INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Rede Social de Voluntariado

Relatório final

Projeto e Seminário
Semestre de Verão 2019/2020
Licenciatura em Engenharia Informática e de Computadores

Autores:

Guilherme Allen
N.º 43571

Leonardo Martins
N.º 43591

Orientador: Nuno Leite

Setembro, 2020

Resumo

Nos dias de hoje, o voluntariado é uma atividade que ganha cada vez mais destaque, pois promove não só o enriquecimento da sociedade como também a solidariedade e a valorização do meio-social. A participação neste tipo de ações permite ao voluntário o desenvolvimento de competências como a liderança e o trabalho em equipa, as quais são valorizadas no meio profissional.

A divulgação e inscrição nas mesmas é realizada normalmente através de redes sociais dedicadas a conexões sociais (por exemplo, Facebook, Twitter, Google+), que não são vocacionadas para este tipo de ações, e em *websites*, o que leva a uma descentralização da informação.

O presente projeto visa abordar este problema através do desenvolvimento de uma rede social orientada ao voluntariado, onde voluntários e organizações coabitam, dando-se foco à divulgação e inscrição neste tipo de ações mas mantendo os atributos típicos de uma rede social (interação entre utilizadores). O sistema é composto por uma REST API (*Javascript*) e duas aplicações cliente: uma aplicação *mobile* (*Android*) orientada aos voluntários e uma aplicação *web* (*React*) orientada às organizações.

Palavras-chave: voluntariado, ações de voluntariado, *web* API, aplicação *mobile*, aplicação *web*.

Abstract

Nowadays, volunteering is an activity that has been gaining more prominence. Being a volunteer promotes not only the enrichment of society but also solidarity and the value of social environments. The participation in these activities allows the volunteer to develop skills such as leadership and teamwork, which are valuable in a professional environment.

The divulgation and enrolment in these actions is usually done through social networks, which aren't developed with this intent in mind, and in websites, which leads to a decentralization of information.

Our project looks to solve these issues through the development of a social network focused on volunteering, where volunteers and organizations coexist, giving focus to the divulgation and enrolment in these actions while maintaining the typical aspects of a social network (interaction between users). The project is composed of a REST API (Javascript) and two client applications: a mobile app (Android), for the volunteers, and a web app (React), for the organizations.

Keywords: volunteering, volunteering activities, web API, mobile app, web app.

Índice

1 Introdução **1**
1.1 Organização do relatório 2

2 Formulação do problema **3**
2.1 Problema Estudado 3
2.1.1 Objetivos Gerais 3
2.2 Estado da Arte 3
2.3 Análise 4
2.4 Requisitos Funcionais 4

3 Modelo de Arquitetura **7**
3.1 REST API 8
3.2 Aplicação *Mobile* 9
3.3 Aplicação *Web* 9
3.4 Tecnologias e ferramentas 9

4 API **11**
4.1 API 11
4.2 Serviço 12
4.3 Repositórios e acesso a base de dados 12
4.4 Base de Dados 12
4.5 Autenticação 13
4.6 Imagens 13
4.7 Paginação e limitação de resultados 13
4.8 Dados e imagens de utilizadores 13
4.9 Documentação e definição da API 14

5 *Mobile App* **15**
5.1 Navegabilidade e utilização da *App* 15
5.2 Sessão 16
5.3 Arquitetura 17
5.3.1 *User Interface* 17
5.3.2 API 18
5.3.3 Implementação de Sessão 18
5.3.4 Modelo 19

6 *Web App* **21**
6.1 Utilização da web app 21
6.2 Arquitetura 22
6.2.1 API 22
6.2.2 Componentes 23
6.2.3 Classe Principal 24
6.3 *Deployment* da aplicação 24

7 Conclusão **25**
7.1 Trabalho futuro 26

8 Anexos **29**
8.1 API - Lista de *endpoints* 29

Lista de Figuras

1 Modelo descentralizado de divulgação de voluntariado 1
2 Conceito do projeto 2
3 Tecnologias do *stack* MERN (MongoDB, Express.js, React.js, Node.js) 7
4 Modelo de arquitetura 8
5 Diagrama de arquitetura da API 11
6 Grafo de navegação 16
7 Representação do ecrã "Voluntários" no estado não autenticado (esquerda) e no estado autenticado (direita) como o utilizador "Utilizador exemplo". 17
8 Arquitetura do sub-módulo API 18
9 Interface de autenticação 21
10 Interface de autenticação 22
11 Exemplo de tipos de componentes na página dos eventos 23
12 Lista de *endpoints* 29

Lista de acrónimos

API Application Program Interface

CRUD Create, Read, Update and Delete

CSS Cascading Style Sheets

DNS Domain Name System

HTML HyperText Markup Language

HTTP HyperText Transfer Protocol

HTTPS HyperText Transfer Protocol Secure

JSON JavaScript Object Notation

JVM Java Virtual Machine

MEAN MongoDB, Express.js, Angular.js and Node.js

MERN MongoDB, Express.js, React.js and Node.js

NPM Node Package Manager

REST Representational State Transfer

SO Sistema Operativo

1 Introdução

Nos dias de hoje, o voluntariado é cada vez mais praticado na nossa sociedade. Segundo um estudo realizado pelo INE - Instituto Nacional de Estatística, em 2019, cerca de 6,4% da população portuguesa realiza trabalho voluntário, uma percentagem que cresceu ligeiramente face aos resultados obtidos em 2012 (5,9%) [1].

Segundo o Diário da República, o trabalho voluntário, ou voluntariado, é definido da seguinte forma:

"O conjunto de ações de interesse social e comunitário realizadas de forma desinteressada por pessoas, no âmbito de projetos, programas e outras formas de intervenção ao serviço dos indivíduos, das famílias e da comunidade desenvolvidos sem fins lucrativos por entidades públicas ou privadas." [2]

Cabe assim ao voluntário (pessoa que realiza o voluntariado) e entidades o papel fulcral na sociedade de tentar enriquecer a mesma sem qualquer contrapartida.

Para os voluntários, a participação em ações de voluntariado permite a obtenção de competências multi-disciplinares que são valorizadas no mundo profissional, e como tal, cada vez mais empresas dão valor a candidatos que participam nestas ações.

Atualmente, a candidatura ao voluntariado é efetuada através de múltiplas plataformas, como redes sociais e *websites*, algo que descentraliza estes serviços porque cada organização usa o seu próprio modelo (Figura 1).

Figura 1: Modelo descentralizado de divulgação de voluntariado

O presente projeto tem como objetivo desenvolver uma rede social com foco no voluntariado. A plataforma proposta irá disponibilizar às entidades organizadoras a possibilidade de divulgar e organizar estas ações, e aos voluntários, serviços que facilitam aos mesmos manterem-se informados e participarem nas ações do seu interesse (Figura 2).

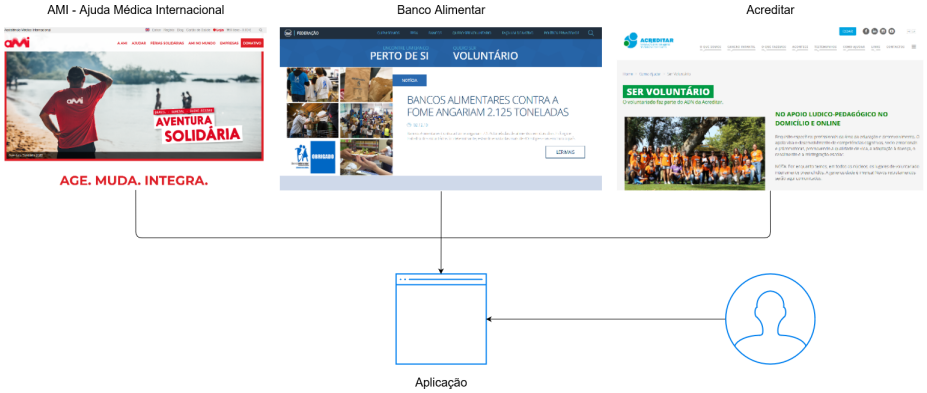

Figura 2: Conceito do projeto

1.1 Organização do relatório

O restante documento encontra-se organizado em seis capítulos, descritos de seguida e pela respetiva ordem.

- Formulação do problema, onde são definidos os objetivos gerais da plataforma e apresentadas outras plataformas similares. A partir das conclusões tiradas, é realizada uma análise, são definidos os requisitos funcionais do projeto e é escolhida a sua arquitetura.
- Modelo de arquitetura, capítulo responsável por discutir a arquitetura geral da plataforma e apresentar os seus módulos principais e as tecnologias utilizadas no desenvolvimento da mesma;
- Web API, onde é descrito o funcionamento da API usada pelas aplicações *web* e móvel desenvolvidas;
- Mobile App, onde se aborda a implementação da aplicação cliente orientada ao voluntário;
- Web App, onde se apresentam detalhes relativos ao funcionamento da aplicação cliente orientada à organização;
- Conclusão, onde são apresentadas as conclusões tiradas após o desenvolvimento do projeto.

2 Formulação do problema

No presente capítulo, são introduzidos os elementos base que definem o sistema a desenvolver. Na secção 2.1, o problema estudado no projeto é introduzido em maior detalhe. A secção 2.2 apresenta os trabalhos relacionados com o presente projeto. Na secção 2.3 é realizada uma análise das vantagens e desvantagens das propostas existentes. Finalmente, a secção 2.4 descreve os requisitos funcionais do sistema.

2.1 Problema Estudado

A interação voluntário-organização é tipicamente feita através de dois tipos de plataformas: redes sociais (de conexões sociais) e *websites* de organizações.

As redes sociais orientadas a conexões de pessoas, como por exemplo Facebook, Twitter, ou Google+, por não serem, por desenho, vocacionadas para ações de voluntariado, apresentam algumas limitações de utilização, como fraca filtragem de informação e impossibilidade de integração de múltiplas plataformas de voluntariado na mesma rede social.

Por norma, cada organização tem o seu próprio *website*, algo que complica o processo de navegação do voluntário, caso este esteja interessado em colaborar com múltiplas associações de voluntariado.

2.1.1 Objetivos Gerais

O projeto tem como objetivo desenvolver uma plataforma onde é disponibilizada informação relativa a ações de voluntariado, desde eventos existentes a perfis de organizações. A plataforma a desenvolver deve também auxiliar o processo de candidatura/inscrição nas ações a levar a cabo e promover a interação entre utilizadores.

A seguir descrevem-se duas plataformas que possuem objetivos semelhantes aos do presente projeto.

2.2 Estado da Arte

A Bolsa de Voluntariado [3] é um projeto lançado em 2006 pela associação ENTRAJUDA com o objetivo de facilitar a procura de trabalho voluntário.

Este objetivo é concretizado através duma plataforma *web* que serve de ponto de encontro entre a procura e oferta de oportunidades de voluntariado. A plataforma permite consultar ações que irão decorrer, oferecendo ainda a possibilidade aos utilizadores de as filtrarem consoante os seus interesses e visa também facilitar o processo de candidatura às mesmas.

A plataforma *Online Volunteering* [4], desenvolvida pela UN (*United Nations*) e lançada em 2000, é uma plataforma que, através do voluntariado *online*,

pretende reunir voluntários de múltiplas origens de maneira a auxiliarem na resolução de desafios tecnológicos das mais variadas áreas.

Esta aplicação permite a filtragem das oportunidades consoante a área de interesse e também auxilia o processo de candidatura às mesmas.

2.3 Análise

Apesar destas plataformas disponibilizarem informação relativa a oportunidades de realizar trabalho voluntário e das mesmas auxiliarem a inscrição nestas ações, estas não promovem a interação entre utilizadores, que é fulcral para o crescimento de uma comunidade solidária e que leva ao aumento do número de participantes em ações de voluntariado.

Outra abordagem possível seria o desenvolvimento de uma ferramenta que aplicasse técnicas de *web scraping*. Contudo, o desenvolvimento de uma ferramenta desta natureza depende da existência de fontes de informação externas e também não cumpre com a necessidade de haver interação entre utilizadores.

Uma rede social com foco em ações de voluntariado cumpre todos os objetivos descritos. Contudo, é necessário haver a migração de utilizadores para esta plataforma, algo que não é possível garantir.

Dadas as soluções apresentadas e suas vantagens/desvantagens, foi tomada a decisão de desenvolver uma rede social de voluntariado.

2.4 Requisitos Funcionais

O sistema a desenvolver tem os seguintes requisitos funcionais:

- permitir a voluntários registarem-se, criarem um perfil e interagir com a plataforma (através da criação de *posts* e sinalização de interesse em eventos);
- possibilitar às organizações solicitarem o registo na plataforma e também permitir às mesmas realizarem *posts* e criarem eventos;
- mostrar às organizações os voluntários interessados nos seus eventos e disponibilizar um contacto dos mesmos (por exemplo, e-mail);
- garantir aos utilizadores da plataforma o acesso a interações como o seguimento de utilizadores, gostarem de *posts*, entre outros.

Levando em consideração os requisitos funcionais enumerados, foi tomada a decisão de elaborar o projeto em três componentes:

1. uma **Web API**, responsável por implementar as funcionalidades necessárias para as aplicações cliente terem o comportamento pretendido. Este módulo foi desenvolvido face à necessidade de expôr os dados de forma comum a todas as aplicações cliente.

2. Uma **aplicação móvel** (*Android*) orientada aos voluntários, onde os mesmos podem interagir com a plataforma. A decisão de desenvolver uma aplicação móvel para este sistema operativo foi tomada devido ao mesmo ser o mais comum no mercado móvel atualmente.

3. Uma **aplicação *web*** desenvolvida para as organizações interagirem com a plataforma, sendo que esta opção foi tomada para que múltiplos utilizadores possam gerir mais facilmente o perfil de uma organização.

3 Modelo de Arquitetura

Este capítulo começa por introduzir o *stack* tecnológico sobre o qual o projeto foi desenvolvido. O mesmo contém também um secção atribuída a cada módulo do projeto e por fim refere as tecnologias usadas no mesmo.

O modelo de arquitetura do nosso projeto, apresentado na Figura 4, é constituído por três módulos principais: uma REST API, e duas aplicações cliente: uma orientada à plataforma *mobile* Android e outra desenvolvida para ser usada num *browser*.

Tendo em conta os módulos constituintes do projeto, o desenvolvimento do mesmo seguirá o *stack* MEAN *stack* [5] (MongoDB, Express.js, Angular, Node.js), alternando a tecnologia utilizada para desenvolver o *front-end* para React em vez de Angular (também conhecido pelo MERN *stack*, referido na Figura 3). Foi escolhido este *standard* pelas seguintes razões:

- familiaridade dos autores com algumas destas tecnologias (como Express.js e Node.js);
- *stack* utilizado no desenvolvimento de múltiplas aplicações, o que leva a existência de uma grande quantidade de recursos, como documentação e exemplos;
- todas estas tecnologias têm em comum características que as tornam apelativas de usar conjuntamente, como por exemplo o facto da utilização de JSON ser transversal entre todas;
- todas as ferramentas associadas a esta pilha tecnológica são *open-source*.

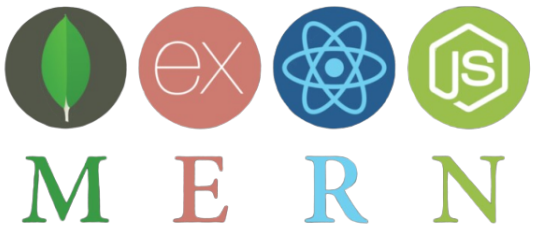

Figura 3: Tecnologias do *stack* MERN (MongoDB, Express.js, React.js, Node.js)

Relativamente à API, esta estabelecerá *endpoints* onde será possível executar pedidos HTTPS de maneira a suportar autenticação e operações na infraestrutura (criação de perfil, "seguimento" de organização, inscrição em ação de voluntariado, etc.), constituindo o *back-end* do projeto.

Relativamente ao *front-end*, foram desenvolvidas duas aplicações cliente:

- um cliente *mobile*, para a plataforma Android, usado pelos voluntários. Nesta interface será possível efetuar por parte do utilizador as operações de uso da plataforma usuais: criação de um perfil, visionamento de um *feed* de *posts* efetuados pelas organizações seguidas, entre outras;
- um cliente *browser*. Esta aplicação é direcionada às organizações e terá a finalidade de permitir às mesmas realizar *posts*, criar e gerir ações de voluntariado, etc.;

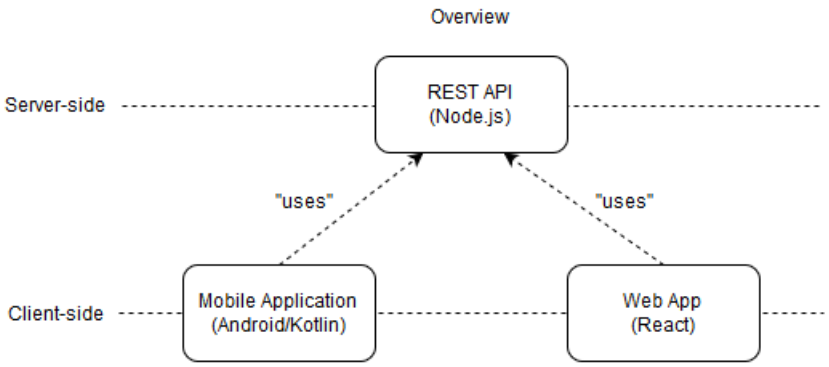

Figura 4: Modelo de arquitetura

3.1 REST API

A REST API, definida como primeira fase do projeto, constituirá o *server-side* do mesmo. É pretendido que este módulo seja completamente independente dos outros, tendo como responsabilidade trabalhar como fonte de dados para os outros componentes (*client-side*).

A tecnologia utilizada para desenvolver este componente foi Node.js em conjunto com vários *packages* do NPM (*node package manager*), como o Express e o Passport. Esta escolha é justificada por múltiplas razões:

- domínio dos autores na linguagem Javascript e em vários módulos do NPM;
- popularidade da ferramenta, algo que simplifica o processo de desenvolvimento do componente devido à existência de grande quantidade de recursos sobre a mesma;
- existência de suporte neste meio de execução de ferramentas para auxiliar o acesso à base de dados escolhida.

O servidor será também responsável por hospedar a base de dados, que irá funcionar no motor MongoDB. A escolha deste serviço foi efetuada devido à fácil integração do mesmo com Node.js e devido ao facto deste ter um modelo de dados baseado em documentos JSON, algo que simplifica a inserção e pesquisa sobre os mesmos.

3.2 Aplicação *Mobile*

A aplicação *mobile* será desenvolvida para a plataforma Android em Kotlin. A mesma irá seguir os princípios definidos pelo Android Jetpack, que disponibiliza ferramentas e bibliotecas que auxiliam o desenvolvimento da aplicação. As razões que levaram a esta decisão foram:

- familiaridade dos autores com esta linguagem de programação e ferramentas (Android Jetpack);
- a nível de quota de mercado dos sistemas operativos de dispositivos móveis, o Android é o mais prevalente;
- o Kotlin é uma das linguagens oficiais para desenvolvimento de aplicações móveis para Android.

3.3 Aplicação *Web*

A aplicação *web* será desenvolvida em React [6]. Esta tecnologia foi escolhida devido a:

- familiaridade dos autores com esta ferramenta;
- quota de mercado (relativamente às *frameworks* de Javascript utilizadas para o desenvolvimento de aplicações *web*) significativa, sendo atualmente a tecnologia mais usada [6];
- fácil integração com as outras ferramentas do projeto.

3.4 Tecnologias e ferramentas

As seguintes tecnologias irão ser utilizadas durante o desenvolvimento do projeto:

- **Javascript**: Principal linguagem para programação *client-side* em *browsers*; Esta é tipicamente utilizada em conjunto com ferramentas como o HTML e CSS para implementar a funcionalidade de uma página *web*;
- **Node.js**: Interpretador e ambiente de execução para Javascript normalmente utilizado para executar código sem ser num cliente *browser*;
- **NPM**: *package manager* do Javascript/Node.js;

- **Express**: *web framework* para Node.js. Auxilia o processo de *routing* e definição de *endpoints*, encapsulando aspetos do HTTP, para tornar mais fácil o desenvolvimento de *Web* APIs;
- **Passport**: *Middleware* de autenticação usado em conjunto com o Express para simplificar o processo de autenticação e gestão de sessões de utilizadores;
- **MongoDB**: Base de dados noSQL baseada em documentos JSON. Tipicamente integrada com Javascript devido à natureza dos seus documentos;
- **Android**: SO *open source* para dispositivos móveis desenvolvido pela Google. Atualmente, cerca de 75% dos dispositivos móveis usam este SO;
- **Kotlin**: Linguagem de programação desenvolvida pela JetBrains que compila para a JVM (*Java Virtual Machine*). Atualmente, o Kotlin é uma das linguagems oficiais para desenvolvimento de aplicações Android;
- **Android Jetpack**: Conjunto de ferramentas e bibliotecas que auxiliam a implementação e desenvolvimento de software para o sistema operativo móvel Android;
- **Volley**: Biblioteca *open-source* desenvolvida para simplificar a realização de pedidos HTTP no ambiente Android;
- **Glide**: *Biblioteca* utilizada para efetuar o carregamento de imagens em aplicações Android;
- **React**: Biblioteca *open-source* de Javascript usada para desenvolver aplicações *web*.
- **Bootstrap**: Biblioteca *open-source* que auxilia o desenvolvimento de interfaces de utilizador, tipicamente utilizada em ambientes *web*.
- **Nginx**: Servidor *web* utilizado para hospedar plataformas.
- **Duck DNS**: Serviço dinâmico de DNS, que permite atribuir um nome de domínio à plataforma gratuitamente.
- **CertBot**: Cliente open source usado para obter certificados SSL fornecidos gratuitamente pela autoridade *Let's Encrypt*.

4 API

A *Web* API (Figura 5) é composta por dois módulos principais (API e Service) e um conjunto de módulos auxiliares. Enquanto que a API lida com pedidos e respostas HTTP, o *Service* é responsável por implementar a lógica do sistema. Existem ainda os *repositórios* (sendo que cada um se responsabiliza por implementar a sua própria lógica de acesso à base de dados) e o *índice de imagens* (responsável por tratar todas as operações associadas às mesmas). [7]

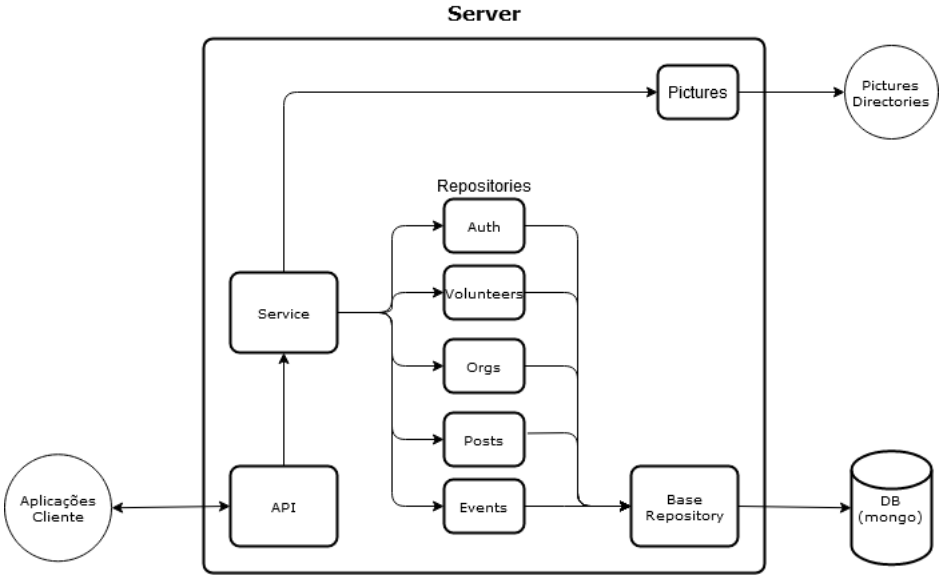

Figura 5: Diagrama de arquitetura da API

4.1 API

Este módulo é responsável por definir os *endpoints* e lidar com a receção e envio dos pedidos/respostas HTTP. Cada *endpoint* tem associada uma função que efetua a chamada ao módulo de serviço correspondente à operação solicitada pelo utilizador.

Encontra-se anexado a este relatório a lista de endpoints da API (anexo 1). Foram implementados os *endpoints* que os autores consideraram necessários para garantir o funcionamento da plataforma. É de notar que todos os pedidos definidos têm o preâmbulo */api* e que os que começam por */auth* necessitam de autenticação prévia por parte do cliente da API.

Para além das funções já referidas, são também definidos (e utilizados) *middlewares* nesta camada de maneira a garantir o cumprimento da necessidade de

autenticação aquando de certas operações. A API é ainda responsável por transformar os parâmetros dados pelo cliente através do *URL*, da *query string* e do corpo do pedido (e do objecto sessão quando aplicável) para objetos conhecidos pelo *serviço*.

4.2 Serviço

O módulo *Service* cumpre a função de implementar a lógica da aplicação, isto é, garantir que as operações chamadas a partir da API se materializam em mudanças no modelo deste componente, seja a nível de base de dados ou tratamento de operações relativamente às imagens, entre outros.

Este módulo tem as seguintes responsabilidades:

- verificar a validade dos parâmetros fornecidos pela API;
- executar as chamadas necessárias aos módulos adjacentes (*repositories* e *pictures*) de maneira a cumprir a operação solicitada (quer esta seja a inserção em base de dados ou a recolha de informações para verificar a lógica necessária);
- implementar a lógica da aplicação, isto é, quando necessário, efetuar as verificações necessárias e gerar os objetos a colocar/alterar na base de dados.

4.3 Repositórios e acesso a base de dados

Para cada índice na base de dados, existe uma variação de implementação de *Repository*. Estas implementações fornecem métodos específicos para inserir/modificar/apagar entradas dos índices (coleção de documentos, como por exemplo: voluntários ou organizações) aos quais se referem. Estes repositórios, por sua vez, têm acesso a uma instância de *BaseRepository* gerada especificamente para si (isto é, inicializada com argumentos personalizados consoante o índice).

BaseRepository é uma infra-estrutura implementada de maneira a ser genérica para todos os índices e que fornece implementações das operações CRUD usuais como também a possibilidade de gerar os índices de pesquisa necessários para efetuar pesquisas por valor de campos nas tabelas.

4.4 Base de Dados

O motor de base de dados escolhido foi o MongoDB que é um motor *noSQL*. Este apresenta um modelo não relacional, tipicamente utilizado no desenvolvimento de redes sociais. Esta escolha é justificada pelas seguintes razões:

- flexibilidade. Tendo em conta que as plataformas desta natureza têm tipicamente um crescimento rápido, é necessário que a implementação de novas funcionalidades seja um processo ágil. Num modelo *SQL*, por vezes é despendido tempo de implementação a efetuar estas alterações, algo que não é necessário neste caso porque os documentos não têm um esquema fortemente definido, podendo ser inseridos dinamicamente (dois documentos com características diferentes podem ser inseridos no mesmo índice);

- integração com Javascript. Os documentos inseridos nesta base de dados têm um formato JSON, algo que se integra facilmente com esta tecnologia.

4.5 Autenticação

A componente autenticação deste módulo foi desenvolvida através da ferramenta Passport.js. Esta é tipicamente utilizada em conjunto com Express e permite a fácil implementação dum objeto *sessão* através do uso de *cookies*.

Aquando do pedido de autenticação bem sucedido por parte de um cliente da API, são guardadas um conjunto de características identificadoras da sessão do mesmo para que se possa personalizar a sua experiência, e neste caso, permitir ao mesmo que possa fazer o carregamento de imagens, criar um *post*, entre outros.

4.6 Imagens

De maneira a implementar na sua totalidade um conceito de rede social, nesta aplicação é permitido o carregamento de imagens para a plataforma. Estas são divididas por índices, consoante a sua classe (aquilo que elas referem) e guardadas.

O carregamento de imagens para esta plataforma é apenas possível para utilizadores autenticados, no entanto, o descarregamento das mesmas é possível a qualquer cliente da API.

4.7 Paginação e limitação de resultados

Todas as operações da API que tenham como resultado um conjunto de entidades (por exemplo, um conjunto de *posts*), retornam por omissão um número fixo de resultados. Contudo, a nossa API suporta paginação e *skipping* (isto é, saltar resultados) através dos parâmetros *limit* e *skip* da *query string*.

4.8 Dados e imagens de utilizadores

Dada a natureza deste projeto, todos os dados (perfis/fotos/*posts*) serão públicos para todos, e como tal, aquando do registo nas aplicações cliente, os utilizadores das mesmas serão sensibilizados relativamente à visibilidade dos seus dados.

4.9 Documentação e definição da API

Toda a documentação sobre as regras e indicações relativamente à utilização desta API encontram-se na *wiki* do repositório do projeto, sendo que o acesso será avaliado quando solicitado aos autores. Esta *wiki* contém informações relativamente à definição mais explícita dos *endpoints*, erros, entre outros.

5 *Mobile App*

Este capítulo descreve o desenvolvimento e implementação da aplicação móvel. O mesmo é composto por uma introdução, seguindo-se a apresentação da navegação na aplicação e a arquitetura da mesma. Por fim, discute-se a implementação de cada sub-módulo da arquitetura.

A aplicação móvel é uma interface de utilizador desenvolvida em *Android* com o intuito de ser utilizada por voluntários.

Este módulo contém alguns requisitos chave que são necessários para garantir a usabilidade da mesma por parte dos seus utilizadores. Alguns dos requisitos são:

- garantir que possam ver os *posts* e eventos que estão registados na plataforma;
- possibilitar que possam gostar *posts*, seguir utilizadores e registarem-se em eventos (ou seja, implementar a interação de uma rede social);
- existência do conceito sessão de maneira a terem uma experiência personalizada, e possibilitar que possam editar o seu perfil, efetuar *posts*, entre outros.

Tal como já foi referido anteriormente, foi tomada a decisão de implementar esta aplicação no sistema operativo Android seguindo as orientações dadas pelo Android Jetpack e um conjunto de bibliotecas auxiliares.

5.1 Navegabilidade e utilização da *App*

A interface gráfica foi desenhada para ser simples de utilizar e o mais intuitiva possível. O grafo de navegação da mesma é ilustrado na Figura 7.

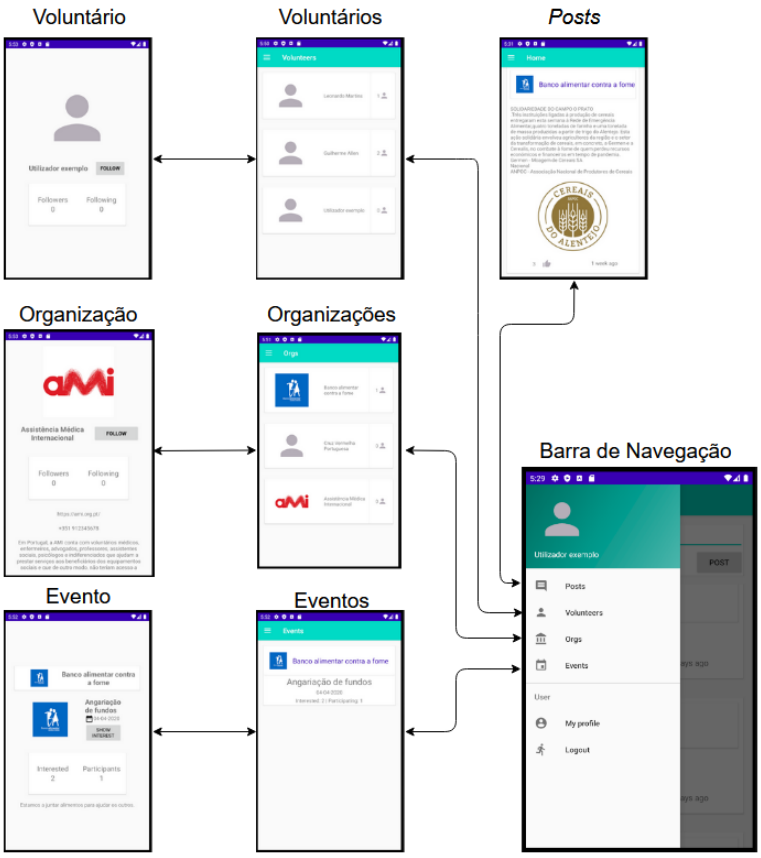

Figura 6: Grafo de navegação

Foram desenvolvidas quatro vistas principais de pesquisa (*posts*, voluntários, organizações e eventos) que apresentam os dados existentes na plataforma. Existem também vistas detalhadas para estes mesmos índices (exceto *posts*) para que o utilizador possa ver os detalhes destes dados.

5.2 Sessão

De maneira a garantir uma experiência personalizada para cada cliente deste módulo, foram desenvolvidos ecrãs e mecanismos de de registo e autenticação de clientes.

A partir do momento que um cliente da aplicação se encontre autenticado, certas operações como "gostar" um *post* ou seguir um utilizador passam a estar disponíveis e determinados ecrãs irão ter uma vista personalizada, tal como exemplificado na Figura 8.

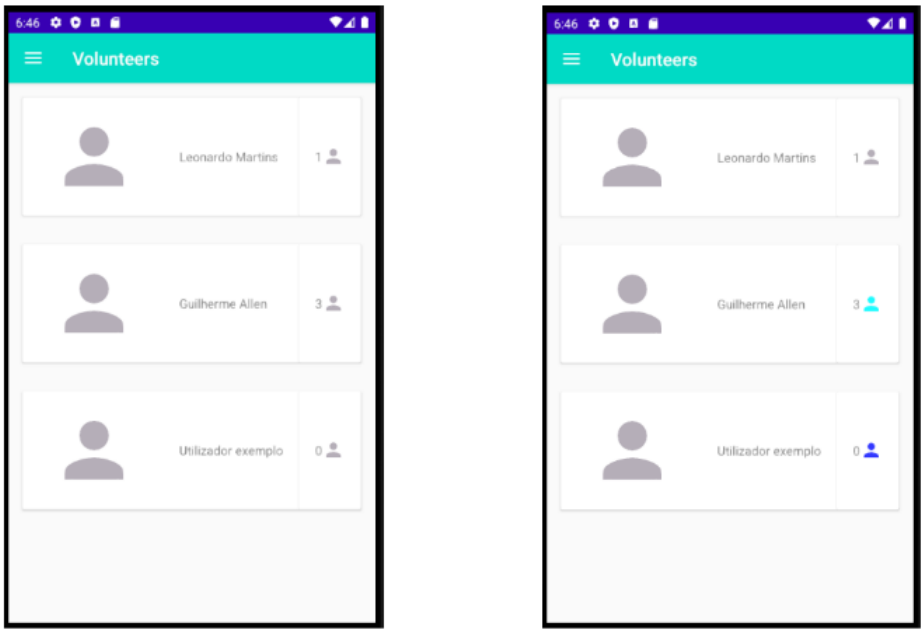

Figura 7: Representação do ecrã "Voluntários" no estado não autenticado (esquerda) e no estado autenticado (direita) como o utilizador "Utilizador exemplo".

5.3 Arquitetura

A arquitetura da aplicação é constituída por três sub-módulos principais: UI (User Interface), API e Modelo.

Enquanto que a UI é responsável por conter as implementações dos mais variados aspetos da interface de utilizador (como atividades, fragmentos, navegação e outros elementos), a API trata a comunicação entre a aplicação e a Web API. Por fim, o Modelo define a estrutura dos dados fornecidos pela API e utilizados na UI assim como a implementação de uma *cache* de maneira a partilhar estes dados e reduzir o número de pedidos efetuados à Web API.

5.3.1 *User Interface*

Este sub-módulo engloba todos os componentes do projeto que são responsáveis por representar os dados da aplicação. Este é constituído por:

- implementações de ecrãs, quer sejam atividades ou fragmentos. É de notar que para cada um destes existe associado um objeto ViewModel que é responsável por manter o estado dos elementos do ecrã e também por interagir com a API quando necessário;

- adaptadores, infraestruturas responsáveis por auxiliar o processo de transformação de dados brutos numa representação dos mesmos. Estes adaptadores são utilizados em conjunto com o componente RecyclerView - utilizado para apresentar uma lista de elementos (como voluntários ou *posts*) ao utilizador;
- Outros componentes utilitários, como por exemplo uma estrutura que auxilia o carregamento de imagens para a representação das mesmas.

5.3.2 API

O sub-módulo API (Figura 9) serve como *proxy* entre a aplicação e a Web API. Este dispõe de um serviço que disponibiliza todas as rotas acessíveis pelos voluntários e sobre o mesmo é possível solicitar a realização de pedidos à API (como a obtenção de voluntários ou o seguimento, por parte do utilizador autenticado, de uma organização). O mesmo é assíncrono e funciona à base de *callbacks*.

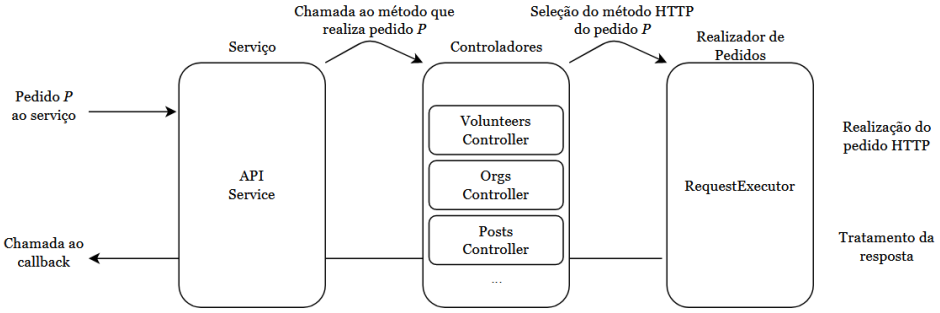

Figura 8: Arquitetura do sub-módulo API

O realizador de pedidos trata da implementação dos métodos HTTP suportados pela Web API (GET, POST, PUT, DELETE) de forma genérica, de maneira a que o mesmo possa ser utilizado pelos controladores. Os pedidos HTTP são efetuados usando a biblioteca *Volley*.

Quando o pedido é concluído, é chamado o *callback* definido pela estrutura que chamou o serviço, que consome a resposta do mesmo, quer esta contenha dados ou apenas sinalize sucesso. A estrutura que efetua a chamada ao serviço é também responsável por definir o *callback* a executar em caso de ocorrer um erro durante o pedido.

5.3.3 Implementação de Sessão

Tendo em conta que o conceito de sessão foi implementado na Web API através da tecnologia Passport.js (que funciona à base de *cookies*), foi necessário redefinir algumas infra-estruturas da biblioteca Volley de maneira a que, quando o utilizador estiver autenticado na aplicação, os pedidos efetuados pela mesma

contenham a informação necessária para a Web API conseguir reconhecer o utilizador.

De maneira a que a representação dos vários ecrãs da aplicação seja dinâmica consoante a sessão do utilizador, é definido e utilizado um objeto Sessão, acessível por todas as classes do projeto, que contém informações relativamente ao voluntário autenticado.

5.3.4 Modelo

O Modelo é responsável por transformar os dados provenientes do sub-módulo API para conjuntos de DTO (*Data Transfer Object*) que irão ser usados pelo sub-módulo UI para efetuar a sua representação.

Este sub-módulo contém também a definição de uma estrutura que realiza *caching* dos dados da aplicação para diminuir o número de pedidos a realizar à API.

6 *Web App*

Neste capítulo é abordado o desenvolvimento da aplicação *web*. É realizada uma introdução, apresentando os objetivos e funcionalidades da mesma. São mostrados detalhes relativos à utilização da aplicação e, de seguida, apresenta-se o modelo de arquitetura utilizado no desenvolvimento da mesma. Por fim, refere-se como aceder à aplicação e define-se o seu processo de *deployment*.

A aplicação *web* é responsável por estabelecer uma interface sobre a qual as organizações podem interagir com a plataforma, disponibilizando às mesmas ferramentas que possibilitam a realização de operações como por exemplo a criação de *posts* ou a realização de pesquisas sobre a plataforma.

Foram definidos alguns requisitos chave no início da conceptualização da aplicação, como por exemplo:

- disponibilizar meios para consultar os *posts*, eventos e outros utilizadores da plataforma e interagir com os mesmos;
- permitir às organizações editarem o seu perfil;
- possibilitar que as mesmas possam criar e editar *posts* e eventos;
- apresentar contactos de voluntários interessados em eventos pertencentes à organização autenticada.

Numa fase inicial do projeto, foi considerada a opção de desenvolver a aplicação *web* usando a *framework* Angular.js. Contudo, após nova avaliação, optou-se por utilizar a tecnologia React. Esta alteração foi efetuada após verificar que React é a tecnologia mais usada no mercado à data da realização do projeto.

6.1 Utilização da web app

Levando em consideração que este componente foi desenvolvido especificamente para organizações, a maioria das operações implicam que um utilizador organização esteja autenticado.

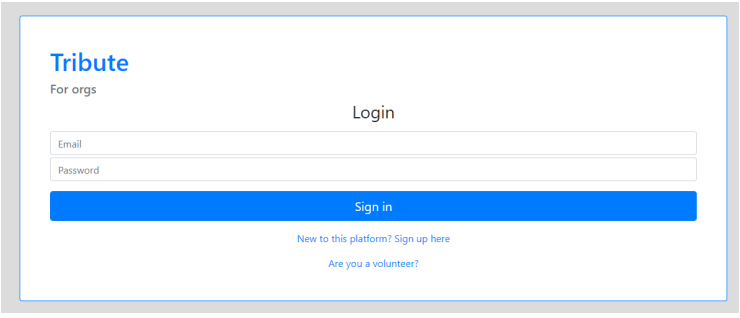

Figura 9: Interface de autenticação

Nesta versão da aplicação, é permitido que sejam registados utilizadores do tipo organização. Contudo, numa versão publicada da plataforma, o registo deste tipo de utilizadores seria realizado através do contacto direto dos mesmos com os gestores da aplicação de maneira a que apenas organizações fidedignas pudessem ter um perfil na plataforma.

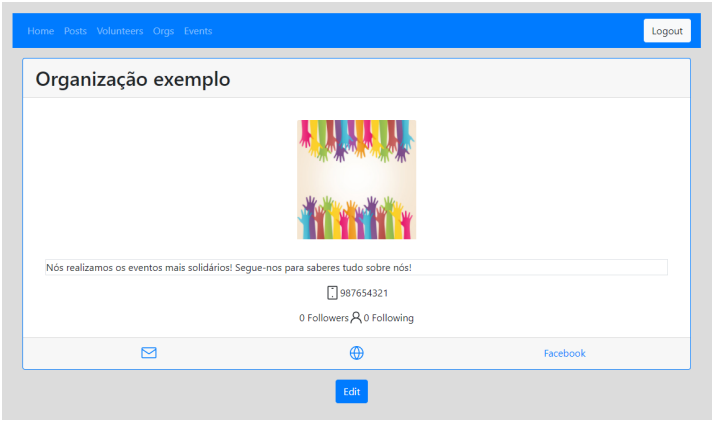

Figura 10: Interface de autenticação

Após autenticação na aplicação, é disponibilizado um painel principal onde é possível navegar entre as várias páginas da aplicação e realizar as operações pretendidas.

6.2 Arquitetura

A arquitetura da aplicação é composta principalmente por 2 módulos: API e Componentes e ainda a classe principal da aplicação.

Similar ao funcionamento do módulo com o mesmo nome na aplicação *mobile*, API disponibiliza operações que realizam pedidos HTTPS à API. Componentes contém a implementação de todos os componentes *react* apresentados ao utilizador, desde as páginas em si a componentes que são utilizados nestas.

A classe principal da aplicação é responsável não só por instanciar os serviços da API assim como definir o roteamento da aplicação *web*.

6.2.1 API

A módulo API é composto por um conjunto de serviços que disponibilizam operações que necessitam de realizar pedidos HTTPS à API para serem realizadas (como por exemplo a solicitação de *posts* ou a criação de um evento).

Para cada entidade (voluntários, organizações, *posts* e eventos) existe um serviço onde são definidas as operações possíveis de efetuar sobre a mesma.

Todos os serviços utilizam uma classe auxiliar que contém a implementação de como efetuar pedidos HTTPS consoante o seu método (neste caso, GET, PUT, POST e DELETE). Esta classe auxiliar contém também a implementação de um requests

6.2.2 Componentes

Tal como já referido, o módulo Componentes é responsável por tratar a estruturação e apresentação da interface da aplicação assim como lidar com operações de *input* por parte do utilizador. Como tal, este contém a definição de:

- **componentes página**. Estes componentes definem os sub-componentes que constituem a página (por exemplo, na figura 11, a página dos eventos é constituída por um formulário para criar um novo evento e a lista dos eventos existentes na plataforma);
- **componentes específicos** por página, como por exemplo, uma lista de *posts* ou um formulário para criar eventos;
- **componentes utilitários**, responsáveis por apresentar certos aspetos comuns da aplicação.

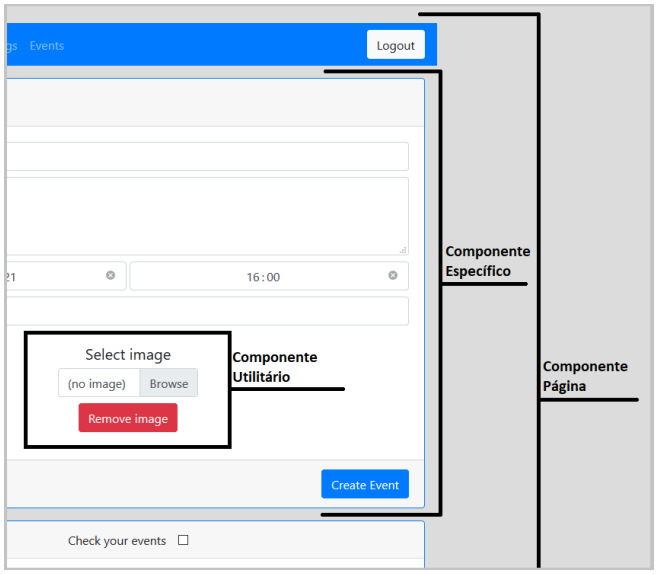

Figura 11: Exemplo de tipos de componentes na página dos eventos

6.2.3 Classe Principal

A classe principal da aplicação instancia os serviços usados pelos componentes para efetuar pedidos à API e define também o roteamento da aplicação *web*. No modo não autenticado, o acesso a quase todas as rotas da aplicação web é restringido. Apenas quando o cliente realiza autenticação com sucesso é permitido ao mesmo aceder às funcionalidades principais da aplicação.

6.3 *Deployment* da aplicação

A aplicação foi *deployed* utilizando uma máquina virtual do serviço *Compute Engine* da *Google Cloud Platform*. Foram instaladas as ferramentas necessárias para executar a aplicação nesta máquina (nomeadamente *Node.js*) e a mesma encontra-se instanciada numa das portas da VM.

De maneira a lidar com pedidos CORS (Cross-Origin Resource Sharing), na máquina onde é executada a aplicação está instanciado um servidor *web* Nginx. Este está configurado de maneira a redirecionar todos os pedidos que comecem com o preâmbulo /api para a máquina onde está *deployed* a *web* API. Todos os outros pedidos são redirecionados para a porta onde está a ser executada a aplicação *web*.

De maneira a associar um nome de domínio ao endereço IP da máquina onde está a ser executado o servidor *web* foi utilizada a plataforma *Duck DNS*, que permite reservar sem qualquer custo um número limitado de *domain names*.

Por fim, e utilizando as ferramentas disponibilizadas pela plataforma *Cert-Bot*, foi automaticamente emitido um certificado SSL e alterada a configuração do servidor *Nginx* de maneira a que a plataforma aceitasse apenas comunicações através do protocolo HTTPS, garantindo segurança ponto-a-ponto entre máquina cliente e servidor *web*.

A aplicação encontra-se acessível em `https://tribute-app.duckdns.org/`.

7 Conclusão

Este projeto trata o desenvolvimento uma rede social de voluntariado, sendo que para tal, foram implementados três módulos: uma *web* API, uma aplicação móvel e uma aplicação *web*. Para cada um destes módulos foi também necessário definir a sua arquitetura e, quando aplicável, o seu design gráfico.

Foram apontadas algumas conclusões durante a realização do componente API:

- apesar dos conhecimentos relativamente à implementação de uma API já obtidos ao longo da licenciatura, tivemos alguma dificuldade no desenho e definição de requisitos funcionais da API, dado o maior nível de complexidade comparado com projetos de disciplinas anteriores. Como tal, e ao longo do projeto, têm sido realizadas iterações à mesma de maneira que esta seja o mais adequada possível;
- a API, apesar de ser auto suficiente e não necessitar de outros módulos para funcionar, a mesma foi desenvolvida para comunicar em primeira mão com os outros módulos deste projeto. Como tal, é expectável que seja necessário definir *endpoints* adicionais consoante as necessidades das aplicações cliente, como por exemplo a definição de um *endpoint* que devolve os posts realizados por utilizadores que um certo utilizador autenticado segue.

Relativamente ao desenvolvimento da aplicação *mobile*, apresentam-se também algumas inferências.

- Ao longo do lecionamento de unidades curriculares que entram em contacto com o desenvolvimento deste tipo de aplicações (nomeadamente, Programação em Dispositivos Móveis) diretamente, é sempre dado um ênfase não ao aspeto e sim à funcionalidade da mesma. Contudo, e dada a natureza deste projeto, foi dado um cuidado acrescentado à interface de utilizador.

A utilização de *stacks* tecnológicos não impostos por unidades curriculares da licenciatura e sim selecionados pelos autores do projeto levou por vezes à necessidade dos autores consultarem não só o orientador do projeto como também artigos *web*, livros e outros recursos de maneira a tentar ultrapassar obstáculos encontrados na implementação do projeto.

Outra problemática encontrada ao longo do desenvolvimento do projeto é o facto dos autores estarem a desenhar a arquitetura dos componentes ao invés de seguir uma apresentada numa unidade curricular.

Contudo, os autores esperam que, dentro do período para a entrega do projeto, consigam concluir todos os requisitos funcionais e não funcionais do mesmo, produzindo uma plataforma que, com poucas alterações, poderia ser utilizada num contexto real e não apenas no contexto da elaboração de um trabalho final de curso.

7.1 Trabalho futuro

No estado atual do mesmo, ainda faltam concluir algumas funcionalidades do projeto, como o desenvolvimento da aplicação *web* e uma revisão geral sobre a aplicação móvel e a API de maneira a refinar o mesmo e prepará-lo para a entrega final.

Referências

[1] Instituto Nacional de Estatística. Inquérito ao trabalho voluntário, 2019.

[2] Assembleia da República. Decreto lei nº 71/98 de 3 de novembro.

[3] Entrajuda. Bolsa do voluntariado.

[4] United Nations. Online volunteering.

[5] Jeremy M Williams. What is the mean stack ?

[6] TechMagic. React vs angular vs vue.js — what to choose in 2020?

[7] Jim R. Wilson. *Node.js 8 the Right Way.*

8 Anexos

8.1 API - Lista de *endpoints*

		Voluntários
Método	**URL**	**Descrição**
GET	/volunteers?name=abc	Procura voluntários (possibilidade de procurar por nome)
GET	/volunteers/{volunteer_id}	Procura o voluntário com o *id* dado
PUT	/auth/volunteers/{volunteer_id}	Edita o perfil do voluntário. (autenticação de voluntário requerida)
PUT	/auth/volunteers/{volunteer_id}/follow	Segue o voluntário
		Organizações
Método	**URL**	**Descrição**
GET	/orgs?name=abc	Procura Organizações (possibilidade de procurar por nome)
GET	/orgs/{org_id}	Procura a organização com o *id* dado
PUT	/auth/orgs/{org_id}	Edita o perfil da organização (autenticação de organização requerida)
PUT	/auth/orgs/{org_id}/follow	Segue a organização
		Posts
Método	**URL**	**Descrição**
POST	/auth/posts	Cria um *post*
GET	/posts?owner_id=123	Procura os *posts* mais recentes (possibilidade de restringir a um *owner*)
GET	/posts/{post_id}	Procura o *post* com o *id* dado
PUT	/auth/posts/{post_id}	Edita o *post* (autenticação requerida)
PUT	/auth/posts/{post_id}/like	Adiciona um gosto (*like*) ao *post*
DELETE	/auth/posts/{post_id}	Apaga o post (autenticação requerida)
		Eventos
Método	**URL**	**Descrição**
POST	/auth/orgs/events	Cria um evento (autenticação de organização requerida)
GET	/events	Procura eventos
GET	/events/{event_id}	Procura o evento com o *id* dado
GET	/orgs/{org_id}/events	Procura eventos da organização com *id* que foi dado
PUT	/auth/posts/{post_id}	Edita o evento (autenticação de organização requerida)
PUT	/auth/orgs/events/{event_id}/interest	Coloca "interessado" no evento (autenticação de voluntário requerida)
PUT	/auth/orgs/events/{event_id}/participate?volunteer_id=123	O voluntário com o *id* dado passa a ser participante do evento(autenticação de voluntário requerida)
DELETE	/auth/orgs/events/{event_id}	Apaga o evento (autenticação de organização requerida)
		Imagens
Método	**URL**	**Descrição**
POST	/auth/images/{image_type}/{image_id	Faz post da imagem (necessita de permissões para tal)
GET	/images/{image_type}/{image_id}	Procura a imagem
		Autenticação e registo
Método	**URL**	**Descrição**
POST	/register	Realiza o registo
POST	/login	Inicializa a sessão
GET	/logout	Termina a sessão

Figura 12: Lista de *endpoints*