



1 2 9 0  
FACULDADE DE  
CIÉNCIAS E TECNOLOGIA  
UNIVERSIDADE DE  
COIMBRA



# Construção de QuizApp

Francisco Ribeiro [2017263852] - fpribeiro@student.dei.uc.pt

Jaime Domingos Marques [2017250310] - jdmarques@student.dei.uc.pt

João Nunes [2017247442] - joaonunes@student.dei.uc.pt

Pedro Miguel Santos [2017247870] - pmcsantos@student.dei.uc.pt



# Conteúdo

<b>1</b>	<b>Introdução</b>	<b>3</b>
<b>2</b>	<b>Requisitos e Arquitetura iniciais</b>	<b>4</b>
2.1	Requisitos . . . . .	4
2.2	Arquitetura . . . . .	6
<b>3</b>	<b>Gestão do Projeto</b>	<b>8</b>
3.1	Metodologia . . . . .	8
3.2	Documentos desenvolvidos e fluxo de trabalho . . . . .	8
<b>4</b>	<b>Implementação</b>	<b>18</b>
<b>5</b>	<b>Manual de Utilizador</b>	<b>20</b>
5.1	<i>Login e Sign Up</i> . . . . .	20
5.2	<i>Homepage</i> . . . . .	21
5.2.1	Criar Novo Quiz . . . . .	22
5.2.2	Pesquisar por Quiz . . . . .	23
5.2.3	Escolher um Quiz . . . . .	24
5.3	Perfil . . . . .	26
5.3.1	Editar Perfil . . . . .	26
5.3.2	Ver Histórico . . . . .	27
5.4	<i>Chat Global</i> . . . . .	28
5.5	Atividades Recentes . . . . .	29
5.6	Social . . . . .	29
5.7	<i>Leaderboard</i> . . . . .	30
<b>6</b>	<b>Conclusão</b>	<b>31</b>

# **Capítulo 1**

## **Introdução**

Durante o primeiro semestre do ano letivo de 2021/2022, foi proposto aos alunos que desenvolvessem uma aplicação utilizando *Android Studio* no âmbito da disciplina de Computação Móvel. Através de várias reuniões entre os elementos do grupo, ficou definido como projeto realizar uma aplicação de *quizzes*. Nesta aplicação seria importante o utilizador conseguir criar *quizzes*, fazer *quizzes* e verificar como se compara relativamente a outros utilizadores da aplicação. Os *quizzes* existentes são relativos a diferentes tecnologias relacionadas com programação, desta forma servindo como uma ferramenta lúdica para utilizadores que pretendem testar os seus conhecimentos ou, aprofundar mais em diferentes áreas.

## Capítulo 2

# Requisitos e Arquitetura iniciais

### 2.1 Requisitos

Inicialmente, foram escritos um conjunto de requisitos funcionais que faziam sentido integrarem a nossa aplicação. Estes requisitos foram organizados por prioridades, de acordo com o método de *Moscow*, por forma a ser despendido mais tempo a programar as funcionalidades nucleares e obrigatórias para o funcionamento da aplicação. A **verde** e representado pela letra *M* (*Must have*), os requisitos mais importantes, a **amarelo** (*Should have*), indicando requisitos de prioridade média, a **vermelho** (*Could have*) os extras que acrescentariam valor ao projeto, mas que não seriam estritamente necessários estarem presentes. Finalmente marcamos com a letra *W* (*Won't have*) os requisitos que não iriam ser implementados numa primeira fase, mas que poderão vir a ser boas adições no futuro.

Tema	Requisito	Prioridade
Quizzes	Ver Quizzes predefinidos	(M)
	Pesquisar Quizzes	(M)
	Escolher Quiz	(M)
	Ver detalhes do Quiz	(M)
	Iniciar Quiz	(M)
	Responder ao Quiz	(M)
	Ver pontuação em cada resposta	(M)
	Ver resultado final do Quiz	(M)
	Criar um Quiz customizado	(S)
	Modificar opções de criação de Quizzes (dificuldade, número de perguntas, tópico)	(S)
Chat	Partilhar um Quiz	(C)
	Aceder ao chat	(M)
	Iniciar Quiz	(M)
	Ler mensagens globais	(M)
	Ler mensagens das salas de chat dos quizzes	(S)
Perfil e Estatísticas	Escrever uma mensagem no chat escolhido	(M)
	Registo pela plataforma	(M)
	Registo por Facebook e Google	(C)
	Login	(M)
	Aceder ao perfil próprio	(M)
	Alterar nome	(M)
	Adicionar imagem de perfil por imagem guardada	(C)
	Adicionar imagem de perfil por câmera	(C)
	Alterar biografia	(M)
	Ver as minhas melhores pontuações	(M)
	Ver percentagem de respostas certas	(M)
	Ver percentagem de respostas certas por tópico de Quiz	(M)
	Ver histórico de Quizzes	(S)
	Ver informação de cada Quiz do histórico (perguntas certas, tópico e número de perguntas, tempo gasto)	(S)
Atividade Recente	Ver leaderboard global (pontuação somada/número de perguntas*dificuldade)	(M)
	Ver leaderboards de quizzes	(M)
	Ver lista de últimos resultados dos utilizadores da app	(S)
Social	Adicionar amigos	(W)
	Procurar Perfis	(S)
	Ver perfil de utilizadores	(S)
	Remover amigos	(W)

## 2.2 Arquitetura

A arquitetura inicialmente pensada para o trabalho foi a seguinte (visível na figura 2.1): 5 componentes *View*, *ViewModel*, *Repository* e *Room*. A *View* contém as atividades e fragmentos que apenas lidam com atualizações de UI e fica à escuta por alterações nos objetos *LiveData* do seu *viewmodel*, seguindo o princípio *Observer*. O *ViewModel* é responsável por observar o estado no ciclo de vida da *View*. Este nunca referencia a *View* porque as atualizações dos dados são feitas pelos listeners que escutam os objetos *LiveData*.

O *Repository* é a classe que faz os acessos às fontes (Base de dados e API externa), transforma os dados em objetos *LiveData* e coloca-os nos respectivos *viewmodels*. *Room* é uma classe que facilita os acessos à base de dados e acesso à memória interna do dispositivo.

Quanto à base de dados, foi planeado usar o *Firebase Authentication* como gestor de contas e autenticações, *Firebase Storage*, para armazenar os conteúdos de multimédia, o *Firebase Realtime Database*, uma base de dados adequada para tratar dados que necessitam de rápida sincronização, como as mensagens do *chat*, e o *Firebase Firestore Database* uma base de dados *NoSQL* que permite guardar as estruturas definidas pelo diagrama da figura 2.2. A informação relativa às perguntas é obtida através de uma REST API chamada *QuizAPI*<sup>1</sup>. São enviados os parâmetros de categoria, dificuldade, número de perguntas e *tags*, sendo certo que a lista de perguntas é retornada num ficheiro com formato JSON.

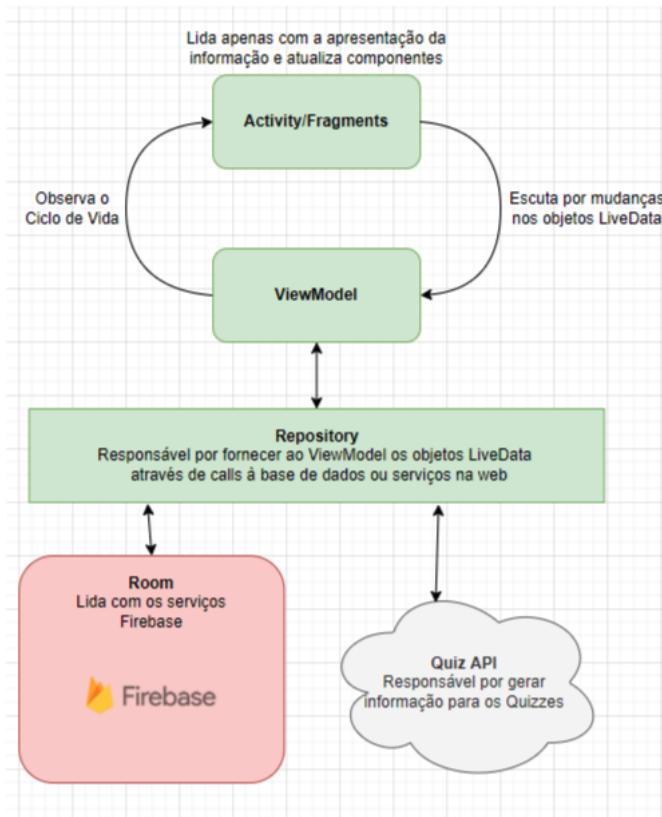


Figura 2.1: Arquitetura definida no documento de requisitos

<sup>1</sup>Link para a API utilizada: <https://quizapi.io>

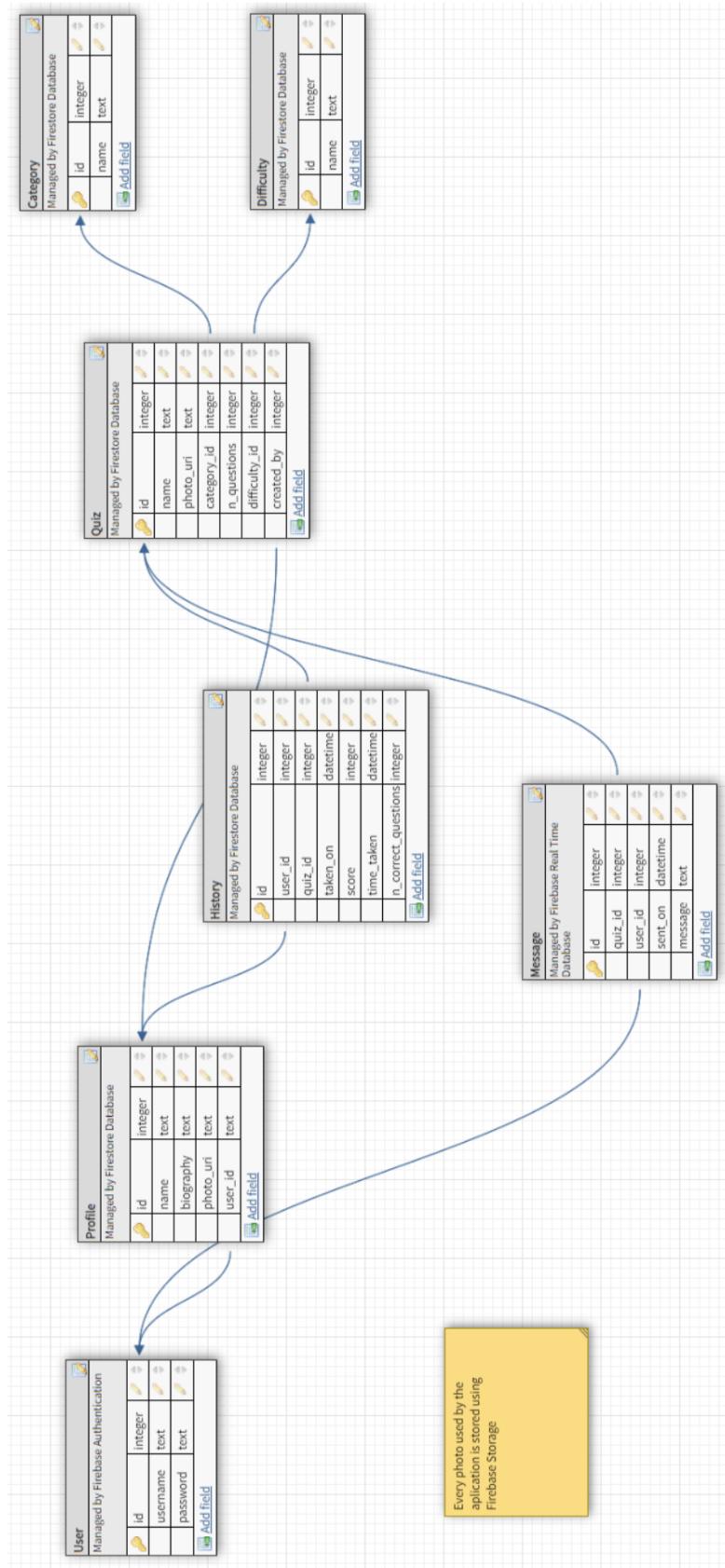


Figura 2.2: Diagrama da base de dados definido no documento de requisitos

# Capítulo 3

## Gestão do Projeto

### 3.1 Metodologia

Na realização deste projeto foi adotada a metodologia *Scrum*. Como tal, foram definidos os papéis adequados como representados a seguir:

- *Scrum Master* - Francisco Ribeiro
- *Product Owner* - Todos
- *Developers* - Todos

De forma a monitorizar as tarefas a serem executadas foi utilizado o Trello. A cada *sprint* era feita uma *review* do trabalho desenvolvido e colocadas novas tarefas para o *sprint* seguinte. Aquando da escrita deste documento, o Trello encontra-se como representado na imagem 3.1.

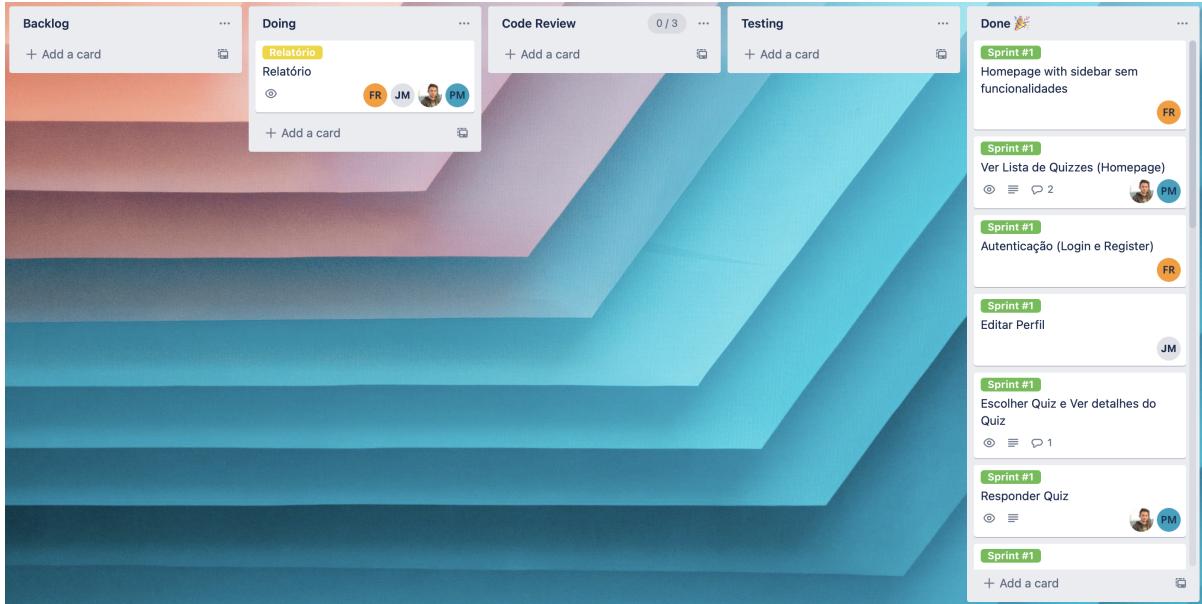


Figura 3.1: Trello onde se manteve registo das tarefas a ser executadas e por quem

### 3.2 Documentos desenvolvidos e fluxo de trabalho

Ao longo da execução deste projeto, foram desenvolvidos vários artefactos. Dentro das quais podemos distinguir:

- Documento de arquitetura e requisitos
- Documento de design
- *Mockups*
- *Spreadsheet de bugs*
- Relatório

Como se pode ver na imagem 3.2<sup>1</sup> o trabalho foi distribuído por *sprints*. Cada *sprint* tem o objetivo de terminar um dado número de tarefas.

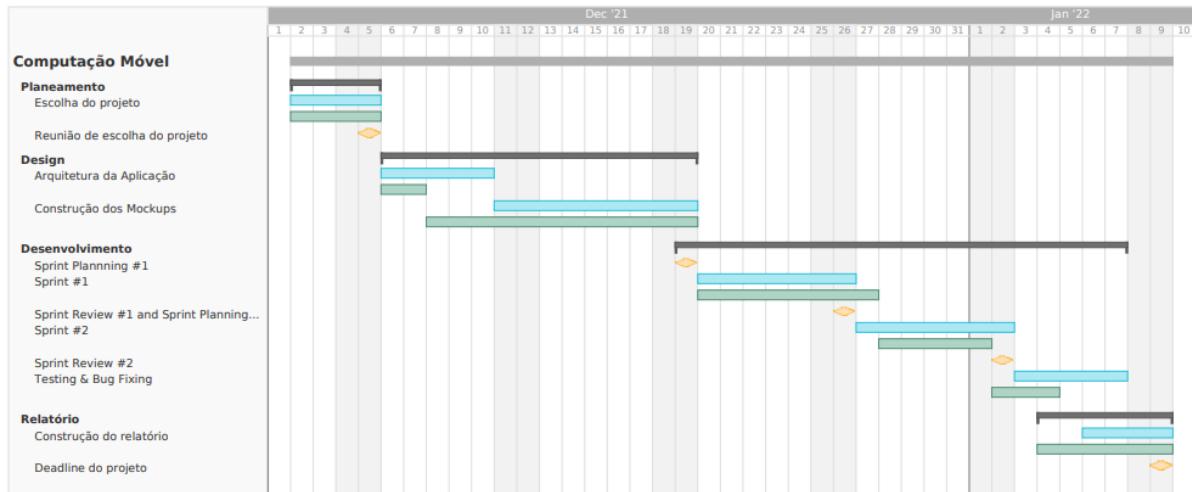


Figura 3.2: Diagrama de Gantt que descreve os *sprints* executados

No diagrama apresentado na figura 3.2 é possível discernir entre 3 cores utilizadas, que funcionam da seguinte forma:

- Azul - Duração prevista para uma dada tarefa
- Verde - Duração real da tarefa
- Amarelo - Fim de uma dada meta

Seguindo uma ordem cronológica, inicialmente foi desenvolvido o documento de arquitetura e requisitos de modo a deixar claro desde o princípio quais seriam os objetivos da aplicação. Através de detalhes claros acerca de como cada funcionalidade deveria interagir com o resto da plataforma, e com especificações que permitem deduzir prioridades entre as funcionalidades, o objetivo tornou-se claro.

Tendo o objetivo e as funcionalidades em mente, foi desenvolvido o documento de design. Neste documento organizaram-se as funcionalidades por ecrãs, auxiliando desta forma a criar uma imagem de como seria o *layout* da plataforma. De modo a manter um design consistente para os *mockups* que se seguiram, foi feita pesquisa para encontrar elementos de Figma pré-configurados de modo a melhor se ajustarem ao que se poderia esperar de uma interface de *android*.

Após a definição dos ecrãs no documento de design, o grupo dividiu-se de modo a conseguir fazer os *mockups* de cada ecrã. Utilizando a ferramenta do Figma e através de reuniões entre os membros, o design ficou definido como apresentado da imagem 3.3 à imagem 3.10.<sup>2</sup>

<sup>1</sup>Imagem original será submetida em conjunto com o trabalho

<sup>2</sup>A página Figma que contém os *mockups* pode ser encontrada em: <https://www.figma.com/file/5VqpP6dnRY0QyK2Xd2UU0b/MockupsCM?node-id=0%3A1>

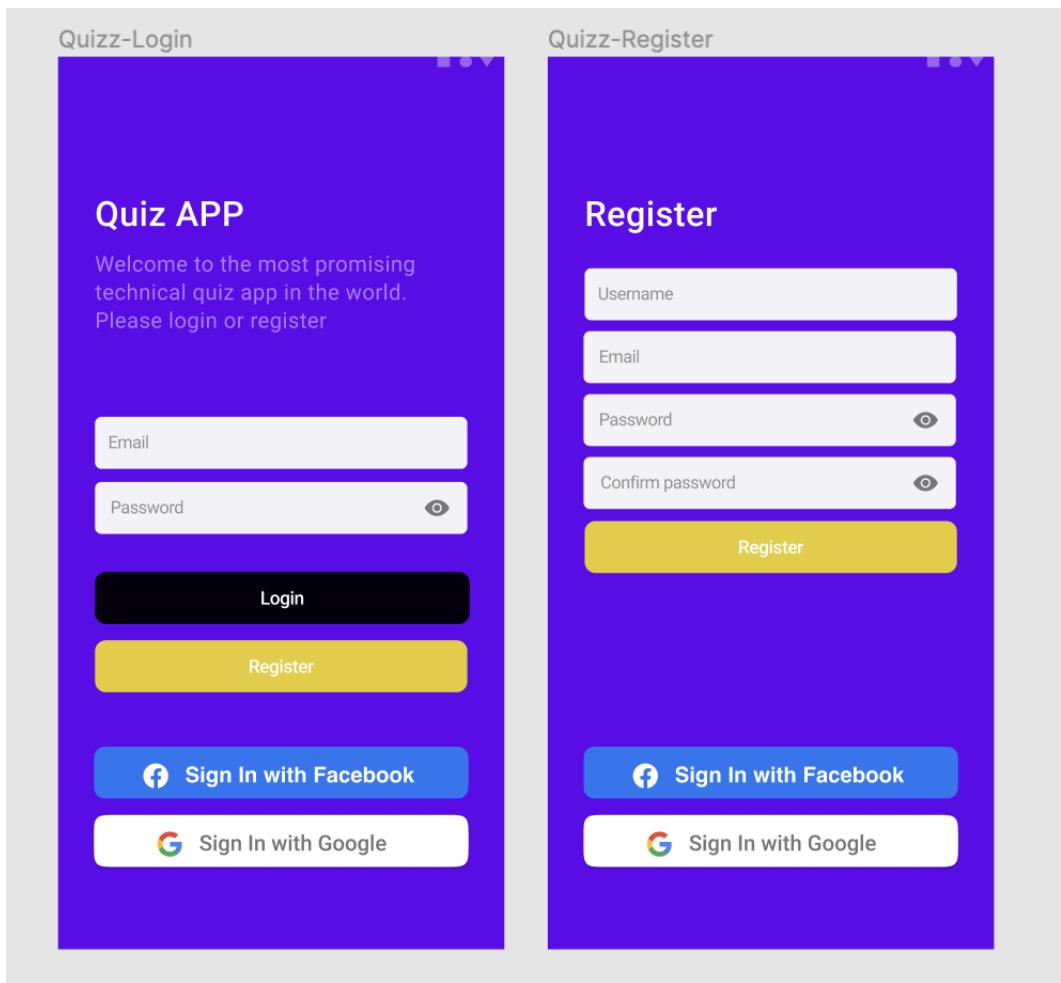


Figura 3.3: Ecrãs *login* e registo

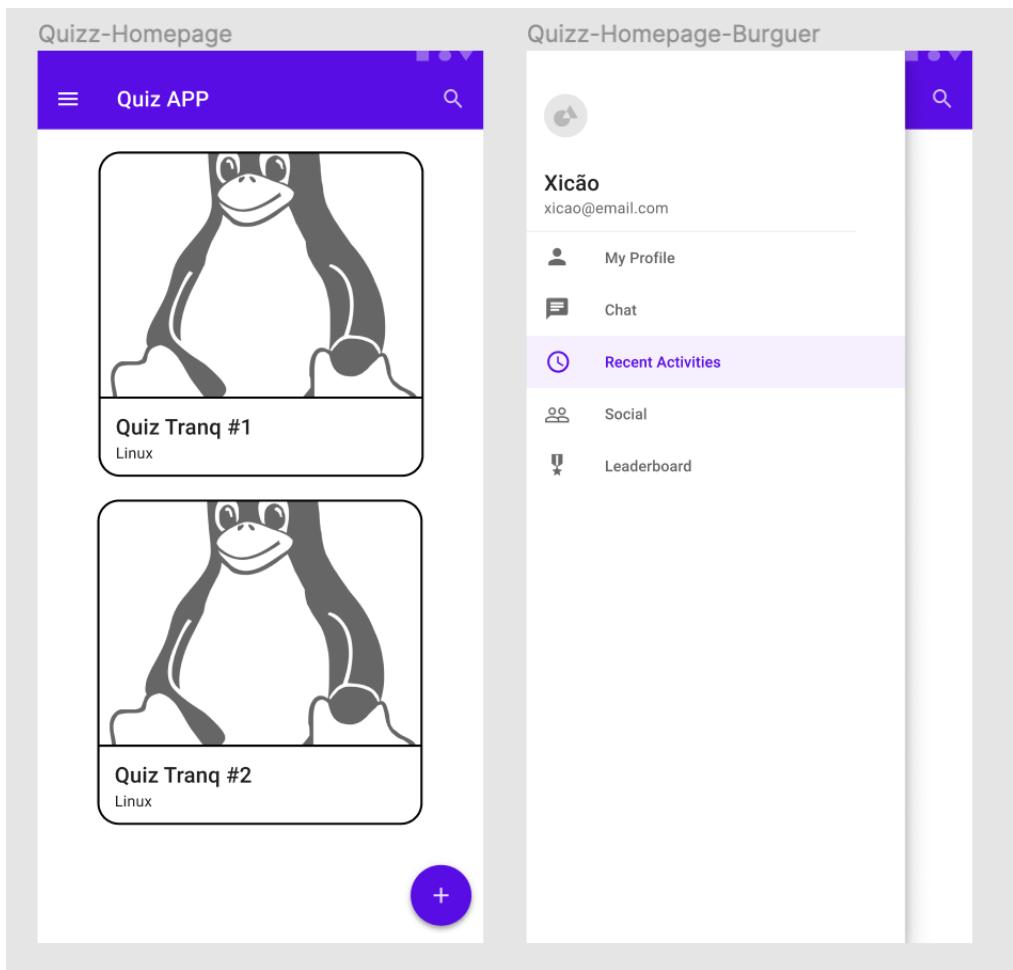


Figura 3.4: Página inicial da aplicação

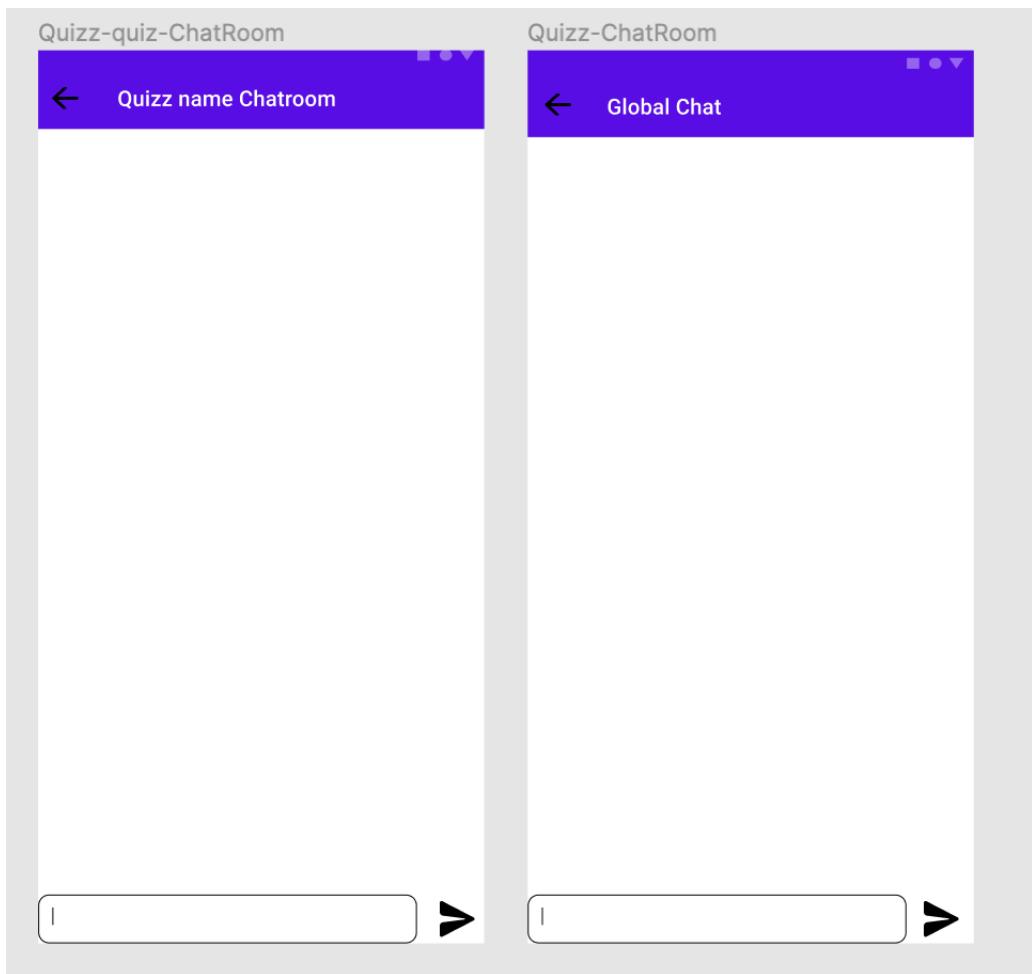


Figura 3.5: *Chatrooms* de quiz e global, respetivamente

Figura 3.6: Ecrãs de realização de quiz

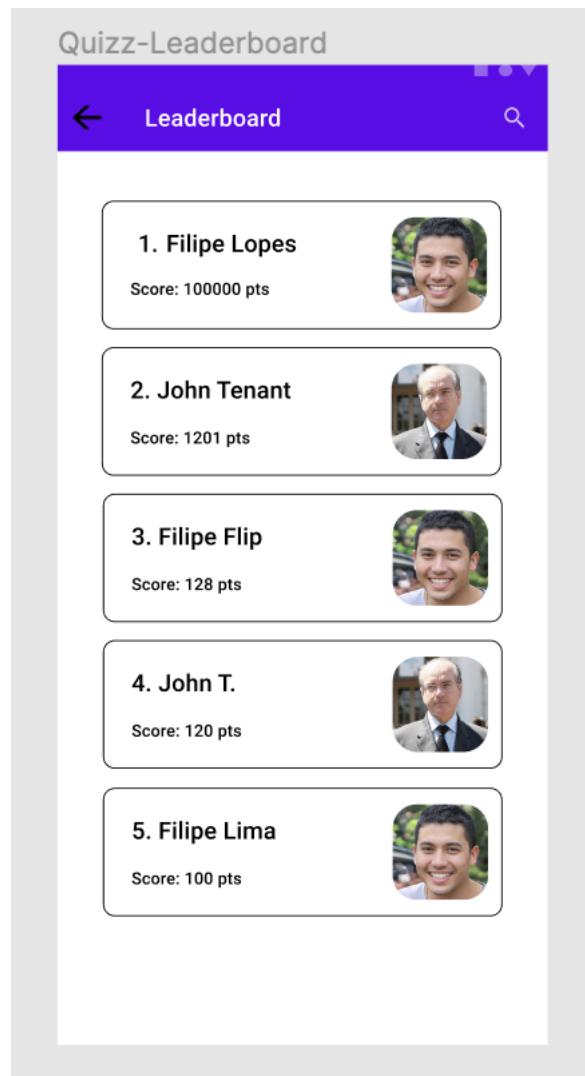


Figura 3.7: Ecrã de *leaderboard*

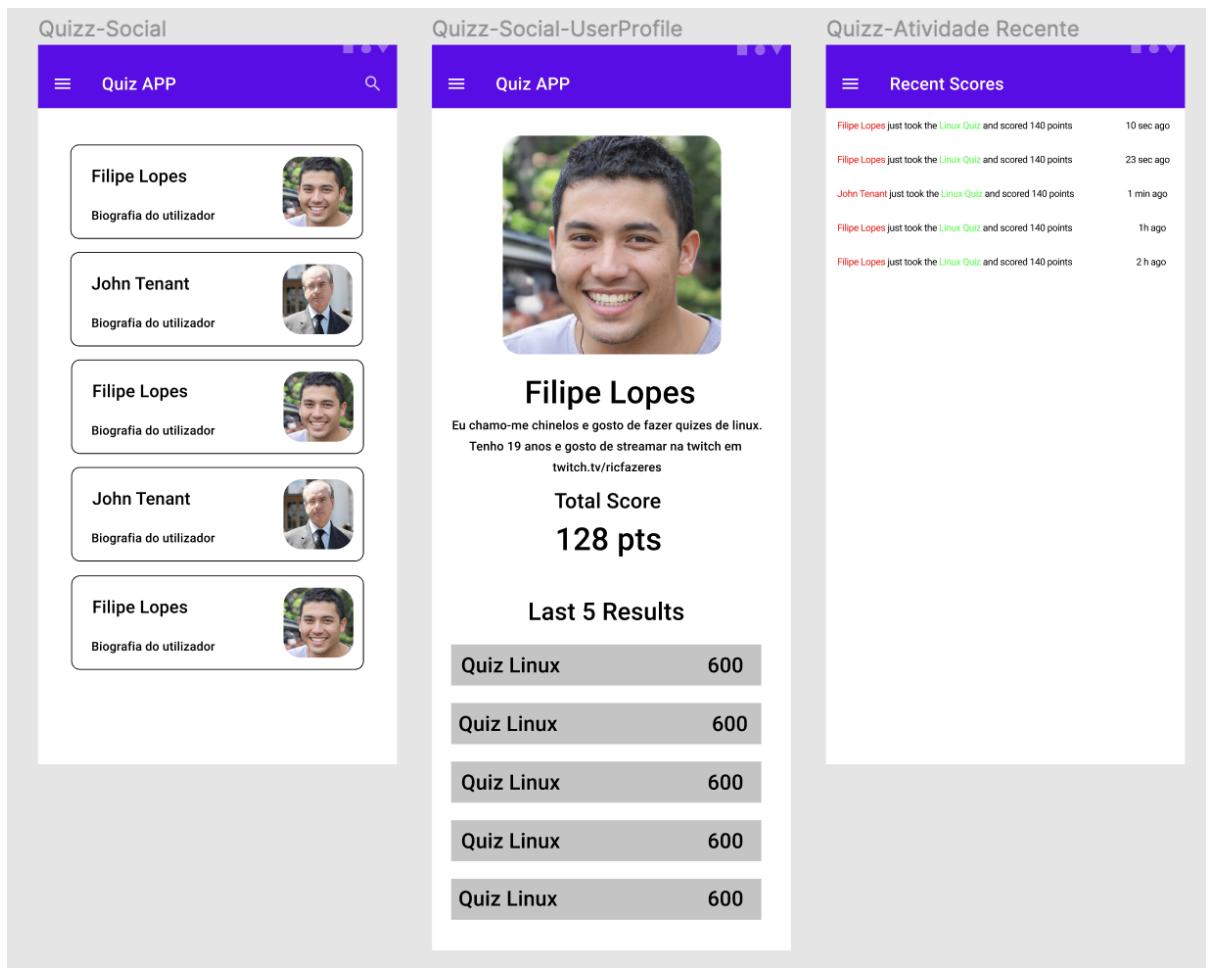


Figura 3.8: Ecrãs com todos os utilizadores, perfil de outros utilizadores e atividade recente

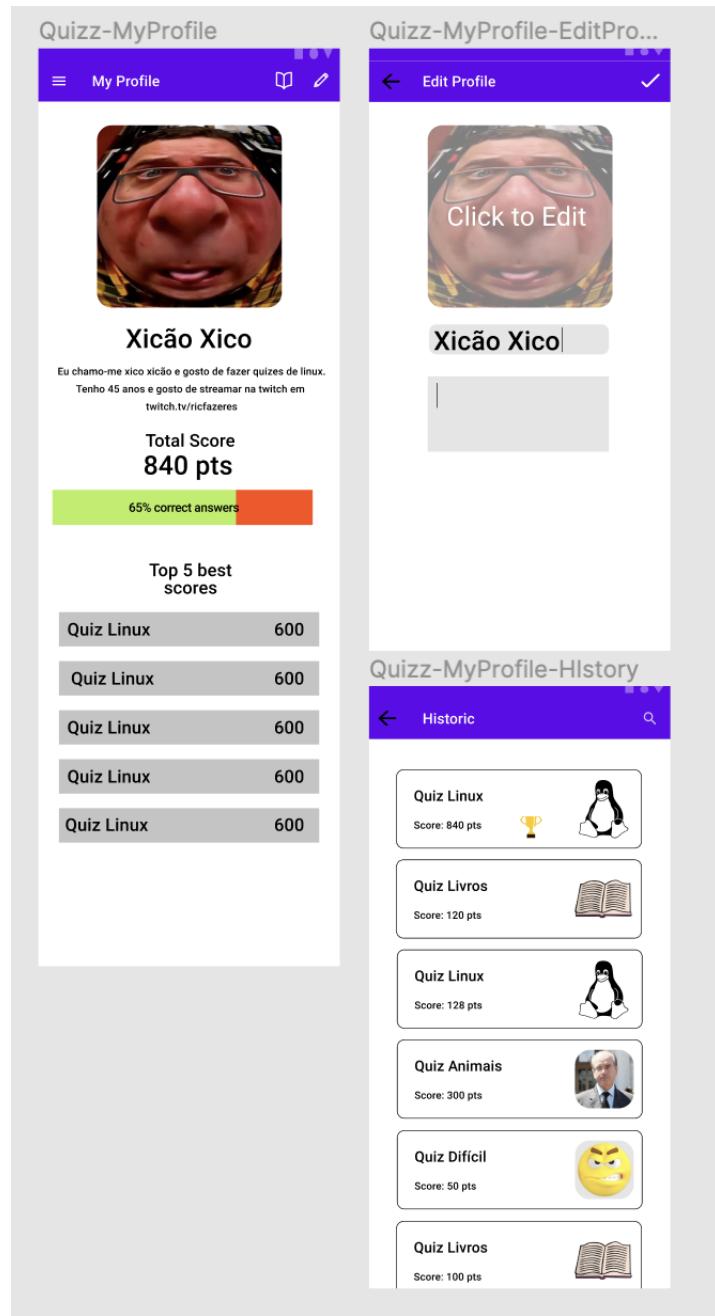


Figura 3.9: Ecrãs de Perfil de utilizador

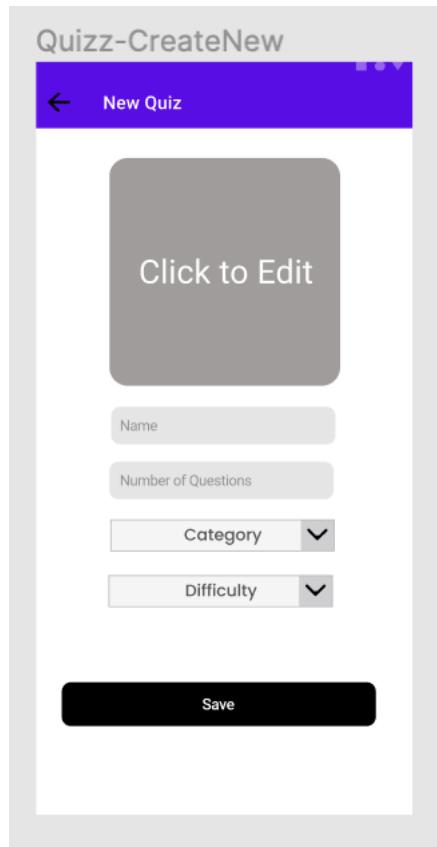


Figura 3.10: *Dialog* para criar quiz

Aquando do término de realização de *mockups*, chega o momento na qual a implementação da aplicação se torna possível, uma vez que as funcionalidades estão especificadas e o design está completo e visível.

Desta forma iniciou-se a implementação de cada ecrã. Para tal, o grupo dividiu-se de novo. Utilizando a ferramenta Trello para visualizar o progresso de cada um na tarefa em mãos e Github como repositório, cada elemento do grupo trabalhou individualmente ou em grupo de 2 para implementar cada ecrã (ou sequência de ecrãs que aplicam uma dada funcionalidade, ou que se relaciona entre todos). Com reuniões bi-semanais, o grupo juntou-se diversas vezes de modo a conseguir focar atenção em áreas na qual pudesse existir problemas ou dificuldades.

Ao terminar a implementação do projeto, foi construído um documento com o objetivo de manter registo de bugs encontrados. Neste documento, para além do problema encontrado, era também relatado o tipo de problema, que se pode dividir em:

- *Behavior* - Uma dada funcionalidade não funciona como suposto
- *Layout/Design* - Algum elemento da interface não tem uma aparência correta
- *Performance* - O método de executar alguma tarefa não era o mais eficaz e poderia ser melhorado

Para além destes dois campos, é também visível qual o ficheiro (.java/.xml) na qual se encontra o problema, bem como uma possível solução. Para terminar, esse documento tem ainda colunas com o responsável por resolver o problema e o estado de resolução:

- *To do*
- *Doing*

- *Fixed*

Este documento consiste numa tabela que pode ser vista na figura 3.11<sup>3</sup>.

No	Problema	Tipo	Passos para reprodução	Localização (ficheiro(s))	Comportamento ideal/Soluções	Assignee	Status
1	Back action bloqueado para todos os ecrãs	Behaviour	Todos os fragmentos depois dos nav tentar premir o back do telemóvel	Todas	Deixar utilizar o back e bloquear apenas nos quiz	Francisco	Fixed
2	Layout dos botões de resposta desformatados	Layout/Design	Quando se responde a uma pergunta os botões ficam desformatados devido ao facto de alguns terem respostas mais longa que outras	QuizAnswerFragment	Adaptar os diferentes botões com base no tamanho da resposta	Pedro	Fixed
3	Chamadas repetidas à QuizAPI	Performance	Iniciar um quiz e ver os logs	QuizAnswerFragment	Adaptar o texto dos botões e tentar meter o botão sempre com o mesmo height	Pedro	Fixed
4	Um dos quizzes iniciais dá 404 no Quiz API	Behaviour	Apagar os quizzes da firestore e iniciar a app	Repository	Meter quizzes predefinidos aceites e já prontos para mostrar ao stor	Pedro	Fixed
5	Melhorar a apresentação dos detalhes do quiz	Layout/Design	Entrar em detalhes de um quiz	fragment_quiz_details.xml	Provavelmente centrar o texto. Aumentar text size em alguns. Mexer com cores. Ficar bonito	Francisco	Fixed
6	Botão de next question pode ficar omitido Melhorar a apresentação do resultado final do quiz	Layout/Design	Responder a um quiz com respostas com textos muito grandes e se houverem respostas suficientes o botão é empurrado para fora do screen e o user não consegue clicar nele	fragment_quiz_answer.xml	Provavelmente alterar o layout para um constraint layout e meter um button constraint no botão para nunca deixar ir para fora	Pedro	Fixed
7		Layout/Design	Finalizar um quiz	fragment_final_results.xml	Mesmas indicações que o Bug #5	Pedro	Fixed
8	Facebook login não funciona	Behaviour	Tentar entrar com Facebook e voltar ao login screen	HomepageActivity.java	Conseguir fazer login com Facebook	Francisco	Closed
9	Google login por vezes não funciona se o token não estiver registado na Firebase console	Behaviour	Tentar entrar com Google voltar ao login screen	HomepageActivity.java	Conseguir fazer login com Google	Francisco	Closed
10	Melhorar a apresentação do login screen	Layout/Design	Entrar no login screen da app	HomepageActivity.java	Tentar meter o título da app e tentar reproduzir o design do figma	Francisco	Fixed
11	Melhorar a apresentação do New Quiz	Layout/Design	No home, clicar no fab button	fab_dialog.xml	Tentar dar paddings, centrar textos, etc	Pedro	Fixed

Figura 3.11: Tabela de bugs encontrados

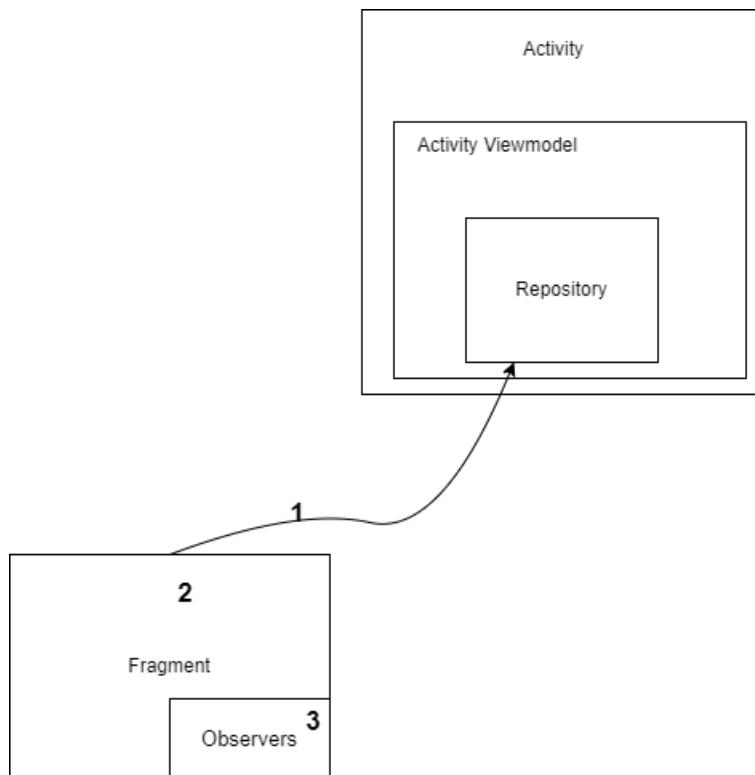
---

<sup>3</sup>Para visualizar melhor a tabela, deixamos o link para aceder em: <https://docs.google.com/spreadsheets/d/1bSKf5bWg5raNY0Nr7Xy xvSxURkfGyB16w0FR0fo5V-w/edit?usp=sharing>

## Capítulo 4

# Implementação

Durante o decorrer do projeto, foi necessário adaptar o modelo descrito no terceiro capítulo tanto na arquitetura como na base de dados. Cada fragmento contém o seu próprio *viewmodel* responsável por fornecer os dados para aquele fragmento. Na arquitetura foram centralizados todos os *viewmodels* na instância da classe *repository* (que vive no *viewmodel* da atividade) de modo a ter as mesmas referências para todos os locais na aplicação. Desta maneira, a figura 4.1 demonstra o típico flow em cada fragmento de modo a obter o seu *viewmodel* e inicializar os seus observadores.



- 1 - Gets repository instance from activiti'e's viewmodel
- 2 - Checks if the *viewmodel* from that *fragment* exists and it's initialized. If not it creates a new one and sets it on the *repository* instance.
- 3 - With the *viewmodel* defined and centralized, the fragment can instantiate the observers and the whole application has the same scope so it can update any *viewmodel*.

Figura 4.1: Típico flow de cada fragmento

Na base de dados foi necessário rever a estrutura da mesma. Como foi utilizado *Firebase* como *backend* da aplicação, os serviços de base de dados que oferecem seguem a tecnologia

NoSQL. NoSQL estrutura os dados como uma árvore em que o elemento inicial é a instância da base de dados que pode ter várias coleções como folhas. Por sua vez, estas coleções têm documentos e estes documentos têm campos do tipo chave-valor. Isto significa que o modelo relacional que foi desenhado não pôde ser implementado com as relações previstas. Para manter as relações, foi necessário adicionar campos de modo a representarem chaves estrangeiras. Na figura 4.2 é apresentado um exemplo da *firebase firestore* de desenvolvimento onde podemos ver as coleções, documentos e campos de cada documento em que dois destes campos são as chaves estrangeiras que apontam para outros documentos noutras coleções através do seu id.

Figura 4.2: Base de dados *Firebase Firestore* de desenvolvimento

A implementação deste projeto utilizou várias dependências sendo a principal a *Navigation Component*. Esta nova tecnologia permite que o desenvolvedor não tenha que gerir a navegação da sua aplicação manualmente através dos *Fragment Managers*, como lecionado nas aulas da cadeira. Para se utilizar esta tecnologia foi necessário criar um ficheiro *navigation XML* que define todos os fragmentos da aplicação bem como as suas ações ou destinos.

Para a implementação da autenticação, foi utilizado *Firebase Pre-built Sign-in* que permite a autenticação por vários fornecedores distintos através de componentes UI já feitos. Estes fornecedores podem ser *Google*, *Facebook*, *Apple*, *Github*, entre outros. Foi apenas necessário configurar este modo de autenticação na consola *Firebase* e alguns passos na plataforma do fornecedor caso seja necessário.

Para a comunicação com a API mencionada, foi utilizada outra dependência chamada *Volley*<sup>1</sup>. *Volley* é uma livraria de HTTP que facilita o processo de obter dados da internet através de requests HTTP. Apenas foram utilizados simples *GET requests* para comunicar com a API que devolvem a resposta em formato JSON que é lida num *callback*.

<sup>1</sup>*Volley*: <https://developer.android.com/training/volley>

## Capítulo 5

# Manual de Utilizador

Neste capítulo será apresentada a aplicação em termos de todos os seus ecrãs de modo a um novo utilizador poder perceber o funcionamento desta.

### 5.1 *Login e Sign Up*

Ao abrir a aplicação o utilizador será encaminhado para um ecrã onde poderá iniciar a sessão tanto por email, pela conta do Google ou do Facebook.



Figura 5.1: Ecrã de *login*

Iniciando a sessão através de email, caso o utilizador insira o seu email e este não exista na base de dados irá receber dados adicionais para preencher, o nome e password de acesso à aplicação.

Inscrever-se

Email  
utilizador.cm@email.com

Nome próprio e apelido  
Filipe Lopes

Nova palavra-passe  
.....

GUARDAR

Figura 5.2: Ecrã de *sign up*

## 5.2 Homepage

Após a entrada no sistema o utilizador será encaminhado para a página principal da aplicação onde se encontra uma lista de *quizzes* predefinidos pela aplicação ou criados por outros utilizadores do sistema.

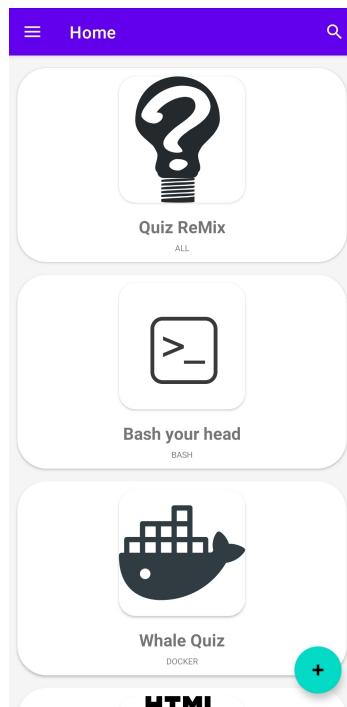


Figura 5.3: Página inicial da aplicação

No canto superior esquerdo existe um *drawer* que, quando o utilizador clica neste, apare-

cerão as diferentes secções da aplicação, que serão explicadas individualmente mais à frente neste manual, assim como dar *logout* da aplicação.

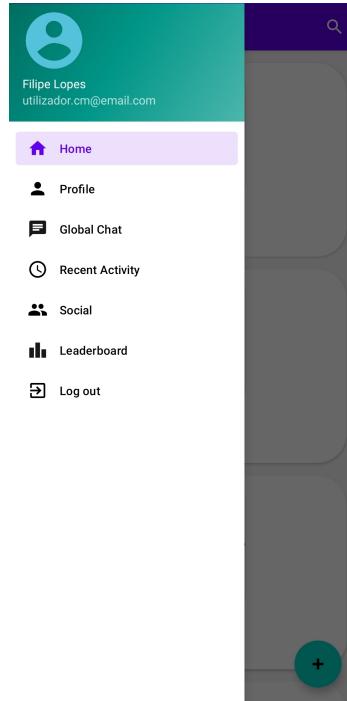


Figura 5.4: *Drawer* da aplicação

Focando agora nas funcionalidades que aparecem na página principal da aplicação podemos destacar o criar um novo *quiz*, pesquisar por um *quiz* e escolher um *quiz* para realizar.

### 5.2.1 Criar Novo Quiz

Para criar um novo *quiz* selecione o *floating action button* (FAB) que está presente no canto inferior direito. Após esta ação um *popup* irá aparecer com todos os detalhes para a criação do novo *quiz*:

- Nome do *quiz*
- Número de perguntas
- Categoria
- Dificuldade
- Imagem - facultativa, caso o utilizador não selecione uma imagem, a imagem do *quiz* será uma imagem *default* respetiva à categoria do *quiz*

#### Notas:

1. Caso ainda não tenha dado permissão, é necessário que se dê permissão à aplicação para aceder à galeria do telemóvel.
2. Existem ainda algumas limitações na criação de um *quiz* (aparecendo um *toast* de erro caso não consiga gerar o novo *quiz*):
  - (a) O número máximo de questões para um *quiz* é 20
  - (b) O *quiz* não pode ter o mesmo nome que um *quiz* existente

- (c) A combinação de detalhes (número de perguntas, categoria e dificuldade) não podem ser as mesmas que um *quiz* já existente

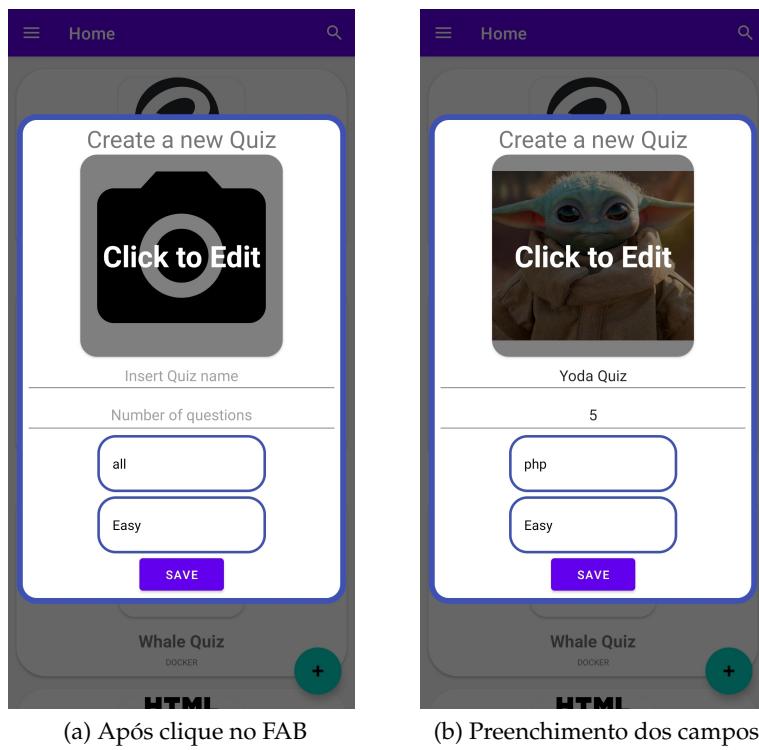


Figura 5.5: Ecrã para criação de um *quiz*

### 5.2.2 Pesquisar por *Quiz*

Após a criação de um novo *quiz*, este já se encontra na lista de *quizzes* que aparecem na página principal. Para facilitar a sua localização poderá selecionar a lupa no canto superior direito e pesquisar pelos *quizzes* existentes (por nome ou por categoria).

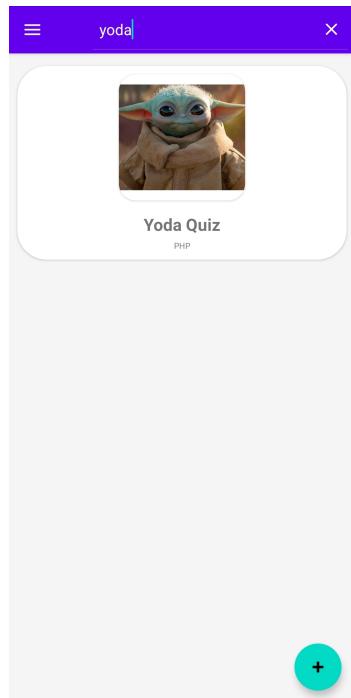


Figura 5.6: Pesquisa de *quizzes*

### 5.2.3 Escolher um *Quiz*

#### Detalhes do *Quiz*

Selecione um *quiz* e será redirecionado para um ecrã que terá todas as informações do *quiz*. Na barra de navegação poderá também aceder ao *chat* e *leaderboard* do *quiz* em específico (semelhantes ao *chat* e *leaderboard* global, apresentados na secção 5.4 e na secção 5.7, respetivamente).

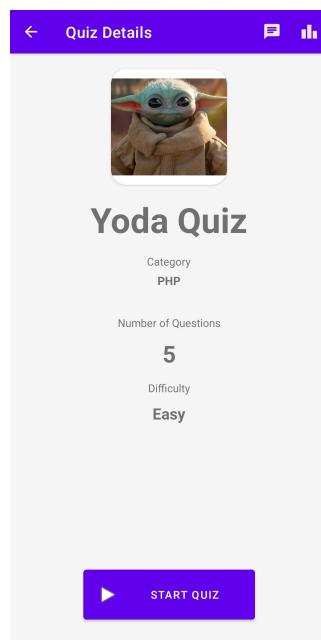
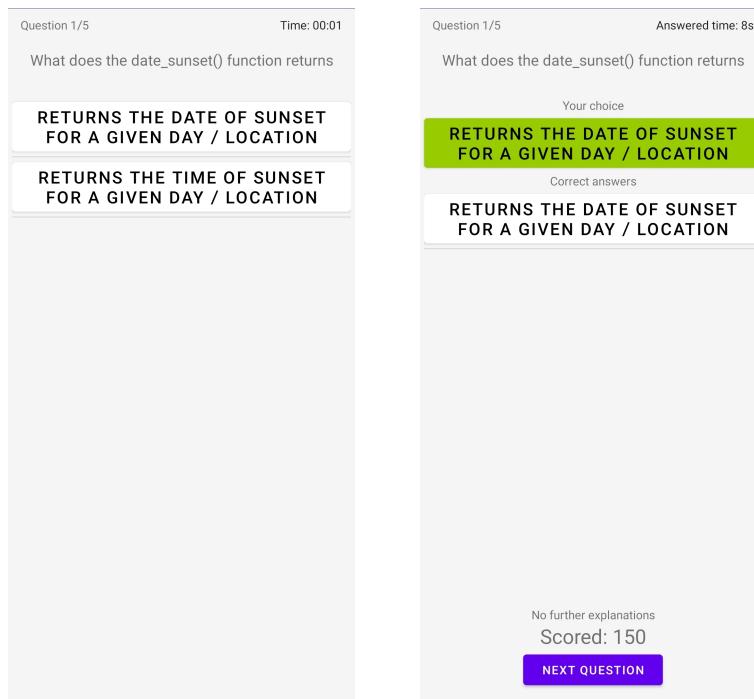


Figura 5.7: Ecrã dos detalhes de um *quiz*

## Responder a Quiz

Selecione o botão “Start Quiz” e o *quiz* será inicializado.

Este terá uma sequência de ecrãs com pergunta de escolha múltipla para o utilizador poder responder. No canto superior direito existe um cronómetro que contará o tempo que o utilizador demorou a responder a uma pergunta (quanto mais tempo o utilizador demorar a responder a uma pergunta menos pontuação terá). Após responder a uma pergunta existirá um ecrã para o utilizador ver todos os detalhes dessa pergunta e verificar se acertou ou errou a questão, apresentando o *score* no fundo do ecrã assim como um botão para prosseguir para a próxima questão.



(a) Pergunta e Respostas

(b) Veredito da resposta

Figura 5.8: Ecrãs de resposta ao *quiz*

Após repetir este processo para todas as questões do *quiz*, chegará a uma página com os resultados finais do *quiz* que acabou de realizar, apresentando o número de questões corretas, o tempo gasto a realizar o *quiz* e a pontuação final.

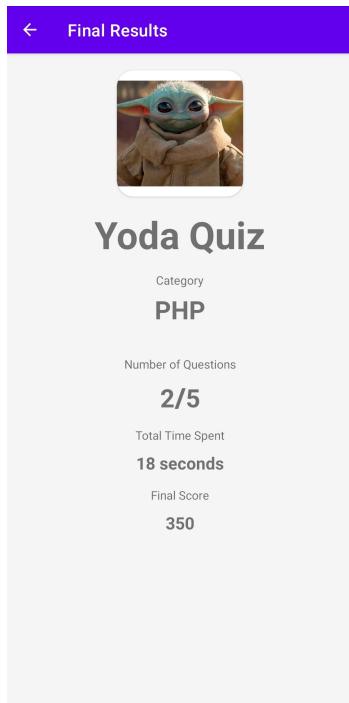


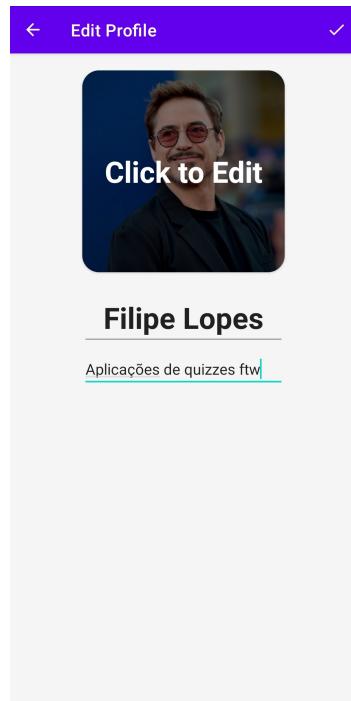
Figura 5.9: Ecrã de resultado final do *quiz*

### 5.3 Perfil

De regresso à página inicial após a realização de alguns *quizzes*, selecione o *drawer* e escolha a opção “Profile”. Aqui poderá visualizar a sua foto, o nome, biografia, pontuação total na aplicação, percentagem de respostas corretas mediante a todos os *quizzes* realizados e o top 5 melhores resultados de *quizzes* feitos.

#### 5.3.1 Editar Perfil

Selecione o lápis no canto superior direito do ecrã para editar o seu perfil: foto, nome e biografia. Edite o perfil como pretender e selecione o visto no canto superior direito para gravar as alterações. Voltando à página de perfil poderá ver as alterações ao perfil que foram feitas.



(a) Editar o perfil



(b) Perfil editado

Figura 5.10: Ecrãs do perfil do utilizador

### 5.3.2 Ver Histórico

Selecione o livro à esquerda do botão de editar perfil para visualizar todo o seu histórico da aplicação. Se pretender pode clicar num dos históricos para rever os detalhes do *quiz* que realizou no passado.



Figura 5.11: Ecrãs do histórico do utilizador

## 5.4 Chat Global

Selecione o *drawer* e escolha a opção “Global Chat”. Envie uma mensagem para que os restantes utilizadores da aplicação o Guyem a conhecer.



Figura 5.12: Ecrã do *chat* global com todos os utilizadores da aplicação

## 5.5 Atividades Recentes

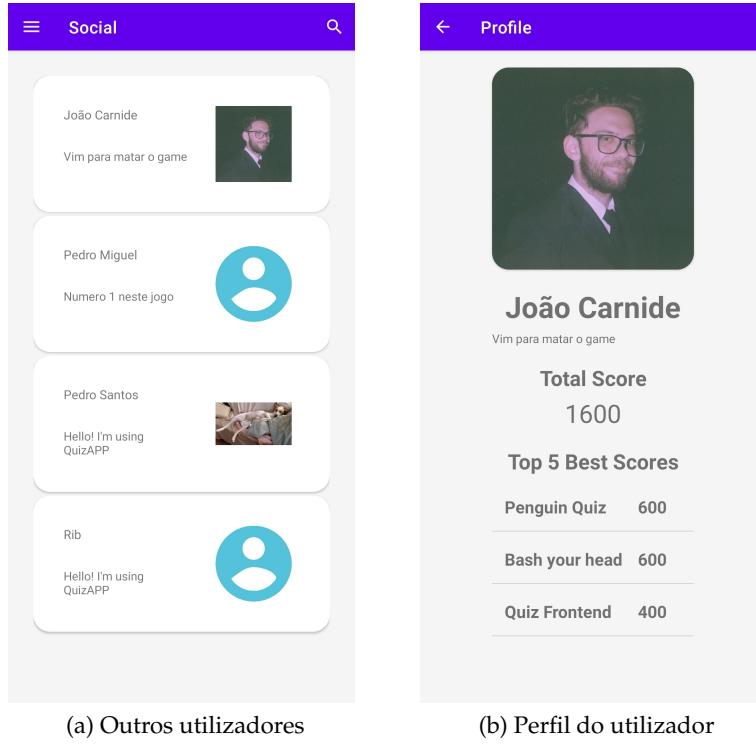
Selecione o *drawer* e escolha a opção “Recent Activity” para visualizar as atividades recentes dos utilizadores da aplicação no que diz respeito à realização de *quizzes*.

Recent Activity	
	28 min ago
Filipe Lopes just took the Penguin Quiz and scored 700 points	29 min ago
Filipe Lopes just took the Whale Quiz and scored 200 points	29 min ago
Filipe Lopes just took the Bash your head and scored 400 points	29 min ago
Filipe Lopes just took the Quiz Frontend and scored 600 points	33 min ago
Filipe Lopes just took the Yoda Quiz and scored 350 points	20 hours ago
Pedro Santos just took the Quiz Frontend and scored 950 points	20 hours ago
Pedro Santos just took the Quiz ReMix and scored 0 points	20 hours ago
Rib just took the Quiz Frontend and scored 1000 points	20 hours ago
João Carnide just took the Bash your head and scored 600 points	20 hours ago
João Carnide just took the Quiz Frontend and scored 400 points	20 hours ago
João Carnide just took the Penguin Quiz and scored 600 points	

Figura 5.13: Ecrã de atividades recentes na aplicação

## 5.6 Social

Selecione o *drawer* e escolha a opção “Social”. Aqui poderá ver todos os utilizadores da aplicação. Selecione um e visualize o perfil deste, que será semelhante ao perfil do utilizador com foto de perfil, nome, biografia, pontuação total e top 5 melhores pontuações em *quizzes*.



(a) Outros utilizadores

(b) Perfil do utilizador

Figura 5.14: Ecrãs de visualização do perfil de outros utilizadores

## 5.7 Leaderboard

Finalmente, no *drawer* selecione a opção “Leaderboard” para visualizar o *ranking* global de todos os utilizadores da aplicação (soma de todas pontuações em todos os *quizzes* realizados).

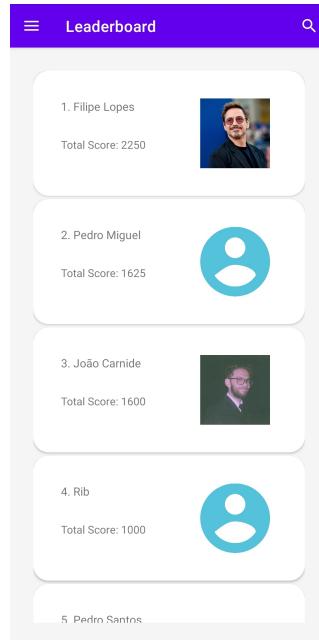


Figura 5.15: Ecrã do *leaderboard* da aplicação

# Capítulo 6

## Conclusão

Neste trabalho foram aplicadas as técnicas desenvolvidas ao longo do semestre na disciplina de Computação Móvel. Entre as mais básicas estão o uso adequado de *layouts*, atividades e fragmentos, bem como o recurso a *view models* para armazenamento de objetos *LiveData*. Quanto ao armazenamento de dados permanentes, estudaram-se bases de dados *SQLite*, embora tenhamos optado por uma integração com os serviços da *Firebase* neste projeto (*NoSQL*). Finalmente, para o chat optámos por usar os serviços da *Firebase*, concretamente a *Realtime Database*, em vez do tradicional *MQTT* estudado nas aulas.

Ao nível da engenharia de *software*, o projeto dividiu-se em *sprints*. A cada *sprint* era atribuído um dado número de tarefas a completar. A primeira fase do trabalho foi dedicada à elaboração dos requisitos, arquitetura e protótipos de alta fidelidade. Posteriormente, a solução foi implementada. Os artefactos produzidos na fase de requisitos serviram como guia para a implementação e recorremos a ferramentas como o *Trello* e o *Git* para gerir e organizar o trabalho. Um documento com os *bugs* encontrados e resolvidos foi sendo atualizado, por forma a deixar a aplicação pronta a ser usada pelo utilizador final. O último estágio deste projeto correspondeu à reunião de toda a documentação produzida ao longo do trabalho, com vista à elaboração deste relatório.