Gestão de entradas e saídas do COA

Relatório de Estágio

Pedro Rafael Felício da Conceição

Licenciatura em

Informática

Ponta Delgada
2022

Gestão de entradas e saídas do COA

Relatório de Estágio

Pedro Rafael Felício da Conceição

Orientadores

1TEN Ana Sofia Bouça Junqueiro Vilas - COA

Doutora Elisabete Maria da Silva Raposo Freire - UAç

Relatório de Estágio submetido como requisito parcial para obtenção do grau de Licenciado em Informática.

AGRADECIMENTOS

Gostaria de agradecer à Doutora Elisabete Freire pelo extraordinário apoio dado no decorrer do projeto.

Gostaria de agradecer também à 1TEN Junqueiro Vilas, ao SAJ João Marques e ao CAB Mário Vieira, pelo apoio prestado ao longo destes meses e que sem eles seria impossível levar este projeto a bom porto.

Gostaria ainda de agradecer ao Miguel Fernandes, à Beatriz Silva, ao Pedro Sousa e ao Salif Faustino, pelo apoio que deram que em muito contribuiu para a superação em diversos problemas no desenvolvimento do presente trabalho.

RESUMO

O estágio curricular foi desenvolvido na entidade Comando Operacional dos Açores e teve como objetivos o desenvolvimento de uma aplicação desktop para controlo de entradas e saídas de visitantes a entidades do Comando e para controlo de entradas e saídas de viaturas pertencentes ao parque automóvel desta unidade militar. Para atingir os objetivos foi necessário desenvolver uma base de dados SQL e operações CRUD sobre a mesma e desenvolver as funcionalidades para efetuar os controlos acima referidos. Todos os objetivos propostos no plano de formação inicial foram cumpridos com sucesso, no entanto poderiam ter existido algumas melhorias como por exemplo, no desenvolvimento de *mockups*. Este trabalho assumiu elevada importância pois até este momento o controlo de pessoas e veículos era efetuado num registo físico e por vezes estes registos eram extraviados, impossibilitando assim garantir totalmente a segurança da unidade militar e de quem lá presta serviço.

Palavras-chave: Engenharia de software; bases de dados; desenho de software; arquitetura cliente-servidor.

ABSTRACT

The curricular internship was developed in the entity Comando Operacional dos Açores and had as objectives the development of a desktop application to control entrances and exits of visitors to entities of COA and to control the entrances and exits of vehicles belonging to the car park of this military unit. To achieve the objectives, it was necessary to develop a SQL database and its CRUD operations and develop the features to carry out the above mentioned controls. All the objectives proposed in the initial formation plan were successfully accomplished , however there could have been some improvements, such as in the development of mockups. This work took great importance because until this time the control of people and vehicles was carried out in a physical register and sometimes these records were misplaced, thus making it impossible to fully guarantee the security of the military unit and of those who work there.

Keywords: Software engineering; data base; software scheme; client-server architecture

ABREVIATURAS, ACRÓNIMOS E SIGLAS

BD - Base de dados

COA - Comando Operacional dos Açores

API - *Application Programming Interface*

UI - *User Interface*

ORM - *Object Relational Manager*

SQL - *Structured Query Language*

NoSQL - *Not Only SQL*

CRUD - *Create, Read, Update, Delete*

CISMIL - Centro de Informações e Segurança Militar

NIP - Número de Identificação Pessoal

NII - Número de Identificação Individual

NIM - Número de Identificação Militar

ÍNDICE

ÍNDICE DE FIGURAS

INTRODUÇÃO

O presente relatório é elaborado no contexto do estágio curricular do 3º ano da licenciatura em Informática profetizada na Universidade dos Açores. O estágio foi desenvolvido no Comando Operacional dos Açores e consistiu no desenvolvimento de uma aplicação desktop para controlar as entradas e saídas de visitantes a entidades do COA e controlar as entradas e saídas de viaturas do parque automóvel do mesmo. Para além dos objetivos referidos anteriormente existiu a necessidade de desenvolver uma base de dados e as operações CRUD sobre a mesma.

Esta solução é desenvolvida devido à necessidade de aumentar o nível de segurança física desta unidade militar, proteger o meio ambiente e a acompanhar a evolução tecnológica. Para proceder à implementação da solução são utilizadas tecnologias como o *Firebase, Pycharm, Notion, Project Libre, Qt Designer* entre outras. A metodologia associada ao desenvolvimento foi a metodologia SCRUM.

O presente relatório, conta com uma breve descrição da entidade de acolhimento, seguida por uma explicação do plano de formação, das metodologias, ferramentas e tecnologias utilizadas. Depois é explicada a fase de desenvolvimento do projeto, de seguida uma reflexão critica dos seminários assistidos no âmbito da cadeira, seguidamente uma breve conclusão com uma análise critica do desenvolvimento e por fim os anexos.

- 1 -

ENTIDADE DE ACOLHIMENTO

O Comando Operacional dos Açores é um órgão de comando e controlo de natureza conjunta, de nível operacional, que tem por missão efetuar o planeamento, o treino operacional conjunto e o emprego operacional das forças e meios que lhe sejam atribuídos. De acordo com as competências do Chefe do Estado-Maior-General das Forças Armadas, no âmbito regional, compete ao COA elaborar e atualizar os planos de defesa militar e de contingência a nível regional, planear e executar as medidas relativas à defesa militar do arquipélago dos Açores, planear, executar e avaliar o treino operacional conjunto, planear, treinar e coordenar a participação das Forças Armadas em ações de proteção civil, colaborar no processo de certificação de forças conjuntas e planear e coordenar a realização de cerimónias militares conjuntas.

- 2 -
PLANO DE FORMAÇÃO

Neste capítulo, é onde será explicado o plano de formação inicial elaborado e apresentado.

2.1. O plano de formação inicial

2.1.1. Enquadramento do projeto

A segurança é um ponto fundamental para todas as instituições. Na instituição militar é ainda mais relevante, pois se existir uma quebra de segurança física numa unidade militar, a defesa e soberania nacional podem ser postas em causa.

Devido às graves alterações climáticas que se têm vindo a acentuar ao longo dos anos e, segundo o Diário de Notícias, "Portugal já ultrapassou o limite em todas as categorias ambientais (…) os limites ecológicos" [1], especialmente nos campos das alterações climáticas, poluição e pressão sobre os ecossistemas, levando a que seja de vital importância reduzir a quantidade de papel gasto nas instituições públicas.

Como o mundo atual está em constante mudança tecnológica, gerou-se a necessidade de implementar um meio para exercer um controlo mais fácil e rigoroso das entradas e saídas deste Comando, eliminando assim a utilização de papel neste processo.

2.1.2. Objetivos gerais e específicos

Em traços gerais, pretende-se o desenvolvimento de uma aplicação desktop que permita efetuar o controlo de entradas e saídas da unidade militar de pessoas exteriores à mesma.

Esta aplicação deve conter os mesmos parâmetros de segurança para as entradas e saídas já em vigor na instituição. Para este propósito, o software deve permitir o registo na plataforma da entrada de pessoas, introduzindo os seus dados pessoais (nome e número de cartão de cidadão), a matrícula da viatura, a entidade a ser visitada, a hora de entrada e de saída e o motivo da visita.

Caso o desenvolvimento do projeto o permita, numa segunda fase, pretende-se expandir este software com um módulo de gestão de controlo de entradas e saídas de viaturas a

cargo do Comando de forma a saber que viaturas saem, quem as conduz, a que horas saíram, a que horas entraram e os quilómetros efetuados.

Através destes objetivos gerais foram definidos os seguintes objetivos específicos:

1) Desenvolver uma base de dados;
2) Desenvolver uma funcionalidade para registar as entradas dos visitantes;
3) Desenvolver uma funcionalidade para registar as saídas dos visitantes;
4) Desenvolver uma funcionalidade para garantir que o visitante de facto visitou a entidade através de uma aplicação local descentralizada;
5) Desenvolver uma funcionalidade para gerir a base de dados dos colaboradores do COA;
6) Desenvolver uma funcionalidade para autenticar os operadores;
7) Desenvolver uma funcionalidade para registar os colaboradores;
8) Redação do relatório de estágio;
9) Preparação das apresentações do estágio;
10) Assistir a seminários.

Objetivos opcionais:

11) Desenvolver uma funcionalidade para registar as viaturas que saem do Comando;
12) Desenvolver uma funcionalidade para registar as viaturas que entram no Comando;

2.1.3. Atividades principais

Como atividades principais para o desenvolvimento do projeto foram identificadas as seguintes atividades a desenvolver: formação, elaboração do documento de requisitos, análise e desenho do sistema, desenvolvimento da base de dados e o desenvolvimento das funcionalidades do sistema.

Na formação, foram definidas formações em *Qt Designer*, *SQLite* e sobre a plataforma *FireBase*.

No caso do documento de requisitos, procedeu-se ao levantamento de requisitos através de processos de engenharia de requisitos nomeadamente, reuniões síncronas com os *stakeholders* e estudo do modelo de negócio. De seguida, foram elaboradas as personas, os respetivos épicos, a matriz da gestão de riscos, o levantamento de *constrains*, a análise

SWOT, as *user stories*, os critérios de aceitação, os *backlogs* gerais e de *sprint* e os *test result report*.

Na fase de análise e desenho foram elaborados diversos diagramas como o diagrama de casos de uso, de classes, de arquitetura, da base de dados e foram ainda elaborados alguns *mockups* para a *user interface*.

Na fase de desenvolvimento, implementou-se a base de dados e as diversas funcionalidades do sistema, nomeadamente, registar entradas de visitantes, registar saídas de visitantes, verificar se as entidades foram visitadas, gerir a base de dados dos funcionários, autenticação e registo de utilizadores. Todas estas funcionalidades são divididas na análise e correção dos respetivos diagramas, a implementação e a realização de testes e documentação. Estas funcionalidades foram divididas em 3 sprints de desenvolvimento.

2.2. Alterações ao plano de formação inicial

Devido ao processo de desenvolvimento do projeto e por motivos profissionais existiu a necessidade de proceder a alterações na duração das tarefas e como resultado dessas alterações, foi necessário alterar também o respetivo diagrama de *Gantt.*

De seguida surgiu a necessidade de fazer duas alterações às tecnologias utilizadas, uma delas por um erro na escrita do software onde se lê *Astha*, deve-se ler *Astah* e a outra devido ao facto de ter referido a API para o *Python*, *PyQt5*, e não o nome do software a utilizar, o *Qt Designer*.

- 3 -
METODOLOGIAS, FERRAMENTAS E TECNOLOGIAS UTILIZADAS

Neste capítulo serão abordadas as metodologias, ferramentas e tecnologias utilizadas no desenvolvimento do projeto, os problemas alvo de cada método, as vantagens e as desvantagens em relação a alternativas já existentes no mercado.

3.1. Ferramentas e tecnologias utilizadas (vantagens em relação às alternativas)

As principais ferramentas e tecnologias utilizadas para o desenvolvimento do projeto foram as seguintes:

- *Bitbucket (Git)*
- *Pycharm*
- *Axure*
- *Notion*
- *Project Libre*
- *Astah*
- *Firebase*
- *Qt Designer*
- *SqlLite*

Bitbucket (Git) vs GitHub

O *Bitbucket* é um repositório para controlo de versões, que permite partilhar documentos e código com diversas pessoas em simultâneo. Neste projeto foi utilizado para a orientadora e supervisora terem um maior controlo e uma melhor perceção do trabalho desenvolvido no decorrer do estágio.

As principais diferenças entre o *Bitbucket* e o *GitHub* são o número de colaboradores por repositório, o tamanho da comunidade, o preço do *software*, a adaptação a diferentes

sistemas operativos e a facilidade de interação através da sua UI. As vantagens que o *Bitbucket* tem são o preço do *software*, o número de colaboradores por repositório, 5 para o Bitbucket e 3 para o *GitHub*, a boa adaptação a diferentes sistemas operativos e a facilidade na utilização da sua UI. Por outro lado, o *GitHub* tem vantagens no tamanho da comunidade e na rapidez de acessos feitos ao repositório.

Pycharm vs VsCode

O *software Pycharm* é uma aplicação de desenvolvimento de código na linguagem de programação *Python*. Escolhi esta linguagem de programação pois é a linguagem com que me encontro mais familiarizado.

O *VsCode* tal como o *Pycharm* é uma aplicação para desenvolvimento de *software*, proprietária da Microsoft, permite desenvolver *software* em diversas linguagens de programação e permite ainda adicionar diversos módulos ao *software* base à medida das necessidades.

Neste caso elegi o *Pycharm* por preferência de utilização e porque só suporta a linguagem de programação *Python*.

Axure vs Moqups

O *Axure* é um *software* para desenho de *mockups* ou protótipos funcionais. Foi utilizado neste projeto para a construção de *mockups* na fase de desenho com o objetivo de mostrar o resultado do levantamento de requisitos ao nível da *user interface*, permitindo assim mostrar aos *stakeholders* uma primeira versão ao nível gráfico do produto final.

O site *Moqups* é um *software* tal como o *Axure* para desenho de *mockups*. Esta plataforma somente pode ser utilizada online, é uma aplicação web ligeiramente mais pobre que o *Axure* e com uma curva de aprendizagem um pouco elevada. O ponto positivo de *Moqups* é a possibilidade de desenvolver *mockups* a partir de *templates*.

Escolhi o *Axure* pois é o software que tenho mais experiência entre os dois.

Notion vs Trello

O *Notion* é um *software* que permite fazer a gestão e organização de projetos, com a funcionalidade de mostrar a informação de diversas maneiras diferentes, partilhando-a com diversas pessoas e contém uma versão *free*. Neste projeto foi utilizado para elaborar os *backlogs* (gerais e de sprint) e os *scrumboards.*

O *trello* é uma plataforma para gestão de projetos somente através da *web*, que permite criar *scrumboards* com cartões que podem receber comentários e terem utilizadores associados a estes cartões. O *Notion* é uma aplicação tanto para *web* como para *desktop* e torna-se mais completa porque permite fazer o que a plataforma *trello* faz e ainda contém mais funcionalidades como as já descritas em cima. Por ser um *software* mais completo, foi o eleito entre os dois.

Project Libre vs Nifty

O *Project Libre* é uma aplicação que permite realizar a gestão de projetos, criando diagramas, alocando pessoas a tarefas e ainda realiza os cálculos monetários para o desenvolvimento do respetivo projeto.

O *nifty* é um software que permite fazer diagramas de *Gantt*, associar pessoas, objetivos e *tags* a cada tarefa, contém um fórum de discussões do projeto e permite gerar relatórios automáticos de todos os projetos em andamento. A versão gratuita deste *software* só permite ter 2 projetos em simultâneo e com um espaço de armazenamento em *cloud* de 100 MB. Permite ainda fazer gestão de clientes.

Foi eleito o *Project Libre* por uma questão de familiaridade, experiência na utilização deste *software* e porque as funcionalidades disponibilizadas pelo *software* foram suficientes para o desenvolvimento do projeto. Para este projeto só foi utilizada a funcionalidade de gerar diagramas de *Gantt* para obter uma melhor visualização e controlo temporal das atividades que decorreram no presente projeto.

Astah vs Visual Paradigm

O *software Astah* é uma aplicação que permite efetuar o desenho do *software* a implementar no nosso sistema. Neste projeto foi utilizado para desenhar o diagrama de casos de uso, o diagrama de classes e o diagrama de arquitetura. Diagramas estes essenciais para o correto e rápido desenvolvimento do projeto.

O *visual paradigm* é um *software* para desenho de diagramas de classes, arquitetura e até diagramas de *flow* e na sua versão paga permite fazer diagramas de *Gantt*, diagramas com animações, contém modelos avançados de gestão, como por exemplo, transições entre modelos e uma ferramenta de desenho de padrões.

Apesar de ambos os *softwares* na sua versão mais básica servirem para a análise e desenho do projeto, escolho o *Astah* pois é o *software* com que me encontro mais à vontade.

Firebase vs Flask

O *Firebase* é uma plataforma que permite criar o registo e *login* numa determinada aplicação, com uma panóplia de escolhas e funcionalidades elevada, permite ainda a utilização da *FireStore*, que é uma base de dados *NoSQL*. Para este projeto foi somente utilizada a funcionalidade de registo e *login*.

O *flask* é uma micro *framework* que não inclui o ORM, no entanto, contem outras funcionalidades como o roteamento de *url* e o processamento de *templates*, ou seja, transforma o código em *templates* para serem usados à posteriori. Esta micro *framework* tal como o *Firebase* permite realizar a autenticação e autorização, sendo uma aplicação simples, escalável e suporta a adição de extensões para aumentar a sua capacidade. O *Firebase* para além daquilo que já foi referido funciona como um *Backend As A Service* e tem a possibilidade de fazer autenticação através de diferentes plataformas, como a *Google*, o *Facebook*, a *Apple* e até através de smartphones.

Optei por escolher o *Firebase* pois é uma plataforma que permite implementar diferentes tipos de autenticação e no futuro poder-se-á utilizar outras funcionalidades como a *cloud Firestore*.

Qt Designer vs Tkinter

O *Qt Designer* é uma ferramenta que permite desenhar a *user interface* de uma forma mais prática e simples. Neste projeto foi utilizada a sua função primordial que é, como referido, o desenvolvimento da *user interface*, seguindo os *mockups* desenvolvidos através da ferramenta *Axure*.

O *Tkinter* funciona como uma interface para *Python* do *Tcl/Tk*. Tal como o *Qt Designer* permite construir UI, cada uma com o seu próprio *design* básico. Ambos os *softwares* são gratuitos, no entanto optei por escolher o *Qt designer* por ser um *software* com o qual não tinha tido qualquer tipo de contacto e porque permite desenhar a interface com *low code*.

Base de dados *SQL* vs *NoSQL*

Uma base de dados *SQL* é uma base de dados utilizada quando o cliente só necessita de guardar, consultar, remover e alterar dados. Este tipo de base de dados obriga a manter uma estrutura rígida, ou seja, é restritivo na tipologia, na ordem e no preenchimento dos

dados, pois todos os dados a serem inseridos nesta BD têm de ser iguais aos definidos na sua estrutura.

As bases de dados *NoSQL* são mais adequadas quando se pretende para além de dados guardar imagens, ficheiros ou gráficos. Esta base de dados é dinâmica, tornando-se mais flexível e é caracterizada por não ter um desenho da sua estrutura e por permitir que cada *row* da base de dados tenha atributos diferentes.

Neste projeto optei por uma base de dados *SQL* por causa da sua estrutura rígida pois permite o foco noutros aspetos do projeto sem que tenha de existir uma grande preocupação com mudanças acidentais à estrutura da base de dados. Elegendo assim a base de dados *SqlLite* em detrimento da *cloud Firestore* do *Firebase*.

3.2. Metodologia utilizada

A metodologia eleita para este projeto foi a metodologia ágil SCRUM. O SCRUM nasceu em 1993, na empresa *Easel Corporation,* após *Jeff Sutherland, John Scumniotales* e *Jeff McKenna* observarem, que os projetos que utilizavam equipas mais pequenas, e constituídas por elementos com diferentes especializações, alcançavam melhores resultados. Por fim, associaram este grau de eficácia à formação de *rugby* altamente eficaz *"Scrum"*, que era uma formação utilizada como forma de recomeçar um jogo de *rugby*.

O SCRUM é uma metodologia ágil para gestão de projetos, através de um conjunto de técnicas que permitem desenvolver projetos complexos e que necessitam de constante adaptação. O SCRUM foi criado para implementar o desenvolvimento de um projeto de forma rápida, iterativa, flexível e com capacidade de adaptação, assim irá permitir acrescentar valor ao produto ao longo de todas as interações do projeto.

Esta metodologia baseia-se nas seguintes características:

- Clientes tornam-se parte da equipa de desenvolvimento;
- Entregas frequentes de funcionalidades finalizadas;
- Plano para mitigação de riscos;
- Pequenas iterações de desenvolvimento do projeto, entre uma a quatro semanas;
- Divisão do projeto em pequenas tarefas passíveis de serem desenvolvidas nas iterações;
- Reuniões diárias sobre o estado do desenvolvimento;

- Colaboradores são encorajados a falar das suas dificuldades;
- Reuniões frequentes com os *stakeholders.*

Para este projeto foi utilizada esta metodologia, adaptada ao número de elementos da equipa, ao número de sprints, ao tamanho do projeto e ao tempo para a realização do mesmo.

- 4 -
DESENVOLVIMENTO DO PROJETO

Neste capítulo apresento todas as atividades realizadas no decorrer do projeto assim como as principais opções tomadas.

4.1. Organização do desenvolvimento

Para organizar o planeamento do projeto de forma a deixar o processo de aprendizagem o mais produtivo possível foram definidas as etapas, pela seguinte sequência:

- Plano de formação inicial
- Formação
- Documento de requisitos
- Análise e desenho
- Desenvolvimento das funcionalidades
- Resumo de seminários
- Elaboração do relatório final

Alocado ao plano de formação inicial estiveram 5 dias, após a conclusão deste foi iniciada a formação pelo período de 3 dias. Na sequência foi elaborado um documento de requisitos planeado para durar 6 dias e de seguida foi feita a análise e desenho por 5 dias. A fase de desenvolvimento de funcionalidades decorreu por um período de 34 dias, divididos por 3 sprints. Para assistir e realizar os respetivos resumos dos seminários foram alocados 3 dias. Por fim, para a redação e apresentação do relatório de estágio foram definidos 13 dias.

4.2. Formação

O conteúdo associado à formação foi numa etapa inicial a leitura de documentação sobre o funcionamento do *Firebase* e do *Qt Designer*. No decorrer do desenvolvimento e para

complementar a leitura da referida documentação foi dedicado tempo para assistir a tutoriais na internet e foi utilizada a metodologia *Hands-on-Learning* como forma de aprendizagem célere e contínua.

4.3. Documento de requisitos

No documento de requisitos, foram elaborados a gestão de riscos, identificação de *constrains* e a análise SWOT, foram definidos os critérios de aceitação e foi definida a organização geral das tarefas a realizar no projeto.

Gestão de riscos

No requisito de riscos, são enumerados os principais riscos a que o projeto poderá vir a ficar exposto.

Gestão de Riscos					
Risco	Grau de Impacto	Grau de Probabilidade	Consequências	Recursos Afetados	Estratégia de Ação
Mistura entre trabalho e estágio	5	4	Projeto inacabado, desmotivação, impossibilidade de finalizar o estágio.	Projeto	Realizar uma separação clara entre horas dedicadas para o trabalho normal diário e o tempo alocado para o desenvolvimento do estágio
Discordância com os stakeholders	5	2	Insucesso do projeto, alterações ao produto/projeto e atraso de prazos.	Projeto, Produto	Efetuar uma reunião com as partes interessadas
Problemas de hardware	5	2	Atraso no projeto, perda do desenvolvimento e progresso realizado e probabilidade de perda de equipamento.	Projeto, Produto e Recursos	Reparar ou adquirir novo hardware
Desmotivação	4	1	Atraso no projeto, atraso na entrega, possibilidade de não existir entrega.	Projeto	Fazer uma revisão ao projeto e realizar entregas mais frequentes aos stakeholders de forma a acrescentar mais valor ao produto e aumentar a interação com os stakeholders
Falta de skills	3	3	Atraso no projeto, projeto incompleto, com falhas.	Projeto e Produto	Solicitar ajuda ao supervisor, dedicar mais tempo para o desenvolvimento do projeto
Problemas de software	3	2	Atraso no projeto e complicações no desenvolvimento do produto.	Projeto e Produto	Encontrar softwares alternativos, executar testes ao software
Covid19	3	2	Atraso no projeto, impossibilidade de desenvolvimento por tempo indeterminado.	Projeto	Revisão do planeamento e tarefas. Escolha das tarefas prioritárias para o projeto

Figura 1- Gestão de riscos

Identificação de *Constraints*

- Não existir ninguém especializado na área de desenvolvimento de *software* na entidade acolhedora do estágio.
- Pouca disponibilidade dos utilizadores finais para alocarem tempo para a realização de reuniões necessárias para o desenvolvimento do projeto.
- "Operação Gaia", crise sismológica na ilha de São Jorge.

Análise SWOT

Esta análise permite refletir e perceber o ambiente interno e externo da empresa. Da análise do ambiente interno é possível concluir as Forças (S) e as Fraquezas (W). Da análise do ambiente externo são retiradas as Oportunidades (O) e as Ameaças (T) para o projeto.

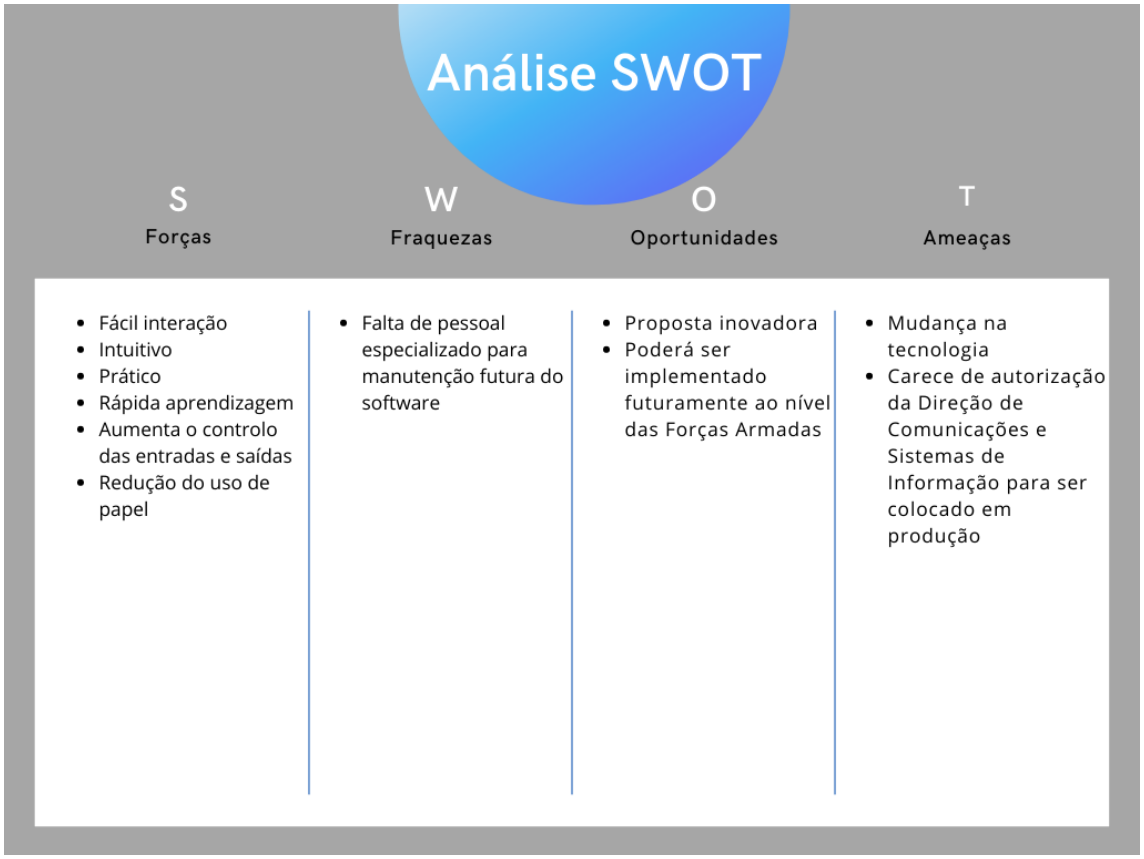


Figura 2- Análise SWOT

Processo de desenvolvimento do projeto

Durante o desenvolvimento deste projeto pretende-se dividir o tempo pelas seguintes tarefas:

- 60% para desenvolvimento da aplicação;
- 40% para testes, documentação e planeamento da próxima etapa de desenvolvimento.

Critérios de aceitação

Como critérios de aceitação foi definido que tudo o que será elaborado no decorrer deste projeto terá de contribuir com algum valor para o produto a desenvolver e para os *stakeholders*.

Os principais critérios de aceitação para o conjunto das *User Stories* (US) desenvolvidas são:

- Qualidade da implementação (isento de erros, bugs e com documentação correta);
- Corresponder às necessidades dos *stakeholders*;
- Traz valor para os *stakeholders*;
- Aprovação nos testes definidos.

É considerado que, caso os critérios em cima enumerados forem todos cumpridos para cada US, os critérios de aceitação serão satisfeitos.

Organização geral do projeto

Nesta secção vai ser mostrada uma visão muito genérica sobre o *backlog* e as tarefas associadas a cada entrega de relatórios.

- **Levantamento de requisitos**
  - Reuniões síncronas com os *stakeholders;*
  - Pesquisa de informação sobre o domínio do negócio;
  - Leitura de documentação sobre o desenvolvimento da aplicação.
- **Desenho do sistema a implementar**
  - Diagrama de casos de uso;
  - Diagrama da arquitetura;
  - Diagrama de classes;
  - Diagrama da base de dados;
  - *Mockups* da aplicação (Anexo 9);
  - *Backlog* geral da aplicação (Anexo 3).
- **Implementação**
  - Desenvolvimento da base de dados;
  - Login na plataforma;
  - Registo na plataforma;
  - Registo de entradas e saídas de pessoas externas à empresa;
  - Registo de entradas e saídas de viaturas da empresa (pelos condutores);
  - Registo da visita à entidade;
  - Gestão das viaturas;
  - Gestão de colaboradores;
  - Criação de *logs.*
- **Redação de relatórios**
  - **Plano inicial de formação**
  - **Relatório inicial (Documento de requisitos)**
    - *Project view*;
    - *Release planning schedule* (Diagrama de *Gantt*);
    - Gestão de riscos;
    - Definição de *constraints*;
    - Levantamento de requisitos;
    - Desenho e análise do sistema.

- **Relatório estruturado**
  - Estrutura do relatório final.
- **Entrega provisória**
  - Primeira versão do relatório final.
- **Relatório final**
  - Relatório final do projeto.

4.4. Análise e desenho

Após o levantamento de requisitos vem a fase da análise dos mesmos pedidos para o sistema e o seu desenho. O primeiro desenho a ser efetuado foi o diagrama de arquitetura, nas suas duas versões, arquitetura por camadas e arquitetura cliente-servidor, que veio desde o início estabelecer como o sistema iria ser implementado.

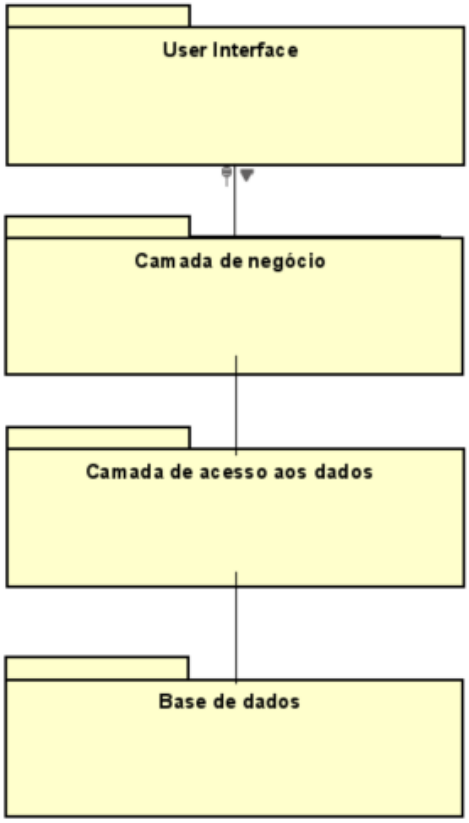


Figura 3- Diagrama de arquitetura versão inicial

A arquitetura eleita para este projeto foi a arquitetura cliente-servidor construída por camadas. Esta arquitetura foi escolhida pois existe uma necessidade de todos os colaboradores terem acesso, em simultâneo ou não, à aplicação, estando o lado do servidor alojado em servidor próprio da entidade de acolhimento e o lado do cliente na máquina de cada utilizador. Esta arquitetura é por definição baseada em camadas, neste caso específico três camadas: a camada do cliente, a camada da lógica de negócio e a

camada de dados, em que cada camada tem um determinado objetivo, com uma interface bem definida que a camada superior utiliza e a comunicação é feita de forma unilateral.

Nas figuras em baixo podemos verificar o diagrama de arquitetura desenvolvido na sua versão final.

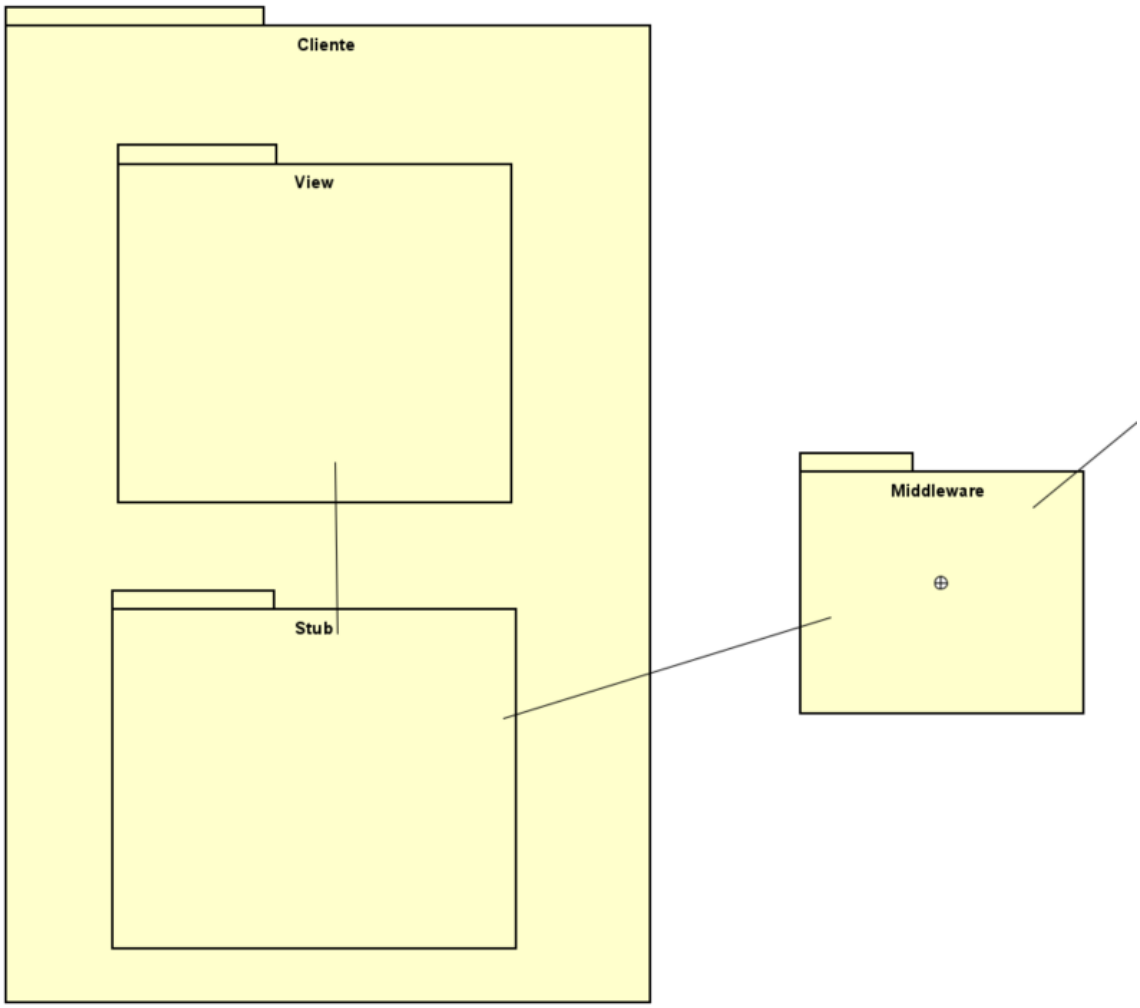


Figura 4- Diagrama de arquitetura versão final lado do cliente

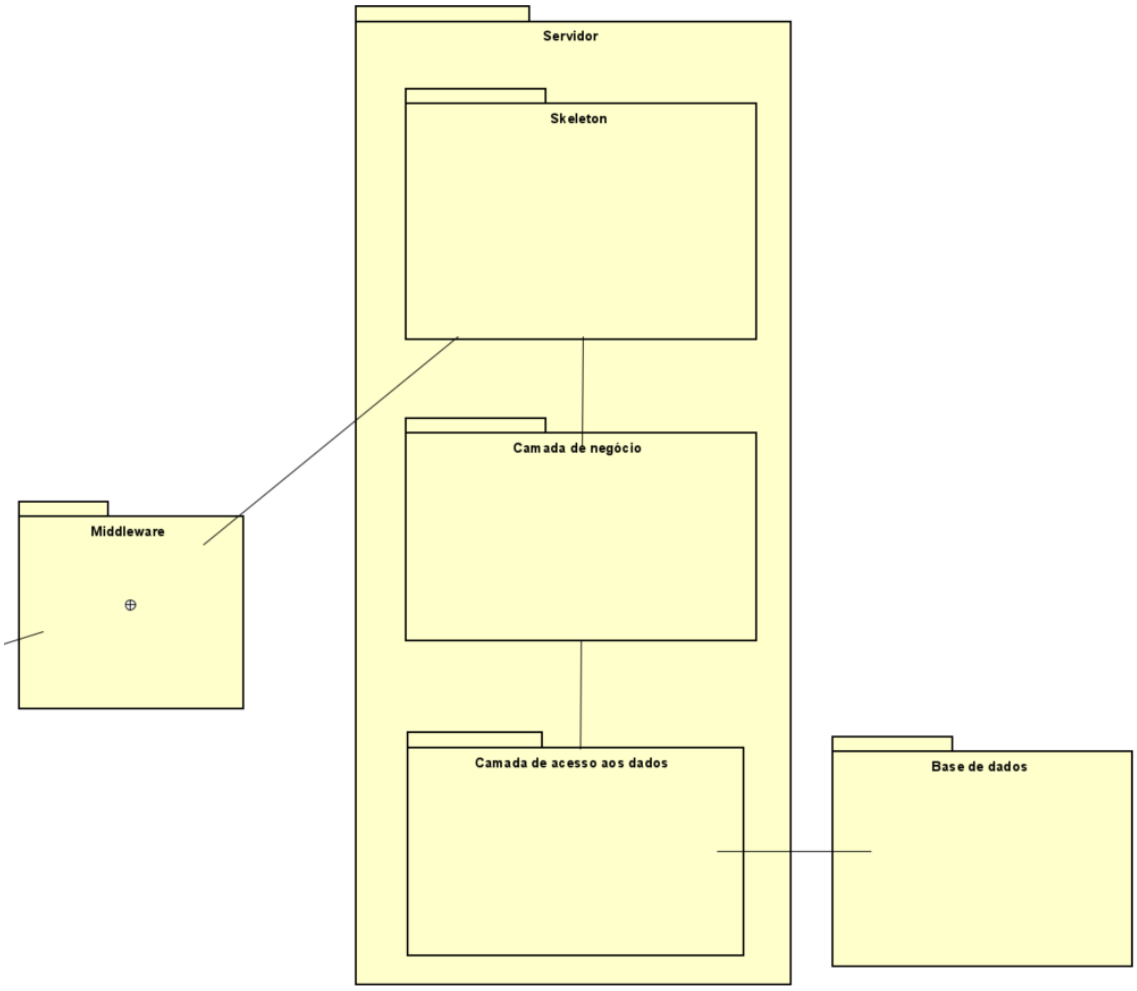


Figura 5- Diagrama de arquitetura versão final lado do servidor

As vantagens da utilização das camadas são o controlo de complexidade do sistema, a abstração do sistema e a realização de alterações a partes de uma camada sem ser necessário alterar o sistema inteiro, desde que as outras interfaces se mantenham iguais.

De seguida, foi produzido o diagrama de casos de uso que vem mostrar ao cliente as funcionalidades do sistema a desenvolver, o tipo de atores que vão utilizar o sistema e com que aplicações externas o sistema a desenvolver vai comunicar.

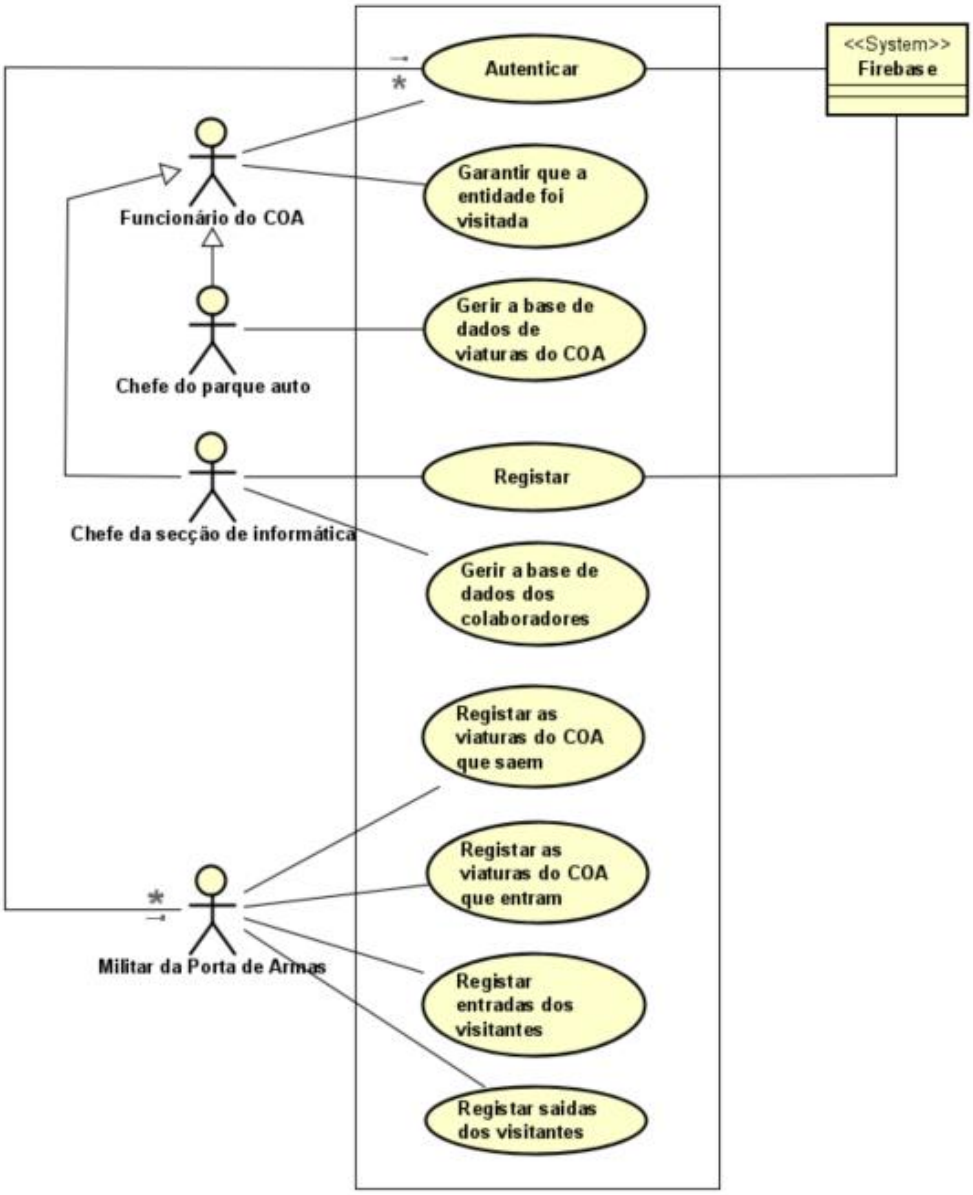


Figura 6- Diagrama de casos de uso

Na próxima etapa foi desenvolvido o diagrama da base dados, de forma a perceber quais as entidades necessárias, os seus atributos e quais dessas entidades seriam candidatas a classes persistentes do sistema.

Na escolha da base de dados existiram alguns aspetos que necessitei ter em conta, como a escalabilidade, a estrutura e a topologia de informação a ser guardada. Todas as tabelas estão normalizadas na 3º Forma Normal, ou seja, todos os atributos não pertencentes à chave primária, dependem única e exclusivamente dela.

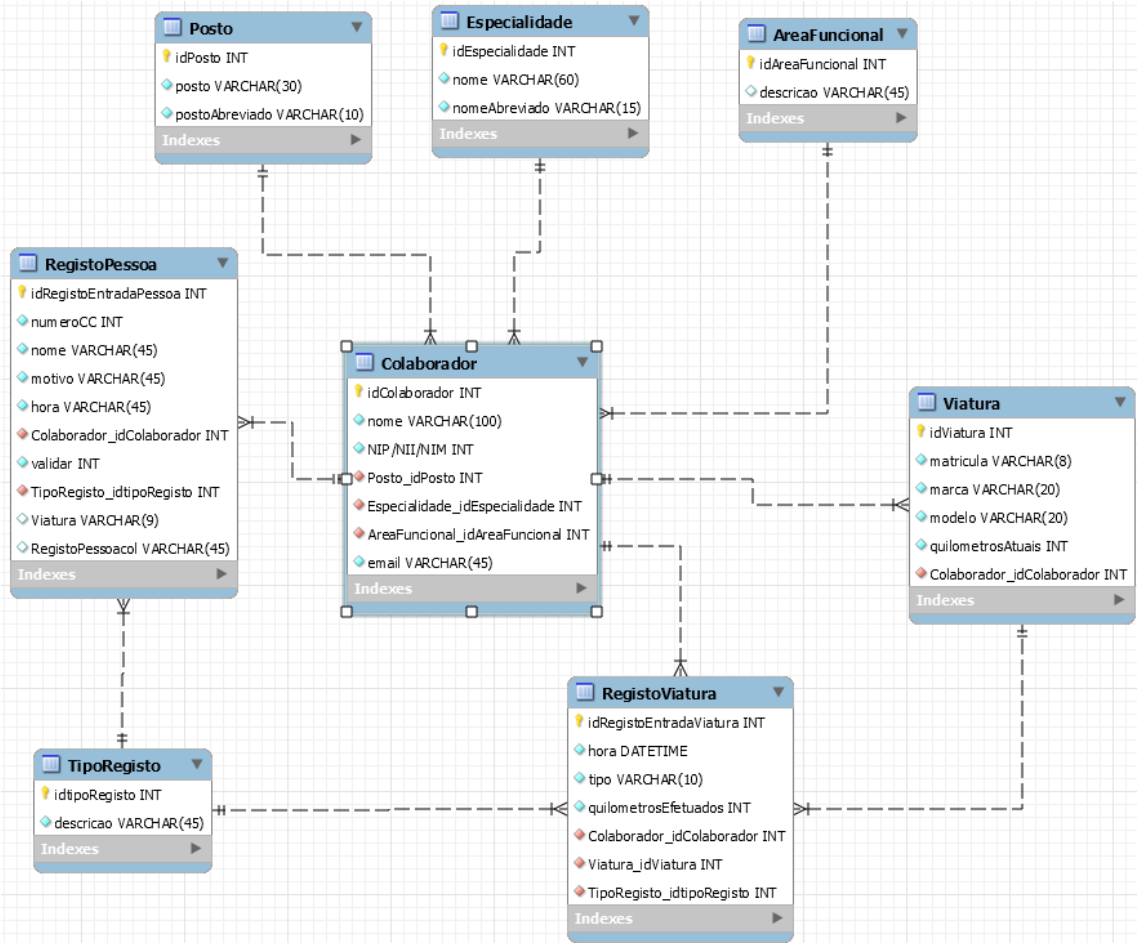


Figura 7- Diagrama da base de dados

O colaborador irá ser a peça central da aplicação, pois é através deste que irá ser feita a interação com a aplicação e é a tabela que tem mais ligações na base de dados. O colaborador é caracterizado pelo posto, nome, NIP/NII/NIM, especialidade e a sua área funcional.

De seguida foram desenvolvidos os *mockups* da aplicação que podem ser consultados no anexo 9 do presente relatório.

Por fim, foi desenvolvido o diagrama de classes, que irá permitir modelar as classes do sistema e as ligações entre elas, de forma a minimizar o *coupling* e a aumentar a coesão das classes. O diagrama de classes facilita o trabalho na etapa de desenvolvimento pois o sistema já se encontra todo estruturado sendo apenas necessário transformar o diagrama em código.

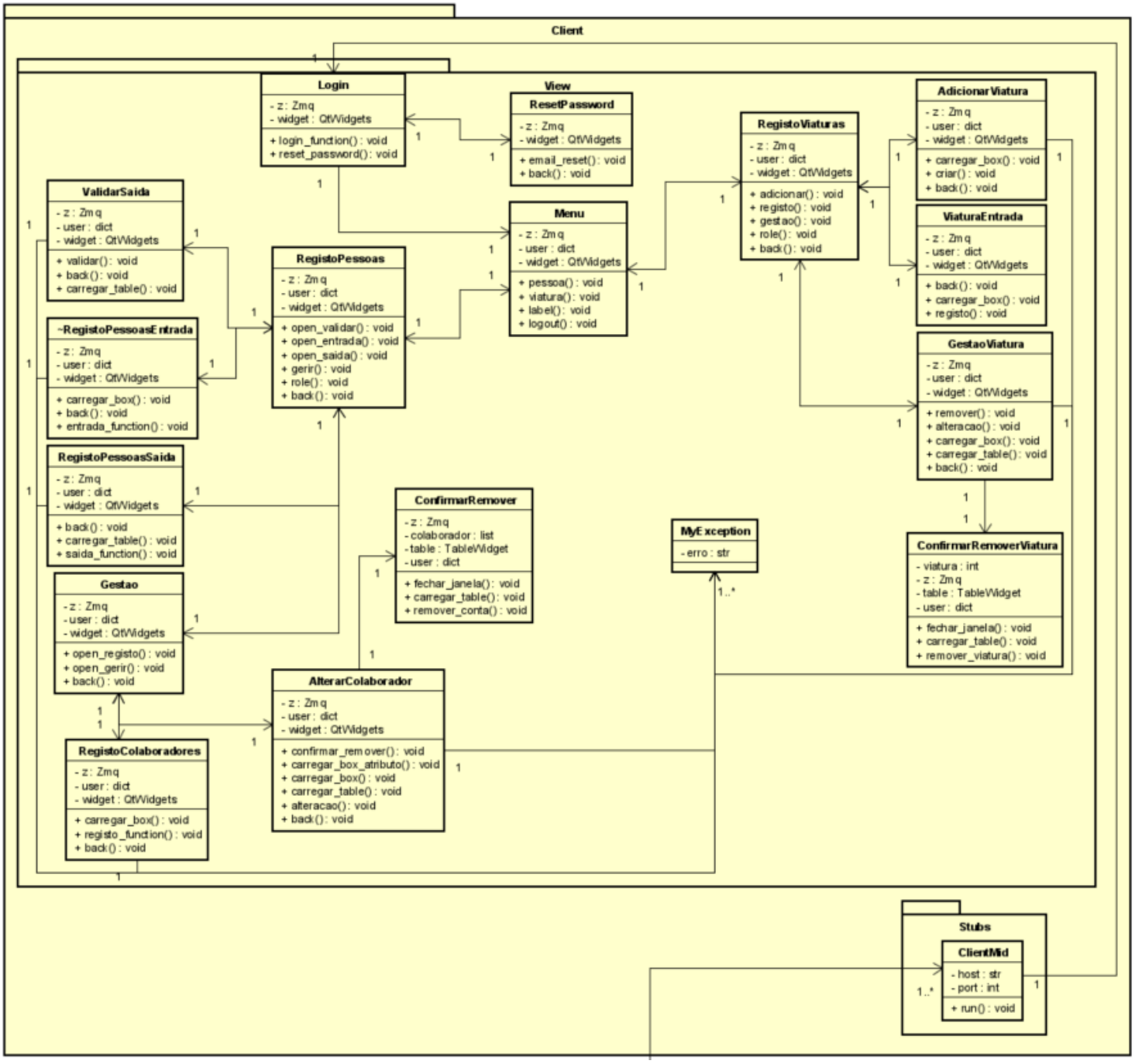


Figura 8- Diagrama de classes lado do cliente

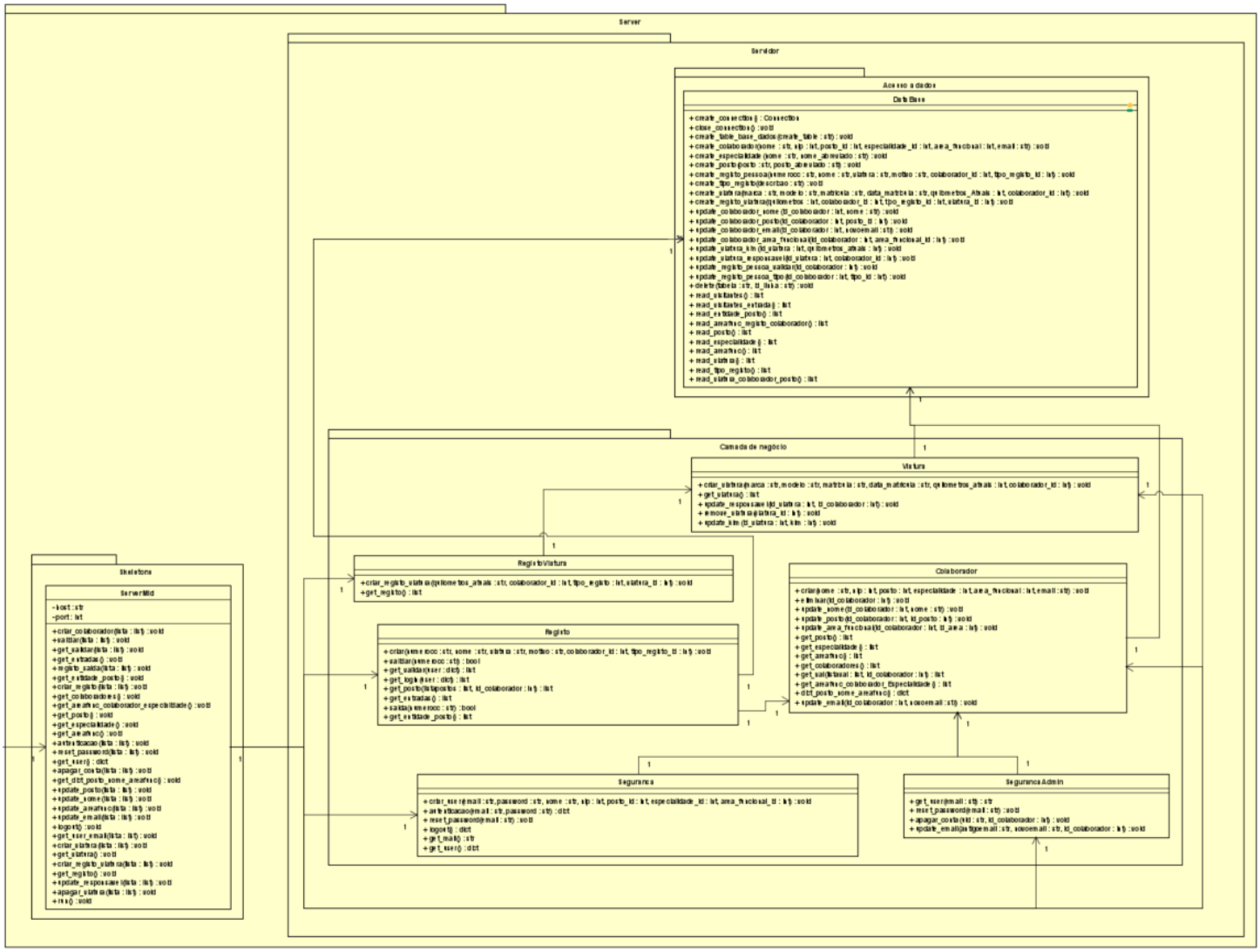


Figura 9- Diagrama de classes lado do servidor

4.5. Implementação

4.5.1. *Sprints* de desenvolvimento

Como já identificado anteriormente, o projeto foi dividido em 3 *sprints* de desenvolvimento, cada *sprint* com o seu backlog, podendo este ser consultado nos anexos 4, 5 e 6 respetivamente.

1º *Sprint* (7 a 28 de abril)

O primeiro *sprint* consistiu no desenvolvimento da base de dados e no desenvolvimento das funcionalidades do módulo de gestão para os visitantes:

- Registar entrada de visitantes;
- Registar saída de visitantes;

- Verificar se a entidade recebeu o visitante;
- Gerir a base de dados dos funcionários.

No início do desenvolvimento foi definido que os colaboradores iriam ser todos os funcionários do COA e os operadores todos os militares e civis que irão utilizar esta aplicação, independentemente de serem ou não funcionários do COA. Isto, pois, os militares responsáveis por gerir as entradas e saídas de visitantes não pertencem ao COA. De seguida foi modelada a base de dados, utilizando scripts, segundo o diagrama da base de dados desenvolvido na fase de análise e desenho.

```
posto_table = """ CREATE TABLE IF NOT EXISTS Posto (
                                    id integer PRIMARY KEY AUTOINCREMENT,
                                    posto varchar NOT NULL,
                                    postoAbreviado varchar NOT NULL
                                ); """
```

Figura 10- Exemplo de script de modelação da BD

Após o desenvolvimento da base de dados, procedeu-se à criação de *querys* para realizar as operações CRUD sobre a base de dados, funcionando assim como o ORM.

Depois do ORM ser testado, chegou a fase de implementar as funcionalidades solicitadas. Para isso foi necessário desenvolver as classes Colaborador e Registo na camada de negócio. De seguida, foram realizados os respetivos testes e deu-se início ao desenvolvimento da UI associada às respetivas funcionalidades, criando a classe Ui. A timeline de desenvolvimento pode ser consultada na imagem abaixo:

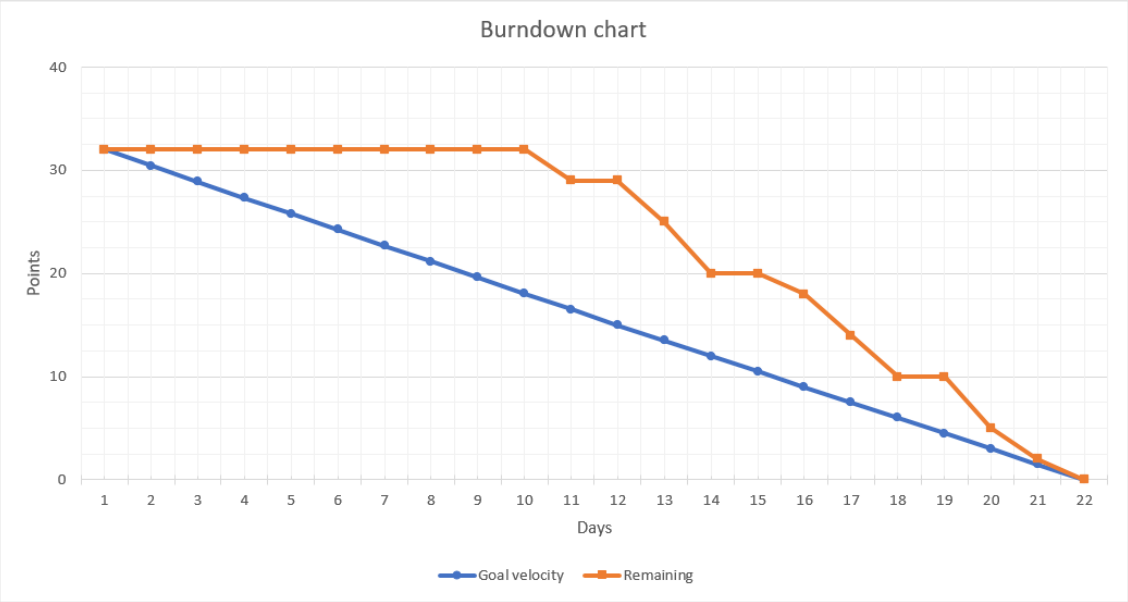


Figura 11- Burndown chart 1º sprint

2º *Sprint* (30 de abril a 8 de maio)

O segundo *sprint* consistiu no desenvolvimento dos *logs*, no início da migração da arquitetura para cliente-servidor e no desenvolvimento de funcionalidades do sistema:

- Registar colaboradores na base de dados;
- Autenticar operadores na aplicação.

No início deste *sprint*, foram desenvolvidas as classes Segurança e SegurançaAdmin na camada de negócio, classes responsáveis por fazer a comunicação com a plataforma *Firebase.*

De seguida foi desenvolvida a classe Log, responsável pela criação de um registo de todos os pedidos feitos ao servidor. Após a realização dos testes ao código desenvolvido neste sprint, foi atualizada a classe Ui com a *User Interface* das funcionalidades. Por fim, foi desenvolvida toda a estrutura do cliente-servidor, assim como a classe Zmq responsável pela comunicação. A timeline de desenvolvimento pode ser consultada na imagem abaixo:

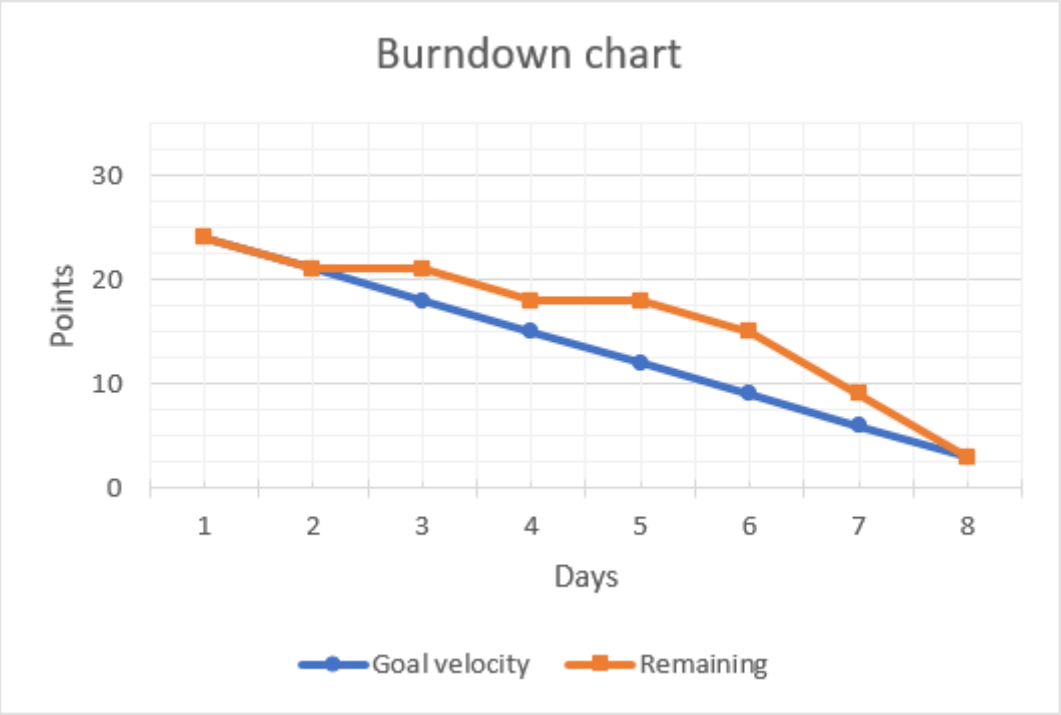


Figura 12- Burndown chart 2º sprint

3º *Sprint (13 a 26 de maio)*

O último *sprint* consistiu na finalização do sistema cliente-servidor, da refatorização do código já desenvolvido, na correção dos diagramas desenvolvidos na etapa de análise e desenho e no desenvolvimento das funcionalidades para as viaturas:

- Registar as viaturas de serviço que saem do COA;
- Registar as viaturas de serviço que entram no COA;
- Gerir a base de dados das viaturas.

No começo deste sprint, foram desenvolvidas as funcionalidades planeadas, para isso foram criadas as classes Viatura e RegistoViatura na camada de negócio. De seguida, foi finalizada a migração para a arquitetura cliente-servidor, corrigindo eventuais *bugs*. Posteriormente foi efetuada a refatorização do código do lado do servidor e do lado do cliente, substituindo a classe Ui, por 17 classes, cada uma delas para um ecrã especifico da aplicação. Por fim, foram realizadas as eventuais correções aos diagramas já desenvolvidos anteriormente. O código desenvolvido pode ser consultado no anexo 10. A timeline de desenvolvimento pode ser consultada na imagem abaixo:

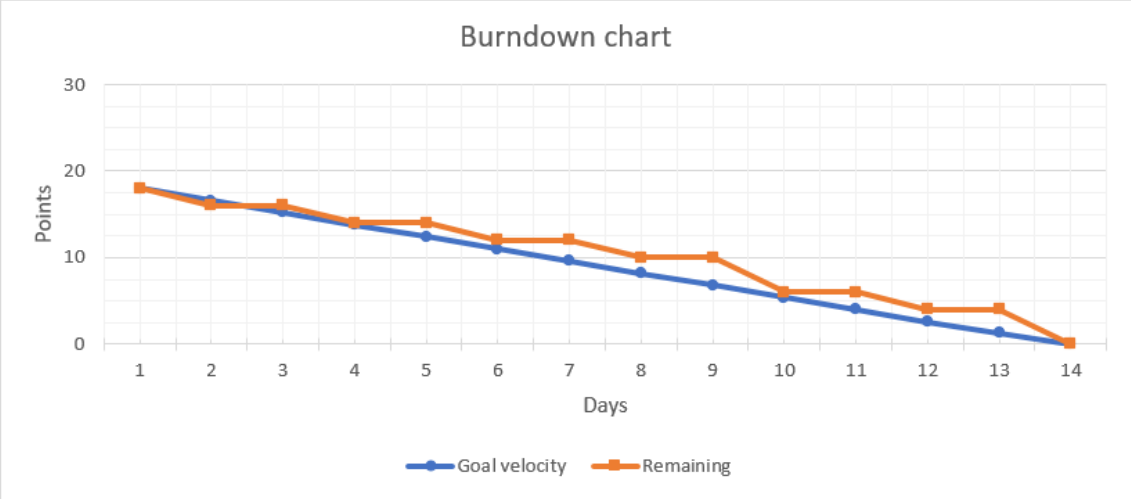


Figura 13- Burndown chart 3º sprint

4.5.2. Decisões no desenvolvimento

Não utilização de *frameworks*

No início do desenvolvimento e após existir alguma ponderação na utilização da *framework django*, optou-se por não utilizar nenhuma *framework* para obter mais liberdade na construção do sistema e pelo motivo que o *django* não tem integrado na

framework o desenvolvimento para *desktop*. Sendo assim, o sistema teria de ser desenvolvido primeiro para a *web* e depois serem utilizados outros módulos para converter a aplicação para *desktop*.

Desenvolvimento da arquitetura

Na criação da aplicação optou-se por desenvolver em primeiro lugar o sistema por camadas e após este estar integralmente testado, evoluí-lo para a arquitetura cliente-servidor. Esta opção deveu-se à existência de um maior controlo e perceção nos erros que decorreram da implementação.

Comunicação cliente-servidor

Para realizar a comunicação entre o cliente e o servidor, foi utilizado o *zeromq* no *middleware*. Esta comunicação, tanto nos pedidos como nas respostas é feita sempre através de *strings*. Quando é feito um pedido ao servidor, a função *send_msg* irá colocar todos os atributos dentro de uma lista e irá converter essa lista numa *string*. O servidor quando recebe o pedido utiliza a função *decode*, para descodificar a mensagem recebida para o tipo de dados anterior à conversão em *string*, neste caso passa para uma lista. Dependendo do caso, pode-se aceder diretamente às posições da lista ou ter-se-á que realizar um novo *decode*.

```
class Zmq:
    def __init__(self, host: str, port: int, tipo: str) -> None:
        """
        Função que estabelece a comunicação no host e port definidos
        :param host: adress da maquina que vai comunicar
        :param port: Port do host em que irá ser feita a comunicação
        :param tipo: tipo de comunicação Request/Reply
        """
        context = zmq.Context()
        p = "tcp://" + host + ":" + str(port)
        if tipo == "REQ":
            self.s = context.socket(zmq.REQ)
            self.s.connect(p)
        else:
            self.s = context.socket(zmq.REP)
            self.s.bind(p)

    def send_msg(self, *msg: str) -> None:...

    def recv_msg(self) -> str:...

    @staticmethod
    def decode(msg: str) -> str:...
```

Figura 14- Código fonte do middleware

RGPD *vs* Legislação de acesso a unidades militares

A legislação presente no RGPD e as normas para acesso a unidades, estabelecimentos ou órgãos militares complementam-se no caso de dados tratados para interesse público como podemos verificar no Capítulo 5, artigo 1. do RGPD:

"1. Os dados pessoais são: (…)

e) Conservados de uma forma que permita a identificação dos titulares dos dados apenas durante o período necessário para as finalidades para as quais são tratados; os dados pessoais podem ser conservados durante períodos mais longos, desde que sejam tratados exclusivamente para fins de arquivo de interesse público (…)"

Entende-se que para qualquer pessoa aceder a uma unidade militar, por motivos de segurança nacional e manutenção da soberania da nação todas as pessoas devem ser identificadas de forma inequívoca e o seu acesso deve ficar arquivado por tempo indeterminado, devido à defesa nacional ser considerada como "interesse público" e é reforçado pela seguinte norma, presente no Capítulo 5, secção I, 5-8, ponto (b).1.c) alínea 3 das Instruções de Segurança Militar:

"Deve ser mantido um registo detalhado e completo sobre os visitantes."

Alteração à tipologia de dados nas tabelas

No início do projeto o número de Cartão de Cidadão foi idealizado como um *int*, uma vez que os cartões portugueses são compostos somente por números, no entanto após uma reflexão mais profunda foi percecionado que podem existir visitantes estrangeiros, e esses visitantes podem ter letras no seu cartão de identificação. Após isso o tipo do número de Cartão de Cidadão foi alterado para *string* de forma a abranger todos estes tipos de cartões de identificação.

Colocação de imagens na *User Interface*

O *Qt Designer* para a colocação de imagens na *interface* obriga a defini-las em dois sítios diferentes. Diretamente no *Qt Designer* e para isso cria-se uma *label* e define-se um *resource* e o *path* da imagem dentro dessa *label*.

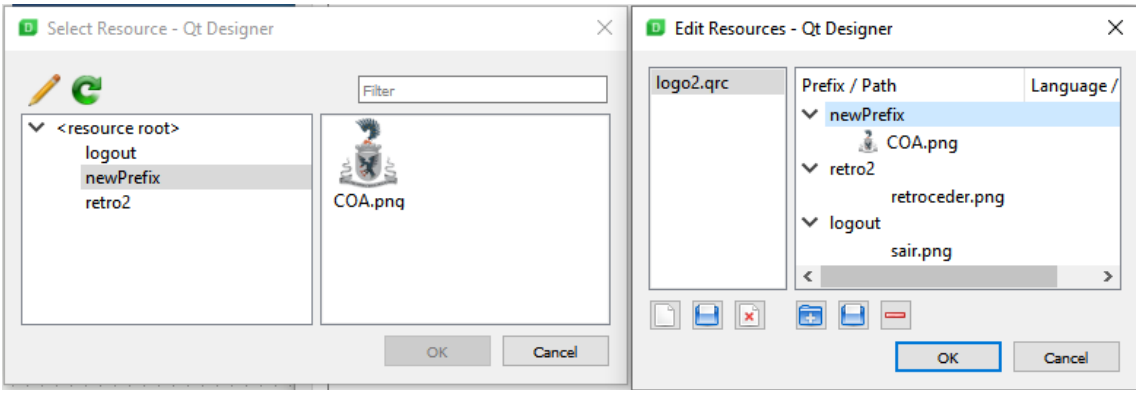


Figura 15- Adicionar uma imagem no Qt Designer

No *Pycharm*, é chamado o nome da label definida no *Qt Designer* e é chamado o método *Pixmap* para renderizar uma imagem e passamos o *path* da imagem escolhida.

```
self.imagem.setPixmap(QPixmap(IMG))
self.imgret.setPixmap(QPixmap(IMGRETROCEDER))
```

Figura 16- Adicionar uma imagem no Pycharm

Extração dos *id's* das *views* para o servidor, dos colaboradores

Numa primeira instância do desenvolvimento os *id's* eram extraídos através do *currentIndex* das *comboBox*, no entanto, se alguma linha da base de dados fosse removida os *id's* do *currentIndex* das *comboBox* iriam ser diferentes dos *id's* presentes na base de dados. Para resolver esse problema foi passado o *currentText* da *comboBox* como parâmetro para a lógica de negócio do servidor e lá foi feita a comparação entre o *currentText* e os atributos posto + nome dos colaboradores para obter o *index* do colaborador que resulta dessa comparação. Isto é possível pois não existem dois colaboradores com o mesmo posto e nome na base de dados.

Utilização de dicionários para implementar *comboBox* dependentes

Para carregar uma *comboBox* que esteja dependente de outra foi utilizado um dicionário, em que as chaves, são os valores da *comboBox* mãe, e os valores são os valores que irão ser carregados na *comboBox* filha. Não foi utilizado o *QtreeWidget* pois iria inserir uma complexidade de implementação desnecessária ao desenvolvimento da aplicação.

Login

O *login* é realizado através da plataforma *Firebase*. Após o *login* ser efetuado, o servidor no *skeleton* vai guardar num dicionário a informação de todos os utilizadores com o login efetuado na aplicação. Os *idtokens* serão guardados como chave e os emails como valor. Estes emails são posteriormente utilizados nos *logs*.

Identificação de clientes no servidor

Para identificar um cliente no servidor é enviado o *idtoken* gerado pelo *Firebase* para o servidor em cada pedido feito pelo cliente. Em todos os métodos do servidor, menos na autenticação e *reset_password*, é verificado se o utilizador se encontra com o *login* efetuado através de uma comparação do seu *idtoken* com o dicionário de *idtokens* do servidor.

Logout

Para realizar o *logout*, basta no menu clicar na figura para fazer *logout* e automaticamente o utilizador passa para o ecrã de *login* e é removido o seu *idtoken* do dicionário no servidor.

Logs

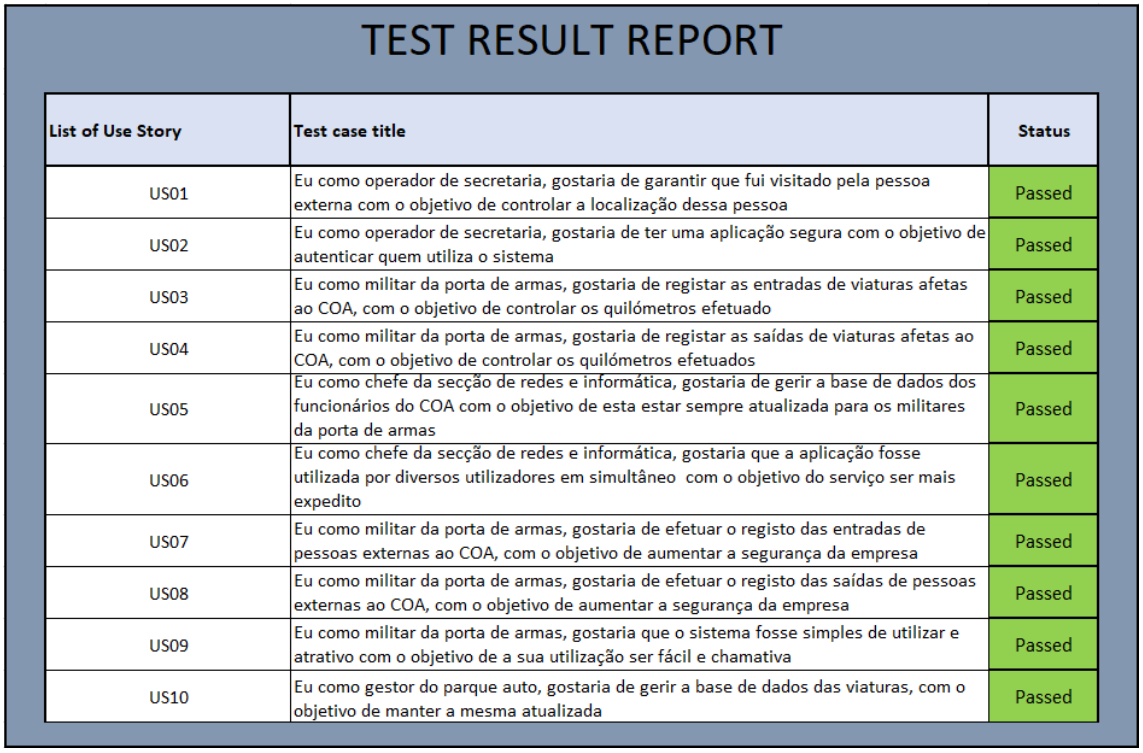
Com o objetivo de deixar registados no sistema os acessos feitos à base de dados, foi desenvolvida uma classe para criar *logs* de todos os pedidos feitos ao servidor. Esta classe irá escrever num ficheiro *txt* o email, o tipo de pedido feito ao servidor e a hora.

Validações são efetuadas nas *views*

De forma a minimizar os pedidos feitos ao servidor, foram colocadas as validações dos campos a serem preenchidos pelo utilizador, diretamente nas *views*, assim como o preenchimento dos campos obrigatórios.

Testes à aplicação

No requisito de testes foram realizados testes unitários, testes de integração e testes de aceitação. Os testes unitários são realizados para testar todas as funções do sistema a fim de garantir que funcionam de forma correta. Os testes de integração verificam que quando se integra um método com o restante sistema, este tem o comportamento esperado. Os testes unitários e de integração foram realizados de forma manual. Os testes de aceitação foram realizados por um grupo de colaboradores da instituição de acolhimento e visaram testar se a aplicação está de acordo com o que foi solicitado.



TEST RESULT REPORT

List of Use Story	Test case title	Status
US01	Eu como operador de secretaria, gostaria de garantir que fui visitado pela pessoa externa com o objetivo de controlar a localização dessa pessoa	Passed
US02	Eu como operador de secretaria, gostaria de ter uma aplicação segura com o objetivo de autenticar quem utiliza o sistema	Passed
US03	Eu como militar da porta de armas, gostaria de registar as entradas de viaturas afetas ao COA, com o objetivo de controlar os quilómetros efetuado	Passed
US04	Eu como militar da porta de armas, gostaria de registar as saídas de viaturas afetas ao COA, com o objetivo de controlar os quilómetros efetuados	Passed
US05	Eu como chefe da secção de redes e informática, gostaria de gerir a base de dados dos funcionários do COA com o objetivo de esta estar sempre atualizada para os militares da porta de armas	Passed
US06	Eu como chefe da secção de redes e informática, gostaria que a aplicação fosse utilizada por diversos utilizadores em simultâneo com o objetivo do serviço ser mais expedito	Passed
US07	Eu como militar da porta de armas, gostaria de efetuar o registo das entradas de pessoas externas ao COA, com o objetivo de aumentar a segurança da empresa	Passed
US08	Eu como militar da porta de armas, gostaria de efetuar o registo das saídas de pessoas externas ao COA, com o objetivo de aumentar a segurança da empresa	Passed
US09	Eu como militar da porta de armas, gostaria que o sistema fosse simples de utilizar e atrativo com o objetivo de a sua utilização ser fácil e chamativa	Passed
US10	Eu como gestor do parque auto, gostaria de gerir a base de dados das viaturas, com o objetivo de manter a mesma atualizada	Passed

Figura 17- Test result report

Refatorização do código

Com a mudança da arquitetura para cliente-servidor existiu a necessidade de modificar os *imports* usados para se resolver o erro de ciclos de *imports*.

Com a divisão do ficheiro da *user interface* por vários ficheiros existiu a necessidade de rever os *imports* para se resolver o erro de ciclos de *imports*.

Para diminuir a complexidade do código no servidor, foi criado um dicionário, em que as chaves são uma *string*, com o pedido do cliente e o valor é o método a ser chamado.

```
while True:
    msg = self.z.recv_msg()
    dados = self.z.decode(msg)
    if dados[-1] == "criar colaborador":
        self.criar_colaborador(dados)
    elif dados[-1] == "validar registo":
        self.validar(dados)
    elif dados[-1] == "get_validar":
        self.get_validar(dados)
    elif dados[-1] == "reset password":
        self.reset_password(dados)
    elif dados[-1] == "get_entradas":
        self.get_entradas(dados)
    elif dados[-1] == "registo saida":
        self.registo_saida(dados)
    elif dados[-1] == "get_entidade_posto":
        self.get_entidade_posto(dados)
    elif dados[-1] == "registo entrada":
        self.criar_registo(dados)
    elif dados[-1] == "get_colaboradores":
        self.get_colaboradores(dados)
    elif dados[-1] == "get_areafunc_colaborador_especialidade":
        self.get_areafunc_colaborador_especialidade(dados)
    elif dados[-1] == "get_posto":
        self.get_posto(dados)
    elif dados[-1] == "get_especialidade":
        self.get_especialidade(dados)
    elif dados[-1] == "get_areafunc":
        self.get_areafunc(dados)
    elif dados[-1] == "autenticacao":
        self.autenticacao(dados)
    elif dados[-1] == "get_user":
        self.get_user(dados)
    elif dados[-1] == "apagar_conta":
        self.apagar_conta(dados)
    elif dados[-1] == "get_dict_posto_nome_areafunc":
        self.get_dict_posto_nome_areafunc(dados)
    elif dados[-1] == "update_posto":
        self.update_posto(dados)
    elif dados[-1] == "update_nome":
```

Figura 18- Refatorização no server_mid antes

```
while True:
    msg = self.z.recv_msg()
    dados = self.z.decode(msg)
    if dados[-1] not in self.dict.keys():
        print("Erro funcao nao encontrada")
    else:
        self.dict[dados[-1]](dados)
```

Figura 19- Refatorização no server_mid depois

Refatorização do design da user interface

No início, o *design* da *user interface* foi construído sob a forma de um protótipo para testar as funcionalidades com uma *user interface*. Após a apresentação intermédia do presente estágio, foi realizada uma profunda alteração ao *design* do projeto como se pode verificar em baixo.

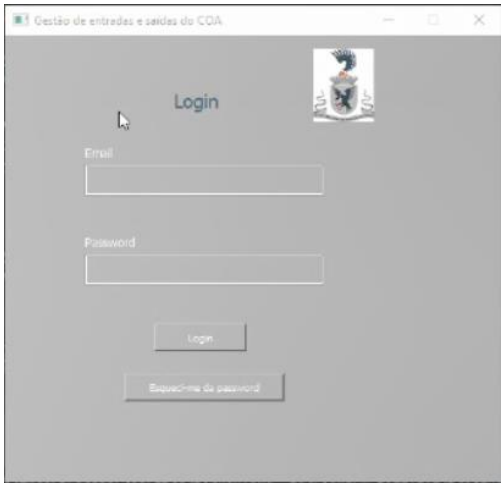


Figura 21- Ecrã de login inicialmente proposto

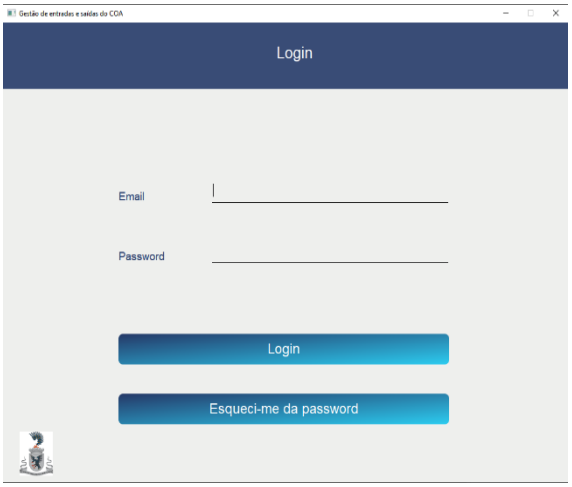


Figura 20- Ecrã de login atual

Dificuldades

A principal dificuldade encontrada no desenvolvimento da aplicação foi aprender a trabalhar com os erros da API do *Qt Designer*. Isto porque independentemente do erro gerado, a API dá sempre um erro de *overflow* da pilha dos *widgets*, devido às threads utilizadas pelo *Pycharm*.

```
Process finished with exit code -1073740791 (0xC0000409)
```

Figura 22- Erro de overflow da pilha dos widgets

- 5 -
5. SEMINÁRIOS FREQUENTADOS

Neste capítulo vou apresentar uma breve reflexão critica sobre os seminários frequentados no âmbito do estágio.

Orientações para a escrita do relatório e apresentação oral do projeto de estágio (parte I) - Doutora Isaura Ribeiro

O principal objetivo deste seminário foi demonstrar como se deve construir um *powerpoint* de forma correta e como se deve proceder à apresentação do mesmo.

A oradora começou por explicar os três componentes principais de uma apresentação: o conteúdo, a performance e o design. No caso do conteúdo este deve de ser equilibrado, para que os recursos visuais apresentados reforcem e transmitam mais facilmente a mensagem pretendida. No caso de só existir preocupação com o design podemos cair no erro de ter um discurso vazio e incoerente. Por fim, se a preocupação for só com a performance podemos realizar uma apresentação pobre, com conteúdo superficial. Portanto, quando planeamos uma apresentação devemos tentar fazer um equilíbrio entre os três aspetos acima referidos. Em suma, o design é um recurso, o conteúdo é a mensagem e a forma de nos expressarmos é fundamental para captar a atenção da plateia.

Após esta breve introdução a oradora explicou a importância de saber comunicar em público, independentemente do tipo de ambiente que estamos inseridos e que uma comunicação não verbal correta fortalece ainda mais a mensagem que tentamos passar. Os objetivos de uma apresentação podem passam por informar, convencer, promover a reflexão ou impressionar o público e o facto de não sabermos esses objetivos poderá produzir conteúdo de baixa qualidade. Outro aspeto importante a ter conta aquando de uma apresentação é o tempo disponível para a mesma, por regra geral, quanto menor for o tempo para realizar uma apresentação mais cuidado é necessário ter na elaboração da mesma para termos a certeza que conseguimos transmitir de forma correta a ideia pretendida. Nunca devemos exceder o tempo e devemos de planear a apresentação para

esta não ser demasiado curta. A metodologia usada para passar a mensagem também deve ser enquadrada com o objetivo da apresentação e esta poderá ser através de *storytelling*, desafio *vs* solução, suspense ou outras.

De seguida a oradora deixou alguns conselhos para a apresentação, como por exemplo, treinar a apresentação em voz alta com o objetivo de compreender melhor a voz e o ritmo da dicção. Conhecer o público a quem vamos apresentar é essencial pois permite estabelecer uma maior ligação com a plateia e adaptar a maneira de passar a mensagem consoante o grau de escolaridade, a experiência sobre o tema, entre outros aspetos. O local da apresentação dita o grau de formalidade e até influencia a escolha das palavras a usar durante a mesma. O contacto visual também é revestido de extrema importância uma vez que permite observar a reação dos ouvintes, tornar a comunicação mais persuasiva e eficiente e de alguma forma prestigiar as pessoas na plateia. A voz deve ser projetada com emoção e devemos enfatizar os pontos chave do discurso, evitando assim falar baixo e sempre no mesmo tom. Uma pessoa que fale com uma respiração ofegante poderá passar ansiedade e incerteza para quem está a assistir à apresentação. A postura deve acompanhar o grau de formalidade da apresentação e deve-se evitar andar com as mãos nos bolsos ou com os braços cruzados numa posição defensiva. Segundo estudos mostrados pela oradora a linguagem corporal pesa 55% na influência de um apresentador. Devemos evitar utilizar sempre as mesmas palavras, virar as costas ao público, manter posições estáticas e fazer movimentos exagerados.

Após esses conselhos a oradora falou das três etapas básicas para fazer uma boa apresentação. Essas etapas são o planeamento, a criação e a execução. Na etapa de planeamento definimos o objetivo da apresentação, assegurando assim que o tópico principal fica claramente declarado, limitando o número de tópicos relacionados com o tópico principal de forma a evitar criar confusão no público. Na segunda etapa, devemos separar as informações que precisam de ser visualizadas das que precisam de ser faladas, elaborando o texto dos diapositivos de forma clara e objetiva e deve- se colocar um título impactante para chamar a atenção do público. Os slides devem ser projetados para captar a atenção do público, resumindo os principais pontos abordados, através de tópicos. O layout do slide deve ser constante, com fontes simples, com um tamanho de letra suficiente para serem lidos em toda a sala, com um fundo que contraste com a cor da letra, evitando mistura de cores, o uso de negrito e itálico e ainda evitar carregar o slide com muita informação. Os slides devem ser elaborados consoante a regra dos terços. Esta regra

defende que os espaços que chamam mais à atenção numa imagem são os que devem ser sempre preenchidos:

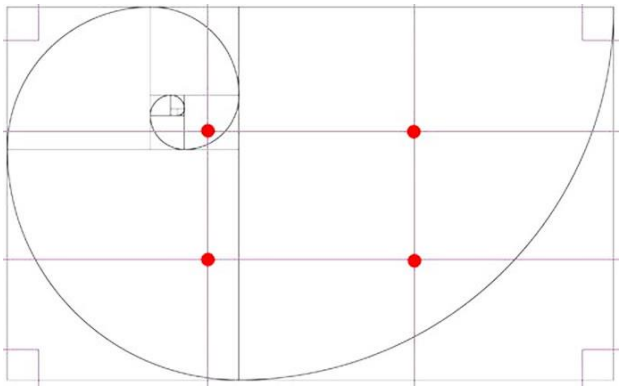


Figura 23- Exemplo da regra dos terços

Por fim na etapa de execução devemos ter cuidado com atrasos, devemos evitar ler dos slides, devemos ser flexíveis com o público e respeitar todos os conselhos já referidos anteriormente.

Seminários das apresentações intermédias - Diogo Botelho

O Diogo está a realizar o estágio na empresa *Do It Lean*. O seu projeto de estágio tem o nome "*Recycle Azores"* e foi desenvolvido com o intuito de resolver problemas com a reciclagem e combater o excesso de informação incorreta que circula sobre o mesmo tópico. Esta aplicação foi desenvolvida em *OutSystems* e veio proporcionar uma plataforma de gestão da reciclagem por município, tendo serviços de localização de ecopontos, pedidos de recolha de lixo, método de comunicação mais eficaz com os serviços municipais, informação sobre a próxima recolha do lixo e outras informações pertinentes.

Um *OutSystem* é uma plataforma de desenvolvimento *low-code* para a *web* e *mobile*, utiliza metodologias ágeis e pode integrar quando necessário sistemas externos à plataforma. A principal vantagem da utilização deste tipo de plataformas é a curva de aprendizagem reduzida, uma semana, em comparação com as plataformas tradicionais em que a curva de aprendizagem é de cerca de 6 semanas. Outras vantagens são a abstração de conceitos, permitindo focar no desenvolvimento do projeto, a interação com a aplicação é feita através de um *frontend* e utiliza *widget trees*, de forma semelhante ao *bootstrap* para organizar o conteúdo das páginas.

Na sua apresentação intermédia faltou um diapositivo em que fizesse um sumário sobre o que ia ser falado durante a sua apresentação.

Seminários das apresentações intermédias - Bernardo Sousa

O Bernardo está a realizar o seu estágio na empresa *Azores Hive*. O seu projeto de estágio tem o nome "*Success factors integrator*" e foi desenvolvido através da plataforma *Outsystems* com o objetivo de desenvolver uma aplicação que consuma informação de um software de recursos humanos, nomeadamente fazer operações CRUD sobre os dados, disponibilizando posteriormente essa informação através de uma API REST*full*. Para conseguir atingir os objetivos do estágio fez formação em *OutSystems* durante 200 horas através de uma metodologia tradicional, uma metodologia *reactive* e após isso desenvolveu duas pequenas aplicações, uma para gestão de reservas de um hotel e a outra, um sistema de compra de bilhetes, para testar os conhecimentos adquiridos durante as primeiras etapas de formação.

Na sua apresentação intermédia faltou uma definição e respetiva explicação clara dos objetivos do projeto de estágio. Faltou ainda um cronograma em que mostrasse as atividades já desenvolvidas e as que faltam desenvolver.

Seminários das apresentações intermédias - Airton Tavares

O Airton está a realizar o seu estágio na empresa SATA. O seu projeto de estágio centra-se no desenvolvimento de uma aplicação para a gestão de tráfego aéreo para a ANAC. A equipa de desenvolvimento é composta por 5 colaboradores da SATA, 2 colaboradores da *TeTrapi* e 2 estagiários da Universidade dos Açores. A metodologia de trabalho associada a este projeto é a metodologia SCRUM, para definição do esforço para cada tarefa do *backlog* da aplicação foi utilizada a técnica *Planning Poker*. Esta técnica, também apelidada por SCRUM *Poker*, permite que cada membro da equipa de desenvolvimento dê a sua estimativa sobre a duração de cada tarefa, sem existir influência dos outros membros. Para decidir o esforço para cada tarefa o grupo deve chegar a um consenso.

O projeto foi desenvolvido em três *sprints*. No primeiro *sprint* foi planeado a criação da entidade operador, no segundo *sprint* foi planeado a entidade de aeronave e o início do desenvolvimento da UI. No último *sprint*, foi planeado o restante desenvolvimento e finalização da UI. Numa fase inicial foram desenvolvidos protótipos da UI.

Foram utilizados os seguintes padrões de desenvolvimento de software:

- *Domain Driven Design* (DDD)
- *Test Driven Development* (TDD)
- *Command Query Responsability Segregation* (CQRS)

O DDD centra o desenvolvimento da aplicação na programação de um modelo de domínio rico em regras de negócio.

O TDD consiste em desenvolver testes automatizados para uma determinada funcionalidade e após os testes é que é produzido o código. Isto cria um ciclo de desenvolvimento em que se escreve os testes, depois são executados os testes para revelar alguma falha, de seguida escreve-se o código, numa fase seguinte executa-se os testes para confirmar se o código faz o pretendido e por fim faz-se a refatorização do código escrito.

O CQRS consiste na utilização de modelos diferentes para a atualização de dados e para a leitura dos mesmos. Em alguns casos essa separação de modelos pode trazer benefícios, no entanto há que ter em conta que na maioria das vezes adiciona complexidade desnecessária ao sistema.

Para este projeto foram utilizadas as tecnologias *PHP, JavaScript, React* e *MySql* através dos softwares, *Azure DevOps, PHPStorm, Adobe Ilustrator* e o *MySQL WorkBench.*

Na sua apresentação intermédia, não foi falado da parte de desenvolvimento do projeto, não explorou as tecnologias utilizadas e não aprofundou os três padrões de desenvolvimento de software mencionados.

Seminários das apresentações intermédias - Filipe Correia

O Filipe está a realizar o seu estágio na empresa SATA. O seu projeto de estágio chama-se “Formulários de Tráfego” e tem como objetivos a recolha, processamento e envio á ANAC dos dados relativos ao movimento de aeronaves, passageiros, carga e correio.

A metodologia utilizada no projeto, tal como no do Airton Tavares é a metodologia SCRUM, utiliza também o *Planning Poker* e o desenvolvimento foi divido por 3 *sprints*. As tecnologias utilizadas foram, PHP, SQL através do *Docker* e *Kubernetes*, para o *frontend* foram utilizadas como linguagens de programação o HTML, CSS e *JavaScript* com *React.Js.* As principais atividades realizadas no projeto foram a implementação de operações CRUD, a utilização do padrão de desenvolvimento *Test Driven Development.*

Na sua apresentação intermédia faltou um diapositivo em que fizesse um sumário sobre o que ia ser falado durante a sua apresentação, faltou ainda um cronograma que mostrasse as atividades já desenvolvidas e as que faltam desenvolver e os slides tinham demasiado texto.

Seminários das apresentações intermédias - Beatriz Silva

A Beatriz está a realizar o seu estágio na empresa *Sublime Cipher*. O seu projeto de estágio, "*Bumpy – Mobile*" consiste no desenvolvimento do frontend de uma aplicação mobile para a área financeira, que permitirá fazer uma centralização do portfólio de investimentos do utilizador. O processo de aprendizagem passou por entender os conceitos do mercado financeiro e o estudo das tecnologias utilizadas no projeto.

Para a elaboração de protótipos da UI foi utilizado o *Figma*, o *firebase* foi utilizado para a autenticação e registo, a *framework Flutter* na linguagem *Dart* para o desenvolvimento do projeto e o BLoC como padrão de arquitetura, permite gerir estados e é inspirado no padrão MVC. A *framework Flutter* é multiplataforma e em março de 2021 já existiam 150 mil aplicações desenvolvidas através dela.

As funcionalidades já desenvolvidas foram o *login*, o portfólio, as transações na aplicação (CRUD), a compra de ativos e a pesquisa através do *Algolia.* O *Algolia* é uma plataforma que permite indexar e procurar dados.

Após isso existiu uma demonstração das funcionalidades já implementadas na UI e uma comparação entre os diferentes ecrãs desenvolvidos.

Seminários das apresentações intermédias - Bruno Miguel Fernandes

O Miguel está a realizar o seu estágio na empresa *Sublime Cipher*. O seu projeto de estágio, "*Bumpy – Mobile*", consiste no desenvolvimento do *backend* de uma aplicação *mobile* para a área financeira, que permitirá fazer uma centralização do portfólio de investimentos do utilizador.

Esta aplicação foi desenvolvida em *JavaScript* através do *node.js*, a tecnologia utilizada para a autenticação e *login* foi o *Firebase,* para a base de dados utilizou o *FireStore* e para o *deploy* de funções utilizou a funcionalidade *Functions* do *Firebase.* O seu processo de desenvolvimento de software passou pelo desenvolvimento dos métodos, de seguida corria esses métodos num emulador, seguidamente carregava para o *Firebase* e por fim testava a aplicação. As restantes tecnologias utilizadas foram a *framework Flutter* na

linguagem de programação *Dart*, o *Github*, o *Notion* e o *SwaggerHub* para documentar a API.

Como tarefas realizadas foram identificadas a integração com API's de terceiros, o desenvolvimento de *triggers, scheduler jobs, full-text search* e um algoritmo de calculo de portefólios. Os *triggers* consistem em funções que correm de forma automática quando existe alguma alteração ao *backend*. Os *scheduler jobs* são funções que correm de forma recorrente dependendo do agendamento definido para as mesmas. O *full-text search* foi uma funcionalidade implementada para procurar *assets* através da aplicação, para isso foi utilizado o *Algolia*.

Na sua apresentação intermédia, faltou um cronograma que mostrasse as atividades já desenvolvidas e as que faltam desenvolver.

CONSIDERAÇÕES FINAIS

Considero que todos os objetivos propostos no plano de formação inicial foram concluídos com sucesso.

Os principais conhecimentos e competências adquiridos foram a utilização do *Qt Designer* e da plataforma *Firebase*, assim como o aumento da experiência no desenvolvimento de software e de todos os processos necessários no levantamento de requisitos, no desenho do software e desenvolvimento do último.

As principais oportunidades de melhoria na aplicação seriam o desenvolvimento dos *mockups*, o *logout* da aplicação e a implementação de testes automatizados.

O estágio permitiu o desenvolvimento de *software* de uma forma mais autónoma e realista, permitindo assim desenvolver um conjunto de estratégias para a resolução de problemas frequentes em todas as etapas do desenvolvimento de *software*. Permitiu também um aumento de aptidões ao nível da interação com pessoas que não possuem o mesmo nível de conhecimento técnico.

Como objetivos após terminar a licenciatura, tendo em conta o estágio e como sou trabalhador-estudante, pretendo realizar trabalho em *part-time* para obter mais experiência no requisito de programação e engenharia de *software*. Em questões de formação, e tendo por base as disciplinas lecionadas no decorrer deste curso, pretendo concorrer a um mestrado na área de informática ou uma pós-graduação nesta mesma área. Pretendo ainda tirar algumas formações na CISCO ao nível de redes e cibersegurança para aumentar a especialização nesta área.

BIBLIOGRAFIA

Sommerville, I. (2015). **Software Engineering** (Tenth Edition). Pearson.

SCRUMstudy, **A Guide to the Scrum Body of Knowledge: Sbok Guide** (Third Edition). SCRUMstudy

CISMIL do Estado-Maior-General das Forças Armadas (2020). **Instruções de Segurança Militar**. Ministério da Defesa Nacional

Supreme Headquarters Allied Powers Europe (2019). **ACO Security Directive 070-001**. NATO

Regulamento Geral de Proteção de Dados. Regulamento (UE) N.º 2016/679, de 27 de abril de 2016

[1] Portugal já ultrapassou o limite em todas as categorias ambientais (2021), Diário de Notícias, consultado em 9 de março de 2022, disponível em: https://www.dn.pt/sociedade/portugal-ja-ultrapassou-o-limite-em-todas-as-categorias-ambientais-14368066.html

Estrutura do Estado-Maior-General das Forças Armadas (2022), EMGFA, consultado em 7 de março de 2022, disponível em: https://www.emgfa.pt/emgfa/estrutura/

Firebase (2022), consultado em 10 de abril de 2022, disponível em: https://firebase.google.com/

Qt Designer Manual (2022), Qt Documentation, consultado em 26 de maio de 2022, disponível em: https://doc.qt.io/qt-5/qtdesigner-manual.html

ANEXOS

Anexo 1 - Diagrama de G*antt* do PIF (Original)

Anexo 2 - Diagrama de *Gantt* do PIF (Alteração)

Anexo 3 - *Backlog* geral da aplicação

Anexo 4 - *Backlog* 1º *Sprint*

Anexo 5 - *Backlog* 2º *Sprint*

Anexo 6 - *Backlog* 3º *Sprint*

Anexo 7 - Diagrama de arquitetura

Anexo 8 - Diagrama de classes

Anexo 9 - *Mockups da UI*

Anexo 10 - Código fonte desenvolvido