

**Formando:** Marco Candeias

**UFCD:** 5425 – Projeto Final

**Curso:** JOVEM+DIGITAL - Programação em Python 001

Índice

[1 - Introdução e Objetivos 3](#_Toc1)

[2 - Descrição do Projeto 3](#_Toc2)

[2.1 - Contexto da Aplicação 4](#_Toc3)

[2.2 - Objetivos de Projetos 4](#_Toc4)

[2.3 – Justificação 5](#_Toc5)

[2.4 - Design do Projeto 5](#_Toc6)

[2.5 - Alocação do Projeto 8](#_Toc7)

[2.6 – Riscos do Projeto 8](#_Toc8)

[2.7 – Cronograma 9](#_Toc9)

[2.8 - Análise de Viabilidade 9](#_Toc10)

[2.9 – Recursos Utilizados 10](#_Toc11)

[3 – Dificuldades na Implementação 11](#_Toc12)

[4 – Código Open Source 12](#_Toc13)

[5 – Análise Crítica 12](#_Toc14)

[6 – Conclusão 13](#_Toc15)

# 1 - Introdução e Objetivos

Este documento tem como objetivo a realização de uma aplicação para a gestão de stock de material de uma empresa de sorting, virada para a industria automóvel, de nome RLS.

Verificando as necessidades da RLS aplicámos as especificações técnicas, nomeadamente em termos de Análise e Desenho da aplicação de Gestão de Stock bem como todos os aspetos relativamente ao desenvolvimento e implementação da aplicação.

O objetivo deste projeto é também aplicar conhecimentos adquiridos.

# 2 - Descrição do Projeto

Este projeto tem como objetivo criar uma aplicação que permita a uma empresa, um maior controlo sobre o inventário de material que tem. Esta aplicação permite registar os materiais e EPI’s e as suas quantidades que são guardadas em cada armário e a quem são entregues. Isto permite manter um registo simples de todo o inventário e rapidamente compreender o que é preciso comprar e reabastecer.

Esta aplicação resolve assim a utilização de registos em papel, que no final do mês teriam de ser inseridos numa folha de Excel, que por esta ser atualizada uma vez por mês não tinha grande utilidade nos avisos da falta de materiais nos armários. Problemas, que esta aplicação pretende minimizar ou eliminar.

## 2.1 - Contexto da Aplicação

A RLS é uma empresa de Sorting na area automóvel, que tem como missão atuar como um parceiro dos nossos clientes transmitindo competências, valores e soluções, através de consultadoria, formação e outsourcing especializado.

Para suportar o outsourcing, a RLS tem hubs nas empresas onde presta mais serviços. Nestas hubs tem armários com materiais como, luvas de vários tamanhos, marcadores e outros materiais que os colaboradores possam necessitar para a realização das suas funções. O abastecimento destes armários é controlado a olho e os pedidos de reabastecimento são muitas vezes feitos por via informal complicando assim o registo de abastecimento dos armários. Quando um item é entregue a um colaborador, a entrega é registada numa folha de papel que fica no armário de onde saiu o item. No final do mês a folha de cada armário é levada para o escritório onde é inserida numa folha de Excel formatada para avisar quando há falta material num dos armários.

Devido ao intervalo de tempo entre atualizações da folha de Excel e à perda de informação relativamente aos abastecimentos dos armários, este método torna-se apenas uma formalidade sem grande utilidade prática.

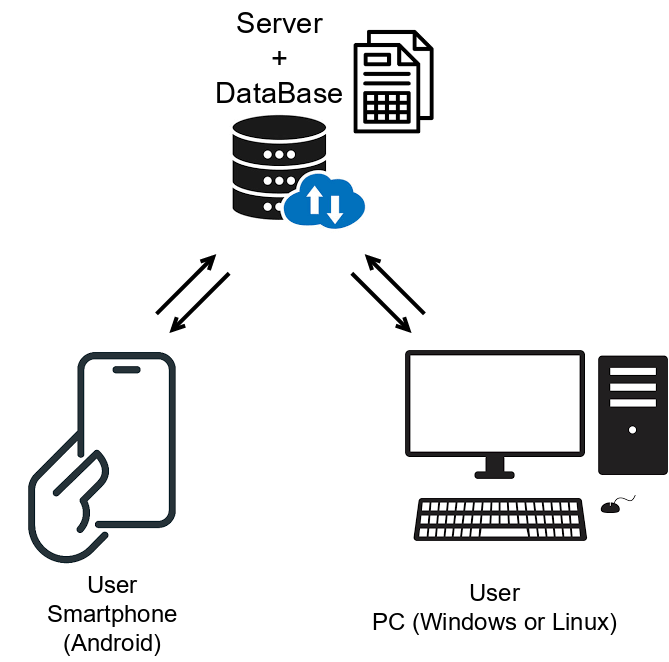
## 2.2 - Objetivos de Projetos

Esta aplicação tenciona resolver os inconvenientes e intervalos de tempo entre a entrega de um item a um colaborador e o registo na folha de Excel, tornando assim estes registos numa ferramenta útil para os supervisores e chefes de equipa, em vez de “só mais uma burocracia”.

## 2.3 – Justificação

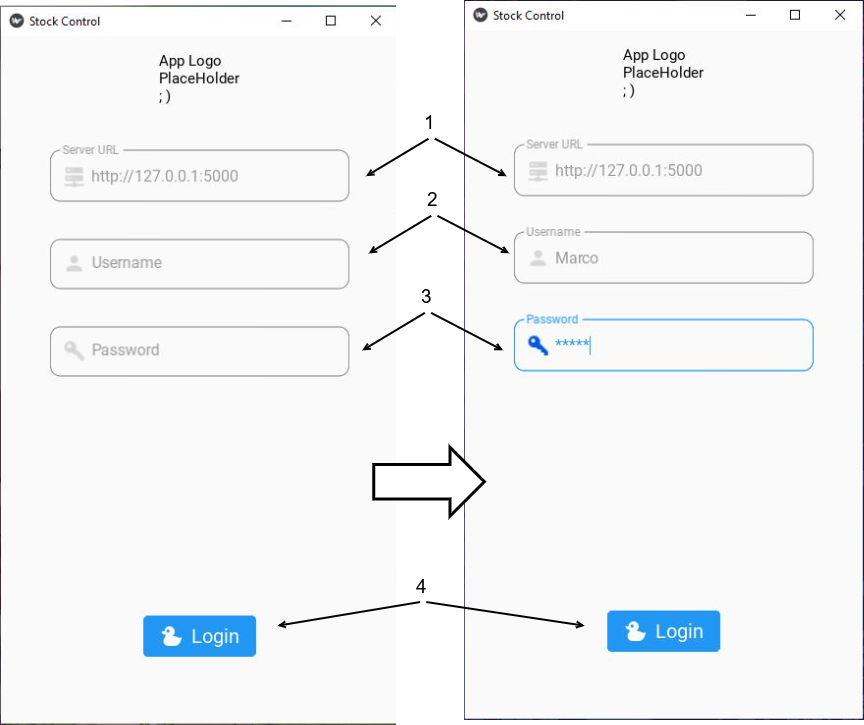
Esta aplicação visa a melhorar e automatizar um sistema já existente que devido à falta de automação se torna “só mais uma burocracia” em vez da ferramenta útil que teoricamente é. Com a implementação desta aplicação e workflow à sua volta, espera-se um aumento de eficiência no controlo de stocks dos armários nas hubs da RLS.

## 2.4 - Design do Projeto



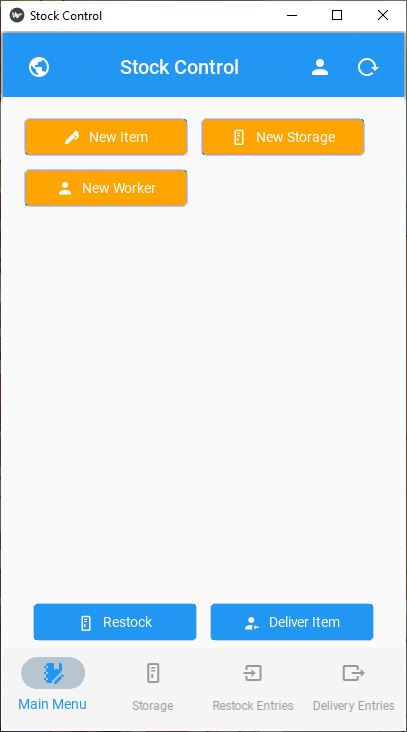
O componente do Servidor foi desenvolvido para correr nativamente num terminal em Linux, através da própria linguagem de Python. Isto tem a vantagem de facilitar o acesso remoto através de SSH.

A aplicação foi concebida com o framework Kivy, que permite a aplicação correr sem alterações em telemóveis Android e computadores com Windows e Linux.

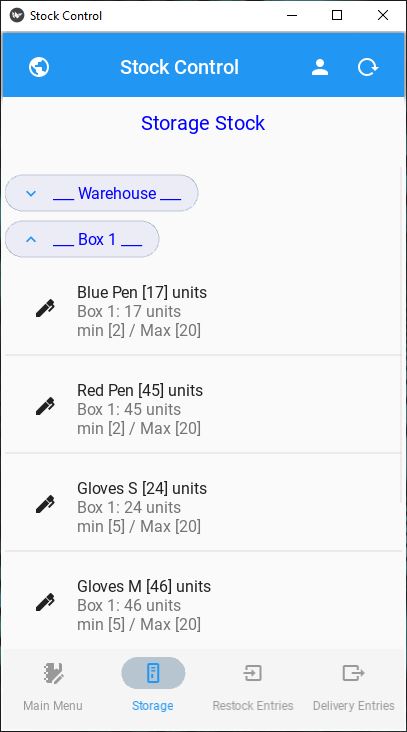


O ecrã de login contém um logótipo, um campo para inserir o endereço do servidor (1), um campo para o nome do utilizador (2), um campo para inserir a palavra passe (3) e por último o botão para fazer o login (4).

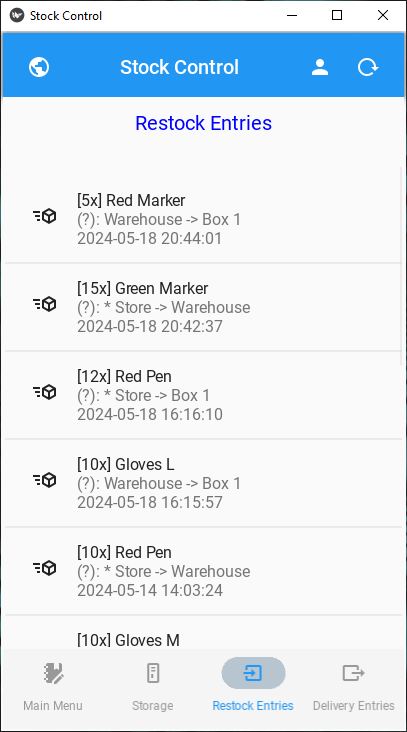
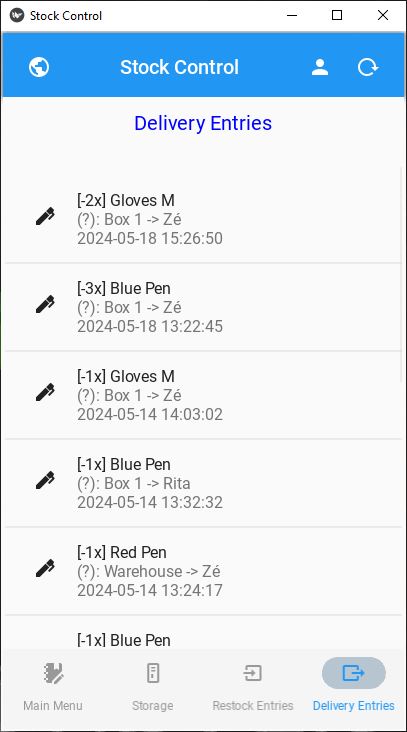
A aplicação terá um ecrã principal com tabs de navegação no fundo do ecrã.



Na tab “main menu” podemos adicionar novos itens, colaboradores e armários, bem como fazer abastecimentos dos mesmos ou entregas de materiais a um colaborador.



Na tab “storage” podemos consultar os itens e quantidades que existe em cada um dos armários, bem como o número mínimo de unidades a que um armário pode chegar até ser reabastecido.

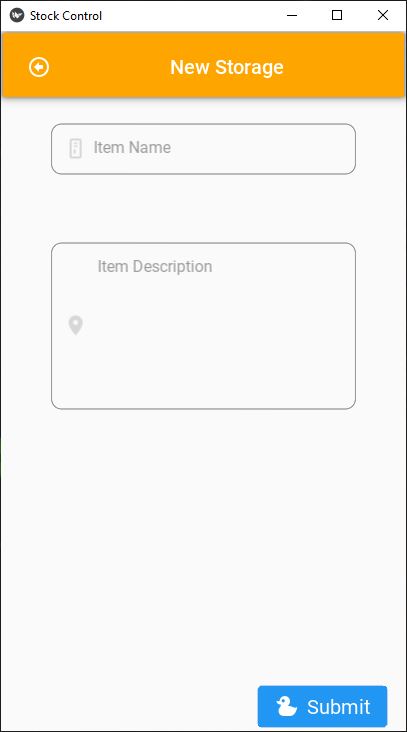
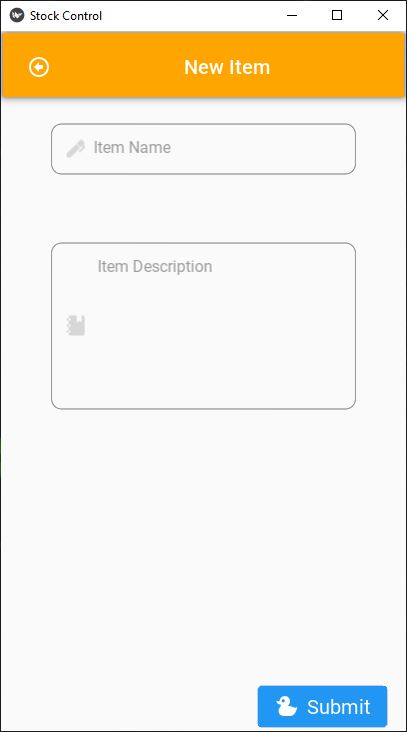
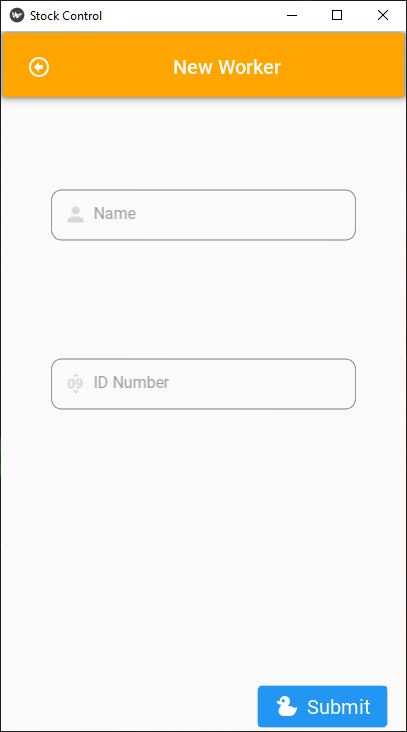


Na tab “restock entries” podemos consultar os últimos 50 abastecimentos feitos.

Na tab “delivery entries” podemos consultar as últimas 50 entregas de materiais aos colaboradores.

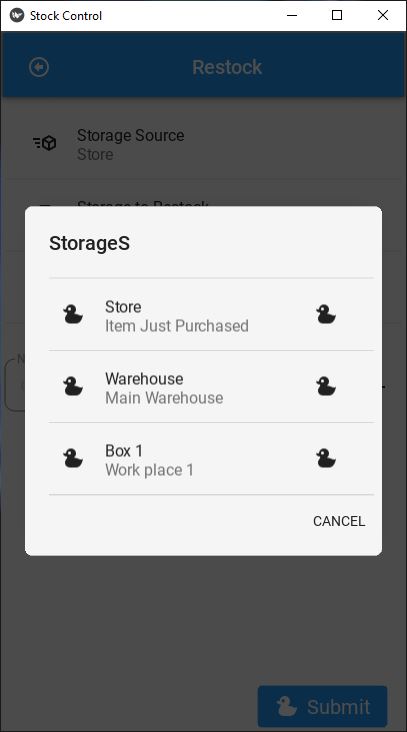
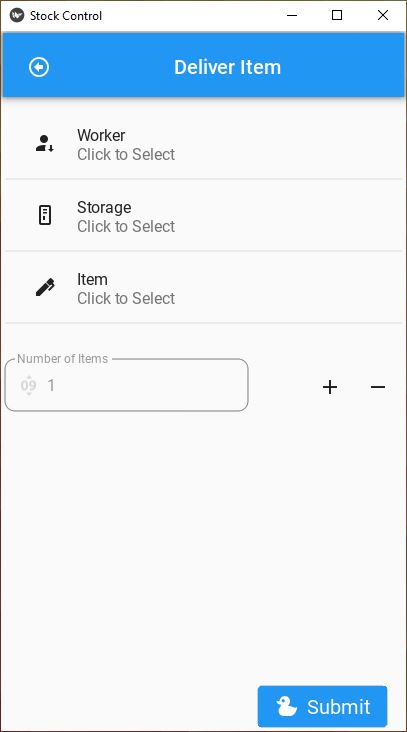
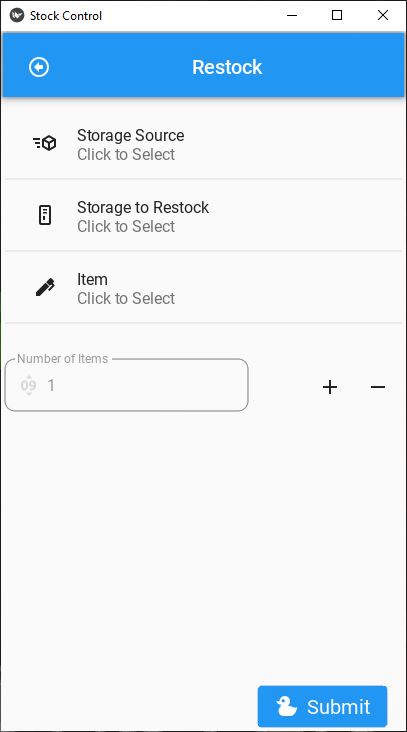
Os ecrãs para adicionar novos itens, armários e colaboradores são bastante semelhantes e intuitivos, pedindo todas as informações necessárias para a criação do respetivo item.

Por fim, o botão “submit” envia os dados para o servidor onde são validados e inseridos na base de dados, em que envia a confirmação que tudo correu bem ou que ocorreu um erro, informando o utilizador que a operação ficou concluída ou não.



O ecrã de reabastecimento e de entrega de materiais, são também muito semelhantes entre si contendo elementos opcionais para o preenchimento das informações requeridas para a operação pretendida.

Cada elemento abre um Popup com a lista de possíveis escolhas para o elemento em causa.



## 2.5 - Alocação do Projeto

Para a implementação deste projeto será necessária a compra de pelo menos um Raspberry Pi 4 ou 5, (com cabo e arrefecimento) que ficariam entre 80 a 110€ por unidade e como alternativa, poderia se usar um ou mais computadores antigos como servidores, reduzindo assim o investimento inicial.

Todo o software suplementar usado neste projeto é Free and Open Source ou escrito por mim.

## 2.6 – Riscos do Projeto

Um dos maiores desafios deste projeto é garantir a robustez da aplicação, visto que após a sua adoção, esta passa a ser parte integral da empresa e como qualquer boa ferramenta, esta deve ser robusta o suficiente para a empresa poder depender dela, sem preocupações.

**Análise SWOT da aplicação:**

Ponto forte da aplicação:

* O ponto mais forte desta aplicação é resolver um problema real, com o qual tenho experiência em primeira mão.

Ponto fraco da aplicação:

* Um dos pontos mais fracos da aplicação é ser desenvolvida e mantida por apenas uma pessoa com relativamente pouca experiência profissional na área.

Oportunidade:

* Esta aplicação é uma grande oportunidade de aprendizagem em várias áreas do desenvolvimento de software e normas ISO.

Ameaça à aplicação:

* Garantir a segurança do dados na aplicação e o cumprimento de normas na indústria, são as maiores ameaças ao sucesso da aplicação.

## 2.7 – Cronograma

Fazes do projeto:

1. Identificar o problema a resolver;
2. Planeamento do projeto e software stack;
3. Desenvolver a aplicação:
   1. Criar a Base de Dados;
   2. Criar o API em Flask para a Base de Dados;
   3. Criar a Aplicação (Frontend);
4. Instalação da aplicação no hardware:
   1. Implementar DNS Resolve, para a aplicação encontrar o servidor através da Internet;
   2. Configurar backups e high availability no cluster;
5. Tornar o software stack fácil de instalar e ligar.

O projeto é para ser realizado entre dia 26 de Abril de 2024 e dia 21 de Maio de 2024, sendo que dia 21 é a entrega do mesmo, com devida apresentação no dia 23, dia em que termina o módulo e o curso.

## 2.8 - Análise de Viabilidade

Viabilidade Técnica:

Tecnicamente, o projeto tem tudo para funcionar como desejado, partindo do princípio que todas as partes do projeto são executadas com a devida qualidade e atenção ao detalhe.

Viabilidade Financeira:

Financeiramente, a viabilidade do projeto varia consoante a forma com que queira fazer o deployment do componente do backend. Caso se opte pela aquisição de hardware, como referido anteriormente (Raspberry Pi), haverá os custos relacionados com a aquisição do equipamento e os seus custos energéticos (de 7 a 25Wh) a ter em conta. Na opção de usar computadores mais antigos, mas funcionais, temos só de considerar o consumo energético, que vai depender do equipamento em si (estimado entre 15 a 150Wh).

Temos também a opção de fazer o deployment do backend na cloud, com serviços como a AWS, Azure, Linode, entre outros, mas com preços que variam consoante os seus serviços, riscos e benefícios.

Viabilidade Operacional:

Operacionalmente, o projeto é viável, visto que foi desenvolvido para automatizar um processo já existente melhorando o mesmo. No entanto, devido a limitações técnicas, nomeadamente a falta de suporte para IOS e telemóveis Apple, pode haver alguma dificuldade na adoção da aplicação móvel, sendo que esta limitação é minimizada pelos telemóveis de serviço da empresa que são todos Android.

## 2.9 – Recursos Utilizados

Linguagem de Programação



Python:

Python é uma linguagem de programação de alto nível, de script interpretada, imperativa, orientada a objetos, funcional, de tipagem dinâmica e forte.

Frameworks usados:

Kivy:



Kivy é uma biblioteca open source escrita em Python para o desenvolvimento de aplicativos móveis ou desktop, para dispositivos multitouch possuindo uma interface natural de usuário. Aplicações escritas podem ser executadas no Android, iOS, Linux, OS X, e Windows.

Flask:



Flask é um pequeno framework web escrito em Python. É classificado como um microframework porque não requer ferramentas ou bibliotecas particulares, mantendo um núcleo simples, porém, extensível.

Base de Dados

SQLite:



SQLite é uma biblioteca em linguagem C que implementa uma base de dados SQL embutida. Programas que usem a biblioteca SQLite podem ter acesso a banco de dados SQL sem executar um processo SGBD separado.

# 3 – Dificuldades na Implementação

Por falta de tempo não foi possível implementar a conectividade através da Internet, apenas foi implementada conectividade através da LAN. No entanto um Raspberry Pi 5 foi usado como servidor.

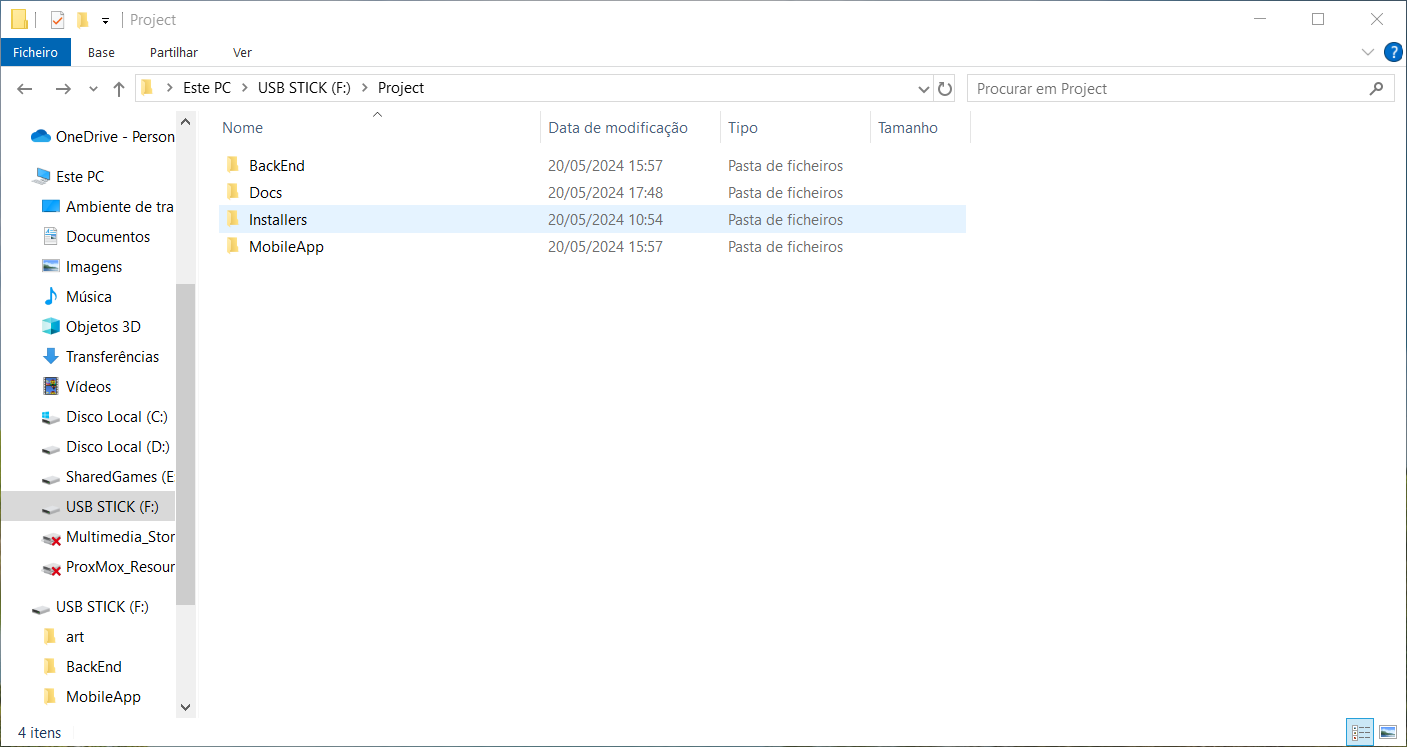
Deparei-me também com vários problemas que levaram a alterações ao UI/UX da aplicação levando a atrasos no desenvolvimento da aplicação.

Devido a todos os atrasos já mencionados, não me foi possível aprender como estabelecer comunicação entre um servidor e uma aplicação através da Internet, nem como criar e manter o cluster.

# 4 – Código Open Source

Todo o código está disponível no ficheiro zip anexado e na minha conta de GitHub: <https://github.com/MK-Dragon/StockControll.>

Na pasta principal encontram-se as sub-pastas contendo o código referente ao BackEnd/servidor e a sub-pasta com o código da aplicação móvel. Nela encontra-se também a sub-pasta “Docs”, que contem vários documentos de apoio e screenshots da aplicação.



A pasta installers contêm o ficheiro de instalação APK da aplicação em dispositivos Android, bem como os instaladores da aplicação e do servidor e os ficheiros EXE para correr o Servidor e a Aplicação em Windows sem instalar.

# 5 – Análise Crítica

Inicialmente achei que o desenvolvimento da backend seria a parte que iria consumir mais tempo e recursos e que o frontend, por ter já alguma experiência com o framework escolhido, que iria consumir muito menos tempo e recursos. No entanto, a experiência foi contrária á minha expectativa inicial, tendo o UI/UX da aplicação, bem como toda a integração lógica com o backend, consumido a grande maioria do tempo gasto no projeto.

Retrospetivamente, o que correu bem:

* A Base de Dados, foi simples de implementar, SQLite simplificou imensamente o BackEnd.
* O desenvolvimento da API com Flask foi bastante mais simples e intuitiva de implementar do que eu estava à espera. Foi apenas escrever uma vez e correu o projeto inteiro sem dar grandes bugs ou dores de cabeça.
* Utilizar PyCharm ligado a uma conta do GitHub foi bastante útil para “saltar” entre o meu PC de torre e o meu Portátil, permitindo-me trabalhar, tanto em casa, como fora dela, sem ter de andar com pen’s ou discos externos de um lado para o outro.

O que, retrospetivamente correu mal:

* Usar o Framework Kivy para a parte gráfica da aplicação não foi um erro, mas mostrou requerer bem mais tempo e dedicação do que estava à espera.
  + Compilar e testar a aplicação no telemóvel acabou por consumir muito mais tempo, devido a vários bugs e detalhes que iam aparecendo aqui e ali, abrandando o projeto.
  + Cada vez que testava a aplicação no telemóvel, aparecia mais um detalhe ou problema com o UI/UX, que abrandava ou parava o projeto por completo, obrigando a várias revisões e re-implementação de features.
* Usar o PyInstaller em combinação com o INNO Setup foi uma dor de cabeça à última da hora. Por algum motivo o INNO Setup não cria pastas vazias onde a aplicação possa criar ficheiros e usar-se a aplicação para criar essas pastas, os anti-vírus “fazem birrinha”, pelo que tive de arranjar uma solução rápida de última hora, que “atira pela janela” toda a “elegância” de separar ficheiros (logs e db) por pastas.

# 6 – Conclusão

Tendo em conta todos os imprevistos encontrados ao longo do projeto, um mês não foi tempo suficiente para criar esta aplicação do zero. No entanto fica a experiência adquirida durante todo o processo.