ISEL – Instituto Superior de Engenharia de Lisboa

ADEETC – Área Departamental de Engenharia de Electrónica e Telecomunicações e de Computadores

LEIM

Licenciatura em Engenharia Informática e Multimédia

Unidade Curricular de Projeto

Anotação Assistida de Dados Fisiológicos

Eduardo Santos 40610

Orientador(es)

Professor Doutor André Lourenço

Professor Doutor Rui Jesus

Setembro, 2020

Resumo

Há alguns anos, pensava-se que a Inteligência artificial poderia ser usada de forma abrangente e ser solução para responder a vários problemas no mundo. Nos dias de hoje, verifica-se que é utilizada para resolver problemas num determinado contexto.

Para certo tipo de problemas, são utilizados algoritmos de aprendizagem automática, que necessitam de uma enorme quantidade de dados e anotações/categorizações. Este trabalho foca-se no problema da anotação de sinais fisiológico. Para isso foi desenvolvida uma aplicação móvel, de anotação assistida de dados fisiológicos.

Esta aplicação tem como funcionalidade permitir de forma usável anotações na perspectiva emocional e de contexto ao utilizador. As anotações de contexto são realizadas a partir de *tags* que são previamente fornecidas pelo utilizador.

Os dados fisiológicos são adquiridos com recurso a *wearable*, ligados através de Bluetooth Low Energy (**BLE**) e adquiridos em segundo plano (*Background Service*).

O *wearable* usado neste projeto é um equipamento da empresa pulseOn, que permite adquirir sinais cardiacos e de actividade inercial do utilizador. Foram estudados vários caso de uso, e a prova de conceito foi realizada no contexto de aquisições de dados fisiológicos num projeto de monitorização de um utilizador num simulador de condução.

Abstract

Some years ago, we used to believe that Artificial Inteligence could be used in an omnibus way and be the solution for various problems in the world.Today, we use it to solve problems in a certain context.

For certain types of problems, we use algorithms of automathic learning, or Machine Learning, that require a lot of data and annotations/categorization. This project focuses on the problem of annotation of physiological data. For that purpose, it was develloped a mobile app of assisted annotation of physiological data.

This application has the functionality to allow, in a usable way, annotations of the emotional perspective and context to the user. The context annotations are made through tags previously provided by the user.

The physiological data is gathered using wearables, connected through Bluetooth Low Energy (BLE), and acquired in background. The wearable used in this project is a pulseOn company equipment, which allows the gathering of cardiac signs and inertial activity of the user. Various use cases were studied, demonstrating the prototype in the context of acquisition.

This mobile app shows the intended effect of acquiring properly registered physiological parameters as a prototype.

Within the scientific scope, it's fundamental to the appropriate acquisition of physiological data, otherwise it can lead to bad results in the algorithms.This mobile app shows the intended effect of acquiring properly registered physiological parameters as a prototype.

Agradecimentos

Aos meus orientadores André Lourenço e Rui Jesus, pelo incansável apoio, disponibilidade e paciência durante todo o projecto, sempre disponíveis e que me deram incentivo e força de vontade para este projecto.

A toda a equipa da CardioID que me apoiaram e ajudaram com equipamentos e duvidas que foram surgindo.

Aos meus amigos Inês Silva e André Costa pela motivação e força transmitida.

Ao docente Arnaldo Abrantes pelo apoio nas aulas de projecto e pela pressão colocada por ele.

Aos Bombeiros Voluntários de Odivelas que por muitas noites me apoiaram enquanto trabalhava no projecto.

À minha família que desde do inicio me deu apoio, e teve muita paciência para me apoiar.

A todos, os meus sinceros agradecimentos.

Dedico este projecto a todos os meus amigos e família, por todo o apoio e dedicação.

"Programming is a skill best acquired by practice and example rather than from books".
Alan Turing

Índice

Lista de Tabelas

Lista de Figuras

Lista de Acrónimos

ATT Attribute Protocol

BLE Bluetooth Low Energy

CDN Content Delivery Network

GATT Generic Attribute Profile

HRV Heart Rate Viariability

IT Instituto de Telecomunicações

JSON JavaScript Object Notation

OHR Optical Heart Rate

PPG Photoplethysmograph

UUID universally unique identifier

Capítulo 1

Introdução

Os sistemas de aprendizagem automática (*Machine Learning*), em particular os sistemas de classificação supervisionados, necessitam de uma grande quantidade de dados anotados para treinarem os modelos. Este requisito torna-se mais difícil de atingir quando os dados de entrada são difíceis de obter, como é o caso da classificação de sinais fisiológicos. Por sinais fisiológicos entende-se todo o tipo de biosinais que são adquiridos usando sensores que monitorizem variáveis fisiológicas [Bronzino, 2000] de um utilizador: eletrocardiograma (ECG), onda de pulso/ fotoplestimografia (PPG), temperatura, etc. A necessidade de anotar a aquisição deste tipo de sinais é a base deste projecto.

Este projecto pretende desenvolver uma aplicação móvel de **anotação assistida de dados fisiológicos**, com a motivação de obter dados fisiológicos e anotações que permitam caracterizar contexto ou actividade que o utilizador está a realizar. Esses dados são agregados numa base de dados para posteriormente ser disponibilizados para treinarem sistemas de aprendizagem automática. Os dados fisiológicos são adquiridos usando sensores que se ligam ao smartphone usando Bluetooth Low Energy (BLE) [SIG, 2020].

A necessidade de especificar contextos, deve-se ao facto de que as anotações a realizar são muito distintas dependendo do contexto. Por exemplo, é muito diferente anotar sinais adquiridos enquanto um utilizador está a assistir a um jogo de futebol, ou enquanto está a conduzir um veículo.

A empresa CardioID Technologies [1] que está a desenvolver sistemas de monitorização baseados em dados fisiológicos é um dos beneficiários desta

[1] `www.cardio-id.com`

aplicação, que tem como finalidade permitir a investigadores ou engenheiros que estejam a a desenvolver aplicações baseadas em sinais fisiológicos ter capacidade de anotar sinais para depois devolver aplicações dedicadas.

A aplicação a desenvolver é baseada no sistema **Android**, uma vez que é líder do mercado, sendo utilizado em 85% dos *smartphones*, e muito receptível a novas aplicações. Pretende-se que a aplicação consiga recolher dados em segundo plano (*Background Service*) e que tenha a capacidade de se ligar a vários sensores de aquisição de dados fisiológicos. Os objectivos da aplicação são:

- Adquirir e agregar dados fisiológicos numa base de dados;
- Anotar de forma assistida;
- Compatibilidade com vários dispositivos de aquisição de dados;
- Adquirir dados em segundo plano;
- Melhorar a experiência do utilizador.

O presente relatório está dividido em nove capítulos, em que o primeiro capítulo é relativo a introdução, motivação e objectivos do projecto, o segundo capítulo ao trabalho relacionado, o terceiro capítulo ao modelo proposto, do quarto capítulo até ao sétimo é descrito a implementação do projecto, e os últimos dois capítulos referentes a testes, trabalho futuro e conclusões.

Capítulo 2

Trabalho Relacionado

A criação de anotações em sistemas de aprendizagem automática não é um assunto recente. Neste capítulo resumem-se alguns dos trabalhos relacionados, tendo por base a ideia de gamificação do processo de aquisição de anotações [Jesus et al., 2010] e a anotação de dados fisiológicos [Lourenço et al., 2014] [Simón et al., 2017].

Em [Jesus et al., 2010], são utilizados métodos de aprendizagem automática e técnicas semi-automáticas para treinar modelos de conceitos semânticos. Em que o método automático, baseado em conceitos semânticos estimados, usando conteúdo visual, metainformação de contexto e informação de áudio. O semi-automático é baseado em resultados fornecidos por um jogo.

Estes métodos reforçam a necessidade de utilização metainformação dos dados adquiridos, focando no ponto da experiência do utilizador através de um jogo. O conceito de anotar já por si é uma tarefa cansativa, do qual se torna ainda mais critica no conceito de anotar dados fisiológicos. Este artigo demonstra que é possível tornar apelativo e interessante para o do utilizador. Esta aplicação pretende obter uma boa experiência por parte do utilizador, essa solução pode partir da gamificação, mas também da simplicidade e de quanto apelativo é aplicação.

O artigo [Lourenço et al., 2014], apresenta uma plataforma baseada na web, para visualização e indexação de anotação de biosinais.

Este sistema em comparação com este projecto permite a representação de dados fisiológicos, e que nesses dados seja adicionado metainformação.

Nesse sentido entramos no conceito aprendizagem supervisionada, em que os dados que são adquiridos através da aplicação seja convertidos para me-

tainformação, com isso podemos relacionar os dados, tanto numa linha de contexto, temporal e geográfica, sem perder informação essencial. Com esta definição obtemos dados mais ricos e fidedignos.

Estes dados podem ser estruturados através de **JSON**. Representado na figura 2.1 ilustrativa deste artigo.

```
Record = {
  'Header': {
    'Session': '#',
    'Date': 'yyyy-mm-dd',
    'Time': 'hh:mm:ss',
    'Supervisor': 'txt',
    'Experiment': {
      'Name': 'txt',
      'Description': 'txt',
      'Goals': 'txt'},
    'Subject': {
      'Name': 'txt',
      'Birthdate': 'yyyy-mm-dd',
      'Gender': 'txt',
      'Contact': 'txt'},
    'Tags': ['txt', ...]},
  'Audit': {},
  'Biosignal': {
    'Type': 'txt',
    'Device': 'txt',
    'Sample Rate': '#',
    'Duration': '#',
    'Resolution': '#',
    'Units': {
      'Time': 'txt',
      'Sensor': 'txt'}},
  'Events': {
    'Source': 'txt',
    'Type': 'txt',
    'Dictionary': {
      'key': 'value'}}}
```

Figura 2.1: Hierarchical structure of a Record

A DAFIESKU [Simón et al., 2017] é um sistema para aquisição de dados fisiológicos.

Este sistema é o que mais se assemelha a este projecto, que se fundamenta em aquisição de dados fisiológicos para contexto cientifico através de experiências pré-definidas e posteriormente passadas a um grupo de participantes, através de aplicação.

No entanto nesta aplicação queremos dar total liberdade ao utilizador, porque dados estão em todo o lado, e restringir um utilizador a uma única experiência pode não ser a melhor abordagem. Nesta aplicação existe a intenção de focar no utilizador, se este conceito não for bem aplicado a aplicação torna-se aborrecida, porque não vai captar a atenção do utilizador podendo tonar estas aquisições inadequadas.

Entretanto esta aplicação demonstra bastantes bem o conceito de anotação através dos utilizadores, como se pode verificar na figura 2.2 ilustrativa

deste artigo.

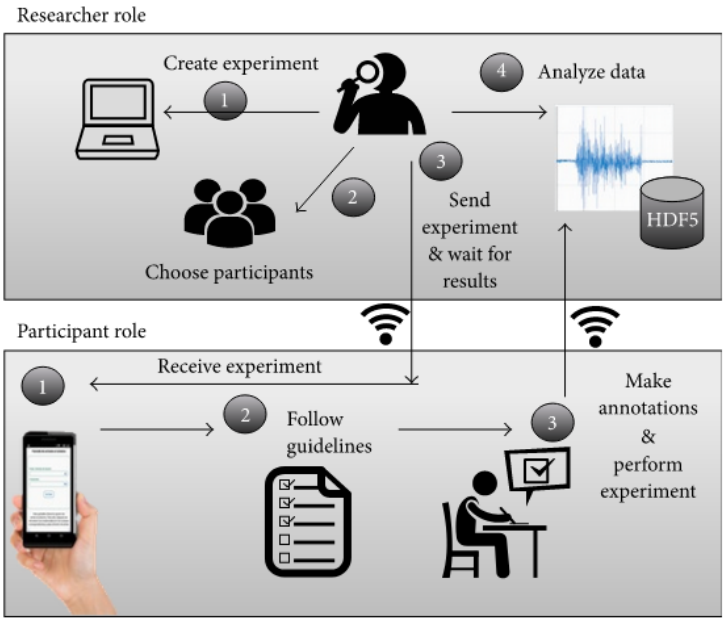

Figura 2.2: Sistema para aquisição de dados fisiológicos (retirada de [Simón et al., 2017])

O **Bitalino** [da Silva, 2012] nasceu no Instituto de Telecomunicações (**IT**) em 2012. Consiste num kit electrónico que disponibiliza vários sensores fisiológicos que se aplica a diversas áreas como eletrocardiografia, a eletromiografia, a atividade eletrodermal ou o desporto, para complementar disponibiliza também uma aplicação que está disponível para **Android**, que é o ***OpenSignals*** [BITalino,].

O ***OpenSignals*** é um software que permite aquisição dados fisiológicos mas que não permite adicionar anotações.

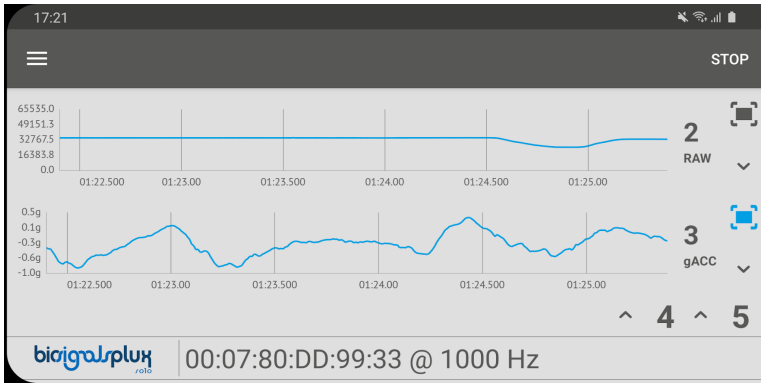

Figura 2.3: OpenSignals - Interface de aquisição de dados (retirada de [BITalino,])

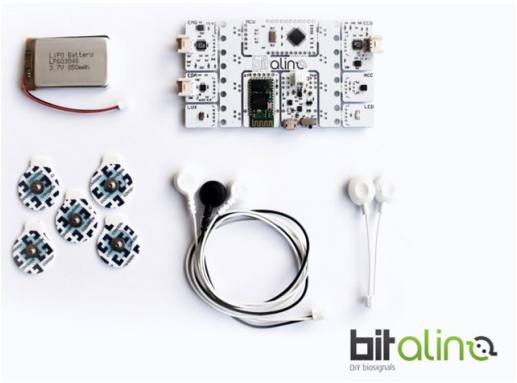

Figura 2.4: BITalino - O kit de desenvolvimento *low cost* português (retirada de [da Silva, 2012])

Capítulo 3

Solução Proposta

Neste Capítulo encontra-se descrito as características do projecto, como análise de requisitos, fundamentos, arquitectura do sistema , objectivos, bem como pontos críticos e abordagem à sua complexidade, fundamentais para este projecto.

3.1 Objectivos

De uma forma resumida este projecto tem como objectivo desenvolver um sistema com recurso a uma aplicação **Android** que permite obter dados fisiológicos e anotações que permitem referenciar e caracterizar, um contexto ou actividade que um utilizador esteja a realizar. Posteriormente os dados serão armazenados numa base de dados e disponibilizados para treinarem sistemas de aprendizagem automática. Os dados fisiológicos são adquiridos através de sensores ligados através Bluetooth Low Energy (BLE) [SIG, 2020], as anotações são fornecidas pelo utilizador na aplicação desenvolvida.

Este projecto pode ser divido em três partes, aquisição de dados, anotações e aprendizagem. Estas são partes fundamentais do projecto não deixando de focar que este projecto concentra-se na aquisição de dados e anotações.

3.1.1 Aquisição de Dados

A **aquisição de dados** tem como responsabilidade adquirir todos os dados disponibilizados pelo utilizador, esses dados podem vir do próprio smartphone ou através de *wearables* previamente seleccionados pela aplicação móvel.

Neste sentido é possível adquirir dados de varias fontes e não apenas dos que estão demonstrados pelos projecto demonstrado como trabalho futuro desta aplicação. Adquirir dados é visto de vários formas, seja a nível temporal ou geográfico, isto obriga a que os dados sejam etiquetados com metainformação. A figura 3.1 mostra o diagrama de aquisição desta aplicação.

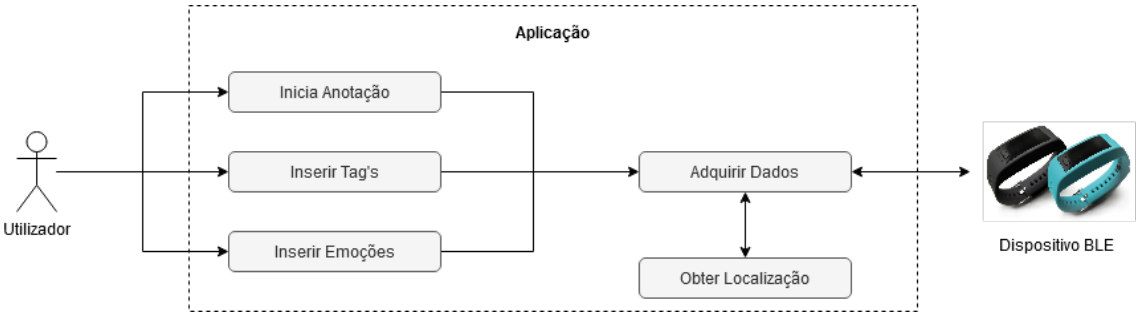

Figura 3.1: Aquisição de Dados

3.1.2 Anotações

O conceito de anotação é muito genérico e a abordagem seguida é de descrever um contexto através de palavras (conceito ou *tags*) e basear a anotação na associação de metainformação com a informação (dados). A utilização de *tags* para definir contextos não é recente, [Trattner et al., 2014] sendo anteriormente usada por utilizadores para associar categorias a recursos. Este tipo de anotação evolui para a criação de "*tag clouds*"onde as palavras são pesadas consoante o seu peso relativo [Garcia-Molina, 2011].

Figura 3.2: Tag Cloud

No contexto das anotações é necessário definir quais as emoções do utilizador. Nesse sentido para este projecto foi seguido o modelo *Arousel-Valence* [hcigames, 2000], que de uma forma gamificada permite entender as emoções do utilizador ao longo do tempo e espaço.Em contrapartida diferentes estímulos levam a emoções semelhantes que podem ser rapidamente distinguidas pelas *tags* de contexto. A figura 3.3 mostra um exemplo do modelo *Arousel-Valence* [hcigames, 2000].

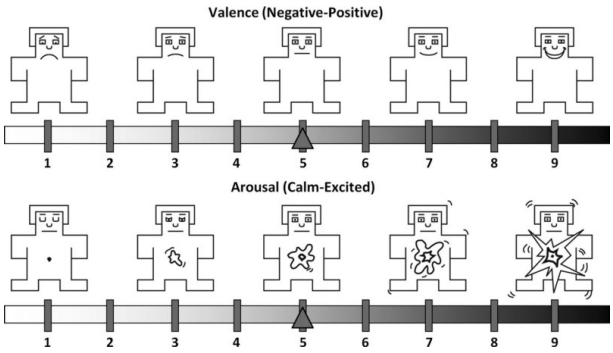

Figura 3.3: *Arousel-Valence*

3.1.3 Aprendizagem

Para o conceito de aprendizagem é necessário complementar os dados com mais informação, isto leva a que seja adicionado metainformação aos dados adquiridos, e isso pode incluir *tags*, localização GPS , actividade ,hora e data. Nesse sentido entramos no conceito de aprendizagem supervisionada onde os dados são etiquetados antes de serem utilizados para treinarem algoritmos. A figura 3.4 mostra um exemplo de estrutura dos dados.

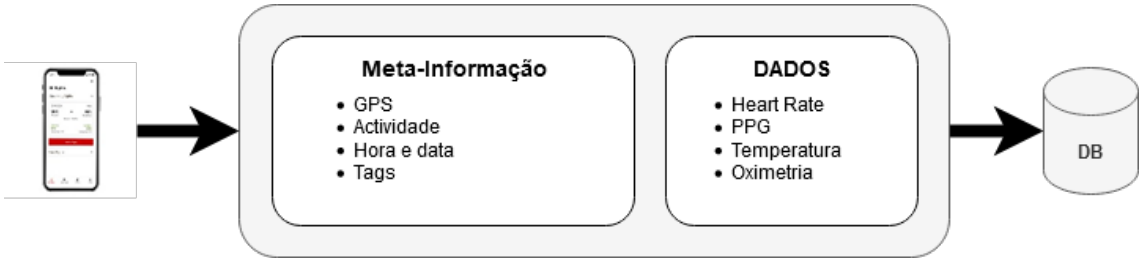

Figura 3.4: Meta-Informação

3.2 Análise de requisitos

A análise de requisitos é uma aspecto importante no desenvolvimento na gestão de projectos, que permite obter dados indispensáveis e exigências, para solucionar o problema descrito e alcançar os seus objectivos.

3.2.1 Publico alvo

Esta é aplicação pode ser utilizada na mais diversas áreas, em que o publico alvo é:

- Investigadores em projectos científicos;
- Engenheiros a desenvolver aplicações baseadas em sinais fisiológicos ou detecção de estados psicofisiológicos.

3.2.2 Definição de Requisitos

Esta secção demonstra todos as funcionalidades requeridas pelo sistema,e que demonstra todas funções do sistema, que representa o que o sistema deve fazer. Estas funções podem ser visíveis ou invisíveis para o utilizador, e que são fundamentais para funcionamento do sistema, estes requisitos estão descritos como requisitos gerias do sistema e de requisitos em segundo plano, como mostra nas tabelas 3.1 e 3.2.

Requisito	Função do sistema - O sistema tem de fazer	Categoria
R1.1	Registo Utilizadores	Visível
R1.2	Autenticação de utilizadores	Visível
R1.3	Iniciar anotações	Invisível
R1.4	Mostrar dados do utilizador	Visível
R1.5	Editar dados do utilizador	Visível
R1.6	Encontrar dispositivos BLE	Visível
R1.7	Definir Tag's pelo utilizador	Visível
R1.9	Sugerir contexto	Visível
R1.10	Criar contexto	Visível
R1.11	Seleccionar tag's e Emoções	Visível
R1.12	Terminar Anotações	Invisível
R1.13	Obter e inserir dados da base de dados	Invisível

Tabela 3.1: Requisitos gerais do sistema

Requisito	Função do sistema - O sistema tem de fazer	Categoria
R2.1	Ligar a dispositivos por BLE	Invisível
R2.2	Descodificar dados vindo BLE	Invisível
R2.3	Obter localização	Invisível
R2.4	Associar localização	Invisível
R2.5	Guardar dados na base de dados	Invisível
R2.6	Manter serviços activos em segundo plano	Invisível
R2.7	Notificar utilizadores	Invisível

Tabela 3.2: Requisitos de segundo plano

3.2.3 Definição de Atributos

Os atributos do sistema representam as qualidades não funcionais do sistema. Que podem ser defendias como características ou dimensões do sistema. No presente projecto encontra descrito os atributos na tabela 3.3.

Atributo	Detalhe / Restrição Fronteira	Categoria
Tempo de resposta < 2	Actualização dos dados	Obrigatório
Tolerância a falhas	Controlo e correção de erros do sistema	Obrigatório
Plataformas	Diferente versões Android e ecrãs	Desejável

Tabela 3.3: Atributos do sistema

3.2.4 Relação Funções e Atributos

È possível relacionar as funções com o atributos, Para tal segue a relação na figura 3.4.

Requisito	Função	Categoria	Atributo	Detalhe	Categoria
R1.1	Registo Utilizadores	Visível	Tolerância a falhas	Controlo e correcção de erros do sistema	Obrigatório
R1.2	Autenticação de utilizadores	Visível	Tolerância a falhas	Controlo e correcção de erros do sistema	Obrigatório
R1.3	Iniciar anotações	Invisível	Tolerância a falhas	Controlo e correcção de erros do sistema	Obrigatório
R1.4	Mostrar dados do utilizador	Visível	Plataformas	Diferente versões Android e ecrãs	Desejável
R1.5	Editar dados do utilizador	Visível	Tolerância a falhas	Controlo e correcção de erros do sistema	Obrigatório
R1.6	Encontrar dispositivos BLE	Visível	Tolerância a falhas	Controlo e correcção de erros do sistema	Obrigatório
R1.7	Definir Tag's pelo utilizador	Visível	Tolerância a falhas	Controlo e correcção de erros do sistema	Obrigatório
R1.9	Sugerir contexto	Visível	Plataformas	Diferente versões Android e ecrãs	Desejável
R1.10	Criar contexto	Visível	Tolerância a falhas	Controlo e correcção de erros do sistema	Obrigatório
R1.11	Seleccionar tag's e Emoções	Visível	Plataformas	Diferente versões Android e ecrãs	Desejável
R1.12	Terminar Anotações	Invisível	Tolerância a falhas	Controlo e correcção de erros do sistema	Obrigatório
R1.13	Obter e inserir dados da base de dados	Invisível	Tempo de resposta < 2	Actualização dos dados	Obrigatório
R2.1	Ligar a dispositivos por BLE	Invisível	Tolerância a falhas	Controlo e correcção de erros do sistema	Obrigatório
R2.2	Descodificar dados vindo BLE	Invisível	Tolerância a falhas	Controlo e correcção de erros do sistema	Obrigatório
R2.3	Obter localização	Invisível	Tempo de resposta < 2	Actualização dos dados	Obrigatório
R2.4	Associar localização	Invisível	Tempo de resposta < 2	Actualização dos dados	Obrigatório
R2.5	Guardar dados na base de dados	Invisível	Tempo de resposta < 2	Actualização dos dados	Obrigatório
R2.6	Manter serviços activos em segundo plano	Invisível	Tolerância a falhas	Controlo e correcção de erros do sistema	Obrigatório
R2.7	Notificar utilizadores	Invisível	Plataformas	Diferente versões Android e ecrãs	Desejável

Tabela 3.4: Relação funções e atributos

3.2.5 Casos de utilização

Os casos de utilização permitem descrever um conjunto de funcionalidades do sistema e suas associações com elementos externos. Desta forma cada caso de uso desenvolve uma narrativa de utilização, desenvolvida no projecto. Um dos possíveis diagramas deste projecto está representado na figura 3.5.

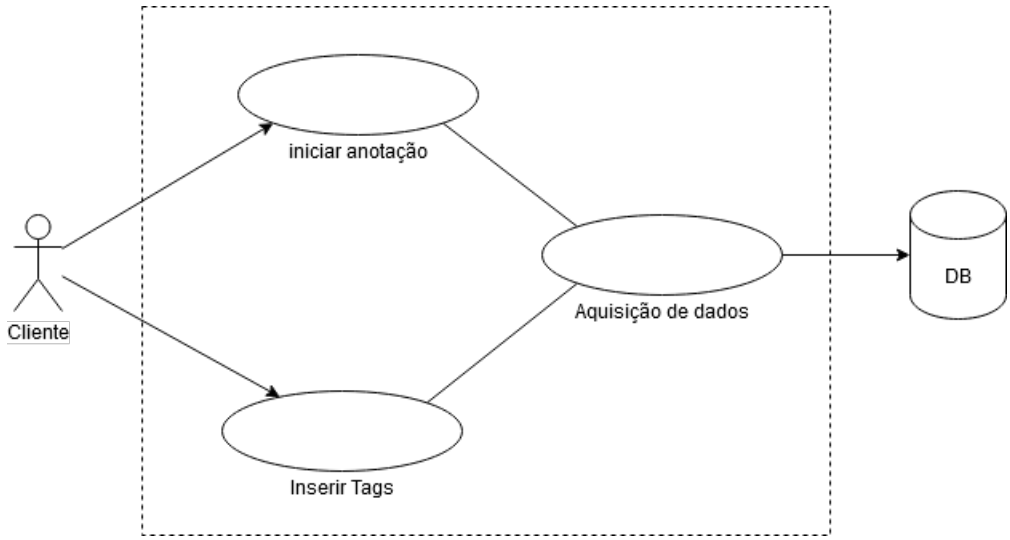

Figura 3.5: Casos de utilização

Cada um dos circulo corresponde a um caso de utilização, que são definidos por varias funções do sistema como se pode verificar nas tabelas 3.5, 3.6 e 3.7.

Caso de utilização **Iniciar Anotação**

Resumo	Utilizador inicia anotação
Referencias	R1.10, R1.1, R.2.1, R.2.7

Tabela 3.5: Caso de uso iniciar anotação

Caso de utilização **Aquisição de Dados**

Resumo	Utilizador inicia anotação
Referencias	R2.3, R2.2, R.24, R.2.5

Tabela 3.6: Caso de uso Aquisição de Dados

Caso de utilização **Inserir Tag's**

Resumo	Utilizador inicia anotação
Referencias	R1.7, R1.13, R.2.3, R.2.4, 2.5

Tabela 3.7: Caso de uso inserir tag's

3.3 Arquitectura

Como uma solução para o problema descrito neste projecto ,encontra-se na figura 3.6 um diagrama da arquitectura proposta para este projecto.

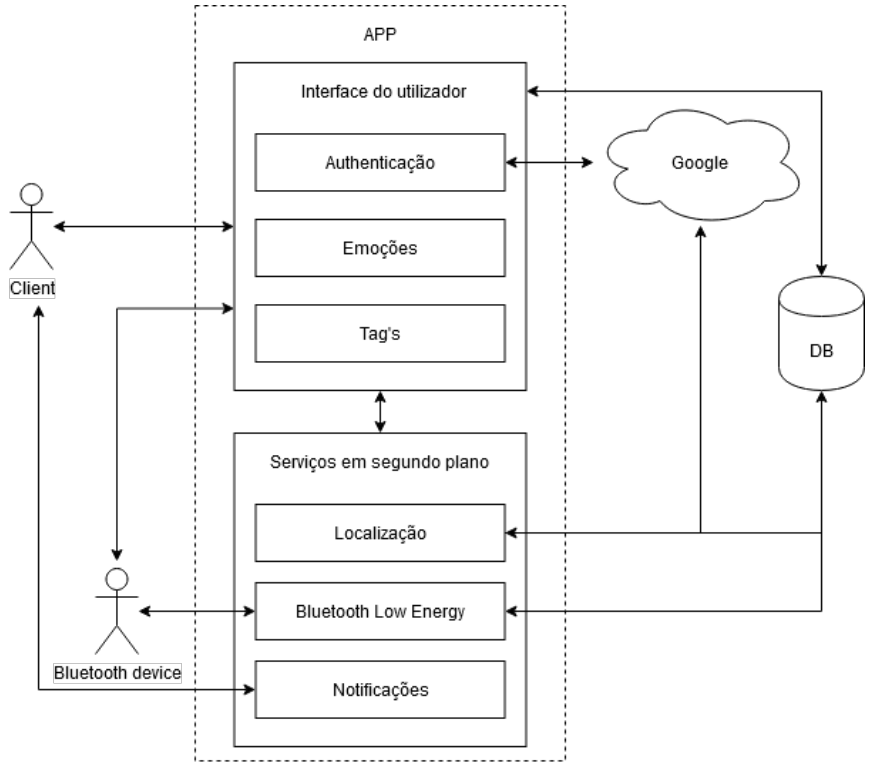

Figura 3.6: Arquitectura do sistema

A da arquitectura apresentada na figura 3.6, dispõe de uma aplicação, em que o utilizador interage com interface gráfica. Esta interface permite ao utilizador autenticar, registar , definir contextos ou actividades, interagir com os seus diapositivos **bluetooth**, inserir e definir tag's e emoções para um

determinado contexto e actividade. A seguir, existe um serviço em segundo plano que é responsável por obter a localização do utilizador , ligar e obter dados dos dispositivos **bluetooth** seccionados pelo utilizador e de notificar o utilizador caso esteja a anotar.

Esta arquitectura não se foca apenas em interagir com utilizador directamente, ou seja, o serviço em segundo plano permite que o utilizador não fique preso a interface gráfica e que possa desligar-se da aplicação. Desta forma o utilizador fica livre para o contexto ou actividade a desempenhar, ficando por trás um serviço que está adquirir dados fisiológicos e sua localização, dando a possibilidade ao utilizador de anotar quando achar necessário.

Fica responsável a interface gráfica e o serviço em segundo plano de enviar e obter dados da base de dados. Em complemento este sistema tira proveito de alguns serviços da google tal como localização e autenticação.

3.4 Tecnologias

Neste secção serão demonstrados os vários tipos de tecnologias utilizados neste projecto. Tais como linguagens de programação, ferramentas de design, base de dados entre outras. Este capítulo descrever também algumas vantagens e desvantagens das tecnologias utilizadas.

3.4.1 AdobeXD

O ***Adobe Experience Design***, [Inc, 2016] é uma aplicação desenvolvido pela pela ***Adobe Experience Design***. Esta é uma ferramenta de design , concentrada na experiência do utilizador e tem com propósito o desenvolvimento de interfaces *web* e aplicativos moveis.

Esta ferramenta para este projecto permite criar o *mockup* da aplicação, com o objectivo de criar um *layout* moderno e focado na experiência do utilizador. Para além disso esta ferramenta tem a possibilidade de visualizar toda a aplicação como se fosse um aplicativo real, dando a possibilidade de uma forma rápida demonstrar a aplicação. Esta aplicação permite também extrair os *layout's* produzidos para uma determinada linguagem, como por exemplo o ***Flutter***. No entanto estes *layout's* são produzidos de uma forma relacional e não responsivo. Do qual obriga a ser adaptado pelo programador para que seja adaptado a vários tipos de ecrã para diferentes dispositivos.

3.4.2 Flutter

O **Flluter**, [Flutter, 2017] é uma *framework* desenvolvida pela **Google** e que permite desenvolver aplicativos web e moveis, compilados de forma nativa, disponibiliza também um vasto conjuntos de bibliotecas para o desenvolvimento. Esta *framework* é composta pela seguintes componentes principais, que são:

- Linguagem de programação **Dart**;
- Flutter Engine;
- Biblioteca *Foundation*;
- *Design-specific Widgets* com implementações prontas para **Android** (Google Material) e **iOS** (Cupertino);

O **Dart** é um linguagem de *script* desenvolvida pela **Google**, bastante idêntica ao **JavaScript**, esta linguagem é depois compilada para código nativo através **Flutter Engine** desenvolvido em **C++**.

Neste projecto foi necessário definir uma linguagem de programação, o **Flutter** foi visto como a melhor alternativa ao projecto desenvolvido, porque facilita o desenvolvimento de interfaces moveis e poupa tempo no seu desenvolvimento.

Esta *framework* pode ser comparada com o **React Native**, [Facebook, 2015] e o **Ionic**, [Drifty, 2013]. No entanto como estas *frameworks* são código de alto nível impõe um risco de processamento. Como neste projecto é necessário utilizar o recurso Bluetooth do próprio equipamento, linguagens de alto nível tornam-se um desvantagem para esta implementação.

No entanto, o **Flutter** ao longo do projecto revelou-se ineficaz no que diz respeito em manter os serviços em segundo plano, por este motivo toda parte que entra em contacto com hardware do dispositivo móvel foi desenvolvido em **Kotlin** e a interface gráfica em **Flutter**.

3.4.3 Kotlin

O **Kotlin**, [JetBrainsc, 2010] é uma linguagem de programação multiplataforma, orientada a objectos e funcional, em 2017 foi considerada pela **Google** a linguagem oficial para o sistema **Android**.Neste projecto é necessário

manter serviços em segundo plano e resiliente a falhas, o **Kotlin** sendo uma linguagem nativa ajuda desenvolver aplicações que tem como recurso o hardware, e por esta razão foi utilizado para desenvolver todo sistema de segundo plano desta aplicação em comunicação com o **Flutter**.

3.4.4 Firebase

O **Firebase**, [Lee, 2010] é uma plataforma de desenvolvimento *web* e e aplicativos moveis, foi adquirida pela **Google** em 2014, e disponibiliza uma série de serviços de auxilio a implementação e gestão de aplicativos. Neste projecto são utilizados alguns serviços do **Firebase**, que são:

- Authentication;
- Cloud Firestore;
- Cloud Storage;

O serviço de *Authentication* possibilita a autenticação através de contas do Google, Facebook, Twitter, Github ou por um conjunto de contas definidos. Que nesta aplicação era servir para gerir as contas de utilizadores recorrento ao email e **Google**

O serviço *Cloud Storage* permite armazenar dados através de uma ligação segura e permite a partilha dos mesmo, como se fosse **CDN**. Este serviço é utilizado para guarda a imagem de perfil do utilizadores.

O serviço *Cloud Firestore* é uma Base de Dados **NoSQL** e que permite armazenar e sincronizar dados numa arquitectura cliente-servidor. Foi utilizado nesta aplicação como base de dados para guardar alguns dados do utilizador,aplicação, e todos os dados respectivos a notação como dados fisiológicos, geolocalização, dados do utilizador e de anotações.

Esta aplicação poderia ter utilizado uma base de dados relacional que poderia se justificar sendo uma mais valia para a aplicação.

3.4.5 Bluetooth Low Energy

O ***Bluetooth Low Energy*** (**BLE**) é uma técnica utilizada que apareceu em 2012 na versão 4.0 do **Bluetooth**, que apresenta apenas 10% dos gastos em comparação com o **Bluetooth**,e que permite que aplicações **Android** possam comunicar com diapositivos **BLE**.

A estrutura do **BLE** consiste *Generic Attribute Profile* (**GATT**) que é construído sobre o *Attribute Protocol* (ATT), e que disponibiliza uma serie de serviços definidos pelo próprio equipamento que esta transmitir os dados, estes serviços contêm características que podem ser do tipo *read*, *write* ou *notify*.

Estas características podem conter dados ou funcionalidades com por exemplo dados do ritmo cardíaco. Para além disto as características podem ser do tipo *Standard* ou *custom*. Os *Standard* corresponde a uma estrutura de dados conhecida, defendia e disponiblizada pela **Bluetooth**, [SIG, 2020], estes dados são muito importantes para descodificação dos dados que vêm do dispositivo. Por outro lado essas características podem ser *custom*, e dessa forma só o fabricante do dispositivo pode disponibilizar a toda a informação sobre a descodificação dos dados. A figura 3.7 representa a organização dos dados do perfil **GATT**.

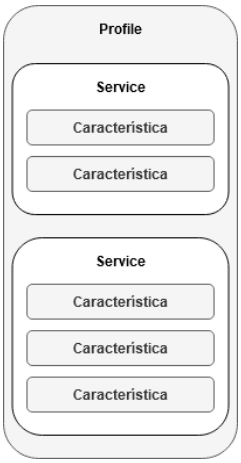

Figura 3.7: Perfil GATT

3.4.6 PulseON

A **PulseON**, [Hattula, 2012] é uma empresa especializada em *wearables* de *optical heart rate* **(OHR)**, baseado em fotoplestimografia (PPG), para os sectores médicos e de bem-estar. Nesta projecto foi decido utilizar um dos equipamento do **PulseON**, fornecido pela empresa CardioID Technologies [1], como diapositivo **BLE**. Este equipamento tem a vantagem de se pode

[1] `www.cardio-id.com`

adquirir dados como, frequência cardíaca instantanea (HR), a concentração de oxigénio (SPO2), actividades do utilizador (número de passos, calorias, entre outros), entre outras características. No entanto este equipamento no que que diz respeito ao **BLE**, não segue um forma standard ao que obriga a seguir a informação do fabricante para a descodificação das mensagens.

Figura 3.8: PulseOn

Capítulo 4

Interface do Utilizador

Este capítulo descreve toda a implementação referente a interface gráfica desenvolvida nesta aplicação. Parte da interface do utilizador é pensada através da experiência do utilizador, isto cria maior fiabilidade na aplicação desenvolvida. Toda a interface do utilizador foi desenvolvida em *Dart* com recurso a *framework Flutter*.

4.1 Mockup da Aplicação

Numa primeira abordagem a este projecto é necessário definir um modelo base da aplicação, para isso foi recorrido ao ***Adobe Experience Design***, [Inc, 2016]. Através desta aplicação foi desenvolvido um *mockup* da aplicação como um prototipo da aplicação.

Tendo em atenção que esta aplicação é focada na experiência, num primeira abordagem é recorrida a *layout's* já definidos e disponibilizados que por sua vez já contem diversas tendências da actualidade das aplicações mais modernas , como também diversos *Plugins* que facilitam o desenvolvimento do *mockup*, um excelente caso de exemplo são as aplicações da **Google** que são fundamentadas, testadas e completamente focadas na experiência do utilizador, dado como simples exemplo a aplicação do **Google Maps**,

Desta forma foi possível obter o *mockup* da aplicação representado na figura 4.1.

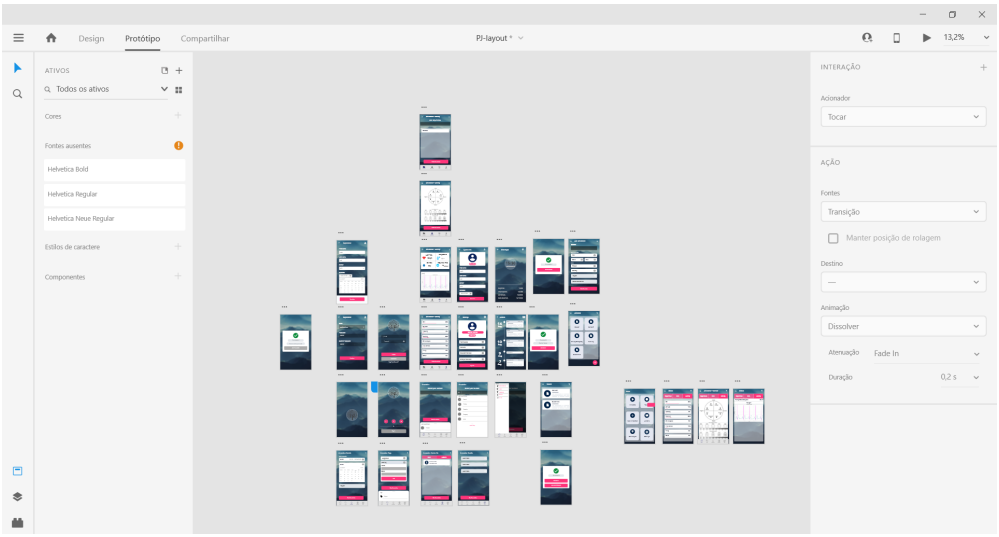

Figura 4.1: Mockup da Aplicação

4.2 Login

O **login** é importante nesta aplicação pelo o motivo de segurança e identificação do utilizador. No entanto na perspectiva da experiência do utilizador, pode tornar-se um complicação, por isso foi desenvolvido dois tipos de autenticação uma manual e outra automática.

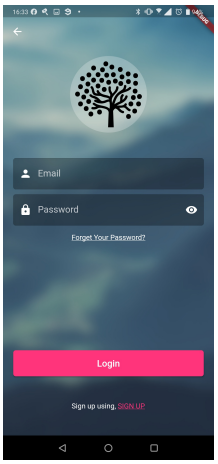

Figura 4.2: Login Manual

O **login** manual representado na figura 4.2, não apresenta uma boa perspectiva na ótica do utilizador. No entanto para o contexto do projecto fornece

mais informação, e essa informação é relevante na perspectiva dos dados fisiológicos, porque cada pessoa é diferente, e dando um exemplo do ritmo cardíaco uma pessoa com sessenta anos não tem o mesmo ritmo que uma pessoa de vinte anos. Logo toda a informação adicional é relevante para caracterização dos dados.

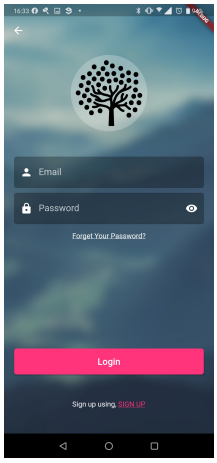

Figura 4.3: Login Automático

O **login** automático representado na figura 4.3 é apresentado com uma boa característica para o utilizador porque permite o utilizador de um forma rápida e prática estar registar e com a sessão iniciada, tentando também não desprezar a informação sobre utilizador.

Estes *Layout's* foram desenvolvidos através do **Flluter**, tendo respeitar ao máximo o *layout* imposto no *mockup*. Estes *layout's* tiram partido do serviço de autenticação do **Firebase** que permite criar e registar utilizadores, mas também integrar com as redes sociais. É de focar que neste projecto como prova de conceito foi apenas implementado para o **Gmail** sendo que a restante implementação como o **Facebook** e o **Twitter** seria o mesmo fundamento apenas teria de ser respeitadas as normas definidas de cada rede social, deixando esta tarefa como trabalho futuro deste projecto.

4.3 Registo

O **registo** permite ao utilizador criar um utilizador de forma manual como é apresentado na figura 4.3. Este *layout* foi desenvolvido através do **Flluter** e tira novamente partido do serviço de autenticação do **Firebase** que permite criar utilizadores. Este Serviço já inclui um série de validação de dados antes de criar o utilizador.

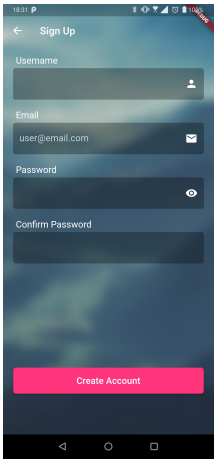

Figura 4.4: Registo Manual

No entanto sendo esta arquitectura cliente-servidor, tem que ser validado também no lado do cliente. Por esse motivo, este *layout* contem uma serie de alertas, que permitem informar o utilizador de eventuais erros que possam surgir na inserção dos dados, garantindo assim que os dados são enviados de forma correta para o servidor. Parte dessas validações são feitas através de expressões regulares.

4.4 Recuperação da password

A **recuperação da password** permite ao utilizador definir uma nova password através do email, ou seja o utilizador cola o seu email de utilizador e de seguida recebe um email para definir uma nova password. O email que é enviado é predefinido no serviço de autenticação do **Firebase** e contém um link para a difinição de uma nova password. Esta funcionalidade permite também alertar o utilizador caso o email não exista e respectivos erros que

possam surgir por parte do utilizador. O respectivo *layout* e recuperação encontra-se representado na figura 4.5.

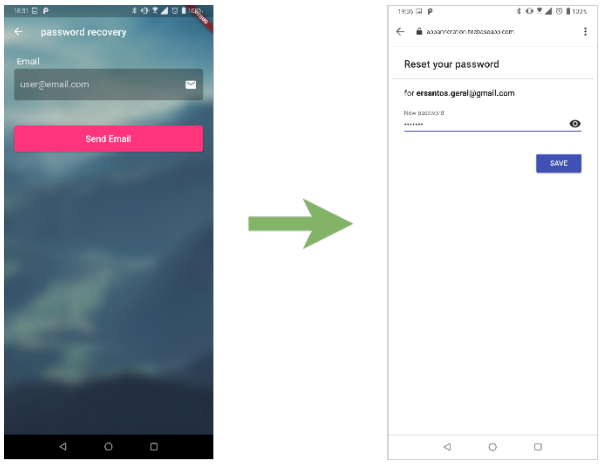

Figura 4.5: Recuperação de Password

4.5 Perfil do utilizador

O perfil do utilizador do utilizador tem como principal objectivo recolher informação do utilizador, essa informação é útil para o relacionamento dos dados filológicos adquiridos.

Deforma a melhorar a experiência do utilizador com este *layout* foi adicionado varias funcionalidades que permitem alterar e visualizar são:

- Foto de perfil;
- Username;
- Primeiro Nome;
- Ultimo Nome;
- Email;
- Password;
- Género;
- Data de nascimento;

Esta funcionalidades tem um controlo de dados vindo utilizador através de expressões regulares e de validações e tirando proveito dos serviços de autenticação e *storage* do **Firebase**. A figura 4.6 demonstra o *layout* descrito.

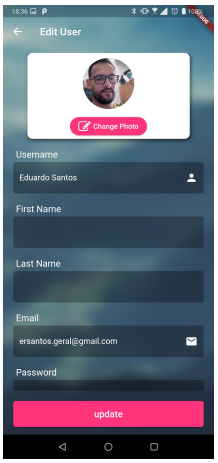

Figura 4.6: Perfil do utilizador

4.6 Home Page

Este *layout* é o principal da aplicação que permite, iniciar anotação, definir o contexto, inserir *tags* para a anotação e seleccionar os dispositivos **BLE**. Este *layout* foi divido em três partes que são:

- **Home** - Responsável pela definição do contexto e pelo inicio da notação, contem um conjunto de anotações já predefino com sugestão, e as ultimas anotações;
- **Tags** - Responsável pela definição de *tags* para o contexto depende do *layout home* para ter referencia *tags* definidas anteriormente;
- **Devices** - Responsável por encontrar dispositivos **BLE**, apenas aparecem dispositivos que são compatíveis com a aplicação;

É relevante salientar que o utilizador para iniciar uma anotação pode apenas necessitar de um click para começar a anotar. Com base na experiência do utilizador este Layout encontra-se de alguma forma facilitado no que diz

respeito a colocação de dados, no caso do utilizador se o mesmo utilizar sempre os mesmos dispositivos o mesmo só tem seleccionar uma única vez. A figura 4.7 demonstra actividade principal da aplicação.

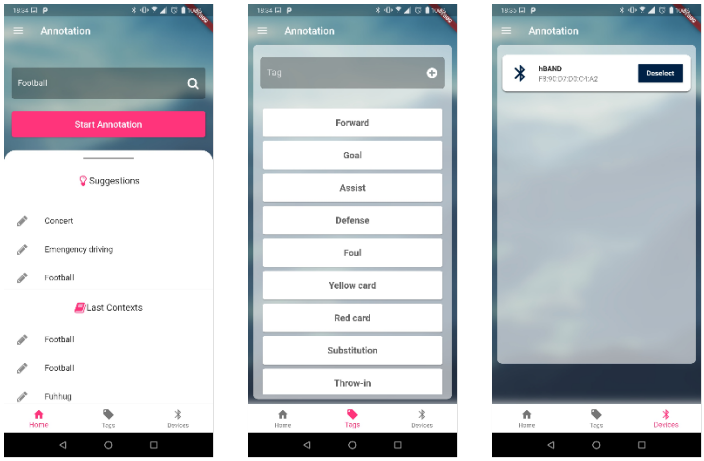

Figura 4.7: Layout principal da aplicação

4.7 Anotações

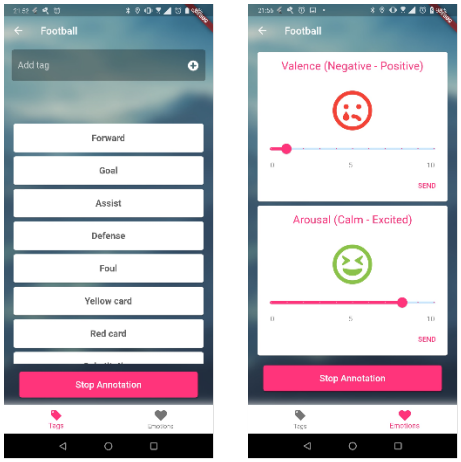

Figura 4.8: Layout de anotação

Este é o *layout* onde se começa anotar, por traz destes layout encontrasse um serviço em segundo plano esse serviço só irá terminar quando o utilizador terminar anotação, este permite que se o utilizador fique sem bateria ou por

alguma razão saia da aplicação possa retornar a anotação sem perder os dados.

Este *layout* concentra-se em colocação de *tags* por isso foi divido em duas partes, uma para as *tags* de contexto e outras para as *tags* emocionais, aqui pretende-se os inputs por parte do utilizador, com a definição do que esta acontecer e de como se sente no contexto da anotação. A figura 4.7 o demonstra o respectivo *layout*.

Capítulo 5

Anotações

Este capítulo pretende descrever toda implementação e abordagem nas anotações, no âmbito das *tags* de contexto, emocionais e outra metainformação relevante (Posição GPS e actividade).

5.1 Emocionais

Ao longo do dia todos sentimos varias emoções, sejam elas positivas ou negativas. Há dias mais calmos ou mais agitados, a grande questão sobre a qual este projecto se depara é como podemos quantificar essas emoções.

A categorização de emoções tem vindo a ser estudada em várias áreas cientificas, desde a psicologia à engenharia informática (affective computing). Neste projecto foi utilizado o modelo *Arousal-Valence* [Posner, 2005].

Este modelo permite quantificar em duas dimensões (x: y:), a emoção do utilizador. Na aplicação desenvolvida estas dimensões foram separadas em duas dimensões independentes e quantificadas em 10 níveis. Desta forma é possível "gamificar"este pedido ao utilizador, como podemos ver na figura 5.1.

Ao juntar os dados podemos ter uma representação a duas dimensões (Figura 5.2) e com isso é possível caracterizar o estado emocional do utilizador. Se for acrescentado meta-informação como data, hora e localização, podemos ver ao longo do tempo e espaço o estado emocional do utilizador.ar Neste projecto foi devolvido um *layout* responsável pela aquisição das emoções, que representa um quantificador a uma dimensão para saber o se o utilizador está num estado negativo ou positivo, e se está calmo ou excitado. Todos estes

Figura 5.1: Emoções

dados são guardados e etiquetados com localização e um *timestamp*.

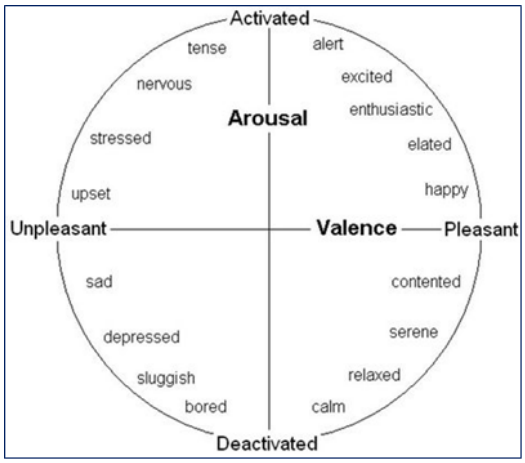

Figura 5.2: Arousal-Valence duas dimensões

5.2 Contexto

No contexto da actividade, não é possível definir o que desperta as emoções. Como por exemplo ir correr, tropeçar numa pedra e ficar num estado emocional negativo, agora o estado emocional é referente a actividade ou a evento que surgiu? Ou seja diferentes estímulos levam a emoções semelhantes para resolver este tipo de problema é necessário outro tipo de anotações, que são as anotações de contexto. que representa o que está a acontecer. Por outro lado pedir ao utilizador para escrever tudo o que acontece não é exequível,

para isto é utilizado conjunto de tag's. Na aplicação é definido um conjunto de palavras pelo utilizador e algumas já são predefinidas nos contextos já previamente definidos pela aplicação, para um determinado contexto, essas palavras entram na definição de *"Tag cloud"*. No entanto para poder quantificar pesos a essas palavras é necessário outro tipo de abordagens que fica definido com trabalho futuro desta aplicação.

Nesta aplicação foi desenvolvido um *layout* responsável pela tag's essas são seleccionadas pelo utilizador e podem ser acrescentadas ao longo da anotação pelo utilizador, isso permite de um forma simples e eficaz que o utilizador seleccione *tags* ao longo da sua anotação. A figura 6.2 representa a comparação de uma *"Tag cloud"* com o que foi desenvolvido na aplicação.

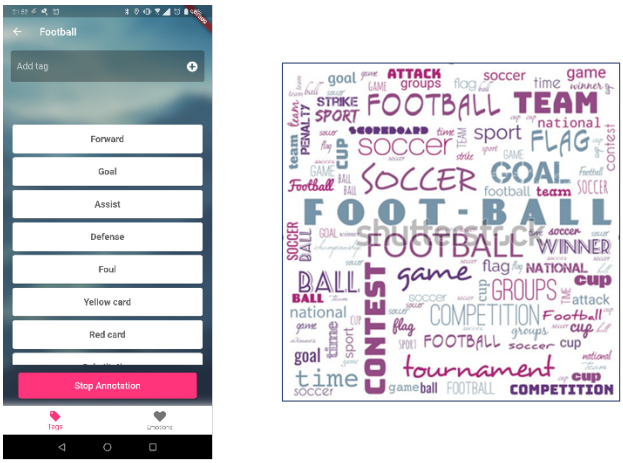

Figura 5.3: Tag cloud comparação com a aplicação

5.3 Meta-informação

A **meta-informação** consiste etiquetar a informação base com mais informação, de forma a ajuda a entender e a relacionar os dados e impor alguma estrutura. Esta estrutura permite ainda aumentar a consistência da informação adquirida.

Todos os dados enviados para a base de dados são adicionados com meta-informações. Desta forma é possível relacionar a informação proveniente dos sensores com *tags*, emoções, data, hora e localização geográfica, dados que referenciam uma anotação num determinado contexto.

Desta forma os dados ficam preparados para serem usados pelos diversos algoritmos de aprendizagem automática.

A figura 5.4 demonstra a estrutura de dados da anotação.

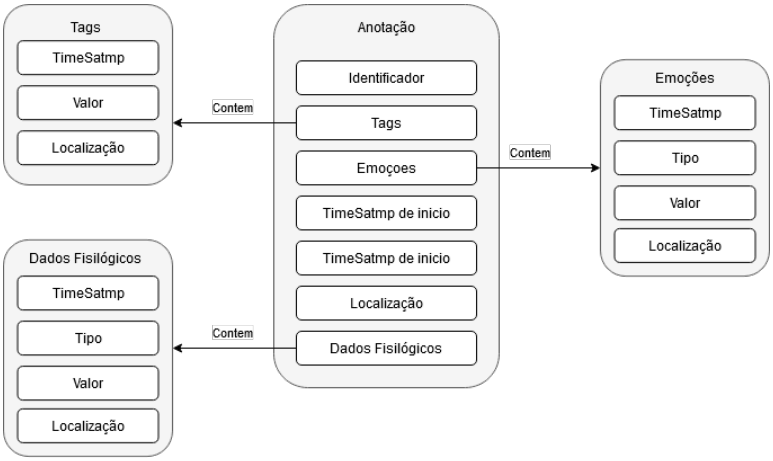

Figura 5.4: Tag cloud comparação com a aplicação

Capítulo 6

Serviço em segundo Plano

Este capítulo pretende descrever toda a implementação e abordagem do serviço em segundo plano, descreve também como foi desenvolvido a aquisição do sensor através do **BLE**, dados de Localização vindos do telemóvel, notificações ao utilizador e a comunicação entre o **Flutter** e o **Kotlin**.

6.1 Lógica

A necessidade de criação de um serviço em segundo plano prende-se pelo requisito de adquirir dados dos sensores mesmo quando a aplicação principal está é terminada.

O serviço em segundo plano consiste num serviço implementado em **Kotlin** que fica a executar como se de uma segunda aplicação se trata-se, sendo responsável apenas pela comunicação com o sensor através do **BLE**, da aquisição de Localização (GPS)e notificações ao utilizador. A figura 6.1 representa o diagrama de comunicação quando é iniciado uma anotação.

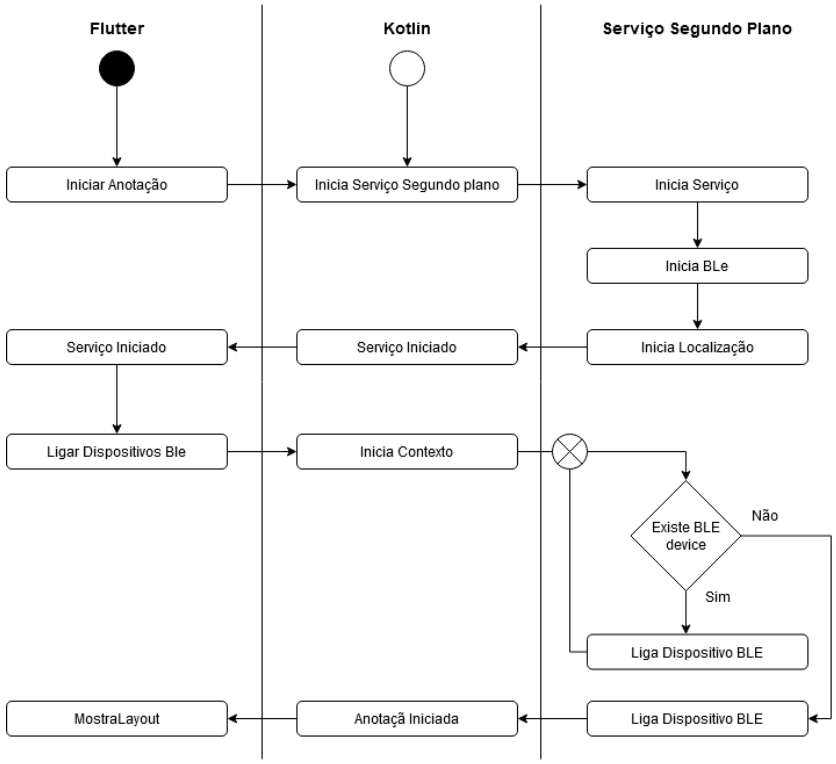

Figura 6.1: Inicio da anotação comunicação Flutter e Kotlin

Para este serviço se estar/manter-se activo necessita de:

- Manter-se activo quando o processo pai é terminado;
- Iniciar-se quando quando telemóvel é ligado;

Para que isto seja possível é necessário dois tipos de configurações que pertencem ao ciclo de vida de um serviço em **Android** [android, 2020].

Para manter o serviço activo é necessário informar o sistema **Android** que o serviço de segundo plano tem de se manter activa após aplicação pai ser fechada. Para isso é utilizado o comando **START STICKY** que matem o serviço activo quando o processo pai termina. A figura 6.2 mostra o código responsável por esta acção.

Quando o telemóvel inicia a primeira vez é necessário executar o serviço. A solução parte de um **BroadcastReceiver** esta acção é configurada no **AndroidManifest** e aplicado um filtro para que quando o equipamento estiver pronto a ser utilizado após ser ligado execute este **BroadcastReceiver**, que

recebe uma mensagem vinda do sistema operativo que o **BOOT** do sistema está concluído e executa o serviço pretendido. As figuras 6.3 e 6.4 mostra o código responsável por esta acção.

```
override fun onStartCommand(intent: Intent?, flags: Int, startId: Int): Int {
    return super.onStartCommand(intent, flags, startId)
    return START_STICKY
}
```

Figura 6.2: Comando START STICKY

```
<receiver android:name=".background.StartActivityOnBootReceiver">
    <intent-filter>
        <action android:name="android.intent.action.BOOT_COMPLETED" />
        <action android:name="android.intent.action.QUICKBOOT_POWERON" />
    </intent-filter>
</receiver>
```

Figura 6.3: Configuração no AndroidManifest

```
override fun onReceive(context: Context?, intent: Intent?) {
    if (Intent.ACTION_BOOT_COMPLETED == intent!!.action) {
        val forService = Intent(context, BackgroundService::class.java)
        if(Build.VERSION.SDK_INT >= Build.VERSION_CODES.O){
            context!!.startForegroundService(forService)
        }else{
            context!!.startService(forService)
        }
    }
}
```

Figura 6.4: BroadcastReceiver

Para além destes passos é necessário existir comunicação entre o **Flutter** e o **Kotlin**, para isso foi desenvolvido um canal de comunicação através do **FlutterEngine**, este canal permite criar funcionalidades e comunicar serviço que se encontra em segundo Plano. As figuras 6.6 e 6.5 representam essa comunicação desenvolvida.

```
Future<String> startService(String annotation) async{
  String data = await _methodChannel.invokeMethod("startService",{
    "annotation":annotation,
  });
  debugPrint(data);
  return data;
}
```

Figura 6.5: Canal de comunicação Flutter

```
override fun configureFlutterEngine(@NonNull flutterEngine: FlutterEngine) {
    GeneratedPluginRegistrant.registerWith(flutterEngine)

    MethodChannel(flutterEngine.dartExecutor.binaryMessenger, name: "com.a40610.messages")
        .setMethodCallHandler{methodCall, result ->
            when(methodCall.method){
                "startService" -> startService(methodCall,result)
                "stopService" -> stopService(methodCall,result)
                "connect" -> connect(methodCall,result)
                "disconnect" -> disconnect(methodCall,result)
                "startAnnotation" -> startAnnotation(methodCall,result)
                "getIdLocation" -> getIdLocation(methodCall,result)
            }
        }
}
```

Figura 6.6: Canal de comunicação Kotlin

6.2 Notificação

É fundamental informar o utilizador da existência de um serviço em segundo plano, este serviço apenas é executado quando existe alguma anotação a decorrer, essa é gerida pelo serviço em segundo plano e mantém uma notificação activa para para ser visualizada pelo utilizador, isto permite abrir abrir a aplicação caso esta fechada e informa que existe uma notação a decorrer.

Esta funcionalidade podia dar mais informações ao utilizador como incentivar a anotação e inserção de *tags*, isto poderia surgir através de alterações por parte dos dados fisiológicos, no entanto foi deixado como trabalho futuro desta aplicação.A figura 6.7 mostra essa notificação.

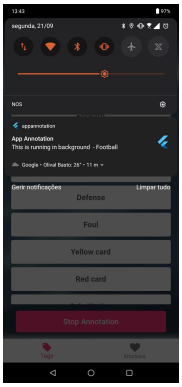

Figura 6.7: Notificação

6.3 Localização

A localização no **Android** é um serviço oferecido pela *Google Play Services* e *fused location provider* [the fused location provider, 2020]. Este serviço é executado através do serviço em segundo plano, e que através de um *listener* permite obter a a localização **GPS** do equipamento.

Com esta funcionalidade é possível associar a localização a todos os dados adquiridos, o objectivo é adicionar meta-informação aos dados e nessa informação tem que conter a localização geográfica.A figura 7.9 representa uma localização adquirida.


```
accuracy: 70.43999481201172
altitude: 80.01641845703125
bearing: 0
bearingAccuracyDegrees: 0
complete: true
elapsedRealtimeNanos: 2839263872883133
extras
   classLoader: null
   empty: false
   parcelled: true
   size: 96
fromMockProvider: false
latitude: 38.79072383
longitude: -9.17858224
provider: "gps"
speed: 0
speedAccuracyMetersPerSecond: 7.729068279266357
time: 1600692303000
verticalAccuracyMeters: 98.08731842041016
```


Figura 6.8: Estrutura de dados da localização

6.4 Bluetooth Low Energy

O ***Bluetooth Low Energy*** é uma das principais funcionalidades desta aplicação, este serviço executado em segundo plano, e tem com objectivo ligar a dispostos **BLE** e obter dados ou funcionalidades do equipamento.

O **Bluetooth** não é um tecnologia recente e existe muitos fabricantes dos mais diversos equipamentos. Por esse motivo vários equipamentos comportam-se de maneira, este tipo aplicações tem que ter a capacidade de se adaptar aos vários dispositivos, isso implica que exista algum estudo sobre os equipamentos alvo bem como a descodificação das mensagens vindas desse equipamento.

6.4.1 Comunicação

Nesta aplicação é necessário detectar e comunicar com equipamentos **BLE**. Essa detecção de equipamentos é feita através de *broadcast*, ou seja o equipamento **BLE** que fornece constantemente um *Advertisement*, que contem informações do dispositivos, como nome endereço MAC entre outras opções disponíveis.Estes dados podem ser lidos por outros equipamentos que estejam na sua área de alcance , e desta forma é possível detectar o equipamentos que estão na área. A figura 6.9 representa o diagrama referente a descrição

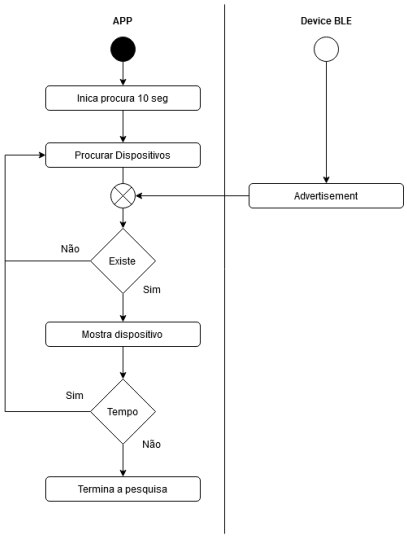

Figura 6.9: Procura de equipamentos BLE

Com base no endereço MAC do equipamento **BLE** é possível detectar e solicitar a ligação. A partir destes ponto a ligação entre o equipamento **BLE** fica estabelecida.

O **Android** a partir da versão 18 integra um plataforma de BLE para este tipo de configurações, Em que neste projecto foi utilizado **Flutter** para detecção de dispositivos moveis, que acaba por utilizar a mesma plataforma, e o **Kotlin** para comunicação e aquisição dos dados vindo dos equipamentos.

6.4.2 Descodificação

A tarefa de ligar aos dispositivos **BLE** é bastante simplificada, mas no que diz respeito a descodificação dos dados vindos desse equipamentos é uma situação diferente. A estrutura do **BLE** consiste num perfil **GATT**, esse perfil contem serviços, e dentro dos serviços características, essas características podem ser do tipo *read*, *write* ou *notify*, ou até mesmo ambos.

A figura 6.10 a baixo representa um diagrama numa subscrição a uma característica.

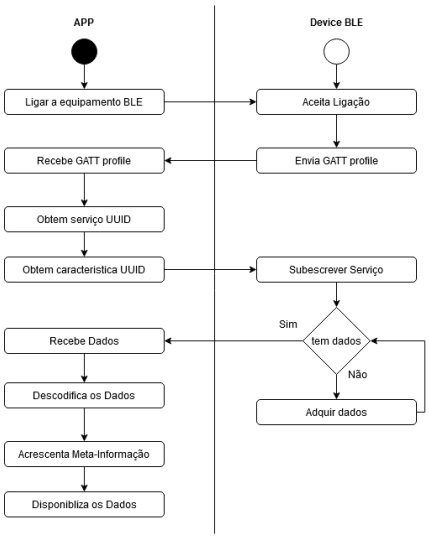

Figura 6.10: Subscrição a uma característica

Os dados que vem nessas características obrigam a descer a nível do bit, e encontram-se acoplados e podem vir codificados. A Bluetooth [SIG, 2020] disponibiliza vários serviços e características *standard*, que ao serem respeitas com os **UUID** correctos e com a codificação correta, estaria disponível

para todos os equipamento de uma forma universal. Mas tal situação não se verifica a maioria dos equipamentos aborda apenas alguns desses serviços *standard* e desenvolve o seus próprios serviços que obriga a saber a descodificação desses dados através do *DataSheet* desses equipamentos.

Neste projecto utilizamos como prova de conceito o **PulseOn**, mas como queremos ter mais diapositivos foi criado um sistema que possa agregar novos equipamento. Isto na óptica do programador permite associar vários equipamentos tendo apenas que implementar a codificação necessária, se por exemplo a descodificação for standard é reaproveitada para todos os equipamentos e isso permite concentrar apenas nos serviços que não são *standard* e desenvolvidos pelo próprio fabricante.

Capítulo 7

Validação e Testes

Neste capitulo pretende-se descreve alguns testes e validações realizadas na aplicação no que diz respeito a aquisição de dados.

7.1 Aquisição de dados

È importante garantir que os dados são adquiridos da melhor forma possível. Para isso é necessário testar desde da descodificação dos dados até ao resultado final, a tabela demonstra o numero total de dados adquiridos ao longo de um contexto executado

Parâmetros	Dados
Context	football
UUID	0dbee54a-ec13-4bc6-ab34-1fcefcf11dff
Time	843 seconds
Tags	20
Emotions	4
Activity	97
br	1
hrm	857
hrv	1102
ppg	1397
location	99

Tabela 7.1: Teste a aquisição de dados

As figuras abaixo demonstram a estrutura dos dados adquiridos:


```
context: "Football"
emotions: [{locationId: "12", timest...]
end: "2020-09-22 07:18:25.557603Z"
running: false
start: "2020-09-22 07:04:22.233011Z"
tags: [{locationId: "12", tag: "...]
userId: "PxJgXYbtZOTEJs8GPdzXH5tQ6sT2"
uuid: "0dbee54a-ec13-4bc6-ab34-1fcefcf11dff"
```


Figura 7.1: Estrutura da anotação


```
locationId: "12"
tag: "goal"
timestamp: "2020-09-22 07:09:11.696014Z"
```


Figura 7.2: Estrutura da tag


```
locationId: "83"
timestamp: "2020-09-22 07:14:57.747342Z"
type: "arousel"
value: 8
```


Figura 7.3: Estrutura da emoção

```
accuracy: 70.43999481201172
altitude: 80.01641845703125
bearing: 0
bearingAccuracyDegrees: 0
complete: true
elapsedRealtimeNanos: 2839263872883133
extras
   classLoader: null
   empty: false
   parcelled: true
   size: 96
fromMockProvider: false
latitude: 38.79072383
longitude: -9.17858224
provider: "gps"
speed: 0
speedAccuracyMetersPerSecond: 7.729068279266357
time: 1600692303000
verticalAccuracyMeters: 98.08731842041016
```


Figura 7.4: Estrutura da localização


```
idLocation: 15
level: 50
timestamp: "2020-09-22T07:11:31.923Z"
type: "Battery Level"
```


Figura 7.5: Estrutura da bateria


```
idLocation: 23
ppg: 16347
time: 39676
timestamp: "2020-09-22T07:12:10.362Z"
type: "ppg"
```


Figura 7.6: Estrutura do PPG

```
energyExpended: ""
heartRateValue: 89
idLocation: 0
rrInterval: ""
sensorContact: 0
timestamp: "2020-09-22T07:04:27.258Z"
type: "Heart Rate Measurement"
```


Figura 7.7: Estrutura do HRM


```
hRQuality: 27
hRReliable: 0
hRatTimeout: 0
hr: 90
ibi: 0
ibiQuality: 0
idLocation: 12
newIBIData: 0
rollingCounter: 522596
timestamp: "2020-09-22T07:09:06.195Z"
type: "Heart Rate Measurement"
```


Figura 7.8: Estrutura do HRP

```
activity_data_changed: 1
activity_type: 0
distance: 2
distance_data_present: 0
idLocation: 0
k_calories: 8062
rollingCounter: 373409
sleep_data_present: 0
speed: 0
speed_data_present: 0
steps: 6
swim_data_present: 0
timestamp: "2020-09-22T07:04:33.258Z"
type: "activity"
workout_state: 0
workout_status_changed: 0
worn_state: 1
worn_status_changed: 1
```


Figura 7.9: Estrutura do actividade

Capítulo 8

Conclusões e Trabalho Futuro

Este projecto implementou uma solução de aquisição e anotação de dados fisiológicos baseada numa aplicação móvel. Todos os dados adquiridos são agregados numa base de dados, implementada no **Firebase**, onde é possível visualizar os dados adquiridos por cada anotação.

A aquisição e anotação de dados fisiológicos revelou ser um tarefa desafiante, onde é necessário uma grande quantidade de dados e esforço de anotação por parte do utilizador. Existe uma grande panóplia de sensores com capacidade de adquirir diferentes dados fisiológicos, no entanto os protocolos variam bastante entre equipamentos. No entanto, no caso do equipamento da **PulseOn**, foi possível descodificar todos os dados fornecidos pelo sensor.

Relativamente à tarefa de anotação, o projecto implementou duas perspectivas: emoções e e contexto. A aplicação permite obter estas duas perspectivas por parte do utilizador, permitindo criar consistência com os dados obtidos através da adição de meta-informação. Com isto é possível relacionar os dados através do tempo localização e também contexto em que se encontra, mas também detectar factores externos que possam comprometer os dados.

Esta aplicação foca-se muito no serviço em segundo plano que retira ao utilizador a responsabilidade de adquirir os dados fisiológicos, sendo total responsabilidade da aplicação.

É necessário focar que todos os dados que são adquiridos através da aplicação ainda são reduzidos. O serviço em segundo plano tem capacidade para ligar a diversos equipamentos em simultâneo e ainda é possível obter mais dados do telemóvel do utilizador, ficando esta uma tarefa para trabalho futuro.r

A visualização dos dados seria algo implementar com trabalho futuro, uma abordagem que poderia ser utilizada, prendia-se com a utilização do **Jupyter** e a linguagem **Python** para implementação da aprendizagem automática.

Na perspectiva do utilizador aplicação tem muito ainda por desenvolver, anotar é uma tarefa difícil para o utilizador e aplicação tem que ser totalmente apelativa e funcional para o utilizador em que possivelmente a melhor abordagem seria através de um jogo.

O **BLE** é um tecnologia ingrata , existem inúmeros fabricantes e muitas vezes comporta-se de maneira diferente, é necessário ter uma aplicação estável nesse sentido, focar realmente nos equipamentos que são necessários para aquisição de dados e na compatibilidade com os vários dispositivos móveis.

Apêndice A

Estrutura da descodificação

Neste apêndice é representado o diagrama referente a implementação da descodificação dos dados vindos do PulseOn. Este foi desenvolvido em **Kotlin** e pretende demonstrar que aplicação pode ser universal para outro tipo de de dispositivos.

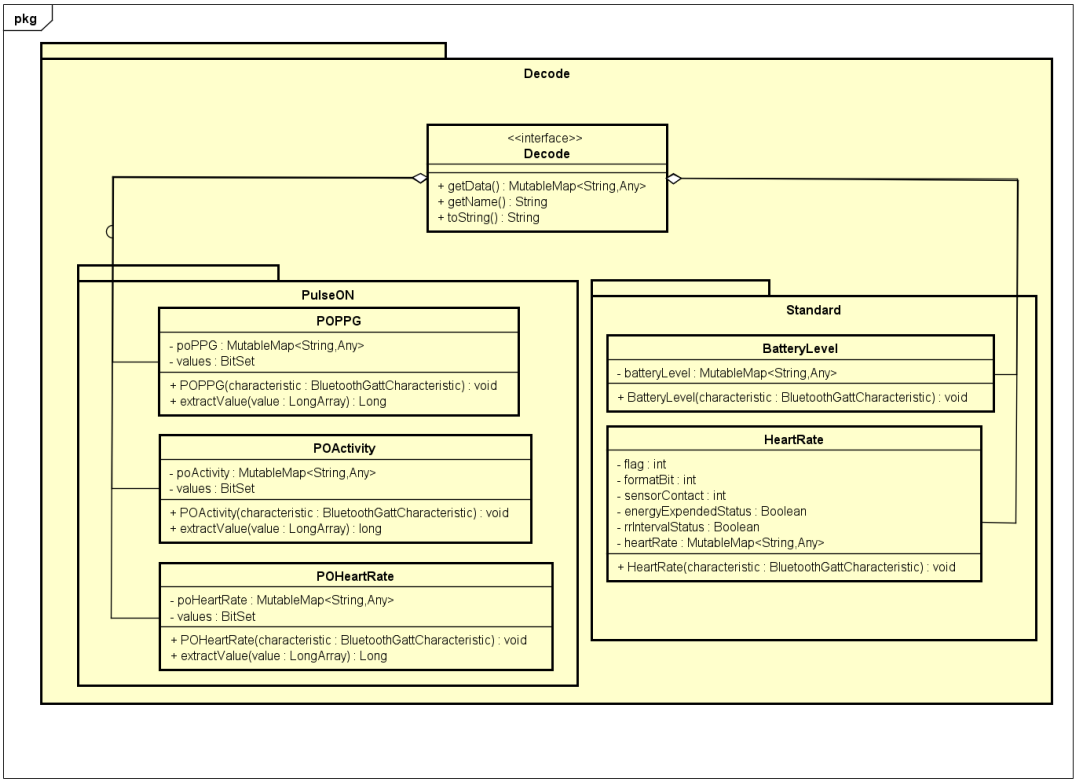

Figura A.1: Decode

Apêndice B

Tabelas de Descodificação

Para poder entender os dados que vem das mensagens **BLE**, que são transmitidas pelo PulseOn é necessário recorrer ao seu *data sheet*, as tabelas a baixo mostra como os dados estão organizados para diferentes características.

Fields	Values	Data Type
Rolling Counter	Integer	Uint32
HR Reliable	Booolean	1 bit
HR at Timeout	Booolean	1 bit
New IBI Data	Booolean	1 bit
HR	Integer	Uint8
HR Quality	Integer	Uint8
RESERVED		Uint16
RESERVED		Uint8

Tabela B.1: Heart Rate Data Characteristic

Fields	Values	Data Type
timestamp	Integer	UInt16
ppg	Integer	UInt16

Tabela B.2: PPG Data Characteristic

Fields	Values	Data Type
RESERVED		4 bits
Workout Status Changed	Booolean	1 bit
Worn Status Changed	Booolean	1 bit
Activity data changed	Booolean	1 bit
Rolling Counter	Integer	Uint32
kCalories	Integer	Uint24
Speed	Integer	Uint16
Steps	Integer	Uint24
Distance	Integer	Uint24
Activity Type	Integer	4 bits
Workout State	Booolean	1 bit
Worn State	Booolean	1 bit

Tabela B.3: Activity Data Characteristic

Bibliografia

[android, 2020] android (2020). service-lifecycle.

[BITalino,] BITalino. Opensignals (r)evolution. https://www.bitalino.com.

[Bronzino, 2000] Bronzino, J. D. (2000). *The Biomedical Engineering Handbook: Second Edition.* CRC.

[da Silva, 2012] da Silva, H. P. (2012). Bitalino. https://www.bitalino.com.

[Drifty, 2013] Drifty (2013). Ionic.

[Facebook, 2015] Facebook (2015). React native.

[Flutter, 2017] Flutter (2017). Intro to bluetooth low energy.

[Garcia-Molina, 2011] Garcia-Molina, P. V. . G. K. . H. (2011). On the selection of tags for tag clouds. *Proceedings of the Forth International Conference on Web Search and Web.*

[Hattula, 2012] Hattula, J. (2012). Pulseon.

[hcigames, 2000] hcigames (2000). Arousel-valence.

[Inc, 2016] Inc, A. (2016). Adobe experience design.

[Jesus et al., 2010] Jesus, R., Abrantes, A. J., e Correia, N. (2010). Methods for automatic and assisted image annotation. *Springer Science+Business Media, LLC 2010.*

[JetBrainsc, 2010] JetBrainsc (2010). Kotlin.

[Lee, 2010] Lee, J. T. . A. (2010). Firebase.

[Lourenço et al., 2014] Lourenço, A., da Silva, H. P., Carreiras, C., Alves, A. P., e Fred, A. L. N. (2014). A web-based platform for biosignal visualization and annotation. *Multimedia Tools and Applications*, 70:433–460.

[Posner, 2005] Posner, Jonathan, e. a. (2005). The circumplex model of affect: an integrative approach to affective neuroscience, cognitive development, and psychopathology. In *Dev Psychopathol.*, volume 17.

[SIG, 2020] SIG, B. (2020). Intro to bluetooth low energy.

[Simón et al., 2017] Simón, M., Sarasua, E., Gamecho, B., Larraza-Mendiluze, E., e Garay-Vitoria, N. (2017). *DAFIESKU*, volume 2017, cap.: https://doi.org/10.1155/2017/7261958. The MIT Press.

[the fused location provider, 2020] the fused location provider, G. P. S. . (2020). Location.

[Trattner et al., 2014] Trattner, C., Helic, D., e Strohmaier, M. (2014). Tag clouds. *Encyclopedia of Social Network Analysis and Mining*, 1.