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 Arquitetura Geral 7
3.2 *Stack* Tecnológico 8
3.3 API 8
3.4 Aplicação *Mobile* 9
3.5 Aplicação *Web* 9
3.6 Tecnologias e ferramentas 10

4 API **13**
4.1 Conceptualização 13
4.2 Operações 14
4.3 Arquitetura 14
4.3.1 Controladores 15
4.3.2 Serviços 15
4.3.3 Repositórios 15
4.3.4 Autenticação 15
4.4 Base de dados 16
4.5 Implantação da API 16

5 *Mobile App* **17**
5.1 Navegabilidade e utilização da *App* 18
5.2 Sessão 19
5.3 Arquitetura 19
5.3.1 *User Interface* 20
5.3.2 API 20
5.3.3 Implementação de Sessão 21
5.3.4 Modelo 21

6 *Web App* **23**
6.1 Utilização da *Web App* 24
6.2 Arquitetura 25
6.2.1 API 25
6.2.2 Componentes 25
6.2.3 Classe Principal 26

6.3 Implantação da aplicação 27

7 Conclusão **29**
7.1 Trabalho futuro 30

8 Referências **31**

Lista de Figuras

1 Modelo descentralizado de divulgação de voluntariado. 1
2 Conceito do projeto. 2
3 Modelo de arquitetura. 7
4 Tecnologias do *stack* MERN (MongoDB, Express.js, React.js, Node.js) [6]. 8
5 Diagrama de arquitetura da API. 14
6 Grafo de navegação. 18
7 Representação do ecrã "Voluntários" no estado não autenticado (esquerda) e no estado autenticado (direita) para o utilizador "Utilizador exemplo". 19
8 Arquitetura do sub-módulo API. 20
9 Interface de autenticação. 24
10 Painel principal da aplicação. 24
11 Exemplo de tipos de componentes na página dos eventos. 26
12 Implantação da aplicação *web*. 27

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

IP Internet Protocol

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

SSL Secure Sockets Layer

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 organizadoras de ações de voluntariado (como organizações governamentais, empresas privadas e organizações sem fins lucrativos) 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).

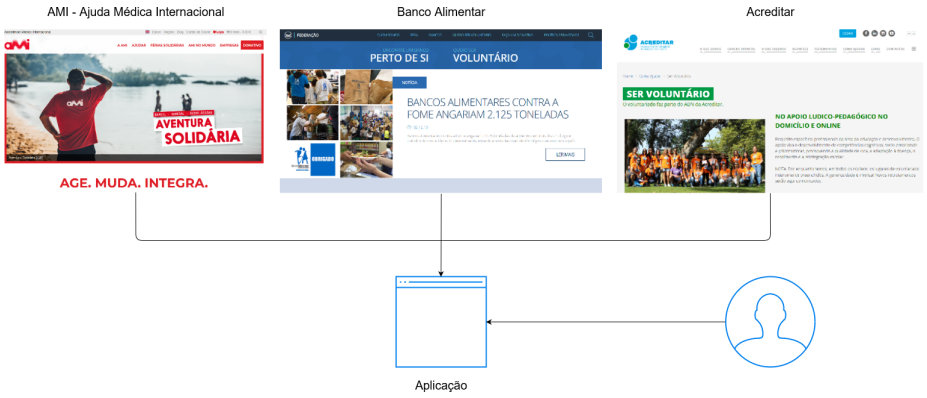

Figura 2: Conceito do projeto.

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. 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 Nações Unidas 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** orientada aos voluntários, onde os mesmos podem interagir com a plataforma. Foi tomada a decisão de desenvolver uma aplicação móvel para permitir que os voluntários possam, de maneira simples e rápida, consultar a plataforma e interagir com a mesma.

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 a arquitetura geral da plataforma e apresentar o *stack* tecnológico selecionado para desenvolver o projeto. O mesmo contém também um secção atribuída a cada módulo do projeto e por fim refere as tecnologias usadas no mesmo.

3.1 Arquitetura Geral

O modelo de arquitetura do presente projeto, apresentado na Figura 3, 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*.

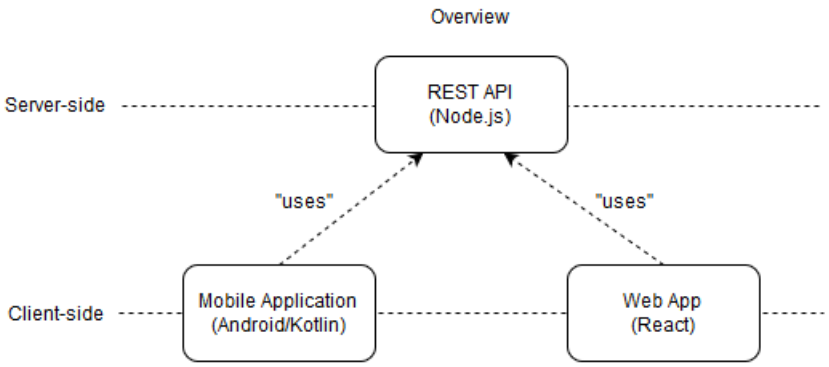

Figura 3: Modelo de arquitetura.

Relativamente à API, esta estabelece *endpoints* onde é 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.

3.2 *Stack* Tecnológico

Tendo em conta os módulos constituintes do projeto, o desenvolvimento do mesmo seguiu o 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 4). 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*.

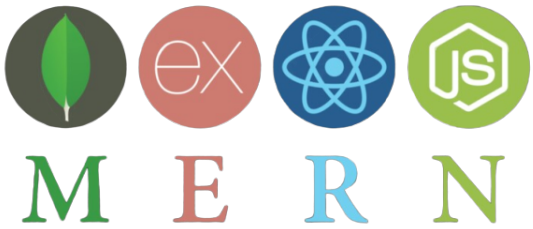

Figura 4: Tecnologias do *stack* MERN (MongoDB, Express.js, React.js, Node.js) [6].

Este projecto é também composto por uma aplicação móvel para Android.

3.3 API

A API constitui o *server-side* do projeto. Este módulo é completamente independente dos outros, tendo como responsabilidade trabalhar como fonte de dados para os restantes módulos (*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 é também responsável por hospedar a base de dados, que usa o 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.

3.4 Aplicação *Mobile*

A aplicação *mobile* foi desenvolvida para a plataforma Android em Kotlin. A mesma segue 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 [7];
- o Kotlin é uma das linguagens oficiais para desenvolvimento de aplicações móveis para Android [8].

3.5 Aplicação *Web*

A aplicação *web* foi desenvolvida usando React. Escolheu-se esta tecnologia pelas seguintes razões:

- 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 [9];
- fácil integração com as outras ferramentas do projeto.

3.6 Tecnologias e ferramentas

As seguintes tecnologias foram 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;
- **Typescript**: linguagem *open source* que extende de Javascript, adicionando um sistema de tipos e outros recursos. Esta linguagem compila para Javascript;
- **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 baseado no Linux. 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 para 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

Neste capítulo, é dado ênfase ao desenvolvimento da API. Como tal, é realizada uma introdução, onde são definidos os requisitos da mesma e são, de maneira reduzida, referidas as tecnologias utilizadas no desenvolvimento desta. De seguida, é apresentada a conceptualização da plataforma e as operações que esta suporta, passando posteriormente para a definição da sua arquitetura e seus sub-módulos. O capítulo é concluído com a realização de uma referência relativamente à base de dados e ao *deployment* da plataforma.

A *web* API é responsável por estabelecer um serviço RESTful [10] com o qual é possível comunicar sobre HTTPS e que implementa as funcionalidades da plataforma. Esta constitui o *back-end* do projeto e como tal, funciona como fonte de dados para as aplicações cliente.

Este módulo necessita de cumprir um conjunto de requisitos:

- implementar as funcionalidades pretendidas da plataforma, como obtenção de listas de voluntários, registo por parte de utilizadores e outras operações;
- interagir com a base de dados;
- comunicar sobre HTTPS e suportar autenticação por parte dos clientes.

Tendo em conta o *stack* tecnológico selecionado para desenvolver este projeto, a API foi desenvolvida em Typescript (sendo que o código é compilado para Javascript) e a mesma é instanciada usando Node.js [11].

4.1 Conceptualização

Tendo em consideração que está a ser desenvolvida uma rede social de voluntariado, foi necessário definir as entidades que estariam disponíveis na plataforma.

Relativamente aos utilizadores da plataforma, foram constituídos dois tipos: **voluntários** e **organizações**. É necessário haver esta distinção dado que na organização de uma ação de voluntariado estes são dois dos sujeitos participantes mais importantes: aqueles que a organizam (organizações) e aqueles que auxiliam a mesma voluntariamente (voluntários).

Sendo o propósito da plataforma divulgar este tipo de ações - é necessário definir mais uma entidade: **eventos**. Estes representam eventos de voluntariado (organizados por organizações registadas na plataforma) que já ocorreram ou que irão ocorrer. É importante também definir que voluntários podem colocar-se como *interessados* num evento - sendo que a partir do momento que estes o façam, é disponibilizado à organização dona do evento o seu contacto.

Dada a natureza da plataforma é necessário permitir que os utilizadores da plataforma (voluntários e organizações) possam ter uma identidade social de

maneira a distinguirem-se dos outros. Sendo assim, foi definido que os utilizadores podem editar o seu perfil (com descrições, contatos e imagens) e realizar ***posts***, sendo possível a outros interagir com estes aspetos (na forma de *gostar de posts* e *seguir outro utilizador*).

4.2 Operações

Levando em consideração o conceito da plataforma definido anteriormente, inicializou-se então a constituição de funcionalidades da API. Definiram-se operações (invocadas através de *endpoints*) que permitem a clientes da plataforma interagir com a mesma, na forma de:

- consultar voluntários, organizações, *posts* e eventos;
- possibilitar que utilizadores possam criar e editar o seu perfil;
- criar *posts* e eventos;
- seguir outros utilizadores, gostar de *posts* e ainda marcar o interesse num evento.

A referência a todas estas operações encontra-se na *wiki* do projeto.

4.3 Arquitetura

A *Web* API, representada na Figura 5, é composta por três módulos: **controladores**, **serviços** e **repositórios** e ainda uma classe principal, responsável por inicializar o servidor e efetuar configurações relacionadas com autenticação e definição de *endpoints* [12, 13].

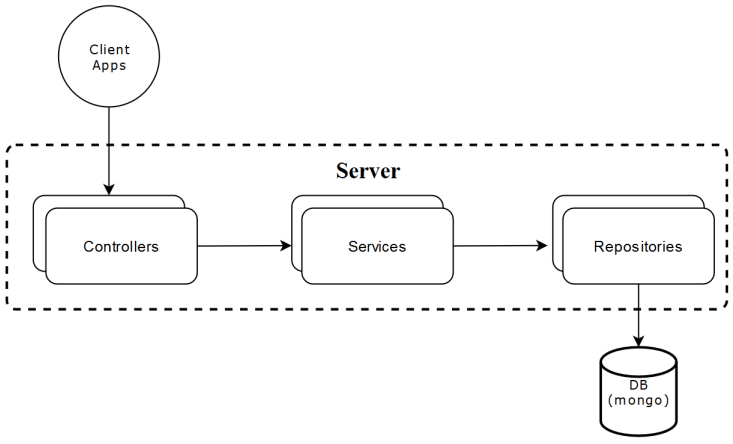

Figura 5: Diagrama de arquitetura da API.

Resumidamente, os **controladores** têm como responsabilidade a definição de *endpoints* e a realização de chamadas aos serviços, sendo que estes últimos (**serviços**) tratam a implementação da lógica da aplicação e realização de chamadas aos repositórios. Estes (**repositórios**) são responsáveis por interagir com a base de dados.

4.3.1 Controladores

Tal como já referido, estes definem os *endpoints* existentes na API e invocam o serviço correspondente. No início da execução da aplicação, é instanciado um controlador para cada entidade existente (voluntários, organizações, *posts* e eventos), sendo que nessa instanciação são definidos *endpoints* utilizando o mecanismo de encaminhamento (*routing*) do Express.

Os controladores tratam também de recolher os parâmetros provenientes nos pedidos HTTPS (contidos no caminho, na *query string*, no corpo do pedido e na sessão do utilizador) e passar os mesmos às funções do serviço, sendo que independentemente do sucesso ou não destas, estes lidam também com a realização das respostas HTTPS.

4.3.2 Serviços

Para cada controlador, existe associado a este um serviço. Neste, são definidas as funções que o controlador invoca de maneira a executar a operação pretendida pelo cliente da API.

Todos os serviços têm uma relação hierárquica de extensão com um serviço base, onde se encontram definidos estaticamente os repositórios de acesso à base de dados.

Os serviços, fazendo uso dos repositórios, interagem com a base de dados para implementar a lógica das operações, sendo que o resultado destas é devolvido assincronamente ao controlador.

4.3.3 Repositórios

Os repositórios têm como função interagir com a base de dados e devolver o resultado das operações efetuadas assincronamente ao serviço, de maneira a este cumprir a operação solicitada.

Cada objeto repositório contém uma referência para uma instância de repositório base. Este repositório base é definido genericamente e contém a implementação de operações CRUD sobre a base de dados.

4.3.4 Autenticação

De maneira a garantir que os clientes da aplicação possam ter uma experiência personalizada, esta API suporta autenticação através do uso de Passport.js, configurado de maneira a fazer uso de *web cookies*. A partir do momento que

um utilizador tem sucesso no processo de autenticação na plataforma, todos os seus pedidos contêm informação relativamente à sessão do mesmo.

Certas operações na API estão protegidas de maneira a que utilizadores não autenticados não as possam efetuar (como por exemplo a realização de um *post* ou a alteração de um perfil). De maneira a que um utilizador possa aceder a estas, é necessário que este tenha realizado autenticação previamente (e por sua vez, tenha realizado registo na plataforma anteriormente) e que o cliente *web* utilizado por este suporte a utilização de *cookies*.

4.4 Base de dados

Tal como introduzido anteriormente, foi tomada a decisão de utilizar MongoDB. MongoDB é um motor de base de dados noSQL [14], isto é, apresenta um modelo não relacional através da utilização de coleções e documentos.

Esta escolha implica que todos os documentos numa coleção não necessitam de ter a mesma estrutura, algo que traz alguma flexibilidade e agilidade no processo de desenvolvimento sobre esta plataforma. Outra característica interessante é que os documentos seguem o formato JSON, simplificando a integração destes dados com a aplicação (devido ao uso de Javascript).

O motor foi selecionado não só para hospedar os dados associados às entidades mas também para albergar o conteúdo das imagens fornecidas pelos utilizadores.

4.5 Implantação da API

Tendo em consideração o meio em que o projeto foi desenvolvido, foi tomada a decisão de hospedar a base de dados na máquina onde está implantada (*deployed*) a API (hospedada no serviço Compute Engine da Google Cloud Platform). Contudo, seria mais interessante do ponto de vista de escalabilidade se a mesma fosse hospedada remotamente (através de uma solução *cloud* como o Amazon Web Services ou a Google Cloud Platform) porque possibilitaria a instanciação de múltiplas aplicações API, e consequentemente, a realização de balanceamento de carga.

A máquina virtual foi configurada com os recursos necessários (como o Node.js) e, numa primeira fase, a aplicação foi instanciada para testes. De maneira a atribuir um nome de domínio ao endereço IP da máquina, foi utilizado o serviço Duck DNS e de seguida, utilizando o CertBot, foi emitido um certificado SSL de maneira a garantir que fosse possível a instanciação da aplicação num modo em que as comunicações fossem realizadas sobre o protocolo HTTPS. A utilização deste protocolo torna a realização de comunicações entre cliente e servidor mais segura devido à encriptação da informação trocada entre ambos. Em plataformas nas quais ocorre a troca de dados sensíveis (como palavras-passe) entre cliente e servidor é essencial a utilização deste protocolo [15].

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 destes são:

- facilitar aos mesmos a consulta de *posts*, eventos, voluntários e organizações da plataforma;
- permitir que estes possam registar-se e autenticar-se na aplicação;
- possibilitar aos utilizadores autenticados a realização de operações como o seguimento de outros utilizadores e gosto de *posts*;

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 [16].

5.1 Navegabilidade e utilização da *App*

Foi desenvolvida uma interface gráfica baseada no grafo de navegação ilustrado na Figura 6.

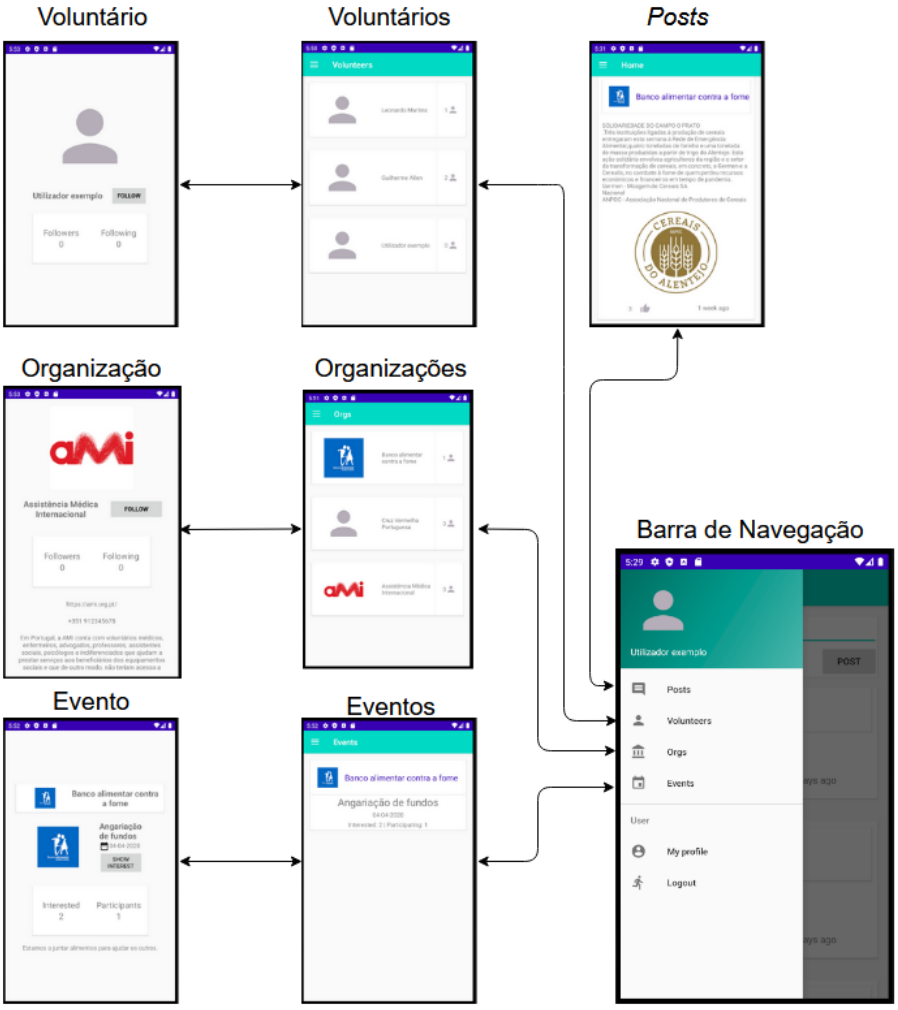

Figura 6: Grafo de navegação.

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 registo e autenticação de clientes.

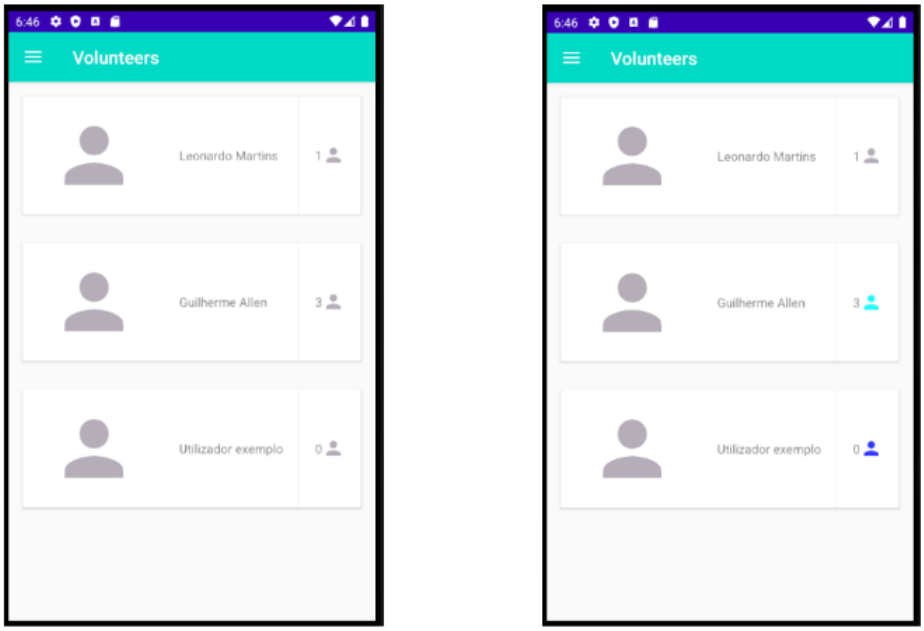

Figura 7: Representação do ecrã "Voluntários" no estado não autenticado (esquerda) e no estado autenticado (direita) para o utilizador "Utilizador exemplo".

A partir do momento em 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 7.

5.3 Arquitetura

Levando em consideração que este projeto utiliza o conjunto de guias dadas pelo Android Jetpack, a arquitetura da aplicação é a recomendada e, como tal, é constituída por três sub-módulos principais: UI (User Interface), API e Modelo [17].

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.

Estes elementos bem como a implementação do conceito de sessão são descritos em maior de detalhes nas subsecções seguintes.

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 [18] 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 [19] - 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 8) 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*.

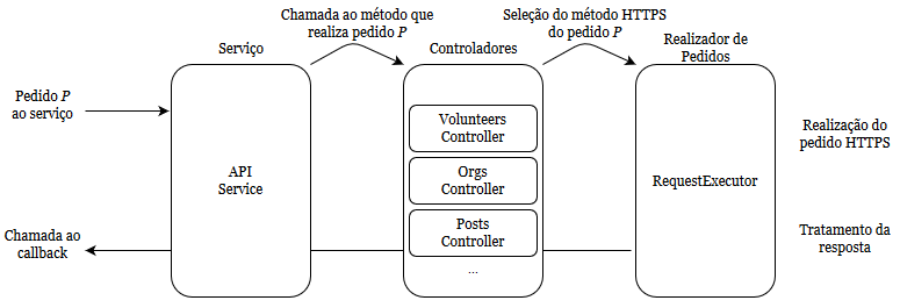

Figura 8: Arquitetura do sub-módulo API.

O realizador de pedidos trata da implementação dos métodos HTTPS suportados pela Web API (nomeadamente GET, POST, PUT e DELETE) de forma

genérica, de maneira a que o mesmo possa ser utilizado pelos controladores. Os pedidos HTTPS 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 a operação.

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 a sessão do cliente. Esta funcionalidade foi implementada através da extensão e redefinição de alguns métodos da classe da biblioteca Volley utilizada para efetuar pedidos HTTPS.

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. Esta hospeda referências para os dados dos voluntários e organizações já recolhidos através de pedidos HTTPS efetuados anteriormente (sendo que esta referência é adicionada à *cache* no momento em que os dados são construídos a partir do seu objeto JSON).

O serviço de *caching* permite ainda, através de uma operação assíncrona, solicitar os dados de um voluntário ou organização em específico, sendo que se a referência para estes dados não estiver já localizada na *cache*, é efetuado um pedido HTTPS para recolher os mesmos.

Se os dados de um voluntário ou organização que já estava referenciada na *cache* são devolvidos como consequência de um pedido HTTPS, os mesmos são atualizados com o estado mais recente destes.

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 biblioteca React.

Esta decisão foi principalmente influenciada devido ao facto de que, tal como já foi referido, React é a tecnologia mais utilizada para desenvolver *front-end* à data da realização do projeto. Outra fator determinante nesta decisão foi a experiência dos autores com esta ferramenta.

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 (através da interface demonstrada na Figura 9).

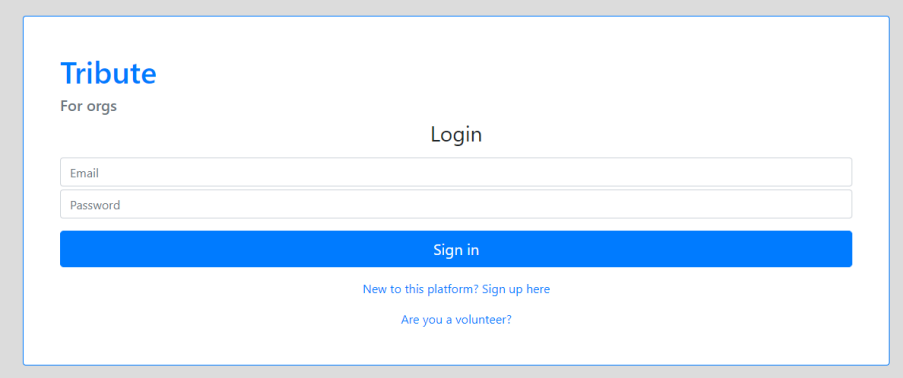

Figura 9: Interface de autenticação.

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.

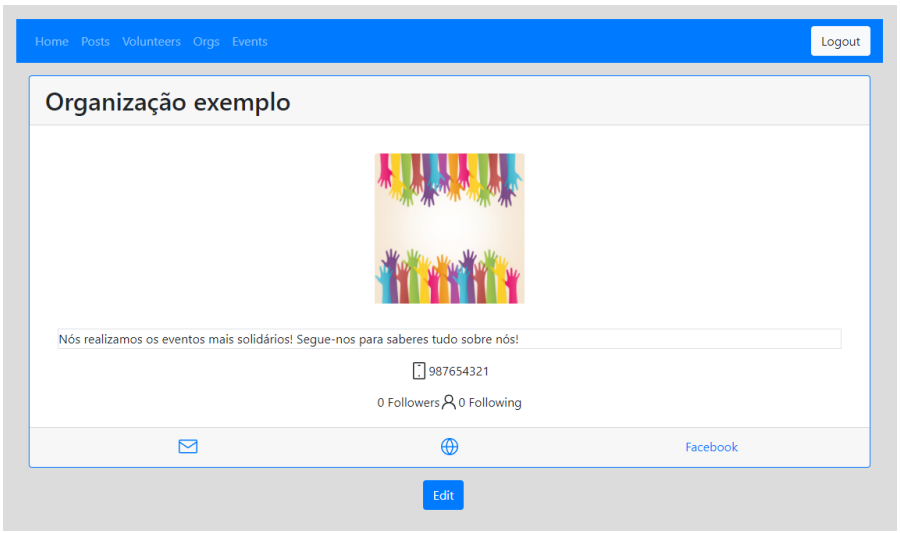

Figura 10: Painel principal da aplicação.

Após autenticação na aplicação é disponibilizado um painel principal (consultar figura 10) 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 dois módulos: API e Componentes e ainda a classe principal da aplicação [20].

Similar ao funcionamento do módulo com o mesmo nome na aplicação *mobile*, o módulo API disponibiliza operações que realizam pedidos HTTPS à *web* API. O módulo 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 mas também por definir o roteamento da aplicação *web*.

6.2.1 API

O módulo API é composto por um conjunto de serviços que disponibilizam operações que necessitam de realizar pedidos HTTPS à API (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).

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 entrada de dados 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.

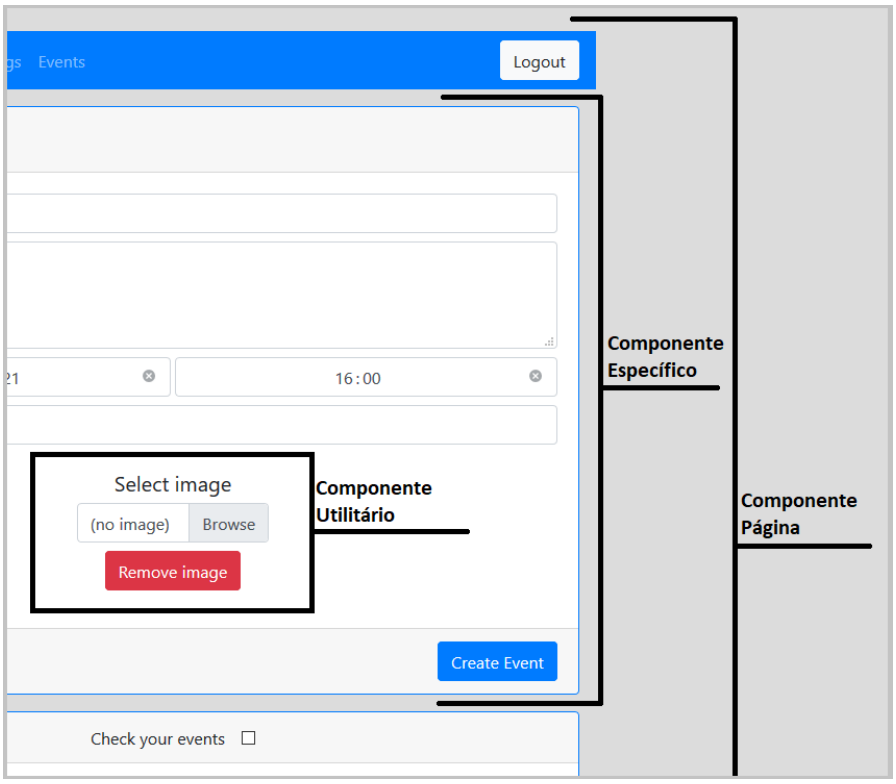

Figura 11: Exemplo de tipos de componentes na página dos eventos.

Os componentes deste projeto que necessitam de atualizar o seu estado após um determinado espaço de tempo (por exemplo, consultar a API de maneira a verificar a existência de novos *posts*) foram desenvolvidos através da definição de classes que extendem de React Component [21] e que re-implementam os métodos necessários para o seu funcionamento (por exemplo, *render* e *componentDidMount*).

Todos os outros componentes foram desenvolvidos através do uso de React Hooks [22] (como o *useState* e o *useEffect*), que simplificam o processo de implementação através da possibilidade do acesso direto a funções que alteram o valor dos estados do componente.

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 Implantação da aplicação

A aplicação foi implantada 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.

Tendo em consideração que a API encontra-se implantada noutra máquina, é necessário lidar com a realização de pedidos CORS (Cross-Origin Resource Sharing), isto é, pedidos efetuados pela aplicação *web* a um domínio externo (neste caso, ao domínio atribuído à API) [23].

De maneira a lidar com estes, 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 implantada a *web* API. Todos os outros pedidos são redirecionados para a porta onde está a ser executada a aplicação *web*.

Como consequência deste redirecionamento, todos os pedidos efetuados pela aplicação são realizados para o seu próprio domínio (lidando com as restrições impostas pelo CORS) e redirecionados pelo servidor *web* para a API.

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/`.

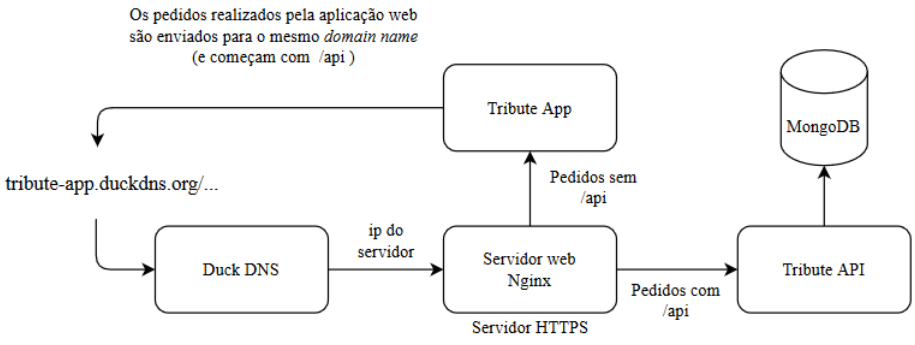

Figura 12: Implantação da aplicação *web*.

7 Conclusão

O presente projeto trata o desenvolvimento de uma rede social de voluntariado, sendo que esta possibilita aos seus utilizadores criarem um perfil e interagirem com outros através de *posts* e seguimentos. Contudo, o ênfase deste projeto é divulgar este tipo de ações, e como tal, permite a entidades organizadoras de eventos marcarem ações e obterem contatos de voluntários interessados em participar nos mesmos.

Tendo isto em consideração, foram desenvolvidos três componentes:

- uma *web* API, que serve como fonte de dados para as aplicações cliente e que trata de definir e implementar as operações da plataforma;
- uma aplicação cliente móvel, que permite a voluntários consultarem a plataforma e registarem-se, criarem um perfil e interagirem com outros utilizadores;
- uma aplicação cliente *web* que permite a organizações criarem o seu perfil e marcarem ações de voluntariado.

Contudo, este projeto teve alguns obstáculos, nomeadamente a seleção e utilização de um *stack* tecnológico e o desenho e implementação da arquitetura da plataforma.

Estes obstáculos apresentam-se porque ao longo da licenciatura é típico a imposição de um *stack* tecnológico e a definição da arquitetura do sistema a desenvolver. Dada esta alteração de contexto, ao longo do projeto houve uma necessidade acrescentada de consultar não só o orientador como também artigos *web*, livros e outros recursos de maneira a utilizar as ferramentas corretamente.

Dado que o objetivo do projeto é desenvolver uma plataforma que poderia ser utilizada num contexto real, durante o desenvolvimento das interfaces de utilizador procurou-se dar a mesma importância ao aspeto funcional e visual das mesmas de maneira a que estas fossem simples, intuitivas e apelativas visualmente, um obstáculo novo para os autores. De maneira a ultrapassar este foram consultados guias de desenvolvimento de interfaces de utilizador.

Apesar dos obstáculos apresentados, o projeto dá-se como concluído e este cumpre todos os requisitos funcionais definidos inicialmente, sendo que esta plataforma, com ligeiras alterações, poderia ser aplicada num contexto real e não apenas no contexto da elaboração de um trabalho final de curso.

7.1 Trabalho futuro

Contudo, e dada a natureza das redes sociais, estas encontram-se em constante desenvolvimento, e vão existir sempre melhorias a realizar e funcionalidades novas a desenvolver.

De seguida, enumeram-se algumas destas:

- aprimoramento do código desenvolvido nas aplicações cliente;
- desenvolvimento de testes unitários;
- adição de novas funcionalidades à plataforma - como comentários em *posts*, noção de voluntário participante em evento e *curriculum vitae* de voluntariado, que recolhe eventos em que voluntário participou;
- integração de registo e autenticação usando contas existentes noutras plataformas (através de OAuth 2.0 [24]), nomeadamente, Facebook e Google;
- melhoramento das soluções de *deployment* aplicadas na plataforma (através do uso de técnicas de balanceamento de carga e de escalonamento automático).

8 Referências

[1] Instituto Nacional de Estatística. Inquérito ao trabalho voluntário, 2019. Acedido a: 09/09/2020. URL: https://www.ine.pt/xportal/xmain?xpid=INE&xpgid=ine_destaques&DESTAQUESdest_boui=379956830&DESTAQUESmodo=2&xlang=pt.

[2] Assembleia da República. Decreto lei nº 71/98 de 3 de novembro. Diário da República n.º 254/1998, Série I-A de 1998-11-03. Acedido a 09/09/2020. URL: https://dre.pt/pesquisa/-/search/223016/details/maximized.

[3] Entrajuda. Bolsa do voluntariado. Acedido a: 09/09/2020. URL: https://bolsadovoluntariado.pt.

[4] United Nations. Online volunteering. Acedido a: 09/09/2020. URL: https://www.onlinevolunteering.org/.

[5] Jeremy Williams. What is the mean stack ? Acedido a: 09/09/2020. URL: https://medium.com/@jeremyvsjeremy/what-is-the-mean-stack-9d11ae2cd384.

[6] Poppy Cooke. MERN Stack image. Acedido a: 09/09/2020. URL: https://morioh.com/p/aa75193629a7.

[7] StatCounter: Global Stats. Mobile operating system market share worldwide. Acedido a: 09/09/2020. URL: https://gs.statcounter.com/os-market-share/mobile/worldwide.

[8] Antonio Leiva. *Kotlin for Android Developers*. Lean Publishing, 2017.

[9] TechMagic. React vs angular vs vue.js — what to choose in 2020? Acedido a: 09/09/2020. URL: https://medium.com/@TechMagic/reactjs-vs-angular5-vs-vue-js-what-to-choose-in-2018-b91e028fa91d.

[10] Roy Thomas Fielding. Architectural styles and the design of network-based software architectures. Acedido a: 09/09/2020. URL: https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm.

[11] Jim R. Wilson. *Node.js 8 the Right Way*. Pragmatic Bookshelf, 2018.

[12] Arnaud Lauret. *The Design of Web APIs*. Manning Publications, 2019.

[13] Glenn Block, Pablo Cibraro, Pedro Felix, Howard Dierking, and Darrel Miller. *Designing Evolvable Web APIs with ASP.NET*. O'Reilly Media, Inc., 2014.

[14] MongoDB. What is nosql? Acedido a: 09/09/2020. URL: https://www.mongodb.com/nosql-explained.

[15] Cloudflare. Why use https? Acedido a: 09/09/2020. URL: https://www.cloudflare.com/learning/ssl/why-use-https/.

[16] Marko Gargenta and Masumi Nakamura. *Learning Android: Develop Mobile Apps Using Java and Eclipse*. O'Reilly Media, Inc, 2014.

[17] Android Jetpack. Common architectural principles. Acedido a: 09/09/2020. URL: https://developer.android.com/jetpack/guide#common-principles.

[18] Android Developers. Viewmodel overview. Acedido a: 09/09/2020. URL: https://developer.android.com/topic/libraries/architecture/viewmodel.

[19] Android Developers. Create a list with recyclerview. Acedido a: 09/09/2020. URL: https://developer.android.com/guide/topics/ui/layout/recyclerview.

[20] Stoyan Stefanov. *React: Up and Running*. O'Reilly Media, Inc., 2016.

[21] ReactJS. React.component. Acedido a: 09/09/2020. URL: https://reactjs.org/docs/react-component.html.

[22] ReactJS. Introducing hooks. Acedido a: 09/09/2020. URL: https://reactjs.org/docs/hooks-intro.html.

[23] MDN contributors. Cross-origin resource sharing (cors). Acedido a: 09/09/2020. URL: https://developer.mozilla.org/pt-PT/docs/Web/HTTP/CORS.

[24] IETF. Oauth 2.0. Acedido a: 09/09/2020. URL: https://oauth.net/2/.