INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Área Departamental de Engenharia Eletrónica e Telecomunicações e de Computadores

CROSSWORKING

DAVID PEREIRA LOURENÇO

LOURENÇO GONÇALVES VALA

Relatório realizado no contexto de Projeto e Seminário,

Licenciatura em Engenharia Informática e de Computadores

Semestre de Verão 2021-2022

Orientador: João Humberto Holbeche Trindade

INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Área Departamental de Engenharia Eletrónica e Telecomunicações e de Computadores

CROSSWORKING

DAVID PEREIRA LOURENÇO

_____

LOURENÇO GONÇALVES VALA

_____

Relatório realizado no contexto de Projeto e Seminário,

Licenciatura em Engenharia Informática e de Computadores

Semestre de Verão 2021-2022

Orientador: João Humberto Holbeche Trindade

_____

Resumo

O desenvolvimento da espécie humana até à atualidade deve-se em grande parte, a ideias que algumas das pessoas mais importantes da nossa história, conseguiram colocar em prática devido aos conhecimentos que possuíam.

No contexto atual, em que cada individuo se especializa numa área de estudos em detrimento de tantas outras, a prospeção para que uma pessoa com ideias concretizáveis, rentáveis, e que muitas vezes poderiam mudar o rumo da evolução da sociedade, não consiga colocar a sua ideia em prática devido a falta de conhecimentos nessa área em específico é elevada.

Como forma de mitigar o problema descrito, foi desenvolvida uma plataforma, que tem como objetivo proporcionar a troca de ideias entre utilizadores. Os utilizadores têm oportunidade de registar as suas ideias bem como os seus conhecimentos, conhecimentos estes que ficaram associados ao seu perfil, por forma a proporcionar ao utilizador uma lista de ideias que se enquadrem com as suas aptidões. Posteriormente é oferecida a possibilidade de um utilizador se candidatar à realização de uma ideia, candidatura esta que deverá ser aceite pelo proponente, para que a ideia fique atribuída e os seus contactos sejam trocados.

A concretização desta plataforma foi desenvolvida uma *Web API* (*Application Programming Interface)*, alojada num servidor com interface *HTTP*, a qual realiza a manipulação dos dados necessários ao correto funcionamento da plataforma. Para disponibilização de interface gráfica ao utilizador foi decidida a utilização da Framework Android. A persistência dos dados foi garantida com recurso a uma base de dados relacional, e a um serviço de armazenamento de objetos.

Apesar da escolha da *framework Android*, a construção da *Web API* teve em consideração a possível expansão da plataforma para outras *frameworks front-end*.

Abstract

Up until the development of the human race was largely due to the ideas of some of the most important people in our history and their ability to put these ideas to practice with their knowledge.

In today's age, most people choose to specialize in a particular field whilst neglecting the rest. This makes it particularly difficult for them to put into practice easy and rentable ideas, which could change society.

To fight this issue a platform was designed, for people to be able to share their ideas, their knowledge, and their skills. The users of this platform can save their ideas and skills in their profile, so the ideas shared to them are related to their expertise. Afterwards they can apply to one of these ideas, and should they be accepted by the creator, they will enter in contact to commence development.

For this platform a *Web API* (*Application Programming Interface*) was developed and deployed in a *HTTP* server, to manipulate the data required for this service. As for the user interface (UI), the *Android Framework* was chosen. For the data to be persistent, relational database was created, as well as an object storage service.

Although the UI chosen for this platform was an *Android* application, the *Web API* was developed with consideration for future expansion into other types of *front-end*.

Índice

Listagem de Figuras

Lista de Listagens

1 Introdução

No início da espécie humana, cada individuo existia de forma singular, sem existência de uma comunidade, neste paradigma, era essencial possuir o maior número de conhecimentos para garantir a sua sobrevivência. Com o nascimento das primeiras comunidades, começou a constatar-se uma divisão de conhecimentos, onde cada individuo continuava a apresentar capacidade de viver isolado, mas, conseguia, em comunidade adquirir conhecimentos mais específicos sobre a sua tarefa, fosse esta a coleta, a caça ou a domesticação entre outras atividades.

Com a evolução da espécie, as comunidades tornar-se-iam sociedades, e cada individuo passaria a assumir um papel extremamente importante na sua área. Atingido este ponto a capacidade de sobrevivência em isolamento atingiu um patamar onde se tornaria difícil regredir, e onde cada individuo apenas conseguiria viver em comunidade.

Com o avançar dos séculos e da evolução, a necessidade de viver em comunidade, ser parte integrante desta, e garantir o seu contributo para a mesma aumentou de forma exponencial. A especialização de cada individuo em determinada área foi elevada ao seu máximo expoente, através das áreas do ensino secundário e as licenciaturas, mestrados e doutoramentos nos Politécnicos e Universidade.

1.1 Problema

A elevada especialização de toda a sociedade numa área especifica levou a elevadas discrepâncias de conhecimentos entre a área de estudos e todas as restantes. Existem hoje indivíduos extremamente cultos em determinadas áreas, mas que apresentam deficiência no conhecimento em tantas outras.

No entanto, a falta de conhecimento sobre múltiplas áreas, não se traduz num desinteresse geral sobre as mesmas, o que leva a que muitas vezes seja possível teorizar uma ideia que poderia levar a uma melhoria efetiva nessa mesma área, mas devido a falta de conhecimento específico que essa ideia viria a necessitar, a mesma é descartada.

Constata-se de igual forma que, existem indivíduos extremamente especializados que não conseguem ter ideias relativas á sua área de estudos suficientemente sustentadas, para que possam empregar o seu tempo e conhecimento.

Por forma a colmatar este cenário, foi idealizada uma plataforma onde é possível trocar ideias e conhecimentos, permitindo juntar no mesmo ambiente pessoas com ideias inovadoras e pessoas com conhecimentos técnicos para as desenvolver. Surge assim a plataforma *CrossWorking*.

1.2 Objetivos

Precedentemente ao desenvolvimento da plataforma, e uma vez detetado e avaliado o problema, foram traçados os objetivos essenciais ao funcionamento da plataforma, e que se esperava cumprir para que o problema identificado pudesse ser mitigado. Pretendia-se assim cumprir os seguintes objetivos com o desenvolvimento da plataforma:

- Compartilha de ideias.
- Partilha de aptidões e conhecimentos.
- Incentivo à inovação e a cooperação.
- Instigação à criação de equipas multidisciplinares
- Melhoria nas capacidades descritivas dos utilizadores, bem como na capacidade de comunicação com outros utilizadores de áreas de estudos dispares.

1.3 Enquadramento

Uma vez identificado o problema, foi realizada uma pesquisa por soluções existentes e alternativas à que se pretendia implementar. Desta pesquisa destacam-se duas soluções onde poderíamos encontrar alguns pontos em comum com a plataforma *CrossWorking*, sendo estas o *LinkedIn*, e a *Toptal*. No entanto, ambas as plataformas têm o seu foco na contratação de pessoas para determinada tarefa, o que não se coaduna com o objetivo da plataforma *CrossWorking*, uma vez que o seu objetivo não reside na contratação, mas sim na partilha e na entreajuda dos utilizadores.

É também necessário referir que já durante a execução do projeto tivemos os autores tomaram conhecimento de uma terceira plataforma denominada de *Fixando* que, mais uma vez apresenta alguns pontos em comum com a solução *CrossWorking*, embora esteja presente o mesmo ponto de repulsão face às anteriormente mencionadas, isto é, a existência de um contrato ou prestação de serviço. Um utilizador na plataforma fixando procura profissionais para resolução de problemas, algo que a distancia da plataforma *CrossWorking* que proporciona um espaço para ideias e para pessoas que desejem realizar ideias que poderão ser revolucionárias.

Analisadas as soluções já existentes, que os autores tenham tido conhecimento, e uma vez comparadas com a implementação que viria a ser levado a cabo, foram reconhecidas as diferenças e considerou-se que as mesmas seriam motivação suficiente para avançar com o desenvolvimento da solução *CrossWorking*, uma vez que existe necessidade e espaço no mercado para a existência desta mesma plataforma.

1.4 Descrição da Solução

Com o realizar da pesquisa por soluções semelhantes, foi possível complementar a solução de forma que apresentasse todas as condições necessárias ao seu sucesso. Deriva desta análise uma descrição de todos os conceitos chave necessários para a correta compreensão da plataforma desenvolvida, e consequentemente a sua correta utilização por parte dos utilizadores.

Para que a interpretação da solução posse ser corretamente compreendida é necessário caracterizar alguns conceitos. Um utilizador é uma qualquer pessoa que se encontre registada na plataforma e utiliza a mesma. Uma ideia é uma imagem ou representação mental de um objeto, seja este físico, digital ou de outro âmbito não palpável; o que para a plataforma *CrossWorking* representa o elemento central do seu funcionamento, os utilizadores registados têm a possibilidade de propor ideias. Por sua vez, uma competência (por vezes denominada pelos autores de skill, importado do idioma inglês, que se traduz em habilidade, capacidade ou competência), é um conhecimento que determinada pessoa possui e que lhe permite realizar uma atividade, na implementação *CrossWorking* é um conceito também fulcral uma vez que representa as capacidades que os utilizadores disponibilizam, e também as competências necessárias ao desenvolvimento de uma determinada ideia. Por fim um candidato é um utilizador que se propôs a implementar ou ser parte da solução de uma ideias e aguarda que o proponente da mesma o selecione de entre uma lista de outros utilizadores que também se propuseram a realizar essa mesma ideia.

O desenvolvimento concreto da plataforma, envolveu a criação de um serviço centralizado que disponibiliza os dados e funcionalidades a vários tipos de dispositivos (este serviço é denominado no restante documento pelo seu nome técnico, *back-end*, designação importada do idioma inglês e que se traduz em infraestrutura na qual não se prossupõe existência de interação direta com o utilizador), foi também decidida a utilização como interface para o utilizador ,uma aplicação *Android* a ser desenvolvida e que permitirá tirar partido da totalidade das funcionalidades do *back-end* (esta aplicação é muitas vezes apelidada no presente documento pelo seu nome técnico, *front-end*, substantivo importado do idioma inglês e que se poderá traduzir em interface disponibilizada aos utilizadores para interação com a plataforma). A implementação do *front-end* e do *back-end* representam a concretização da solução. Esta solução será detalhada na sua componente mais técnica nos capítulos que se seguem.

2 Infraestrutura

Neste capítulo será descrita o funcionamento e organização da plataforma *CrossWorking*. Também será abordada a sua utilização desde a criação dos utilizadores, preenchimento das suas informações e registo das ideias, a candidatura a uma destas e a aceitação ou rejeição do candidato.

Abordar-se-á ainda, a escolha das tecnologias e dos serviços presentes da plataforma CrossWorking, e o seu contexto de utilização, será também objeto deste capítulo a escolha do serviço de hospedagem e as soluções que os mesmos propõem.

2.1 Utilização da plataforma

Para a utilização da plataforma é necessária a autenticação do utilizador. Feito isto existem várias opções, desde a personalização do seu perfil, adicionar as suas skills e ideias, ou então explorar os outros utilizadores e as suas ideias.

Esta exploração é permitida através de um ecrã de atividade (também denominado de feed, importado do idioma inglês), no qual é possível consultar ideias de todos utilizadores inscritos na plataforma. Se o utilizador tiver inserido as suas competências, este feed será personalizado de forma a demonstrar as ideias que correspondam às suas habilidades, caso contrário, o feed contem a lista de ideias existentes, ordenadas pela sua data de inserção.

Se o utilizador tiver skills correspondentes a uma ideia, poderá candidatar-se a ela e caso o utilizador que a criou o aceite, serão partilhados os contactos do candidato ao criador.

Para além da criação existe a opção de editar ou ainda apagar os dados do seu perfil, bom como inserção de skills.

A criação de ideias envolve a introdução de um título, uma breve descrição, uma descrição detalhada da ideia que propõe, neste campo o utilizador deverá colocar dados como a descrição geral da ideia, a sua utilização prática, o contexto em que se aplica, entre outros dados que considere relevante. Será também necessário ao utilizador introduzir as skill que pondera necessárias à concretização da sua ideia.

2.2 Organização geral

O *back-end* da plataforma é composto por uma *Web API*, composta por um serviço de autenticação, uma base de dados e um serviço de armazenamento de objetos (para as imagens submetidas pelos utilizadores); o conjunto destes módulos representa o servidor da plataforma. Existindo a possibilidade de crescimento desta plataforma, partiu-se do princípio de que o servidor poderia vir a servir múltiplas *Frameworks* de *front-end*, como tal foi adotada uma arquitetura de desenho que viabiliza a existência de múltiplos front-end. Como seria inviável realizar a implementação de aplicações para as múltiplas *Frameworks* no contexto sobre o qual foi realizada a plataforma (projeto final de licenciatura), foi escolhida pelos autores a *Framework Android* como opção a desenvolver, tendo sido tomado como critério a familiarização dos respetivos criadores com a referida *Framework*. Os autores consideraram ainda que esta Framework seria a indicada para mostrarem os seus conhecimentos. Inerente a todos estes fatores é também necessário ter em consideração o universo *Android* que segundo *Counterpoint Research, Gartner, IDC* [1] atingiu em 2021 o marco de 2,5 mil milhões de utilizadores, representando 71.3% da quota de mercado dos sistemas operativos para dispositivos móveis, segundo dados da *statcounter GlobalStats* [2]. Foi ainda ponderada a possibilidade da criação de uma aplicação web com interface compatível com dispositivos móveis, no entanto, concluiu-se que apesar das tecnologias web apresentarem um elevado nível de sofisticação que permitiriam tal solução, continuariam a não apresentar o mesmo tipo de interatividade e experiência de utilização que uma aplicação nativa consegue oferecer.

2.3 Hospedagem de serviços

Com o surgimento dos serviços de *cloud*, ocorreu uma alteração de paradigma na área das tecnologias de informação. Empresas um pouco por todo o mundo começaram a utilizar estes serviços a seu favor, diminuindo drasticamente o investimento inicial necessário para levar a que uma empresa tecnológica inicie a sua atividade.

Existem hoje disponíveis uma grande oferta de serviços de *cloud*, pelos mais variados valores, sendo disso exemplo a *Microsoft Azure*, a *Google Cloud Plataform* e a *Amazon Web Services* (abreviada para a sigla *AWS*).

Como desenvolvedores de uma plataforma inovadora e sem qualquer tipo de infraestrutura, também os autores, se depararam com a necessidade de hospedar um servidor com a *Web API*. Para que um serviço *cloud* pudesse ser considerado como viável, necessitava de oferecer um serviço para hospedagem do servidor com a *Web API*, compatível com as tecnologias

da mesma; uma base de dados relacional e um serviço de armazenamento de ficheiros de grandes dimensões como fotos.

Para a seleção foi também preciso ter em consideração o fator monetário, uma vez que, os autores não possuíam qualquer tipo de fundo monetário para o desenvolvimento desta plataforma, nem decorria nenhum investimento monetário na mesma.

Tendo todos estes fatores em consideração, optou-se pelos serviços oferecidos pela *AWS*, uma vez que permitia acesso gratuito a um serviço de hospedagem (denominado pela *AWS* de *Beanstalk*), uma base de dados relacional na qual existia um máximo no armazenamento de 20GB (serviço ao qual a *AWS* denomina de *RDS*) e também um serviço de *cloud storage*, para armazenamento de fotos, com um máximo de 5GB (apelidado pela *AWS* de *S3*).

3 Implementação do back-end

No capítulo que se segue é descrita a implementação do servidor back-end da plataforma CrossWorking. Será apresentado a estruturação da base de dados relacional bem como a utilização do serviço de autenticação para validação de entidade do utilizador. Focar-se-á na arquitetura utilizada no desenvolvimento do servidor, bem como na gestão do acesso à base de dados, onde se incluem tópicos como o isolamento de determinadas operações de acesso.

É também neste capítulo apresentada a interação esperada com o servidor, isto é, os recursos que o servidor disponibiliza, o formato dos pedidos ao servidor, o formato das respostas fornecidas pelo mesmo e o tipo de erros que poderão ocorrer.

3.1 Persistência de dados

A base de dados tem como responsabilidade o armazenamento de todos os dados da Web API. Este domínio engloba os dados pessoais dos utilizadores, das ideias criadas na plataforma e das interações existentes entre estes.

Inicialmente foi definida a entidade *USER*. Esta entidade representa um utilizador, o qual possui um identificador, estabelecido pela *Google Firebase Authentication* quando a conta é criada; um nome, email, uma descrição do perfil e o *URL* para o *blob* com a imagem do utilizador, guardado na *AWS S3*. Devido à utilização da *Google Firebase Authentication*, não existe necessidade de guardar dados sensíveis do utilizador para além do seu email, que é usado na troca de contactos após um utilizador ser aceite para uma ideia.

Definidos os utilizadores, procedeu-se ao desenvolvimento da entidade *IDEA*, que representa as ideias propostas pelos mesmo utilizadores, composta pelo seu identificador único, gerado pela própria base de dados, o identificador do utilizador que a criou, um título, uma descrição pequena e uma descrição detalhada, e a data de criação. A existência de duas descrições distintas pretende facilitar a explicação da ideia e a escolha da mesma por parte de outro utilizador para a realizar, uma explicação breve e sucinta de forma a estabelecer os pontos essenciais e ganhar o interesse do utilizador que se propõe; caso se consiga obter tal interesse, é apresentada a descrição detalhada onde a ideia é dissecada nas suas partes.

Com estas entidades concluídas, foi desenhada a entidade *CATEGORY*, que corresponde a uma categoria, na qual as skills serão inseridas de forma a dividir as mesmas, uma categoria é composta por um identificador único e um nome. Seguiu-se a criação da entidade *SKILL*, que representa as competências que os utilizadores têm, e também competências que são pedidas nas ideias. Esta entidade é composta por um identificador único, um nome e o id da categoria correspondente. Tendo as skills desenhadas, avançou-se para as entidades *USERSKILL* e *IDEASKILL*, que associam skills a utilizadores e ideias, respetivamente, através dos seus identificadores. Ambas as entidades têm ainda uma descrição que possibilita os utilizadores descreverem melhor essa skill associada.

Por fim foi desenhada a entidade *CANDIDATE*, que representa a candidatura de um utilizador a uma ideia, composta pelos identificadores destes e um estado da candidatura.

3.2 Web API

Uma *API*, *Application Programming Interface* ou em português, *Interface de programação de aplicações*, é um intermediário de software que permite que dois módulos de software distintos comuniquem entre si. No caso específico de uma *Web API* trata-se de um intermediário disponível via protocolo *HTTP* e que está a ser executado num sistema que atende esses mesmos pedidos e se prossupõem que tenha uma disponibilidade constante.

A *Web API* da plataforma *CrossWorking*, que cumpre as restrições *REST*, recebe pedidos em formato *JSON*, e responde também em formato *JSON*, exceto em situações de erro ou falha onde a resposta é fornecida em *PROBLEM/JSON*. Este serviço foi construído de acordo com o padrão de desenho *Model-View-Controller* (*MVC*). Este padrão separa a aplicação em 3 componentes, o modelo, responsável por toda a lógica da aplicação, a apresentação, que trata da disponibilização dos componentes ao utilizador e o controlador que liga estas duas partes. Assim sendo a *API* ficou composta por 5 partes:

- *Data Access Objects* (*DAOs*) – Interfaces que disponibilizam os contratos de acesso à base de dados relacional do servidor.
- *Models* – Objetos do domínio do servidor portadores da informação para todas as partes do sistema à exceção dos *controllers*.
- *Services* – Responsáveis pela lógica de negócio, são chamados pelos *controllers* e chamam os métodos das *DAOs* para obterem dados da base de dados.
- *Controllers* – Responsáveis por atender com os pedidos feitos à API e chamar o método conversor dos *DTOs* em objetos de domínio do servidor (*models*), para que todas as camadas inferiores apenas tenham conhecimento dos objetos de domínio.
- *Data Transfer Object* (*DTOs*) – Objetos usados na comunicação entre o cliente e o servidor através da *API*, ou seja, são estes objetos que serão instanciados a partir

da informação em formato *JSON* recebida nos pedidos do cliente (*requests*) e mapeados em *JSON* nas respostas do servidor (*responses*).

A *API* foi desenvolvida usando a *Framework Spring Boot* e a dependência *Spring Web MVC*, juntamente com a linguagem de programação *Kotlin* desenvolvida pela *JetBrains*. A escolha desta linguagem deteve-se devido à sua versatilidade, ao elevado grau de familiarização dos autores com a mesma, a possibilidade de conjugar princípios da programação funcional com programação orientada a objetos, e por fazer uso da *Java Virtual Machine* (abreviada para a sigla *JVM*), tornando a implementação compatível com um grande número de serviços de *cloud*, visto que a *JVM* representa uma grande quota de mercado e a grande maioria dos serviços de *cloud* garantem compatibilidade de hospedagem de aplicações que correm na *JVM*. Uma vez eleita a linguagem, tornou-se evidente a escolha da *Framework Spring Boot*, uma vez que se trata da mais utilizada na indústria para *JVM*, e oferece um elevado número de serviços uteis, bem como uma documentação completa e detalhada. Para esta eleição foram também tidas em conta as seguintes vantagens ao utilizar *Spring Boot* com a dependência *Spring Web MVC*:

- Promoção da construção de uma *API Restful*, seguindo o modelo *MVC*.
- Injeção de dependências.
- Anotações capazes de configurar objetos com papéis importantes, como um *@RestController*.
- Servidor *HTTP* pré-contruído com atendimento de pedidos e interpretação do protocolo.

3.2.1 Acesso à base de dados relacional

Para persistência dos dados foi escolhido utilizar *PostgreSQL*. Este é um sistema de base de dados relacional por objetos que estende a linguagem *SQL*. Esta escolha deve-se também ao facto do *PostgreSQL* ser *open-source* e ter mais compatibilidade com os serviços de hospedagem que foram considerados durante a conceção inicial do projeto.

De forma a comunicar com a base de dados decidiu-se utilizar a biblioteca *JDBI*, que permite a gestão de conexões e o mapeamento dos *resultSet* em objetos de forma automática. A implementação desta biblioteca é relativamente simples, e possibilita a utilização de scripts em *SQL* de forma a interagir com a base de dados.

3.2.2 Rotas da API

A *Web API* da plataforma *CrossWorking* disponibiliza um conjunto de recursos, disponíveis através de um rotas especificas, as quais permitem aos clientes comunicar com o servidor. O formato das respostas fornecidas perlo servidor é *JSON*.

Funcionalidades sem necessidade de autenticação:

- Obter as ideias mais recentes;
- Criar um utilizador;
- Obter lista de utilizadores;
- Obter um utilizador através do seu identificador;
- Obter lista de ideias de um utilizador;
- Obter ideia específica;
- Obter lista de categorias;

Funcionalidades que necessitam de sessão:

- Editar o perfil de um utilizador;
- Obter feed personalizado de um utilizador;
- Criar uma ideia;
- Editar uma ideia;
- Apagar uma ideia;
- Obter lista de candidatos a uma ideia;
- Adicionar candidato a uma ideia;
- Obter estado de uma candidatura;
- Aceitar ou rejeitar uma candidatura;
- Cancelar uma candidatura;
- Adicionar skill a um utilizador;
- Obter lista de skills de um utilizador;
- Obter skill específica de um utilizador;
- Apagar uma skill de um utilizador;
- Adicionar skill a uma ideia;
- Obter lista de skills de uma ideia;
- Obter skill específica de uma ideia;
- Apagar uma skill de uma ideia;

Se houver um erro a *API* responderá usando o *Media-Type Application/Problem+JSON*. Desta forma é possível retornar os erros descritas num padrão ideal para pedidos *HTTP*. Estes documentos têm o seguinte formato:

- *type*: Tipo de erro;
- *title*: Título descritivo do erro;
- *detail*: Descrição detalhado do erro;
- *status*: Código de erro;

3.2.3 Autenticação

A autenticação de um utilizador é uma ação extremamente complexa, uma vez que pressupõem a circulação de dados sensíveis, e que deverão ser protegidos, entre a aplicação apresentada ao cliente e o servidor usado como suporte. Assim é necessário cumprir determinadas regras de segurança que poderiam alargar o tempo necessário para desenvolver a plataforma *CrossWorking*, podendo mesmo comprometer a conclusão do projeto dentro do prazo estipulado.

Por forma a reduzir o tempo necessário para a implementação de um serviço de autenticação, são hoje disponibilizados diversos serviços se autenticação. Destes serviços foi tomada como opção a utilização do *Firebase Auth* nas aplicações *client side*, e *Firebase Admin* na aplicação *back-end* (*Servidor*).

O serviço *Firebase* foi selecionado pela sua fácil adaptação à *framework Android*, e pelos seus serviços extra que permitem criação de contas de utilizador e inicio de sessão através de contas já existentes noutras plataformas como o *Facebook* e o *Gmail*.

Quando o servidor *CrossWorking* recebe um pedido de criação de utilizador, guarda a instância de utilizador que recebe, a qual contem para além da informação providenciada pelo próprio utilizador, o identificador gerado pelo servidor de autenticação. Uma vez guardada esta instância o utilizador passa a estar efetivamente registado na plataforma. Os pedidos de obtenção e criação de um utilizador não têm associados qualquer tipo de autenticação, pelo que o header Authentication poderá ser enviado sem qualquer valor.

Uma vez iniciada a sessão do lado do cliente, todos os pedidos *GET* ao servidor passam a necessitar de um *Authorization Header*, header este que é gerado pelo servidor de autenticação *Firebase Auth*. Recebido o pedido no servidor *CrossWorking* o mesmo é intercetado num *filter*, este *filter* irá obter o *header* de autenticação e extrair o *token* do mesmo. Através de uma *API* disponibilizada pelo servidor de autenticação, denominada de *Firebase Admin*, é disponibilizado o método *verifyIdToken*, que descodificará e validará o *token* recorrendo ao servidor de autenticação. Se a validação suceder o pedido prosseguirá até um controlador que possa atender a rota do pedido, caso contrario, será gerada uma resposta de erro com o status code *401 Unauthorized*.

Caso a ação seja de *POST* ou de *PUT* a verificação ocorre exatamente da mesma forma, no entanto existe um procedimento adicional onde se extrai do *token* o identificador do utilizador ao qual o *token* pertence. Se o identificador extraído do *token* corresponder ao *identificador* do utilizador indicado no *path* o pedido é atendido por uma das rotas definidas nos controladores existentes, caso contrário será automaticamente gerada uma resposta com o código de erro *401 Unauthorized*.

4 Implementação do front-end

No presente capítulo será descrita a organização do *front-end* da plataforma *CrossWorking*. Será apresentada a estrutura da aplicação, o padrão de desenho adotado e as bibliotecas presentes. Existirá uma análise ao serviço de autenticação adotado, à implementação do serviço de imagens e a lógica necessária para o carregamento destas.

A par da estrutura técnica da aplicação *Android* será também descrita a interação existente entre o utilizador e a aplicação, as funcionalidades disponíveis, os prossupostos para a correta utilização da aplicação e ainda os requisitos mínimos para poder executar a aplicação sem esperar comportamento anómalo.

4.1 Organização

Em 2017 a *Google* durante o evento *Google I/O* introduziu o padrão *MVVM* como a arquitetura a seguir durante o desenvolvimento de uma aplicação *Android*, a introdução deste padrão trouxe consigo a existência de classes *ViewModel*, e a possibilidade de manter instâncias de *ViewModel* mesmo durante reconfigurações; resolvendo assim um problema estrutural da *Framework Android*, que ao destruir as atividades destruía também consigo os dados existente. Um ano após a recomendação da *Google*, surgiu junto da comunidade de desenvolvedores *Android* a aplicação de uma variante do *MVVM* denominada de *MVVM Clean*, sendo esta hoje a variante mais usada e mais recomendada.

Uma vez cientes da existência desta variante, foi decidido pelos autores tomarem como melhor opção o padrão de desenho *MVVM Clean*, tendo sido exatamente esse o padrão implementado.

A aplicação encontra-se assim estruturada em ecrãs, *ViewModels*, *use cases*, repositórios e serviços, o que por si só provoca uma grande necessidade de gestão de dependências. Como auxílio para a gestão de dependências foi, mais uma vez, tomada a sugestão da *Google* que recomenta a biblioteca de injeção de dependências *Hilt*. Esta biblioteca permite declarar as dependências injetáveis, e, através de anotações, injetar as mesmas nos construtores das classes. A conceção da aplicação *Android* teve recurso a muitas outras dependências, das quais se destacam as *coroutines*. As *coroutines* são rotinas cooperativas, isto é, são métodos, funções ou procedimentos que aceitam parar a sua execução, permitindo que outra rotina seja executada, enquanto aguarda que o resultado da chamada que dessepultou o parar da execução, para que possa depois retomar a execução, possivelmente noutra *thread*.

Os ecrãs da aplicação têm como incumbência a apresentação da informação ao utilizador e a captação de ações do mesmo, recebendo e transmitindo essa respetiva informação para a camada inferior, isto é o *ViewModel*. Por opção de desenho, cada ecrã da aplicação *CrossWorking* tem o seu próprio *ViewModel*, responsável por guardar os dados através das múltiplas reconfigurações que possam vir a acontecer, esta retenção de dados ocorre sobre a forma de estado, o que permite que a sua alteração despolete uma recomposição dos elementos visuais do ecrã; é também responsabilidade do *ViewModel* o lançamento e gestão das *coroutines*, onde irá ser executada a chamada aos métodos das camadas inferiores. Nas camadas inferiores ao *ViewModel*, todo o código será executado no contexto de uma *coroutine*, o que permitirá a realização de chamadas assíncronas.

Os *UseCases* são responsáveis pela implementação da lógica de negócio, realizar as chamadas ao repositório e devolver os resultados prontos a serem apresentados ao utilizador; por sua vez, o repositório realiza a gestão dos diversos fornecedores de dados, fornecendo uma interface uniforme para as camadas superiores. Por fim, o serviço tem como ónus realizar o acesso à *Web API* recorrendo para isso à biblioteca *Retrofit*, que permite a realização de pedidos *HTTP*, o mapeamento de objetos em *JSON* e de *JSON* em objetos do domínio da aplicação.

4.2 Jetpack Compose

No ano 2021, durante o *Google I/O* foi anunciado à comunidade de desenvolvedores android o *Jetpack Compose*; um kit de ferramentas que permite de forma moderna e declarativa, desenhar uma interface de utilizador nativa. Esta tecnologia rapidamente foi aplicada na indústria uma vez que permitia obter melhores resultados e facilitar a criação das interfaces gráficas. Mais recentemente empresas de grande dimensão começaram também a sua migração para a utilização de *Jetpack Compose* em substituição de interfaces *legacy*, desenhadas em *XML*.

Assim foi também adotado o *Jetpack Compose* como ferramenta para desenvolvimento da interface gráfica na aplicação *CrossWorking*. A utilização de *Compose* facilitou o desenvolvimento de algumas funcionalidades, mas visto tratar-se de uma tecnologia recente exigiu também um estudo pré desenvolvimento, e o desenvolvimento de alguns componentes que ainda não se encontram disponíveis na versão 1.0.1 do *Compose*.

5 Melhorias

Neste capítulo serão apresentados os pontos que poderão vir a ser alvo de melhorias, bem como funcionalidades que poderão vir a ser incrementadas. À data atual ainda se encontra por redigir.

6 Conclusões

Neste capítulo serão apresentadas as conclusões ao trabalho realizado, bem como algumas elações sobre as decisões tomadas. À data atual ainda se encontra por redigir.

Referências

[1] BusinessofApps. (). “Android Stats”, https://www.businessofapps.com/data/android-statistics. (Consulta em: 04/06/ 2022).

[2] GlobalStats. (). “Mobile Operating System Market Share Worldwide” https://gs.statcounter.com/os-market-share/mobile/worldwide