

\_registar.xml        \_login.xml        \_main.xml    \_area\_pessoal.xmll  
\_MenuQuestionarios.xml    \_QuestionarioMatinal.xml    \_teste\_tremor.xml  
\_teste\_tremor.xml

# Universidade da Beira Interior

## Departamento de Informática



Departamento de  
Informática

**NºVF01 - 2024: *Aplicação móvel para avaliação de múltiplos sintomas na Doença de Parkinson.***

Elaborado por:

**Fábio Augusto Moreira Rodrigues**  
Informática Web

Orientador:

**Professora Virginie dos Santos Felizardo**

14 de julho de 2025



# ***Dedicatória***

Ao meu Avô.



# ***Agradecimentos***

A todas as pessoas que conheci através da praxe Ubiana, um obrigado.

À Maria Inês, ao Gonçalo Roqueiro e ao João Lindeza, um enorme obrigado, por me mostrarem o que é ser da Beira Interior.

Carlos Filipe Pinto Barros Robalinho, de colega de escola, se tornou num amigo para vida, que se tornou num colega de Curso e Padrinho de Praxe, um gigantesco agradecimento, os dois primeiros anos de Universidade foram os melhores tanto a nível académico como a nível extracurricular, devido a tudo que ele me ensinou, e a paciência que sempre teve, um agradecimento do fundo do coração.

Um especial e grande agradecimento à Professora Virginie dos Santos Felizardo, pela compreensão e disponibilidade que teve ao longo dos últimos meses no apoio à realização deste Projeto.

Como não pode deixar de ser, um obrigado à minha Mãe, por ser a minha inspiração e acreditar que um dia eu poderia realizar este sonho de poder ser Licenciado.



# Conteúdo

<b>Conteúdo</b>	<b>v</b>
<b>Lista de Figuras</b>	<b>ix</b>
<b>Lista de Tabelas</b>	<b>xi</b>
<b>1 Introdução</b>	<b>1</b>
1.1 Âmbito . . . . .	1
1.2 Enquadramento . . . . .	2
1.3 Motivação . . . . .	2
1.4 Problema e Objetivo . . . . .	3
1.5 Abordagem . . . . .	3
1.6 Organização do Documento . . . . .	4
<b>2 Estado da Arte</b>	<b>5</b>
2.1 Introdução . . . . .	5
2.2 Estado da Tecnologia . . . . .	6
2.2.1 <i>Android™Platform</i> . . . . .	6
2.2.2 Aplicações <i>Android</i> . . . . .	7
2.2.3 Interface de Utilizador . . . . .	7
2.2.4 Interfaces de Utilizador Orientadas por Eventos . . . . .	7
2.2.5 <i>Model-View-Controller</i> . . . . .	8
2.2.6 Sensores em Dispositivos Móveis . . . . .	9
2.3 Estratégia de Pesquisa . . . . .	9
2.4 Resultados . . . . .	10
2.5 Análise de Funcionalidades . . . . .	12
2.6 Proposta . . . . .	13
2.7 Conclusão . . . . .	14
<b>3 Engenharia de Software</b>	<b>15</b>
3.1 Introdução . . . . .	15
3.2 Análise de Requisitos . . . . .	15
3.2.1 Requisitos Funcionais . . . . .	16



3.2.2	Requisitos Não Funcionais . . . . .	19
3.3	Diagramas . . . . .	21
3.3.1	Casos de Uso . . . . .	21
3.3.2	Ilustração da Aplicação . . . . .	22
3.3.3	Diagrama de Caso de Uso - Login/Registo . . . . .	23
3.3.4	Diagrama de Caso de Uso Questinario . . . . .	24
3.3.5	Diagrama Caso de Uso Teste Tremor . . . . .	25
3.4	Diagramas de Sequência . . . . .	26
3.4.1	Diagrama de Sequência Novo Utilizador . . . . .	26
3.4.2	Diagrama de Sequência Questionário . . . . .	28
3.4.3	Diagrama de Sequencia Teste . . . . .	29
3.5	Tecnologias Importantes . . . . .	30
3.6	Conclusões . . . . .	32
<b>4</b>	<b>Desenvolvimento</b>	<b>33</b>
4.1	Introdução . . . . .	33
4.2	Dependências . . . . .	34
4.3	Detalhes de Implementação . . . . .	35
4.3.1	Atividade Launcher . . . . .	35
4.3.2	Atividade Registrar . . . . .	36
4.3.3	Atividade Login . . . . .	40
4.3.4	Atividade Menu Principal . . . . .	42
4.3.5	Atividade Área Pessoal . . . . .	43
4.3.6	Menu Questionários . . . . .	45
4.3.7	Menu Questionários . . . . .	46
4.3.8	Menu Teste de Tremores . . . . .	48
4.3.9	Teste de Tremores Mão Direita . . . . .	49
4.4	Conclusões . . . . .	51
<b>5</b>	<b>Testes e Validação</b>	<b>53</b>
5.1	Introdução . . . . .	53
5.2	Especificação e Execução dos Testes . . . . .	53
5.2.1	Teste Sistema <i>Login</i> e de Registo . . . . .	53
5.2.2	Teste do Menu Questionários . . . . .	58
5.2.3	Teste do Menu Questionários . . . . .	59
5.3	Conclusões . . . . .	60
<b>6</b>	<b>Conclusões e Trabalho Futuro</b>	<b>61</b>
6.1	Conclusões Principais . . . . .	61
6.2	Trabalho Futuro . . . . .	62

<b>CONTEÚDO</b>	<b>vii</b>
<b>Bibliografía</b>	<b>63</b>



## ***Lista de Figuras***

3.1	Ilustração da Aplicação . . . . .	22
3.2	Diagrama Caso de Uso Login e Registrar . . . . .	23
3.3	Diagrama de Caso de Uso Questinario . . . . .	24
3.4	Diagrama Caso de Uso Teste Tremor . . . . .	25
3.5	Diagrama de Sequência Novo Utilizador . . . . .	27
3.6	Diagrama de Sequência Questionário . . . . .	28
3.7	Diagrama de Sequencia Teste . . . . .	29
3.8	Ciclo de Vida de uma Atividade . . . . .	32
4.1	activity_registar.xml . . . . .	35
4.2	activity_registar.xml . . . . .	36
4.3	activity_login.xml . . . . .	40
4.4	activity_main.xml . . . . .	42
4.5	activity_area_pessoal.xml . . . . .	43
4.6	activity_MenuQuestionarios.xml . . . . .	45
4.7	activity_QuestionarioMatinal.xml . . . . .	46
4.8	activity_teste_tremor.xml . . . . .	48
4.9	activity_teste_tremor.xml . . . . .	49
5.1	Idade tem de ser um Valor Numérico . . . . .	54
5.2	Email tem de estar num formato válido . . . . .	55
5.3	A palavra Passe pequena . . . . .	56
5.4	Autenticação no FireBase . . . . .	57
5.5	Criação da Árvore no RealTime DataBase . . . . .	57
5.6	Responder a todas as Perguntas . . . . .	58
5.7	Editar Informações Pessoais . . . . .	59



## ***Lista de Tabelas***

2.1	Tabela de Aplicações Móveis para a pesquisa <i>Parkinson</i> . . . . .	10
2.2	Tabela de Aplicações Móveis para a pesquisa <i>Tremors Meter</i> . . . . .	11
3.1	Requisitos Funcionais da Aplicação . . . . .	16
3.2	Requisitos Não Funcionais da Aplicação - Usabilidade . . . . .	19
3.3	Requisitos Não Funcionais da Aplicação - Compatibilidade de Plataforma . . . . .	20
3.4	Requisitos Não Funcionais da Aplicação - Segurança e Privacidade	20



# ***Acrónimos***

<b>UBI</b>	Universidade da Beira Interior
<b>UC</b>	Unidade Curricular
<b>P1</b>	Programação I
<b>PDM</b>	Programação de Dispositivos Móveis
<b>POO</b>	Programação Orientada a Objetos
<b>IDE</b>	<i>Integrated Development Environment</i>
<b>SDK</b>	<i>Software Development Kit</i>
<b>API</b>	Application Programming Interface
<b>SO</b>	Sistema Operativo
<b>MVC</b>	Model-View-Controller
<b>XML</b>	Extensible Markup Language
<b>GPS</b>	Global Positioning Sensor
<b>JVM</b>	Java Virtual Machine
<b>IU</b>	Interface de Utilizador
<b>UID</b>	User Id





## Capítulo

# 1

## Introdução

### 1.1 Âmbito

O seguinte documento, trata-se de um Relatório Técnico desenvolvido pelo aluno Fábio Augusto Moreira Rodrigues, número 41923, atualmente matriculado no curso de Informática Web da Universidade da Beira Interior (UBI), no âmbito da Unidade Curricular (UC) de Projeto.

O objetivo deste Projeto, será a construção de uma aplicação móvel sobre a doença de *Parkinson*, em que seja possível a monitorização e o diagnóstico da doença. A doença de *Parkinson* afeta cerca de quatro milhões de pessoas mundialmente, com o contínuo aumento da esperança de vida e o envelhecimento da população, este número poderá dobrar até 2040. Serão usados o acelerómetro, giroscópio e o microfone do *smartphone* para a recolha de dados, dos tremores e alterações da voz, respetivamente.

Com o profundo desenvolvimento tecnológico que a sociedade tem vindo a sofrer, nomeadamente nas últimas duas décadas, especialmente na área das aplicações móveis, é da máxima importância usar ferramentas como *smartphones*, como instrumento de apoio à saúde dos utilizadores. Já que estes fazem parte do cotidiano de grande parte das pessoas atualmente.

Para a realização deste trabalho foi tido em conta o estudo realizado, anteriormente, por dois colegas envolvendo a doença de *Parkinson* (Vitaliy Tsymborsky e Beatriz Dos Reis Lopo Ferreira). Fica aqui um especial agradecimento aos dois colegas.

## 1.2 Enquadramento

O desenvolvimento tecnológico tem permitido ao longo dos últimos anos um admirável avanço na construção de *smartphones*. No início do século XXI os *smartphones*, eram chamados de *mobile telephones*, os telemóveis, ou telefones para a versão não portátil.

As principais diferenças entre eles são: no caso dos *smartphones* o teclado é virtual e enquanto nos *mobile telephones* é físico; os *smartphones* têm a possibilidade de fazer *downloads* de aplicações; e a maior diferença entre ambos é o facto de os *smartphones* terem sistema operativo, entre outras diferenças.

Estima-se que haja quatro mil milhões de *smartphones* atualmente no planeta, sendo este um mercado muito concorrido onde os aspectos da inovação e da novidade são os que despertam o interesse dos utilizadores para adquirir estes dispositivos.

Com a evolução das ferramentas, os *smartphones* podem ser utilizados em vários contextos e com funções variadas. Neste projeto serão usados o giroscópio, acelerómetro e microfone, os dois primeiros instrumentos referidos permitem verificar oscilações e detectar movimentações, enquanto o microfone permite a captura de som.

## 1.3 Motivação

A doença de *Parkinson* [1] resulta da redução dos níveis de dopamina, que funciona como mensageiro químico cerebral que comanda os movimentos. Quando os seus níveis reduzem, ocorre a morte das células cerebrais que a produzem. A sua prevalência aumenta com a idade, sendo rara antes dos 50 anos, e é mais comum nos homens do que nas mulheres.

Contudo, em 5% dos casos, surge antes dos 40 anos. Uma vez que a dopamina controla a atividade muscular, os sintomas relacionam-se essencialmente com os movimentos. Para lá dos tremores, rigidez e lentidão, existem outras manifestações que se traduzem no sono, no pensamento, na fala e no estado de espírito dos pacientes.

Aliando o tipo de sintomas que a doença provoca com o uso de *smartphones*, é possível a criação de uma aplicação que ajuda os pacientes que sofrem desta doença ou que procuram ter um diagnóstico. Num sub-capítulo adiante, 2.5, será discutido que tipo de aplicações existem no contexto de apoio aos doentes de *Parkinson* e devida conclusão das soluções encontradas no mercado.

## 1.4 Problema e Objetivo

Baseado no contexto do projeto o seguinte problema pode ser identificado: a partir do fundamento que é necessário recolher dados de tremores usando o giroscópio e o acelerômetro de um determinado *smartphone* é obrigatório o uso dos sensores do *smartphones*. Com isto é preciso identificar que tipos de sensores vão ser usados e as respectivas funções que serão criadas para o tratamento dos tremores.

O principal objetivo do projeto será o desenvolvimento de uma Aplicação que registre tremores, através do giroscópio e acelerômetro de um *smartphone* e registre gravações de som, através do microfone. Para a avaliação destes dados é necessário criar algoritmos que permitam a sua análise e estudo. Estes algoritmos podem ser considerados um objetivo secundário do projeto, contudo algo essencial ao funcionamento da aplicação.

## 1.5 Abordagem

1. Inicialmente a escolha recaiu sobre a *framework Flutter*. Contudo houve uma mudança em que tipo de *framework* o projeto foi desenvolvido.
2. A escolha portanto foi o *android studio*, um *Integrated Development Environment* (IDE), onde contém ferramentas que permitem aos programadores de *software* projetar, criar, executar e testar *softwares*.
3. O *android studio* foi escolhido, baseado na minha experiência prévia, com o IDE em questão e com a linguagem de programação *Java*. Para esta experiência foi essencial o conhecimento da UC de Programação I (P1), da UC de Programação Orientada a Objetos (POO) e da UC de Programação de Dispositivos Móveis (PDM).
4. Para a Base de Dados a escolha foi o *FireBase*. Baseado também na experiência prévia que tenho com esta ferramenta. Num capítulo 3.5 adiante será abordado com mais detalhe o tipo de tecnologias usadas neste projeto e uma análise mais profunda às tecnologias.
5. Posteriormente irá ser feito o estudo do estado da arte, onde existe a pesquisa de aplicações similares ao que se pretende elaborar neste projeto.
6. Mais tarde é indispensável ser feito todo o tipo de diagramas da engenharia de *software* e uma lista de requisitos.
7. Por fim desenvolver o projeto no IDE e fazer os devidos testes.

## 1.6 Organização do Documento

De modo a refletir o trabalho que foi feito, este documento encontra-se estruturado da seguinte forma:

1. O primeiro capítulo – **Introdução** – apresenta o âmbito projeto, o enquadramento para o mesmo, a motivação para a sua escolha, os seus problemas e objetivos, a abordagem que foi realizada e por fim a respetiva organização do documento;
2. O segundo capítulo – **Estado da Arte** – descreve todas as aplicações já existentes no mercado sobre o tema do projeto, como foi efetuada a respetiva pesquisa, uma proposta de aplicação e quais foram as conclusões da pesquisa;
3. O terceiro capítulo – **Engenharia de Software** – descreve os requisitos funcionais e não funcionais, os respetivos casos de uso, diagrama de sequência;
4. O quarto capítulo – **Desenvolvimento** – apresenta os passos no desenvolvimento do projeto;
5. O quinto capítulo – **Testes** – descreve como alguns testes da aplicação são feitos;
6. O sexto capítulo – **Conclusões e Trabalhos Futuros** – descreve as conclusões finais sobre o projeto e os trabalhos futuros que podem ser feitos para complementar, melhorar e expandir todo o trabalho até ao momento.

## **Capítulo**

# 2

## ***Estado da Arte***

### **2.1 Introdução**

Com base no problema exposto na secção 1.4 Problema e Objetivo, no final do capítulo 1 Introdução, numa primeira fase o capítulo 2 Estado da Arte irá abordar o 2.2 Estado da Tecnologia que permite a resolução do problema apontado.

Para desenvolver este projeto, foi essencial realizar uma análise detalhada das aplicações existentes dedicadas à recolha de dados de tremores e voz em indivíduos diagnosticados com a doença de *Parkinson*.

Este capítulo visa também esmiuçar o processo envolvido na investigação das aplicações disponíveis, no mercado, abordando desde a 2.3 metodologia de pesquisa adotada até a 2.5 avaliação do conteúdo, design, 2.6 uma proposta baseada no âmbito do próprio projeto em função da avaliação feita da pesquisa com os resultados obtidos e por fim a 2.7 conclusão.

## 2.2 Estado da Tecnologia

Este sub-capítulo tem como objetivo explicar os principais elementos na construção de Aplicações Móveis como as tecnologias existentes na respetiva área.

[2] Considera-se um dispositivo móvel àquele que tem as seguintes características:

- **Computação** - capacidade de Processar e Armazenar Dados;
- **Interatividade com o Utilizador** - possui instrumentos de saída e entrada de dados;
- **Portabilidade**;
- **Dimensão e Peso**;
- **Comunicação sem Fios** - possuem tecnologias que permitem a comunicação com dispositivos semelhantes.

Em seguida vai ser apresentada uma lista das principais tecnologias à cerca da Área de Dispositivos Móveis.

Com isto é necessário fazer um apontamento importante sobre o conteúdo deste capítulo, os trechos de texto usados nas tecnologias são da autoria do Professor Paulo Fazendeiro [3], docente da UC de PDM. Estes trechos de texto foram usados durante as aulas da UC de PDM fazendo parte da "sebenta" da UC.

### 2.2.1 *Android™ Platform*

[4] Ultimamente tem-se assistido, à utilização do *Android™* em dispositivos diversos, nomeadamente *smart TVs*, e *diversos wearables*, entre outros.

A popularidade da plataforma, o facto de ter o código aberto e ser relativamente simples desenvolver aplicações para a mesma, tem ditado uma evolução fora do comum, tornando-a hoje num sistema bastante completo e complexo, no sentido de albergar já um vasto conjunto de componentes.

De uma maneira geral, pode dizer-se que a plataforma *Android™* é composta por:

1. uma pilha de *software*, com várias camadas, desenhada para permitir a construção e execução de aplicações móveis;
2. um *kit* de desenvolvimento de *software* (da designação *Software Development Kit* (SDK));
3. e uma extensa documentação.

### 2.2.2 Aplicações *Android*

[5] As aplicações *Android*<sup>TM</sup> são constituídas por vários componentes, que colaboram entre si, e que a plataforma despoleta e corre quando necessário. Cada um desses componentes tem o seu próprio objetivo e, portanto, a sua própria *Application Programming Interface (API)*.

Dado que, em *Android*<sup>TM</sup>, as aplicações nativas correm numa máquina virtual *Java*, cada um destes componentes são também implementados em *Java*. Os 4 blocos fundamentais sobre os quais as aplicações *Android*<sup>TM</sup> são construídas são:

1. Atividades (Activities);
2. Serviços (Services);
3. Fornecedor de Conteúdos (ContentProviders);
4. Recetores Difusão (Broadcastreceiver).

### 2.2.3 Interface de Utilizador

[6] Um dos mais importantes componentes de aplicações móveis, pelo menos daquelas que interagem com o utilizador, é a interface do utilizador.

*"Uma interface do utilizador é um programa de computador, ou parte dele, que permite que o utilizador comunique com esse computador ou com outras partes do programa."*

### 2.2.4 Interfaces de Utilizador Orientadas por Eventos

[6] Os desenvolvimentos tecnológicos que permitiram a produção em massa e adoção de ecrãs táteis constituem pontos chave na história das interfaces de utilizador, contribuindo ainda mais para a solidificação do modelo de interação orientado por eventos. Este modelo, utilizado na maior parte das aplicações móveis modernas, permite disponibilizar um conjunto maior de funcionalidades em simultâneo e de forma intuitiva, correspondendo ainda a uma simplificação na estrutura do programa. Neste caso, é o programa que espera pelo utilizador (controlo externo), nomeadamente que este lhe forneça um dos possíveis inputs. A manipulação destas interfaces pode ser feita de forma direta, i.e., para cada ou alguns dos objetos mostrados ao utilizador. Tecnicamente, tal é conseguido através de mecanismos de comunicação entre objetos interativos e o sistema de entrada e saída (e.g., um ecrã tátil), concretizada por eventos:



A rotina principal de captação de eventos é da responsabilidade do Sistema Operativo (SO), que a executa constantemente e colecciona os eventos numa pilha *First In First Out*, assegurando que estes são tratados pela ordem que chegaram.

### 2.2.5 *Model-View-Controller*

[6] De um modo simplista e geral, a arquitetura Modelo-Visão-Controlador (da designação inglesa *Model-View-Controller (MVC)*) é um modelo de arquitetura para desenvolvimento de software que separa a lógica da aplicação, a representação da informação e a interação do utilizador através da definição de 3 componentes: o modelo, o controlador e a visão .

1. **Model** - Numa implementação de uma aplicação móvel, pode-se pensar no Modelo como o conjunto de classes que concretizam os seus objetivos principais. Fazendo uso do que é discutido antes nesta aula, é no Modelo que conceitualmente devem estar as implementações dos métodos que, de forma concisa, segura e uniformizada alteram os dados/informação interna da aplicação. De uma forma geral, é o modelo que contém as classes e métodos para aceder e manipular bases de dados ou memória persistente, bem como toda a informação que pode ser exibida ao utilizador através de uma visão. No MVC, um utilizador nunca interage diretamente com o Modelo.
2. **View** - No MVC, uma Visão consiste numa possível representação dos dados da aplicação. É o componente Visão que se definem as Interfaces de Utilizador. Na plataforma *Android*<sup>TM</sup>, por exemplo, é permitido que a interface do utilizador seja definida em ficheiros Extensible Markup Language (XML) completamente independentes do código da aplicação escrito em *Java*. Estes ficheiros estão normalmente na diretoria `app/src/main/res/` (de *resources*) e o seu nome termina tipicamente com a palavra *layout*. O código da aplicação (que concretiza o Modelo e o Controlador) encontra-se na pasta `app/src/main/java/`.
3. **Controller** - O Controlador é o intermediário entre o Modelo, as Visões e o utilizador. São procedimentos do Controlador que atualizam os objetos da Visão e que despoletam mudanças no estado da aplicação através do envio de instruções para o Modelo.

### 2.2.6 Sensores em Dispositivos Móveis

[7] A maior parte dos dispositivos móveis de hoje integram sensores que medem o movimento, a localização, a orientação e vários parâmetros ambientais, como a temperatura ou a humidade. A plataforma *Android*<sup>TM</sup> suporta três categorias principais de sensores, estruturadas da seguinte forma:

1. Sensores de Movimento, que incluem os sensores que medem forças de aceleração e rotacionais em três eixos. É nesta categoria que são incluídos os acelerômetros, giroscópios, sensores de gravidade e os rotacionais;
2. Sensores de Ambiente, que incluem os sensores que medem parâmetros de ambiente, como a temperatura, a pressão do ar, a humidade ou a iluminação. Aqui se incluem os barômetros, fotômetros e termômetros
3. Sensores de Posicionamento, que medem/identificam a posição física dos dispositivos, e que incluem o *Global Positioning Sensor (GPS)*, sensores de orientação e os magnetômetros.

Como seria de esperar, a plataforma *Android*<sup>TM</sup> permite um acesso bastante simplificado a todos os sensores disponíveis no dispositivo móvel a partir da *framework* de sensores *Android*<sup>TM</sup>.

## 2.3 Estratégia de Pesquisa

O estudo de aplicações que detetam tremores e avaliem utentes com a doença de *Parkinson* foi realizada no dia 15 de maio de 2024 na *Google Play Store*, dando importância apenas a aplicações gratuitas. Foram efetuadas três pesquisas: "*Parkinson*", "*Tremors Meter*" e "*Voice Analysis Parkinson*".

O critério de escolha das alternativas baseou-se na ordem de relevância das mesmas no momento da pesquisa. Após análise, foram identificadas quatro aplicações no contexto de "*Parkinson*" e apenas uma aplicação no contexto de "*Tremors Meter*". É importante salientar que, na segunda pesquisa, foram encontradas aplicações com vários fins, desde detetar terremotos até medidores de distância e decibéis. Em relação à pesquisa "*Voice Analysis Parkinson*", não foi encontrada nenhuma aplicação relevante no contexto sugerido.

O primeiro passo após as pesquisas mencionadas foi fazer o *download* das alternativas disponíveis, para realizar uma análise/avaliação das funcionalidades, como da Interface de Utilizador e da Experiência de Utilizador.

## 2.4 Resultados

A Tabela 2.1 apresenta as quatro aplicações identificadas com base nos critérios da primeira pesquisa ("*Parkinson*"), seguidas de uma descrição com base na informação resultante dos testes realizados nas aplicações.

Tabela 2.1: Tabela de Aplicações Móveis para a pesquisa *Parkinson*

Nome	Autor	Downloads	Opiniões de Utilizadores	Descrição
Swiss Parkinson	SkyScraper Software GmbH	1000+	4.2 estrelas - 16 avaliações	App de Parkinson suíço para pacientes de Parkinson e seus parentes
Parkinson	CogniFit Inc	5000+	4.6 estrelas - 49 avaliações	Estudos Sobre Parkinson
Parkinson's ON	Play Protect	500	Sem Avaliações	Assistente de Parkinson
Parkinson Symptom Tracking	Parkinson CPR	1000+	3.4 estrelas - 9 avaliações	PRO-PD

A Tabela 2.2 apresenta a aplicação identificada com base nos critérios da segunda pesquisa ("*Tremors Meter*"), seguida de uma descrição com base na informação resultante dos testes realizados na aplicação.

Tabela 2.2: Tabela de Aplicações Móveis para a pesquisa *Tremors Meter*

Nome	Autor	Downloads	Opiniões de Utilizadores	Descrição
Steady Hands	Vilimed	1000+	4.3 estrelas - 48 avaliações	Ferramenta de medição de estabilidade e tremor das mãos

## 2.5 Análise de Funcionalidades

Neste sub-capítulo irão ser descritas de uma forma resumida as funcionalidades de cada aplicação.

- **Swiss Parkinson** - A aplicação não possui sistema de *login*. É uma aplicação com sistema multilíngue, contendo tradução em 4 idiomas. O utilizador tem a possibilidade de escolher entre 4 perfis: Paciente, Membro da família, Terapeuta/Enfermeiro e, por fim, Médico. Após escolher o perfil, neste caso, o de Paciente, a aplicação faz um breve questionário sobre o estado do utilizador.

Entre as questões, estão incluídos temas como mobilidade, saúde mental, sintomas cognitivos e sono. Após o questionário, a aplicação redireciona o utilizador para o menu "*Home*". O menu é composto por Programa de Treino, onde o utilizador pode definir um programa de treino, Medicação, Diário, Informação sobre *Parkinson*, Registo de Quedas e Exercícios. A aplicação está dividida em "*Home*" menu e "*Overview*" menu, onde o utilizador pode ver gráficos das atividades e exercícios, o progresso da doença, medicação, e tem acesso ao protocolo de quedas.

Em suma, é uma aplicação que faz o acompanhamento de um Paciente, no entanto, não regista tremores nem variações na voz. *Design* simples e intuitivo que facilita a execução de todas as funcionalidades.

- **Parkinson** - A aplicação possui sistema de *login/register*, onde o utilizador pode registar-se com o seu próprio *email* ou conectar-se com a sua conta *Google* ou conta *Facebook*. No entanto, a única funcionalidade desta aplicação é realizar um questionário com cerca de 20 perguntas acerca da doença de *Parkinson*.
- **Parkinson's ON** - A aplicação possui sistema de *login/register*, onde o utilizador pode registar-se com o seu próprio *email* ou conectar-se com a sua conta *Google*, conta *Facebook* ou conta *Apple*. Após o registo, a aplicação apresenta um pequeno questionário com perguntas sobre o utilizador, incluindo nome, género, data de nascimento e o ano em que foi feito o diagnóstico da doença.

A principal função desta aplicação é acompanhar o Paciente, lembrando-o da toma de medicação, permitindo o registo de sintomas e atividades. Além disso, disponibiliza informação sobre a doença de *Parkinson* e inclui a área "*Explore*", onde o Paciente pode ouvir *Podcasts*, meditar e praticar *Yoga*. Com um *design* simples e intuitivo, facilita a utilização de todas as funcionalidades.

- ***Parkinson Symptom Tracking*** - A aplicação possui sistema de *login/register*, onde o utilizador pode registar-se utilizando o seu *email*. A aplicação acompanha o Paciente, que deve responder a perguntas relacionadas com os sintomas que tem sentido ao longo dos últimos 7 dias.
- ***Steady Hands*** - A aplicação possui sistema de *login/register*, onde o utilizador pode registar-se utilizando o email ou conectando-se com a conta *Google*.

De todas as aplicações testadas, esta é a única que se destaca no contexto de registar tremores. Permite registar tremores utilizando diversas atividades, como desenhar, levantar um copo de água e pressionar o ecrã. O registo de tremores é feito em ambas as mãos. No final, os testes são guardados, possibilitando acompanhar a evolução dos tremores ao longo do tempo. Com um *design* simples e intuitivo, facilita a execução de todas as funcionalidades.

## 2.6 Proposta

A análise do estado da arte forneceu-me os testes necessários, no sentido de estabelecer as funcionalidades principais para a aplicação móvel de apoio a doentes de *Parkinson*. Após o estudo das 5 aplicações constatou-se que nenhuma das que foi testada reunia todas as funcionalidades seguintes:

- Sistema de *Login* e Registo;
- Testes de tremor, usando o acelerômetro e giroscópio do *smartphone*;
- Testes de voz, usando o microfone do *smartphone*;
- Questionários diários, sobre os efeitos que os pacientes vêm sentindo ao longo do dia;
- Questionários com perguntas gerais sobre a doença de *Parkinson*;

- Implimentação do algoritmo para tratar os dados dos tremores e posteriormente verificar a evolução da doença;
- Implimentação do algoritmo para tratar os dados da voz e posteriormente verificar a evolução da doença.

A minha proposta para a aplicação de monitorização e diagnóstico da doença de *Parkinson* terá de ser capaz de reunir todas as funcionalidades anteriormente enunciados neste subcapítulo.

O objetivo deste projeto é desenvolver uma aplicação capaz de registar a evolução da doença de *Parkinson* num utilizador e, se possível, diagnosticar a própria doença. Para isso, serão usados vários métodos, tanto ativos, como testes manipulando dados vindos do giroscópio, acelerômetro e do microfone do *smartphone*, como métodos passivos, por exemplo, questionários.

## 2.7 Conclusão

Durante a preparação deste estudo, foi essencial realizar uma análise exaustiva do estado da arte. Este procedimento permitiu-me identificar o que já havia sido desenvolvido e, assim, procurar integrar funcionalidades inovadoras e específicas para a avaliação de tremores em indivíduos com a doença de *Parkinson*. Além disso, conseguimos extrair algumas ideias relevantes para a criação da interface de utilizador e das funcionalidades.

É importante destacar que, durante esta pesquisa, não encontrei uma funcionalidade específica para recolha de voz pelo microfone do *smartphone*. Essa funcionalidade teria a finalidade de permitir que o paciente com *Parkinson* lesse um texto, enquanto a aplicação avalia eventuais variações na sua voz. A ausência dessa tecnologia levou-me a considerar a possibilidade de integrá-la no projeto, uma vez que poderia contribuir significativamente para a monitorização da doença.

Em síntese, os principais desafios deste projeto incluem o desenvolvimento de uma aplicação que seja acessível e apta para uma série de pessoas com a doença de *Parkinson*, abrangendo desde jovens até idosos. Além disso, será necessário criar um algoritmo capaz de recolher, processar e interpretar os dados provenientes do acelerômetro, giroscópio e microfone do *smartphone* do paciente.

## **Capítulo**

# 3

## ***Engenharia de Software***

### **3.1 Introdução**

Neste capítulo, serão detalhadas todas as funcionalidades e aspectos da aplicação, proporcionando ao utilizador uma compreensão abrangente de como utilizar eficazmente a mesma.

Este capítulo está dividido em três secções distintas. Na primeira secção, serão delineados os 3.2.1 requisitos funcionais e 3.2.2 não funcionais da aplicação, estabelecendo os critérios necessários para o seu desenvolvimento e funcionamento adequados.

Na segunda secção, serão descritos os possíveis cenários de utilização, incluindo 3.3.1 casos de uso e outros diagramas, visando a oferecer uma visão clara das interações entre o utilizador e a aplicação, bem como entre os diferentes componentes do sistema.

Para a realização deste trabalho foram implementadas várias 3.5 tecnologias, por esse motivo vai-se proceder à apresentação das mesmas na terceira secção deste capítulo.

### **3.2 Análise de Requisitos**

A etapa de análise de requisitos é crucial no desenvolvimento de *software*, pois reúne as informações essenciais para que o utilizador possa resolver problemas e alcançar os seus objetivos. Esta fase de análise é dividida em duas partes distintas: requisitos de funcionalidade e requisitos não funcionais.



### 3.2.1 Requisitos Funcionais

Tabela 3.1: Requisitos Funcionais da Aplicação

ID Re- quisito	Nome Requisito	Descrição do Requisito
RF01	Registo de Utilizador	A aplicação deve permitir o registo de novos utilizadores, com um email e uma palavra-passe.
RF02	Registo de Utilizador Conta Google	A aplicação deve permitir o registo de novos utilizadores, com conta Google.
RF03	Palavra-Passe com Menos de oito caracteres	A aplicação não deve permitir o registo de um utilizador usando uma palavra-passe com menos de oito caracteres
RF04	Nome em Falta	A aplicação não deve permitir o registo de um utilizador que não inseriu o primeiro e último nome.
RF05	Email em Falta	A aplicação não deve permitir o registo de um utilizador que não inseriu o email.
RF06	Número de Utente em Falta	A aplicação não deve permitir o registo de um utilizador que não inseriu o número de utente
RF07	Campos em Branco	A aplicação não deve permitir o registo de um utilizador que deixe qualquer tipo de campo de registo em branco
RF08	Email e Número de Utente Único	A aplicação deve permitir o registo de um email/número de utente único por utilizador
RF09	Mudar Email/Palavra-Passe	A aplicação deve permitir a mudança de email e palavra-passe
RF10	Recuperar Palavra-Passe	A aplicação deve permitir a recuperação da palavra-passe
RF11	Gravar dados de Login	A aplicação deve guardar os dados do login do utilizador para manter a sessão iniciada
RF12	Logim de Utilizador	A aplicação deve permitir o login dos utilizadores
RF13	LogOut de Utilizador	A aplicação deve permitir o logout dos utilizadores
RF14	Últimos Registos	A aplicação deve permitir o acesso aos últimos registos de tremores do utilizador
RF15	Últimas Gravações	A aplicação deve permitir o acesso às últimas gravações de voz do utilizador

RF16	Últimos Questionários	A aplicação deve permitir o acesso aos últimos questionários de monitorização do utilizador
RF17	Realizar Registo de Tremores	A aplicação deve permitir a realização de novos registos de tremores
RF18	Realizar Qestionários	A aplicação deve permitir a realização de novos questionários de monitorização
RF19	Registo de Gravações	A aplicação deve permitir a realização de novos testes de gravação de voz
RF20	Permitir medir Tremores	A aplicação deve permitir a medição dos tremores
RF21	Detetar alterações na voz	A aplicação deve permitir a detetar alterações na voz
RF22	Aceder a Dados do Giroscópio e Acelerómetro	A aplicação deve permitir o acesso aos dados do giroscópio e acelerómetro do smartphone
RF23	Processar Dados do Giroscópio e Acelerómetro	A aplicação deve permitir o processamento dos dados do giroscópio e acelerómetro, para verificar padrões de tremores e calcular métricas relevantes para o diagnóstico da doença de Parkinson;
RF24	Aceder a Dados da voz	A aplicação deve permitir o acesso aos dados da captura de voz
RF25	Processar Dados da Voz	A aplicação deve permitir o processamento dos dados da captura de voz, para verificar padrões e calcular métricas relevantes para o diagnóstico da doença de Parkinson
RF26	FeedBack Diário	A aplicação deve transmitir um feedback diário acerca dos sintomas do utilizador
RF27	FeedBack Adaptativo	A aplicação deve transmitir um feedback adaptativo, isto é, fornecer sugestões personalizadas para melhorar a sua qualidade de vida
RF28	Enviar Notificações Tremores	A aplicação deve enviar notificações aos utilizadores, lembrete, para a realização do registo de tremores

RF29	Enviar Notificações Questionários	A aplicação deve enviar notificações aos utilizadores, lembrete, para a realização dos questionários de monitoramento
RF30	Enviar Notificações Voz	A aplicação deve enviar notificações aos utilizadores, lembrete, para a realização dos teste de voz
RF31	Suporte de Vários Idiomas	A aplicação deve permitir o suporte a vários idiomas
RF32	Armazenamento com Segurança	A aplicação deve permitir o armazenamento de dados dos utilizadores com segurança

### 3.2.2 Requisitos Não Funcionais

Tabela 3.2: Requisitos Não Funcionais da Aplicação - Usabilidade

ID Re- quisito	Nome Requisito	Descrição do Requisito
RNF01	Aceder ao Sistema	O sistema deve permitir que os utilizadores acedam ao sistema sem dificuldades.
RNF02	Vários utilizadores a Aceder ao Sistema	O sistema deve permitir que vários utilizadores acedam ao sistema ao mesmo tempo
RNF03	Interface Intuitiva	O sistema deve fornecer uma interface de utilizador intuitiva e fácil de usar para pacientes, profissionais de saúde, independentemente da sua experiência técnica.
RNF04	Aplicação Responsiva	O sistema deve fornecer uma aplicação responsiva e de alto desempenho.
RNF05	Arquitetura Escalável	O sistema deve fornecer uma aplicação com uma arquitetura facilmente escalável
RNF06	Código Estruturado	O código-fonte da aplicação deve ser bem estruturado e comentado, para facilitar a manutenção e alterações num futuro.
RNF07	Aumento Significativo de Utilizadores	O sistema deve ser capaz de lidar com um aumento significativo no número de utilizadores e volume de dados sem comprometer o desempenho ou a disponibilidade
RNF08	Experiência Fluida	O sistema deve fornecer tempos de resposta rápidos e consistentes para as operações mais comuns, garantindo uma experiência de usuário fluida e satisfatória

Tabela 3.3: Requisitos Não Funcionais da Aplicação - Compatibilidade de Plataforma

ID Requisito	Nome Requisito	Descrição do Requisito
RNF09	Compatibilidade	O sistema deve permitir uma compatibilidade com uma variedade de dispositivos móveis e sistemas operativos, incluindo iOS e Android, para garantir um amplo mercado

Tabela 3.4: Requisitos Não Funcionais da Aplicação - Segurança e Privacidade

ID Requisito	Nome Requisito	Descrição do Requisito
RNF10	Garantir Integridade e Confidencialidade	A aplicação deve garantir a integridade e confidencialidade dos dados dos pacientes de acordo com as regulamentações de dados.
RNF11	Dados do Utilizador são Armazenados de Forma Segura	O sistema deve garantir que todos os dados do utilizador são armazenados de forma segura, utilizando criptografia e medidas de segurança adequadas, para proteção contra acessos não autorizados

## 3.3 Diagramas

### 3.3.1 Casos de Uso

Os Casos de Uso tem como objetivo permitir que terceiros compreendam completamente todas as funcionalidades da aplicação. Por isso, é essencial realizar o estudo detalhado de cada funcionalidade.

Num diagrama de Caso de Uso o primeiro passo é escolher o contexto do diagrama, isto delimita o sistema. É representado por um retângulo que são todas as ações que a aplicação vai realizar mediante a interação com o utilizador e outros sistemas fora da aplicação, para isso tem-se de identificar os atores que interagem entre si.

O ator principal é o utilizador, representado do lado esquerdo do diagrama, que desencadeia as ações do sistema, o ator secundário é será o Firebase, representado do lado direito, que reage a estímulos do sistema. Neste caso o envio de dados, da autenticação, login, questionários e testes. Foram realizados quatro diagramas de Caso de Uso:

O 3.3.2 primeiro mostra a lógica da aplicação. A aplicação está dividida em seis partes. *Login*, Registo, Menu Questionários, Menu Testes e Histórico.

O utilizador pode, fazer Login, Registar-se, fazer questionários, testes, alterar as suas informações pessoais e visualizar o histórico de testes e questionários.

Os três diagramas seguintes mostram com mais detalhe outras funcionalidades da Aplicação. O sistema de 3.3.3 *Login* e Registo, o 3.3.4 Sistema de Questionários e o 3.3.5 Sistema de Testes

### 3.3.2 Ilustração da Aplicação

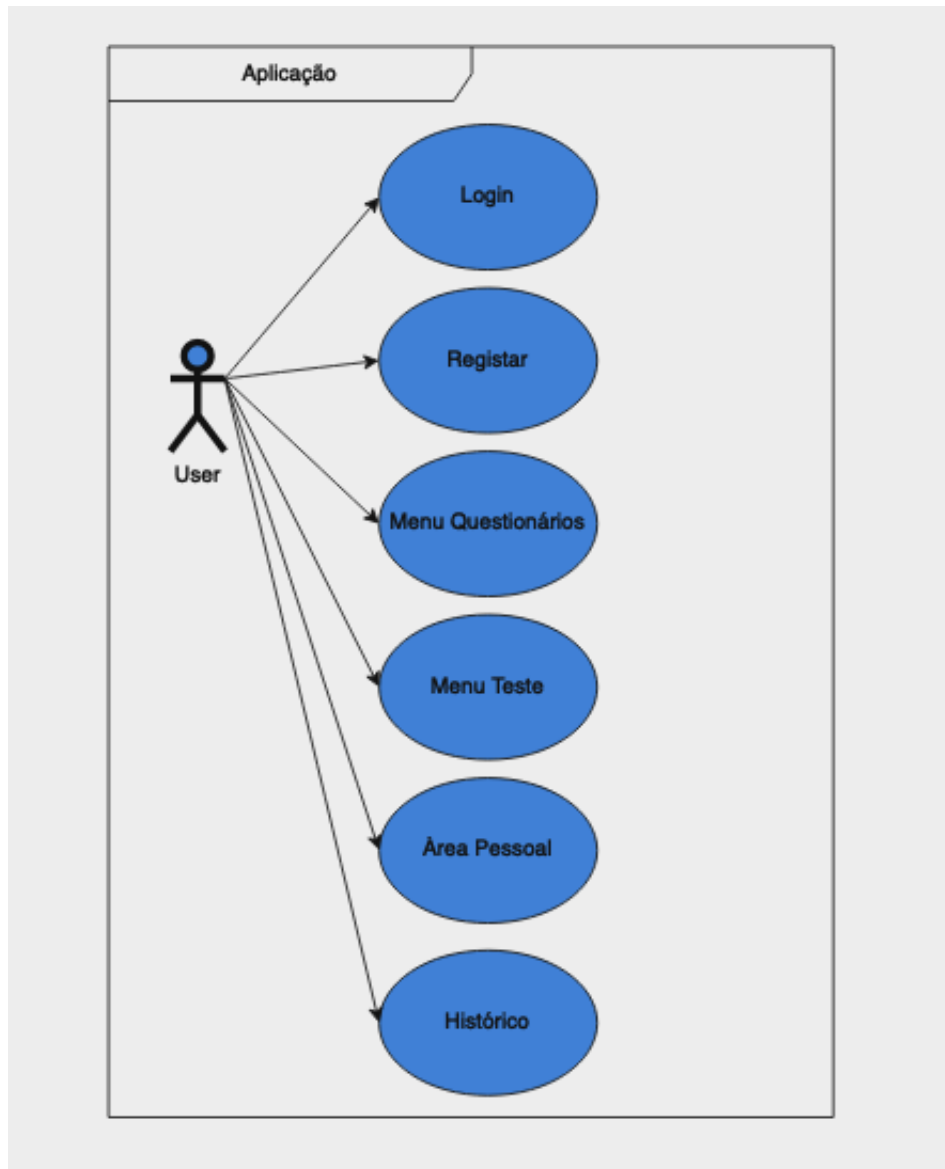


Figura 3.1: Ilustração da Aplicação

## 3.3.3 Diagrama de Caso de Uso - Login/Registo



Figura 3.2: Diagrama Caso de Uso Login e Registar



### 3.3.4 Diagrama de Caso de Uso Questinario

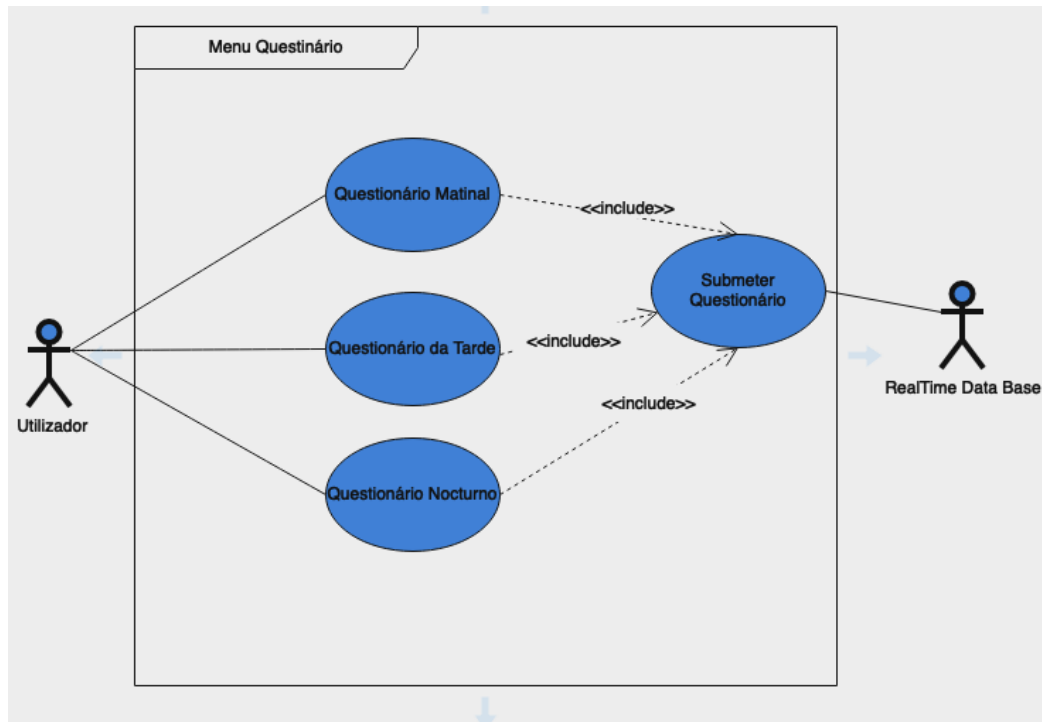


Figura 3.3: Diagrama de Caso de Uso Questinario

### 3.3.5 Diagrama Caso de Uso Teste Tremor

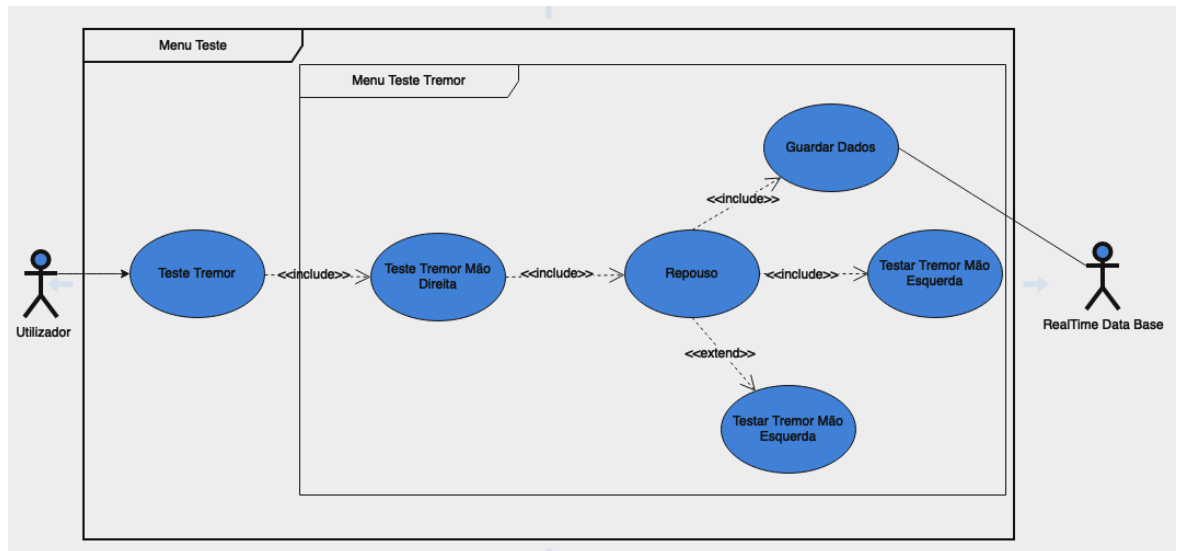


Figura 3.4: Diagrama Caso de Uso Teste Tremor

## 3.4 Diagramas de Sequência

[8] Um Diagrama de Sequência visa identificar as interações entre os atores, objetos ou classes ao longo de um período de tempo, a ordem e a sequência é essencial para a representação deste diagrama.

O primeiro ponto é identificar o ator principal que vai despoletar a interação, que pode ser um subsistema, classe ou objeto que vai dar origem à sequência.

As componentes principais presentes nos diagramas são o ator principal, Utilizador a Aplicação que representa a lógica da aplicação com que o utilizador está a interagir, *Authentication*, que representa um serviço externo que gere a autenticação do utilizador e o *Realtime Database* que representa uma base de dados em tempo real que armazena e sincroniza dados.

Foram realizados três Diagramas de Sequência.

### 3.4.1 Diagrama de Sequência Novo Utilizador

No primeiro, quem despoleta a sequência é o utilizador. Este diagrama representa a sequência de acontecimentos e o período de tempo em que cada componente está ativa, para o Registo de um novo utilizador.

O utilizador tem acesso à aplicação, a aplicação inicia a atividade registar o utilizador preenche os dados para uma nova autenticação e submete-os. A aplicação faz o pedido de uma nova autenticação e criação de uma referência de base de dados ao *FireBase*. O *FireBase* responde com a criação de um novo utilizador, a criação da nova referência de base de dados e com a confirmação da gravação de dados. A aplicação abre a atividade *login* e o utilizador submete os dados para o *login*, a aplicação faz um pedido de autenticação ao *FireBase*, se for bem sucedido a aplicação abre a atividade Menu Principal.

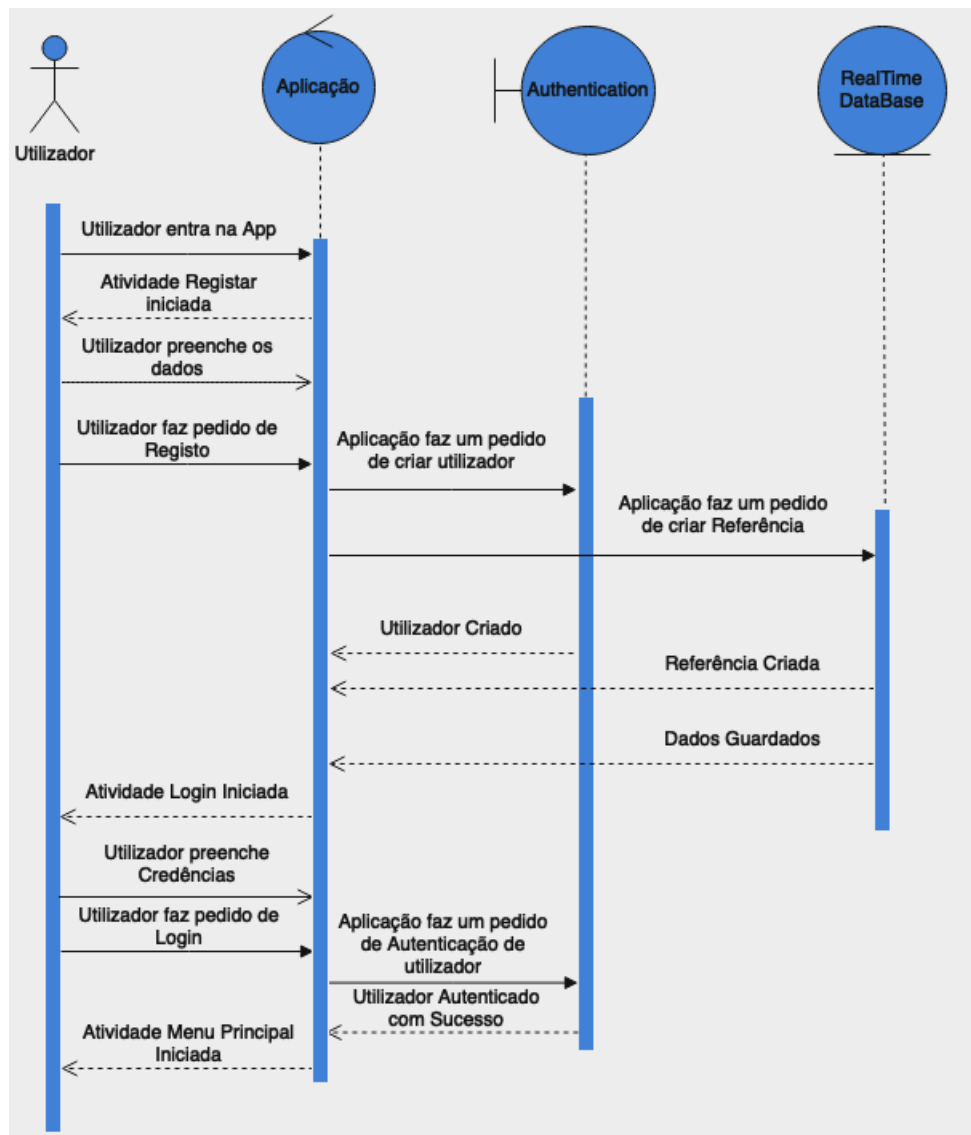


Figura 3.5: Diagrama de Sequência Novo Utilizador

### 3.4.2 Diagrama de Sequência Questionário

No segundo Diagrama de Sequência quem despoleta a ação é a aplicação. Visa representar a sequência de interações para a realização de um Questionário. Está abre a atividade Menu Principal, o utilizador quer fazer um Questionário, a aplicação abre a Atividade Questionários, o Utilizador preenche o Questionário, Submete o Questionário a Aplicação envia os dados do Questionário para o *RealTime DataBase* e se os dados forem salvos com sucesso a Aplicação redirecionada o utilizador para a Atividade Menu Principal

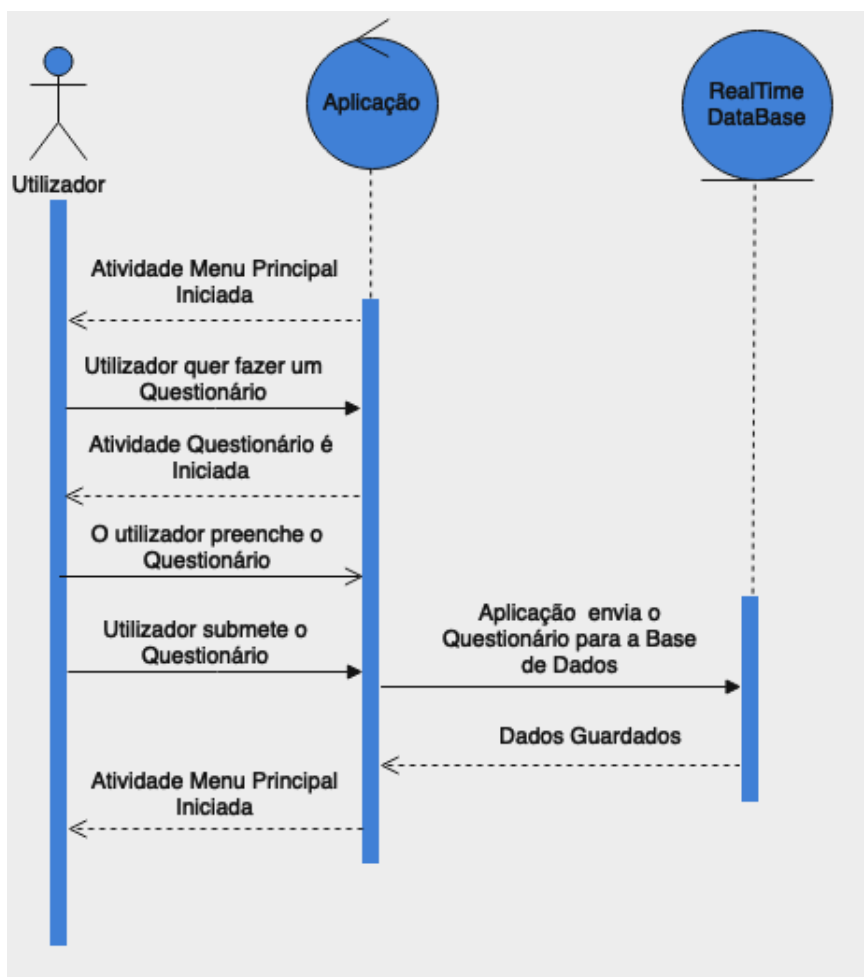


Figura 3.6: Diagrama de Sequência Questionário

### 3.4.3 Diagrama de Sequencia Teste

No terceiro Diagrama de Sequência quem despoleta a ação é a Aplicação. Visa representar a sequência de interações para a realização de um Teste. Está abre a Atividade Menu Principal, o utilizador quer fazer um Teste, a aplicação abre a Atividade Teste, o utilizador faz o teste, submete o teste, a aplicação pede ao utilizador para esperar 10 segundos, o utilizador realiza o teste na outra mão, o utilizador submete os testes, a aplicação envia os dados do Teste para o *RealTime DataBase* e se os dados forem salvos com sucesso a Aplicação re-direciona o Utilizador para a Atividade Menu Principal.

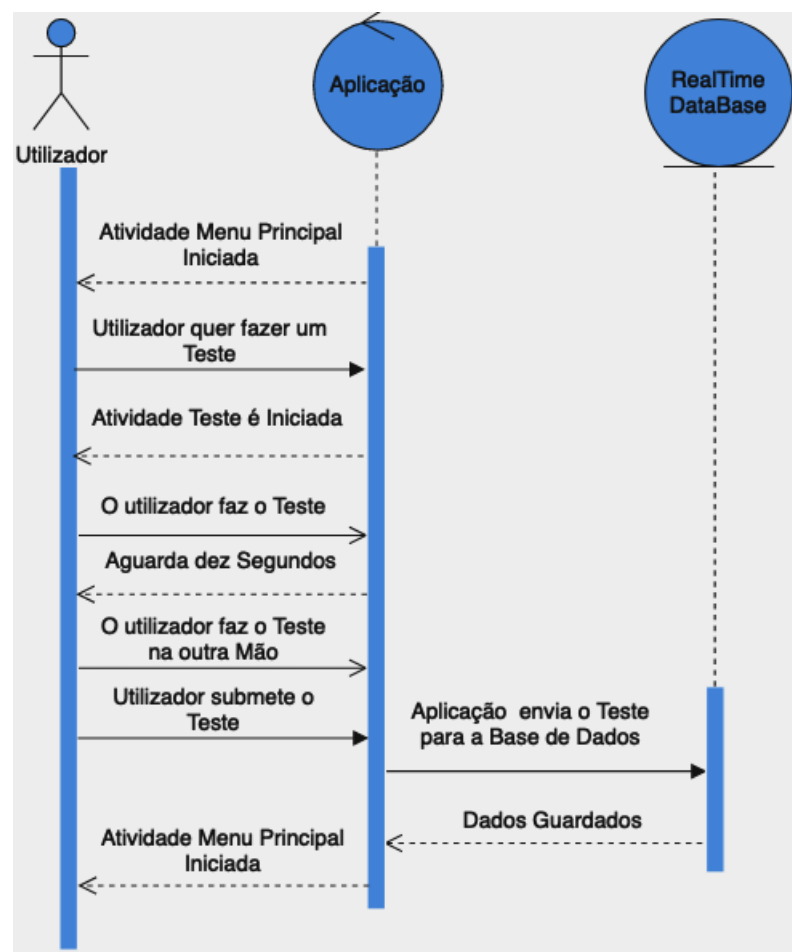


Figura 3.7: Diagrama de Sequencia Teste

## 3.5 Tecnologias Importantes

Neste subcapítulo vão ser apresentadas as Tecnologias usadas no desenvolvimento do Projeto.

- **O *Android Studio*** - é o ambiente de desenvolvimento integrado IDE usado para desenvolver aplicações para a plataforma *Android*. A versão que foi usada foi *Android Studio Koala* | 2024.1.1.

O *Android Studio* utiliza o *Gradle*, uma ferramenta avançada de compilação, para automatizar e gerir o processo de construção, permitindo que se defina configurações de compilação flexíveis e personalizadas.

Tem acesso a um Emulador rápido e de alto desempenho que permite testar e depurar aplicações em várias configurações de dispositivos *Android* sem a necessidade de hardware físico. Com possibilidade de suporte para várias versões do *Android* e diferentes tamanhos de ecrã e resoluções. Podem ser utilizadas as linguagens de programação *Kotlin* e *Java*.

- ***Java*** - é uma linguagem de programação orientada a objetos desenvolvida na década de 90. Diferente das linguagens de programação modernas, que são compiladas para código nativo, *Java* é compilada para um bytecode que é interpretado por uma máquina virtual Java Virtual Machine (JVM).

A linguagem Java é assente em 3 princípios fundamentais: **encapsulamento, herança e poliformismo**.

Uma ***Java Classe*** é um elemento do código *Java* que é utilizado para representar objetos do mundo real. Dentro dela é comum declarar atributos e métodos, que representam, respectivamente, as características e comportamentos desse objeto. São declaradas com a primeira letra em maiúscula.

Um **Objeto** é uma instância de uma classe. Quando se cria um objeto, está-se a criar uma instância específica dessa classe com valores específicos para os atributos definidos na classe.

- ***XML*** - é uma linguagem de marcação que funciona com tags, é usado para definir a estrutura e aparência dos layouts da Interface de Utilizador (IU).

- **FireBase** - disponibiliza vários tipos de serviços, contudo os que serão utilizados neste projeto são:
  - **Firestore Authentication** [9] - tem como objetivo facilitar a construção de sistemas seguros de autenticação, melhorando ao mesmo tempo a experiência de *login* dos utilizadores;  
Quando um utilizador é autenticado usando o Firestore Authentication, é-lhe atribuído um identificador único chamado de UID. Este UID é exclusivo para cada utilizador dentro de cada projeto Firestore e serve para identificá-lo de forma única. O UID é usado muitas vezes para associar dados a um utilizador específico no Firestore RealTime Database.
  - **Firestore Realtime Database** [10] - é uma base de dados *NoSQL* que armazena e sincroniza dados em tempo real. Os dados são armazenados em formato *JSON*, permitindo uma estrutura flexível e hierárquica.
- **Ciclo de Vida de uma Atividade** -  
O ciclo de vida de uma Atividade é um conjunto de callbacks que permitem que a Atividade saiba que seu estado está mudando: se está sendo criada, parada, reiniciada ou destruída.  
Aqui fica uma imagem que ilustra o métodos do Ciclo de Vida de uma Atividade:



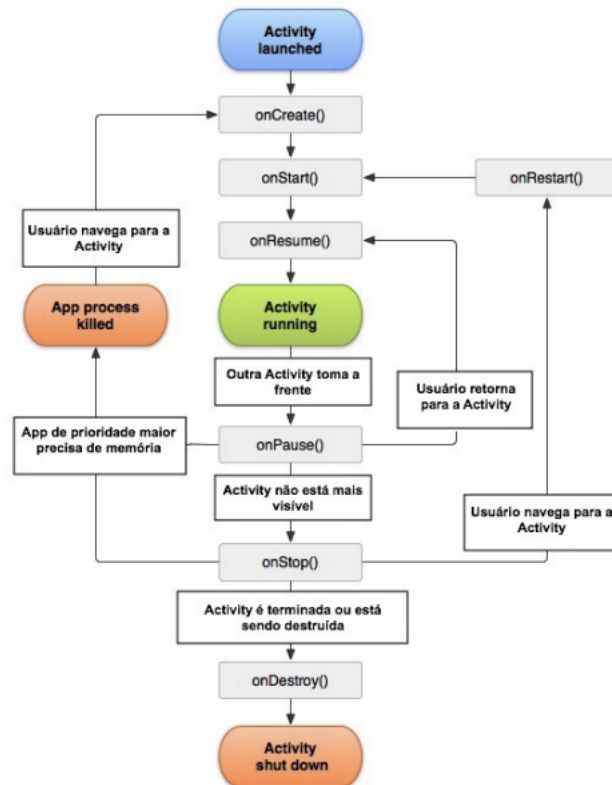


Figura 3.8: Ciclo de Vida de uma Atividade

- **Intentos** - Em *Android*<sup>TM</sup>, um *intent* é um objeto que encapsula, de forma abstrata, a intenção de uma determinada componente realizar uma ação. Um *intent* é ignorado se a componente alvo não existir ou se não houver componentes capazes de lidar com ele.

### 3.6 Conclusões

Com o objetivo de realizar um projeto deste tipo, a experiência de desenvolver este capítulo foi importantíssima. Foi crucial para decidir as melhores funcionalidades que a aplicação podia ter e para ter uma noção das tecnologias que são necessárias para concluir o objetivo.

## Capítulo

# 4

## Desenvolvimento

### 4.1 Introdução

A relação entre os ficheiros XML e as atividades *Java* no *Android Studio* é fundamental para o desenvolvimento de aplicações *Android*.

Vamos explorar essa relação detalhadamente:

#### **Ficheiros XML(*Layouts*)**

Em *Android*, a IU é definida nos ficheiros XML, localizados na pasta *res/layout* do projeto *Android*;

- Os ficheiros XML são responsáveis pela aparência e estrutura da IU que compõem a Aplicação.
  - Exemplo de componentes em XML: `TextView`, `EditText`, `Button`, `LinearLayout`;
  - Exemplo de atributos em XML: `id`, `layout_width`, `layout_height`.

#### **Atividades *Java*(*Activities*)**

As Atividades são Classes *Java* que são responsáveis pela lógica das aplicações em *Android*. Cada atividade trata do comportamento e da lógica de uma única IU. Esta associação é feita pelo método `setContentView` no método `onCreate`.

#### ***Java Custom Objects***

*Java Custom Objects* são instâncias de Classes *Java* que servem para representar e estruturar dados dentro de uma aplicação *Android*, no projeto a título de exemplo existe a Classe `User`.

## 4.2 Dependências

A nível de dependências foi necessário adicionar no app a nível do gradle:

- `implementation(libs.firebase.auth);`
- `implementation(libs.firebase.database);`
- `implementation(libs.firebase.firestore);`
- `implementation(platform("com.google.firebase:firebase-bom:33.1.0"))`.

Estas dependências foram necessárias para integrar o *Firebase Authentication* e *FireBaseRealtime Database* na aplicação *Android* o plugin :

- `id("com.google.gms.google-services")`

Este *plugin* é necessário para integrar serviços *Google Play* na aplicação *Android*, como o *Firebase* é usado em conjunto com o *Firebase BoM (Bill of Materials)* para configurar e sincronizar as configurações do *Firebase*.

## 4.3 Detalhes de Implementação

### 4.3.1 Atividade Launcher

A IU `activity_launcher.xml` é IU inicial da aplicação.

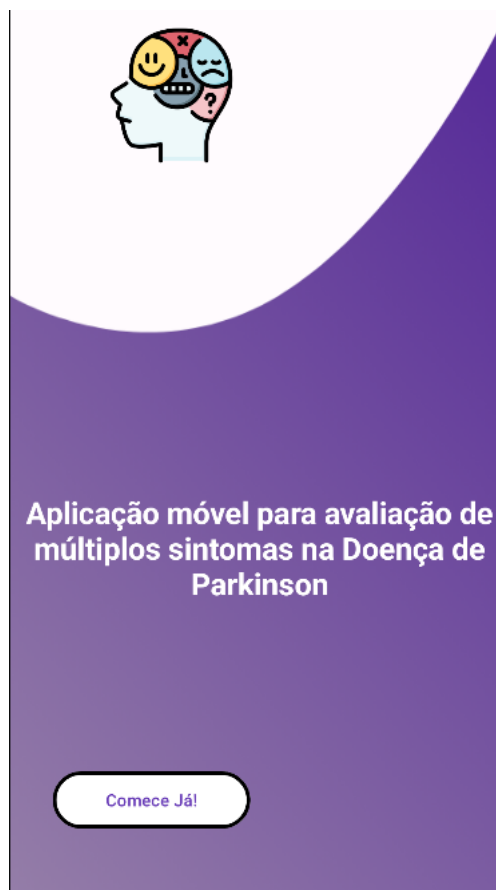
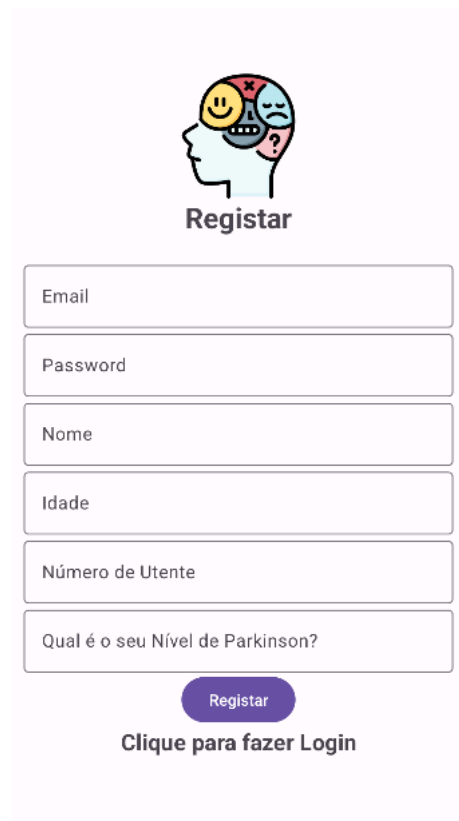


Figura 4.1: `activity_registar.xml`

### 4.3.2 Atividade Registrar

[11] A IU `activity_registar.xml` é onde os utilizadores que não se encontram autenticados na aplicação vão usar para poderem ter acesso à aplicação. Para o registo ser bem sucedido, os utilizadores terão de preencher um pequeno formulário com as suas informações pessoais.



The image shows a registration form with a light pink background. At the top center is a logo of a head profile with a brain divided into four colored sections (yellow, red, blue, green) each containing a different icon (smiley face, sad face, question mark, and a gear-like shape). Below the logo is the word "Registrar" in bold black text. The form consists of six white input fields with rounded corners and thin grey borders, stacked vertically. The labels for these fields are "Email", "Password", "Nome", "Idade", "Número de Utente", and "Qual é o seu Nível de Parkinson?". Below the last field is a purple rounded button with the text "Registrar" in white. At the bottom of the form is a link that says "Clique para fazer Login" in bold black text.

Figura 4.2: `activity_registar.xml`

O ficheiro `activity_registar.xml` é constituído por um `LinearLayout`, dentro dele tem duas `TextView`, a “Registar”, onde indica o nome da *View* em que o utilizador se encontra e a `TextView @+id/loginnow`, onde se o utilizador clicar será redirecionado para a *View Login*. Tem um pequeno formulário onde o utilizador pode introduzir os dados para o registo, formado por seis `TextInputEditText`, a `@+id/email`, `@+id/password`, `@+id/Nome`, `@+id/Idade`, `@+id/nutente` e a `@+id/nivel_parkinson` e por fim o botão `@+id/btn_registar`, que é usado para submeter os dados.

O trecho de código seguinte mostra a `activity_registar.xml`:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:orientation="vertical"
    android:gravity="center"
    android:layout_height="match_parent"
    android:padding="15dp"
    tools:context=".Registar">

    <TextView
        android:text="@string/regarstar"
        android:textSize="25sp"
        android:textStyle="bold"
        android:gravity="center"
        android:layout_marginBottom="20dp"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

    <com.google.android.material.textfield.TextInputLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <com.google.android.material.textfield.TextInputEditText
            android:id="@+id/email"
            android:hint="@string/email"
            android:layout_width="match_parent"
            android:layout_height="wrap_content" />
    </com.google.android.material.textfield.TextInputLayout>
    ...
</LinearLayout>
```

Excerto de Código 4.1: Trecho de código XML usado na `activity_registar.xml`.

Um dos conceitos falados num capítulo anterior, foi o de **Herança**, que é um dos pilares da POO, que permite que uma Classe, chamada de sub-classe, herde todas as características, neste caso métodos e atributos de outra Classe, a superclasse. Esta relação é feita com a palavra *extends* na linguagem *Java*. Neste caso a Classe *Registrar.java* vai ser subclasse da Classe *AppCompatActivity.java*.

Outro dos conceitos referidos anteriormente, foi o de **Poliformismo**, que também é um dos pilares da POO. É um conceito relacionado à extensão e reutilização de métodos da SuperClasse, a palavra *@Override* é usada para indicar que um método na subclasse está substituindo um método da super-classe. Existem dois tipos de Poliformismo: o de sobrecarga e o de sobreposição. Herança e Poliformismo são dois conceitos da POO, que estarão presente ao longo de todo o projeto.

O trecho de código seguinte mostra a Atividade *Registrar.java* e o seu funcionamento:

```
public class Registrar extends AppCompatActivity {
    TextInputEditText editTextEmail, editTextPassword, editTextNome,
        editTextIdade, editTextNivelParkinson, editTextNUtente;
    Button btnRegistrar;
    FirebaseAuth mAuth;
    ProgressBar progressBar;
    TextView textView;
    DatabaseReference ref;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_registar);
        editTextEmail = findViewById(R.id.email);
        editTextPassword = findViewById(R.id.password);
        btnRegistrar = findViewById(R.id.btn_registar);
        editTextNome = findViewById(R.id.Nome);
        editTextIdade = findViewById(R.id.Idade);
        editTextNUtente = findViewById(R.id.nutente);
        editTextNivelParkinson = findViewById(R.id.nivel_parkinson);
        mAuth = FirebaseAuth.getInstance();
        progressBar = findViewById(R.id.progressbar);
        textView = findViewById(R.id.loginnow);

        ref = FirebaseDatabase.getInstance().getReference("users");

        ...
    }
}
```

Excerto de Código 4.2: Trecho de código *Java* usado na *Registrar.java*.

Neste trecho de código podemos visualizar a declaração das variáveis `TextInputEditText`, `Button`, `TextView`, `FirebaseAuth` e `DatabaseReference`. É necessário fazer os devidos Imports durante este processo. Dentro do método `onCreate` são atribuídos valores às variáveis. Como exemplo é usado o `editTextEmail = findViewById(R.id.email)`, é atribuído o `volar` da `TextInputEditText` `email` à variável anteriormente declarada.

`FirebaseAuth mAuth` declara uma variável do tipo `FirebaseAuth`, é uma classe que o *FireBaseAuthentication* fornece para a autenticação de utilizadores. Depois ocorre a inicialização da variável `mAuth = FirebaseAuth.getInstance()`; que chamada o método `getInstance()`; que retorna uma instância que é usada para autenticar, registar novos utilizadores, iniciar sessão etc...  
`ref = FirebaseDatabase.getInstance().getReference("users");` Aqui a variável do tipo *FirebaseDatabase* é inicializada, representa uma referência a um nó específico na base de dados. Assim é possível usar esta referência para escrever e ler dados na base de dados. Neste caso o no `users`.

São usados dois `setOnClickListener`, um fica à escuta se o utilizador clica na `TextViewLogin`, para o redirecionar para a IU `activity_login`. O outro é o que trata do registo do utilizador, de uma forma simples. São criadas novas variáveis com os valores das variáveis que recebem os valores da IU. Há uma verificação se as `TextInputEditText` estão vazias ou não e há uma conversão das Strings em valores numéricos.

Depois à criação do utilizador com o método `createUserWithEmailAndPassword` quando o a criação do utilizador for finalizada há verificação se foi bem sucedida, se sim, é criada uma variável `String uid`, que vai usar o `uid` da autenticação do utilizador como valor de um nó filho da referência `users` criada anteriormente. É instanciado um objeto tipo `User` e vai ser adicionado à base de dados no caminho `"users/userUid"`. O `uid` é usado para identificar as informações do utilizador autenticado na base de dados.

Muitos destes conceitos discutidos nesta subsecção da Atividade Registrar vão ser usados ao longo do projeto. Por isso fica aqui uma explicação detalhada do conteúdo desta atividade.



### 4.3.3 Atividade Login

[12] A IU `activity_login.xml` é onde os utilizadores podem fazer o *login*. Para isso é necessário preencher as credenciais, *email* e *password*.

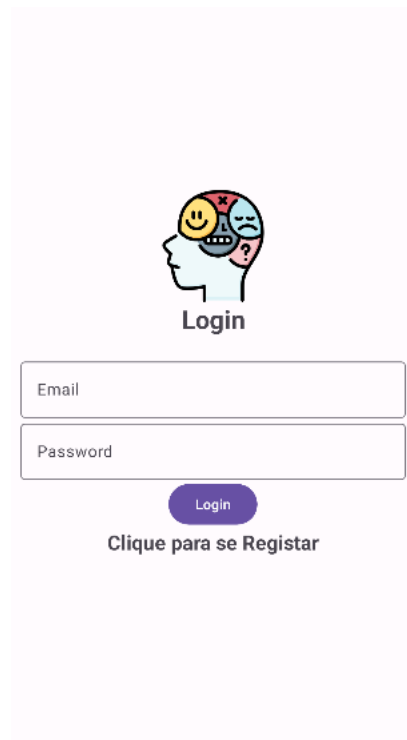


Figura 4.3: `activity_login.xml`

O ficheiro `activity_login.xml` é constituído por um `LinearLayout`, dentro dele tem duas `TextView`, a “Login”, onde indica o nome da IU em que o utilizador se encontra e a `TextView @+id/registarnow`, onde se o utilizador clicar será redirecionado para a IU Registrar. Tem um pequeno formulário onde o utilizador pode introduzir os dados para o Login, formado por duas `TextInputEditText`, a `@+id/email` , `@+id/password` e por fim o botão `@+id/btn_login`, que é usado para submeter os dados.

O trecho de código seguinte mostra a `activity_login.xml`:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:orientation="vertical"
    android:gravity="center"
    android:layout_height="match_parent"
    android:padding="15dp"
    tools:context=".Login">
    ...
</LinearLayout>
```

Excerto de Código 4.3: Trecho de código XML usado na `activity_login.xml`.

O trecho de código seguinte mostra a Atividade `Login.java` e o seu funcionamento:

```
...
 mAuth.signInWithEmailAndPassword(email, password)
    .addOnCompleteListener( new OnCompleteListener<AuthResult>() {
        @Override
        public void onComplete(@NonNull Task<AuthResult> task) {
            progressBar.setVisibility(View.GONE);
            if (task.isSuccessful()) {
                Toast.makeText(Login.this, "Autentica o Completa.",
                    Toast.LENGTH_SHORT).show();
                Intent intent = new Intent(getApplicationContext(),
                    MainActivity.class);
                startActivity(intent);
                finish();
            } else {
                Toast.makeText(Login.this, "Autentica o falhada.",
                    Toast.LENGTH_SHORT).show();
            }
        }
    });
...

```

Excerto de Código 4.4: Trecho de código *Java* usado na `Login.java`.

Na Atividade `Login` é usado o método `signInWithEmailAndPassword` que avalia se existe algum utilizador já criado com as credenciais inseridas. É fornecido pelo *Firebase Authentication*, e o método faz parte da Classe `FirebaseAuth` e faz a autenticação dos utilizadores. Caso a autenticação seja bem sucedida o utilizador é redirecionado para a IU `activity_main`.

#### 4.3.4 Atividade Menu Principal

A IU `activity_main.xml` é onde os utilizadores encontram o Menu Principal da Aplicação.

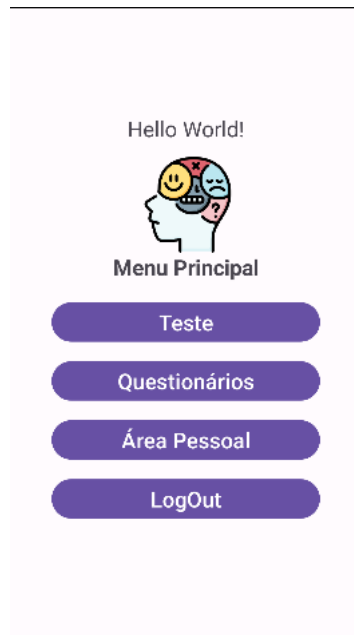


Figura 4.4: `activity_main.xml`

### 4.3.5 Atividade Área Pessoal

A IU `activity_area_pessoal.xml` é onde os utilizadores podem alterar as suas informações pessoais.



Figura 4.5: `activity_area_pessoal.xml`

A IU `activity_area_pessoal.xml` é bastante semelhante à `activity_registar.xml`, contudo tem dois botões onde, o `+id/btn_atualizar` que quando clicado, envia as novas informações preenchidas no formulário para a base de dados e altera as informações antigas pelas novas. O botão `+id/btn_voltar` funciona para retroceder para a IU `activity_main.xml`.

O trecho de código seguinte mostra a Atividade `AreaPessoal.java` e o seu funcionamento:

```
...
FirebaseUser user = mAuth.getCurrentUser();
if (user != null) {
    String uid = user.getId();
    User userInfo = new User(nome, idade, nutente, nivelParkinson);

    ref.child(uid).setValue(userInfo)
        .addOnSuccessListener(new OnSuccessListener<Void>() {
            @Override
            public void onSuccess(Void aVoid) {
                progressBar.setVisibility(View.GONE);
                Toast.makeText(AreaPessoal.this, "Atualiza o
                    completa.", Toast.LENGTH_SHORT).show();
            }
        })
        .addOnFailureListener(new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception e) {
                progressBar.setVisibility(View.GONE);
                Toast.makeText(AreaPessoal.this, "Falha ao atualizar
                    os dados na base de dados.", Toast.LENGTH_SHORT
                ).show();
            }
        });
}
...
```

Excerto de Código 4.5: Trecho de código *Java* usado na `AreaPessoal.java`.

O trecho de código anterior é muito semelhante ao que foi implementado anterior na Atividade `Registrar`, ele encontra-se dentro de um `setOnClickListener`, que fica à escuta se o utilizador clica no botão `@+id/btn_atualizar` para atualizar as informações pessoais na base de dados. As alterações vão ser feitas no nó `users` com o `uid` do atual utilizador. O resultado é alcançado com o comando `ref.child(uid).setValue(userInfo)`.

### 4.3.6 Menu Questionários

A IU `activity_QuestionarioMatinal.xml` é onde os utilizadores podem encontrar os vários Questionários .

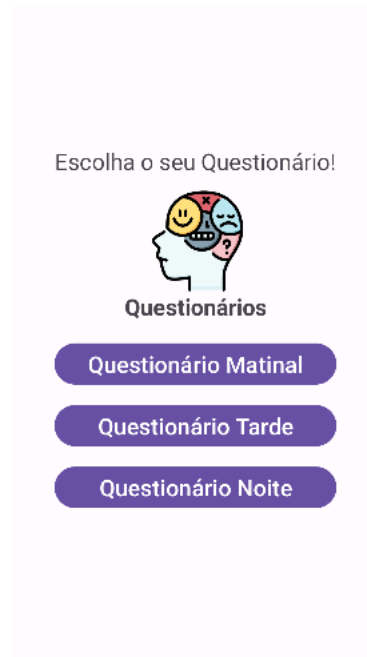


Figura 4.6: `activity_MenuQuestionarios.xml`

### 4.3.7 Menu Questionários

A IU `activity_QuestionarioMatinal.xml` é onde os utilizadores podem fazer os Questionários Matinais.

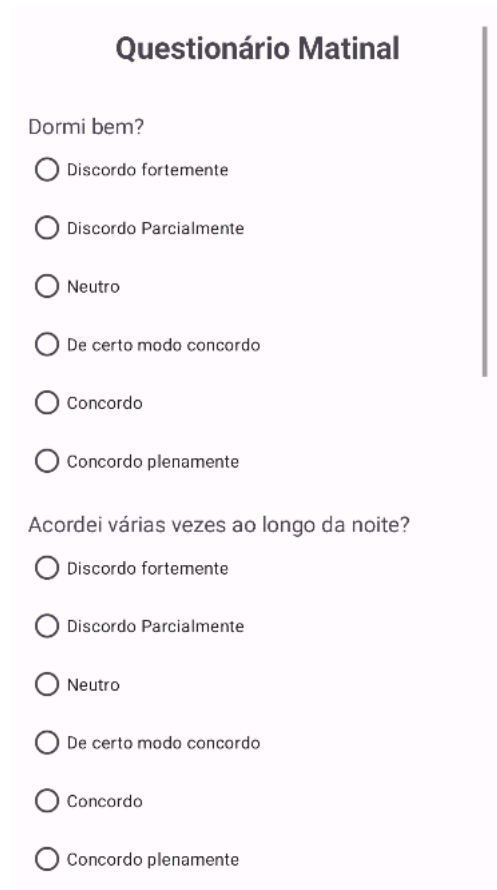


Figura 4.7: `activity_QuestionarioMatinal.xml`

A IU `activity_QuestionarioMatinal.xml` é constituída por uma `ScrollView`, que permite descer e subir o ecrã. Dentro tem um `LinearLayout`, que é constituído por `RadioGroups`, que representam as Perguntas e Respostas, as opções de Respostas são representadas por `RadioButtons`, com um *id* e uma tag. Este *id* é um identificador único para cada resposta e a tag é um valor de sentimento associado à resposta.

Este valor de sentimento pode ser usado numa fase futura do projeto, por exemplo para avaliar o desenvolvimento da doença de *Parkinson* no paciente, ou para criar um sistema de notificações ou feedback adaptativo baseado no valor de sentimento.

O trecho de código seguinte mostra a Atividade `QuestionarioMatinal.java` e o seu funcionamento:

```
...
public class QuestionarioMatinal extends AppCompatActivity {

    Button btn_Submeter;
    RadioGroup RD_Pergunta1, RD_Pergunta2, RD_Pergunta3, RD_Pergunta4,
        RD_Pergunta5;

    FirebaseAuth mAuth;
    DatabaseReference ref;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_questionario_matinal);

        RD_Pergunta1 = findViewById(R.id.Pergunta1);
        RD_Pergunta2 = findViewById(R.id.Pergunta2);
        RD_Pergunta3 = findViewById(R.id.Pergunta3);
        RD_Pergunta4 = findViewById(R.id.Pergunta4);
        RD_Pergunta5 = findViewById(R.id.Pergunta5);
    }
    ...
}
```

Excerto de Código 4.6: Trecho de código *Java* usado na `QuestionarioMatinal.java`.

Inicialmente as variáveis globais são declaradas. Dentro do método `onCreate`, as variáveis recebem os valores dos `RadioGroups`, existe um `setOnClickListener` que fica à escuta se o utilizador clica no `btn_Submeter`, quando isso acontece, é chamado o método `submeterRespostas()`, que é responsável pelo processamento dos objetos `Resposta`, para mais tarde serem inseridos no *RealTime DataBase*.

Dentro o `submeterRespostas()` é criada uma variável inteira com o valor do *id* do `RadioButton`, que foi selecionado no correspondente `textttRadioGroups`.

Este valor inicialmente serve para avaliar se o utilizador respondeu a todas as perguntas, isto acontece na condição que está associada ao `if`. Dentro do `if`, é criado um objeto `RadioButton` o `findViewById()` procura a view correspondente ao *id* da variável inteira anteriormente criada. Depois é necessário encontrar o valor de sentimento da resposta, para isso usa-se o objeto `RadioButton` criado, já que este tem a si associada uma tag, que será o valor de sentimento.



Entretanto é instanciado cada objeto Resposta, com os dados que foram processados. Depois é chamado o metodo `salvarRespostaNoFirebase()`, com o argumento Resposta. Dentro deste método há a criação de uma nova `DatabaseReference`. O objeto `mapaResposta` do tipo `HashMap<>` é instanciado e enviado para a base de dados.

#### 4.3.8 Menu Teste de Tremores

A IU `activity_teste_tremor.xml` é onde os utilizadores podem fazer testes de Tremores.



Figura 4.8: `activity_teste_tremor.xml`

### 4.3.9 Teste de Tremores Mão Direita

A IU `activity_teste_tremor.xml` é onde os utilizadores podem fazer testes de Tremores.



Figura 4.9: `activity_teste_tremor.xml`

O trecho de código seguinte mostra a Atividade `TesteTremor.java` e o seu funcionamento:

```
...
public class TesteTremor extends AppCompatActivity implements
    SensorEventListener {

    SensorManager sensorManager;
    Sensor accelerometer;
    Sensor gyroscope;
    TextView textView;
    Button btnComecarTeste, btnSairMenu;
    DatabaseReference ref;
    String uid;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_teste_tremor);

        textView = findViewById(R.id.textView);
        btnComecarTeste = findViewById(R.id.btnComecarTeste);
        btnSairMenu = findViewById(R.id.btn_Sair);
        sensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
        ;
        accelerometer = sensorManager.getDefaultSensor(Sensor.
            TYPE_ACCELEROMETER);
        gyroscope = sensorManager.getDefaultSensor(Sensor.TYPE_GYROSCOPE
            );
        ref = FirebaseDatabase.getInstance().getReference();
        FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
        uid = user != null ? user.getId() : null;
    }
    ...
}
```

Excerto de Código 4.7: Trecho de código *Java* usado na `QuestionarioMatinal.java`.

As variáveis globais são declaradas, são declaradas duas variáveis do tipo `Sensor`, e uma do tipo `SensorManager`. Dentro do método `onCreate` a variável do tipo `SensorManager` é inicializada com uma instância de `SensorManager`, `getSystemService(SENSOR_SERVICE)` é um método que obtém o serviço de um sensor do sistema.

As outras duas variáveis são inicializadas com o sensor de acelerômetro e de giroscópio. Existe um `setOnClickListener` que escuta se o utilizador vai clicar no `btnComecarTeste`, se sim, o método `iniciarTeste()` é chamado. A principal função deste método é ligar o acelerômetro e giroscópio através do comando `sensorManager.registerListener` quando o temporizador chegar a 0, os sensores são desligados e o método `onSensorChanged` deixa de

detectar tremores.

O método `onSensorChanged` faz parte da `SensorEventListener`, e é usado para detectar mudanças nos dados do sensor registado. Depois de as mudanças serem detectadas é chamado o método `enviarDadosParaFirebase()`, dentro deste método é criada uma nova referencia da base de dados `TesteRef` que contém a ref `users` e vai adicionar dois nós a árvore JSON teste e um subnó teste mão direita.

## 4.4 Conclusões

Neste capítulo foram abordadas as implementações e lógica por de trás de algumas funcionalidades da aplicação. Foi também explicada a conexão entre as Atividades *Java*, com as IU XML.



## **Capítulo**

# 5

## ***Testes e Validação***

### **5.1 Introdução**

Neste capítulo vai ser abordada a componente de testes depois da implementação estar realizada, contudo muitos dos testes foram feitos, ao longo da implementação, resultante da evolução do próprio projeto.

### **5.2 Especificação e Execução dos Testes**

#### **5.2.1 Teste Sistema *Login* e de Registo**

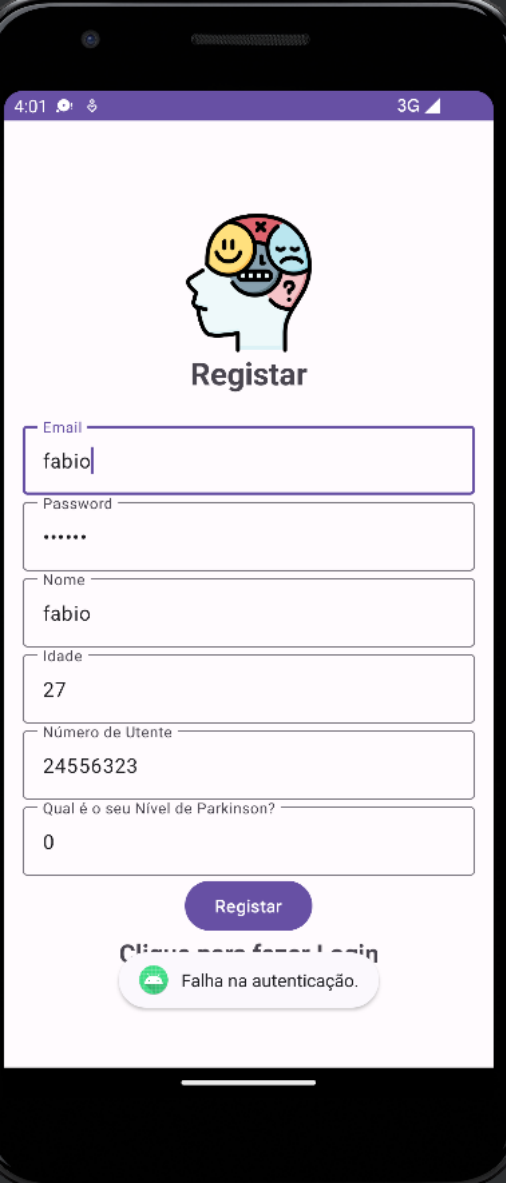
O primeiro conjunto de teste foi feito no sistema de *Login* e Registo, como demonstra as seguintes imagens.



The image shows a smartphone screen with a registration form. At the top, there is a status bar with the time 4:00, signal strength, and 3G connectivity. Below the status bar is a header with a brain icon containing a smiley face, a sad face, and a question mark, and the word "Registrar" in bold. The form consists of several input fields: "Email" with the value "fabio@gmail.com", "Password" with masked characters ".....", "Nome" with the value "fabio", "Idade" with the value "vinte e sete" (where "vinte" is underlined in red), "Número de Utente" with the value "dois quatro" (where "dois" is underlined in red), and "Qual é o seu Nível de Parkinson?" with the value "zero". Below the form is a purple "Registrar" button. At the bottom, there is a green circular icon with a checkmark and a message box that says "A Idade deve ser um valor numérico." (The Age must be a numerical value).

Figura 5.1: Idade tem de ser um Valor Numérico

A título de exemplo foi escolhido a Idade para teste, contudo o Número de Utente e Nível de *Parkinson* tem de ser obrigatoriamente valores numéricos.

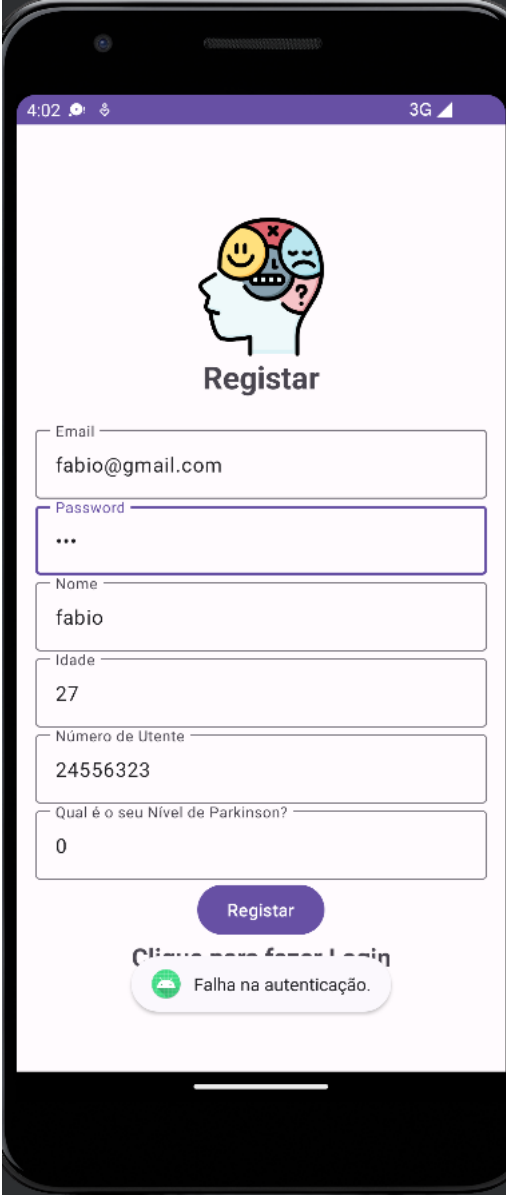


The image shows a smartphone screen with a registration form. At the top, there is a status bar with the time 4:01, signal strength, and 3G connectivity. Below the status bar is a header with a colorful icon of a head with a smiley face, a sad face, and a question mark, and the word "Registar" below it. The form consists of several input fields: "Email" (containing "fabio"), "Password" (containing "\*\*\*\*\*"), "Nome" (containing "fabio"), "Idade" (containing "27"), "Número de Utente" (containing "24556323"), and "Qual é o seu Nível de Parkinson?" (containing "0"). Below the form is a purple "Registar" button. At the bottom, there is a green button with a checkmark and the text "Clique para fazer login". A red error message "Falha na autenticação." is displayed below the login button.

Figura 5.2: Email tem de estar num formato válido

A autenticação não é bem sucedida se não for usado um formato de *email* válido para o registo.





The image shows a smartphone screen with a registration form. At the top, there is a status bar with the time 4:02, signal strength, and 3G connectivity. Below the status bar is a large icon of a head with a brain inside, divided into four colored sections (yellow, red, blue, green) with different expressions (smiley face, 'X', sad face, question mark). Below the icon is the word "Registrar". The form consists of several input fields: "Email" with the value "fabio@gmail.com", "Password" with three dots, "Nome" with the value "fabio", "Idade" with the value "27", "Número de Utente" with the value "24556323", and "Qual é o seu Nível de Parkinson?" with the value "0". Below the form is a purple "Registrar" button. At the bottom, there is a green button with a checkmark icon and the text "Clique para fazer login". Below that is a red error message box with a white exclamation mark icon and the text "Falha na autenticação."

Figura 5.3: A palavra Passe pequena

A palavra passe tem de ter no mínimo 6 caracteres

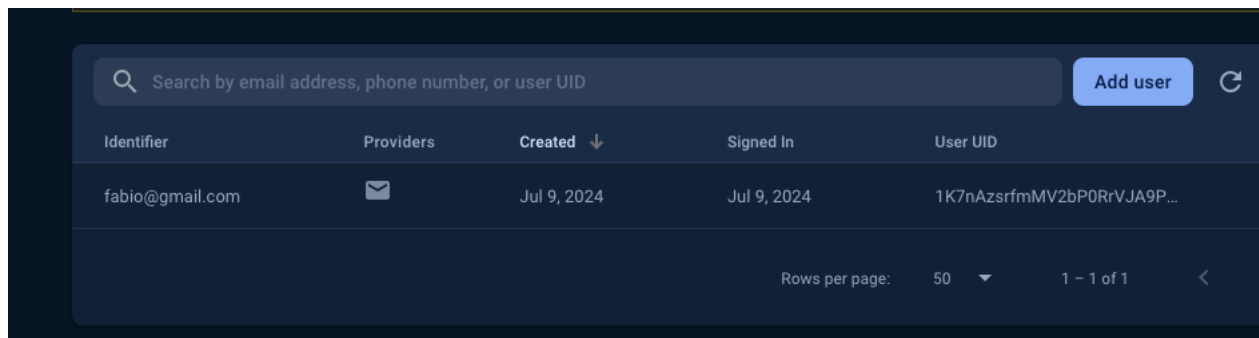


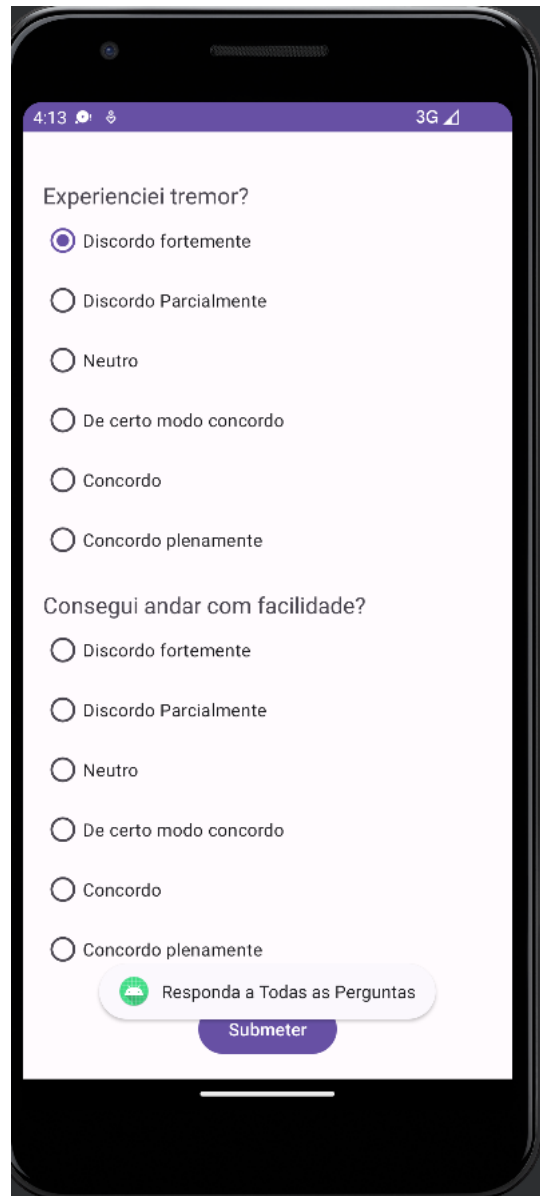
Figura 5.4: Autenticação no FireBase

O User Id (UID) que é criado no *FireBase Authentication* vai ser o mesmo do nó que vai identificar este utilizador.



Figura 5.5: Criação da Árvore no RealTime DataBase

### 5.2.2 Teste do Menu Questionários




The image shows a smartphone screen displaying a questionnaire. The status bar at the top shows the time 4:13, signal strength, and 3G connectivity. The first question is "Experienciei tremor?" with five radio button options: "Discordo fortemente" (selected), "Discordo Parcialmente", "Neutro", "De certo modo concordo", and "Concordo". The second question is "Conseguir andar com facilidade?" with the same five radio button options. At the bottom, there is a green button with a checkmark icon and the text "Responda a Todas as Perguntas", and a purple button labeled "Submeter".

Figura 5.6: Responder a todas as Perguntas

É necessário responder a todas as perguntas para submeter um questionário.

### 5.2.3 Teste do Menu Questionários



The screenshot shows a mobile application interface for editing personal information. At the top, the status bar displays the time 4:15, signal strength, and 3G connectivity. The app's header bar is purple and contains the text "Edite as suas Informações!". Below the header is a stylized icon of a head with a smiley face, a sad face, and a question mark. The main title "Editar Informações" is centered. The form consists of five input fields: "Email" (containing "fabioteste@gmail.com"), "Nome" (containing "Fabioteste"), "Idade" (containing "30"), "Número de Utente" (containing "31313131"), and "Nível de Parkinson" (containing "2"). A purple "Atualizar" button is positioned below the form. At the bottom, a green notification bubble with a checkmark icon displays the message "Atualização completa."

Edite as suas Informações!



**Editar Informações**

Email  
fabioteste@gmail.com

Nome  
Fabioteste

Idade  
30

Número de Utente  
31313131

Nível de Parkinson  
2

**Atualizar**

 Atualização completa.

Figura 5.7: Editar Informações Pessoais

O editar perfil não está a editar o *email*.

## 5.3 Conclusões

Com o desenvolvimento deste capítulo foi possível fazer os testes das funcionalidades e constatar alguns erros ou funcionalidades mal implementadas. Por fim, e ainda com alguma falta de experiência na área de testes, foram descritos todos aqueles que foram feitos, em diferentes plataformas, e da melhor maneira possível.

## **Capítulo**

# 6

## ***Conclusões e Trabalho Futuro***

### **6.1 Conclusões Principais**

Em relação aos objetivos propostos com o estudo do capítulo Estado da Arte, muitos dos recursos funcionais e não funcionais, ficaram por desenvolver. Em relação aos objetivos da minha proposta o sistema de login/registo foi completado, contudo sem recuperar palavra passe ou registo com conta Google, os questionários foram implementados com sucesso, com vista a um trabalho futuro de a aplicação dar feedback baseado nas respostas que o utilizador dá, usando o valor de sentimento. Contudo os objetivos mais ambiciosos não foram atingidos, criar algoritmos de tratamento de dados dados baseados nas frequências de tremores e variações da voz do utilizador.

Em relação a conclusões tiradas com este trabalho, tenho várias, desde uma reflexão dos meus métodos de planeamento e trabalho. A minha abordagem a este projeto não foi a melhor, tentei aprender uma ferramenta nova e quando voltei atrás na minha decisão já tinha perdido tempo precioso. Em relação a conclusões mais técnicas, fico bastante feliz, mesmo não atingindo os principais objetivos, gostei de trabalhar neste projeto, com uma linguagem de programação que aprecio e com um paradigma que me agradou.

Para mim um dos maiores objetivos, era construir um código de qualidade, usando os pilares da linguagem java, principalmente a herança, contudo passei muito do meu tempo a debater o que deveria implementar, usar herança nas classes User, criar uma classe User, em que o objeto User podia receber questionários, testes etc. . . . Chegando à conclusão tarde demais que deveria ter aproveitado o bom de estar a programar em Android, sendo que é uma linguagem à base de Intentos. Os pilares java deviam ser usados nas Classes Atividades, para prevenir atividades iguais trocando um ou dois va-

lores ao longo de todo o código, é o caso da implementação que realizei nas classes Questionarios e Testes. As classes User e outras servem como apoio às Atividades, esta foi a conclusão que cheguei no âmbito da programação.

## 6.2 Trabalho Futuro

O principal objetivo deste trabalho, inicialmente, era o de criar uma aplicação móvel para assistência de pacientes com a doença de *Parkinson*. Como foi discutido ao longo deste relatório, os pontos principais seriam o de detectar tremores usando os sensores do *smartphone*, vindos do giroscópio e acelerômetro e a captura de voz a partir do microfone. A primeira funcionalidade está implementada, contudo só existe uma recolha de dados, não existe um tratamento, nem análise desses dados. Por isso não será possível constatar se um paciente tem Parkinson ou não, para além que a funcionalidade implementada inunda a base de dados com dados, causando constrangimentos e dificuldades para gerir os dados na base de dados.

O objetivo de recolher dados vindos do microfone não foi atingido.

Seria bastante interessante desenvolver esta aplicação, usando algumas idéias baseadas em aplicações similares. A doença de Parkinson atinge muitas pessoas idosas, ou de idade muito avançada, os sintomas vão piorando ao longo do envelhecimento, muitos destes pacientes vivem sozinhos, ou não tem uma assistência médica 24 horas caso tenha uma grave convulsão por exemplo. Seria interessante criar uma aplicação com perfis, onde médicos, auxiliares e familiares podiam acompanhar os pacientes.

Numa outra fase, quando existisse um algoritmo com alguma precisão, tentar utilizar técnicas de *Machine Learning*, para prever padrões e fazer previsões de crises de tremores, por exemplo. Ou até usar o *Machine Learning* só para treinar o algoritmo de detenção, seria uma excelente ideia.

# Bibliografia

- [1] Medical News Today Mayo Clinic. Doença de parkinson, 2024.
- [2] Professor Paulo Fazendeiro. *Aula 1 - Apresentação do docente e discussão do funcionamento geral da unidade curricular: horário das aulas, critérios de avaliação, material de apoio e programa. Discussão do âmbito e da motivação para estudo dos temas que a unidade curricular aborda. Definição de dispositivo móvel.* UBI, 2022/2023.
- [3] Professor Paulo Fazendeiro. *Sebenta de Programação de Dispositivos Móveis.* UBI, 2022/2023.
- [4] Professor Paulo Fazendeiro. *Aula 3 - Discussão da pilha de software que define a plataforma Android™, bem como das 4 camadas que a compõem. Apresentação e discussão de algumas ferramentas para depuração de aplicações Android™.* UBI, 2022/2023.
- [5] Professor Paulo Fazendeiro. *Aula 4 - Descrição dos blocos fundamentais que constituem as aplicações Android™ e análise do seu processo de preparação e compilação. Foco na componente Activity, descrevendo com detalhe o ciclo de vida e os mecanismos de gestão das atividades no sistema operativo.* UBI, 2022/2023.
- [6] Professor Paulo Fazendeiro. *Aula 2 - Discussão do tema relativo ao desenvolvimento de Interfaces de Utilizador como forma de introduzir conceitos específicos ao desenho de aplicações interativas, partindo das Interfaces de Utilizador para programas sequenciais. Introdução à arquitetura para desenvolvimento de aplicações interativas conhecida por ModelView-Controller.* UBI, 2022/2023.
- [7] Professor Paulo Fazendeiro. *Acesso a sensores em dispositivos móveis. Discussão da framework que abstrai esse acesso na plataforma Android™, mais especificamente através das classes SensorManager, Sensor e SensorEvent, bem como da interface SensorEventListener.* UBI, 2022/2023.
- [8] Creately Blog. Tutorial do diagrama de sequência: Guia completo com exemplos, 2024.



- [9] Google. Firebase authentication, 2024.
- [10] Google. Firebase realtime database, 2024.
- [11] Master Coding. Firebase firestore tutorial 9 - custom java object - using model class to receive data, 2019.
- [12] Codes Easy. Login and registration using firebase in android, 2022.