



RELATÓRIO FINAL  
TRABALHO DE PROJETO  
DA LICENCIATURA EM INFORMÁTICA

**DESENVOLVIMENTO DE UMA PLATAFORMA DE  
MONITORIZAÇÃO DE DADOS MULTISENSORIAIS EM TEMPO REAL**

DIOGO MANUEL CRISPIM PEREIRA  
A036751

Orientado por  
Prof. Doutor Luís Carlos Gonçalves dos Santos Seco

Maia, 26 de Abril, 2023



## FICHA DE CARACTERIZAÇÃO

**Instituição de Ensino Superior:** UNIVERSIDADE DA MAIA - UMAIA

**Licenciatura em:** INFORMÁTICA

**Tema do Trabalho de Projeto:** DESENVOLVIMENTO DE UMA PLATAFORMA DE MONITORIZAÇÃO DE DADOS MULTISENSORIAIS EM TEMPO REAL

**Nome Completo do Aluno:** DIOGO MANUEL CRISPIM PEREIRA

**Número de Aluno:** A036751

**Nome do Supervisor:** LUÍS CARLOS GONÇALVES DOS SANTOS SECO

**Repositório:** [https://github.com/diogopereira00/UMAIA\\_Movesense](https://github.com/diogopereira00/UMAIA_Movesense)

Diogo Crispim

Assinatura do Aluno

Luís Gonçalves Seco

Assinatura do Orientador

## **Resumo**

Este relatório de projeto apresenta o desenvolvimento de uma plataforma de monitorização de dados multissensoriais em tempo real, criada no âmbito de um projeto de Licenciatura em Informática da Universidade da Maia. A plataforma foi desenvolvida com o objetivo de recolher e analisar dados precisos e em tempo real sobre o desempenho físico e mental das pessoas, permitindo que os profissionais de saúde desenvolvam programas de terapia personalizados e adaptados às necessidades individuais de cada paciente.

Para isso, foi criada uma aplicação para dispositivos móveis Android, que inclui a recolha de dados via sensores Movesense, bem como um conjunto de questionários de pré-treino e pós-treino para avaliar o estado emocional dos participantes antes e após a atividade física. Foi também desenvolvida uma estrutura de base de dados necessária, bem como a estrutura de suporte à aplicação no servidor, permitindo o armazenamento e posterior processamento dos dados. A plataforma criada permite a recolha contínua e em tempo real de dados do sensor Movesense de vários participantes para os seus telemóveis e posteriormente os envia para um servidor web na cloud.

O projeto foi concluído com sucesso, cumprindo todos os objetivos estabelecidos, desde o levantamento e planeamento do projeto, à configuração de servidor e base de dados, desenvolvimento de API e aplicação Android, até aos testes e validações. Em resumo, a plataforma desenvolvida é uma solução útil e inovadora para a monitorização de dados multissensoriais em tempo real, permitindo a avaliação da saúde física e mental das pessoas, bem como o desenvolvimento de programas de terapia personalizados.

**Expressões-chave:** Movesense, Kotlin, Node.JS, Mysql e Recolha de Dados.

## **Abstract**

This project report presents the development of a real-time multisensorial data monitoring platform, created as part of a Bachelor's Degree project in Computer Science at the University of Maia. The platform was developed with the objective of collecting and analyzing accurate and real-time data on the physical and mental performance of individuals, allowing healthcare professionals to develop personalized therapy programs adapted to the individual needs of each patient.

To achieve this, an Android mobile application was created, which includes data collection via Movesense sensors, as well as a set of pre- and post-workout questionnaires to assess the emotional state of participants before and after physical activity. In addition, a necessary database structure was developed, as well as the necessary support structure for the application on the server, allowing for storage and later processing of the data. The platform created allows for continuous and real-time collection of Movesense sensor data from multiple participants on their mobile phones, which are then sent to a web server in the cloud.

The project was successfully completed, meeting all established objectives, from the survey and planning of the project, to the configuration of the server and database, development of API and Android application, to testing and validation. In summary, the developed platform is a useful and innovative solution for the real-time multisensorial data monitoring, allowing for the evaluation of physical and mental health of individuals, as well as the development of personalized therapy programs.

**Key expressions:** Movesense, Kotlin, Node.JS, Mysql and data collection.



## DECLARAÇÃO DE HONRA

Por este meio se declara que este documento é conteúdo original, produzido pelo identificado como autor, não tendo sido previamente apresentado noutra percurso académico ou unidade curricular de qualquer instituição.

Quaisquer referências a outros trabalhos respeitam rigorosamente os regulamentos e melhores práticas de atribuição, sendo apropriadamente marcadas no texto e identificadas na secção *Referências* na página 36, sob as normas IEEE<sup>1</sup> de referenciação.

O autor igualmente declara não se encontrar em qualquer situação efectiva ou potencial de conflito de interesses.

---

<sup>1</sup>IEEE — *Institute of Electrical and Electronics Engineers*



## AGRADECIMENTOS

Desejo exprimir os meus sinceros agradecimentos a todos os que, de alguma forma, permitiram que este projeto se realizasse.

Em primeiro lugar quero agradecer aos meus familiares e amigos por todo o suporte que me deram ao longo deste percurso académico.

Ao meu orientador e professor Luís Seco quero agradecer a oportunidade de poder trabalhar neste projeto e por toda a ajuda e *feedback* dado durante o mesmo.

À professora Isabel Basto e à Vânia Ferreira agradeço a oportunidade de ter conseguido adaptar o meu projeto a um estudo de caso real.

Ao professor João Paredes agradeço toda a ajuda dada ao longo deste percurso académico e todo o conhecimento passado, assim como agradeço também pela disponibilização do modelo de relatório em *Latex* sobre o qual escrevi este relatório.

À professora Cláudia Freitas agradeço a disponibilidade e ajuda dada durante este projeto para o esclarecimento de duvidas e problemas.

Ao Hugo Freire e ao Miguel Silva agradeço todo o tempo em que tiveram disponíveis para me ajudar a estruturar o projeto e a configurar o servidor.

A todos aqueles que experimentaram a aplicação durante os testes agradeço as opiniões e sugestões que foram fundamentais para melhorar aplicação, bem como a paciência e disposição para lidar com alguns *bugs* que possam ter ocorrido.

E por fim, mas não menos importante, agradeço a toda a equipa do LICA da Universidade da Maia que me disponibilizou todos os recursos necessários para o desenvolvimento deste projeto.



## ÍNDICE

<b>Lista de Figuras</b>	<b>iv</b>
<b>Lista de Tabelas</b>	<b>vi</b>
<b>Lista de excertos de código e terminal</b>	<b>vii</b>
<b>Siglas</b>	<b>ix</b>
<b>1 Introdução</b>	<b>1</b>
<b>2 O sensor Movesense</b>	<b>3</b>
2.1 Como é realizada a comunicação? . . . . .	4
2.1.1 API de Conectividade . . . . .	5
2.1.2 MDS REST API . . . . .	5
<b>3 Material e Métodos</b>	<b>6</b>
3.1 Materiais Utilizados . . . . .	6
3.2 Métodos . . . . .	8
3.2.1 Levantamento de requisitos e planeamento do projeto . . . . .	9
3.2.2 Configuração do servidor e criação da base de dados . . . . .	11
3.2.3 Desenvolvimento da API . . . . .	14
3.2.4 Desenvolvimento da aplicação Android . . . . .	18
3.2.5 Testes e Validações . . . . .	25
<b>4 Resultados e Discussão</b>	<b>27</b>
4.1 Levantamento de requisitos e planeamento . . . . .	27
4.2 Servidor e da base de dados . . . . .	27
4.3 API no servidor . . . . .	28
4.4 Aplicação Android . . . . .	29
4.5 Testes e Validações . . . . .	30



<b>5 Conclusões</b>	<b>34</b>
5.1 Objetivos concretizados . . . . .	34
5.2 Problemas e dificuldades . . . . .	34
5.2.1 Problemas . . . . .	34
5.2.2 Dificuldades . . . . .	35
5.3 Trabalho Futuro . . . . .	35
5.4 Considerações finais . . . . .	35
<b>Referências</b>	<b>36</b>
<b>A Vídeos de demonstração Movesense</b>	<b>A1</b>
<b>B Lista de Requisitos Simplificada</b>	<b>A2</b>
<b>C Distribuição das tarefas pelas <i>Sprints</i></b>	<b>A5</b>
<b>D Diagrama de Entidade e Relação</b>	<b>A8</b>
<b>E Questionários</b>	<b>A9</b>
<b>F Referência técnica da API</b>	<b>A12</b>
<b>G Diagrama de atividades</b>	<b>A13</b>
<b>H Interfaces da aplicação</b>	<b>A14</b>
<b>I Blocos de comandos e ficheiros de configuração principais</b>	<b>A31</b>



## LISTA DE FIGURAS

2.1 Movesense 2.0 Developer Kit . . . . .	3
2.2 Sensor Movesense. . . . .	4
3.1 Metodologia Agile . . . . .	8
3.2 <i>Board</i> de tarefas no Trello. . . . .	10
3.3 Arquitetura utilizada no desenvolvimento da plataforma. . . . .	10
3.4 Tabelas da base de dados MySQL . . . . .	14
3.5 Criação de um novo utilizador diretamente pela API . . . . .	16
3.6 Arquitetura utilizada na aplicação . . . . .	18
4.1 API <i>movesense.service</i> em execução . . . . .	27
4.2 Serviço <i>mysql.service</i> em execução . . . . .	27
4.3 Vídeo de demonstração da aplicação . . . . .	29
4.4 Respostas 1 e 2 do formulario . . . . .	30
4.5 Respostas 3,4,5 e 6 do formulario . . . . .	31
4.6 Respostas 8 e 9 do formulario . . . . .	32
A.1 Movesense benefits . . . . .	A1
A.2 What would you create with a programmable movement sensor? .	A1
D.1 Estrutura de base de dados. . . . .	A8
G.1 Diagrama de atividades . . . . .	A13
H.1 <i>Splash Screen</i> . . . . .	A14
H.2 Ecrã de Autenticação . . . . .	A14
H.3 Dados dos questionários adicionados no ecrã de consentimento .	A15
H.4 “Por favor, lê tudo antes de continuar”, ecrã de consentimento .	A15
H.5 Pedido permissão localização do dispositivo. . . . .	A16
H.6 Pedido de permissão de dispositivos próximos. . . . .	A16
H.7 Pedido de permissão notificações . . . . .	A17



H.8 Ecrã de procura de dispositivos Movesense proximos . . . . .	A17
H.9 Pesquisa em curso. . . . .	A18
H.10 A conectar a um sensor. . . . .	A18
H.11 Ecrã principal . . . . .	A19
H.12 Ecrã das definições . . . . .	A19
H.13 Pedido de <i>password</i> administrador . . . . .	A20
H.14 Definições dos sensores, após introduzir a <i>password</i> . . . . .	A20
H.15 Ecrã inicial do questionário pré-treino . . . . .	A21
H.16 Pergunta 1 do questionário pré-treino . . . . .	A21
H.17 Pergunta 2 do questionário pré-treino . . . . .	A22
H.18 Pergunta 3 do questionário pré-treino . . . . .	A22
H.19 Terminar pré-treino e começar atividade . . . . .	A23
H.20 Notificação a avisar da recolha de dados . . . . .	A23
H.21 Notificação a avisar a perda de conexão ao sensor . . . . .	A24
H.22 Mensagem de aviso atividade inferior a 30 minutos . . . . .	A24
H.23 Ecrã inicial do questionário pós-treino . . . . .	A25
H.24 Pergunta 1 do questionário pós-treino . . . . .	A25
H.25 Pergunta 2 do questionário pós-treino . . . . .	A26
H.26 Pergunta 3 do questionário pós-treino . . . . .	A26
H.27 Pergunta 4 do questionário pós-treino . . . . .	A27
H.28 Pergunta 5 do questionário pós-treino . . . . .	A27
H.29 Pergunta 6 do questionário pós-treino . . . . .	A28
H.30 Pergunta 7 do questionário pós-treino . . . . .	A28
H.31 Terminar pós-treino e terminar atividade . . . . .	A29
H.32 Notificação envio de dados para o servidor . . . . .	A29
H.33 Terminar sessão . . . . .	A30



## LISTA DE TABELAS

2.1	Lista de rotas utilizadas na <i>REST like API</i> do Movesense . . . . .	5
3.1	Materiais Utilizados . . . . .	7
3.2	Story points . . . . .	9
4.1	Teste de usabilidade — Media dos participantes . . . . .	33
B.1	Requisitos Funcionais . . . . .	A3
B.2	Requisitos Não Funcionais . . . . .	A4
C.1	Sprint 1 . . . . .	A5
C.2	Sprint 2 . . . . .	A5
C.3	Sprint 3 . . . . .	A5
C.4	Sprint 4 . . . . .	A6
C.5	Sprint 5 . . . . .	A6
C.6	Sprint 6 . . . . .	A6
C.7	Sprint 7 . . . . .	A7
C.8	Sprint 8 . . . . .	A7
C.9	Sprint 9 . . . . .	A7
E.1	Questionário pré-treino . . . . .	A10
E.2	Questionário pós-treino . . . . .	A10
E.3	Teste de usabilidade — Participante 1 . . . . .	A11
E.4	Teste de usabilidade — Participante 2 . . . . .	A11
E.5	Teste de usabilidade — Participante 3 . . . . .	A11
F.1	Caminhos da API . . . . .	A12



## LISTA DE EXCERTOS DE CÓDIGO E TERMINAL

3.1	Comando para a criação de um novo utilizador . . . . .	11
3.2	Comando para a criação da chave SSH . . . . .	11
3.3	Comandos para a instalação do MySQL. . . . .	12
3.4	Comandos para a criação de um novo utilizador no MySQL. . . . .	12
3.5	Comandos para a instalação do <i>phpMyAdmin</i> . . . . .	12
3.6	Ficheriro /etc/systemd/movesense.service. . . . .	15
3.7	Exemplo jsonString acelerometro. . . . .	16
3.8	Exemplo de RxAndroidBle. . . . .	19
3.9	Exemplo de subscrição MDS. . . . .	19
3.10	Classe abstrata safeApiCall. . . . .	23
3.11	Guardar o token de autenticação do utilizador. . . . .	24
4.1	Comandos para instalar os pacotes necessários e correr a API . . . . .	28
I.1	Funçao connectBLEDevice(). . . . .	A31
I.2	Funçao hasInternetWifi(). . . . .	A34
I.3	Funçao hasInternet(). . . . .	A35



## SIGLAS

**API** *Application Programming Interface*

**BLE** *Bluetooth Low Energy*

**ECG** Eletrocardiograma

**EEMPROM** *Electrically-Erasable Programmable Read-Only*

**IEEE** *Institute of Electrical and Electronics Engineers*

**LED** *Light Emitter Diode*

**LICA** *Laboratório de Informática e Computação Avançada*

**MDS** *MoveSense Mobile Library*

**MVVM** *Model, View, ViewModel*

**npm** *node package manager*

**REST** *Representational State Transfer*

**SSH** *Secure Shell*



## **1. INTRODUÇÃO**

Com o avanço da tecnologia e o aumento do uso de dispositivos móveis, é cada vez mais comum o interesse em soluções que utilizem sensores e dispositivos móveis para a monitorização da saúde. Essas soluções podem fornecer aos profissionais de saúde informações precisas e em tempo real sobre o desempenho físico e mental dos pacientes, permitindo assim o desenvolvimento de programas de terapia personalizados e adaptados às necessidades individuais de cada paciente.

No entanto, a monitorização de dados multissensoriais em tempo real apresenta diversos desafios, como a recolha, armazenamento e processamento de grandes quantidades de dados, bem como a integração desses dados com outras fontes de informação, como questionários de saúde. Além disso, a criação de uma plataforma que permita a recolha contínua e em tempo real de dados do sensor para vários participantes requer uma infraestrutura complexa e escalável, que seja capaz de lidar com um grande volume de dados.

Nesse contexto, surge a necessidade de desenvolver uma plataforma de monitorização de dados multissensoriais em tempo real que seja capaz de recolher, armazenar e processar grandes quantidades de dados de forma eficiente e precisa, e que permita a integração desses dados com outras fontes de informação, como questionários de saúde.

Com base nessa necessidade, este projeto de Licenciatura em Informática da Universidade da Maia teve como objetivo o desenvolvimento de uma plataforma de monitorização de dados multissensoriais em tempo real. A plataforma foi desenvolvida com a finalidade de recolher e analisar dados precisos e em tempo real sobre o desempenho físico e mental das pessoas, permitindo que os profissionais de saúde desenvolvam programas de terapia personalizados e adaptados às necessidades individuais de cada paciente.

Para alcançar esse objetivo, foi criada uma aplicação para dispositivos móveis Android, que inclui a recolha de dados via sensores Movesense, bem como um conjunto de questionários de pré-treino e pós-treino para avaliar o estado emocional dos participantes antes e após a atividade física. Foi também desenvolvida toda a estrutura de base de dados necessária, bem como a estrutura de suporte à aplicação no servidor, permitindo o armazenamento e posterior processamento dos dados. A estrutura de base de dados foi concebida para adaptar-se a um projeto de doutoramento de Psicologia Clínica cujo objetivo é avaliar a “Eficácia



da Combinação de Psicoterapia com a Prática de Exercício Físico no Tratamento da Depressão”, mas também para no futuro poder ser utilizada noutros projetos da universidade, uma vez que está desenhada para se adaptar as necessidades específicas de cada projeto.

No percurso traçado para este objetivo principal, foram estabelecidos os seguintes objetivos específicos:

- Efetuar um levantamento de requisitos e planeamento do projeto;
- Configurar o servidor e criar uma base de dados estruturada e escalável que armazene todos os dados dos sensores, dos participantes, e toda a estrutura dos questionários.
- Desenvolver uma plataforma de *backend*<sup>1</sup> no servidor que permita guardar os dados da aplicação no servidor por meio de uma API;
- Desenvolver uma aplicação de fácil uso em *Android* que permita fazer a recolha dos dados sensoriais e dos questionários dos participantes;
- Realizar os testes e validações necessárias para comprovar o bom funcionamento da plataforma.

---

<sup>1</sup>backend — é uma parte da plataforma que os utilizadores não conseguem ver, porém, é fundamental para lidar com tarefas técnicas e lógicas do sistema



## 2. O SENSOR MOVESENSE

O sensor Movesense é um dispositivo eletrónico produzido pela empresa finlandesa *Suunto* e foi desenvolvido para ser utilizado em aplicações de desporto, de saúde e bem-estar. Porém, como é um sensor de código aberto, permite-nos criar várias aplicações para o sensor, o que da outras utilidades ao sensor.[1] (Consultar o apêndice A)

Este dispositivo contém diversos sensores de baixo consumo como:

- Sensores de Movimento (Acelerómetro, Giroscópio e Magnetómetro)
- Sensor de Eletrocardiograma (ECG)
- Sensor de Frequência Cardíaca
- Sensor de Temperatura

Cada sensor tem também um LED para uma indicação visual, e a versão mais recente (e utilizada neste projeto) contém também uma memória EEPROM, que serve para guardar todos os dados do sensor internamente no dispositivo.

A *Suunto* disponibiliza também diversos acessórios para o sensor como, pulseiras de pulso, clippers para a roupa, cintas para o peito, elétrodos de correção de ECG, entre outros.

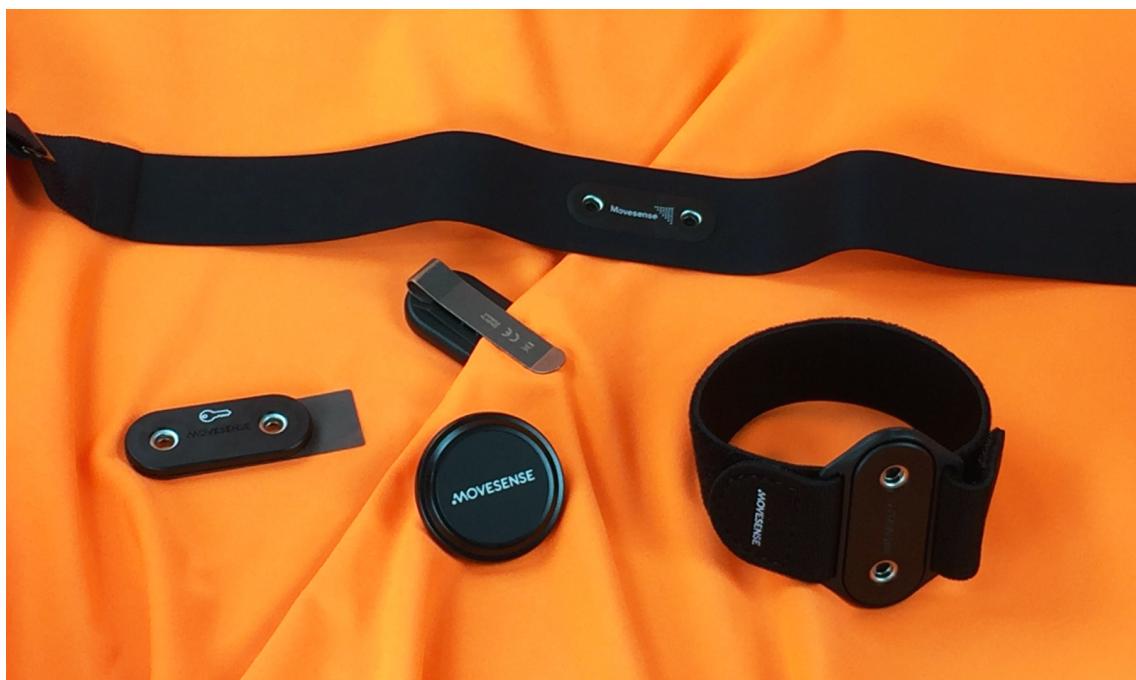


Figura 2.1: Movesense 2.0 Developer Kit

## 2.1 Como é realizada a comunicação?

O sensor utiliza o protocolo *Bluetooth Low Energy* (BLE) para comunicar por *wireless* com o *smartphone* através de uma arquitetura de microsserviços chamada *Whiteboard*, sendo um tipo de comunicação semelhante a uma REST like API onde o funcionamento do sistema baseia-se no envio assíncrono de pedidos pelos clientes e no reenvio assíncrono de uma resposta.

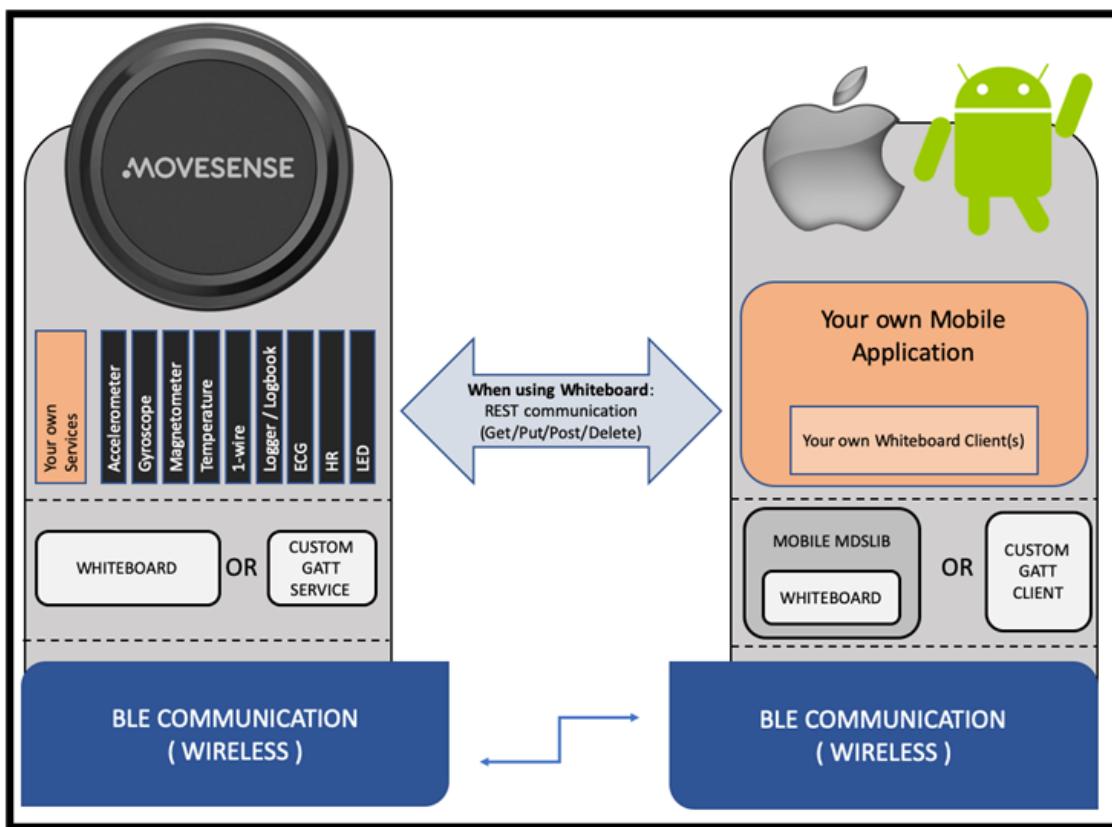


Figura 2.2: Sensor Movesense.

A API da *Whiteboard*, em conjunto com a biblioteca, *Movesense Mobile Library* (MDS) [2] (para *Android*) permite que dispositivos *Android* acessem aos dados do sensor. Esta biblioteca possui dois tipos de métodos: os métodos da API de conectividade e os métodos da API REST do sensor.



### 2.1.1 API de Conectividade

Esta API contém dois métodos: o *connect()* e o *disconnect()* que utilizam o endereço BLE do dispositivo como um parâmetro e permite facilmente conectar e desconectar o sensor ao telemóvel.

### 2.1.2 MDS REST API

Esta API consiste em cinco tipos de métodos: *GET*, *PUT*, *POST*, *DELETE* e *SUBSCRIBE*. Os quatro primeiros métodos são semelhantes aos serviços REST da internet. Já o *subscribe()* é um método especial no sistema *Movesense* que permite subscrever um determinado sensor para receber dados constantemente do mesmo. Por exemplo, para receber os dados do acelerómetro devemos subscrever o sensor utilizando o caminho da API *Meas/Acc/frequencia*.

Neste projeto foi utilizado o método *subscribe()* em cada um dos sensores para receber os dados, pode consultar a lista completa na Tabela 2.1 Lista de rotas utilizadas na *REST like API* do *Movesense*.

Sensor	Rota	Tipo
Acelerometro	/Meas/Acc/{SampleRate}	Subscribe
Giroscopio	/Meas/Gyro/{SampleRate}	Subscribe
Magnetometro	/Meas/Magn/{SampleRate}	Subscribe
Acelerometro + Giroscopio + Magnetometro	/Meas/IMU9/{SampleRate}	Subscribe
Acelerometro + Giroscopio	/Meas/IMU6/{SampleRate}	Subscribe
Acelerometro + Magnometro	/Meas/IMU6m/{SampleRate}	Subscribe
Eletrocardiograma	/Meas/ECG/{RequiredSampleRate}	Subscribe
Frequencia Cardiaca	/Meas/HR	Subscribe
Temperatura	/Meas/Temp	Subscribe

Tabela 2.1: Lista de rotas utilizadas na *REST like API* do *Movesense*

A *SampleRate*<sup>1</sup> pode variar entre 13, 26, 52, 104, 208, 416, 833, 1666 para o acelerómetro, giroscópio ou magnetómetro e entre 125, 128, 200, 250, 256, 500, 512 para o ECG. Quanto menor for a frequência, mais dados são recolhidos durante a subscrição. Neste projeto a *SampleRate* foi sempre a menor, ou seja, 13 no acelerómetro, giroscópio e magnetómetro e 125 no ECG.

<sup>1</sup>SampleRate — frequência da amostra



### 3. MATERIAL E MÉTODOS

Neste capítulo estão detalhados todos os materiais e métodos utilizados no desenvolvimento desta plataforma para recolha de dados sensoriais, para alcançar todos os objetivos estabelecidos.

#### 3.1 Materiais Utilizados

Para conseguir recolher os dados de movimento, de ECG e da frequência cardíaca de cada participante de forma rápida e segura, foi escolhido o sensor *Movesense* da empresa *Suunto* (consultar capítulo *O sensor Movesense*).

No desenvolvimento da aplicação *Android* foi utilizado como ambiente de desenvolvimento a plataforma *Android Studio* com a linguagem de programação *Kotlin*. Estas ferramentas foram escolhidas devido a sua popularidade e devido a minha familiarização as mesmas durante este ciclo de estudos. Foi também necessário criar uma base de dados local, com o intuito de guardar os dados do sensor *offline* e para isso foi utilizada a *Room database*. Para o desenvolvimento dos questionários na aplicação foi utilizada a biblioteca *SurveyKit* da *QuickBirdEng*. Foi utilizado um *Samsung Galaxy S20 FE* para efetuar todos os testes da aplicação.

Foi utilizado um servidor *cloud* com o sistema operativo *Ubuntu*. Neste foi configurado, via *SSH*, um servidor de base de dados *MySQL* com o painel do *PHP MyAdmin* do *Apache* e outro servidor *Express* em *NodeJS* que serve para receber os dados da aplicação *mobile* por meio de uma API e armazena-os na base de dados.

Foi também utilizado o *Postman* para projetar, construir, testar a API.



Sensores Movesense	
Samsung Galaxy S20 FE	 <b>SAMSUNG</b> Galaxy
Android Studio	
Kotlin	
SurveyKit	
Android Room Database	
Ubuntu	
Apache	
MySQL	
NodeJS	
Postman	

Tabela 3.1: Materiais Utilizados

### 3.2 Métodos

Para o planeamento deste projeto foi utilizada a metodologia *agile SCRUM*, que é, cada vez mais utilizada no desenvolvimento de *software* pois permite ter uma maior flexibilidade, maior envolvimento com o cliente final e permite-nos ter uma melhoria contínua e uma maior qualidade relativamente às metodologias mais tradicionais como a de cascata.

O *SCRUM* baseia-se num ciclo de vida espiral que consiste num conjunto de etapas que se repetem em ciclos curtos, as chamadas *sprints*.

O ciclo de vida da metodologia *agile* é composto pelas seguintes etapas, como mostrado na figura 3.1 Metodologia Agile:



Figura 3.1: Metodologia Agile



### 3.2.1 Levantamento de requisitos e planeamento do projeto

#### I. Levantamento de Requisitos

Nesta etapa foram identificados, em reuniões com o professor Luís Seco, com o Miguel Silva, com a professora Isabel Basto e com a Vânia Ferreira de Psicologia, os requisitos necessários de toda a plataforma. Esta é uma parte importante do desenvolvimento de *software* pois permite-nos desenvolver a aplicação ideal para o cliente.

Existem os requisitos funcionais e não funcionais. Os funcionais são aqueles que descrevem as funcionalidades que o *software* deve cumprir para atender as necessidades do cliente, enquanto os não funcionais são aqueles que não estão diretamente relacionados a funcionalidade, mas que afetam a qualidade e desempenho da plataforma.

Os requisitos funcionais e não funcionais podem ser consultados no *Apêndice B, na página A2*.

#### II. Planeamento

Os objetivos de cada *sprint* foram definidos gradualmente com os requisitos anteriormente levantados. Inicialmente estava estimado serem necessárias oito *sprints* de duas semanas cada uma, mas ao longo do projeto, com alguns problemas e com algumas alterações necessárias, foram utilizadas nove *sprints* onde as oito primeiras foram de duas semanas e a última de quatro semanas essencialmente para testes. O calendário de *sprints* pode ser consultado no *Apêndice C, na página A5*.

A seguinte escala de *story points*<sup>1</sup> foi definida para prever e planear cada *sprint*.

Pontos	Duração
1 sp	menos de meio-dia
2 sp	meio-dia a um dia
3 sp	dois a três dias
5 sp	três a quatro dias
8 sp	cinco a sete dias
13 sp	sete a dez dias

Tabela 3.2: Story points

---

<sup>1</sup>story points — estimativas de tempo



Para me ajudar no planeamento das tarefas utilizei o *Trello* que permite criar quadros para gestão de projetos. <https://trello.com/b/MnPafydx/movesense>

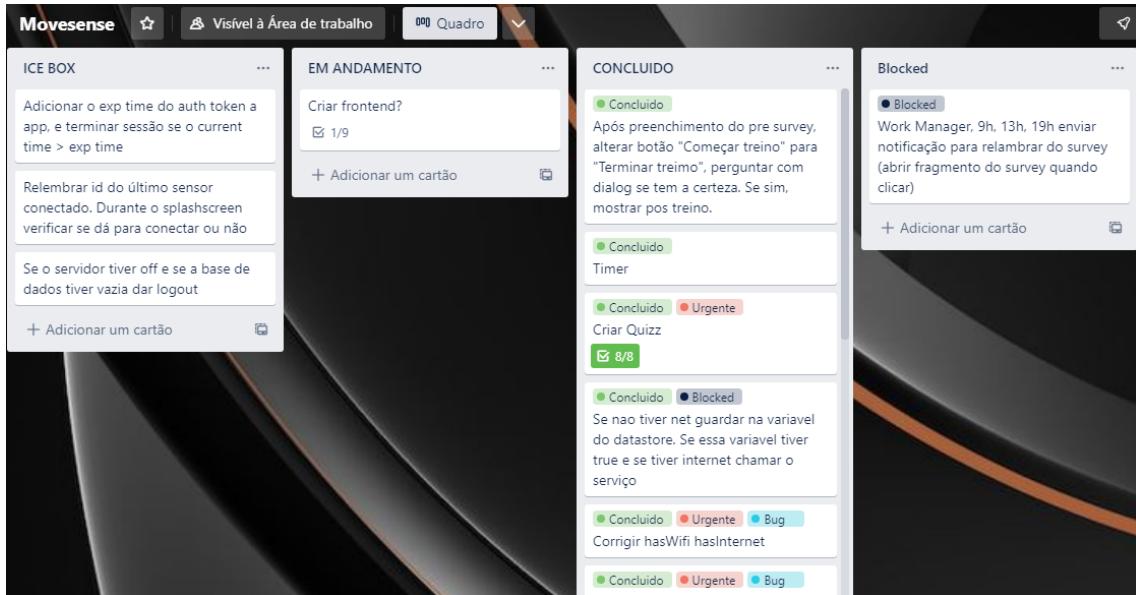


Figura 3.2: Board de tarefas no Trello.

O quadro pode ser consultado em <https://trello.com/b/MnPafydx/movesense>.

### III. Arquitetura da plataforma

Para o desenvolvimento deste projeto foram utilizados sensores *movesense*, um dispositivo *android* e um servidor na nuvem que permita armazenar todos os dados do sensor e dos questionários para mais tarde serem analisados.

O sensor comunica com o *smartphone* via do *Bluetooth Low Energy* (BLE), e este por sua vez comunicará com o servidor através de pedidos *HTTP*.

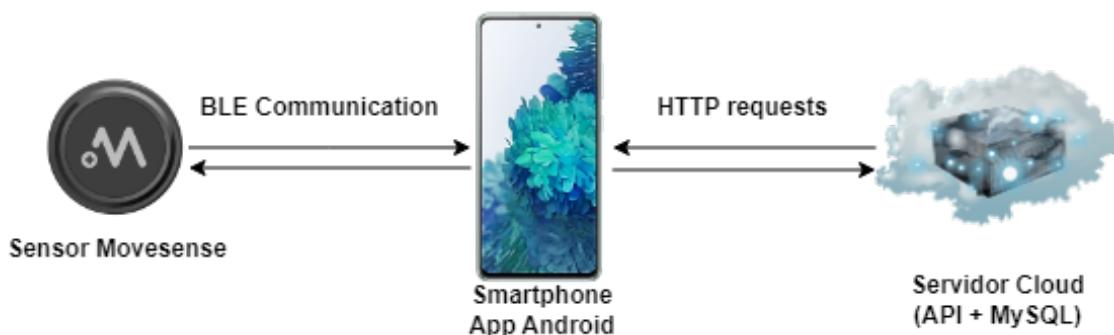


Figura 3.3: Arquitetura utilizada no desenvolvimento da plataforma.



### 3.2.2 Configuração do servidor e criação da base de dados

No servidor foi utilizado como sistema operativo o *Ubuntu 20.04* baseado em *Linux* por ser um sistema operativo leve e rápido de configurar. Numa fase inicial foi utilizado uma máquina virtual no meu computador para o desenvolvimento do projeto e mais tarde, durante os testes, foi adquirida uma *droplet*<sup>2</sup> na *DigitalOcean* e toda a infraestrutura da minha máquina foi transferida para a *droplet*.

Nestas máquinas foi necessário instalar e configurar o *MySQL* da *Oracle Corporation* que é uma base de dados relacional, e assim os dados podem ser divididos em diferentes tabelas, sem agrupar tudo numa grande unidade de armazenamento, conseguindo assim criar as relações necessárias para, por exemplo, poder ver todos os dados do sensor do acelerómetro do participante A. Para facilitar a visualização dos dados foi também instalado o *phpMyAdmin*<sup>3</sup>.

Posteriormente foi também criada a API em *Node.js* que serviu para apoiar a aplicação. (consultar 3.2.3 Desenvolvimento da API).

#### I. Configuração do servidor

No servidor foi criado um utilizador com privilégios de administrador para evitar utilizar a conta *root*.

```
sudo adduser diogo
sudo usermod -aG sudo diogo
```

**Excerto 3.1:** Comando para a criação de um novo utilizador

Para aceder remotamente ao servidor foi criado um conjunto de pares de chaves SSH pois é uma alternativa mais confiável e segura do que a autenticação por palavra-chave, uma vez que é mais eficaz contra os ataques de *brute force*<sup>4</sup> que o servidor pode sofrer[4].

```
ssh-keygen
```

**Excerto 3.2:** Comando para a criação da chave SSH

<sup>2</sup>*droplet* — "As *droplets* são máquinas virtuais baseadas em Linux executadas sobre *hardware* virtualizado. Cada *Droplet* é um novo servidor, independente ou como parte de uma infraestrutura maior baseada em nuvem." [3]

<sup>3</sup>*phpMyAdmin* - é um gestor de base de dados em *MySQL* e serve para facilitar as criações, alterações ou consultas as tabelas, criando assim uma interface gráfica de fácil uso.

<sup>4</sup>*brute force* — é um ataque de força bruta onde um *script* tenta diversas palavras-psses até encontrar a certa



## II. Instalação e configuração do *mysql* e do *phpMyAdmin*

Na base de dados *mysql*, logo após a instalação foi necessário criar um utilizador, neste caso o “diogo” com privilégios de administrador, para evitar utilizar a conta *root*.

Para atualizar o sistema operativo e instalar os pacotes necessários foram utilizados os seguintes comandos do Excerto 3.3 e no Excerto 3.5.

```
sudo apt update && sudo apt upgrade
sudo apt install apache2
sudo apt install php libapache2-mod-php
sudo apt install mysql-server mysql-client
```

**Excerto 3.3:** Comandos para a instalação do MySQL.

```
sudo mysql -u root
mysql> USE mysql;
mysql> UPDATE user SET plugin='mysql_native_password' WHERE ←
      User='root';
mysql> CREATE USER 'diogo'@'localhost' IDENTIFIED BY '{password←
      }';
GRANT ALL PRIVILEGES ON *.* TO 'diogo'@'localhost' WITH GRANT ←
      OPTION;
mysql> FLUSH PRIVILEGES;
mysql> exit;

sudo service mysql restart
```

**Excerto 3.4:** Comandos para a criação de um novo utilizador no MySQL.

```
sudo apt install phpmyadmin
sudo service apache2 restart
```

**Excerto 3.5:** Comandos para a instalação do phpMyAdmin.

Após isso, foi criada um base de dados chamada “movesense” e nela foram criadas todas as tabelas da base de dados, utilizando os comandos disponíveis no repositório em [https://github.com/diogopereira00/UMAIA\\_Movesense/tree/main/baseDados](https://github.com/diogopereira00/UMAIA_Movesense/tree/main/baseDados) no ficheiro “movesense\_estrutura.sql” e no “movesense\_dados\_surveys.sql”.



### III. Estrutura da base de dados do servidor.

No servidor, a estrutura de base de dados abrange todos os dados dos utilizadores, dos sensores *movesense* e dos questionários.

#### Dados dos sensores

Na base de dados foram criadas as tabelas *acc\_table*, *gyro\_table*, *magn\_table*, *ecg\_table*, *hr\_table* e *temp\_table* que contêm todos os dados (do acelerómetro, giroscópio, magnetómetro, eletrocardiograma, frequência cardíaca e da temperatura) dos sensores *movense*s de todos os participantes.

Estas tabelas têm o campo *user\_id* que relaciona estes dados a um participante referenciado na tabela *users*. Na tabela *users* todas as palavras-pases dos utilizadores estão armazenadas de forma encriptada e cada *user\_id* é criado automaticamente para cada utilizador ter um único *id* como chave primária (consultar 3.2.3 Desenvolvimento da API).

#### Dados dos questionários

Para os questionários foi necessário criar uma estrutura reutilizável e editável que permita a alteração, remoção ou edição de dados. Para isso foi criada uma tabela chamada *studies* que contêm toda a informação do estudo, desde o nome do estudo, a descrição, a palavra-passe de administrador, a data de início, de fim e a versão do estudo.

Cada estudo pode ter um ou mais participantes (*users*) e cada participante pode estar em um ou mais estudos. Cada estudo pode ter também um ou mais questionários, por isso foi preciso criar a tabela *surveys* que contêm o *id* do estudo, o título, a descrição e o tempo expectável de realização do questionário.

Cada questionário pode ter uma ou mais secções, sendo que por isso foi preciso criar a tabela *sections* que tem um nome o *id* do questionário. Cada secção pode ter uma ou mais questões, sendo que a tabela *questions* contêm o texto da pergunta, o tipo de pergunta (referência a tabela *question\_types* que contêm o nome do tipo de pergunta) e o *id* da secção.

Cada questão pode ter também uma ou mais opções e por isso foi criada também a tabela *options* que contêm o texto da opção, uma variável que permite saber se é de escala de likert ou não (*isLikert*), e a escala de likert em si (se não for likert iguala zero).

Cada estudo pode ter também um ou mais *researchers* e tem de pertencer a uma ou mais organizações.



Todas as questões e informações do estudo (à exceção das respostas) foram adicionados diretamente através do MySQL com o ficheiro “*movesense\_dados\_surveys.sql*”.

Os questionários de pré-treino e pós-treino podem ser consultados no *Apêndice E, na página A9* e todo o diagrama da base de dados pode ser consultada no *Apêndice D, na página A8*.

Tabela	Acções	Registros	Tipo	Agrupamento (Collation)	Tamanho	Suspenso
acc_table	★ Procurar Estrutura Pesquisar Insere Limpa Elimina	~633,686	InnoDB	utf8mb4_0900_ai_ci	19.1 MB	-
answers	★ Procurar Estrutura Pesquisar Insere Limpa Elimina	2,272	InnoDB	utf8mb4_0900_ai_ci	336.0 KB	-
ecg_table	★ Procurar Estrutura Pesquisar Insere Limpa Elimina	~458,000	InnoDB	utf8mb4_0900_ai_ci	13.0 MB	-
gyro_table	★ Procurar Estrutura Pesquisar Insere Limpa Elimina	~635,092	InnoDB	utf8mb4_0900_ai_ci	19.1 MB	-
hr_table	★ Procurar Estrutura Pesquisar Insere Limpa Elimina	6124	InnoDB	utf8mb4_0900_ai_ci	32.0 KB	-
magn_table	★ Procurar Estrutura Pesquisar Insere Limpa Elimina	~635,618	InnoDB	utf8mb4_0900_ai_ci	19.1 MB	-
options	★ Procurar Estrutura Pesquisar Insere Limpa Elimina	13	InnoDB	utf8mb4_0900_ai_ci	16.0 KB	-
organization	★ Procurar Estrutura Pesquisar Insere Limpa Elimina	1	InnoDB	utf8mb4_0900_ai_ci	16.0 KB	-
questions	★ Procurar Estrutura Pesquisar Insere Limpa Elimina	10	InnoDB	utf8mb4_0900_ai_ci	48.0 KB	-
question_options	★ Procurar Estrutura Pesquisar Insere Limpa Elimina	16	InnoDB	utf8mb4_0900_ai_ci	48.0 KB	-
question_types	★ Procurar Estrutura Pesquisar Insere Limpa Elimina	3	InnoDB	utf8mb4_0900_ai_ci	16.0 KB	-
researchers	★ Procurar Estrutura Pesquisar Insere Limpa Elimina	2	InnoDB	utf8mb4_0900_ai_ci	48.0 KB	-
sections	★ Procurar Estrutura Pesquisar Insere Limpa Elimina	3	InnoDB	utf8mb4_0900_ai_ci	32.0 KB	-
studies	★ Procurar Estrutura Pesquisar Insere Limpa Elimina	1	InnoDB	utf8mb4_0900_ai_ci	16.0 KB	-
studies_organizations	★ Procurar Estrutura Pesquisar Insere Limpa Elimina	1	InnoDB	utf8mb4_0900_ai_ci	48.0 KB	-
surveys	★ Procurar Estrutura Pesquisar Insere Limpa Elimina	2	InnoDB	utf8mb4_0900_ai_ci	32.0 KB	-
temp_table	★ Procurar Estrutura Pesquisar Insere Limpa Elimina	9,476	InnoDB	utf8mb4_0900_ai_ci	208.0 KB	-
users	★ Procurar Estrutura Pesquisar Insere Limpa Elimina	13	InnoDB	utf8mb4_0900_ai_ci	16.0 KB	-
user_studies	★ Procurar Estrutura Pesquisar Insere Limpa Elimina	9	InnoDB	utf8mb4_0900_ai_ci	48.0 KB	-
user_surveys	★ Procurar Estrutura Pesquisar Insere Limpa Elimina	524	InnoDB	utf8mb4_0900_ai_ci	208.0 KB	-

Figura 3.4: Tabelas da base de dados MySQL

### 3.2.3 Desenvolvimento da API

Para permitir o envio e a receção de dados entre a aplicação e o servidor foi necessário desenvolver uma API que permitisse efetuar a transferência de forma segura e confiável entre ambos. Para tal optou-se pelo *Node.js* que é um ambiente *open source* que permite utilizar código em *javascript* e que é frequentemente utilizada em projetos semelhantes.

Utilizou-se também o *Express.JS* que é uma das *frameworks* mais populares para o desenvolvimento de HTTP em *Noje.js*. Atua como um *middleware* pois facilita a criação e a configuração de rotas para enviar e receber *requests* entre o *front-end*(a aplicação) e a base de dados, sendo assim uma opção ideal para a criação da API.

Foram também utilizadas outras bibliotecas como a *bcrypt* para encriptar as *passwords* antes de guardar na base de dados, pois se destaca por ser uma das mais seguras da actualidade por adicionar um código aleatório ao *hash* original



[5] e a *cors* para garantir a segurança e confiabilidade dos pedidos *HTTP*. A biblioteca *uuid* foi utilizada para criar um *id* único (que se estima que demore 85 anos até que o servidor encontre uma colisão se houver mil milhões de *UUIDs* por segundo)[6] e o *jsonwebtoken* para criar e verificar *tokens* de autenticação [7]. O *loadash* foi utilizado para dividir os dados dos grandes pedidos por pequenos conjuntos de dados (*chunks*)[8] para não sobrecarregar o servidor *MySQL*.

Todas as bibliotecas foram instaladas utilizando o gestor de pacotes, o *npm* (nome da biblioteca) e para iniciar o servidor de *node.js* utilizei o comando *npm run dev*.

Para facilitar o processo criei um serviço para correr o servidor automaticamente.[9]

```
[Unit]
Description=movesense
[Service]
ExecStart=/usr/bin/npm run dev /home/UMAIA/UMAIA_Movesense/←
    backend
Restart=always
[Install]
WantedBy=multi-user.target
```

**Excerto 3.6:** Ficheriro /etc/systemd/movesense.service.

## I. Estrutura da API

Na API foram criadas duas rotas de autenticação, uma para o registo e outra para o *login*. A rota de registo recebe como parâmetros o nome de utilizador, a *password* e a confirmação da *password*. A única maneira de criar utilizadores, agora é apenas pela API, no futuro existirá uma plataforma ‘online’ que vai permitir a um administrador dos estudos criar contas de participantes e adicionar los aos estudos.

A rota de *login* recebe o nome de utilizador e a *password* e devolve, se o *login* for efetuado com sucesso, os dados do utilizador e o *access token* criado com o *jsonwebtoken*.

Foram criadas cinco rotas que permitem adicionar os dados dos sensores, a rota *addAccData* , a rota *addGyroData* , a rota *addMagnData* , a rota *addECGData* e a rota *addHRData* (para o acelerómetro, giroscópio, magnetómetro, ECG e frequência cardíaca). Estas rotas recebem como parâmetros o *token* de autenticação e uma *jsonString* com os dados dos sensores como exemplificado no



The screenshot shows the Movesense API interface. At the top, it displays the URL `movesense / Authentication / SignUp`. Below this, there are tabs for `POST`, `Params`, `Auth`, `Headers (9)`, `Body` (which is selected), `Pre-req.`, `Tests`, and `Settings`. The `Body` tab contains a JSON payload:

```
1 {"username": "user7",  
2 "password": "123456",  
3 "password_repeat": "123456"}  
4  
5
```

Below the body, the status bar shows `201 Created`, `208 ms`, and `270 B`. The response body is shown as:

```
1 {"msg": "Registered!"}  
2  
3
```

Figura 3.5: Criação de um novo utilizador diretamente pela API

excerto 3.7. Para facilitar o processo de inserção dos dados criei outra rota chamada `addAllData` que recebe todos os dados de todos os sensores ao mesmo tempo. São utilizadas *chunks* para dividir os conjuntos de dados em conjuntos menores para não sobrecarregar o *mysql*.

```
[{  
    "created": 1670031376250,  
    "id": 1,  
    "timestamp": 13134,  
    "userID": "fb6b5d3b-3bb8-46ab-b644-233e206ebfd0",  
    "x": "-0.0047858161851763725",  
    "y": "-0.11485958844423294",  
    "z": "-9.84203052520752"  
}, {  
    "created": 1670031376316,  
    "id": 2,  
    "timestamp": 13208,  
    "userID": "fb6b5d3b-3bb8-46ab-b644-233e206ebfd0",  
    "x": "0.02632198855280876",  
    "y": "-0.11725249886512756",  
    "z": "-9.789386749267578"
```



}]

**Exerto 3.7:** Exemplo jsonString acelerometro.

Foram também criadas as rotas `addUserSurveys`, para adicionar um questionário de um utilizador, a rota `studies/allInfo/user_id` para ver todas as informações de todos os estudos dos utilizadores (estudos, questionários, secções, questões e opções), a rota `options` para ver todas as opções, a rota `question_options` para receber todas as questões para cada pergunta, a rota `question_types` para receber os tipos de perguntas (`linkert`, escolha múltipla, etc...) e a rota `studies/:studyID/version` que recebe como parâmetro o id do estudo e retorna a sua versão se existir.

Todas as rotas necessitam do `token` de autenticação e todas ou retornam o resultado, ou um erro e podem ser consultadas no Apêndice F Referência técnica da API.

### 3.2.4 Desenvolvimento da aplicação Android

A aplicação foi desenvolvida em *Android* e foi construída utilizando o padrão de arquitetura MVVM baseado no *Model*, *View* e *ViewModel* e separa a lógica de negócios da camada de dados, tornando o código mais fácil de ler e manter.

- *Model*: o modelo que representa a camada de dados da aplicação.
- *View*: é responsável por exibir toda a interface do utilizador
- *ViewModel*: é a “ponte” entre o *model* e a *view*.

Nesta aplicação as atividades e os fragmentos comunicam com a *viewmodel* que acede aos repositórios que contêm os modelos da base de dados *room* e a *remote data source* que comunica com a API através do *retrofit*. Como demonstrado na Figura 3.6

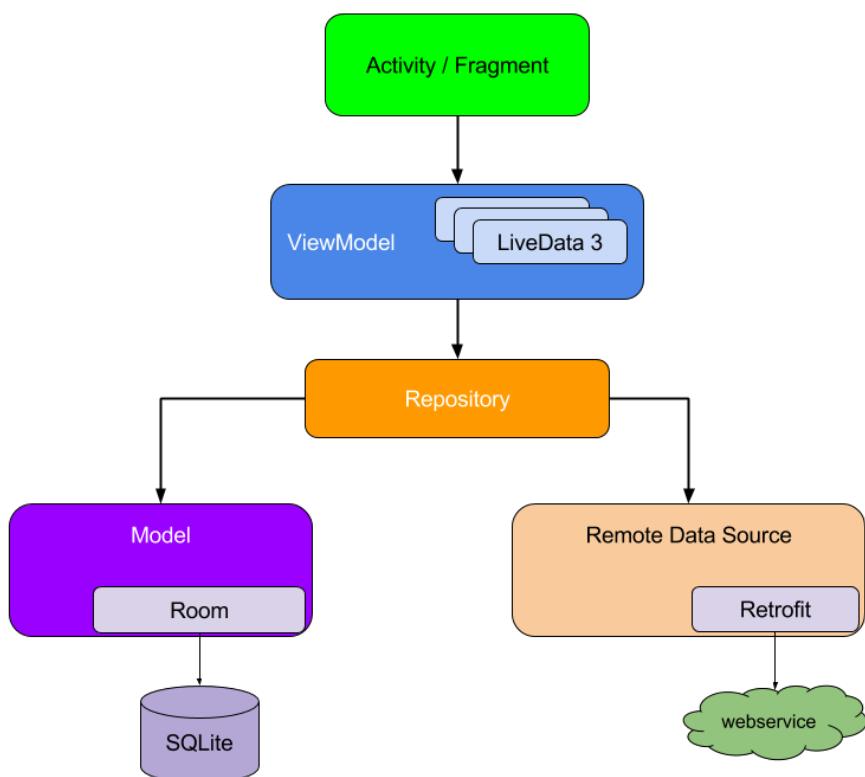


Figura 3.6: Arquitetura utilizada na aplicação



Foi utilizada a biblioteca *Movesense Mobile Library* (MDS) para lidar com a comunicação através da *Whiteboard* aos sensores Movesense com a biblioteca *RxAndroidBle* que fornece uma maneira fácil e eficaz de interagir com os dispositivos *Bluetooth Low Energy* (BLE), assim é possível conectar aos dispositivos BLE e recolher informações dos mesmos.

```
private fun getBleClient(): RxBleClient? {
    // Init RxAndroidBle (Ble helper library) if not yet ↵
    initialized
    if (mBleClient == null) {
        mBleClient = RxBleClient.create(this)
    }
    return mBleClient
}
val bleDevice = getBleClient()!!.getBleDevice(device.macAddress←
)
```

**Excerto 3.8:** Exemplo de RxAndroidBle.

Com o *macAddress* do dispositivo é possível conectar com o sensor Movesense utilizando a MDS como mostrado no Excerto I.1 no Apêndice I.

```
var subscription: MdsSubscription = mds.subscribe(
    "suunto://MDS/EventListener",
    "{\"Uri\": \"\" + deviceSerialNumber.toString() + "/" + ←
        uriToSubscribeTo.toString() + "\",",
    callback
) // MdsNotificationListener callback class
```

**Excerto 3.9:** Exemplo de subscrição MDS.



### A. *Splash Screen*

O *Splash Screen*, ou em português o ecrã inicial, é o primeiro ecrã que aparece quando a aplicação arranca. Neste ecrã é verificado, através da *datastore user preferences*[10] se existe alguma sessão ativa, se houver são procuradas novas atualizações da base de dados (consultar o tópico C Transferencia dos questionários). Se tiver sessão ativa o utilizador é reencaminhado para o ecrã de *scan* de dispositivos movesense ou para o ecrã inicial se tiver uma conexão ativa, caso contrário é reencaminhado para o ecrã de autenticação.

É também verificado através da *user preferences* se o utilizador consentiu com os termos, verificando os dados das definições dos sensores (consultar Figura H.14) no apêndice H que por defeito estão todos ligados a exceção da live data.

### B. Ecrã de Autenticação

Este ecrã serve para o utilizador poder iniciar a sua sessão com as credenciais atribuídas pelo administrador do estudo. Caso seja a primeira vez que o utilizador inicia sessão no dispositivo, os questionários devem ser transferidos e armazenados na base de dados. (Consultar o tópico C Transferencia dos questionários).

Após o início de sessão o utilizador deve ser reencaminhado para o ecrã de consentimento (se ainda não tiver consentido com os termos do estudo) ou para o ecrã de *scan* de dispositivos movesense.

### C. Transferencia dos questionários

Para transferir ou atualizar os questionários na base de dados local no *Splash Screen* ou no *Ecrã de Autenticação* a aplicação necessita de ter conexão a internet, e se não houver *wifi* o utilizador deve ser alertado para evitar o uso de dados móveis sendo mostrado uma *dialog*<sup>5</sup>. Após a decisão de prosseguir com dados moveis ou a de se conectar a uma rede wifi os questionários devem ser transferidos e armazenados na base de dados.

No caso dos *updates*, é consultado sempre que existir uma conexão a internet o número da versão do estudo se coincidir é porque não há *updates*, mas se o número for inferior ao da base de dados do servidor então existe um *update* e os dados devem ser substituídos localmente.

---

<sup>5</sup>dialog- é uma janela *popup*



#### D. Ecrã de Consentimento

Este ecrã é apresentado logo após o primeiro início de sessão do utilizador e permite confirmar que o utilizador aceita os termos e condições da aplicação. É necessário o utilizador aceitar os termos caso contrario não consegue prosseguir na aplicação. Após consentir, a variável da *user preferences* “*consent*”, que por defeito é falsa, é atualizada para verdade.

#### E. Scan de dispositivos Movesense

Este ecrã serve para o utilizador poder-se conectar-se a um sensor sendo apresentado sempre que não existe uma conexão ativa com o sensor. Tem um botão que permite fazer uma pesquisa pelos dispositivos na proximidade utilizando a biblioteca *RxAndroidBle*. Para tal o utilizador deve aceite as permissões (que podem variar conforme a versão *android* do dispositivo) como do *bluetooth*, localização e dispositivos na proximidade e a de notificações em *Androids* mais recentes.

#### F. Ecrã principal

No ecrã principal da aplicação, o utilizador tem uma *navbar* que lhe permite alternar (entre fragmentos) do ecrã principal para o ecrã das definições. Tem também um cronómetro, que demonstrará o tempo da atividade e um botão que lhe irá permitir iniciar ou terminar a atividade.

#### G. Ecrã definições

Neste ecrã o utilizador pode terminar sessão e existe também uma aba onde o administrador do estudo pode ativar e/ou desativar a recolha de dados de determinados sensores, bem como ativar e/ou desativar o envio em direto dos dados para o servidor, podendo ser utilizada a rede móvel.

#### H. Questionários

Para facilitar o desenvolvimento do sistema dos questionários na aplicação foi utilizado, como referido anteriormente, a biblioteca *SurveyKit*[11] que permite criar as interfaces dos questionários mais rapidamente, para tal a aplicação consulta os dados da base de dados local, que deverá estar sincronizada com a do servidor, e recolhe todos os dados dos questionários, como título, descrição, tempo estimado, as suas secções, as questões e as opções de cada questão.



## I. Recolha de dados

A recolha dos dados dos sensores é iniciada sempre que o participante inicia uma atividade (e termina o questionário de pré-treino). Para tal foi criado um *lifecycle service* para permitir efetuar tarefas em segundo plano, e assim a aplicação mesmo estando minimizada ou com o telemóvel bloqueado continuará a recolher os dados necessários. Neste serviço (*MovesenseService*) os dados são recolhidos e a atividade do ecrã inicial é atualiza a interface com a utilização de *observers* que consultam o estado das respetivas variáveis. Neste mesmo serviço existe outro *timer* que a cada 2 minutos tenta enviar os dados recolhidos para o servidor se houver *wifi* para não sobrecarregar o servidor. Assim que o servidor retorna uma mensagem a dizer que os dados foram adicionados, os mesmos são eliminados da base de dados local.

Quando o serviço é iniciado é mostrada a notificação *foreground*<sup>6</sup> que informa que os dados estão a ser recolhidos. A mesma notificação pode ser alterada se a conexão ao sensor for perdida durante a atividade.

## J. Envio dos dados para o servidor

Para enviar os dados para o servidor criei outro serviço que também é executado em segundo plano para poder enviar os dados mesmo que a aplicação seja minimizada ou que o telemóvel bloqueie. O serviço recolhe todos os dados dos sensores e das respostas dos questionários e envia tudo para o servidor através da biblioteca *retrofit* que simplifica o envio de pedidos *HTTP*. Este serviço é chamado assim que a aplicação é iniciada, quando uma atividade é iniciada, a cada dois minutos durante a atividade e quando a atividade é terminada, isto tudo se houver *wifi* disponível.

---

<sup>6</sup>foreground- é um tipo de notificação demonstrada mesmo quando a aplicação se encontra aberta, é uma notificação persistente que só desaparece quando o é cancelada ou quando a aplicação é terminada pelo gestor de aplicações.



Importante realçar que todos os pedidos feitos através da biblioteca *retrofit* passam por uma classe abstrata que tem como principal objetivo diminuir o código e que retorna um objeto “Resource” que contem o resultado do pedido da API ou o erro.[12]

```
abstract class BaseRepository {

    suspend fun <T> safeApiCall(
        apiCall: suspend () -> T
    ): Resource<T> {
        return withContext(Dispatchers.IO) {
            try {
                Resource.Success(apiCall.invoke())
            } catch (throwable: Throwable) {
                when (throwable) {
                    is HttpException -> {
                        Resource.Failure(false, throwable.code(),
                            throwable.response()?.errorBody())
                    }
                    else -> {
                        Resource.Failure(true, null, null)
                    }
                }
            }
        }
    }
}
```

**Excerto 3.10:** Classe abstrata safeApiCall.



## K. Base de Dados da aplicação

A base de dados local da aplicação é semelhante a do servidor, é também uma base de dados relacional, mas é construída na biblioteca *Room*, desenvolvida pela *Google* e é construída sobre a base de dados *SQLite*. Esta contém praticamente todas as tabelas da base de dados do servidor, exceto a tabela de *users*. Todos os dados dos utilizadores, incluindo o *token* de autenticação, o *user\_id* e as preferências dos sensores são armazenados no *DataStore* que é uma maneira rápida e eficaz de guardar as preferências do utilizador na aplicação. O *DataStore* armazena os dados em pares de chave-valor[10].

O Exerto 3.11 da-nos um exemplo de como guardar dados no *DataStore* na aplicação.

```
class UserPreferences(context: Context) {
    private val applicationContext = context.applicationContext

    //Observador do valor do token
    val token: Flow<String?>
        get() = applicationContext.dataStore.data.map { ←
            preferences ->
            preferences[KEY_AUTH]
        }

    //Função para alterar o estado do token
    suspend fun saveAuthToken(token: String) {
        applicationContext.dataStore.edit { preferences ->
            preferences[KEY_AUTH] = token
        }
    }
}
```

**Exerto 3.11:** Guardar o token de autenticação do utilizador.



### 3.2.5 Testes e Validações

Para garantir que a aplicação está pronta a servir o seu propósito e atende aos requisitos de qualidade e funcionalidade, a mesma teve de passar por um conjunto de testes e validações para ser aprovada.

#### Testes Funcionais

Estes testes permitem verificar se todas as funcionalidades funcionam corretamente e devem ser realizados em diferentes cenários (com *wifi*, sem *wifi*, com rede móvel, etc...) e em diferentes *smartphones* para garantir assim que a aplicação consegue lidar com todos os casos diferentes e em diferentes telemóveis. Para tal a mesma deve ser testada por um conjunto de pessoas externas ao seu desenvolvimento.

Para validar a aplicação durante os testes foram estabelecidas algumas metas que deveriam ser atingidas, e para isso a aplicação deve:

- Funcionar corretamente em diferentes dispositivos.
- Ser utilizada pelo menos uma vez por semana, mas preferencialmente mais de duas vezes por semana.
- Ser capaz de se conectar ao sensor sem problemas.
- Conseguir permitir a realização dos questionários sem problemas.
- Conseguir realizar uma atividade sem problemas.

Para isto foi desenvolvido um questionário no *Microsoft Forms* que foi utilizado para recolher o *feedback* dos utilizadores que testaram a aplicação. As perguntas podem ser consultadas no Apêndice E Questionários.

#### Testes de desempenho

Estes testes são feitos para verificar se a aplicação consegue lidar com grande volume de dados sem comprometer a sua capacidade de resposta. Nestes testes foram verificados durante o desenvolvimento da aplicação o tempo médio da realização de cada atividade na aplicação e no servidor. .



### Testes de usabilidade

Estes testes permitem garantir que a interface do utilizador é fácil de usar. Para isso foi desenvolvido um teste prático onde foram cronometradas as seguintes tarefas.

- Efetuar o login
- Consentir com a recolha de dados
- Conectar com o sensor
- Ir até as definições
- Começar uma atividade
- Responder ao questionário pré-treino
- Terminar atividade
- Responder ao questionário pós-treino



## 4. RESULTADOS E DISCUSSÃO

Neste capítulo são apresentados os resultados obtidos desta plataforma de recolha dos dados sensoriais dos sensores Movesense.

### 4.1 Levantamento de requisitos e planeamento

Neste projeto foi criada uma lista de requisitos funcionais e não funcionais que permitiu elaborar um planeamento muito mais ágil e eficaz, o que facilitou o desenvolvimento do projeto.

Os requisitos podem ser consultados no *Apêndice B, na página A2*.

### 4.2 Servidor e da base de dados

No servidor, tanto a API como o servidor de base de dados *MySQL* são executados para estarem sempre a “escuta” de novos dados.

```
diogo@umaiaMovesense:~$ sudo systemctl status movesense.service
[sudo] password for diogo:
● movesense.service - Movesense Node.js app
    Loaded: loaded (/etc/systemd/system/movesense.service; disabled; vendor pr>
    Active: active (running) since Sun 2023-04-02 13:59:02 UTC; 4 days ago
      Docs: https://www.example.com
   Main PID: 2070 (npm)
     Tasks: 30 (limit: 2338)
    Memory: 88.9M
   CGroup: /system.slice/movesense.service
           └─2070 npm
                 ├─2087 sh -c nodemon index.js
                 ├─2088 node /home/diogo/UMAIA_Movesense/backend/node_modules/.bin/>
                 └─2101 /usr/bin/node index.js
```

Figura 4.1: API *movesense.service* em execução

```
diogo@umaiaMovesense:~$ sudo systemctl status mysql.service
● mysql.service - MySQL Community Server
    Loaded: loaded (/lib/systemd/system/mysql.service; enabled; vendor preset:>
    Active: active (running) since Sun 2023-04-02 13:26:02 UTC; 4 days ago
  Main PID: 821 (mysqld)
    Status: "Server is operational"
     Tasks: 47 (limit: 2338)
    Memory: 724.3M
   CGroup: /system.slice/mysql.service
           └─821 /usr/sbin/mysqld
```

Figura 4.2: Serviço *mysql.service* em execução



### 4.3 API no servidor

Neste projeto foi desenvolvida uma API capaz de fornecer os recursos necessários a aplicação e de armazenar os dados recebidos na aplicação.

A documentação da API está disponível no Apêndice F, na página A12 e o seu código-fonte está disponível no repositório git em [https://github.com/diogopereira00/UMAIA\\_Movesense/tree/main/backend](https://github.com/diogopereira00/UMAIA_Movesense/tree/main/backend).

Para correr o código-fonte da API deve correr os comandos referenciados no Excerto 4.1 no diretório `/UMAIA_Movesense/backend/`.

```
npm install  
npm run dev
```

**Excerto 4.1:** Comandos para instalar os pacotes necessários e correr a API

#### I. Recolha de Dados

Esta API permitiu realizar a recolha dos dados da aplicação e, até ao momento de redação deste relatório, foram recolhidos mais de 700 mil registos de dados dos sensores do acelerómetro, do giroscópio e do magnetómetro, mais de 350 mil registos dos dados do eletrocardiograma, mais de oito mil registos de frequência cardíaca, e mais de cinco mil dados de temperatura. Estes dois tiveram menos registos, pois a maioria dos dados foram recolhidos utilizando a pulseira do movesense que não permite fazer uma recolha precisa da frequência cardíaca, para tal deveria ser utilizada a fita do peito que essa sim é indicada para realizar a recolha de dados de frequência cardíaca e eletrocardiograma enquanto a pulseira é mais indicada para os dados de movimento (acelerómetro, giroscópio e magnetómetro).

Foram também recolhidos cerca de quinhentos questionários e cerca de três mil perguntas respondidas, sendo que uma parte foi durante a correção dos bugs e do envio dos dados para o servidor onde, por vezes, falhavam algumas perguntas. Este problema foi resolvido e até ao momento está tudo a funcionar como esperado.

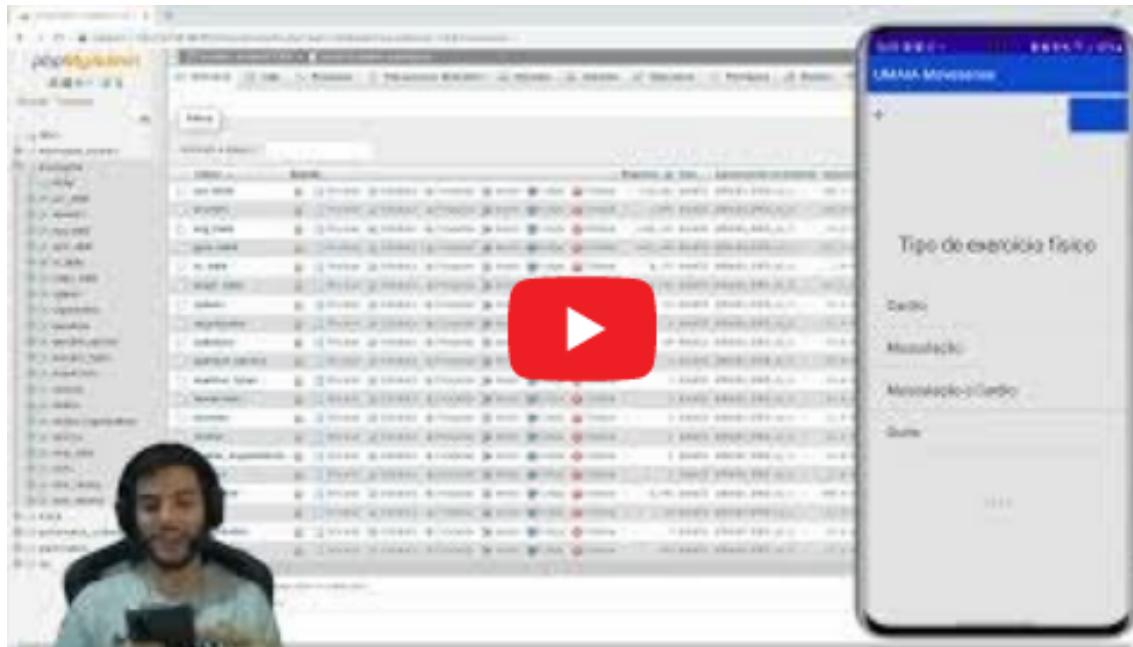


#### 4.4 Aplicação Android

Com este trabalho foi desenvolvida uma aplicação em *Android* totalmente capaz de satisfazer os requisitos propostos.

O diagrama de atividades da figura G.1 do apêndice G demonstra-nos todos os passos feitos na aplicação desde o *Splash Screen* até ao *Ecrã principal*. Já no ecrã principal o utilizador deve decidir de começa/termina uma atividade, ou se vai às definições e altera algum parâmetro do sensor, ou se termina de sessão e sai da aplicação.

Foi gravado um pequeno vídeo que pretende demonstrar a aplicação em funcionamento, que pode ser consultado em <https://www.youtube.com/watch?v=NMhC6SrpsIo>.



**Figura 4.3:** Vídeo de demonstração da aplicação

Todas as interfaces criadas podem ser consultas no Apêndice H Interfaces da aplicação na página A14



## 4.5 Testes e Validações

Esta foi uma parte importante durante o desenvolvimento deste projeto, pois foi possível identificar e corrigir problemas críticos tanto na aplicação como no servidor.

### Testes funcionais

Durante os testes funcionais, realizados com a ajuda da professora Isabel Bastos e de algumas alunas que testaram a aplicação durante 3 semanas, foram identificados diversos *bugs* tanto a nível da criação dos questionários na aplicação, bem como no envio dos dados para o servidor. Em alguns dispositivos os dados dos questionários não eram bem carregados e por isso foi necessário alterar como os dados eram recolhidos da base de dados local *Room*.

1. Que smartphone Android utilizou durante os testes?



2. Com que frequência utilizou a aplicação?

● Menos de uma vez por semana	1
● Uma vez por semana	1
● Duas vezes por semana	3
● Três vezes por semana	1
● Mais de três vezes por semana	0



Figura 4.4: Respostas 1 e 2 do formulario

Após a correção dos *bugs* identificados foram realizados novamente os testes funcionais onde geralmente correu tudo bem. A aplicação foi testada em diversos *smartphones* e já não houve problemas na criação dos questionários, bem como também não houveram grandes problemas durante a conexão com o sensor nem com a recolha dos dados durante a atividade física.

3. Durante a sua utilização, com que frequência encontrou problemas a conectar com o sensor Movesense?

● Nunca	3
● Raramente	3
● Muitas vezes	0
● Sempre	0



4. Durante a sua utilização, com que frequência encontrou problemas com os questionários?

● Nunca	4
● Raramente	1
● Muitas vezes	1
● Sempre	0



5. Durante a sua utilização, com que frequência encontrou problemas durante a atividade?

● Nunca	4
● Raramente	2
● Muitas vezes	0
● Sempre	0



6. Durante a sua utilização teve mais algum problema?

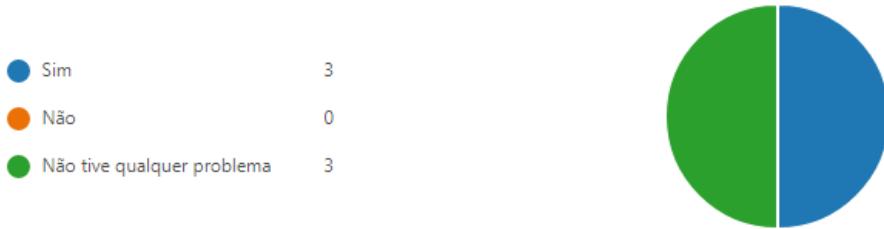
● Sim	0
● Não	6



**Figura 4.5:** Respostas 3,4,5 e 6 do formulario

Foram identificadas algumas melhorias durante os testes, como, por exemplo, alertar o utilizador que não tem o *bluetooth* ligado, coisa que atualmente a aplicação não faz.

8. Se encontrou problemas durante os testes, os mesmos foram resolvidos?



9. Existe alguma funcionalidade que gostava de ver adicionada ou modificada na aplicação?

1  
Respostas

## Respostas Mais Recentes

**Figura 4.6:** Respostas 8 e 9 do formulario

As respostas dos formulários podem ser consultadas na totalidade no repositório em [https://github.com/diogopereira00/UMAIA\\_Movesense/blob/main/docs/Respostas%20UMAIA%20Movesense.pdf](https://github.com/diogopereira00/UMAIA_Movesense/blob/main/docs/Respostas%20UMAIA%20Movesense.pdf).

## Testes desempenho

Durante os testes funcionais foram também identificados alguns problemas relativamente ao envio dos dados para o servidor, porque se fossem muitos dados carregados de uma só vez sobrecarregava o servidor de *mysql* e acabava por *crashar* o mesmo. Por isso foi necessário alterar algum código na API e dividir os grandes conjuntos de dados em *chunks* mais pequenas e tratar de cada *chunk* de forma assíncrona.

Também se reparou que o tempo de *timeout* do pedido na aplicação (ou seja, o tempo de espera por uma resposta) era demasiado pequeno e por isso os dados não eram eliminados da base de dados local quando carregados, pois a resposta “ok” não chegava. Aumentando o *timeout* da aplicação, o problema ficou resolvido e os dados a medida que eram carregados começaram a ser eliminados da base de dados local.

## Testes de usabilidade

Os testes de usabilidade foram feitos com três participantes com idade diferentes e demonstram que a aplicação é fácil e eficaz de utilizar. Contudo, como melhoria é necessário acrescentar a informação que permite informar o utilizador que não tem o *bluetooth* ligado.



A tabela 4.1 contém a média de tempo dos três participantes a realizar as tarefas.

<b>Tarefa</b>	<b>Tempo</b>
Efetuar o login	22 segundos
Consentir com a recolha de dados	5 segundos
Conectar com o sensor	33 segundos
Ir até as definições	2 segundos
Começar uma atividade	3 segundos
Responder ao questionário pré-treino	33 segundos
Terminar atividade	5 segundos
Responder ao questionário pós-treino	66 segundos

**Tabela 4.1:** Teste de usabilidade — Media dos participantes

Os resultados de cada participante pode ser consultado no apêndice E na página A11.



## 5. CONCLUSÕES

O facto de ter escolhido a opção de “Projeto de Informática” para finalizar a minha licenciatura, permitiu-me colocar em prática todos os conhecimentos adquiridos ao longo deste ciclo de estudos.

### 5.1 Objetivos concretizados

Ao longo deste projeto foram concretizados todos os objetivos propostos com sucesso.

- ✓ Efetuar um levantamento e planeamento do projeto
- ✓ Configurar um servidor e criar uma base de dados estruturável e escalável para armazenar os dados dos sensores, dos participantes e dos questionários.
- ✓ Desenvolver uma API que permita o armazenamento dos dados da aplicação no servidor
- ✓ Desenvolver uma aplicação de fácil uso em Android que permita a coleta dos dados sensoriais e dos questionários dos participantes
- ✓ Realizar os testes e validações necessárias para comprovar o bom funcionamento da plataforma.

### 5.2 Problemas e dificuldades

No entanto, durante o desenvolvimento deste projeto surgiram alguns problemas e dificuldades em desenvolver esta plataforma, pois nunca havia feito uma aplicação do género onde utilizasse tantas tecnologias diferentes e que não tinha muita prática.

#### 5.2.1 Problemas

Durante o desenvolvimento do servidor houve alguns problemas no primeiro servidor, que sofreu alguns ataques de *brute force* e por isso toda a infraestrutura teve de ser movida para outro servidor, desta vez com uma *ssh key* como referido no subcapítulo da página 11. Outro problema deveu-se ao o número reduzido de participantes nos testes, pois esta aplicação só ganhou um contexto na fase final de desenvolvimento, por isso foram feitos alguns testes, mas podem existir ainda pequenos *bugs* que não foram identificados. No entanto, estão previstos mais testes para o futuro.



### 5.2.2 Dificuldades

Existiram também diversas dificuldades ao longo deste projeto que se deveram principalmente a inexperiência em algumas tecnologias.

Houve dificuldades principalmente em:

- Criar um sistema de questionários
- Guardar os dados na base de dados local *Room*
- Utilizar a arquitetura MVVM
- Sincronizar os dados da aplicação com os do servidor.

### 5.3 Trabalho Futuro

Apesar de achar que tenha desenvolvido uma boa plataforma de recolha de dados, com tudo funcional e com algum nível de perfeccionismo, há sempre espaço para melhorias e para trabalho futuro como:

- Testar a aplicação com mais utilizadores
- Desenvolver uma aplicação para *iOS*
- Desenvolver uma plataforma online para facilitar a visualização e a análise dos dados
- Melhorar a interface da aplicação.
- Desenvolver uma biblioteca própria para a apresentação dos questionários na aplicação, uma vez que a *SurveyKit Tool* limita um pouco o desenvolvimento do aspeto gráfico da aplicação.
- Otimizar algumas funções.
- Alertar o utilizador caso não tenha o *bluetooth* ativo quando procura dispositivos.

### 5.4 Considerações finais

Este foi um projeto bastante enriquecedor, no sentido em que aprendi bastante sobre tecnologias que nunca explorara enquanto consegui completar todos os objetivos propostos para o trabalho. Foram aplicados conhecimentos adquiridos ao longo deste ciclo de estudos bem como conhecimentos que foram adquiridos de forma autónoma.



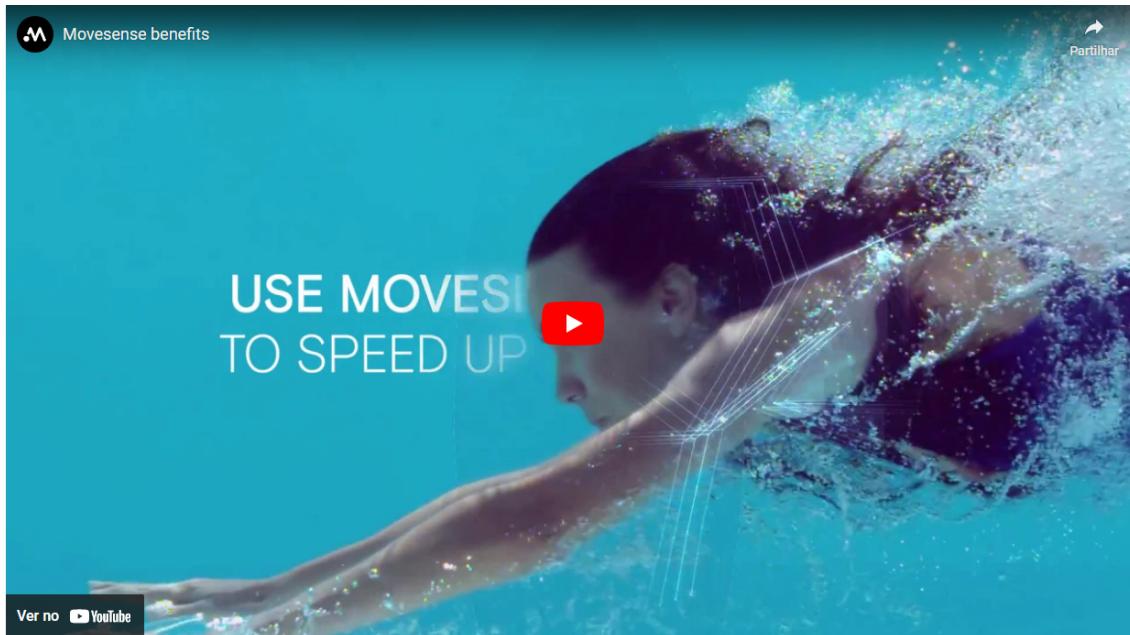
## REFERÊNCIAS

- [1] Movesense. *Learn how companies have used the Movesense platform to build great consumer products that are changing the game.* <https://www.movesense.com/showcase/>.
- [2] Movesense. *Mobile software development overview.* [https://www.movesense.com/docs/mobile/mobile\\_sw\\_overview/](https://www.movesense.com/docs/mobile/mobile_sw_overview/).
- [3] DigitalOcean. *What are DigitalOcean Droplets?* <https://www.digitalocean.com/products/droplets>.
- [4] DigitalOcean. *Como configurar a autenticação baseada em chaves SSH em um servidor Linux.* <https://www.digitalocean.com/community/tutorials/how-to-configure-ssh-key-based-authentication-on-a-linux-server-pt>.
- [5] Ronaldo B. *Criptografia em Node JS com Bcrypt.* <https://hcode.com.br/blog/criptografia-em-node-js-com-a-lib-bcrypt>. 2022.
- [6] Ukeje Goodness. *Understanding UUIDs in Node.js.* <https://blog.logrocket.com/uuids-node-js/>. 2022.
- [7] Auth0. *jsonwebtoken library.* <https://www.npmjs.com/package/jsonwebtoken>. 2023.
- [8] TARuN. *Lodash \_chunk() Method.* [https://www.geeksforgeeks.org/lodash\\_-\\_chunk-method/](https://www.geeksforgeeks.org/lodash_-_chunk-method/). 2022.
- [9] Alexey Baryshnikov. *How to start Node.js projects as service (without Docker).* <https://medium.com/@alexeybaryshnikov/how-to-start-node-js-projects-as-service-without-docker-8a04f8a8b469>. 2018.
- [10] Florina Muntenescu. *Working with Preferences DataStore.* <https://developer.android.com/codelabs/android-preferences-datastore#0>. 2022.
- [11] QuickBirdEng. *SurveyKit: Create beautiful surveys on Android (inspired by ResearchKit Surveys on iOS).* <https://github.com/QuickBirdEng/SurveyKit>.
- [12] Ashutosh Wahane. *How to handle error response return from server api in kotlin+retrofit+coroutine+mvvm.* <https://stackoverflow.com/questions/68917967/how-to-handle-error-response-return-from-server-api-in-kotlinretrofitcoroutine>.



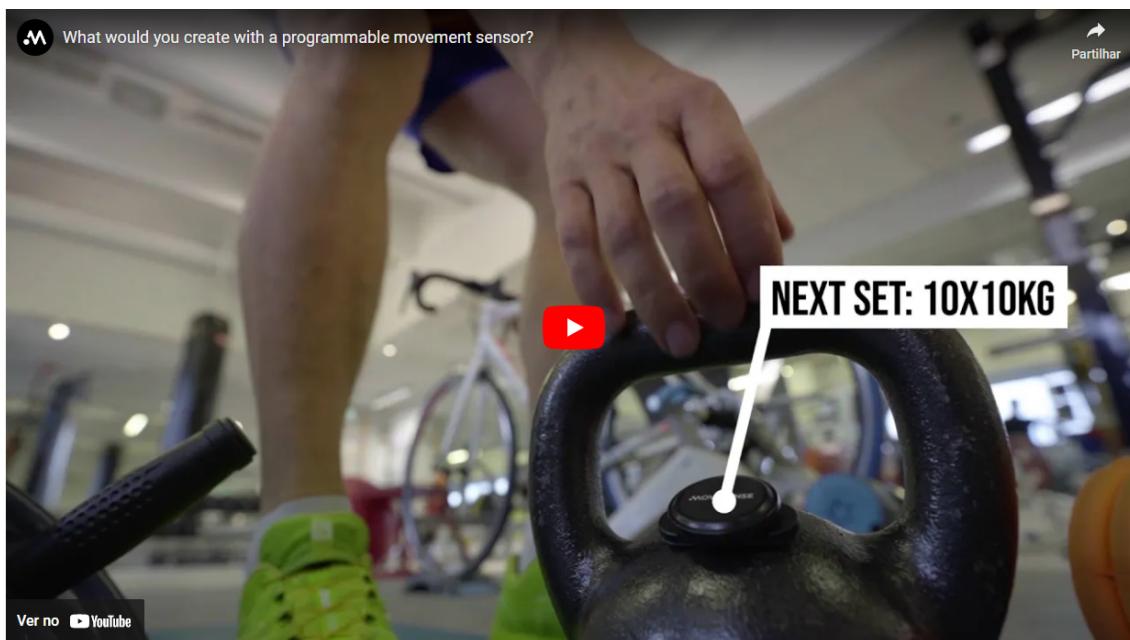
## Apêndice A

### Vídeos de demonstração Movesense



**Figura A.1:** Movesense benefits

[https://www.youtube.com/watch?v=pyZ9rFE\\_9A8](https://www.youtube.com/watch?v=pyZ9rFE_9A8)



**Figura A.2:** What would you create with a programmable movement sensor?

<https://www.youtube.com/watch?v=XIn2Rl4qiz0>



## Apêndice B

### Lista de Requisitos Simplificada

#### Requisitos funcionais

Requisito	Descrição do Requisito	Prioridade
RF 01	Deve ser possível o utilizador autenticar-se na aplicação com as credenciais fornecidas pelos administrador do estudo. Estas credenciais são constituídas pelo nome de utilizador e pela palavra-passe.	Essencial
RF 02	Logo após a autenticação do utilizador todos os estudos e questionários devem ser transferidos, pelo <i>wifi</i> , e armazenados no telemóvel.	Essencial
RF 03	Cada vez que utilizador abre a aplicação, se tiver com o <i>wifi</i> ligado e se tiver autenticado, deve ser feita uma pesquisa por novas versões dos questionários, caso haja uma nova versão a mesma deve ser transferida e armazenada na base de dados local.	Essencial
RF 04	Após a primeira autenticação o utilizador deverá ler completamente a declaração de consentimento e aceitar a mesma para prosseguir com o estudo na aplicação.	Essencial
RF 05	A aplicação deve pedir ao utilizador as permissões necessárias para o seu bom funcionamento. Neste caso do <i>bluetooth</i> , localização e das notificações.	Essencial
RF 06	O utilizador deverá poder pesquisar os sensores Movesense disponíveis e próximos.	Essencial
RF 07	O utilizador deverá poder conectar-se a um dispositivo Movesense presente na lista de dispositivos disponíveis e próximos. Deve receber <i>feedback</i> da aplicação ao longo do processo.	Essencial
RF 08	Após se conectar com o sensor o utilizador deve ser redirecionado para a página principal. Aqui deve existir um cronometro e um botão que permita começar a atividade.	Essencial
RF 09	Na página principal deve existir uma <i>navbar</i> que de para ir as definições da aplicação ou para a página principal.	Importante



Requisito	Descrição do Requisito	Prioridade
RF 10	Deve ser possível dar a opção ao administrador do estudo para ativar e/ou desativar a recolha de determinados sensores.	Importante
RF 11	Deve ser possível dar a opção ao administrador do estudo para alterar a frequência de recolha de dados do sensor.	Desejável
RF 12	Deve existir uma opção que de para ativar o envio dos dados para o servidor em direto, podendo assim ser utilizada rede móvel para esse fim.	Desejável
RF 13	Ao clicar no botão "Começar atividade" o questionário pré treino deve ser exibido.	Importante
RF 14	Ao finalizar o questionário pré treino o utilizador deverá ser novamente reencaminhado para a página principal. Desta vez o cronometro deve começar a contar e o botão que era "Começar atividade" deve ser alterado para "Terminar atividade".	Importante
RF 15	Durante a atividade, o utilizador deve ser notificado que os dados estão a ser recolhidos através de uma notificação.	Importante
RF 16	Durante a atividade, os dados devem ser recolhidos mesmo com a aplicação minimizada ou com o telemóvel bloqueado.	Importante
RF 17	Durante a atividade, se houver wifi, os dados devem ser enviados para o servidor de 2 em 2 minutos.	Importante
RF 18	Ao clicar no botão "Terminar atividade" o questionário pós treino deve ser exibido.	Importante
RF 19	Ao finalizar o questionário pós treino o utilizador deverá ser novamente reencaminhado para a página principal. Desta vez o cronometro deve ser reiniciado e o botão que era "Terminar atividade" deve ser alterado para "Começar atividade".	Importante
RF 20	O envio dos dados da aplicação para o servidor deve ser assegurado sempre que houver wifi. Preferencialmente ao abrir a aplicação, a iniciar uma atividade e a terminar.	Importante
RF 21	O utilizador deve ser alertado para atualizações dos questionários	Desejável
RF 22	O utilizador deve ser alertado caso não tenha wifi	Desejável

Tabela B.1: Requisitos Funcionais



## Requisitos não funcionais

Requisito	Descrição do Requisito	Prioridade
RNF 01	Todas as palavras-pases devem ser encriptadas quando guardadas na base de dados <i>mysql</i> .	Essencial
RNF 02	A API deve ser criada para permitir o envio dos dados usando sessões encripto-grafadas <i>JWT</i> . Apenas usuarios autenticados conseguem enviar ou receber dados do servidor.	Essencial
RNF 03	A aplicação deve permitir o armazenamento <i>offline</i> dos dados dos questionários e dos sensores e a sua sincronização posterior.	Essencial
RNF 04	A aplicação deve ser rápida e compatível com dispositivos mais antigo e/ou com menor largura de banda.	Essencial
RNF 04	A aplicação deve evitar o uso da rede móvel a menos que seja pedido pelo utilizador.	Essencial

**Tabela B.2:** Requisitos Não Funcionais



## Apêndice C

### Distribuição das tarefas pelas *Sprints*

#### Sprint 1 (3 de outubro de 2023 - 14 de outubro de 2023)

Tarefa	Story Points
Escolher as tecnologias necessárias para o projeto	3
Estudar a documentação das tecnologias escolhidas	2
Instalar e configurar o ambiente de desenvolvimento	3
Criar uma aplicação de teste que permita conectar via <i>bluetooth</i> ao sensor	5

Tabela C.1: Sprint 1

#### Sprint 2 (17 de outubro de 2022 - 28 de outubro de 2022)

Tarefa	Story Points
Continuar a implementação da conexão ao servidor	5
Guardar os dados do sensor na base de dados local da aplicação	8

Tabela C.2: Sprint 2

#### Sprint 3 (31 de outubro de 2022 - 11 de novembro de 2022)

Tarefa	Story Points
Finalizar conexão ao servidor na app definitiva	2
Criação de um serviço em <i>background</i> na aplicação para permitir a recolha de dados mesmo com o telemóvel bloqueado	5
Configurar o servidor, estruturar base de dados MySQL e criar rotas na API para os sensores	8
Começar a planear a estrutura dos questionários	1

Tabela C.3: Sprint 3



### Sprint 4 (14 de novembro de 2022 - 25 de novembro de 2022)

Tarefa	Story Points
Continuação do serviço <i>background</i> de recolha de dados do sensor	3
Envio dos dados da base de dados local para o servidor	8
Continuação do planeamento da estrutura dos questionários de forma a que sejam possíveis de ser alterados e reutilizados em diversos estudos	3

Tabela C.4: Sprint 4

### Sprint 5 (28 de novembro de 2022 - 9 de dezembro de 2022)

Tarefa	Story Points
Criação da base de dados dos questionários	3
Correção de bugs	3
Implementação da biblioteca <i>SurveyKit</i> para a apresentação dos questionários na <i>app</i>	8
Adicionar notificações de recolha de dados e perda de conexão ao sensor	2

Tabela C.5: Sprint 5

### Sprint 6 (12 de dezembro de 2022 - 23 de dezembro de 2022)

Tarefa	Story Points
Implementar a funcionalidade de ativar/desativar a recolha de dados dos sensores	3
Implementar a funcionalidade de ativar/desativar o envio de dados em direto para o servidor	3
Implementar a funcionalidade de informar que não há Wi-Fi ativo para download de questionários	5

Tabela C.6: Sprint 6



### Sprint 7 (26 de dezembro de 2022 - 6 de janeiro de 2023)

Tarefa	Story Points
Correção de bugs	5
Alteração das rotas da api	3

Tabela C.7: Sprint 7

### Sprint 8 (9 de janeiro de 2023 - 20 de janeiro de 2023)

Tarefa	Story Points
Implementar a declaração de consentimento e a lógica para forçar o participante a lê-la antes de prosseguir	1
Migração para um servidor na <i>cloud</i>	2
Alteração das rotas da api	3
Implementar a funcionalidade de transferir novos questionários	3
Início dos testes de usabilidade <i>logout</i>	1

Tabela C.8: Sprint 8

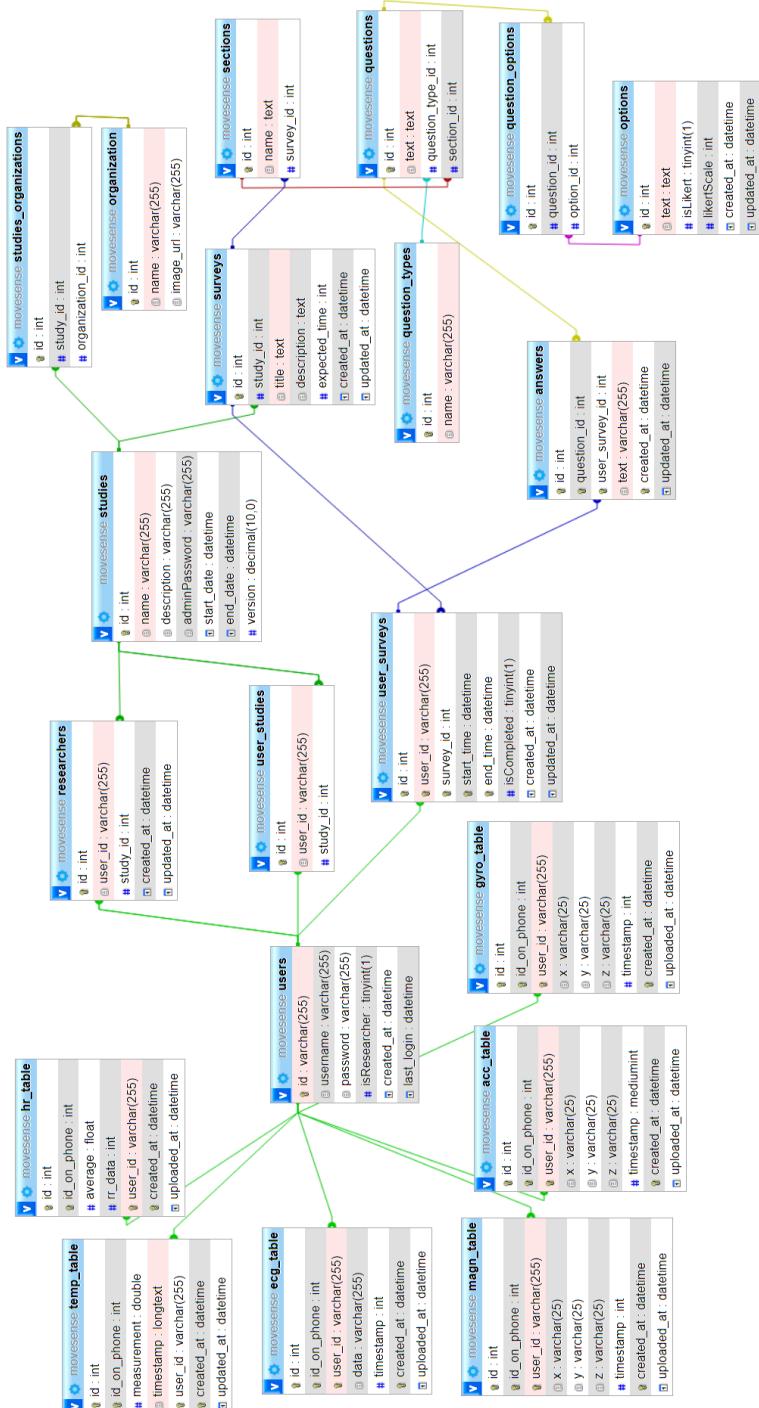
### Sprint 9 (23 de janeiro de 2023 - 24 de fevereiro de 2023)

Tarefa	Story Points
Continuação dos testes e melhorias	3
Corrigir a forma de mostrar os questionários na aplicação	13
Corrigir a forma de enviar grandes quantidades de dados para o servidor	13
Adicionar notificações de envio de dados para o servidor	1
Início do relatório final	1

Tabela C.9: Sprint 9

## Apêndice D

### Diagrama de Entidade e Relação



**Figura D.1:** Estrutura de base de dados.

Pode consultar a imagem completa em [https://github.com/diogopereira00/UMAIA\\_Movesense/blob/main/docs/baseDados.pdf](https://github.com/diogopereira00/UMAIA_Movesense/blob/main/docs/baseDados.pdf).



## Apêndice E

### Questionários

#### Questionário dos Testes Funcionais

**Que dispositivo utilizou durante os testes?**

**Com que frequência utilizou a aplicação?**

- Menos de uma vez por semana
- Uma vez por semana
- Duas vezes por semana
- Três vezes por semana
- Mais de três vezes por semana

**Durante a sua utilização, com que frequência encontrou problemas a conectar com o sensor Movesense?**

- Nunca
- Raramente
- Muitas vezes
- Sempre

**Durante a sua utilização, com que frequência encontrou problemas com os questionários?**

- Nunca
- Raramente
- Muitas vezes
- Sempre

**Durante a sua utilização, com que frequência encontrou problemas durante a atividade?**

- Nunca
- Raramente
- Muitas vezes
- Sempre

**Durante a sua utilização teve mais algum problema? Se sim. qual?**

- Sim
- Não

**Se encontrou problemas durante os testes, os mesmos foram resolvidos?**

- Sim
- Não

**Existe alguma funcionalidade que gostava de ver adicionada ou modificada na aplicação?**



## Questionários de Pré-treino

Pergunta	Tipo	Opções
Neste momento, como avalio o meu nível de bem-estar geral?	Escala de Likert	1=Baixo bem-estar; 10=Elevado bem-estar
Neste momento, quão deprimido/‘em baixo’ me sinto?	Escala de Likert	1=Nada deprimido, “em baixo”; 10=Muito deprimido, “em baixo”
Neste momento, quão ansioso/stressado me sinto?	Escala de Likert	1=Nada ansioso/stressado; 10=Muito ansioso/stressado

Tabela E.1: Questionário pré-treino

## Questionários de Pós-treino

Pergunta	Tipo	Opções
Neste momento, como avalio o meu nível de bem-estar geral?	Escala de Likert	1=Baixo bem-estar; 10=Elevado bem-estar
Neste momento, quão deprimido/‘em baixo’ me sinto?	Escala de Likert	1=Nada deprimido; 10=Muito deprimido
Neste momento, quão ansioso/stressado me sinto?	Escala de Likert	1=Nada ansioso; 10=Muito ansioso
Tipo de exercício físico	Escolha Múltipla	Cardio, musculação, cardio e musculação, outro
Intensidade da prática	Escala de Likert	0=Nada intenso; 10=Extremamente intenso
Como foi realizado o exercício?	Escolha Múltipla	Individual ou em grupo
Contexto onde foi realizado o exercício físico	Escolha Múltipla	Ginásio, clube desportivo, casa/ar livre

Tabela E.2: Questionário pós-treino



## Resultados dos testes de usabilidade por participante

Tarefa	Tempo
Efetuar o login	23 segundos
Consentir com a recolha de dados	5 segundos
Conectar com o sensor	29 segundos
Ir até as definições	3 segundos
Começar uma atividade	2 segundos
Responder ao questionário pré-treino	35 segundos
Terminar atividade	7 segundos
Responder ao questionário pós-treino	67 segundos

Tabela E.3: Teste de usabilidade — Participante 1

Tarefa	Tempo
Efetuar o login	16 segundos
Consentir com a recolha de dados	2 segundos
Conectar com o sensor	25 segundos
Ir até as definições	2 segundos
Começar uma atividade	2 segundos
Responder ao questionário pré-treino	27 segundos
Terminar atividade	5 segundos
Responder ao questionário pós-treino	56 segundos

Tabela E.4: Teste de usabilidade — Participante 2

Tarefa	Tempo
Efetuar o login	26 segundos
Consentir com a recolha de dados	7 segundos
Conectar com o sensor	45 segundos
Ir até as definições	4 segundos
Começar uma atividade	3 segundos
Responder ao questionário pré-treino	38 segundos
Terminar atividade	2 segundos
Responder ao questionário pós-treino	74 segundos

Tabela E.5: Teste de usabilidade — Participante 3



## Apêndice F

### Referência técnica da API

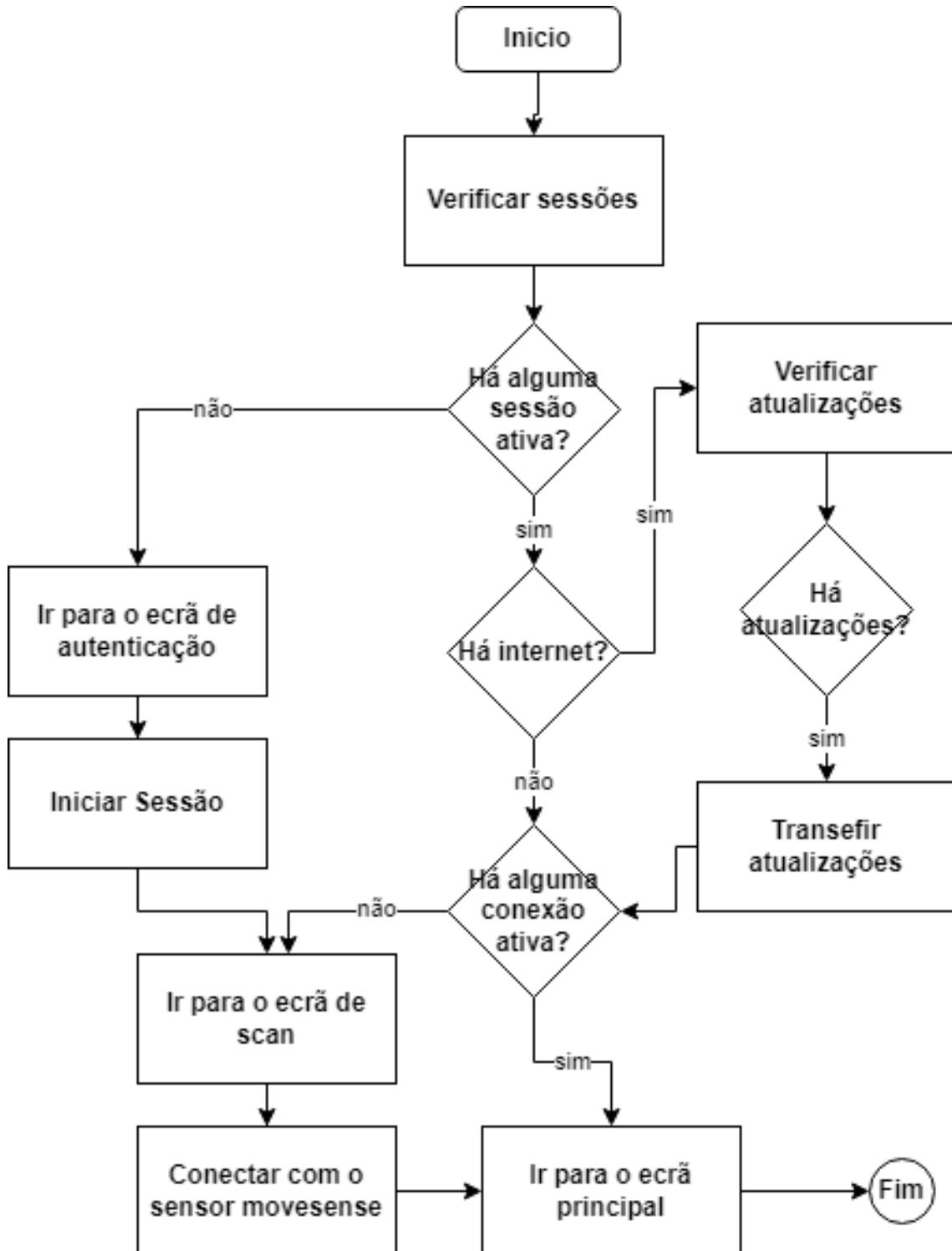
A API pode ser consultada na sua totalidade aqui <https://documenter.getpostman.com/view/14703969/2s93RZM9Zu>. O código-fonte pode ser consultado no [https://github.com/diogopereira00/UMAIA\\_Movesense/tree/main/backend](https://github.com/diogopereira00/UMAIA_Movesense/tree/main/backend)

Método	Endpoint	Função
POST	/sign-up	Regista um novo utilizador
POST	/login	Autentica um utilizador
GET	/options	Retorna todas as opções das perguntas dos questionários
GET	/questions/options	Retorna todas as opções das perguntas dos questionários
GET	/questionsTypes	Retorna o id e o nome de cada tipo de pergunta
GET	/studies/:user_id	Retorna todos os estudos de cada utilizador
GET	/studies/allInfo/:user_id	Retorna a informação de todos os estudos através do user_id
GET	/surveys/:user_id	Retorna todos os questionários do utilizador
GET	/studies/:study_id/-version	Retorna a versão do estudo através do id
POST	/studies/add/user-Surveys	Adiciona o questionário respondido e todas as respostas às perguntas
POST	/addAccData	Adiciona todos os dados do acelerómetro
POST	/addGyroData	Adiciona todos os dados do giroscópio
POST	/addMagnData	Adiciona todos os dados do magnetômetro
POST	/addECGData	Adiciona todos os dados do ECG
POST	/addTempData	Adiciona todos os dados da temperatura
POST	/addHRData	Adiciona todos os dados da frequência cardíaca
POST	/addAllData	Adiciona todos os dados dos sensores

Tabela F.1: Caminhos da API

## Apêndice G

### Diagrama de atividades



**Figura G.1:** Diagrama de atividades



## Apêndice H

### Interfaces da aplicação

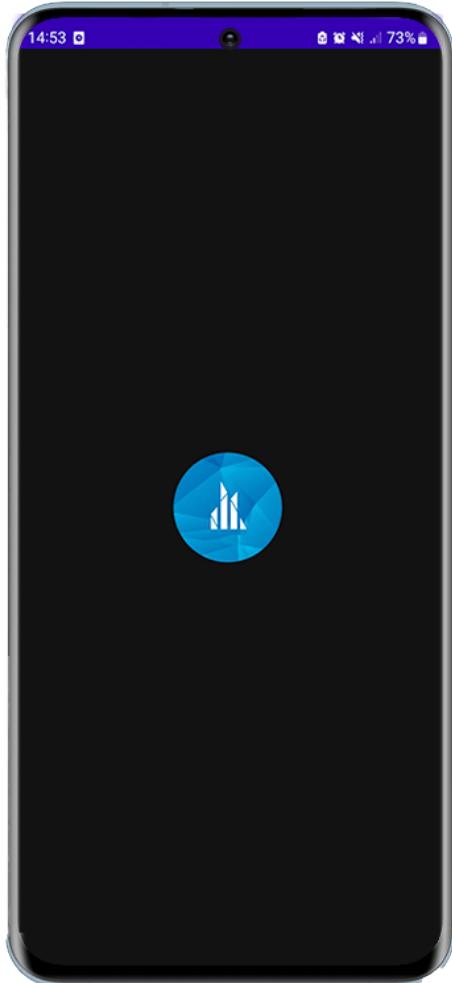


Figura H.1: *Splash Screen*

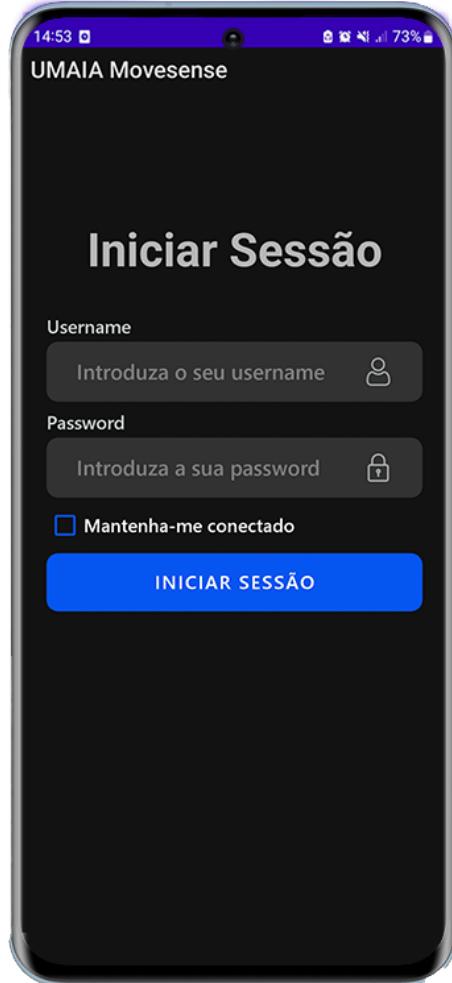
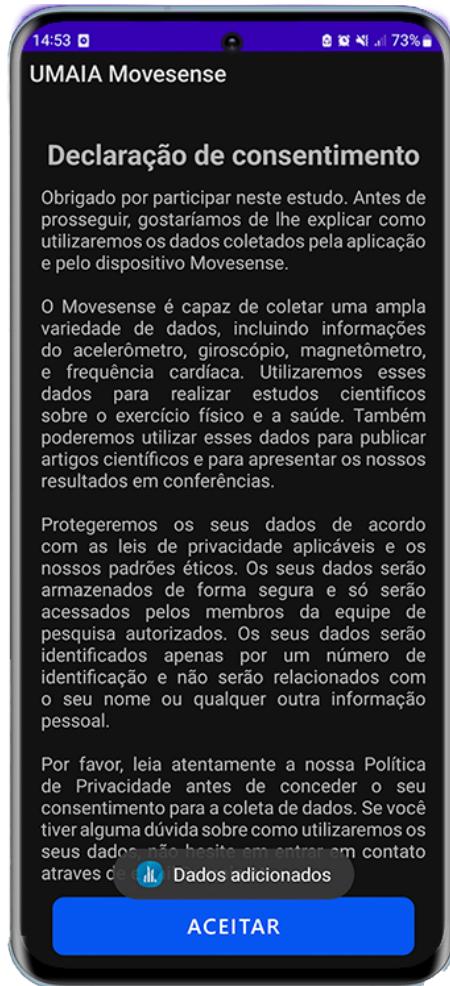
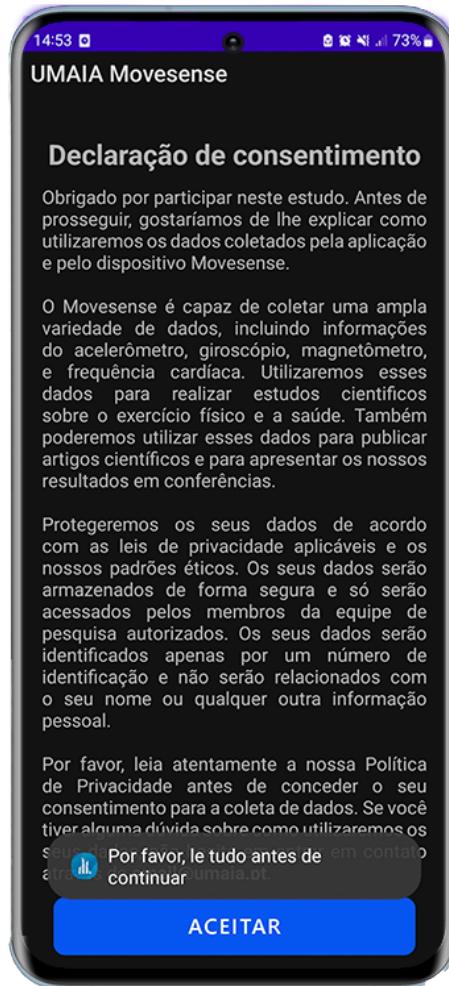


Figura H.2: Ecrã de Autenticação



**Figura H.3:** Dados dos questionários adicionados no ecrã de consentimento



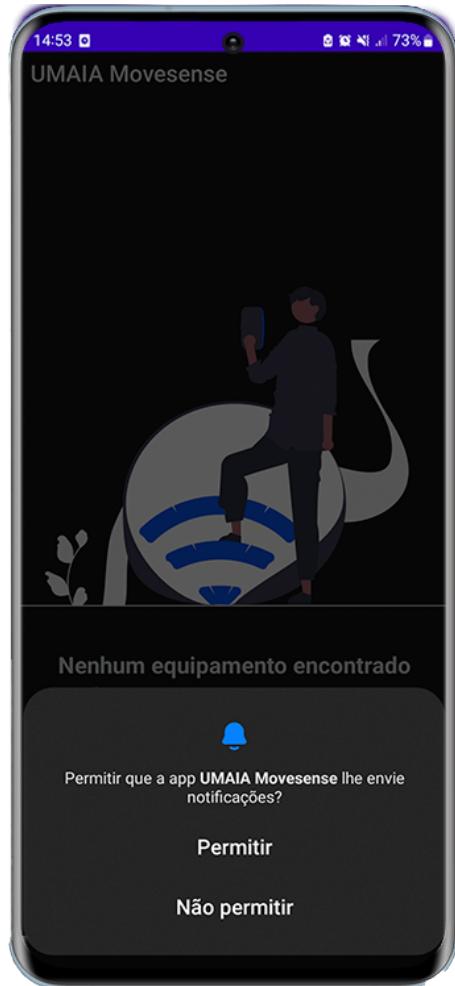
**Figura H.4:** “Por favor, lê tudo antes de continuar”, ecrã de consentimento



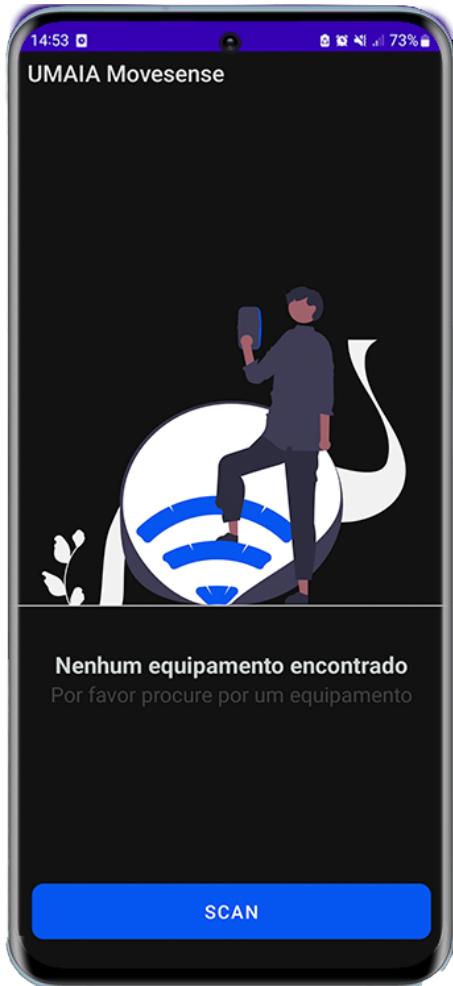
**Figura H.5:** Pedido permissão localização do dispositivo.



**Figura H.6:** Pedido de permissão de dispositivos próximos.



**Figura H.7:** Pedido de permissão notificações



**Figura H.8:** Ecrã de procura de dispositivos Movesense próximos

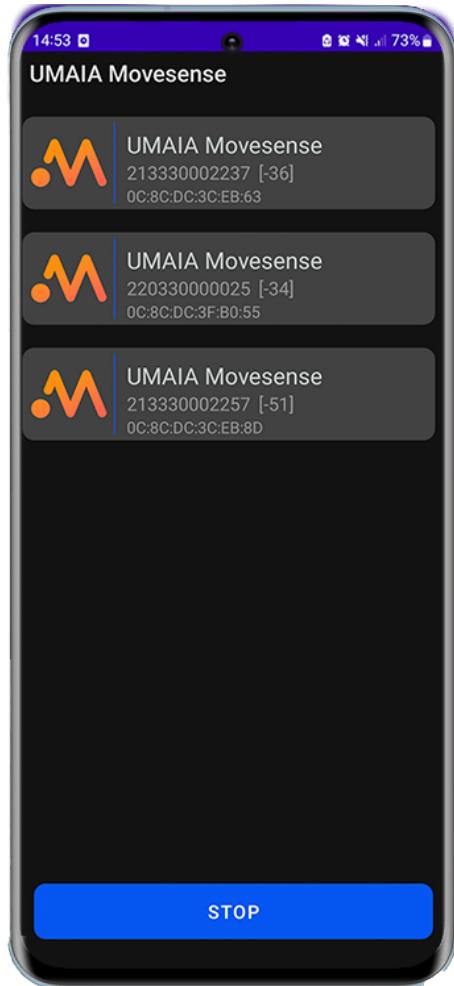


Figura H.9: Pesquisa em curso.

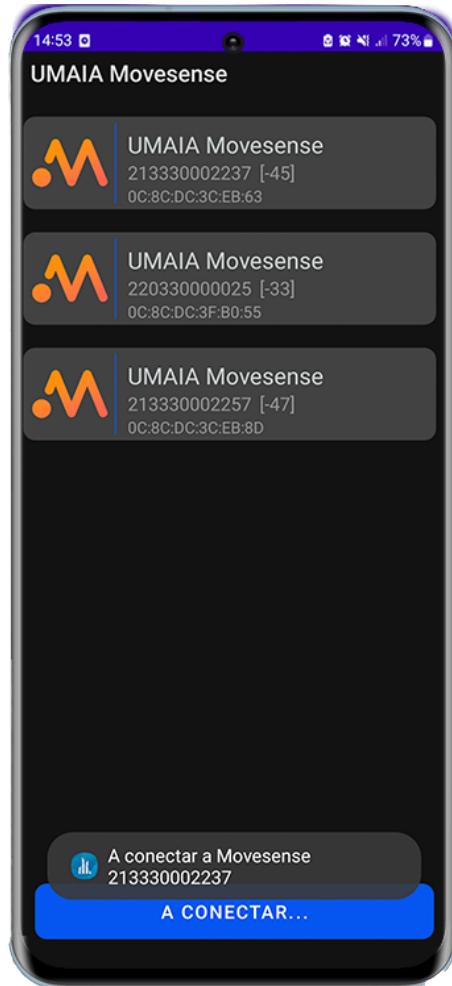


Figura H.10: A conectar a um sensor.

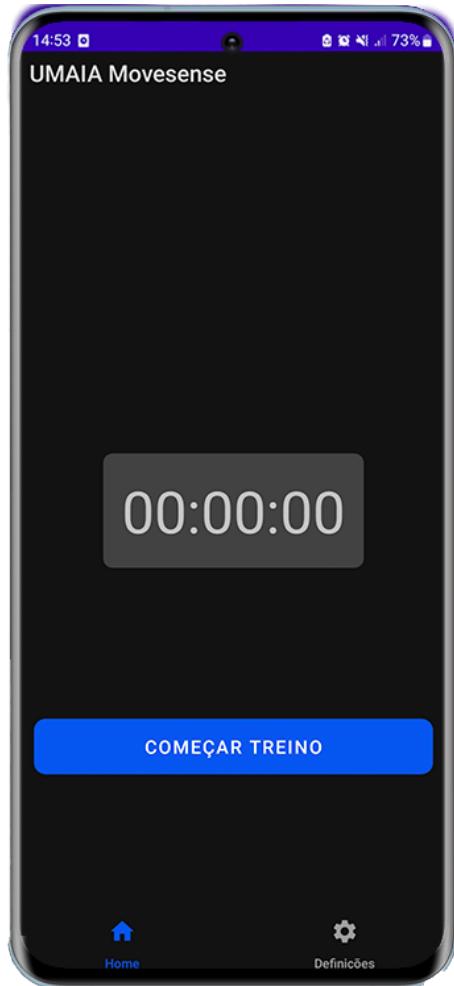


Figura H.11: Ecrã principal



Figura H.12: Ecrã das definições

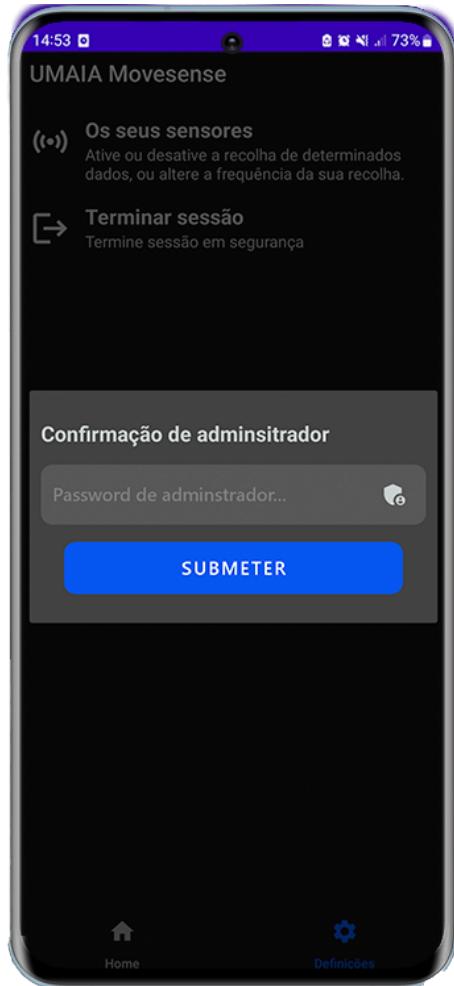
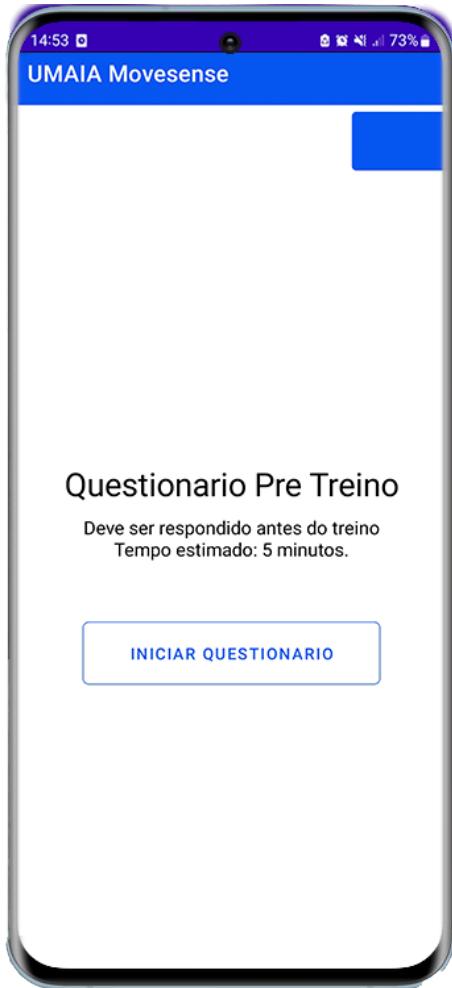


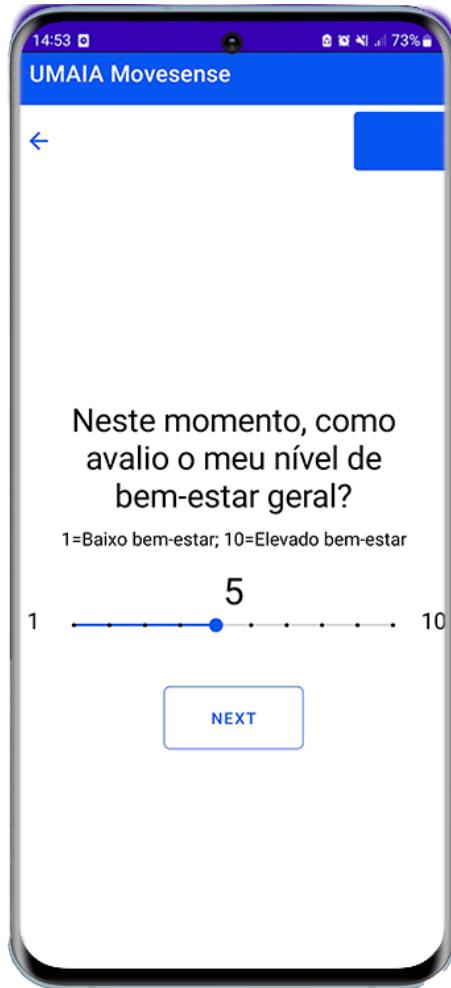
Figura H.13: Pedido de password administrador



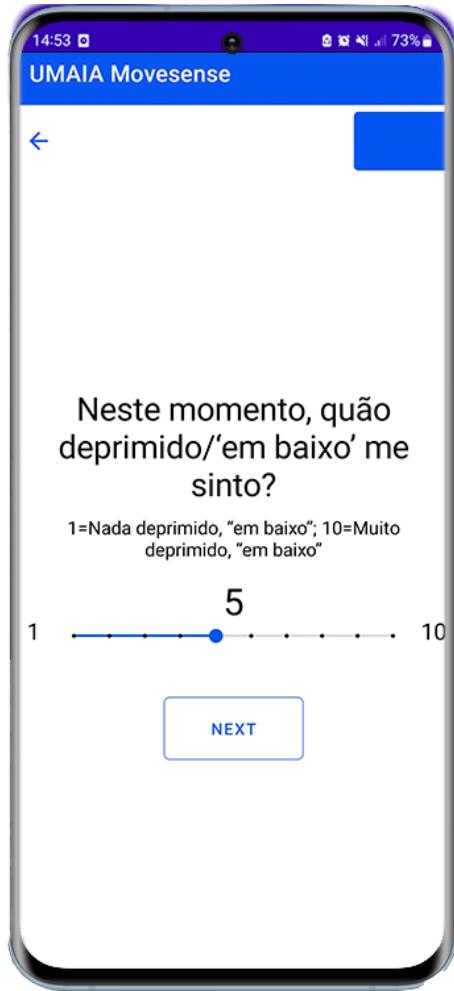
Figura H.14: Definições dos sensores, após introduzir a password



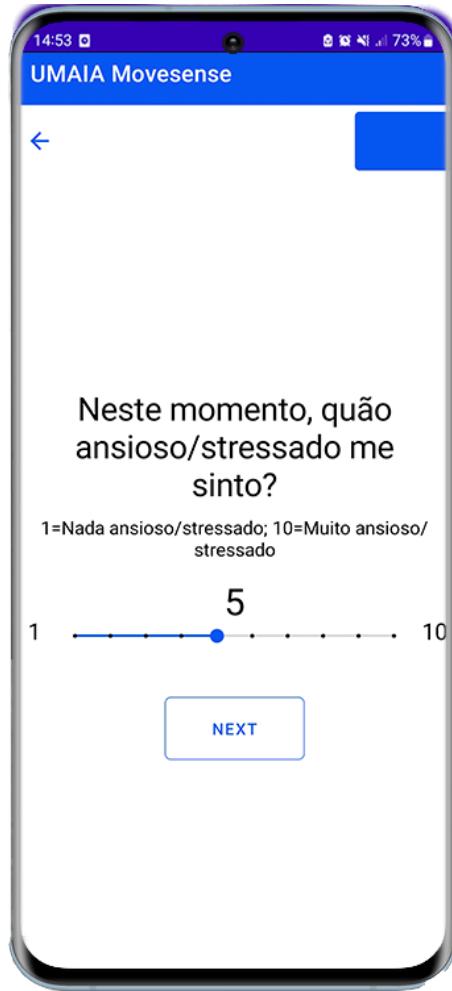
**Figura H.15:** Ecrã inicial do questionário pré-treino



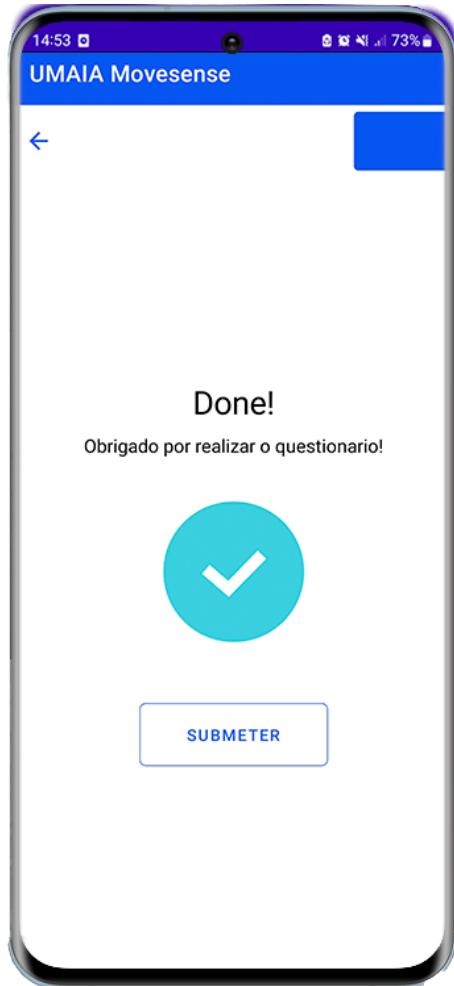
**Figura H.16:** Pergunta 1 do questionário pré-treino



**Figura H.17:** Pergunta 2 do questionário pré-treino



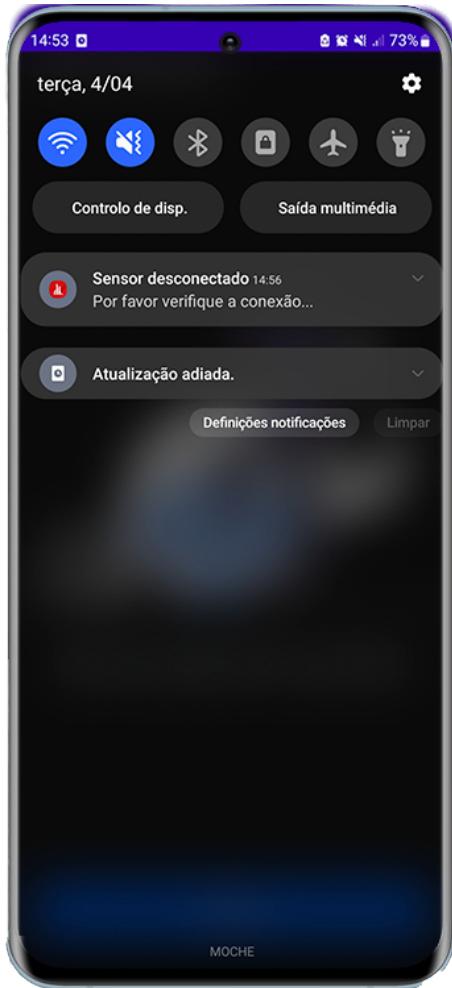
**Figura H.18:** Pergunta 3 do questionário pré-treino



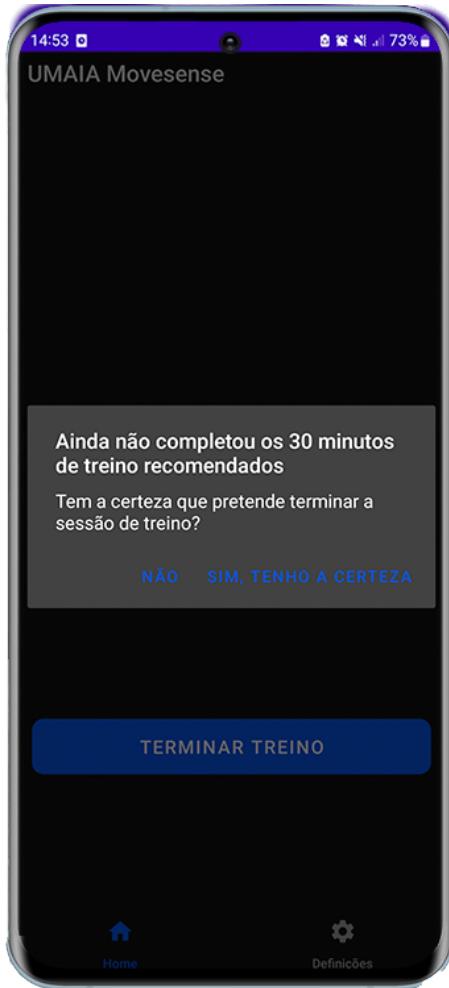
**Figura H.19:** Terminar pré-treino e começar atividade



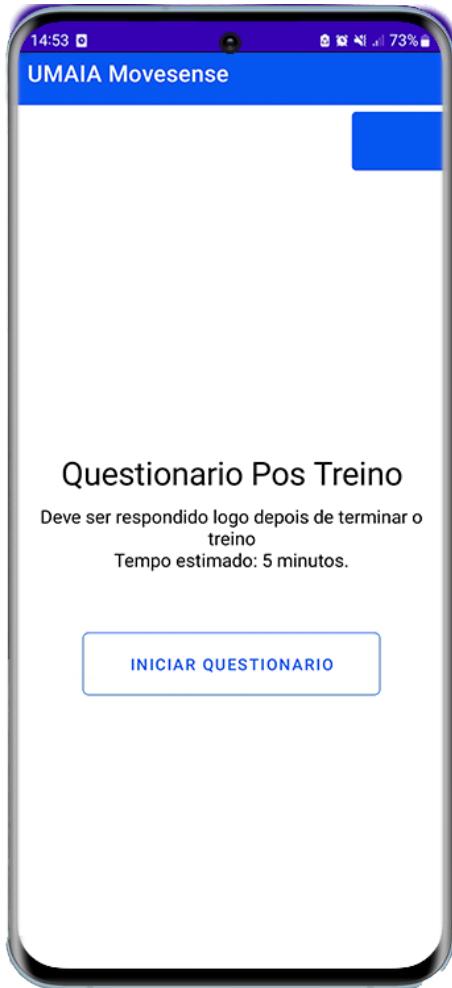
**Figura H.20:** Notificação a avisar da recolha de dados



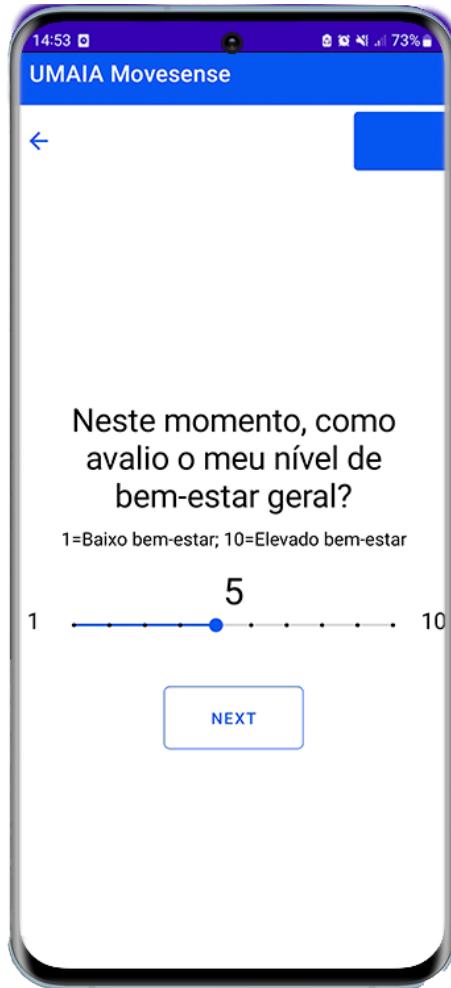
**Figura H.21:** Notificação a avisar a perda de conexão ao sensor



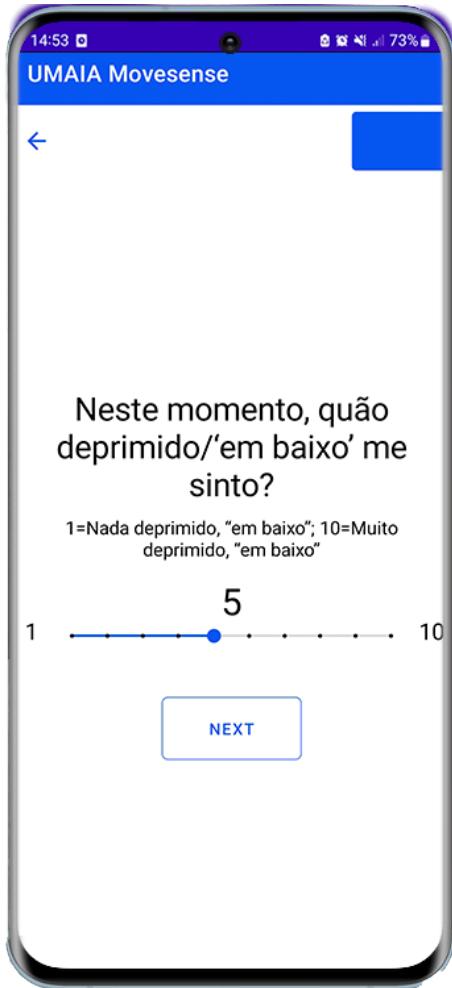
**Figura H.22:** Mensagem de aviso atividade inferior a 30 minutos



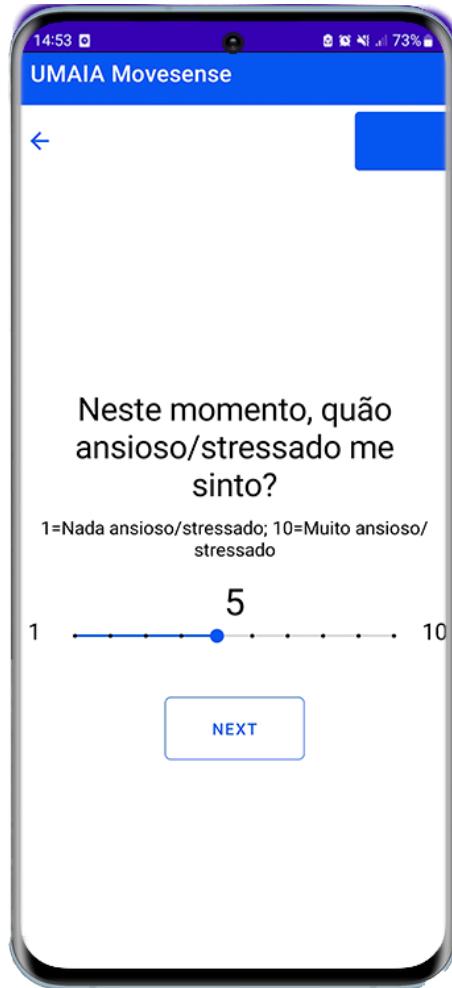
**Figura H.23:** Ecrã inicial do questionário pós-treino



**Figura H.24:** Pergunta 1 do questionário pós-treino



**Figura H.25:** Pergunta 2 do questionário pós-treino



**Figura H.26:** Pergunta 3 do questionário pós-treino



14:53 73%  
UMAIA Movesense

Tipo de exercício físico

Cardio

Musculação

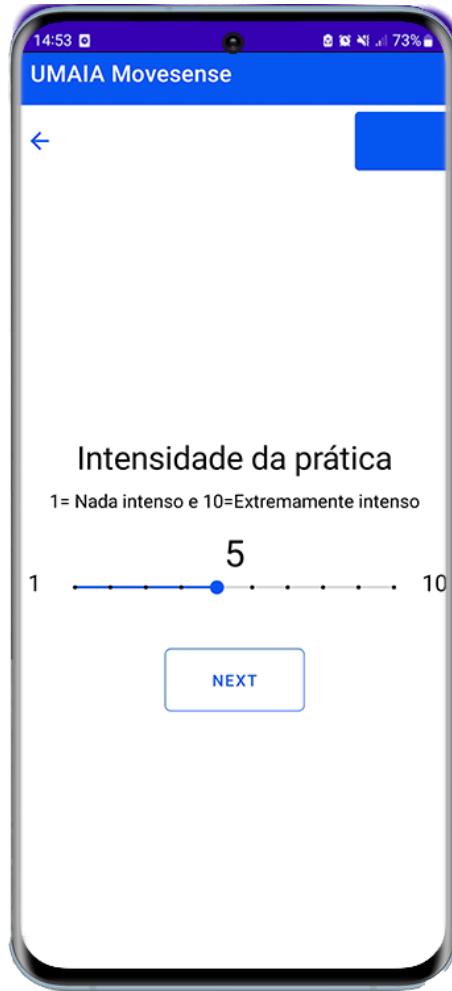
Musculação e Cardio

Outro

**NEXT**

This screenshot shows a mobile application interface for a post-training questionnaire. The title 'UMAIA Movesense' is at the top. Below it, the question 'Tipo de exercício físico' is displayed. A list of options follows: 'Cardio', 'Musculação', 'Musculação e Cardio', and 'Outro', with the last option having a checked checkbox. At the bottom is a blue 'NEXT' button.

**Figura H.27:** Pergunta 4 do questionário pós-treino



**Figura H.28:** Pergunta 5 do questionário pós-treino



14:53 73% UMAIA Movesense

Exercício realizado individualmente ou em grupo

Sozinho

Grupo

NEXT

This screenshot shows a mobile application interface for a questionnaire. At the top, it displays the time (14:53), battery level (73%), and the app name (UMAIA Movesense). Below this is a question in Portuguese asking if the exercise was performed individually or in a group. Two options are listed: "Sozinho" (alone) and "Grupo" (group). A large blue rectangular button labeled "NEXT" is centered at the bottom of the screen.

**Figura H.29:** Pergunta 6 do questionário pós-treino

14:53 73% UMAIA Movesense

Contexto onde foi realizado o exercício físico

Ginásio ✓

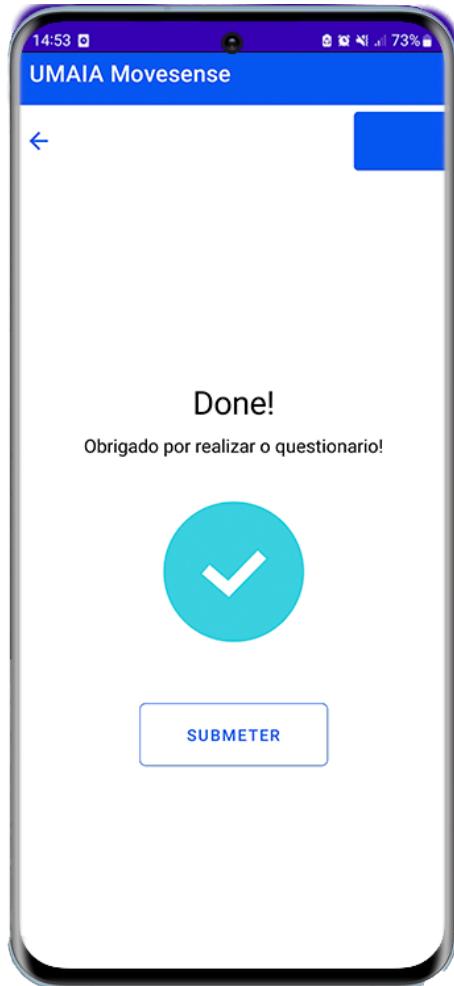
Clube Desportivo

Casa/Ar livre

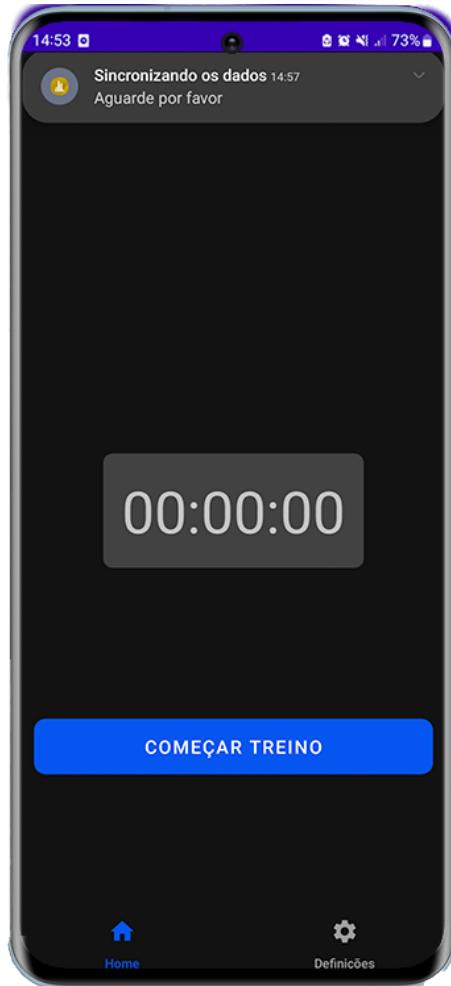
NEXT

This screenshot shows a mobile application interface for a questionnaire. At the top, it displays the time (14:53), battery level (73%), and the app name (UMAIA Movesense). Below this is a question in Portuguese asking about the context where the physical exercise was performed. Three options are listed: "Ginásio" (Gym), "Clube Desportivo" (Sports Club), and "Casa/Ar livre" (Home/Free). The option "Ginásio" has a blue checkmark next to it. A large blue rectangular button labeled "NEXT" is centered at the bottom of the screen.

**Figura H.30:** Pergunta 7 do questionário pós-treino



**Figura H.31:** Terminar pós-treino e terminar atividade



**Figura H.32:** Notificação envio de dados para o servidor



Figura H.33: Terminar sessão



## Apêndice I

### Blocos de comandos e ficheiros de configuração principais

#### Função `connectBLEDevice()`

```
private fun connectBLEDevice(device: MyScanResult) {
    val bleDevice = getBleClient()!!.getBleDevice(device.-
        macAddress)
    Timber.e("Connecting to BLE device: " + bleDevice.-
        macAddress)

    mMds!!.connect(bleDevice.macAddress, object : ←
        MdsConnectionListener {
        override fun onConnect(s: String) {
            Timber.e("onConnect:$s")
        }

        override fun onConnectionComplete(macAddress: ←
            String, serial: String) {
            //Cria notificação, sensor conectado.
            gv.connected = true
            createNotification()

            for (sr in bluetoothList) {
                if (sr.macAddress.equals(macAddress, true))←
                    {
                    sr.markConnected(serial)
                    break
                }
            }
        }
    }
}

// mScanResArrayAdapter.notifyDataSetChanged()
Toast.makeText(
    this@MovesenseService, "Conectado ao sensor←
        ${serial}.", Toast.LENGTH_SHORT
).show()

val intent = Intent(
    this@MovesenseService, MainActivity::class.-
        java
).addFlags(Intent.FLAG_ACTIVITY_NEW_TASK)
```



```
        startActivity(intent)
        if (!isServiceStopped) {
            startForegroundService()
        } else {
//            stopService()
        }
    }

    override fun onError(e: MdsException) {
        Timber.e("onError:$e")
        gv.connected = false
        val intent = Intent("update_button")
        intent.putExtra("action", "enable")
        LocalBroadcastManager.getInstance(←
            this@MovesenseService).sendBroadcast(intent)
        showConnectionError(e)
    }

    override fun onDisconnect(bleAddress: String) {
        Timber.e("onDisconnect: $bleAddress")
        gv.connected = false
        createNotification()
//        Toast.makeText(this@MovesenseService, "←
//DESCONECTADOz<x<z.", Toast.LENGTH_SHORT)
//            .show()

        for (sr in bluetoothList) {
            if (bleAddress == sr.macAddress) {

                // Unsubscribe all from possible
                if (sr.connectedSerial != null && Home.←
                    s_INSTANCE != null && sr.←
                    connectedSerial.equals(
                        gv.currentDevice.←
                        connectedSerial
                    )
                ) {
                    unsubscribeAll()
                }
            }
        }
    }
}
```

```
        Home.s_INSTANCE!!.activity?.finish()
                ()
        // stopService(Intent(←
        this@ScanDevice, MyService::class.java))
                }
        sr.markDisconnected()
                }
        }
    }
})
```

#### **Excerto I.1: Função connectBLEDevice().**



## Função hasInternetWifi()

```
private val connectivityManager =  
    context.getSystemService(Context.CONNECTIVITY_SERVICE) ↪  
        as ConnectivityManager  
    //Verifica se existe conexão wifi.  
    fun hasInternetWifi(): Boolean {  
        return if (Build.VERSION.SDK_INT >= Build.VERSION_CODES↪  
            .M) {  
            val network: Network = connectivityManager?.↪  
                activeNetwork ?: return false  
            val capabilities = connectivityManager.↪  
                getNetworkCapabilities(network) ?: return false  
  
            return capabilities.hasTransport(↪  
                NetworkCapabilities.TRANSPORT_WIFI) || ↪  
                capabilities.hasTransport(  
                    NetworkCapabilities.TRANSPORT_VPN  
                )  
        } else {  
            val activeNetworkInfo = connectivityManager?.↪  
                activeNetworkInfo  
            if (activeNetworkInfo != null) {  
                return activeNetworkInfo.type == ↪  
                    ConnectivityManager.TYPE_WIFI  
            }  
            false  
        }  
    }
```

**Exerto I.2:** Função hasInternetWifi().



## Função hasInternet()

```
private val connectivityManager =  
    context.getSystemService(Context.CONNECTIVITY_SERVICE) ↪  
        as ConnectivityManager  
    //Verifica se existe internet, quer seja wifi ou com dados ↪  
    moveis  
    fun hasInternet(): Boolean {  
        return if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {  
            val network: Network = connectivityManager?.activeNetwork ?: return false  
            val capabilities = connectivityManager.getNetworkCapabilities(network) ?: return false  
            return capabilities.hasTransport(  
                NetworkCapabilities.TRANSPORT_WIFI) ||  
                capabilities.hasTransport(  
                    NetworkCapabilities.TRANSPORT_VPN) ||  
                capabilities.hasTransport(  
                    NetworkCapabilities.TRANSPORT_CELLULAR)  
        } else {  
            val activeNetworkInfo = connectivityManager?.activeNetworkInfo  
            if (activeNetworkInfo != null) {  
                return activeNetworkInfo.type == ConnectivityManager.TYPE_WIFI ||  
                    activeNetworkInfo.type == ConnectivityManager.TYPE_MOBILE  
            }  
            false  
        }  
    }
```

**Exerto I.3:** Função hasInternet().