INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Rede Social de Voluntariado

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

Autores:

Guilherme Allen
Nº 43571

Leonardo Martins
Nº 43591

Orientador: Nuno Leite

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 efetuada, por norma, através de redes sociais, que não são vocacionadas para este tipo de ações, e em *websites*, o que leva a uma descentralização da informação.

O presente projeto visa resolver esta problemática 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 mesmo será 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 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 will be 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

Lista de Figuras

Lista de acrónimos

REST Representational State Transfer

API Application Program Interface

HTTP HyperText Transfer Protocol

HTTPS HyperText Transfer Protocol Secure

NPM Node Package Manager

JSON JavaScript Object Notation

MEAN MongoDB Express.js Angular.js Node.js

MERN MongoDB Express.js React.js Node.js

HTML HyperText Markup Language

CSS Cascading Style Sheets

SO Sistema Operativo

JVM Java Virtual Machine

CRUD Create, Read, Update and Delete

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]

O trabalho voluntário, ou voluntariado, segundo o diário da república, tem como definição:

"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 nosso 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).

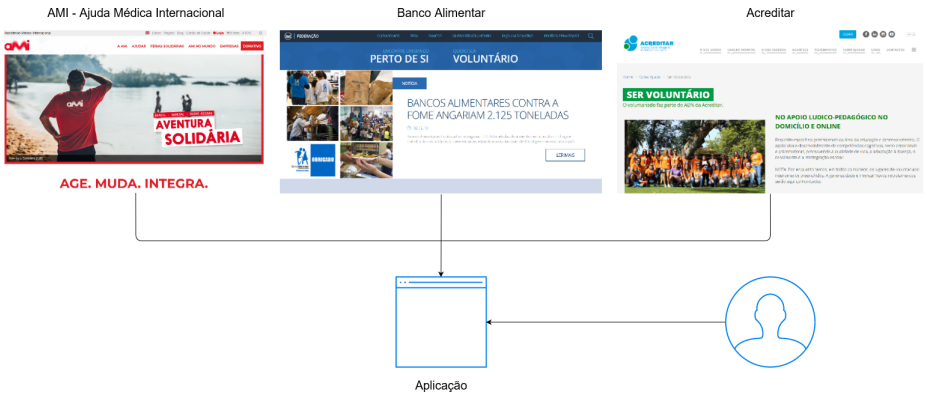

Figura 2: Conceito do projeto

1.1 Organização do relatório

O presente relatório começará por analisar a problemática atual da divulgação de ações de voluntariado, apresentando de seguida o modelo de arquitetura proposto, mencionando também as soluções e as tecnologias a usar.

Foi desenvolvido um capítulo associado a cada componente do projeto (*web* API, aplicação *mobile* e cliente *browser*), discutindo mais em detalhe as opções tomadas durante o desenvolvimento dos mesmos.

Na fase final do relatório é também discutido o planeamento e desenvolvimento do projeto e as conclusões tiradas durante a evolução do mesmo.

2 Formulação do problema

Neste capítulo, vamos começar por referir os problemas típicos do uso de redes sociais e *websites* para divulgação de ações de voluntariado, apresentando também dois casos de plataformas já existentes e levantando a problemática principal das mesmas. Concluiremos este capítulo definindo os requisitos funcionais do projeto.

Tal como já foi referido, a interação voluntário-organização é tipicamente feita através de dois tipos de plataformas: redes sociais e *websites* das organizações.

As redes sociais, por não serem, por desenho, vocacionadas para este tipo de ações, apresentam alguns problemas de utilização, como filtragem de informação e 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.

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

2.1 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.

2.2 Análise

A principal problemática presente nestas plataformas é o facto de as mesmas realizarem uma divulgação passiva (apresentar ações solicitadas pelo utilizador) em vez de uma divulgação ativa (sugerir aos utilizadores ações de possível interesse).

Esta limitação pode ser combatida através do uso de mecanismos de interação similares aos das redes sociais, como ferramentas de "seguimento" de organizações ou tipos de ações. Essas ferramentas irão simplificar o processo de executar a divulgação ativa, e como tal, a personalização da experiência do uso da aplicação de utilizador para utilizador.

2.3 Requisitos funcionais

O sistema a desenvolver é composto por uma *web* API, uma aplicação *mobile* e uma aplicação *web*. A seguir elencam-se os requisitos funcionais destes componentes.

Web API

1. Possibilidade de efetuar pesquisas relativamente ao dados existentes na plataforma (voluntários, organizações, eventos, *posts*);
2. possibilidade de criar/alterar/apagar entradas já existentes na plataforma;
3. autenticação de maneira a disponibilizar uma experiência personalizada para cada utilizador;
4. disponibilizar operações típicas de redes sociais (como seguimento de perfis);
5. auxiliar no processo de inscrição de voluntários em eventos.

Aplicação *mobile*

1. Possibilidade de efetuar registo na plataforma;
2. existência de um modo para utilizadores não autenticados e autenticados, sendo que certas operações apenas poderão ser efetuados por estes últimos;
3. apresentação de resultados das pesquisas possíveis de efetuar na API e possibilidade de efetuar filtros nas mesmas;
4. funcionalidade de poder modificar o seu perfil, criar/alterar *posts* e seguir outros utilizadores/organizações, entre outros.

Aplicação *Web*

1. Possibilidade de solicitar registo na plataforma (esta plataforma será apenas destinada a organizações autenticadas);
2. apresentação de resultados das pesquisas da API;
3. funcionalidade de edição de perfil;
4. criação e gestão de *posts* e eventos.

3 Modelo de Arquitetura

O modelo 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 standard 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);
- *standard* 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 este modelo de desenvolvimento são *open-source*.

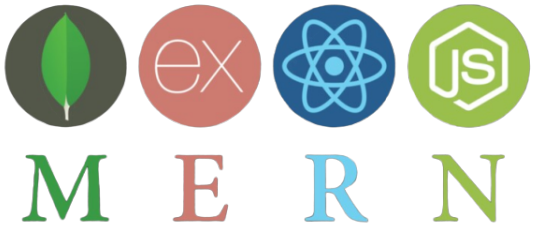

Figura 3: Tecnologias do *standard* MERN (MongoDB, Express.js, React.js, Node.js)

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*, serão 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.;

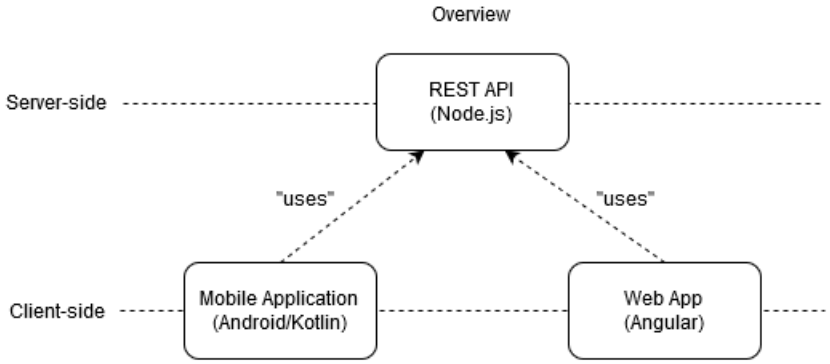

Figura 4: Modelo de arquitetura

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 será Node.js em conjunto com vários *packages* do NPM (*node package manager*), como o Express e o Passport. Esta escolha foi 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 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;
- atualmente, o Kotlin é a linguagem oficial 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;
- 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**: Sistema operativo *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 a linguagem oficial do Android;
- **Android Jetpack**: Conjunto de ferramentas e bibliotecas as quais 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**: *Framework* utilizada para efetuar o carregamento de imagens em aplicações Android;
- **React**: Biblioteca *open-source* de Javascript usada para desenvolver a interface de utilizador de aplicações web.

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].

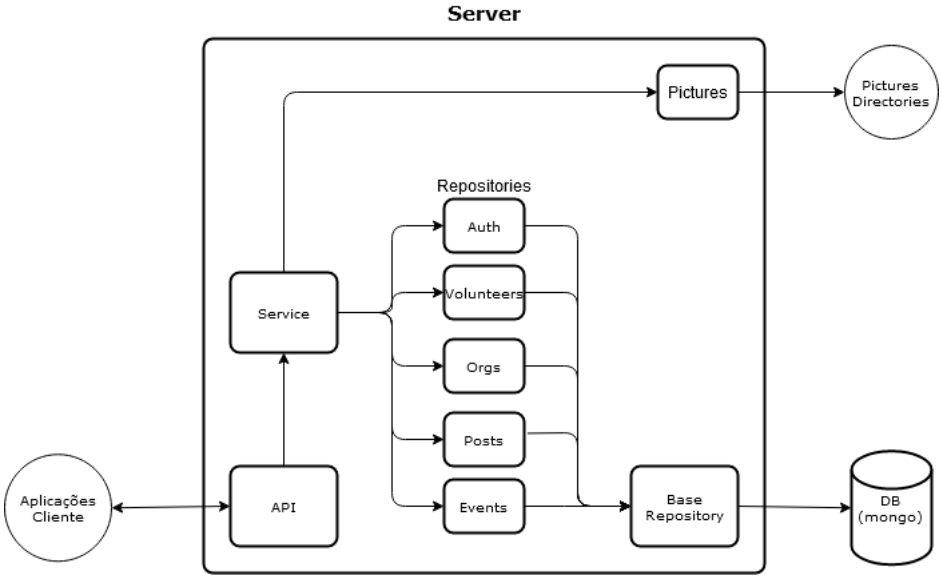

Figura 5: Diagrama de arquitetura da API

4.1 API

Este módulo é responsável por definir os *endpoints* e lidar com a recepção e envio dos pedidos/respostas HTTP. Cada *endpoint* tem associada uma função que é executada e que efetua a chamada adequada ao serviço para realizar a operação solicitada pelo utilizador.

A figura 6 ilustra a tabela de operações disponibilizadas pela API. Refere-se também o método e o URL do pedido.

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.

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 6: Lista de *endpoints*

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 *body* do *request* (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 materializem 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 alterados para inserir/modificar/apagar entradas dos índices aos quais se referem. Estes repositórios, por sua vez, têm acesso a uma instância de *BaseRepository* gerada especificamente para si (isto é, inicializada dando parâmetros 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. Este segue o padrão *noSQL*,apresentando um modelo não relacional, tipicamente utilizado no desenvolvimento de redes sociais. Esta escolha é justificada pelas seguintes razões:

- Flexibilidade. Tendo em conta que este tipo de plataformas costuma ter um crescimento exponencial, a adição de novas *features* é constante. Num modelo SQL, por vezes é dispendido 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 do conceito 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 *upload* 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 *upload* de imagens para esta plataforma é apenas possível para utilizadores autenticados, no entanto, o *download* 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.

Este capítulo incidirá sobre o desenvolvimento e implementação da aplicação móvel do nosso projeto. Começaremos por realizar uma pequena introdução, de seguida apresentando a navegação na aplicação e a sua arquitetura, concluindo este capítulo discutindo cada componente da mesma.

5 *Mobile App*

O segundo componente deste projeto é a aplicação móvel. Esta funciona como interface para o tipo de utilizador mais comum (Voluntário) e foi desenvolvida para a plataforma Android.

Este módulo contém alguns requisitos chave que são necessários para garantir a usabilidade da mesma por parte dos seus utilizadores. Seguem-se alguns dos requisitos a cumprir diante dos voluntários:

- 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 ou 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 (como o Volley e o Glide).

5.1 Navegabilidade e utilização da *App*

Tendo em conta que este componente constitui uma interface gráfica (ou seja, irá ser utilizada em primeira instância por utilizadores humanos), existiu durante o desenvolvimento do mesmo um cuidado em termos de navegação e desenho de maneira a tornar a mesma o mais intuitiva possível.

De seguida, encontra-se o padrão de navegação definido na figura 7.

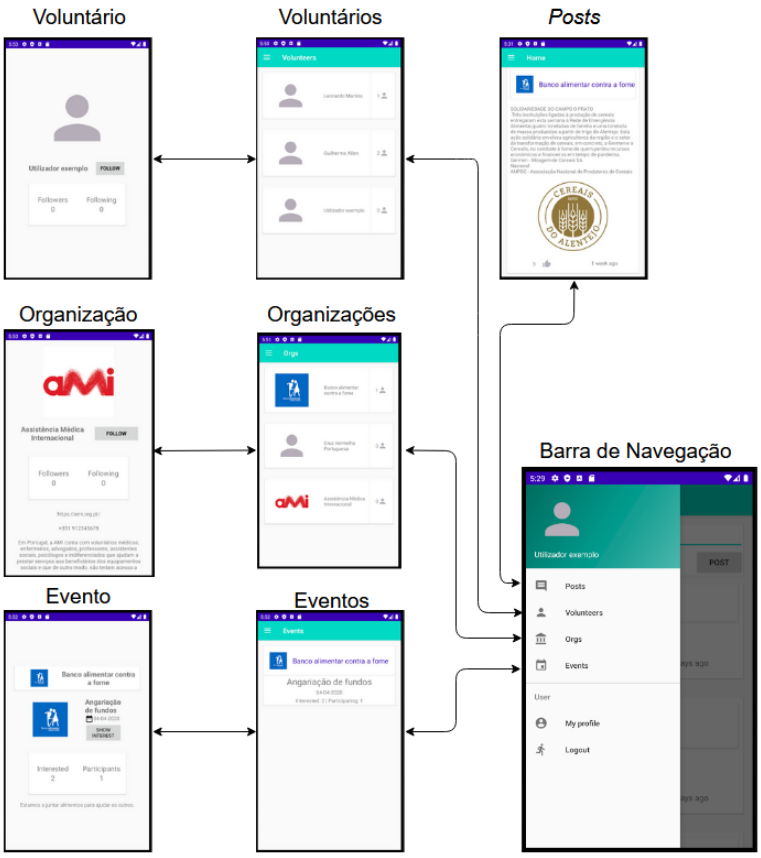

Figura 7: Grafo de navegação

Foram desenvolvidas 4 vistas principais de pesquisa (*posts*, voluntários, organizações e eventos) de maneira a apresentar os dados na plataforma já existentes e também vistas detalhadas para estes mesmos índices (excepto *posts*) para que o utilizador pudesse 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 maneira a que clientes do mesmo se possam registar/autenticar.

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 iram ter uma vista personalizada, tal como exemplificado na figura 8.

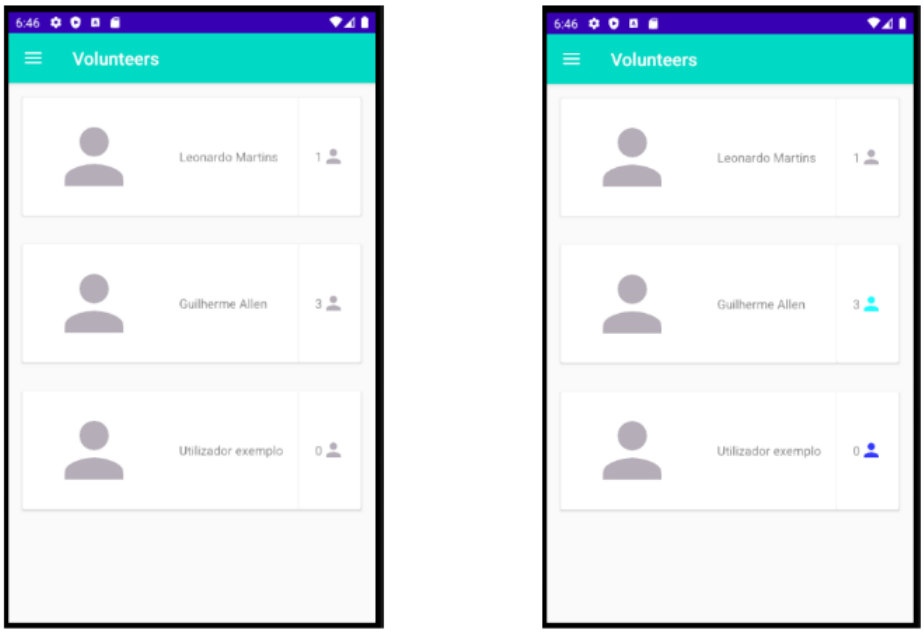

Figura 8: Representação do ecrã ”Voluntários” no estado não autenticado (figura da esquerda) *versus* no estado autenticado como o utilizador "Utilizador exemplo"(figura da direita)

5.3 Arquitetura

A arquitetura da aplicação é constituída por 3 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 do projeto. 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 mesmos dados e reduzir o número de pedidos efetuados à Web API.

5.4 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 (utilizada para efetuar pedidos HTTP) de maneira a que, quando o utilizador estivesses autenticado na aplicação, os pedidos efetuados pela mesma tivessem 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, cujo contém informações relativamente ao voluntário autenticado.

5.5 *User Interface*

Este sub-módulo engloba todos os componentes do projeto que são responsáveis por representar os dados da aplicação, sendo que não são efetuadas alterações na interface de utilizador noutros módulos. Como tal, este é constituído por:

- implementações de ecrãs, quer sejam atividades ou fragmentos. É de notar que para cada um destes existe um objeto ViewModel associado, responsável por manter o estado dos elementos do ecrã e também por interagir com a API quando necessário;
- adaptadores. Estes são utilizados pelos ecrãs quando existe a necessidade de apresentar uma lista de elementos (como voluntários ou *posts*) tomando proveito do componente RecyclerView;
- Outros componentes utilitários, como por exemplo uma estrutura que auxilia o carregamento de imagens para a representação das mesmas.

5.6 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*.

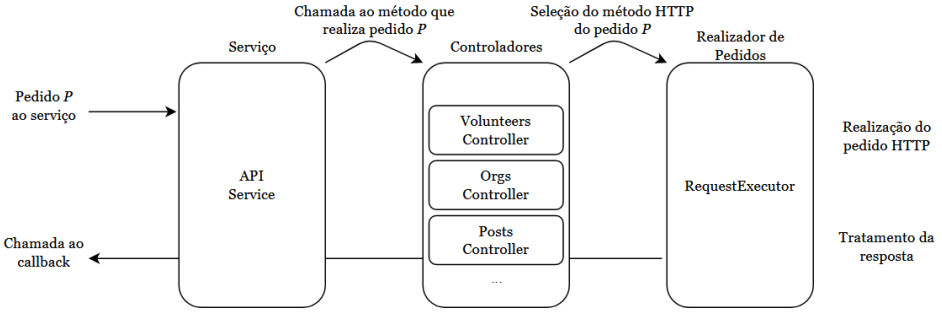

Figura 9: Arquitetura do sub-módulo API

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.

Quando o pedido é concluído, é chamado o *callback* definido pela estrutura que chamou o serviço, que consume 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.

5.7 Modelo

O Modelo é responsável por transformar os dados provenientes do sub-módulo API para DTO's (*Data Transfer Objects*) que iram ser usados pelo sub-módulo UI para efetuar a representação destes mesmos.

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

6 *Web App*

Face à proposta de projeto, ouve uma alteração relativamente à tecnologia usada para desenvolver este componente. Numa abordagem inicial, foi tomada como decisão o desenvolvimento deste componente usando a ferramente Angular.js.

No entanto, após nova avaliação, foi tomada a decisão de se usar a tecnologia React. Esta alteração foi feita devido a algumas pesquisas efetuadas durante o desenvolvimento do relatório.

O desenvolvimento deste componente ainda não foi inicializado. No entanto, está previsto que à data da entrega da versão beta, os requisitos funcionais do mesmo estejam parcialmente desenvolvidos.

7 Planeamento e desenvolvimento do projeto

Foi elaborada a seguinte calendarização no inicio do projeto:

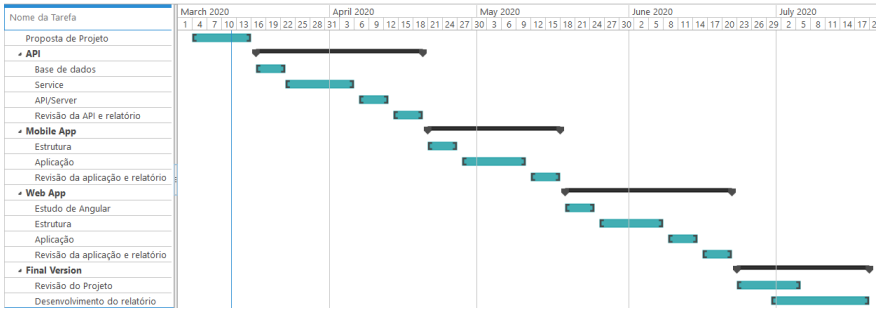

Figura 10: Diagrama de planeamento

À data da entrega do relatório de progresso, ocorreu um ligeiro deslize de menos de uma semana, no entanto, dado ao período atribuído no final do projeto para revisão e desenvolvimento do relatório, e tendo em conta o desenvolvimento do mesmo, não irá causar distúrbios na entrega do projeto.

8 Conclusão

À data da entrega do relatório de progresso, podemos apontar algumas conclusões a partir daquilo que foi o desenvolvimento do componente API:

- apesar dos conhecimentos relativamente à implementação de uma API já obtidos ao longo da licenciatura, tivemos alguma dificuldade naquilo que foi o desenho e definição de requisitos funcionais da API. Após uma primeira versão da mesma, voltámos< a implementá-la uma segunda vez, esta com uma estrutura melhor definida e que cumpre todas as funcionalidades que um cliente da API irá necessitar;
- como já foi referido, apesar da mesma já ter sido consolidada uma segunda vez, podemos necessitar de adicionar alguns componentes extra aquando do desenvolvimento das aplicações cliente, sendo que estes componentes extra irão surgir quando forem necessários.

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.*