ISEL – Instituto Superior de Engenharia de Lisboa

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

LEIM

Licenciatura em Engenharia Informática e Multimédia

Projeto LEIM

# Monitorização do Ritmo Cardíaco para Personalização de Playlists de Músicas

moodler

Rui Santos Nº 39286

Fábio Rolo Nº41513

*Orientador(es)*

*Professor Doutor André Lourenço*

*Professor Doutor Rui Jesus*

*Julho, 2018*

# Resumo

Com a popularização de plataformas de streaming de músicas, como por exemplo o Spotify/Google Music/Apple Music, o número de músicas disponiveis aumentou exponencialmente. Isso deve-se à quantidade gigantesca de músicas que são lançadas diariamente, bem como a criação de *playlists*[1].

A maior parte das *playlists* existentes são criadas por utilizadores com base no conhecimento musical.Estas *playlists* que podem ser partilhadas.

Assim, torna-se importante arranjar mecanismos que permitam ao utilizador criar as suas playlists de forma dinâmica e automática.Para isso iremos desenvolver um sistema que tenta aferir emoções a partir da informação presente na variadabilidade do ritmo cardiaco para poder recomendar novas músicas.

Para treinar o sistema, num primeiro plano é criada uma aplicação para recolher emoções de uma amostra populacional.

Esse treino consiste em o utilizador ouvir uma playlist(estática), com uma variedade de musicas elevada.O utilizador desempenha um papel essencial, sem ele seria impossivel conseguir ter um classificador fidedigno. A arquitetura do projeto assenta sobre um modelo Cliente-Servidor. Sendo que o Cliente é a aplicação android.O Servidor foi desenvolvido em Python. Com base nestes conceitos e tendo por objetivo a criação de playlists dinâmicas, é apresentado este projeto.

[1] Uma playlist é um conjunto de musicas, cuja duração é variável.

# Abstract

With popularization of music streaming platforms, such as Spotify/Google Music/Apple Music, the number of songs available increased exponentially. This is due to the amount of musics that are released daily, as well as creating *playlists*[2].

Most existing *playlists* are created by users based on musical knowledge.These *playlists* can be shared.

With that said, it is important to arrange mechanisms that allow the user to create his playlists dynamically and automatically. For this we will develop a system that will try to gauging emotions from the information present in the heart rate variability to be able to recommend new songs. To train the system, in the first plan its created an aplication to reacall emotion of a population sample.

Create an application that will be used to get emotions of a population sample, to be able to train the classifier. This training consists in the user listening to a playlist (static), with huge variability of musics. User plays an essential role, without him it would be impossible to get a good classification.

The architecture of this project rests on a Client-Server model. Client is the android application. Server was developed on Python.

Based on this concepts and having goals such as the criation of dinamic playlists, this project is presented.

[2] A playlist is a set of songs whose duration is variable.

# Agradecimentos

Aos orientadores, Rui Jesus e André Lourenço, que sempre mostraram disponibilidade para ajudar, que nos guiaram e contribuiram com o seu conhecimento para conseguirmos ter um trabalho bem feito e fidedigno.

A Patricia Padeiro, Vanessa Nunes, Filipe Costa e João Pinto, que sempre nos acompanharam, agradecemos a paciência, apoio e todo o carinho oferecido por estes. Sem eles seria impossivel realizar o trabalho.

Aos nossos Pais, que sempre nos motivaram a ser melhores e a fazer mais.

Ao meu Tio Fernando Cruz, que sem ele nunca tinha chegado aqui.

A todos aqueles que ajudaram a treinar o classificador.

*"Deus quer, o homem sonha, a obra nasce"*
*Fernando Pessoa*

# Índice

# Lista de Tabelas

# Lista de Figuras

# Capítulo 1

# Introdução

A emoção é normalmente definida como uma alteração intensa e passageira do ânimo, podendo esta ser agradável ou penosa, que surge de uma certa comoção somática. As emoções são reacções psicofisiológicas, que representam modos eficazes de adaptação face às mudanças ambientais, contextuais e/ou situacionais. Em termos psicológicos, as emoções alteram a atenção e levam o nivel de determinados comportamentos na hierarquia de respostas do individuo[1].

Passando ao cérebro e falando acerca do sistema nervoso em geral, há que salientar o papel do sistema nervoso autónomo, dividido em sistema nervoso simpático e parassimpático. Quando um acontecimento desencadeia uma resposta emocional, o organismo prepara-se para a ação. Para nos mobilizar para a "luta"ou para a "fuga"decorrentes da raiva ou cólera e do medo, o hipotálamo ativa o sistema nervoso simpático e lança na corrente sanguínea hormonas como a adrenalina mediante o estimulo.

O sistema nervoso simpático prepara o corpo para responder a situações de tensão, de emergência. É o "sistema de emergência"do nosso organismo, é ativado para enfrenter problemas e ameaças.

O sistema nervoso parassimpático é a divisão do sistema nervoso autónomo responsável pela manutenção e conservação das reservas energéticas do organismo.

Em suma, o sistema nervoso simpático e parassimpático trabalham frequentemente em conjunto, coordenam a sua atividade de modo a assegurar a manutenção do equilíbrio do organismo.

As emoções resultam de estados fisiológicos desencadeados por estimulos ambientais[2].

Posto isto, podemos afirmar que o sistema parassimpático tem um papel fundamental no controlo do corpo porque liga o sistema nervoso central ao resto do corpo.

O coração possui um sistema de controlo que é "auto- regulador"e tal sistema é intrinseco ao coração e pode funcionar sem quaisquer influências nervosas, mas a eficácia da ação cardiaca pode ser melhorada através de impulsos que têm origem no Sistema Nervoso Central.

Com isto provamos que o coração pode ter o seu ritmo cardiaco alterado pelos outros sistemas do corpo e não somente pelas suas funções intrinsecas. Em suma, é possivel aferir o estado emocional do utilizador recorrendo ao ritmo cardiaco proveniente do coração, mas apenas se as emoções forem relativamente fortes aquando a aplicação de um estimulo exterior [3]. Os estimulos utilizados foram visuais.

O projeto descrito neste documento consiste em desenvolver um serviço para uma das plataformas de *music streaming* existentes, serviço esse que será simulado numa aplicação desenvolvida em Android. A plataforma de *music streaming* escolhida foi o Spotify, foi a eleita pois oferecia mais mecanismos que as outras para conseguir desenvolver a aplicação/serviço.

O objetivo deste projeto é tornar mais fácil aos utilizadores o processo de criação de uma *playlist*, de modo a conseguir que esta seja criada de forma dinâmica e automática. E deste modo, incentivar os utilizadores a utilizares estas aplicações de streaming.

A aplicação num primeiro plano, irá servir para obter emoções de uma amostra populacional, de modo a conseguir treinar o classificador. Esse processo de captura para o treino consiste em colocar o utilizador a ouvir uma playlist(estática), com uma variedade de musicas elevada.O utilizador desempenha um papel essencial, sem a sua participação seria impossivel conseguir ter um classificador fidedigno.

Assim,a participação dos utilizadores é essencial para o sucesso desta aplicação, uma vez que são estes que fornecem dados para treinar o classificador, e deste modo obter a melhor classificação possivel.

## 1.1 HR e HRV

O Heart rate (ritmo cardiaco) é a velocidade do batimento cardiaco, esta é medida em batimentos por minuto (bpm). O HRV(Heart rate variability) deriva do HR (ver figura 1.1), sendo que este é a diferença de tempo entre batimentos cardiacos consecutivos designados por intervalo de RR(ondas provenientes de um ECG), medidas em milisegundos.O HR e o HRV sao inversamente proporcionais um do outro.

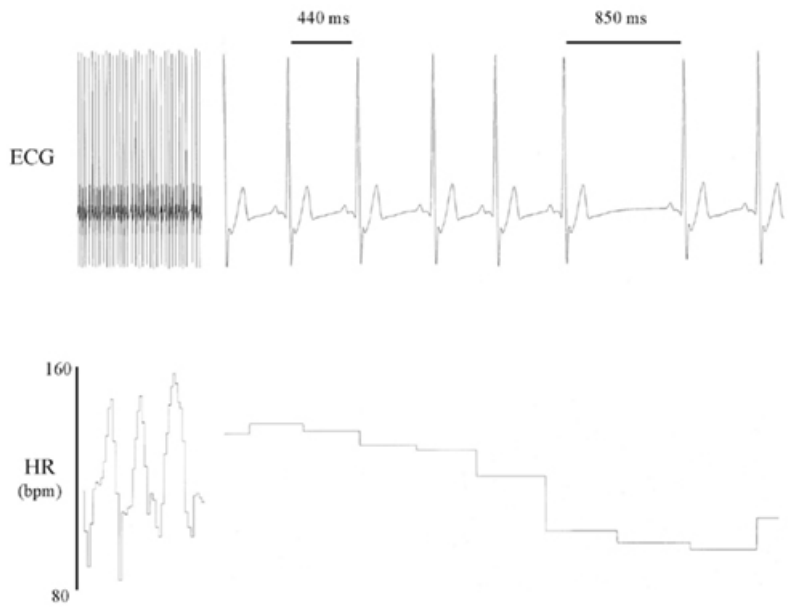


Figura 1.1: Heart rate variability.

## 1.2 Emoções e HRV

As emoções que as pessoas sentem aquando estimulos do meio, são associadas a vários graus de excitação psicológica[4]. O culpado pela excitação psicologica sentida pelo uillizador é do sistema nervoso central, como foi visto anteriormente.

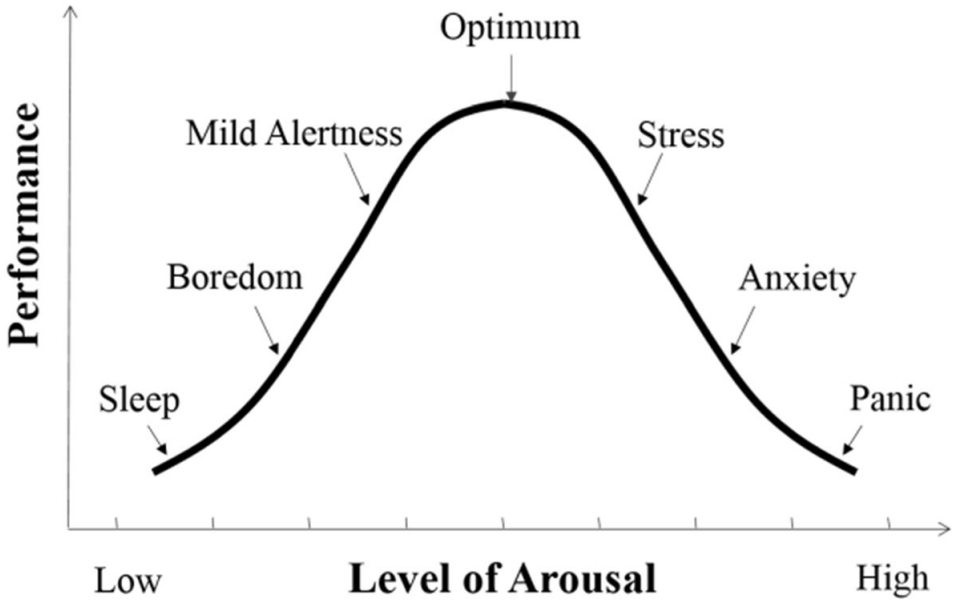


Figura 1.2: Emoções.

# Capítulo 2

# Trabalho Relacionado

Com a constante evolução dos dispositivos móveis e dos acessórios, há cada vez mais aplicações a utilizar sensores. O principal objetivo deste tipo de aplicações é, essencialmente, cuidar da saúde dos utilizadores nos mais variados campos, quer seja na prática de desporto, quer seja na alimentação. O nosso projeto em certa medida, procura ajudar a melhorar o bem estar dos utilizadores. O carácter informativo, é semelhante ao da aplicação **Instant Heart Rate Android App**, aplicação esta que é destinada à monitorização do ritmo cardiaco (ver figura 2.1). O utilizador coloca a ponta do seu dedo indicador na câmara do telemóvel durante alguns segundos, e mostra um grafico com o batimento cardiaco.


Figura 2.1: Ícone da aplicação *Instant Heart Rate*.

Outra das aplicações, com um âmbito semelhante à nossa, é a aplicação **Emotion Detector** ou reconhecimento de emoção (ver figuras 2.2 e 2.3). Esta aplicação pede como entrada um estimulo visual na forma de uma foto de uma face, e retorna a confiança através de um conjunto de emoções.

Figura 2.2: Ícone da aplicação *Emotion Detector*.

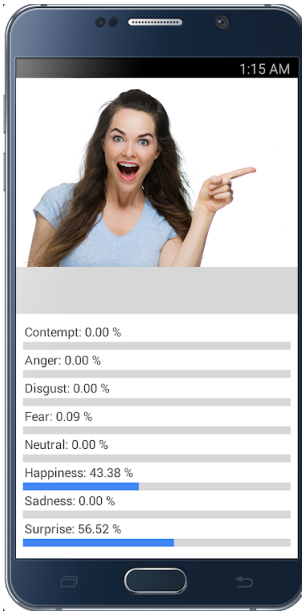


Figura 2.3: *Emotion Detector app*.

Em relação aos sistemas de recomendação, existem multiplas aplicações que ajudam a aumentar o campo de visão do utilizador num determinado tema. Um caso desse tipo de aplicações, é o **Zomato**. Esta aplicação, serve para descobrir os melhores restaurantes e bares mais próximos de uma determinada localização, esta permite tambem escolher um restaurante com base no feedback oferecido pelos amigos/utilizadores que já visitaram um determinado sitio, e deste modo facilitar a escolha de onde comer. A aplicação desenvolvida neste projeto funciona com os mesmo princípios mas para recomendação de músicas.


Figura 2.4: Ícone da aplicação Zomato.

Um dos serviços de music streaming mais utilizadas neste momento, é o Spotify (ver figura 2.5). Este tem não só um grande número de músicas, como também mecanismos de recomendação e personalização de conteúdos. Existem outros serviços de music streaming semelhantes a este, como é o caso do Google Music (ver figura 2.7) e Apple Music (ver figura 2.6).

Figura 2.5: Spotify Icon.

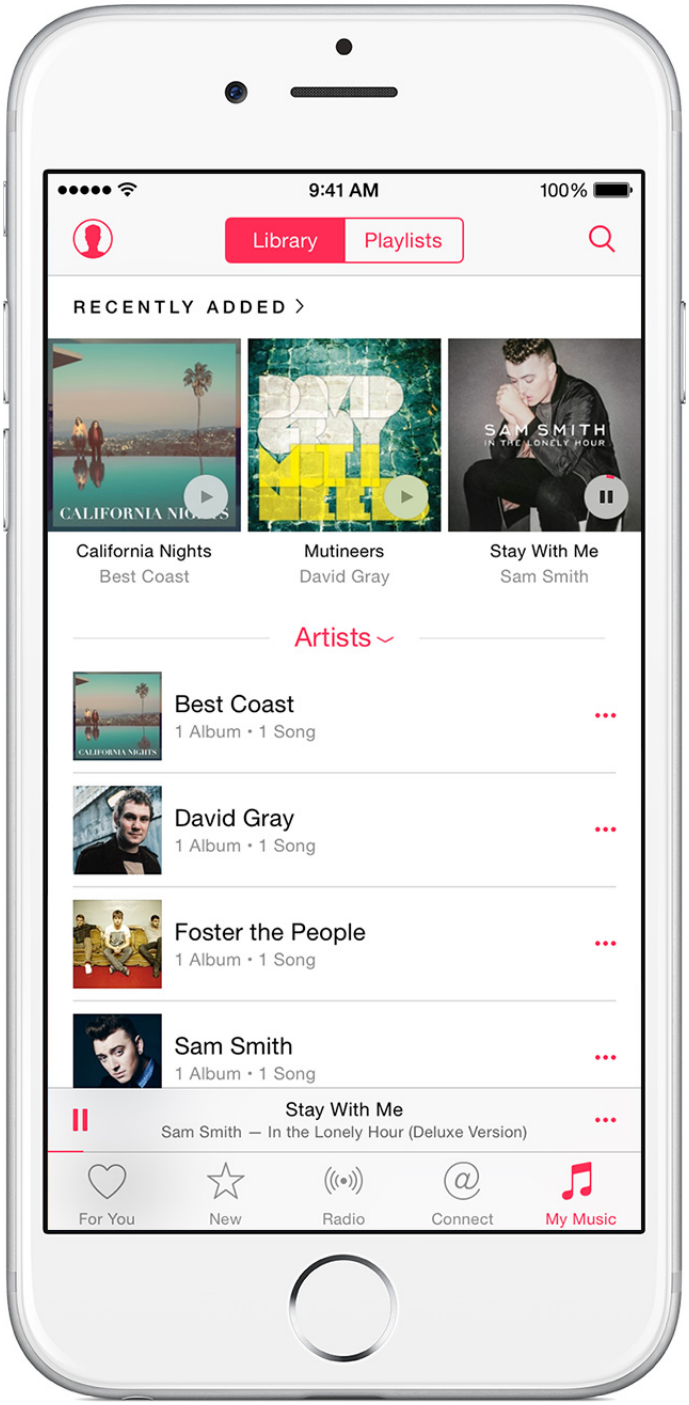


Figura 2.6: Apple Music screen.

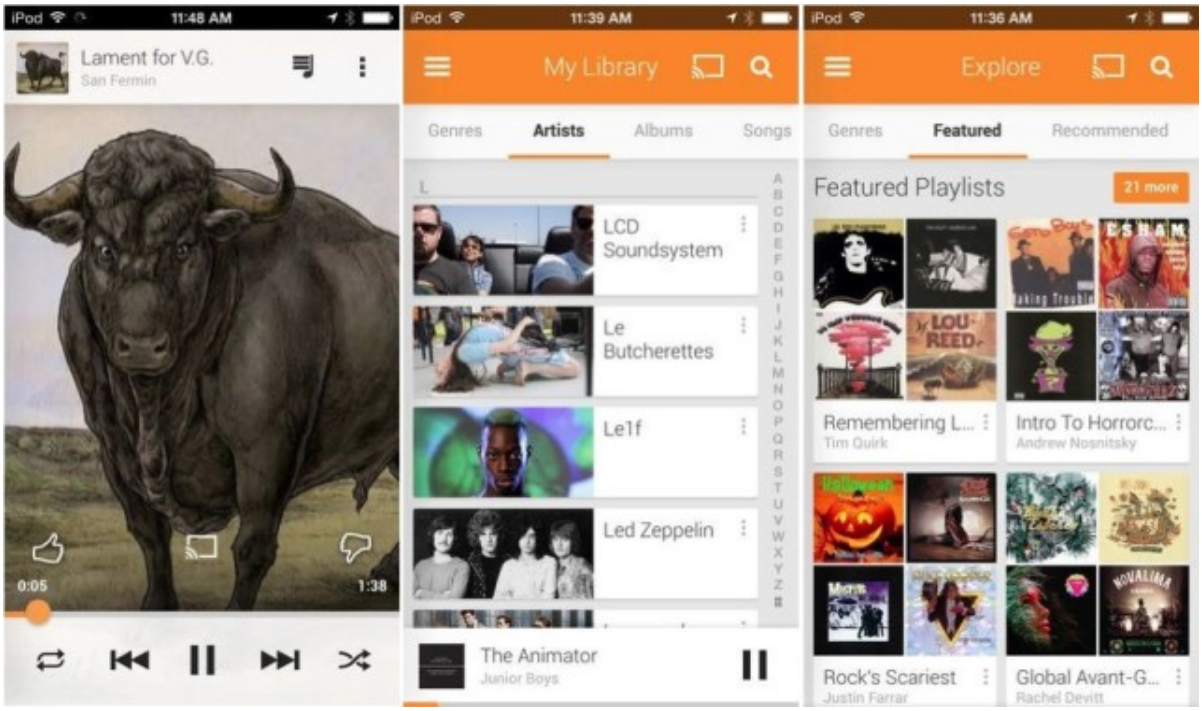


Figura 2.7: Google Music screen.

Estas plataformas, têm todas mecanismos de recomendação estáticos. O nosso objetico é implementar uma versão dinâmica que melhore os mecanismos de recomendação existentes. Existem outras plataformas que usam recomendação estática, como é o caso do Youtube. Este recomenda canais aos utilizadores para realizar subscrição, com base no que anda a circular nas tendências.

Figura 2.8: Youtube Icon.

## Sensores Biométricos

Os *smartwatches* oferecem múltiplos sensores, desde temperatura, localização a sensores de monitorização cardíaca. Estes últimos utilizam um sensor ótico PPG[1] que permite apenas medir os intervalos RR. Existem inúmeros dispositivos que oferecem estes sensores.

Apesar do vasto mercado de sensores capazes de fazerem as funções desejadas, nós utilizamos um sensor fornecido pelo *CardioID*, o sensor Movesense.


Figura 2.9: Sensor Movesense.

[1]Photoplethysmography (PPG) é uma técnica óptica simples usada para detectar mudanças volumétricos no sangue na circulação periférica. É um baixo custo e um método não invasor que faça medidas na superfície da pele.

O projeto desenvolvido, apesar de apresentar diversas semelhanças com as aplicações referidas, tendo sido desenvolvido com base em alguns aspetos nelas contidos, distingue-se essencialmente no facto de ter como principal foco facilitar a criação de playlists aos utilizadores.

# Capítulo 3

# Modelo Proposto

Neste capítulo é apresentado o modelo proposto para atender aos objetivos do projeto.

## 3.1 Requisitos

Nesta secção são apresentados todos os modelos da aplicação utilizada para o treino da aplicação utilizada para classificação. Bem como todas as especificações da aplicação/serviço.

### 3.1.1 Caracterização Geral

Nesta secção é descrito uma síntese de obejetivos, os clientes e as metas a alcançar.

#### Síntese de objetivos

Neste projeto é desenvolvido um serviço para complementar as aplicações de *music streaming*. O objetivo é com base nas emoções sentidas pelo utilizador, seja possivel completar a plalylist do mesmo. Como foi dito anteriormente as emoções são obtidas através do *Heart Rate.*

## Audiência

O público alvo desta aplicação está destinado para jovens e adultos sem género fixo. É ideal para utilizadores que gostam de ouvir musica e de criar playlists, bem como para aqueles que gostam de expandir o seu conhecimento musical.

## Metas a alcançar

Pretende-se que a aplicação contribua para:

1. Criação de playists de forma dinâmica;
2. Aumentar a visibilidade musical do utilizador;
3. Aferir o estado emocional do utilizador;
4. Aplicação de algoritmos de aprendizagem automática a sinais fisiológicos.

## Plataforma de Streaming

Tendo o trabalho relação direta com a música, iria ser necessária a escolha de alguma forma de fornecimento de música. Hoje em dia existem das mais variadas plataformas de *streaming* de música online, mas nem todas disponibilizam acesso à API de desenvolvimento. Poderiamos optar pela plataforma TIDAL, Google Music, Apple Music ou Spotify, no entanto nem todas são vantajosas de utilizar.

Posto isto, foi escolhido o Spotify como plataforma de *streaming*.

O Spotify disponibiliza a sua Web API, ao qual podemos aceder a muitas funcionalidades, sendo que algumas apenas com a vantagem de Premium.
Além de disponibilizar acesso à sua Web API, fornece tambem um Spotify SDK pronto a ser integrado em qualquer aplicação Android, este SDK permite que a comunicação à API seja facilitada, sobretudo no aspecto da autenticação.

### 3.1.2 Casos de Utilização

Nesta secção são apresentados todos os casos de utilização referentes a aplicação descrita anteriormente, bem como os atores.

## Enquadramento de Fases

Como dito anteriormente e estando presente nas nossas metas, pretende-se a recomendação de músicas automaticamente através da avaliação das caracteristicas extraídas da variação cardíaca de um utilizador (HRV) através de um classificador de aprendizagem automática.

#### Fase de treino

Esta fase surge da necessidade de recolher dados para a construção de um banco de dados para que este nosso classificador se consiga basear e assim providenciar resultados mais fidedignos.

#### Fase final

A fase final consiste na continuação da anterior, em que o classificador já se encontra treinado, e da mesma maneira que o utilizador fornecia dados para o banco, ouve músicas presentes numa playlist e o sistema recomenda novas músicas para ouvir, do mesmo artísta, album ou estilo musical, inserindo-as na playlist à qual escuta de momento.

## Atores

O Atores intervenientes no sistema são os seguintes:

- **Utilizador** - representa o utilizador da aplicação;
- **Spotify API** - representa a API disponibilizada pelo Spotify, responsável pelo fornecimento de musicas;
- **Sensor (Movesense)** - representa a monitorização do ritmo cardiaco instantâneo do utilizador.

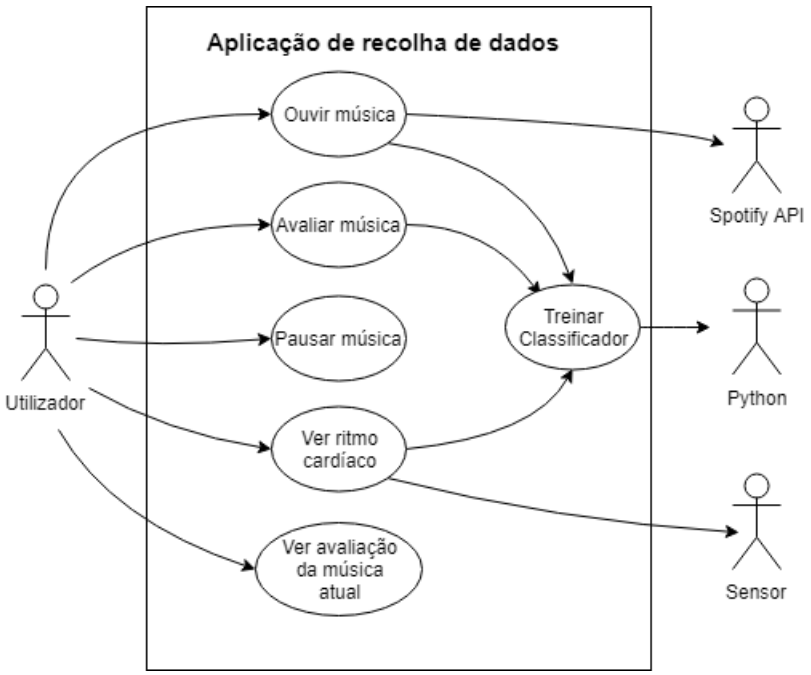


Figura 3.1: Funcionalidades da aplicação intermédia (fase de treino).

Em seguida uma breve descrição de cada caso de utilização desta fase apresentado na figura 3.1.
Alguns casos não envolvem ação direta do utilizador para acontecerem e portanto são considerados como invisiveis. Assim temos os seguintes casos de utilização:

- **Ouvir música** - O utilizador consegue ouvir uma música contida na playlist que disponibilizamos para a recolha de dados nesta fase.
- **Avaliar música** - O utilizador avalia a música à medida que a escuta, pressionando botões de gosto e não gosto.
- **Pausar música** - O utilizador pode pausar a música que atualmente escuta.
- **Ver ritmo cardiaco** - É disponibilizada a informação do intervalo entre as ondas R.
- **Ver avalização da música atual** - À medida que o utilizador vai ouvindo a playlist no caso de utilização anteriomente falado, e assim

avaliando-a, consegue observar se a avaliação média do mesmo é negativa ou positiva.

- **Treina classificador** - Quando um utilizador acaba de escutar a playlist até ao final, é-lhe mostrado um novo ecrã agradecendo a sua participação e efetuado um pedido ao servidor para que o classificador se treine com os novos dados inseridos. Este caso de uso, entra nos especificos em que o utilizador não indica diretamente para treinar o classificador mas tal acontece *invisivelmente*.

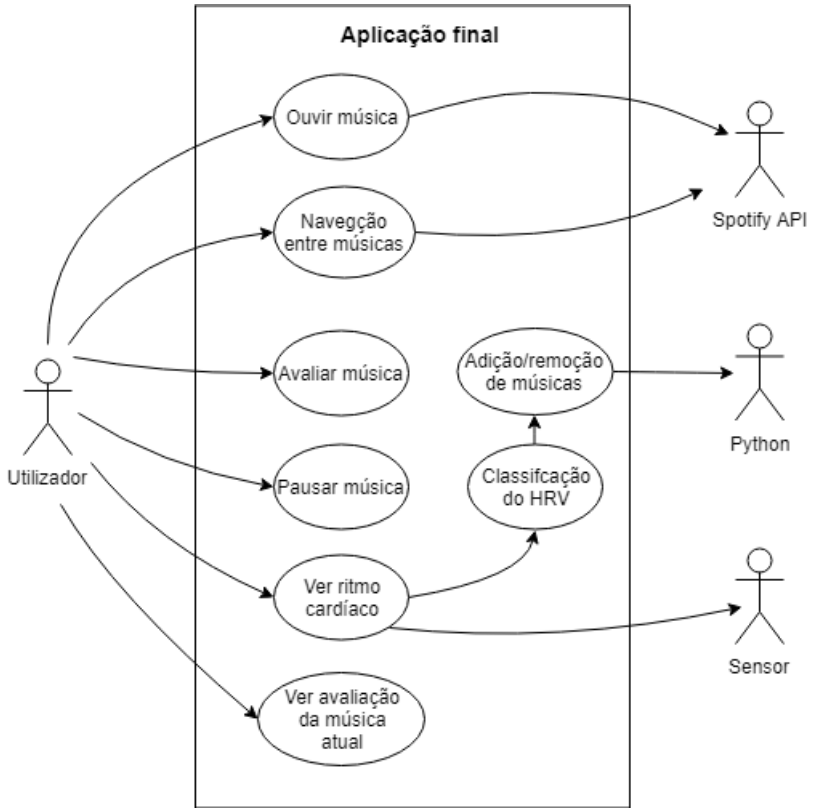


Figura 3.2: Funcionalidades da aplicação final (fase final).

Na figura 3.2 apresenta-se os casos de utilização da fase final.
É de notar que alguns dos casos são semelhantes aos da primeira fase e portanto somente se irá referir os novos casos.Assim temos:

- **Navegação entre músicas** - O utilizador pode agora navegar entre as diversas músicas da *playlist*.

- **Classificação do HRV** - Quando o utilizador acaba de escutar a música é enviado para o servidor os seus dados de HRV, são extraídas as caracteristicas e atribuida uma classificação de gosto ou não gosto.
- **Adição/remoção de músicas** - Após a classificação do valores do HRV, são adicionadas novas músicas à playlist ou removida a música atual.

  Este últimos dois casos de uso, entram no espectro dos casos invisiveis, reforçando o facto de isto ser um mecanismo automático de recomendações de música.

## 3.2 Abordagem

Nesta secção são apresentadas as decisões tomadas no desenvolvimento do projeto, como a arquitetura pela qual se optou, as tecnologias utilizadas e o motivo das suas escolhas e a divisão das tarefas do projeto ao longo do tempo de duração do mesmo.

### 3.2.1 Arquitetura e Tecnologias

Nesta secção são apresentadas a arquitetura e as tecnologias utilizadas durante o desenvolvimento desta aplicação/serviço.

### Arquitetura da aplicação

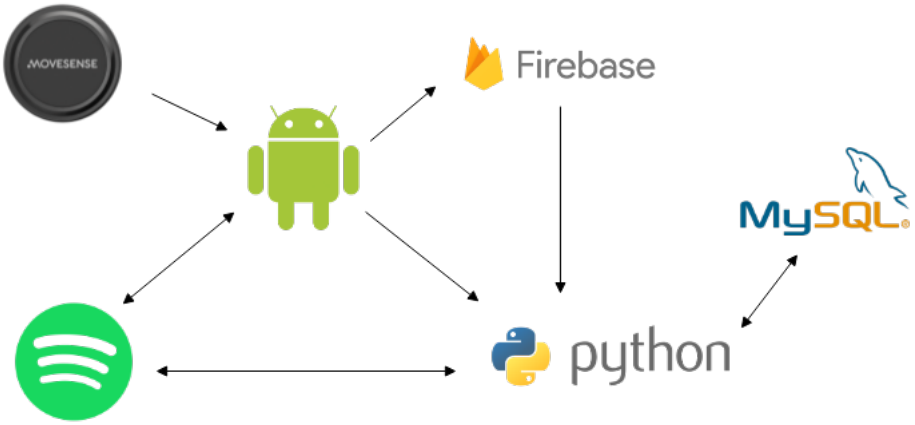


Figura 3.3: Arquitetura da aplicação.

### 3.2.2 Diagrama de Blocos

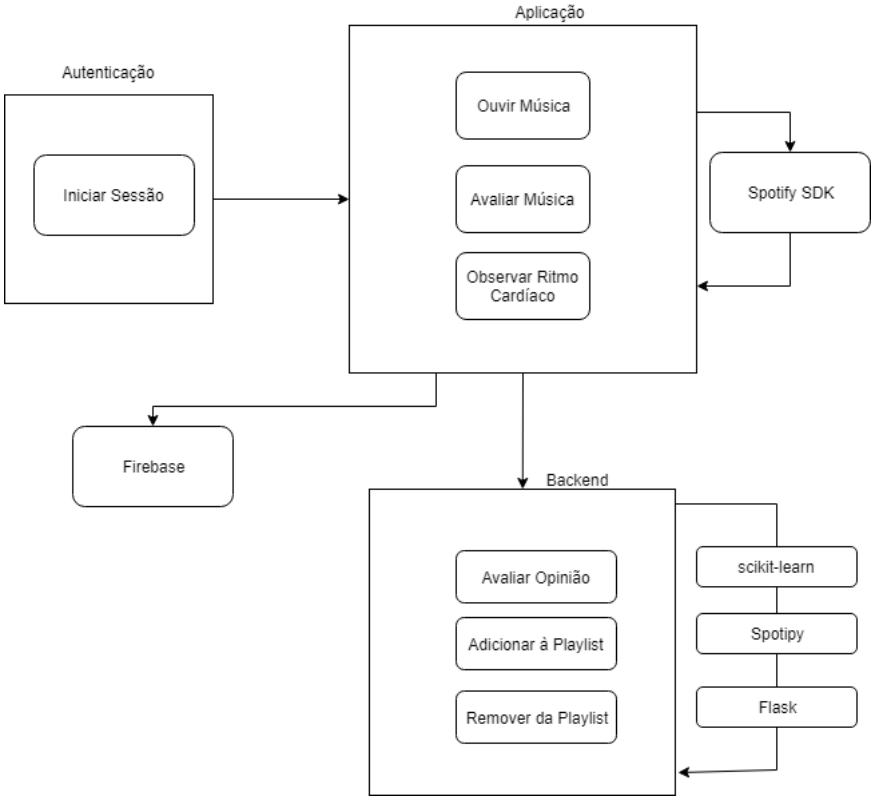


Figura 3.4: Diagrama de blocos da aplicação.

## Tecnologias

Aqui são apresentadas todas as tecnologias utilizadas na conceção e desenvolvimento desta aplicação. As tecnologias utilizadas são as seguintes:

- **Android** - Utilizada para desenolver a aplicação intermédia, bem como a final;
- **Python** - Utilizada para fazer o servidor, bem como o classificador;
- **Firebase** - Utilizada para guardar a informação da aplicação intermédia.
- **MySql** - Utilizada para guardar a informação, na aplicação final;

## Arquitetura do Sistema

Este tipo de aplicações necessita de um elevado conjunto de informações, acerca dos utilizadores. Há, assim, uma necessidade de incluir uma base de dados que não pode ser local ao dispositivo Android (impossibilita a utilização de MySqlLite), uma vez que os batimentos cardiacos e a avaliação de cada utilizador tem que ser guardadas.
Uma vez que não pode ser local a base de dados é implicita a necessidade de arranjar uma solução para a mesma. Tendo em conta que os dados que estamos a guardar não irão ter qualquer tipo de relação com outros dados, podemos optar por uma base de dados não relacional.
Posto estas condições, a arquitetura Cliente - Servidor passou a ser mandatório. O nosso Cliente é interpretado como o Android e o nosso servidor como o Python / MySQL.

## Base de dados não relacional

A base de dados não relacional é usada de maneira de facilitar o armazenamento na aplicação intermédia (Fase de treino). Foi utilizada a Firebase de modo a guardar os dados, visto que esta plataforma é gratuita e contem inúmeros mecanismos úteis no desenvolvimento desta. A comunicação é feita através de mensagens HTTP contendo ficheiros JSON no *body* da mesma, onde é enviado as informações necessárias como os valores da diferença entre os picos dos batimentos cardíacos (ondas R) e outras que sejam relevantes.

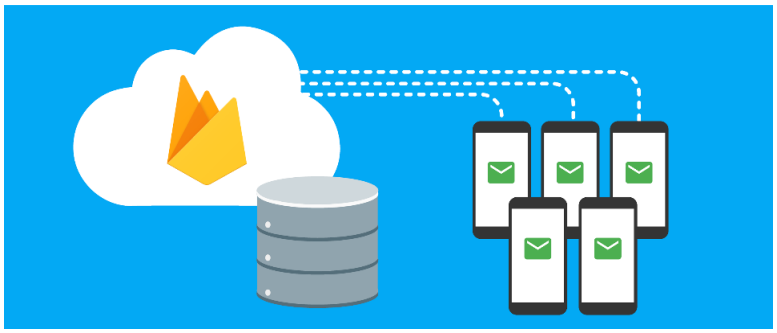

Figura 3.5: Arquitetura Firebase.

O Firebase providência uma base de dados *Real time* e um serviço de backend. Este serviço permite ainda que os dados presentes no mesmo estejam constantemente sincronizados por todos os clientes que desejem utilizá-la, irá provar-se bastante útil quando avançar-mos para a segunda parte do trabalho em que iremos utilizar estes dados.
Este serviço tem imensas bibliotecas que permitem a sua integração facilmente em qualquer tipo de aplicações, desde Node.js , Python, Javascript, Android e Java.

## Base de dados relacional

Tendo em conta que o que a aplicação a desenvolver parte de uma ideia da criação de um serviço que poderia ser incorporado em qualquer plataforma de streaming musical, de alguma maneira teriamos de simular as relações e possiveis dados que existem nas bases de dados dessas mesmas aplicações, pois apesar do Spotify nos dar acesso a bastantes dados do utilizador, há certas relações que se revelam mais complicadas de obter com os endpoints limitados da sua API, e portanto torna-se mais acessível a criação de um modelo de dados relacional, ao qual nós vamos populando com dados relevantes para a ligação entre o utilizador, a sua playlist e as músicas que para ele foram recomendadas.
A base de dados de eleição é o **MySQL**. A nossa experiência com o MySQL obtida em unidades curriculares do curso levou-nos a decidir utilizar esta base de dados que nos levou a ser alvo de escolha, por facilidade de adaptação da nossa parte e por conhecimento à priori. O facto de ser uma linguagem open-source com bastante suporte, dispõe de diversos mecanismos para futura escalabilidade caso necessário, ser fácil de aprender e de comprender, são factores que também têm grande peso.

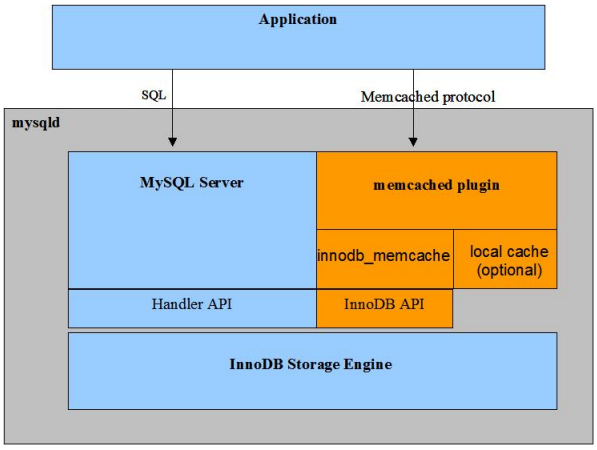


Figura 3.6: Arquitetura do MySQL.

**Servidor** Existem diversas linguagens de servidor que possibilitam a criação de uma estrutura backend fiável e que satisfaça todos os requisitos para este projeto, no entanto, por motivos semelhantes aos explicados na escolha do MySQL, Python foi a linguagem preferida, por ser bastante

leccionada ao longo do curso e pelo facto de a parte core da análise da variação dos batimentos cardíacos do utilizador ser providênciada pela empresa **CardioID** e estas mesmas funções terem sido desenvolvidas em Python.
Outra das razões que nos levou a escolher o Python, passa pela vasta quantidade de bibliotecas que o mesmo dispõe, sobretudo no tema *Machine Learning*, a biblioteca *Sckitlearn* disponibiliza de variadissímos classificadores, assim sendo, podemos juntar o melhor de todos os mundois refiridos e apenas usar uma linguagem.

## Sensor BLE

O sensor BLE é um elemento fulcral no nosso projeto, é este que está encarregado de providenciar os valores das medições da variação do ritmo cardíaco à nossa aplicação Android, sem este, não seria possível a análise de caracteristicas dos batimentos do utilizador.
Para este efeito, o nosso sensor é o *Movesense*, é um open source development kit que permite a medição de qualquer objecto ou pessoa em movimento, usado para diversos fins, como por exemplo, qual o truque efetuado com um *skateboard*.
Este sensor foi-nos emprestado por parte da **CardioID**.
**BLE** é uma tecnologia relacionada com dispositivos bluetooth, que permite um consumo de bateria muito mais reduzido e fornece a mesma distância para comunicação em relação ao bluetooth clássico.

# Capítulo 4

# Implementação do Modelo

Neste capítulo é descrita a implementação do modelo apresentado no capítulo anterior. Iremos começar por apresentar as dependências tecnológicas, seguidamente iremos explicar todo o processo de implementação.

## 4.1 Dependências tecnológicas

No desenvolvimento deste projeto foram usadas várias tecnologias. A aplicação/ serviço divide-se em duas partes, cliente e servidor, sendo que cada uma delas utiliza tecnologias distintas. Na aplicação cliente, optou-se pela utilização da ferramenta Android Studio, esta já tinha sido utilizada na cadeira de DAM. Esta ferramenta, utiliza a linguagem de programação Java para o desenvolvimento de aplicações.

Uma das principais caracteristicas desta é a utilização de um player de música, para o utilizador ouvir as suas playlists. Uma opção, poderia ser a de desenvolver um *player* de raiz. Contudo, recorremos à utilização do Spotify SDK. Este disponibiliza um conjunto de funcionalidas que são descritas abaixo, e entre estas um player.

Quanto ao servidor, este é desenvolvido em Python, como foi descrito anteriormente.

## 4.2 Base de dados

O projeto possui uma base de dados cujo futuro irá ser constituída por dados relacionados com cada utilizador que usufrua da aplicação construida. De forma a projetar a base de dados pretendida irá ser necessário após a análise de requisitos:

- elaborar o modelo conceptual da base de dados através do modelo entidade-associação;
- construir o modelo lógico a partir da adaptação do modelo conceptual, descrito a partir do modelo de dados suportado, que será um modelo relacional;
- construir o modelo físico, que consiste na organização física dos dados, dependente do SGBD (sistema de gestão de base de dados) usado que será o MySQL.

Para construir o modelo entidade-associação é necessário o entendimento de alguns conceitos:

- **Entidade** - representação abstrata de um qualquer objeto ou conceito que existe e pode ser distinguido de qualquer outro forma inequívoca. Cada entidade tem um conjunto de características que lhe são relevantes, estas características designam-se por atributos.
- **Atributo** - propriedade que caracteriza as entidades. Podem apenas descrver ou caracterizar uma entidade ou associação, ou, podem ser chave, permitindo a identifcação unívoca da entidade a que pertencem. É obrigatório que cada entidade possua uma chave primária.
- **Associação** - representa a interação entre entidade, ou seja, uma associação entre duas entidades indica que elas estão interligadas por algo de interesse para o contexto. Uma associação pode ter duas ou mais entidades que participam nela.

O modelo relacional irá fazer corresponder cada entidade a um esquema de relação com os mesmos atributos e chave. Caso necessário, uma associação poderá ser representada por um esquema de relação com o próprio

tendo como chave a junção dos esquemas associados.
Uma chave (candidata) é um conjunto de um ou mais atributos, que permite indentificar univocamente os registos de um esquema de relação, ou seja, cujo valor deve ser único quando comparando com os restantes registos. A chave primária consiste numa chave candidata elegida para identificar cada registo.

Na escolha de chaves primárias existiam duas opções: chaves naturais e chaves auto-incrementais.
Uma chave natural é uma coluna ou várias colunas de uma tabela que permitem identificar univocamente um registo de uma tabela. Esta coluna ou colunas contêm informação real que está relacionada com as restantes colunas da tabela.
Uma chave auto-incremental consiste na introdução dew uma colunas numa tabela, que contém valores numéricos únicos que podem ser usados para identificar univocamente um registo de uma tabela.
Qualquer uma das opções apresenta vantagens e desvantagens. A principal vantagem de uma chave primária natural, passa por possuir algum significado e assim são necessárias *querys* menos complicadas quando na busca de informação especifica dos registos. No entanto como poderemos ver no modelo de entidade-associação na figura 4.1, algumas das chaves naturais que poderiam surgir seriam um conjuntos de caracteres de texto o que tornaria a junção de tabelas mais complicada por a dimensão dos valores ser maior, tal não acontece com o uso de chaves auto-incrementais.
No entanto, se a lógica da base de dados for mudada e for necessário alterar o valor da chave primária ou até mesmo o tipo da mesma, iria provocar um efeito cascata, visto que será necessário alterar em todas as chaves estrangeiras noutras tabelas relacionadas, com uma chave auto-incremental não iria ocorrer este problema.
Após avaliação das vantagens e desvantagens, optou-se por escolher chaves auto-incrementais.

### 4.2.1 Modelo EA

Após a análise dos requisitos elaborados no capitulo 3, irá ser feita nesta subsecção, a descrição das necessidades impostas pelos mesmo sob a forma de um diagrama de entidade-associação.
Visto serem necessárias poucas entidades para o efeito, iremos falar brevemente sobre o significado de cada uma.

#### User

Esta entidade representa o user no spotify. Sempre que efetuamos um login na aplicação através do Spotify SDK, podemos ter acesso ao ID deste user, que por sua vez pode ser descrito de duas formas; quando efetuado login através do facebook, é atribuido um numero inteiro normalmente constituido por um número inteiro de 11 algarismos, quando efetuado login com uma conta Spotify, é somente o username que este mesmo escolheu no ato da criação da sua conta.
É importante guardar este valor, pois é necessário quando mais tarde realizarmos operações sobre as playlists futuras do utilizador em questão.
Guardamos a data como uma mera referência de quando o user utilizou esta aplicação pela primeira vez.

#### Playlist

A entidade Playlist representa uma playlist do utilizador, esta tem os atributos playlist-id que pode ser um pouco confuso, no entanto é diferente do id da sua chave primária. Este id é atribuido pelo Spotify aquando a criação de uma playlist, é composto por números e letras e é guardado para mais tarde o servidor conseguir ter esta informação para assim fazer pesquisas/alterações sob a mesma. Tem um atributo que corresponde ao atributo id da tabela User, que nada mais é a atribuição da relação que o utilizador tem uma playlist.

#### Track

A entidade Track, nada mais é do que uma faixa de música que irá integrar uma playlist pertencente ao user que usufrui da aplicação no momento. Assume um papel muito importante no que toca à manutenção da playlist, pois

se a classificação das emoções do utilizador enquanto estiver ouvir a playlist for negativa, é retirada essa mesma música, e por isso mesmo o track-id é guardado nesta entidade. Este id é constituído por um conjunto de caracteres númericos e de texto.
Pertence-lhe também um atributo playlist-id que nada mais identifica a relação da música com a playlist.

### Reaction

Outra entidade muito importante neste sistema. Irá representar todas as reações que os diferentes utilizadores tiveram **após** o treino do classificador. Desta maneira conseguimos ter algum histórico sob o desempenho do mesmo e assim fazer um levantamento se este as classifica fidedignamente ou não. Tem atributos como hrv e evaluation que indicam os respetivamente a variação cardiaca do utilizador e em que classe foi classificado pelo nosso classificador. O atributo *user_evaluation* é uma referência à avaliação do utilizador sobre a música que foi classificada. Os restantes atributos representam as relações que tem com as outras tabelas, sendo sempre referências às chaves auto-incrementais das mesmas.

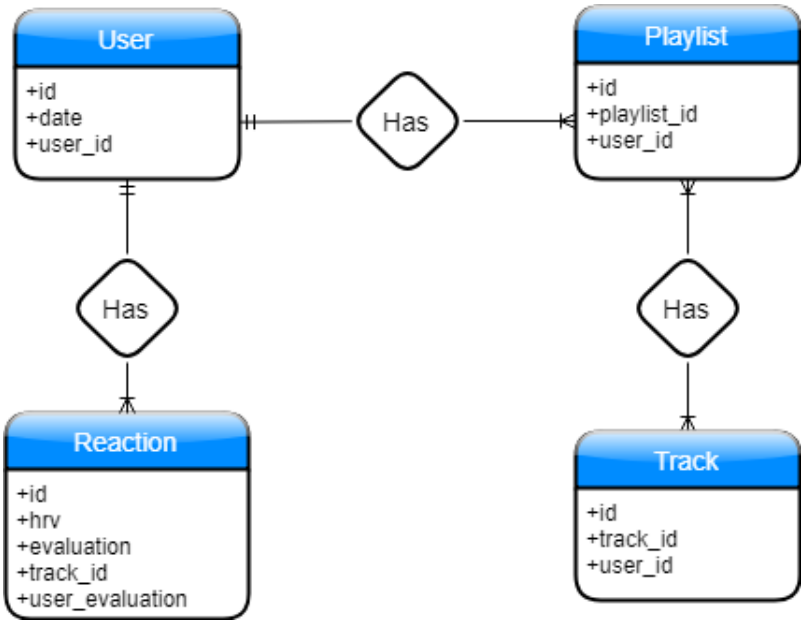


Figura 4.1: Modelo EA.

### 4.2.2 Modelo Relacional

| Entidades | Chaves & Atributos | Chaves Candidatas | Chaves Estrangeiras |
|---|---|---|---|
| User | id, date, user_id | id | |
| Playlist | id, user_id, playlist_id, name | id | user_id |
| Track | id, playlist_id, track_id | id | playlist_id |
| Reaction | id, hrv, evaluation, user_id, track_id | id | user_id, track_id |

Tabela 4.1: Modelo Relacional.

## 4.3 Client

### 4.3.1 Android

Como foi dito anteriormente, a aplicação cliente foi desenvolvida em Android, utilizando o Android Studio. Para isso utilizamos um SDK [1] fornecido pelo Spotify.

### Spotify SDK

O SDK Android torna mais facil adicionar funcionalidades do Spotify a qualquer aplicação, é de notar que para conseguir utilizar este módulo o utilizador tem que ter uma conta premium. O SDK contém duas bibliotecas:

- **Spotify Authentication Library** - Esta biblioteca fornece uma forma de obter o acess token[2] que pode ser utilizado para ouvir musica ou utilizar funçoes da Web Api do Spotify;
- **Spotify Player Library**- Esta bilioteca contem classes para reproduzir áudio, bem como manutenção de stream. Esta cuida de todos os pedidos ao backend do Spotify.

O player do spotify tem múltiplos playback events, que permitem controlar o programa. A seguir descreve-se alguns eventos do *player*.

[1] SDK é tipicamente um conjunto de ferramentas de desenvolvimento de software que permite a criação de aplicações para um certo software já existente.

[2]O Acess Token é uma credencial que pode ser usada por uma aplicação para aceder á API.[3]Uma API é um conjunto de funções e procedimentos que permitem a criação de aplicações que acessem a recursos ou dados de um sistema operativo, ou outros serviços

- **kSpPlaybackEventAudioFlush** - A aplicação deve libertar os seus buffers. Este evento ocorre, por exemplo, quando se pretende colocar numa posição diferente dentro da música;
- **kSpPlaybackNotifyAudioDeliveryDone** - A biblioteca não vai enviar mais dados. Este evento acontece quando a playlist chega ao fim;
- **kSpPlaybackNotifyBecameActive** - O disposivo torna-se o dispositivo de reprodução;
- **kSpPlaybackNotifyBecameInactive** - O dispositivo deixa de ser o dispositivo de reprodução;
- **kSpPlaybackNotifyContextChanged** - A reprodução mudou para um contexto diferente do Spotify;
- **kSpPlaybackNotifyLostPermission** - O dispositivo perdeu as permissões para utilizar a stream de audio do Spotify;
- **kSpPlaybackNotifyMetadataChanged** - A metadata mudou. Este evento acontece quando começa a tocar outra música;
- **kSpPlaybackNotifyNext** - Passou a música a frente;
- **kSpPlaybackNotifyPause** - Player colocado em pausa;
- **kSpPlaybackNotifyPlay** - Player retomou a reprodução;
- **kSpPlaybackNotifyPrev** - Passou a uma música numa posição anterior á música atual;
- **kSpPlaybackNotifyRepeatOff** - A repetição foi desativada;
- **kSpPlaybackNotifyRepeatOn** - A repetição foi ativada;
- **kSpPlaybackNotifyShuffleOff** - a reprodução aleatória foi desativada;
- **kSpPlaybackNotifyShuffleOff** - a reprodução aleatória foi ativada;
- **kSpPlaybackNotifyTrackChanged** - A música atual ou a sua metadata mudou;

- **kSpPlaybackNotifyTrackDelivered** - A aplicação recebeu toda a informação acerca da música atual.

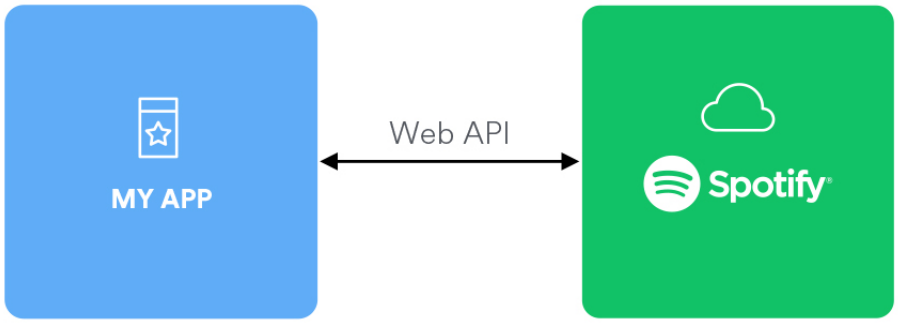


Figura 4.2: Comunicação da Aplicação com o Spotify.

Este SDK não tem funções de manutenção de *playlists*. Posto isto, tivemos que utilizar outros meios para conseguirmos adicionar ou remover de uma determinada *playlist*, assim utilizámos o Spotipy.

### 4.3.2 Autenticação

A autenticação é realizada através da biblioteca **Spotify Authentication Library**,lembrando que para utiliza-la é necessário que a conta seja premium. Esta biblioteca fornece duas formas para a aplicação autenticar o utilizador, transmitindo assim todos os dados necessários para obter a autorização para aceder a todos os campos deste:

- **A utilização do *Spotify Client* (forma recomendada) para fazer a autenticação do utilizador(login realizado através de uma WebView.** Se o Spotify estiver instalado no dispositivo, o SDK conecta-se com o Spotify client e utiliza a sessão atual. Caso contrário, irá ser aberta uma WebView para que o utilizador faça login. Lembrando que a autenticação e a autorização acontecem na WebView sem nunca ter deixado a aplicação.

- **O Login é realizado através de um browser (esta forma só é utilizada se não for possível a primeira).** Este método faz o login através de um browser externo, completa o processo de autenticação e depois volta à aplicação.

Esta biblioteca do Spotify SDK utiliza o OAuth2.

## OAuth2

O OAuth2 é uma estrutura de autorização que permite as aplicações obter acesso limitado a contas de utilizadores num serviço HTTP, tal como o Facebook, Spotify e outros. Este atribui a autenticação do utilizador ao serviço, e autoriza aplicações de terceiros a aceder a conta do utilizador. O OAuth 2 fornece fornece um fluxo de autorização para aplicações web, desktop e para dispositivos móveis (ver figura 4.3).

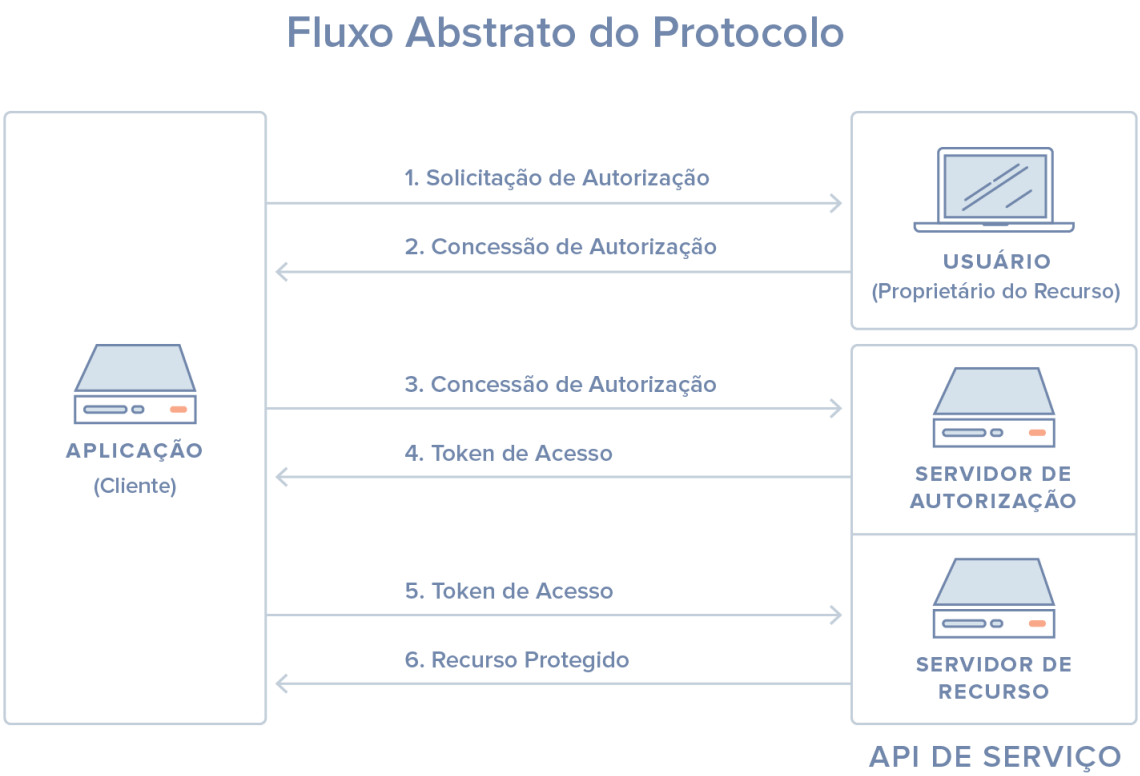


Figura 4.3: Troca de Mensagens do Protocolo OAuth2.

### 4.3.3 Firebase

Como foi dito no capitulo 3, foi utilizada uma base de dados não relacional, foi optada por esta uma vez que fornece inúmeros mecanismos que facilitaram a criação da aplicação de recolha de dados.

Cada entrada na base de dados tem a seguinte informação:

- **artist** - É indicado o nome do artista da música, é utilizado como fator de recomendação, ou seja se o utilizador gostou deste artista vao ser recomendadas mais deste;
- **evaluation** - É indicada a avaliação do utilizador acerca da música. Os valores possiveis são True ou False;
- **hrv** - Contém uma lista com o batimento cardíaco instantâneo registado pelo utilizador durante toda a música;
- **music** - É indicado o nome da música;
- **name** - É indicado o nome do utilizador que está a treinar o classificador;
- **uri** - É indicado o uri[4] de toda a metadata da música.Estes uri's vão ser utilizados na parte do servidor, pois necessitam desses dados para futuras operações.

[4] Um URI é uma cadeia de carateres compacta usada para identificar ou denominar um recurso na Internet.

**Firebase - estrutura de dados**


```
{
    "artist": "Rick Astley",
    "evaluation": "true",
    "hrv": "[90,90,90,90,90,90,90]",
    "music": "Never gonna give you up",
    "name": "fabio",
    "uri": {
            "trackUri": "spotify:track:5A6OHHy73AR5tLxgTc98zz",
            "albumUri": "spotify:album:6ZOXiVL8rmk2ATHJiFJhiD",
            "artistUri": "spotify:artist:137W8MRPWKqSmrBGDBFSop"
    }
}
```


Figura 4.4: Estrutura de dados

## 4.4 Servidor

O servidor segue o estilo de arquitetura REST - Representional State Transfer. Esta arquitetura segue quatro principios principais:

- Usa métodos HTTP explicitamente;
- É stateless;
- Cada recuro é representado por um URI;
- Permite vários tipos de formatos de informação (JSON, XML, HTML).

O servidor irá realizar diversas operações com o auxilio de alguns modulos de Python e de classes especificas criadas por nós. Estas acções podem ser principalmentre interpretadas como, inserção de registos na base de dados, busca desses mesmo dados, modificação de playlists, comunicações à Spotify API.

### 4.4.1 Flask

O Flask é considerada uma *microframework*[5] pois não depende de nenhuma ferramenta ou biblioteca em particular. Não usufruir de nenhuma camada de abstração de base de dados, validação de formulários ou outros componentes que forneçam bibliotecas pré-existentes. No entanto, esta suporta extensões que podem adicionar funcionalidades à aplicação como se fizessem parte do Flask.

O Flask usa as seguintes dependencias:

- **WerkZeug** é uma biblioteca para desenvolvimento de apps **WSGI**[6] que é a especificação universal de como deve ser a interface entre uma app Python e um web server. Possui a implementação básica deste padrão para interceptar requests e lidar com responses.
- **Jinja2** é um template engine escrito em Python que se encarrega simplesmente de renderizar as views para o utilizador com diversos mecanismo de acesso às variáveis com os valores requerentes.

Esta micro framework é também conhecida por ser simples a sua utilização devido à sua facilidade de compreensão, possibilitando ao *developer* modelar a sua aplicação como desejar.


Figura 4.5: Flask logo

[5]É o termo minimalista usado para a descrição de uma pequena *framework* sendo que esta significa uma abstração que une códigos comuns entre vários projetos de software providênciando uma funcionalidade genérica.

[6]Web Server Gateway Interface - É uma simples convenção para pedidos HTTP serem redirecionados para os webservers

### 4.4.2 Arquitetura

O servidor foi desenvolvido segundo a arquitetura MVC (Model View Controller) que é um padrão de desenho de arquitetura de software.
O MVC permite separar a representação da informação da intenção do utilizador com ele e descreve uma forma de estrutura que permite a divisão das funções do sistema por diferentes partes da mesma.
Com a utilização deste padrão pretende-se possuir um servidor escalável e portável. Está dividido em 4 componentes:

- *Routes* - encaminham os pedidos HTTP para as funções do controller;
- *Controllers* - responsáveis por responder aos pedidos HTTP, obtendo ou atualizando informações nos models;
- *Models* - nível mais abaixo da arquitetura que é responsável por gerir e manipular os dados da aplicação;
- *Views* - Não foram utilizadas views neste trabalho, pois todas as nossas interfaces gráficas são realizadas na aplicação android.

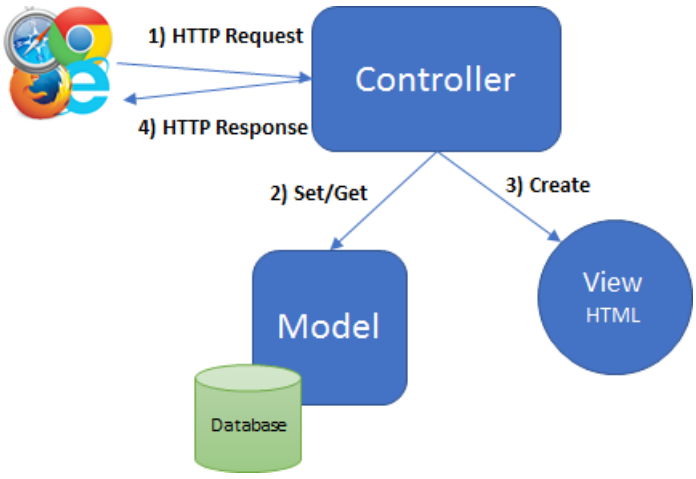


Figura 4.6: Diagrama MVC.

### Routes

As *routes* representam os pedidos de um cliente para um determinado endpoint que é representado por um URI e um método HTTP especifico. Neste projeto iremos somente usar 2 endpoints ativamente no funcionar normal da aplicação. Sendo respetivamente um pedido do tipo GET e outro do tipo POST.

### Controllers

*Controllers* são funções chamadas por routes para tratar os pedidos que recebem, numa abordagem MVC. Os *controllers* geram respostas para os pedidos, através dos dados que obtém a partir dos *models*. A resposta é estruturada de acordo com *views*. No projeto, existe um grupo de main *routes* que são as mais usadas, no entanto existe outros grupos que servem para inserir dados nas tabelas, mas estes são reservados.

### Models

*Models* são objetos que representam linhas individuais de uma tabela de uma base de dados e permitem indicar o nome da tabela a que pertencem, bem como as relações com outros models. No projeto dispomos de 4 models, equivalentes às tabelas existentes: User, Playlist, Track e Reaction.

Além do padrão já mencionado, é também utilizado o padrão **Singleton**, que nada mais é do que usar uma instância de uma classe se esta já havia sido anteriormente instânciada, mantendo assim um ponto global de acesso ao objeto da mesma.

O projeto de momento, como havia dito anteriormente, só usa ativamente 2 das routes que configuramos:

- **train_classifier** - Endpoint que é utilizado sempre que algum utilizador ouve até ao fim a playlist de treino. É efetuado todos os calculos das caracteristicas do HRV e assim treinado o classificador. Esta informações são todas obtidas através de um pedido efetuado à firebase onde estarão alojados todos os batimentos colhidos no modo de treino.

- **hrv_modify_playlist** - route principal que se encarrega de receber os dados do Android relevantes para as funcionalidades devidas, classifica a variação do batimento cardíaco do utilizador, adiciona ou remove músicas consoante a avaliação.


Figura 4.7: Diagrama de atividades - hrv_playlist_modify.

Na figura 4.7 podemos verificar a sequência de operações quando comunicamos com o endpoint responsável por efetuar a classificação do HRV obtido e de adicionar músicas à playlsit do utilizador.

### 4.4.3 Sckitlearn - SVM

O *sckitlearn* é um modulo do Python especialmente dedicado a aprendizagem automática. Fornece os mais variadissímos algoritmos de classificação, desde regressão linear, *clustering* e estes ainda incluem *support vector machine*, *random forest*, *gradient boosting* e *k-means*.

Para este trabalho iremos utilizar o classificador SVM (*Support Vector Machine*). No entanto, de forma a conseguir perceber melhor o porquê de usar este classificador e não outro, é necessário a compreensão de alguns conceitos:

- **Aprendizagem supervisionada** - é o termo usado sempre que o classificador é treinado sobre um conjunto de dados pré-definido. Baseado no treino inicial, o classificador pode tomar decisões precisas quando recebe dados novos.

- **Aprendizagem não supervisionada** - é o termo usdo quando um programa pode automaticamente encontrar padrões e relações num conjunto de dados.

- **Classificação** - A classificação é uma sub-categoria da aprendizagem supervisionada. Classificação é o processo da receção de algum tipo de informação e atribuir-lhe um rótulo, ou uma classe.

- ***Hyperplane*** - é um subespaço cuja dimensão é -1 do que o ambiente onde ela reside. Normalmente usada para a divisão entre pontos de interesse.

#### O que é o SVM?

É um algoritmo de machine learning do tipo aprendizagem supervisionada, que pode ser usado para propósitos de classificação e regressão. Este mesmo é baseado numa ideia de achar a *hyperplane* que melhor divide todos os *datasets* em duas classes. Tendo em conta que a nossa classificação vai ser binária (gosto ou não gosto) este último paragrafo constitui uma das razões mais forte para a nossa escolha.

### Classificação

Os *support vectors* podem ser entendidos como as nossas características extraídas do HRV, e são estes que vão ser o ponto principal para a classificação, pois o classificador analisa-os e atribui-lhes um rótulo.

Algumas destas características são o número médio e desvio padrão dos batimentos cardíacos e das ondas RR consecutivas, o erro quadrático médio das diferenças entre as ondas RR, no entanto estas são apenas no dominio do tempo. No dominio da frequência são as *Very low frequency band power*, *Low frequency band power*, *High frequency band power*.

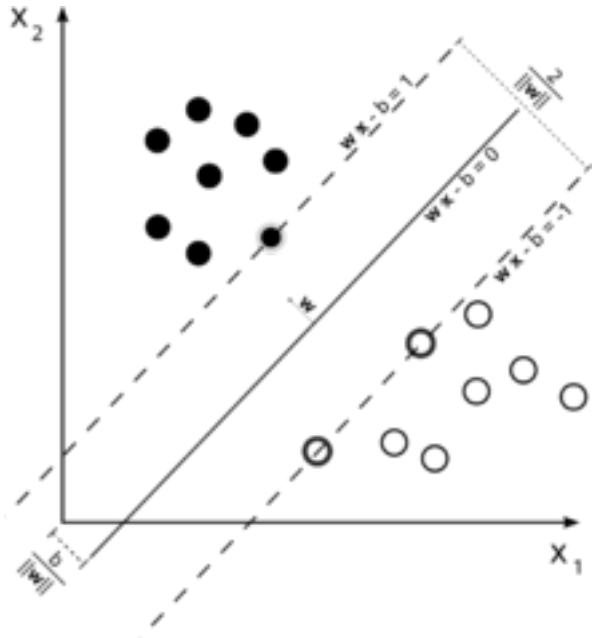


Figura 4.8: SVM.

Na figura 4.8 podemos observar como é que a melhor *Hyperplane* é descoberta. É procurada uma margem igual para os pontos mais próximos de cada classe e assim a linha que representa a *hyperplane* ficar equidistante das duas classes.

### 4.4.4 Spotipy

O Spotipy é uma biblioteca do Python que com a ajuda de outros módulos, permite usufruir de todos os *endpoints* possiveis e públicos do Spotify.
A utilização desta biblioteca, facilitou a implementação de todas as funções necessárias sem ter que criar um objeto especifico de comunicação com o spotify. A pesar de todas as funcionalidades que esta biblioteca fornece, apenas iremos listar as que foram cruciais para a realização do projeto:

- **Spotify** - É a classe que nos permite ter interação com os endpoints da Web API e que têm informação relevante das músicas e utilizadores.
- **artist_top_tracks(artist_name)** - Esta função permite-nos ter acesso ás músicas mais populares do artista passado como parametro;
- **user_playlist_add_tracks(playlist_id, user_id, tracks)** - Das principais funções em que se baseia o nosso projeto, sempre que é classificada uma reacção do utilizador como positiva, e queremos recomendar músicas, esta será utilizada;
- **user_playlist_remove_all_occurrences_of_tracks(user_id, playlist_id, tracks_id)** - Esta função permite-nos remover todas as ocorrências das músicas passadas como parâmetros de um determinada playlist. Neste casso é usado quando as emoções do utilizador foram classificadas como menos boas.

### 4.4.5 Heroku


O Heroku é uma plataforma cloud que permite dar deploy para a internet variadissimos tipos de aplicações desde Python, PHP, Java, Ruby, Node.js etc.
Achamos que seria uma mas valia ter o nosso servidor disponível em qualquer parte da internet sem ter que estar constantemente em ambiente local, tornando assim também qualquer demonstração da aplicação a funcionar muito mais fácil.
De momento a aplicação encontra-se disponivel no seguinte endereço:
**https://moodlerisel.herokuapp.com**

Para efetuar um deploy sem que ocorra qualquer tipo de erros, é importante seguir os seguintes passos:

- **Gunicorn** - é um module do Python que implementa WSGI e atua como o nosso load-balancer. É uma ferramenta crucial no core da nossa aplicação, pois pensando no futuro, iria ser utilizada por diversos utilizadores podendo assim de alguma forma balancear.

- **Criação de um ficheiro de dependências** - de forma a ser possível o Heroku reconhecer o nosso servidor como sendo Python, a criação deste ficheiro era inevitável. Nada mais contém o ficheiro do que o nome de cada depência instalada e a sua respetiva versão para o seu funcionamento e o heroku trata de as instalar remotamente.

# Capítulo 5

# Validação e Testes

Tendo em conta que a nossa aplicação foi dividida em 2 fases, a primeira de treino do classificador e a segundo como usabilidade normal, os nossos testes foram feitos faseadamente.

Na primeira fase foi desafiado ao utilizador escutar uma *playlist* de 10 músicas escolhidas aleatoriamente por nós. O objetivo seria este escutar as músicas sem qualquer tipo de interação que não a utilização dos botões gosto ou não gosto de forma a conseguir avaliar a sua opinião sobre a música. Em baixo (ver figura 5.1) podemos observar as músicas da *playlist*.


Figura 5.1: *Playlist* incial.

Como haviamos referido, a interação do utilizador irá determinar quais os dados a enviar para a firebase. Na figura 5.2, podemos observar alguns destes dados.

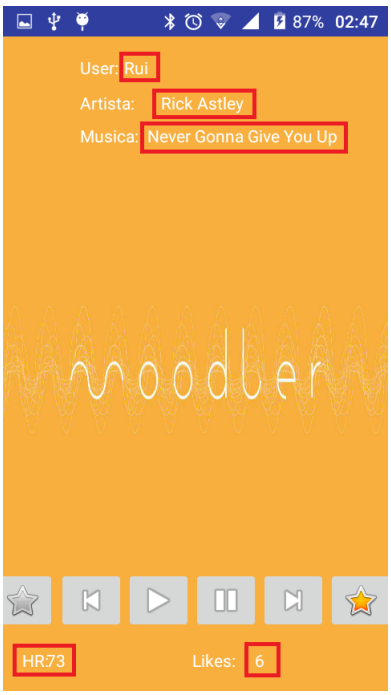


Figura 5.2: *Firebase info.*

Podemos ver que o número de *likes* é positivo, logo irá ser enviado para a *Firebase* uma indicação que o utilizador gostou da música juntamente com todos os outros campos assinalados. Na figura 5.3 podemos observar o estes mesmo campos a serem inseridos na *Firebase*. De notar que recolhemos dados de 6 pessoas, para treinar o classificador.

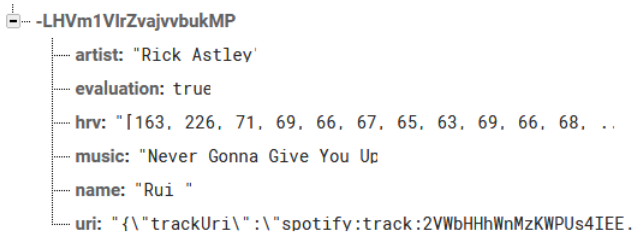


Figura 5.3: Dados na *Firebase*.

Após o envio destes dados, uma comunicação ao *Python* (ao *endpoint* - *train_classifier* ) é feita dizendo que existem novos dados para serem tidos em conta no treino do classificador e este é novamente treinado. Extraindo as características dos valores de HRV presentes na figura 5.3 e atribuído uma classe, neste caso a classe 1 por ser uma avaliação positiva.

Na segunda fase da aplicação estando o classificador treinado, foi proposto novamente um desafio ao utilizador mas este seria apenas ele escutar música da mesma playlist livremente. Sendo que o ideal será enviar valores de HRV correspondentes a 2 minutos de música, foi aconselhado este ouvir até ao fim qualquer música, contudo pode navegar na *playlist* sem qualquer restrição.
Em relação aos dados enviados para o *Python*, não são muito diferentes da primeira fase, à mesma é enviado o HRV e os *URIS* para ser classificado e usado para pesquisas respetivamente.
São extraídas as características do HRV e então classificadas.
Após a classificação são então inseridos dados na nossa base de dados MySQL, o HRV é inserido na tabela *Reaction* juntamente com a classificação atribuida pelo classificador, numa tentativa de histórico para mais tarde analisar o seu desempenho (ver figura 5.4).

| | id | user_id | track_id | hrv | evaluation | user_evaluation |
|---|---|---|---|---|---|---|
| 1 | 2 | 2 | 35pucCKt28NvIK88gpEEjI | [81, 87, 81, 79, 79, 84, 89, 87, 89, 87, 87, 84, 91, 87, 91, 90, | 1 | 1 |
| 2 | 12 | 2 | 35pucCKt28NvIK88gpEEjI | [73, 69, 74, 79, 82, 87, 83, 77, 66, 68, 70, 69, 72, 76, 80, 84, | 1 | 1 |
| 3 | 22 | 12 | 35pucCKt28NvIK88gpEEjI | [69, 73, 76, 78, 81, 86, 85, 82, 82, 77, 77, 89, 86, 87, 92, 91, | 1 | 1 |
| 4 | 32 | 12 | 35pucCKt28NvIK88gpEEjI | [71, 73, 70, 69, 74, 71, 65, 76, 82, 85, 82, 79, 76, 75, 74, 76, | 1 | 1 |
| 5 | 42 | 12 | 2VWbHHhWnMzKWPUs4IEEW9 | [68, 76, 73, 71, 73, 72, 69, 68, 68, 73, 72, 75, 79, 78, 76, 74, | 1 | 1 |

Figura 5.4: *Reaction data*

Quando a classificação é efetuada e caso esta seja positiva, são adicionadas 3 músicas do mesmo artista que atualmente se escutou (ver figura 5.5).

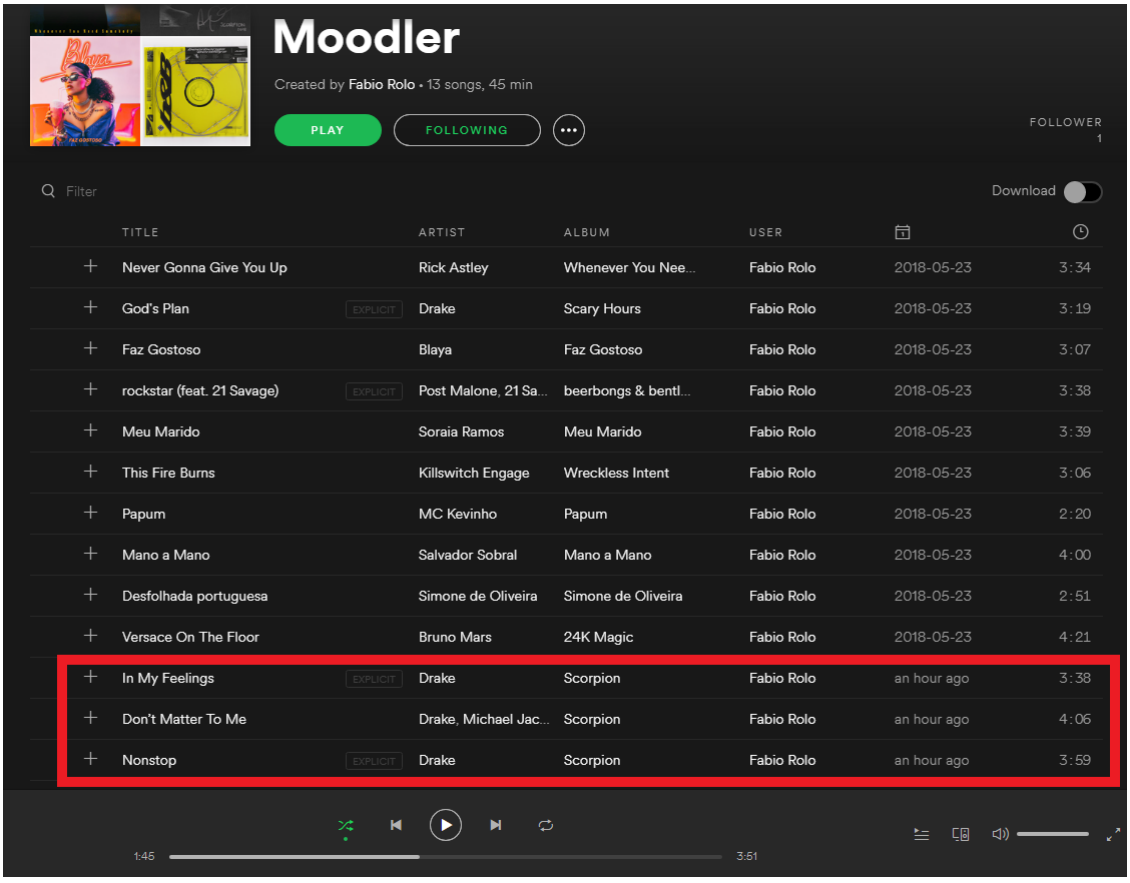


Figura 5.5: Músicas sugeridas.

# Capítulo 6

# Conclusões e Trabalho Futuro

O nosso problema residia na criação de um sistema que classificasse o estado emocional do utilizador com base na variação do seu ritmo cardíaco, contudo para isto ser possível uma parte do nosso trabalho envolvia o treino deste classificador.
Para este efeito, desenvolvidas duas aplicações, uma intermédia e uma final. No entanto só a intermédia acede diretamente à Firebase. Foi desenvolvido também um sistema distribuído para a aplicação final, com Servidor Python a aceder a uma base de dados MySql e a comunicar com clientes Android.

A utilização das bibliotecas do Spotify(Spotify Authentication e Spotify Player) foi indispensável no desenvolvimento, embora algumas limitações que as mesmas apresentam tenham sido mais trabalhosas de contornar.

A estruturação do servidor Python de acordo com o modelo MVC foi, também, determinante, pelo que o desenvolvimento organizado do servidor simplifica bastante a sua compreensão e a adição de novos métodos e funcionalidades.

Com a observação de alguns valores presentes na *firebase* correspondentes ao HRV de cada utilizador e as suas avaliações, pode-se concluir que o classificador iria ter alguns erros no seu "julgamento", pois apenas quando temos emoções fortes se consegue observar realemente uma diferença entre gostar e não gostar, quando não acontecendo as diferenças são minimas. Apesar de tudo, alguns pontos pensados tiveram que ser descartados, no entanto podem vir a ser desenvolvidos futuramente:

- Avaliação do estado emocional do utilizador;
- Melhor conhecimento do verdadeiro significado das caracteristicas extraidas e a sua relação com os sentimentos do utilizador;
- Desenvolvimento de uma interface gráfica mais apelativa;
- Melhor gestão de como são enviados os dados para o python, pois de momento só são enviados após ouvir a música na totalidade;
- Uma maior recolha de dados para o treino do classificador;
- Mais testes por parte de utilizadores, para que assim o desempenho do classificador seja testado mais a fundo e assim obter uma matriz de confusão.

.

# Apêndice A

# MySql Scripts

### Create tables Script

```
CREATE TABLE IF NOT EXISTS User
(
    id INT NOT NULL UNIQUE AUTO_INCREMENT,
    user_id VARCHAR(250) NOT NULL,
    date DATETIME NOT NULL,
    PRIMARY KEY(id)
)ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE IF NOT EXISTS Playlist
(
    id int NOT NULL unique AUTO_INCREMENT,
    user_id INT NOT null REFERENCES user(id),
    playlist_id VARCHAR(250),
    PRIMARY KEY(id)
)ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE IF NOT EXISTS Track
(
    id INT NOT NULL AUTO_INCREMENT,
        track_id varchar(250) not null,
        playlist_id INT not null references Playlist(id),
    PRIMARY KEY(id)
)ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE IF NOT EXISTS Reaction
(
```

```
    id INT NOT NULL AUTO_INCREMENT,
    user_id INT NOT NULL references user(id),
    track_id varchar(250) NOT NULL,
    hrv LONGTEXT NOT NULL,
    evaluation bool not null,
    user_evaluation bool not null,
    PRIMARY KEY(id)
)ENGINE=InnoDB DEFAULT CHARSET=utf8;

ALTER TABLE Reaction
    ADD    FOREIGN KEY (user_id)
    REFERENCES User(id)
;

ALTER TABLE Playlist
    ADD    FOREIGN KEY (user_id)
    REFERENCES User(id)
;

ALTER TABLE Track ADD FOREIGN KEY (playlist_id)
REFERENCES Playlist(id);}
```

## Drop Tables Script

```
DROP TABLE Reaction;
DROP TABLE Track;
DROP TABLE Playlist;
DROP TABLE User;
```

# Bibliografia

[1] Everyday conception of Emotion An Introduction to the Psycho logy,Anthropology and Linguistic of Emotion -James A.Russell, Jose-Miguel Fernandez-Dols Antony S.R.Manstead and J.C.Wellenkamp.

[2] James-Lange Theory of Emotion 2011.

[3] Psychiatry Research Volume 251, May 2017,Pages 192-196.

[4] Levenson RW(2003) Blood, sweat,and fears.

# Webgrafia


1 https://developer.android.com/studio/

2 https://developer.spotify.com/

3 http://flask.pocoo.org/

4 https://firebase.google.com/

5 https://www.heroku.com/

6 http://scikit-learn.org/stable/

7 https://www.news-medical.net/health/Photoplethysmography-(PPG)-(Portuguese).aspx

8 http://www.cienciaedados.com/conceitos-fundamentais-de-machine-learning/

9 https://en.wikipedia.org/wiki/Support_vector_machine

10 https://www.fullstackpython.com/green-unicorn-gunicorn.html

11 https://en.wikipedia.org/wiki/Application_programming_interface