumo do conteúdo presente no carrinho de compras, onde o utilizador pode adicionar/remover elementos e modificar as quantidades.

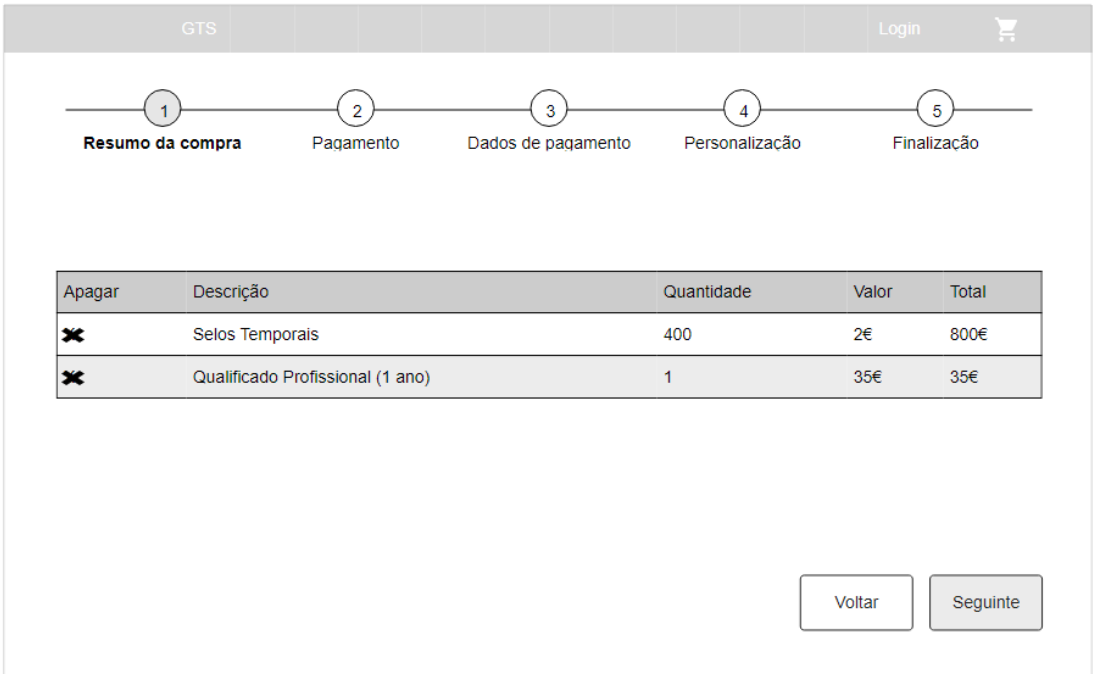


Figura 82 - Mockup do 1º passo do wizard

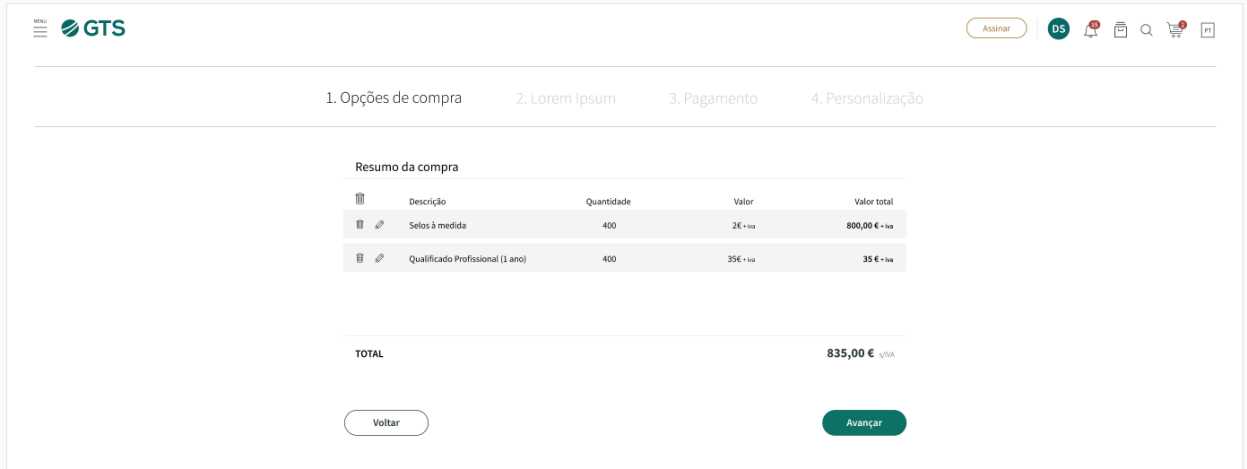


Figura 83 - Protótipo do 1º passo do wizard

Já no segundo passo, o utilizador deve selecionar o meio de pagamento, assim como inserir os dados que pretende ter na fatura da compra. Por fim, é novamente apresentado o resumo da compra, contudo este já não permite alterações.

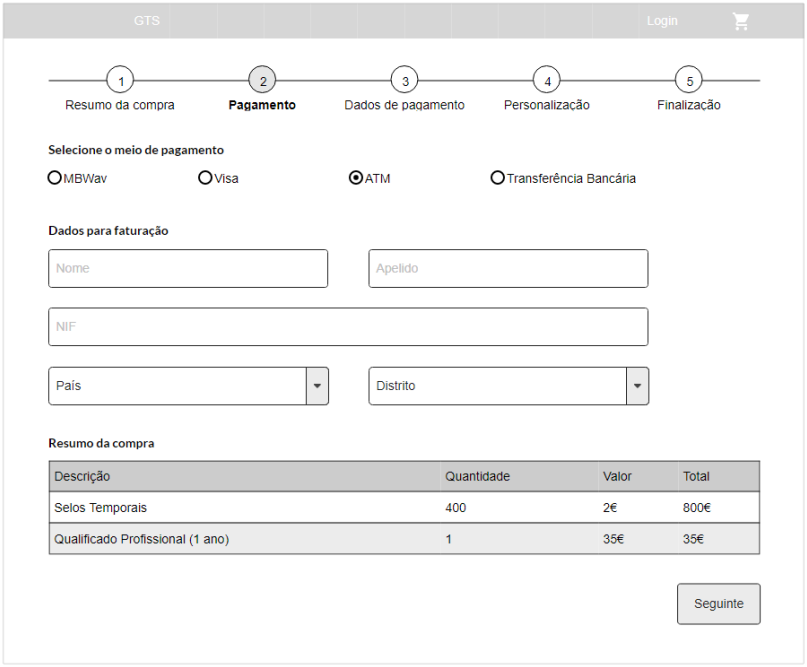


Figura 84 - Mockup do 2º passo do wizard

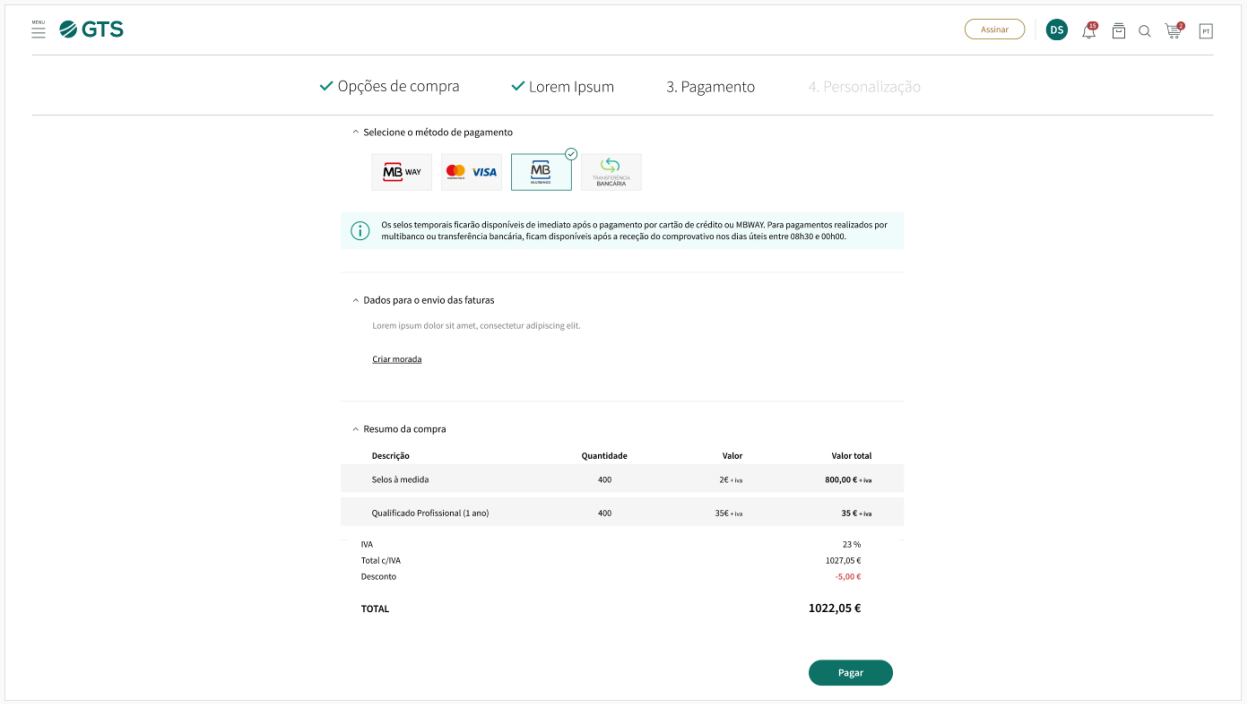


Figura 85 - Protótipo do 2º passo do wizard

No terceiro passo, é apresentado ao utilizador o título de acordo com o meio de pagamento escolhido, se for MBWay ou cartão de crédito, o pagamento já foi efetuado com sucesso, caso seja transferência bancária ou por referência multibanco, é apresentado ao utilizador os dados de pagamento, seja a referência ou o NIB.

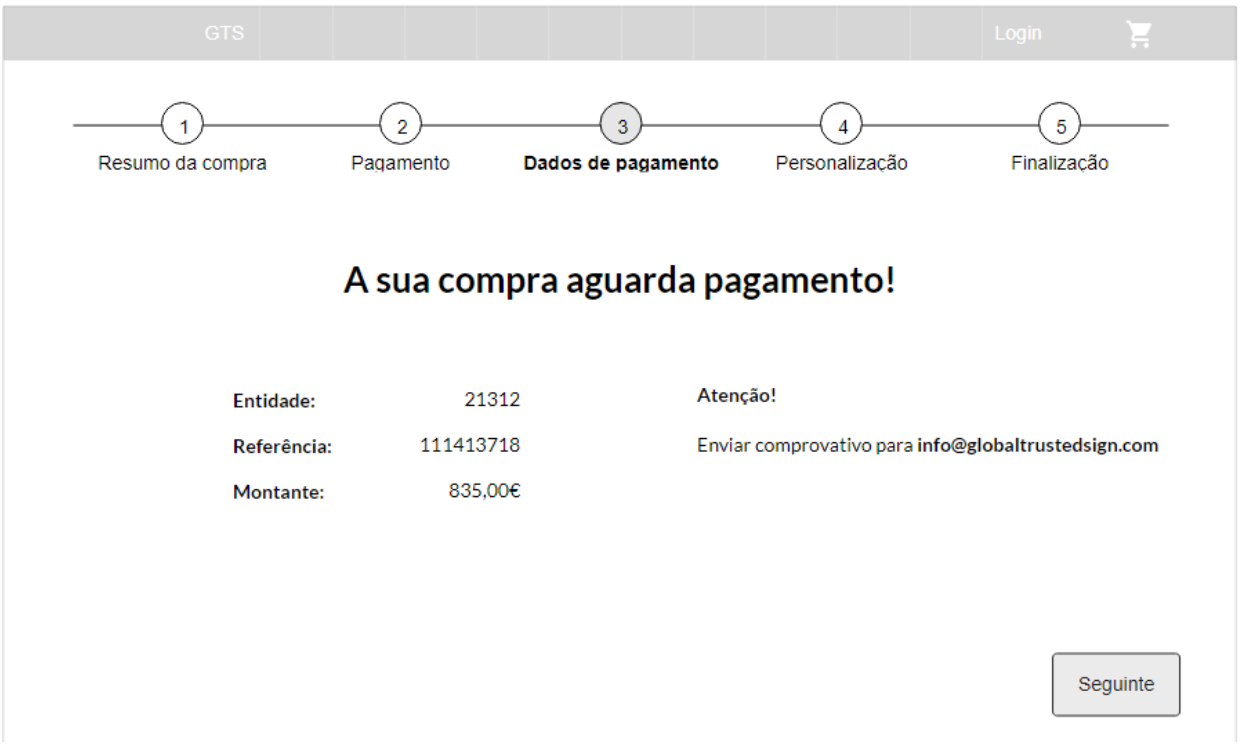


Figura 86 - Mockup do 3º passo do wizard

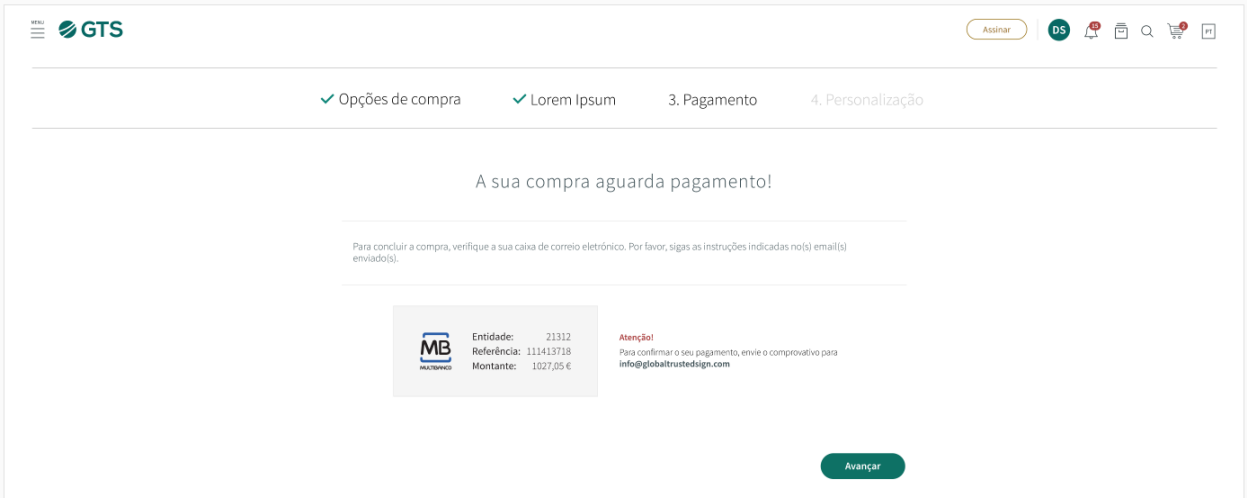


Figura 87 - Protótipo do 3º passo do wizard

No quarto passo, o utilizador deve preencher os dados que pretende que o seu certificado apresente, nomeadamente o nome, apelido, número de identificação fiscal, número do documento de identificação, validade do mesmo, país, assim como fazer o upload de uma cópia do seu documento de identificação de modo a confirmar os dados que foram submetidos. Este passo é opcional, pois pode ser preenchido posteriormente na área do cliente, contudo não irá conseguir obter o certificado sem preencher estes dados.

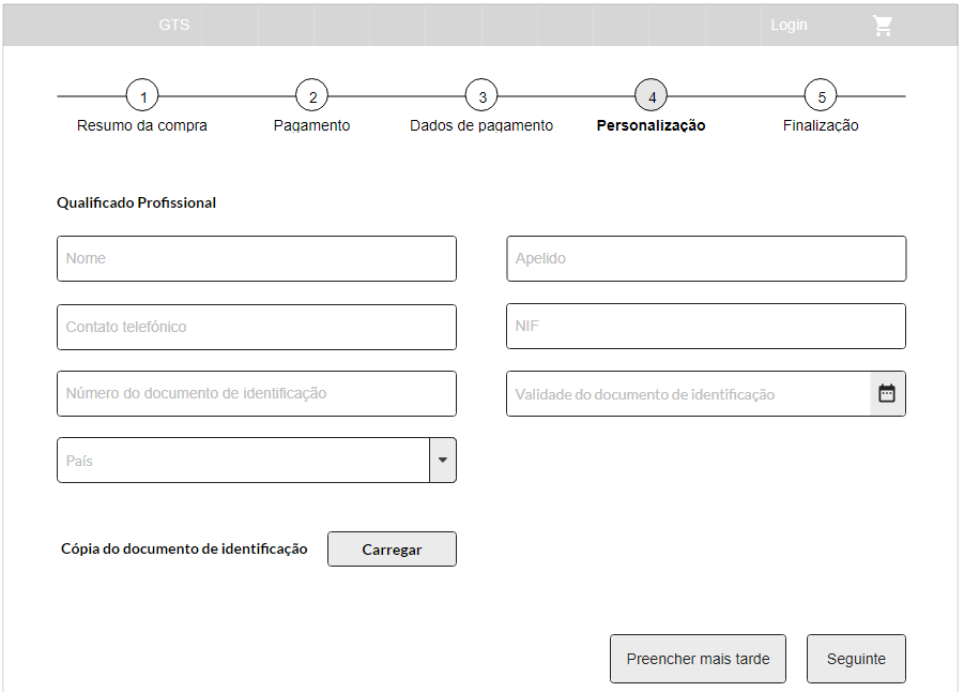


Figura 88 - Mockup do 4º passo do wizard

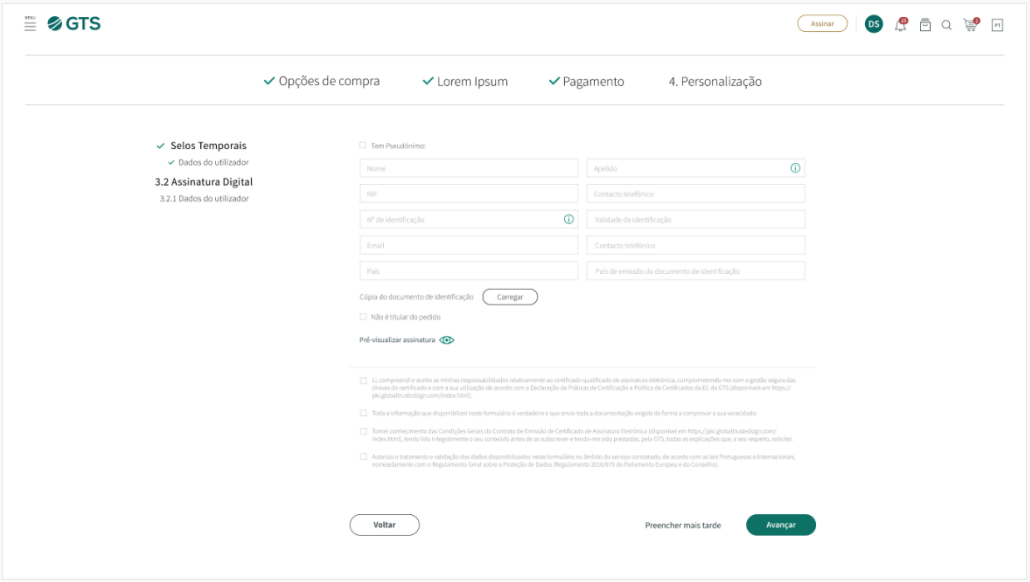


Figura 89 - Protótipo do 4º passo do wizard

Por fim, no quinto passo, é pedido ao utilizador para efetuar uma avaliação do que achou do processo de aquisição de novos produtos. Este pode efetuar uma pontuação de 1 a 5, assim como escrever um pequeno texto sobre esta mesma avaliação. Este passo é também opcional e pode ser efetuado posteriormente.

Figura 90 - Mockup do 5º passo do wizard

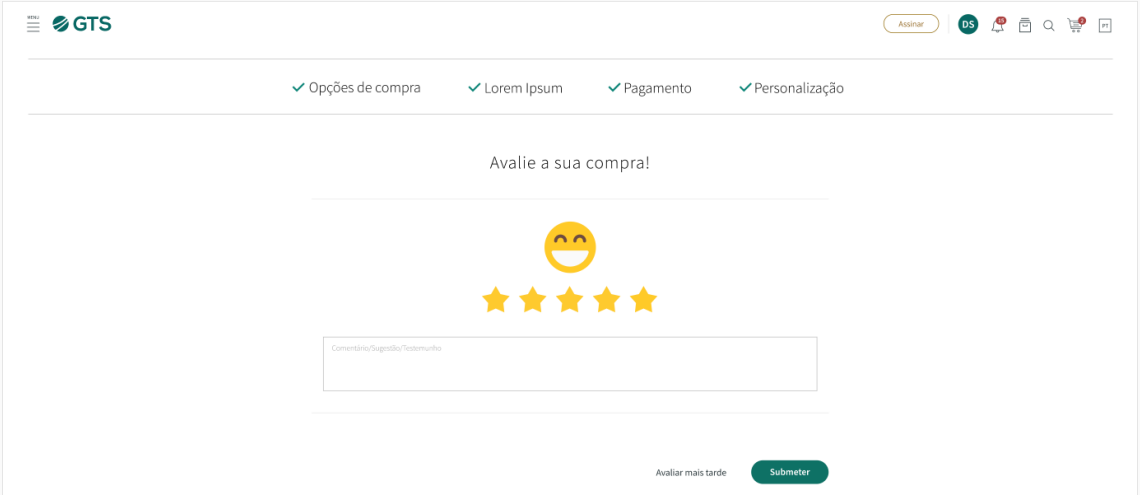


Figura 91 - Protótipo do 5º passo do wizard

9.2.8 Página de mudança de idioma

Na barra de navegação, deverá também ser possível efetuar a alteração do idioma da plataforma (ID IX). Este deverá contemplar no mínimo as três línguas atualmente disponíveis.

Figura 92 - Mockup da mudança de idioma

Figura 93 - Protótipo da alteração de idioma

9.3 Anexo C – Testes unitários e de integração

Selos Electrónicos	
✓	library renders correctly (167 ms)
✓	send correct default electronic seal form values (49 ms)
✓	change quantity in electronic seal form (78 ms)
✓	electronic seal form qualified 2 years with signature (34 ms)
✓	electronic seal form non-qualified 3 years without signature (45 ms)
✓	electronic seals resume with default mock store (63 ms)
✓	electronic seals resume with mock store without signature, non-qualified 2 years, quantity 2 (15 ms)
✓	component renders correctly (104 ms)
✓	component integration electronic seals form and resume (70 ms)
✓	component integration electronic seals form and resume, change store values (68 ms)
✓	get correct electronic seals products (322 ms)
Testes	**11 corretos**, 11 total
Snapshots	**2 escritos**, 2 total
Tempo	**15.052 s**

Tabela 15 - Testes aos selos electrónicos

Selos Temporais	
✓	library renders correctly (186 ms)
✓	get correct number of products in form (32 ms)
✓	check Timestamp 100 is in document (38 ms)
✓	check error on submit without selecting nothing (280 ms)
✓	change quantity in timestamps form (220 ms)
✓	empty timestamps resume with mock store (4 ms)
✓	not empty timestamps resume with mock store (101 ms)
✓	integrate timestamps form and resume (334 ms)
✓	get correct timestamps products (278 ms)
Testes	**9 corretos**, 9 total
Snapshots	**1 escrito**, 1 total
Tempo	**15.725 s**

Tabela 16 - Testes aos selos temporais

Idioma	
✓	language btn renders correctly (75 ms)
✓	default language (21 ms)
✓	es language (18 ms)
✓	en language (22 ms)
✓	wrong language, get pt default (17 ms)
✓	language menu default (153 ms)
✓	language menu change to EN (60 ms)
✓	language menu change to ES (57 ms)
✓	language menu change to PT (83 ms)
Testes	**9 corretos**, 9 total

Snapshots	**2 escritos**, 2 total
Tempo	**9.771 s**

Tabela 17 - Testes ao idioma

Carousel principal	
✓	renders component without crashing (32 ms)
✓	get correct number of products in carousel from mock (106 ms)
✓	check SELOS\nTEMPORAIS is in document (73 ms)
✓	select buy timestamps (65 ms)
✓	get correct products in carousel from API (163 ms)
Testes	**5 corretos**, 5 total
Snapshots	**1 escrito**, 1 total
Tempo	**13.541 s**

Tabela 18 - Teste carousel

Menu lateral	
✓	menu component renders correctly (217 ms)
✓	get correct sub menus names (100 ms)
✓	get correct sub menus links (51 ms)
Testes	**3 corretos**, 3 total
Snapshots	**1 escrito**, 1 total
Tempo	**13.541 s**

Tabela 19 - Teste menu lateral

Menu superior	
✓	navbar component renders correctly (207 ms)
✓	get correct components on navbar (74 ms)
✓	open navbar menu (376 ms)
✓	navbar logo with default href (65 ms)
✓	navbar logo with en href (63 ms)
✓	navbar logo with en href (63 ms)
✓	navbar login menu (83 ms)
✓	navbar language menu (161 ms)
✓	navbar shopping cart (82 ms)
Testes	**9 corretos**, 9 total
Snapshots	**1 escrito**, 1 total
Tempo	**16.578 s**

Tabela 20 - Teste menu superior

Componente de notificações	
✓	component renders correctly (20 ms)
✓	notify success message (52 ms)
✓	notify error message (18 ms)
✓	notify info message (15 ms)
✓	notify warning message (16 ms)
Testes	**5 corretos**, 5 total
Snapshots	**1 escrito**, 1 total
Tempo	**9.691 s**

Tabela 21 - Teste notificações

Carrinho de compras	
✓	component renders correctly (312 ms)
Testes	**1 correto**, 1 total
Snapshots	**1 escrito**, 1 total
Tempo	**10.014 s**

Tabela 22 - Teste carrinho de compras