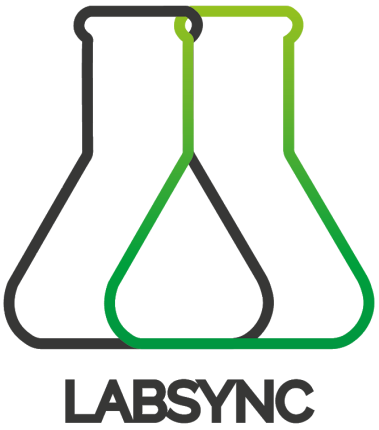
• U C •
FCTUC FACULDADE DE CIÊNCIAS
E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Engenharia de Software

Documento sobre a Arquitetura de software do produto

Departamento de Engenharia Informática



(Equipa PL7)

Controlo do Manual	Data	Validade de todas as páginas
Versão 1.3	2018-11-19	Elaborado por: António Fraga Fernando Felício João Miranda

Índice

1-Introdução

O objetivo principal deste documento é descrever a arquitetura de software do produto) que estamos a desenvolver para compreensão do seu funcionamento e as tecnologias que foram e estão a ser usadas no seu desenvolvimento e o porquê da escolha das mesmas.

Para melhor compreensão da estrutura na qual é baseada o nosso software usamos diversos diagramas com base no modelo C4 de arquitetura de software sendo esses os seguintes:

- “C4 - System Context”
- “C4 - Container Diagram”
- “C4 - Component Diagram”
- “C4 - User-Component Diagram”

Sendo que estes estes estão devidamente identificados ao logo do documento.

2 - Possibilidades de Software

2.1 - Possíveis Frameworks

2.1.1 - Django (Python)

O primeiro software considerado pela nossa equipe foi o Django. É uma framework de Python.

É uma estrutura de alto nível com uma curva de aprendizagem suave e possui uma ampla variedade de pacotes que permitiriam que a equipa de implementação rapidamente
desenvolve-se e aperfeiçoa-se a plataforma para o cliente. A principal diferença entre este quadro e os outros considerados é o fato de que a equipa implementação
nunca tocará nos arquivos de migração, pois o Django gere a base de dados por si só com base nos atributos e relacionamentos de cada modelo.

Também é muito fácil configurar nos vários sistemas operativos
equipe de implementação e teste.

Um outro aspecto digno de ser mencionado é o fato de que a maioria dos membros da equipa são alunos de 3ºano de LEI têm experiência anterior em desenvolvimento em Python e, como tal, aprendem a desenvolver a partir deste framework com facilidade.

2.1.2 - Ruby on Rails (Ruby)

Ruby on Rails é uma framework de Ruby. É uma framework de alto nível que também
tem uma curva de aprendizado muito suave e uma ampla gama de packages (sockets).

2.1.3 - Laravel (PHP)

O Laravel é uma framework de PHP para desenvolvimento rápido, livre e open-source. Tem como vantagens a possibilidade de trabalhar rápido e de forma estruturada.
PHP não era um linguagem conhecida amplamente e de forma abrangente pelos membros da equipa de Implementação.

2.2 - API's Possíveis

2.2.1 - Twitter API

Esta API permite facilmente dar display a tweets, descobrir e comissionar Tweets assim como partilhar conteúdo via Twitter.

2.2.2 - Reddit API

A API do Reddit permite aceder aos posts enviados e avaliados pelo user no site/app do Reddit (reddit.com). Também fornece uma opção de funcionalidade avançada que inclui informações de contas de users e moderações para sub-reddits.

2.2.3 - ORCID API

A API do ORCID permite que os sistemas e aplicações se conectem ao registo ORCID, incluindo a leitura e gravação em registros ORCID. A API tem 3 modelos, Public, Basic Member e Premium Member, pelo que terá de ser estudado qual das subscrições contempla o pretendido pelo nosso software.

2.3 - DBMS

2.3.1 - SQLite

O SQLite é um single-threaded de SQL DBMS, incorporado no programa final. isto
vem por padrão com o framework de desenvolvimento web descrito neste documento (Django). O problema com este DBMS vem ao nível do deployment, visto que despeja toda a informação para um ficheiro e alguns web services não permite a escrita direta para ficheiros

É perfeito para o desenvolvimento, mas é difícil de usar na produção.

3 - Software Escolhidos

Após escolha inicial e interna por parte de Implementação e com algumas mudanças e adaptações necessárias (maioritariamente por parte dos Requisitos impostos pelo cliente) estas são as escolhas efectuadas:

Framework da WebApp:

- Django;

API's utilizadas:

- Twitter;
- Reddit;
- ORCID;

DBMS:

- SQLite;

A framework Django foi escolhida com base na discussão entre os membros da equipa de implementação e foi escolhida, visto que, se chegou à conclusão que esta seria a melhor opção com base nos conhecimentos da equipa, já que não existiam conhecimentos unânimes a todos os membros da subunidade entre as outras opções, e como descrito e explicado na secção anterior esta framework tem um curva de aprendizagem suave e a maioria dos membros sabia desenvolver em Python.

As API's do Twitter, Reddit e ORCID foram utilizadas para satisfazer os requisitos impostos para o cliente, como a inclusão de posts de ambos Twitter e Reddit (com base nos interesses do utilizador) e autenticação no login/sign-in através da plataforma ORCID.

A DMBS SQLite não foi necessariamente "escolhida", mas sim a opção mais viável por ser a DMBS default para a framework de Python, Django.

4 - Utilizadores

4.1 Utilizador

Para se tornar utilizador da nossa webapp é necessário efectuar um registo inicial em que o utilizador providencia um leque de informações sendo que as obrigatórias são, um endereço de e-mail, nome, ORCID, entidade à qual está associado. Sendo que depois de ter conta registada, tem a opção de editar o seu perfil, incluindo foto de perfil, breve descrição, links para redes sociais e interesses que vão gerar feed de tweets e notícias do reddit sendo que estas vão variar dependendo desses mesmos interesses escolhidos pelo utilizador

4.2 Administrador

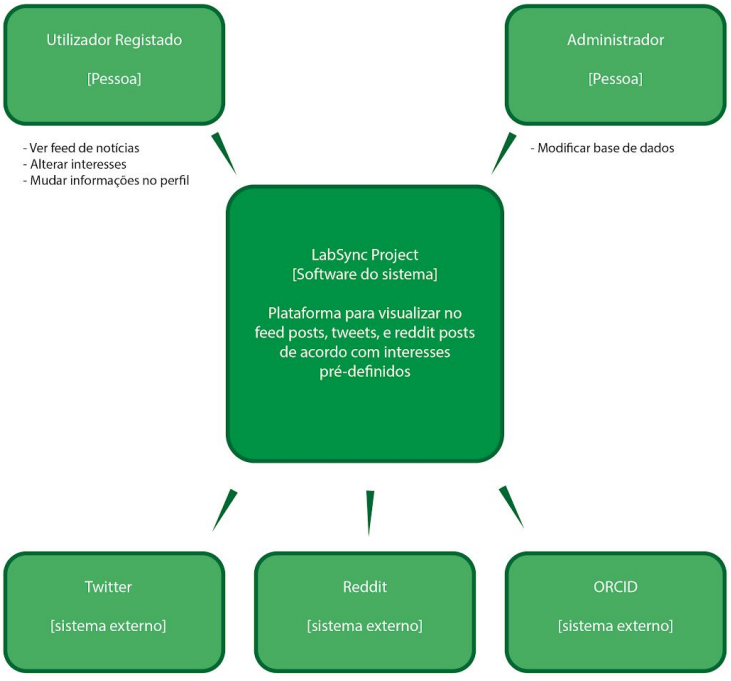
Este tipo de utilizador tem as mesmas características que um utilizador apenas com a excepção de que pode alterar e manipular dados contidos na base de dados.

5 – Descrição da Arquitetura

5.1- “C4 - System Context ”

Este diagrama tem como principal utilidade demonstrar de uma forma simples os utilizadores e a sua interação com o sistema, ou seja, todas as ações que este pode executar enquanto usa o nosso sistema. Para além disso, este diagrama também serve para mostrar os sistemas externos que são necessários usar para cumprir os requisitos deste projeto.

Passando a informação exibida pelo diagrama abaixo exposto, podemos ver os users da nossa plataforma e as ações que estes podem executar na mesma, sendo que essas ações estão descritas extensivamente na secção 2 deste documento. Também é possível ver os sistemas externos e a sua função/relação com a nossa plataforma descrita muito simplificadamente, sendo que a informação mais aprofundada também já tinha sido dita noutra secção deste documento neste caso na secção 1.

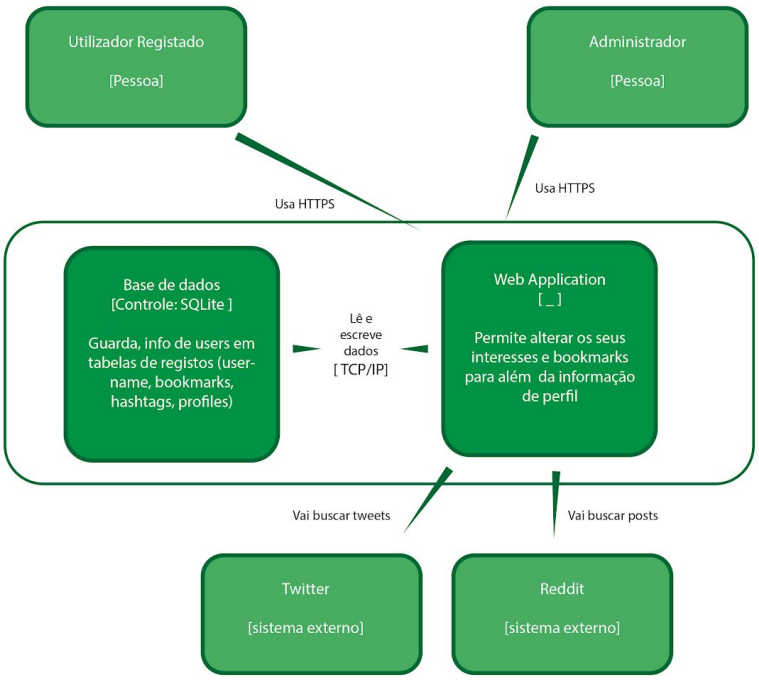


5.1 Figura #1 - System Context Diagram para LabSync Project

5.2 - “C4 - Container Diagram ”

Depois de percebermos a interação dos utilizadores com o sistema, passa a ser importante fazer um “zoom in” na nossa plataforma e representar também os “Container”, sendo que estes são as unidades que podem ser corridas separadamente com intuito de executar ou guardar memória, sendo que no nosso caso temos por exemplo a nossa base de dados. Este diagrama é então a representação destes “containers” e a sua interação com os utilizadores e com o sistema em si.

Cada “container” tem uma descrição breve da sua função em relação a plataforma e como interage com ela, ou seja, o tipo de protocolo de comunicação que usa para comunicar com esta e também com os sistemas externos.



5.2 Figura #2 - Containers Diagram para LabSync Project

5.3 - “C4 -Component Diagram”

Os componentes são necessários para implementar os requisitos especificados. Precisamos também de saber como estes interagem com o projeto e entre si.

Componentes do projeto:

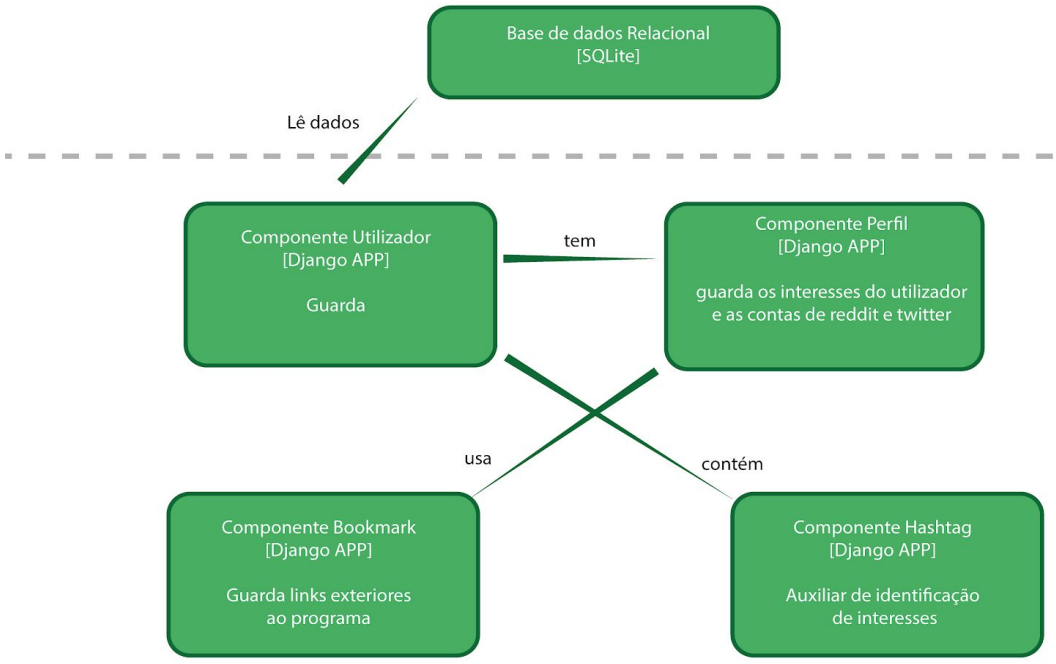
- **Componente Reddit** – é utilizada uma API para procurar os posts dos subreddits escolhidos pelo utilizador;

- **Componente Twitter** – para encontrar os posts no Twitter, é utilizada uma API que identifica os tweets com hashtags que correspondem aos interesses do utilizador;

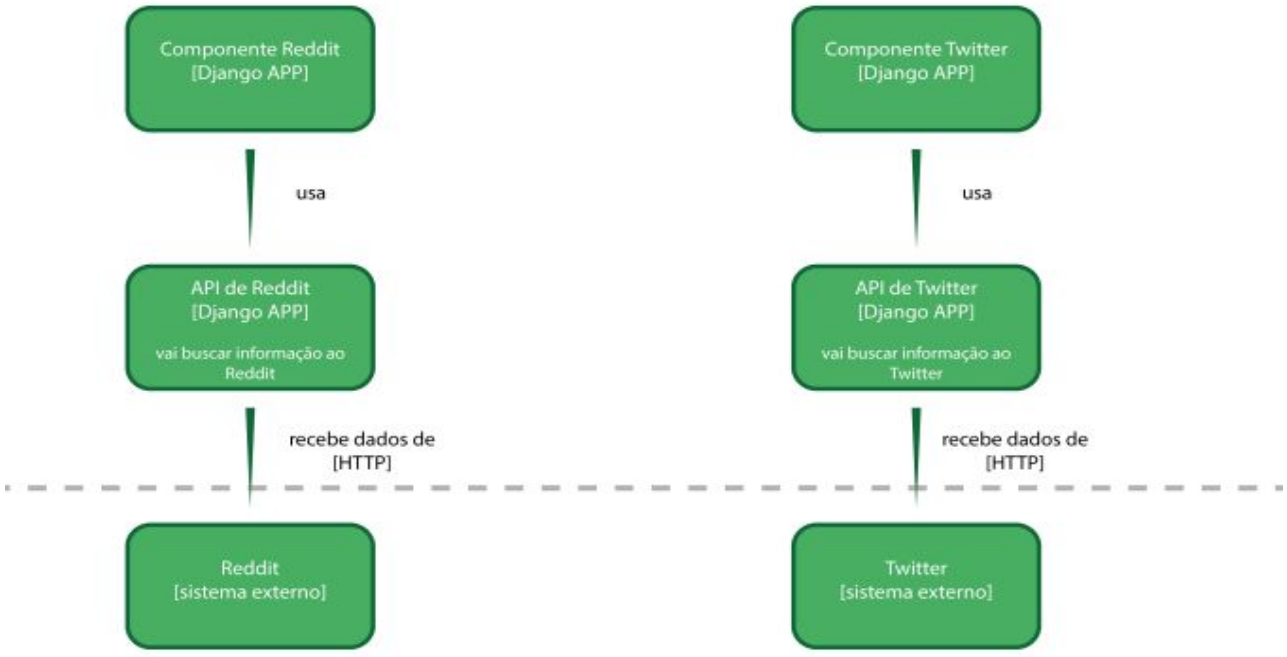
- **Componente Hashtag** – serve como auxiliar de identificação dos interesses do utilizador;

- **Componente Bookmark** – é uma forma de guardar links exteriores ao próprio programa de forma organizada;

- **Componente Utilizador e Perfil** – representa o username e a password do utilizador na app. Cada utilizador contém um perfil com informação importante como os interesses do utilizador, os subreddits escolhidos, as contas do Reddit e do Twitter associadas ao utilizador, entre outras opções de customização.


Base de dados Relacional
[SQLite]
Lê dados
Componente Utilizador
[Django APP]
Guarda
tem
Componente Perfil
[Django APP]
guarda os interesses do utilizador
e as contas de reddit e twitter
usa
contém
Componente Bookmark
[Django APP]
Guarda links exteriores
ao programa
Componente Hashtag
[Django APP]
Auxiliar de identificação
de interesses

5.3

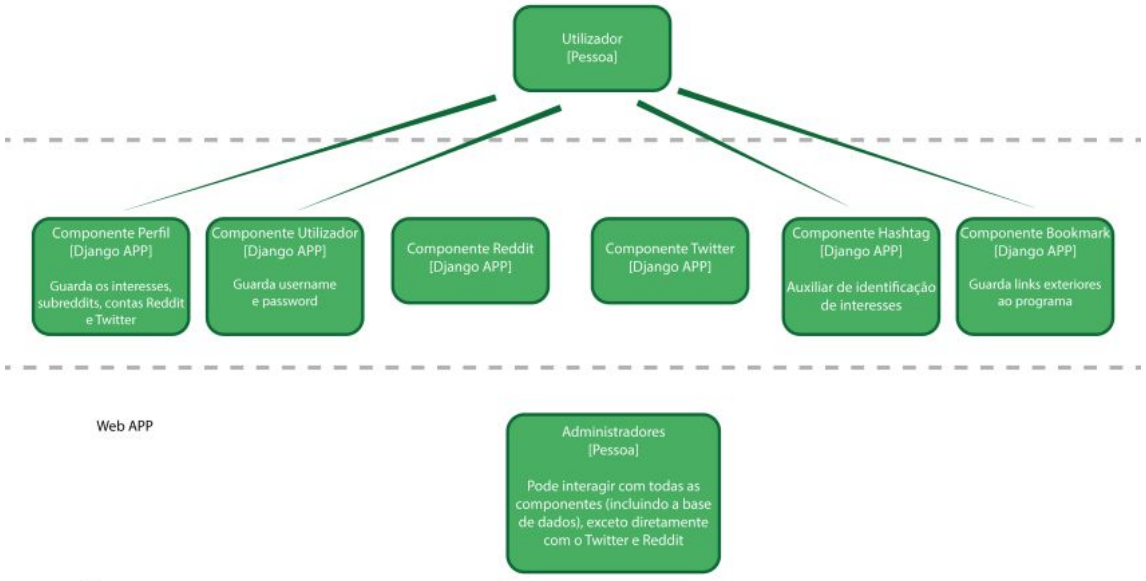

Componente Reddit
[Django APP]
Componente Twitter
[Django APP]
usa
usa
API de Reddit
[Django APP]
vai buscar informação ao
Reddit
API de Twitter
[Django APP]
vai buscar informação ao
Twitter
recebe dados de
[HTTP]
recebe dados de
[HTTP]
Reddit
[sistema externo]
Twitter
[sistema externo]

5.3

5.4 - “C4 - User-Component Diagram”

Este diagrama consiste nas relações que vão existir entre os utilizadores e os componentes do sistema.

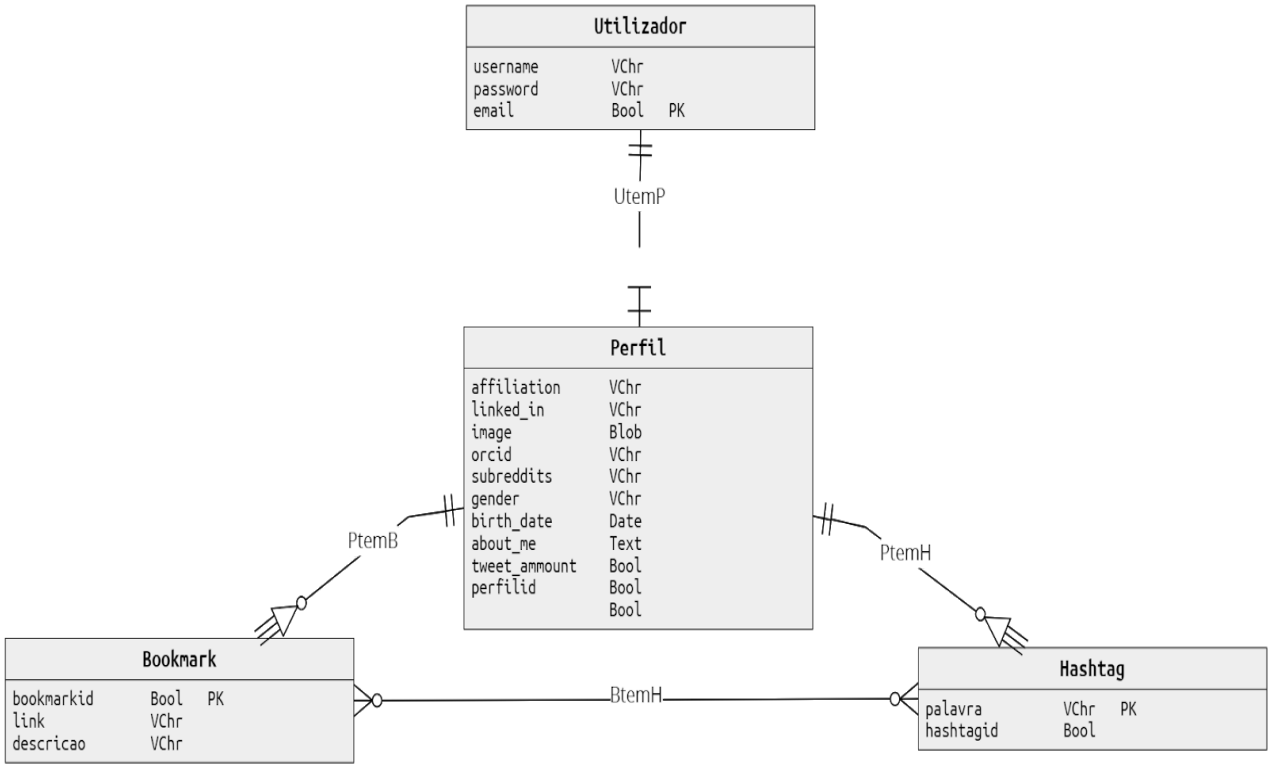
O diagrama utilizador-componente serve apenas para simplificar o que já foi explicado no diagrama de componentes.



5.4

5.5 - Software Data Model

O diagrama seguinte representa a relação entre as entidades do projeto (diagrama ER). Seguidamente será explicada cada uma das entidades com mais pormenor.



5.5.1 - Entidades

5.5.1.1 - Utilizador

O utilizador pode ser um user normal ou ser um Administrador sendo que a diferença entre estes é que o Administrador consegues mexer nos objetos guardados nas bases de dados.

Atributos da entidade Utilizador:

- username;
- password;
- email - chave primária que serve para distinguir os diferentes utilizadores.

Relações da entidade Utilizador:

- cada utilizador tem um perfil e se este for apagado, o utilizador também é apagado.

5.5.1.2 - Perfil

O perfil guarda as informações do utilizador.

Atributos da entidade Perfil:

- affiliation;
- linked_in;
- image;
- orcid;
- subreddits;
- gender;
- birth_date;
- about_me;
- tweet_ammount - quantidade de tweets que apereceram no feed;
- perfilid - chave primária da entidade (atua como identificador de cada perfil).

Relações da entidade Perfil:

- cada utilizador tem bookmarks e hashtags;

5.5.1.3 - Bookmarks

As bookmarks são links exteriores ao projeto e têm hashtags

Atributos da entidade Bookmarks:

- bookmarkid - chave primária para distinguir as bookmarks;
- link;
- descrição

Relações da entidade Bookmars:

- As bookmarks estão presentes nos perfis;
- têm hashtags;

5.5.1.4 - Hashtag

As Hashtags funcionam como auxiliares de identificação de interesses dos utilizadores

Atributos da entidade Hashtag:

- palavra - chave primário para distinguir as hashtags;
- hashtagid;

Relações da entidade Hashtag:

- As hashtags pertencem aos perfis e às bookmarks;

6 - Conclusão

Em conclusão, as decisões feitas em termos de software para a resolução deste projeto podem não ter sido as mais adequadas mas foram tomadas com base nas competências que a equipa de implementação já trazia adquiridas do seu passado acadêmico, tentado ao máximo usar os softwares em que eles apresentassem o máximo de conforto. Neste caso sendo o Django a escolha mais apropriada, já que a equipa tinha toda formação em python e pouca experiência com bases de dados. As restantes escolhas já tinha sido feitas devido a necessidade de corresponder aos requisitos apresentados pelo o cliente.

Posto isto, esperemos que com este documento e com os diagramas nele inseridos qualquer membro da nossa equipa consiga facilmente compreender a estrutura, funcionalidades, e softwares utilizados na concretização deste projeto desenvolvida pela equipa LabSync