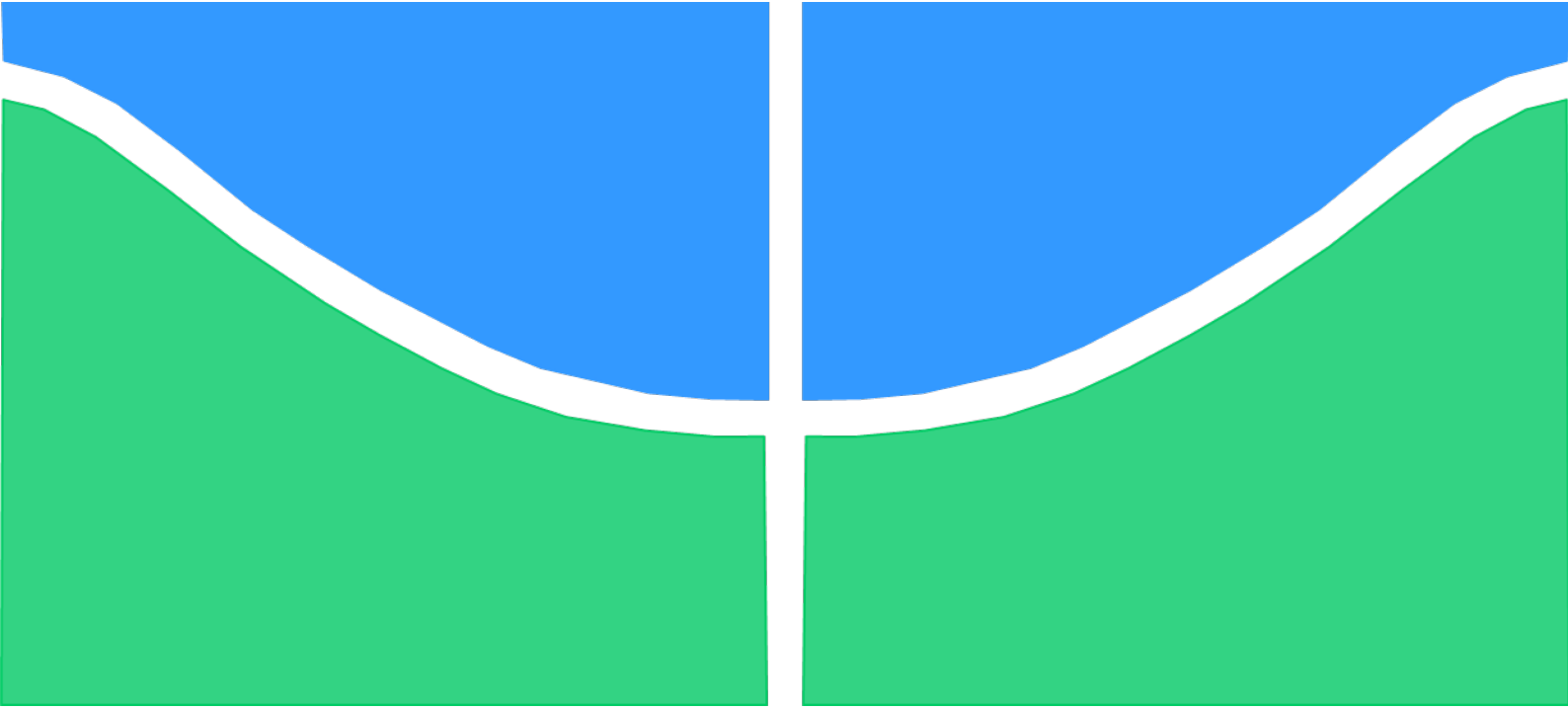


TRABALHO DE GRADUAÇÃO

# INTERFACE HOMEM-MÁQUINA BASEADA EM PLATAFORMA ANDROID PARA ELETROESTIMULAÇÃO EM TERAPIA INTENSIVA


Por,

**Gabriel Huff Theodoro**


**Brasília, julho de 2015**

UNIVERSIDADE DE BRASILIA
Faculdade de Tecnologia
Curso de Graduação em Engenharia de Controle e Automação

TRABALHO DE GRADUAÇÃO

# INTERFACE HOMEM-MÁQUINA BASEADA EM PLATAFORMA ANDROID PARA ELETROESTIMULAÇÃO EM TERAPIA INTENSIVA

Por,
**Gabriel Huff Theodoro**

*Relatório submetido como requisito parcial de obtenção de grau de Engenheiro de Controle e Automação*

Banca Examinadora

Prof. Antônio Padilha Lanari Bó, ENE/UnB _____
*Orientador*

Prof. João Luiz Quagliotti Durigan, FCE/UnB _____
*Examinador interno*

Prof. Lélio Ribeiro Soares Júnior , ENE/UnB _____
*Examinador interno*

**Brasília, julho de 2015**

**FICHA CATALOGRÁFICA**

GABRIEL, HUFF THEODORO
Interface Homem-Máquina baseada em Plataforma Android para Eletroestimulação em Terapia Intensiva ,

[Distrito Federal] 2015.

x, 72p., 297 mm (FT/UnB, Engenheiro, Controle e Automação, 2015). Trabalho de Graduação – Universidade de Brasília.Faculdade de Tecnologia.

1. Plataforma Android 2.Eletroestimulação
3. Unidade de Terapia Intensiva

I. Mecatrônica/FT/UnB II. Título (Série)

**REFERÊNCIA BIBLIOGRÁFICA**

THEODORO, G. H., (2015). Interface Homem-Máquina baseada em Plataforma Android para Eletroestimulação em Terapia Intensiva. Trabalho de Graduação em Engenharia de Controle e Automação, Publicação FT.TG-$n°$022, Faculdade de Tecnologia, Universidade de Brasília, Brasília, DF, 109p.

**CESSÃO DE DIREITOS**

AUTOR: Gabriel Huff Theodoro

TÍTULO DO TRABALHO DE GRADUAÇÃO: Interface Homem-Máquina baseada em Plataforma Android para Eletroestimulação em Terapia Intensiva.

GRAU: Engenheiro ANO: 2015

É concedida à Universidade de Brasília permissão para reproduzir cópias deste Trabalho de Graduação e para emprestar ou vender tais cópias somente para propósitos acadêmicos e científicos. O autor reserva outros direitos de publicação e nenhuma parte desse Trabalho de Graduação pode ser reproduzida sem autorização por escrito do autor.

---

Gabriel Huff Theodoro

SHIN QI 4 Conjunto 8 Casa 13 - Lago Norte.

71510-280 Brasília – DF – Brasil.

**Dedicatória**

*A todos os amigos e familiares que me ajudaram direta e indiretamente durante a minha formação.*

*Gabriel Huff Theodoro*

## Agradecimentos

*Agradecimentos ao professores e funcionários da Universidade de Brasília que contribuiram para o meu crescimento acadêmico.*

*Gabriel Huff Theodoro*

## RESUMO

A Polineuromiopatia da Doença Crítica (PNMDC) é um quadro que atinge um número cada vez maior de pacientes em Unidades de Terapia Intensiva (UTIs). Infelizmente a complexidade dos métodos disponíveis para o tratamento desse problema, bem como a falta de equipamentos adequados e pessoal capacitado contribuem para que essa situação seja agravada.

Tendo em vista esse cenário, foi criado um projeto intitulado *Prototipagem de um Estimulador Neuromuscular e Transferência de Tecnologia Para Uso Durante a Internação Hospitalar* por uma equipe de professores e alunos da Universidade de Brasília (UnB). O projeto tem como principal objetivo projetar um estimulador elétrico que possibilite o tratamento de pacientes pelo método da estimulação elétrica neuromuscular (NMES), uma operação de complexidade relativamente baixa que, ainda assim, gera resultados eficientes; bem como o treinamento de equipes de fisioterapeutas para utilizar o equipamento desenvolvido.

No contexto desse projeto, surgiu a necessidade da criação de uma Interface Homem-Máquina que possibilite a utilização do estimulador criado pela equipe da UnB por profissionais da saúde. É nesse contexto que se insere este trabalho, que consiste no desenvolvimento dessa interface.

De modo geral, este projeto tem como o objetivo o desenvolvimento de uma Interface Homem-Máquina para a plataforma Android que forneça uma curva de aprendizado rápida por parte do usuário e ao mesmo tempo implemente determinadas operações adotadas nas metodologias do projeto de estimulador da UnB, de modo que o produto final das atividades realizadas possa ser utilizado no tratamento de pacientes internados em UTIs.

Este documento descreve as etapas de desenvolvimento do produto final através da descrição do código fonte gerado e da organização do projeto, bem como os resultados obtidos através de testes com usuários potenciais.



Palavras Chave: Android, Interface Homem-Máquina, Eletroestimulação, UTI



## ABSTRACT

Critical Illness Polyneuromyopathy (CIPNM) is a complication that affects a growing number of patients on Intensive Care Units (ICUs). Unfortunatelly, the complexity of the available treatment methods as well as the lack of equipments and qualified professionals are factors that contribute to the aggravation of this problem.

Bearing this scenario in mind, a group of professors and students from the University of Brasília (UnB) created a project called *Electrostimulator Prototyping and Tecnology Transfer for Use*

*During Hospitalization*. This project's main objectives are to develop an electrical stimulator that supports Neuromuscular Electrical Stimulation (NMES) operations in the treatment of patients and to qualify selected groups of physiotherapists to use the created equipment. NMES has a relatively low complexity, yet produces very efficient results.

In this context, there was the need to create an Human Machine Interface that would allow the use of the new stimulator by health professionals. This motivated the creation of this work, which consists in developing this interface.

Generally speaking, this project aims to create an Human Machine Interface for Android that provides a steep learning curve for users and, at the same time, implements certain operations adopted in the UnB stimulator project methodologies, so it can be used for treating patients on ICUs.

This document describes the final product development stages by describing the generated source code and the project organization, as well as the results obtained in tests with potential users.

Keywords: Android, Human Machine Interface, Electrostimulation

# SUMÁRIO

# LISTA DE FIGURAS

# LISTA DE TABELAS

# LISTA DE SÍMBOLOS

### Siglas

| | |
|---|---|
| UTI | *Unidade de Terapia Intensiva* |
| PNMDC | *Polineuromiopatia da Doença Crítica* |
| PNDC | *Polineuropatia da Doença Crítica* |
| MDC | *Miopatia da Doença Crítica* |
| ENMG | *Eletroneuromiografia* |
| PACM | *Potencial de Ação Composto do Músculo* |
| PANS | *Potencial de Ação Nervoso Sensorial* |
| EDE | *Eletrodiagnóstico de Estímulo* |
| NMES | *Neuromuscular Electrical Stimulation (Estimulação Elétrica Neuromuscular)* |
| CNS | *Central Nervous System(Sistema Nervoso Central)* |
| PID | *Proporcional-Integral-Derivativo* |
| API | *Application Program Interface (Interface de Pogramação da Aplicação)* |
| XML | *Extensible Markup Language* |
| JNI | *Java Native Interface (Interface Nativa Java)* |
| SQL | *Structured Query Language (Liguagem Estruturada de Busca)* |
| URI | *Uniform Resource Identifier (Identificador Uniforme de Recurso)* |
| BLE | *Bluetooth Low-Energy (Bluetooth de Baixo Consumo)* |
| RFCOMM | *Radio Frequency Communication (Comunicação de Rádio Frequência)* |
| CSV | *Comma-Separated Values (Valores Separados por Vírgula)* |
| JPEG | *Joint Photographic Experts Group* |
| CoD | *Class of Device (Classe do Dispositivo)* |
| UUID | *Universally Unique Identifier (Identificador Único Universal)* |
| FAB | *Floating Action Button (Botão Flutuante de Ação)* |
| ADB | *Android Debugging Bridge (Ponte de Debug Android)* |
| IMC | *Índice de Massa Corporal* |

# Capítulo 1

# Introdução

## 1.1 Contextualização

A evolução dos cuidados em terapias intensivas gerou aumento nas taxas de sobrevida. Mais especificamente, 80% dos pacientes internados em unidades de terapia intensiva (UTIs) sobrevivem[1]. Contudo, muitos desses pacientes desenvolvem sequelas como consequência dos períodos de internação. Atualmente, bilhões de dólares são gastos no pós-alta desses pacientes, o que faz com que as alterações funcionais, decorrentes de tais processos, sejam consideradas problemas de saúde pública em vários países[1].

Durante a internação, é muito comum que o paciente desenvolva uma desordem conhecida como Polineuromiopatia da Doença Crítica (PNMDC), que provoca alterações funcionais neuromusculares. Estima-se que a incidência desta entre pacientes internados em UTIs por sete dias ou mais seja de 49 a 77%[2].

A PNMDC é definida na literatura como a coexistência da Polineuropatia da Doença Crítica (PNDC) e a Miopatia da Doença Crítica (MDC). Na verdade, ainda se debate se a PNDC e a MDC são desordens distintas, ou meramente manifestações em sistemas diferentes (muscular e nervoso)[3, 4, 5]. O fato é que pouco se sabe sobre as causas desses quadros.

A PNMDC é caraterizada pela dificuldade do desarme da ventilação mecânica, tetraparesia com perda de massa muscular e redução ou ausência dos reflexos tendinosos. Essas mudanças ocorrem de modo mais brando nos músculos faciais e de forma mais acentuada nos membros inferiores. Tais sintomas podem levantar a suspeita da PNMDC, porém a sua confirmação só pode ser feita pelo exame de eletroneuromiografia (ENMG) ou pela biópsia neuromuscular.

Exames de eletrodiagnóstico vêm sendo empregados nas unidades de terapia intensiva desde 1984[6], e são utilizados para determinar parâmetros como o potencial de ação composto do músculo (PACM) e o potencial de ação do nervo sensorial (PANS).

Pesquisas apontam que anomalias podem ser detectadas pelo ENMG a partir de 72 horas da admissão[4]. Baixas amplitudes do PACM e do PANS podem ser interpretados como indicativos da PNMDC.

No entanto, a PNMDC acaba sendo deixada em segundo plano de diagnóstico, já que as avaliações por biópsia e ENMG não estão disponíveis na UTI. Isso ocorre por conta da ausência de equipamentos, falta de profissionais qualificados para realizar e interpretar os exames, e pelo fato de serem invasivos, o que aumenta a complexidade da operação e introduz riscos de infecções.

Como alternativa, pode-se utilizar o eletrodiagnóstico de estímulo (EDE) para detectar alterações de excitabilidade neuromuscular através de dados como a cronaxia, reobase e acomodação, medidas relacionadas a excitabilidade to tecido neuromuscular. Esses dados são obtidos por meio de eletrodos de superfície e geradores de corrente. Sua complexidade é relativamente mais baixa, pelo fato de ser um exame não invasivo.

Apesar de o ENMG ser relativamente mais preciso que o EDE para determinar a natureza, importância e a localização de uma lesão neuromuscular periférica, este não fornece nenhuma parametrização que possibilite a determinação de qual seria a melhor forma de tratamento por meio de estimulação elétrica.

Os parâmetros adquiridos pelo EDE permitem não só o monitoramento das alterações neuromusculares, como também o tratamento da PNMDC. Os valores de cronaxia, reobase e acomodação são fundamentais para a parametrização da estimulação elétrica neuromuscular (NMES), que é utilizada para realizar o tratamento.

A NMES consiste em contrair a musculatura esquelética do paciente por meio de eletrodos de superfície. Possui uma vantagem pelo fato de não depender da cooperação do paciente, podendo ser iniciada mesmo com o paciente sedado[7]. Foram documentados resultados positivos com a sua utilização, tais como: redução da perda de massa muscular, ganho de força muscular, diminuição na incidência de polineuropatia e aumento das taxas de sucesso no desmame[7]. Tais fatores fazem com que a NMES venha sendo considerada uma alternativa aceitável para o tratamento de pacientes com PNMDC.

## 1.2 Definição do problema

Apesar das vantagens oferecidas pelo uso da NMES no tratamento da PNMDC, ainda há barreiras para a sua implementação nas UTIs.

Pesquisas preliminares apontaram que não existe no mercado mundial um equipamento específico para essa função. O tempo gasto para se realizar o procedimento em questão em estimuladores elétricos genéricos é de em média 120 minutos, só para os membros inferiores, segundo estudo piloto realizado por colaboradores desse trabalho. Além disso, as normas técnicas de funcionamento das UTIs no Brasil ditam que, por padrão, deve haver um fisioterapeuta para cada dez pacientes. Tais fatos inviabilizam a prática desse tipo de tratamento.

Por conta disso foi idealizado um projeto intitulado "Prototipagem de um estimulador neuromuscular e transferência de tecnologia para uso durante a internação hospitalar" por uma equipe de professores e alunos da Universidade de Brasília, que possui como objetivos:

- Desenvolvimento de um estimulador elétrico que possibilite a realização de EDEs e NMESs.
- Treinamento de equipes em UTIs selecionadas para utilizar o estimulador desenvolvido.
- Coleta de dados de pacientes submetidos aos tratamentos realizados para a validação da metodologia terapêutica.

No entanto, para a realização de tal projeto, é necessária a implementação de uma interface entre o estimulador e o usuário, no caso o fisioterapeuta.

O desenvolvimento dessa interface é o tema central deste projeto, que consiste basicamente em criar e testar uma interface simples e amigável, porém completa para ser utilizada por fisioterapeutas durante o tratamento de pacientes internados em UTIs utilizando o estimulador desenvolvido pela equipe da UnB.

## 1.3 Objetivos do projeto

Dada a situação descrita na seção 1.2, entre os objetivos principais desse projeto estão incluídos:

- Criar uma aplicação para dispositivos móveis que poderá ser utilizada para o controle de estimuladores elétricos desenvolvidos pela equipe da UnB. A aplicação em questão deve ser desenvolvida para sistemas Android, com compatibilidade mínima para a API 14 do sistema (Versão 4.0 - Ice Cream Sandwich). Utilizando a aplicação desenvolvida, o usuário deverá ser capaz de cadastrar pacientes, afim de se manter um histórico do tratamento, e selecionar, parametrizar, acompanhar e registrar três tipos de operações de diagnóstico: Cronaxia, Reobase ou Acomodação; e duas operações de terapia: Terapia em Malha Aberta ou Terapia em Malha Fechada.
- Desenvolver uma interface gráfica simples, de modo a possibilitar o aprendizado rápido por parte do usuário, porém funcional, provendo o controle necessário para a realização das operações necessárias durante o tratamento de pacientes.
- Criar um protocolo de comunicação entre a aplicação e estimulador, de modo que o primeiro possa controlar o segundo de acordo com comandos enviados pelo usuário via interface gráfica. Além disso, o estimulador deverá enviar informações para a aplicação de modo que o usuário possa ter uma realimentação visual acerca de o que está acontecendo.

## 1.4 Organização do Documento

Este documento busca descrever detalhes relativos a implementação e desenvolvimento da aplicação descrita na seção 1.2 de uma maneira clara e ao mesmo tempo técnica. Os próximos capítulos estão organizados da seguinte maneira:

### Capítulo 2 - Fundamentos Teóricos

Descreve conceitos teóricos específicos nas áreas de Eletroestimulação e Programação em Android necessários para o melhor entendimento da implementação descrita no Capítulo 3.

### Capítulo 3 - Desenvolvimento

Descreve detalhes técnicos específicos acerta da implementação da aplicação, bem como algoritmos utilizados, organização de dados, detalhes do código fonte, etc.

### Capítulo 4 - Resultados e Discussão

Descreve os resultados obtidos com a aplicação desenvolvida. Esses resultados estão divididos em: Sistema de Armazenamento de Dados de Pacientes, que descreve os resultados do sistema utilizado para armazenar os dados da aplicação; Comunicação com o Estimulador, que descreve a interação entre a aplicação desenvolvida e o estimulador da UnB; e Interface Gráfica, que descreve a interação entre a aplicação e o usuário.

### Capítulo 5 - Conclusão

Compara os resultados obtidos descritos no Capítulo 4 com o que foi proposto no item 1.3, apontando algumas pendências e pontos que podem ser melhorados, assim como pontos que superaram as expectativas.

# Capítulo 2

# Fundamentos Teóricos

*Nesse capítulo serão apontados e descritos alguns conceitos específicos necessários para o entendimento do projeto, referentes às áreas de Eletroestimulação e Programação em Android.*

## 2.1 Eletroestimulação

Essa seção busca definir conceitos relevantes para este projeto na área Eletroestimulação, tais como a descrição dos sinais elétricos e músculos estimulados nos procedimentos de eletrodiagnóstico de estímulo (EDE) e estimulação elétrica neuromuscular (NMES), além de descrições técnicas das operações suportadas pelo estimulador da UnB.

### 2.1.1 Características gerais de procedimentos de Eletroestimulação

Entende-se por procedimento de Eletroestimulação operações que consistem em gerar contrações musculares por meio de impulsos elétricos aplicados em tecidos neuromusculares, conceito que generaliza as operações de EDE e NMES. Neste documento, será definido o termo Diagnóstico para se referir à procedimentos de EDE. Da mesma maneira, o termo Terapia será usado como referência à procedimentos de NMES.

Tanto operações de Diagnóstico como operações de Terapia são procedimentos não-invasivos que enviam sinais elétricos via eletrodos conectados a pele do paciente. A diferença funcional entre esses dois conceitos está ligada ao fato de que o objetivo primário dos Diagnósticos é obter parâmetros que serão utilizados para a realização de Terapias e realizar o diagnóstico de doenças, enquanto operações de Terapia são realizadas com o objetivo de gerar contrações musculares, afim de tratar o paciente. Os parâmetros obtidos através de operações de Diagnóstico estão descritos em mais detalhes na seção 2.1.3, enquanto a utilização desses parâmetros está especificada na seção 2.1.4.

Para que as contrações musculares sejam geradas, os trens de impulsos elétricos transmitidos por eletrodos buscam copiar o comportamento das excitações produzidas pelo Sistema Nervoso Central (CNS). Na interface entre os tecidos excitáveis e eletrodos, esses impulsos depolarizam as

membranas dos decidos, produzindo potenciais de ação artificiais[8].

Os sinais elétricos gerados nesses procedimentos consistem em pulsos de corrente elétrica espaçados homogeneamente no tempo, apresentando as seguintes características (ilustradas na figura 2.1):

- *Forma de Onda:* O formato de pulso em relação ao tempo. Esse formato resulta em diferentes performances em termos de limiares de depolarização, corrosão de eletrodos e danos aos tecidos[8]. Nas metodologias adotadas nesse projeto serão usadas as formas quadrada e exponencial.
- *Polaridade:* Indica se o pulso inverte ou não a polaridade da corrente no decorrer do tempo (i.e. se possui correntes positivas e negativas). Pulsos que não invertem a polaridade da corrente são chamados *unipolares*, enquanto os que invertem são denominados *bipolares*.
- *Frequência:* Frequência da ocorrência dos pulsos.
- *Amplitude:* Intensidade de um pulso, geralmente medida em $mA$.
- *Largura de Pulso:* Duração de um pulso, geralmente medida em $\mu s$.

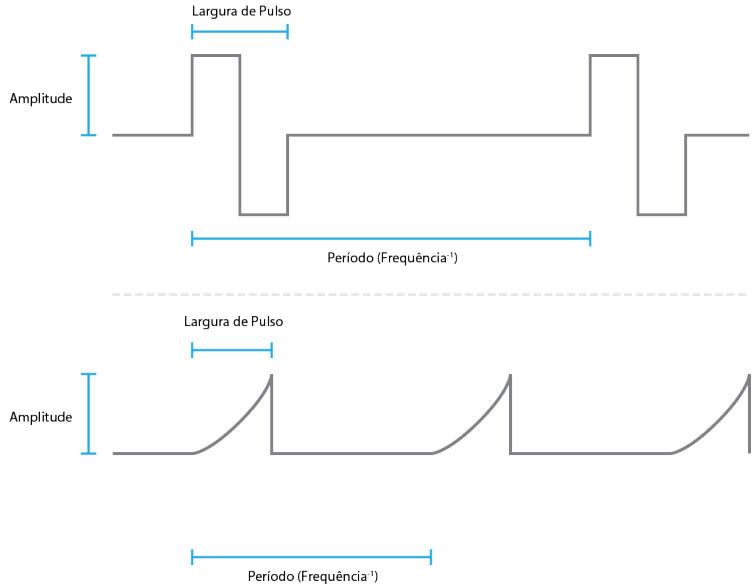



Figura 2.1: Pulsos de eletroestimulação. O sinal na parte superior da imagem é constituído por pulsos quadrados bipolares, enquanto o sinal na parte inferior corresponde a uma sequência de pulsos exponenciais unipolares.

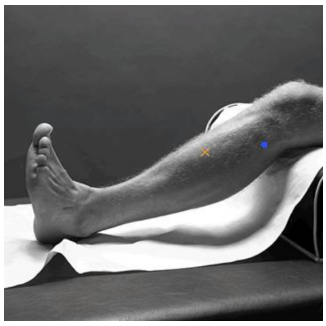
### 2.1.2 Músculos de membros inferiores estimulados

Quanto ao posicionamento dos eletrodos nos músculos, segue-se as referências fornecidas pela tabela 2.1, representadas pelas figuras 2.2(a), 2.2(b), 2.2(c), 2.2(d) e 2.2(e). Essas referências

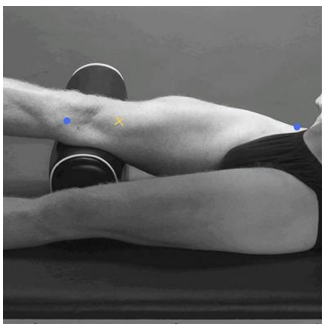
assumem que o paciente deve estar em decúbito dorsal, cabeceira a 45°, joelhos em flexão de 15° e braços recostados sobre o leito.

| Músculo | Referência anatômica |
|---|---|
| Tibial Anterior | Orientação: linha entre a cabeça da fíbula e o maléolo medial – O eletrodo negativo deve ser posicionado sobre esta linha no terço proximal a cabeça da fíbula. O eletrodo positivo deve ser posicionado em volta do tornozelo. |
| Vasto Medial | Orientação: linha entre a crista ilíaca ântero-superior e o côndilo medial do fêmur – A partir do côndilo medial, o eletrodo negativo deve ser posicionado a 20% da linha e o eletrodo positivo deve ser posicionado em volta do tornozelo. |
| Vasto Lateral | Orientação: linha entre a crista ilíaca ântero-superior e o côndilo lateral do fêmur – A partir do côndilo lateral 03 do fêmur o eletrodo negativo deve ser posicionado a 30% da linha e o eletrodo positivo deve ser posicionado em volta do tornozelo. |
| Reto Femoral | Orientação: linha entre a crista ilíaca ântero-superior e 04 a parte superior da patela – O eletrodo negativo deve ser posicionado a 50% da linha e o eletrodo positivo deve ser posicionado em volta do tornozelo. |
| Bíceps Braquial | Orientação: linha entre o acrômio e o epicôndilo medial - A partir do epicôndilo medial o eletrodo positivo deve ser posicionado na distância média entre o Q50% e o Q75% e eletrodo negativo no Q25%. |

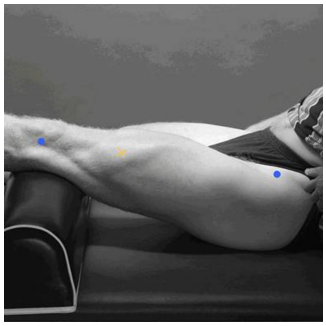
Tabela 2.1: Referência para o posicionamento de eletrodos.



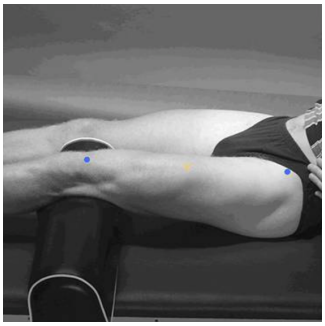
(a) Tibial Anterior



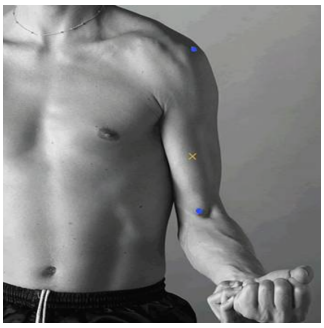
(b) Vasto Medial



(c) Vasto Lateral



(d) Reto Femoral



(e) Bíceps Braquial

Figura 2.2: Ilustração das referências para o posicionamento de eletrodos. Os pontos azuis são usados para determinar as linha de referência. A cruz amarela indica o posicionamento do eletrodo negativo.

### 2.1.3 Operações de Diagnóstico

Operações de Diagnóstico são fundamentais para a definição dos parâmetros que serão utilizados no tratamento da PNMDC por meio da NMES[9].

Nesse documento serão abordados três subtipos de Diagnóstico:

- *Reobase*
- *Cronaxia*
- *Acomodação*

A Reobase é definida como a menor amplitude do sinal de corrente, em $mA$, necessária para produzir uma contração mínima do músculo em questão, utilizando pulsos quadrados, com largura de 1000 $ms$ espaçados em 2000 $ms$[9].

A Cronaxia é definida como a menor largura de pulso, em milissegundos, necessária para produzir uma contração mínima do músculo em questão, utilizando pulsos quadrados, com amplitude igual ao dobro da Reobase espaçados em 2000 ms[9].

A Acomodação é obtida de maneira similar à Reobase, porém utilizando pulsos exponenciais, ou seja, é definida como a menor amplitude do sinal de corrente, em mA, necessária para produzir uma contração mínima do músculo em questão, utilizando pulsos exponenciais, com largura de 1000 ms espaçados em 2000 ms[10, 11].

Essas grandezas são obtidas com o objetivo de parametrizar operações de Terapia para gerar resultados otimizados e podem auxiliar no diagnóstico da PNMDC.

### 2.1.4 Operações de Terapia

Operações de Terapia podem utilizar tanto parâmetros obtidos através de Diagnósticos quanto valores empíricos para gerar sinais elétricos que causarão contrações no músculo em questão de modo a efetivamente realizar o tratamento.

Utilizando-se como base a metodologia utilizada pela equipe dos desenvolvedores do estimulador da UnB, as operações de Terapia realizam a eletroestimulação utilizando pulsos quadrados unipolares a uma frequência de 20 a 100 $Hz$ com o dobro da largura de pulso encontrada na Cronaxia e intensidade suficiente para se gerar contrações visíveis e rigorosas.

A amplitude da sequência de pulsos é modulada por um envelope determinado pelos seguintes parâmetros (ilustrados na figura 2.3):

- *Tempo de Estimulação:* Tempo no qual a amplitude dos pulsos é não-nula.
- *Tempo de Repouso:* Tempo no qual a amplitude dos pulsos é nula.
- *Rampa de Subida:* Tempo necessário para a amplitude dos pulsos crescer linearmente de 0 até o seu valor máximo. Pode também ser representada por uma fração do Tempo de Estimulação.
- *Rampa de Descida:* Tempo necessário para a amplitude dos pulsos descer linearmente do seu valor máximo até 0. Pode também ser representada por uma fração do Tempo de Estimulação.

Nesse documento serão abordados dois subtipos de Terapia: *Terapia em Malha Aberta* e *Terapia em Malha Fechada*. A diferença primordial entre os dois é a realimentação de informação: a Terapia em Malha Fechada utiliza uma realimentação da contração muscular para regular a pequenas variações na amplitude dos pulsos, enquanto a Terapia em Malha Aberta não oferece essa capacidade.

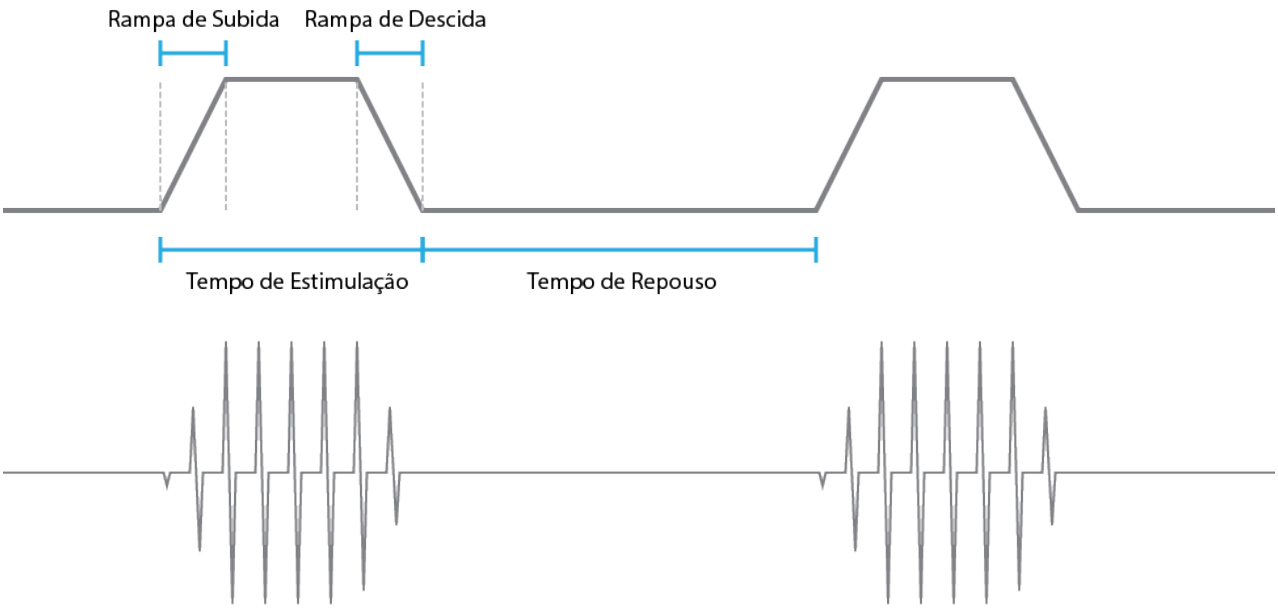



Figura 2.3: Modulação da amplitude dos pulsos de Terapia por curvas periódicas. A curva na parte superior da imagem representa o envelope de amplitude do sinal. A curva na parte inferior representa o sinal modulado pelo envelope de amplitude.

O uso de estratégias de realimentação possibilita a compensação dos efeitos causados por erros introduzidos por imprecisões no modelo adotado e perturbações. Porém, a aplicação de tais metodologias possui algumas dificuldades relacionadas a limitações de sensores e atuadores[12]. Um dos tipos mais comuns de controles mais simples utilizados em porcessos de estimulação com realimentação é o uso de controladores proporcional-integral-derivativos (PIDs), que possuem uma implementação simples e não depende da planta para a configuração de seus parâmetros. No caso de implementações mais complexas, pode-se considerar o modelo da junta no processo de desenvolvimento do controlador. Em todos os casos, o controle é realizado através da realimentação da posição adquirida por sensores.

## 2.2 Programação em Android

Alguns conceitos relativos ao desenvolvimento em Android devem ser levados em consideração para o melhor entendimento do projeto. Essa seção busca fornecer breves descrições desses conceitos.

### 2.2.1 Organização do Código-Fonte

A organização dos arquivos de um projeto Android depende do *software* utilizado (IDE - *Integrated Development Environment*) para o desenvolvimento. Atualmente os ambientes mais utilizados são Android Studio e Eclipse. No final de 2014, o Android Studio passou a ser a IDE oficial para o desenvolvimento de projetos[13].

Apesar de eventuais diferenças entre os ambientes disponíveis atualmente, todos organizam o código-fonte de seus projetos da mesma maneira. Abaixo segue uma breve descrição de alguns dos componentes mais importantes nessa organização:

**AndroidManifest**

Um arquivo XML que descreve a aplicação em um nível geral. Nele podem ser incluídas informações como o nome do pacote da aplicação; versão; permissões de sistema necessárias, como por exemplo acesso à internet, acesso ao cartão SD, acesso ao Bluetooth, etc; declaração de Componentes de Aplicação (descritos no item 2.2.2) e outras definições que descrevem as funcionalidades da aplicação.

**Pacotes Java**

Conjunto de pacotes que efetivamente geram a aplicação. A linguagem de programação utilizada é Java, adicionalmente a uma camada de Frameworks para o desenvolvimento específico em Android. Também é possível a utilização de código nativo (C/C++) através de JNI - *Java Native Interface* - porém o uso desse recurso é recomendado em situações específicas.

**Recursos**

Definições, constantes ou propriedades gráficas definidas antes da compilação, como por exemplo imagens, strings, valores numéricos, layouts, etc. Esses recursos podem ser dependentes de características do dispositivo, como por exemplo idioma, versão do sistema operacional, tamanho ou densidade da tela.

Por exemplo, um mesmo recurso de string pode ser declarado paralelamente em português e inglês. Caso o usuário selecione 'Português' como o idioma do sistema, o recurso utilizado durante a execução da aplicação será a versão declarada como português.

Essa arquitetura busca isolar elementos que não precisam ser gerados em tempo de execução com o objetivo de simplificar mudanças na parte da aplicação visível ao usuário.

### 2.2.2 Componentes de Aplicação

O conceito de aplicação no contexto de um projeto Android pode ser descrito como um conjunto de Componentes de Aplicação rodando em um ou mais processos Linux (o Android é puramente um sistema operacional baseado em Linux). Componentes de Aplicação são entidades declaradas no AndroidManifest da aplicação (descrito no item 2.2.1). Existem 4 tipos de Componentes de Aplicação:

#### 2.2.2.1 Activities

Activities são Componentes de Aplicação visíveis ao usuário, ou seja, fornecem um conjunto de interfaces gráficas de modo a possibilitar a interação deste com a aplicação. São implementadas no código-fonte em Java através de extensões da classe *Activity*.

Possuem um ciclo de vida diretamente relacionado a essa interação com o usuário, ou seja, eventos tais como mudança entre telas de diferentes aplicações, mudança da orientação do dispositivo (vertical ou horizontal) ou qualquer outro tipo de mudança de foco de telas por parte do usuário afetam diretamente o seu ciclo de vida. Activities que não possuem o foco do usuário podem até ser destruídas pelo sistema.

Isso implica que operações mais longas, que devem ser desvinculados do foco do usuário em relação a aplicação, como por exemplo downloads de arquivos extensos ou comunicações Bluetooth, devem ser preferivelmente implementadas em outro tipo de componente. Nesses casos a opção ideal seria o uso de um Serviço, componente que será descrito no item a seguir.

#### 2.2.2.2 Services

Services, ou Serviços, são Componentes de Aplicação que, assim como Activities, possuem um ciclo de vida definido. A diferença primordial entre os dois é que Serviços, ao contrário de Activities, são, a princípio, independentes do foco do usuário. São implementados no código-fonte em Java através de extensões da classe *Service*.

O seu ciclo de vida é gerenciado por si mesmo ou por outros Componentes de Aplicação. Serviços podem ser iniciados de duas maneiras distintas: starting ou binding. Cada uma implica em um ciclo de vida diferente.

Serviços executados por starting são iniciados de modo que o seu ciclo de vida é independente do componente que o iniciou. Precisam ser explicitamente parados (por si ou por componentes externos) para depois serem destruídos.

Serviços executados por binding são iniciados por Componentes de Aplicação externos e o seu ciclo de vida está diretamente relacionado a estes componentes.

Além disso, um serviço pode rodar em foreground. Esse recurso pode ser utilizado em situações nas quais o usuário está ciente da existência do serviço em questão, como por exemplo no caso de um serviço que toca músicas. Do ponto de vista do sistema, um serviço em foreground possui uma prioridade maior do que um serviço normal, o que implica que este é menos suscetível a ser destruído em situações de recursos escassos.

#### 2.2.2.3 Content Providers

Content Providers são responsáveis por fornecer conteúdo da aplicação para outros componentes. Há duas possíveis formas de conteúdo a ser fornecido: arquivos ou dados relacionais (de modo mais comum, dados armazenados em bancos SQLite). São implementados no código-fonte em Java

através de extensões da classe *ContentProvider*.

O conteúdo é acessado por outras aplicações por meio de URIs com o esquema content. Cabe ao Content Provider interpretar essas URIs e relacioná-las com o conteúdo disponível.

Não possuem um ciclo de vida complexo: são apenas criados quando acessados pela primeira vez e assim permanecem, até que sejam destruídos pelo sistema.

#### 2.2.2.4 Broadcast Receivers

Broadcast Receivers são Componentes de Aplicação que respondem a eventos transmitidos pelo sistema, tais como mudança do estado do rádio Wi-Fi, nível baixo de bateria, desligamento da tela e outros eventos provenientes do sistema, assim como eventos definidos em aplicações. São implementados no código-fonte em Java através de extensões da classe *BroadcastReceiver*.

Geralmente são utilizados para fazerem uma conexão entre esses eventos e outros Componentes de Aplicação, por exemplo atualizar a interface gráfica de uma Activity ou iniciar um Serviço.

### 2.2.3 Design Material

Em 2014, foi lançado a versão 5.0 do Android, chamada Lollipop. Uma das maiores mudanças dessa atualização foi a introdução do *Material Design*[14], ou Design Material, um conceito desenvolvido pelo Google, empresa que desenvolve o Android, relativo ao design de aplicações.

O Design Material consiste no uso de animações responsivas, cores vivas, formas simples e efeitos de profundidade através do uso de luzes e sombras, afim de criar uma materialização do conteúdo de uma forma minimalista em composições de 'papel e tinta'. Foi uma grande mudança na parte visual do sistema operacional, e está documentada como um conjunto de regras e padrões disponibilizados pelo próprio Google para auxiliar no design de novas aplicações.

Figura 2.4: Conceitos de Design Material. Formas simples organizadas no espaço tridimensional através do uso de luzes e sombras.

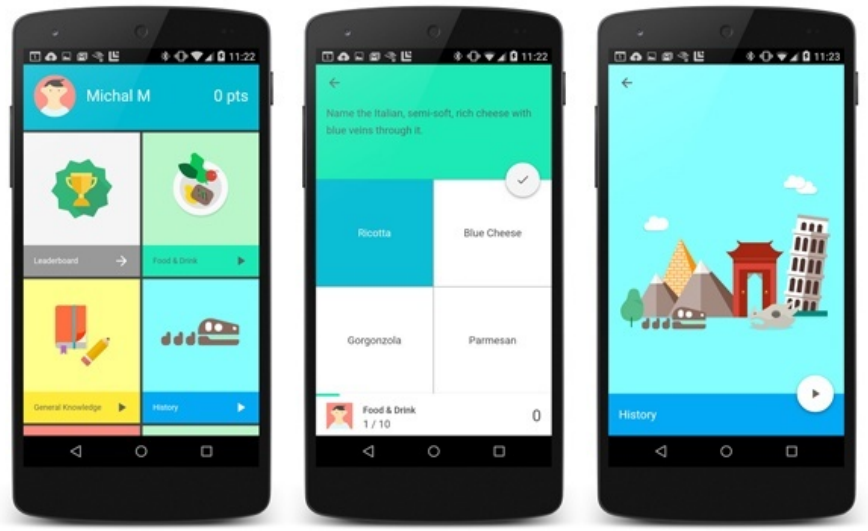



Figura 2.5: Exemplo de Design Material em aplicações.

### 2.2.4 Fragmentos

A API de Fragments foi lançada no Android 3.0. O conceito de Fragment, ou Fragmento, pode ser descrito como um pedaço (ou, como o nome já diz, fragmento) de uma tela ou comportamento fornecido por uma Activity (descrita no item 2.2.2). O seu uso busca modularizar o conteúdo de uma Activity, de modo a facilitar o reaproveitamento de componentes visuais e tornar o código mais organizado.

O ciclo de vida de um Fragmento está intimamente ligado à Activity que o criou, o que significa que, consequentemente, está ligado também ao foco do usuário em relação ao conteúdo que ele fornece.

A modularização fornecida pelo uso de fragmentos possibilita, entre outras coisas, a criação de interfaces gráficas flexíveis em relação ao tamanho da tela do dispositivo, conforme ilustrado na figura 2.6.

### 2.2.5 SQLite

O Android utiliza bancos de dados SQLite para o armazenamento de conteúdo em formato relacional. As APIs necessárias para a criação e manipulação destes bancos de dados estão disponíveis no pacote Java android.database.sqlite, e possibilitam o uso de funcionalidades básicas, tais como busca, inserção, atualização ou remoção de informações não-voláteis no dispositivo. Pode-se também usar a própria sintaxe SQL para a manipulação destes mesmos dados.

Bancos SQL são criados e armazenados no diretório interno da aplicação, o que implica que, a princípio, os dados estão disponíveis apenas no contexto dessa aplicação. Por esse motivo, são frequentemente vinculados à Content Providers (descritos no item 2.2.2) de modo que a abstração do conceito de informação saia do nível do sistema de arquivos e passe ao nível de comunicação entre Componentes de Aplicação.


Figura 2.6: Modularização da interface gráfica através do uso de Fragmentos. Dependendo do tamanho disponível da tela, os fragmentos podem ser apresentados de maneiras diferentes.

### 2.2.6 Bluetooth Clássico

Bluetooth é um protocolo de comunicação sem fio criado pela empresa sueca Ericsson em 1984[15] com o propósito inicial de ser uma alternativa sem fio ao RS-232.

O Android oferece APIs para o suporte da pilha do protocolo de comunicação Bluetooth. Atualmente são suportados dois tipos de Bluetooth: o Classic Bluetooth, ou Bluetooth Clássico, utilizado em operações mais intensas do ponto de vista de energia, e Bluetooth LE (low-energy), disponível a partir do Android 4.3 e utilizado em operações que requerem menos energia. O primeiro tipo será brevemente descrito nessa subseção.

Para o uso desses recursos, é necessária a declaração da permissão BLUETOOTH no *AndroidManifest* da aplicação (descrito no item 2.2.1).

Com as APIs disponíveis para o Bluetooth Clássico, é possível realizar as seguintes operações:

- Buscar dispositivos ativos próximos. Dispositivos são abstraídos por objetos da classe BluetoothDevice, que contém informações necessárias para a realização de conexões, como por exemplo endereços MAC, classes de dispositivo, etc.
- Listar dispositivos pareados. Dispositivos pareados tem suas informações armazenadas no sistema para conexões posteriores.
- Estabelecer canais RFCOMM (radio frequency communication) com dispositivos encontrados ou pareados. Esses canais são simplesmente protocolos da camada de transporte que emulam portas seriais RS-232.

# Capítulo 3

# Desenvolvimento

*Nesse capítulo serão descritos as características e metodologias utilizadas no desenvolvimento da aplicação.*

## 3.1 Definições

Alguns conceitos foram definidos para modelar o funcionamento da aplicação. Tais definições são descritas nessa seção.

### 3.1.1 Paciente

Um paciente representa, do ponto de vista do programa, uma entidade primitiva cadastrada na aplicação. Possui características, tais como nome, data de nascimento, gênero, data de internação, data de saída, etc. Essas características podem ser primárias, ou seja, devem ser necessariamente definidas para um paciente, ou secundárias, que não precisam necessariamente ser definidas.

Um paciente também possui um histórico de *Sessões*. Cada paciente pode ter apenas uma *Sessão* aberta a cada momento.

Atualmente, as seguintes características primárias são definidas:

- Nome
- Gênero
- Data de Nascimento
- Data de Entrada

As características secundárias são:

- Data de Saída
- Telefone

- E-mail
- Observações

### 3.1.2 Sessão

Uma sessão é um agrupamento de *Operações*.

Sessões podem estar abertas ou fechadas. Uma sessão aberta possibilita a adição de novas *Operações*, enquanto sessões fechadas não fornecem esse recurso.

Normalmente, uma sessão deve necessariamente estar vinculada a um *Paciente*. Isso não se aplica, no entanto, a *Sessões Rápidas*.

Possuem as seguintes características:

- Data de início
- Data de fechamento
- Tag (valor utilizado pelo usuário para qualificar a sessão)

### 3.1.3 Sessão Rápida

Sessões rápidas são sessões que não estão vinculadas a nenhum *Paciente*. Podem ser criadas em situações na qual o usuário não queira adicionar um novo *Paciente* no banco de dados da aplicação.

Sessões rápidas podem ser descartadas pela a aplicação assim que o usuário deixa de manipulá-las. Por isso, podem ser adicionadas a *Pacientes* existentes (ou novos) a qualquer momento.

### 3.1.4 Operação

Operação é uma abstração para procedimentos de eletroestimulação criados em processos de comunicação com o estimulador. São quantificadas por resultados, onde cada resultado pode ser descrito como um conjunto de valores numéricos que quantifiquem a operação. O significado desse valor numérico dependerá do tipo da operação. Atualmente existem cinco tipos: Cronaxia, Reobase, Acomodação, Terapia em Malha Aberta ou Terapia em Malha Fechada.

Operações são classificadas em grupos, que atualmente pode ser Diagnóstico ou Terapia. Dependendo do grupo ao qual pertencem, podem apresentar características diferentes. Diagnósticos possuem as seguintes características:

- Tipo (Cronaxia, Reobase ou Acomodação)
- Data

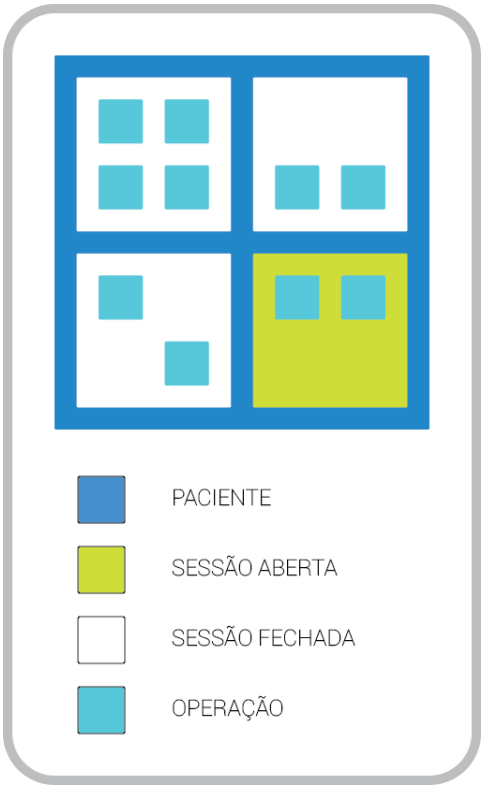



Figura 3.1: Abstração dos conceitos de Paciente, Sessão e Operação.

Enquanto terapias são caracterizadas da seguinte maneira:

- Tipo (Malha Aberta ou Malha Fechada)
- Data de início
- Data de fechamento

## 3.2 Funcionalidades da Aplicação

Nessa sessão estão listadas as funcionalidades da aplicação desenvolvida.

De modo geral, o objetivo principal da aplicação é fornecer uma interface gráfica que seja capaz de manipular um conjunto de dados que representam os conceitos de Paciente, Sessão e Operação apresentados na seção 3.1. Mais especificamente, um usuário deve ser capaz de:

- Visualizar, adicionar, modificar ou remover dados relativos a pacientes e armazenar esses dados de maneira não-volátil. Quando um paciente é removido, todas as suas sessões devem ser automaticamente removidas.
- Criar sessões para pacientes, desde que o paciente em questão não possua uma sessão aberta.

- Remover sessões de pacientes. Quando uma sessão é removida, todas as suas operações devem ser automaticamente removidas.
- Criar sessões rápidas. Apenas uma sessão rápida pode existir em cada momento. Criar uma nova sessão rápida implica em apagar a antiga sessão rápida, se existente.
- Adicionar sessões rápidas a pacientes já existentes ou a novos pacientes.
- Visualizar sessões.
- Configurar conexões Bluetooth com estimuladores durante sessões abertas.
- Configurar pares canal-músculo usados durante sessões abertas
- Adicionar ou remover operações de sessões abertas.
- Visualizar resultados de operações.

## 3.3 Propriedades do Projeto

O projeto foi desenvolvido no *Android Studio*, versão 1.2.x, que é a plataforma oficial para o desenvolvimento Android.

Todas as informações relativas à estrutura da aplicação podem ser encontradas no arquivo *AndroidManifest.xml* do projeto. Vale ressaltar que alguns detalhes adicionais, como por exemplo versão do projeto e versões mínima e alvo do sistema operacional podem ser encontrados no arquivo *build.graddle* do módulo da aplicação (essas informações poderiam ter sido declaradas no arquivo *AndroidManifest.xml*, no entanto as boas práticas do Android Studio estabelecem que essas informações devem ser declaradas no projeto *Graddle*). Todas essas informações estão descritas a seguir:

### Características do Pacote

O nome definido para o pacote da aplicação foi *com.huff.tg2*.

O código da versão atual do projeto é 1 e o seu nome é *1.0*.

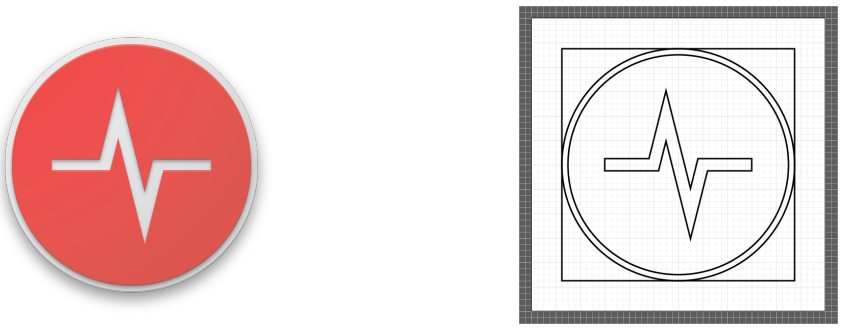
### Versões Mínima e Alvo do Sistema Operacional

A versão mínima do Android é 4.0 (API 14 - *Ice Cream Sandwich*).

A versão alvo do Android é 5.1 (API 22 - *Lollipop MR1*)

### Identidade do Produto

A aplicação recebeu um nome temporário de *TG2* durante o desenvolvimento.



(a) Ícone da aplicação (b) Geometria do Ícone

Figura 3.2: Desenvolvimento do ícone da aplicação

Foi criado um ícone para a aplicação utilizando-se o software *Illustrator*, seguindo os padrões de iconografia do Design material. O produto final ilustrado na figura 3.2.

#### Permissões

A aplicação requer as seguintes permissões do sistema:

#### BLUETOOTH

Necessária para realizar operações básicas com as APIs de bluetooth.

#### BLUETOOTH_ADMIN

Necessária para realizar a busca de dispositivos Bluetooth próximos.

## 3.4 Componentes de Aplicação

Componentes de Aplicação também são declarados no arquivo AndroidManifest.xml. Essa sessão fornece uma breve descrição dos componentes declarados e utilizados no projeto. As inter-relações entre esses componentes, isto é, como eles comunicam entre si, está descrita em mais detalhe na seção 3.5.

### 3.4.1 Declaração de Activities

As seguintes Activities foram declaradas como Componentes de Aplicação:

#### MainActivity

Activity inicial, chamada pelo sistema quando a aplicação é lançada. Nela o usuário pode:

- Selecionar uma entre três opções básicas - Pacientes, Sessão Rápida ou Configurações.
- Verificar informações da aplicação, como nome, versão e desenvolvedor.

**PatientsActivity**

Responsável por exibir os pacientes cadastrados na aplicação. Pode separar pacientes com sessões abertas de pacientes sem sessões abertas, ou pacientes antigos (com data de saída anterior ao dia atual) de pacientes atuais. Nela o usuário pode:

- Pesquisar por pacientes.
- Adicionar pacientes.
- Selecionar pacientes cadastrados.

**AddPatientActivity**

Responsável por exibir uma interface de adição de pacientes. Nela o usuário pode:

- Especificar características, primárias e secundárias, do novo paciente.
- Cancelar a adição do novo paciente.
- Adicionar o novo paciente (desde que todas as características primárias sejam informadas).

**SettingsActivity**

Fornece uma interface gráfica para a manipulação de configurações da aplicação. Nela o usuário pode:

- Ver / modificar as configurações da aplicação.

**PatientDetailActivity**

Apresenta as informações de um paciente específico, ou seja, suas características, primárias e secundárias e o seu histórico de sessões. Nela o usuário pode:

- Ver / modificar características do paciente.
- Deletar o paciente do banco de dados.
- Selecionar sessões fechadas.
- Iniciar uma nova sessão, caso não haja uma sessão aberta.
- Verificar o estado de uma sessão aberta.

**SessionDetailActivity**

Apresenta as informações de uma sessão específica, ou seja, suas características, tais como data de início e o seu conjunto de operações. Nela o usuário pode:

- Selecionar operações

Caso a sessão em questão esteja aberta, o usuário também pode:

- Fechar a sessão.
- Deletar operações.
- Verificar as configurações de conexão com estimuladores.
- Adicionar operações, caso haja um estimulador selecionado e canais configurados.

Além disso, caso a sessão em questão seja temporária, o usuário também pode:

- Adicionar a sessão a um paciente existente.
- Adicionar a sessão a um novo paciente.

**StimulatorConfigActivity**

Responsável por fornecer uma interface de configuração de conexões com estimuladores. Nela o usuário pode:

- Selecionar um estimulador para realizar operações.
- Configurar canais para estimulações, ou seja, definir pares canais-músculo que serão utilizados na estimulação.

**OperationDetailActivity**

Responsável por exibir os resultados de uma determinada operação. Nela o usuário pode:

- Ver os resultados da operação em questão.

**StimulationActivity**

Responsável por mostrar os detalhes de uma operação em progresso. Nela o usuário pode:

- Cancelar a operação.
- Ver os resultados parciais (em tempo real) da operação.
- Enviar comandos para o estimulador durante a estimulação.
- Concluir ou descartar a operação, caso esta tenha sido concluída pelo estimulador.

### 3.4.2 Declaração de Serviços

Os seguintes Serviços foram declaradas como Componentes de Aplicação. A implementação destes está descrita na seção 3.10.

#### DiagnosisService

Responsável por realizar operações de diagnóstico, administrando a comunicação com o estimulador. Também deve fornecer detalhes da estimulação em tempo real através de interfaces de *binding*.

#### StimulationActivity

Responsável por realizar operações de terapia, administrando a comunicação com o estimulador. Também deve fornecer detalhes da estimulação em tempo real através de interfaces de *binding*.

### 3.4.3 Declaração de Content Providers

Os seguintes Content Providers foram declaradas como Componentes de Aplicação. A organização de URIs, bem como a arquitetura utilizada para bancos de dados e arquivos estão descritos na seção 3.6.

#### TreatmentsProvider

Responsável por fornecer acesso aos dados relativos a Pacientes, Sessões e Operações (conceitos abstratos descritos na seção 3.1). Interage diretamente com o banco de dados TreatmentsDatabase, descrito mais detalhadamente na seção 3.6.

#### FileProvider

Responsável por fornecer acesso à uma parte da Cache do diretório interno. Este Content Provider (na verdade um File Provider, que é uma subclasse de Content Provider especializada no compartilhamento de arquivos) é usado especificamente para fornecer acesso temporário da aplicação de Câmera do dispositivo a uma determinada parte do diretório interno da aplicação, de modo que esta possa salvar imagens capturadas.

## 3.5 Fluxo da Aplicação

Esta seção descreve como os Componentes de Aplicação descritos na seção 3.4 estão conectados, de modo a gerar as funcionalidades descritas na seção 3.2. A figura 3.3 ilustra essas conexões.

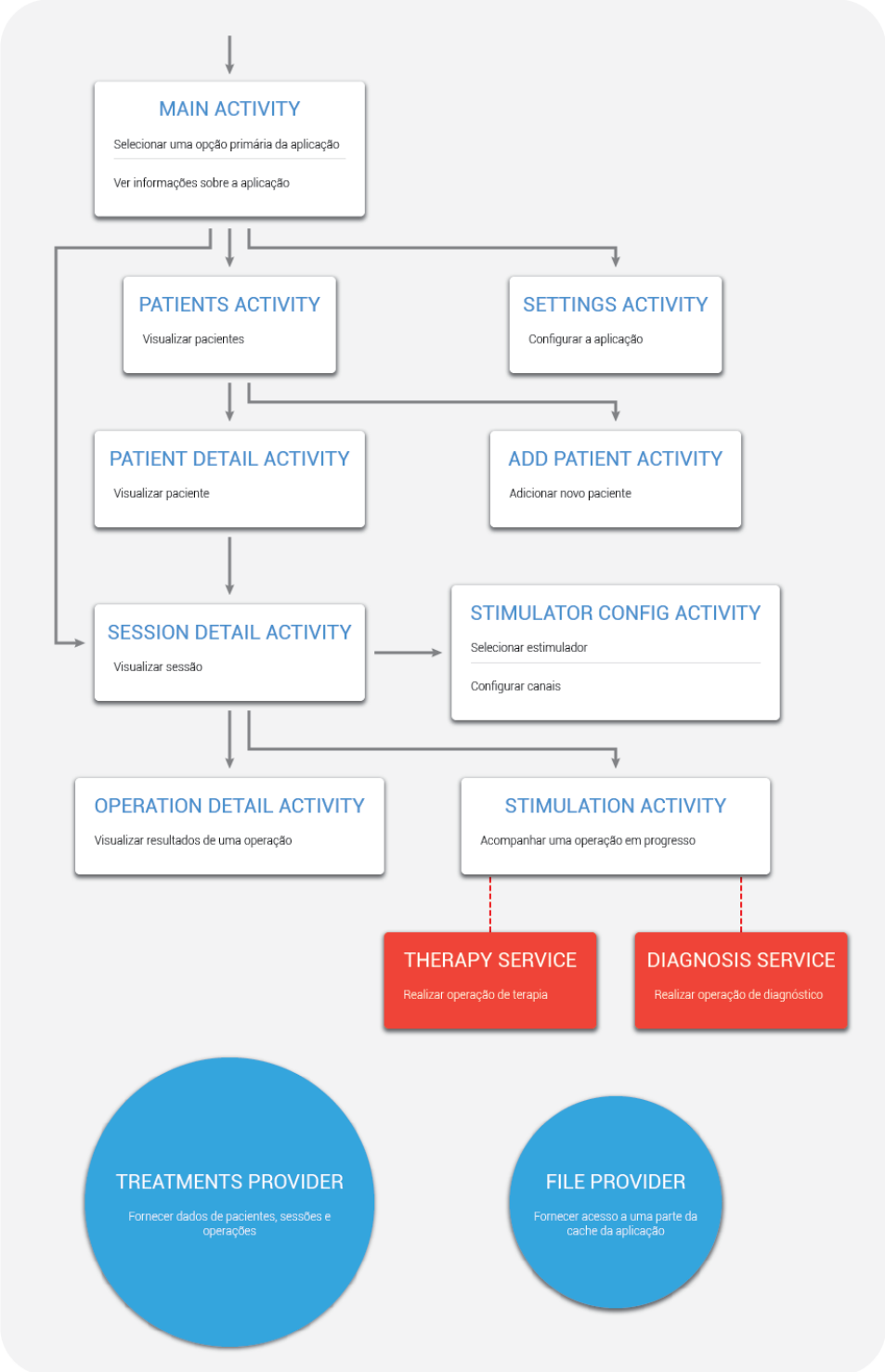



Figura 3.3: Fluxo da Aplicação. Activities são representadas por retângulos brancos, serviços por retângulos vermelhos e content providers por círculos azuis. As conexões indicadas por setas representam o potencial que uma activity possui de iniciar uma outra activity. Conexões pontilhadas representam potenciais ligações de serviços à activities por binding. Na imagem, não estão iustradas conexões com os content providers, porém estes estão acessíveis a todos os outros componentes de aplicação.

Ao ser iniciada pelo usuário, a aplicação lança a *MainActivity*. O usuário pode então escolher uma entre as três opções básicas da aplicação: visualizar pacientes, que iniciará a *PatientsActivity*; iniciar uma sessão rápida, que iniciará a *SessionDetailActivity* com os argumentos necessários para criar uma sessão termporária; ou ajustar as configurações da aplicação, que iniciará a *SettingsActivity*.

Na *PatientsActivity*, o usuário pode visualizar os pacientes cadastrados na aplicação ou adicionar um novo paciente. Caso um paciente seja selecionado, a *PatientDetailActivity* será selecionada passando a URI do paciente em questão como dado. Caso a opção de adicionar um paciente seja selecionada, a *AddPatientActivity* será iniciada.

Na *SettingsActivity* o usuário pode realizar alguns ajustes nas configurações da aplicação. Essa *activity* não inicia nenhuma outra *activity* ou serviço, pode apenas, obviamente, retornar para o componente que a iniciou (no caso do fluxo normal da aplicação, returnar para a *MainActivity*).

Na *PatientDetailActivity*, pode-se vizualizar as acaracterísticas do paciente, bem como seu histórico de sessões. Caso o paciente possua uma sessão aberta, pode-se vizualizar essa sessão, o que iniciaria a *SessionDetailActivity* passando a URI da sessão em questão com dado. Caso contrário, pode-se criar uma nova sessão: isso criaria uma nova sessão no banco de dados e também iniciaria a *SessionDetailActivity*, passando a URI da nova sessão. Em ambos os casos (vizualização de sessão aberta ou criação de nova sessão) será passada um valor extra para a *SessionDetailActivity* para que esta saiba que a sessão em questão está aberta. O usuário também pode vizualizar sessões antigas do paciente, nesse caso a *SessionDetailActivity* também será iniciada, porém sem o valor extra indicando que a sessão está aberta.

Na *AddPatientActivity*, pode-se fazer o cadastro de um novo paciente. Essa *activity* não inicia nenhuma outra *activity* definida da aplicação. No entanto, ela pode iniciar uma *activity* externa de câmera, para que o usuário possa capturar uma foto do novo paciente.

A *SessionPatientActivity* pode ser iniciada em duas situações diferentes: a primeira é a partir da *MainActivity*, o que cria uma nova sessão temporária, passando um valor extra indicando que essa nova sessão é temporária; a segunda é a partir da *PatientDetailActivity*, seja para a vizualização de uma sessão antiga, a criação de uma nova sessão ou a vizualização de uma sessão aberta. Em todos os casos, pode-se vizualizar as operações feitas durante a sessão em questão, o que iniciaria a *OperationDetailActivity*. No caso de sessões abertas, pode-se configurar a conexão com um estimulador (o que iniciaria a *StimulatorConfigActivity*), criar novas operações (o que iniciaria a *StimulationActivity* e um dos dois serviços de estimulação - *TherapyService* ou *DiagnosisService*) ou fechar a sessão. No caso de sessões temporárias, que por definição seriam também abertas, pode-se adicionar a sessão em questão para o histórico de um paciente já cadastrado ou de um novo paciente.

Na *StimulatorConfigActivity* pode-se selecionar um estimulador próximo aos dispositivo desde ambos estejam com o rádio Bluetooth ligado. Além disso, pode-se definir uma lista de pares canal-músculo a serem estimulados. Essa *activity* não inicia nenhuma *activity*, apenas retorna o resultado das configurações de conexão feitas pelo usuário para a *SessionDetailActivity*.

Na *OperationDetailActivity* pode-se visualizar os retultados de uma operação. Essa *activity* não inicia nenhuma outra *activity*.

Na *StimulationActivity* pode-se acompanhar os resultados parciais de uma sessão em tempo real. Isso é possível a partir de uma conexão por *binding* com o serviço de estimulação em progresso, i.e. *TheraphyService* para operações de terapia ou *DiagnosisService* para operações de diagnóstico. Essa *activity* não inicia nenhuma outra *activity*, porém pode parar o serviço de estimulação caso o usuário deseje cancelar a operação em progresso.

Tanto o *TherapyService* quanto o *DiagnosisService* são iniciados pela *SessionDetailActivity* e gerenciam o seu próprio ciclo de vida. Porém, podem ser finalizados pela *StimulationActivity* caso o usuário queira cancelar a operação em progresso.

O *TreatmentsProvider* está acessível a todos os componentes da aplicação, caso seja necessário a verificação de algum dado referente a pacientes, sessões ou operações. Não está acessível, porém, a componentes de aplicações externas.

O *FileProvider* está disponível para componentes externos por meio da concessão de permissões provisórias. Atualmente, só é usado pela aplicação externa de câmera para a captura de imagens de novos pacientes.

## 3.6 Organização de Dados

Os dados relativos a pacientes, sessões e operações foram armazenados em um banco de dados SQLite chamado *TreatmentsDatabase*. Esse armazenamento busca implementar as características dos conceitos definidos na seção 3.1, bem como as suas inter-dependências.

O banco TreatmentsDatabase atualmente (versão 1) é composto exclusivamente de *Tables*, ou Tabelas, que contém a informação propriamente dita, e *Triggers*, ou Gatilhos, entidades que realizam certas operações nas tabelas na ocorrência de eventos observados.

Elementos da aplicação não interagem diretamente com o banco de dados quando precisam ter acesso à informação contida nele. Essa interação é intermediada pelo content provider *Treatments-Provider*.

Além do SQLite, a aplicação também armazena algumas informações em seu diretório interno no sistema de arquivos, como por exemplo imagens e resultados de operações. Alguns desses arquivos são compartilhados com componentes de aplicação externos por meio do content provider *FileProvider*.

Essa seção fornece detalhes acerca das implementações das características referentes à organização de dados adotada pela aplicação.

### 3.6.1 Tabelas SQLite

As tabelas do banco TreatmentsDatabase estão descritas a seguir. Colunas que armazenam valores de tempo são do tipo *TEXT*, no formato padrão do SQLite para *Timestamps*, ou seja: *YYYY-MM-DD HH:MI:SS*.

#### patients

Armazena informações relativas a pacientes. Possui as seguintes colunas:

| Coluna | Tipo | Extra | Descrição |
|---|---|---|---|
| _id | INTEGER | PRIMARY KEY AUTOINCREMENT | Chave primária. Identificador único do paciente |
| patient_name | TEXT | NOT NULL | Nome do paciente |
| patient_gender | TEXT | NOT NULL | Gênero do paciente. Pode assumir 2 valores: “male” ou “female” |
| patient_enter_date | TEXT | DEFAULT CURRENT_TIMESTAMP | Momento do cadastro do paciente |
| patient_exit_date | TEXT | NOT NULL | Momento de saída do paciente |
| patient_birthdate | TEXT | - | Momento de nascimento do paciente |
| patient_phone | TEXT | - | Telefone do paciente |
| patient_email | TEXT | - | Endereço de e-mail do paciente |
| patient_observations | TEXT | - | Observações adicionais sobre o paciente |

Tabela 3.1: Tabela *patients*

#### sessions

Armazena informações relativas a sessões. No caso de uma sessão rápida, o campo *patient_id* é nulo. Possui as seguintes colunas:

| Coluna | Tipo | Extra | Descrição |
|---|---|---|---|
| _id | INTEGER | PRIMARY KEY AUTOINCREMENT | Chave primária. Identificador único da sessão |
| patient_id | INTEGER | REFERENCES patients(_id) | Chave estrangeira. Referencia um paciente da tabela *patients* |
| session_start_date | TEXT | DEFAULT CURRENT_TIMESTAMP | Momento início da sessão |
| session_end_date | TEXT | - | Momento fechamento da sessão |
| session_tag | TEXT | DEFAULT tag_0 | Tag para qualificação da sessão pelo usuário. Pode assumir 4 valores: "tag_0", "tag_1", "tag_2" ou "tag_3" |

Tabela 3.2: Tabela *sessions*

**diagnoses**

Armazena informações relativas a diagnósticos. Possui as seguintes colunas:

| Coluna | Tipo | Extra | Descrição |
|---|---|---|---|
| _id | INTEGER | PRIMARY KEY AUTOINCREMENT | Chave primária. Identificador único do diagnóstico |
| session_id | INTEGER | REFERENCES sessions(_id) | Chave estrangeira. Referencia uma sessão da tabela *sessions* |
| diagnosis_type | TEXT | NOT NULL | Tipo de diagnóstico. Pode assumir 3 valores: "type_chronaxie", "type_rheobase" ou "type_accommodation" |
| diagnosis_date | TEXT | DEFAULT CURRENT_TIMESTAMP | Momento de registro do diagnóstico |

Tabela 3.3: Tabela *diagnoses*

**therapies**

Armazena informações relativas a terapias. Possui as seguintes colunas:

| Coluna | Tipo | Extra | Descrição |
|---|---|---|---|
| _id | INTEGER | PRIMARY KEY AUTOINCREMENT | Chave primária. Identificador único da terapia |
| session_id | INTEGER | REFERENCES sessions(_id) | Chave estrangeira. Referencia uma sessão da tabela *sessions* |
| therapy_type | TEXT | NOT NULL | Tipo de diagnóstico. Pode assumir 2 valores: "type_open_loop" ou "type_closed_loop" |
| therapy_start_date | TEXT | DEFAULT CURRENT_TIMESTAMP | Momento de início da terapia |
| therapy_end_date | TEXT | DEFAULT CURRENT_TIMESTAMP | Momento fechamento da terapia |

Tabela 3.4: Tabela *therapies*

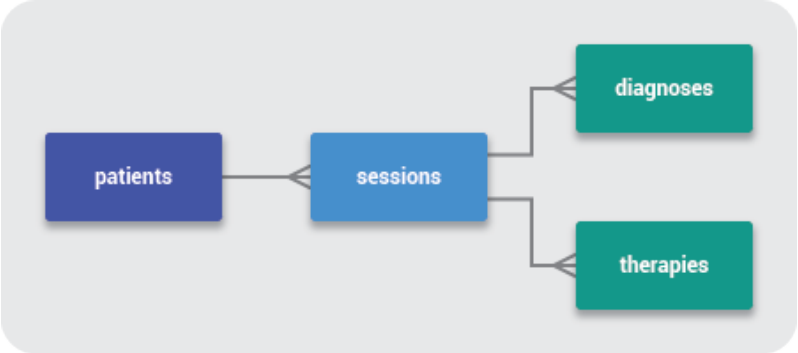



Figura 3.4: Organização do Banco de Dados

### 3.6.2 Gatilhos

Os gatilhos do banco TreatmentsDatabase estão descritas a seguir.

**patient_sessions_delete**

Remove todas as sessões de um paciente quando esse é removido.

**session_diagnoses_delete**

Remove todos os diagnósticos de uma sessão quando essa é removida.

**session_therapies_delete**

Remove todas as terapias de uma sessão quando essa é removida.

### 3.6.3 TreatmentsProvider

O content provider TreatmentsProvider é responsável por fazer uma ponte ente outros componentes da aplicação e o banco de dados TreatmentsDatabase. Essa ponte é feita da seguinte maneira: um componente externo seleciona uma operação básica (busca, inserção, atualização ou remoção) e passa uma URI, que indica uma seleção de dados da tabela. O TreatmentsProvider interpreta essa URI e realiza a operação desejada diretamente no TreatmentsDatabase, retornando os resultados para o componente externo.

O formato das URIs é definido pelo próprio TreatmentsProvider, e as correspondências entre os possíveis formatos e seleções do banco de dados estão descritas a seguir:

| URI | Seleção |
|---|---|
| content://com.huff.tg2.treatments/patients | Seleciona todos os pacientes |
| content://com.huff.tg2.treatments/patients/* | eleciona o paciente com o campo $\_id$ igual a "*", se existente |
| content://com.huff.tg2.treatments/sessions | Seleciona todas as sessões |
| content://com.huff.tg2.treatments/sessions/temp | Seleciona a sessão temporária, se existente |
| content://com.huff.tg2.treatments/sessions/* | Seleciona a sessão com o campo $\_id$ igual a "*", se existente |
| content://com.huff.tg2.treatments/diagnoses | Seleciona todos os diagnósticos |
| content://com.huff.tg2.treatments/diagnoses/* | Seleciona o diagnóstico com o campo $\_id$ igual a "*", se existente |
| content://com.huff.tg2.treatments/therapies | Seleciona todas as terapias |
| content://com.huff.tg2.treatments/therapies/* | Seleciona a terapia com o campo $\_id$ igual a "*", se existente |

Tabela 3.5: Possíveis URIs do TreatmentsProvider

Por exemplo, se a activity OperationDetailActivity iniciar uma operação de busca passando a uri *content://com.huff.tg2.treatments/diagnoses/3535* o TreatmentsProvider deve realizar uma busca no banco TreatmentsDatabase pelo elemento na tabela diagnoses que contenha o campo $\_id$ igual a 3535 e então retornar os resultados para OperationDetailActivity.

### 3.6.4 Arquivos

A aplicação faz uso de seu diretório interno para armazenar algumas informações em forma de arquivos. Os diretórios utilizados para guardar esses arquivos estão descritos a seguir (com caminhos relativos ao diretório interno):

**cache/shared/**

Parte da cache interna disponível para componentes externos através do content provider FileProvider (descrito em mais detalhes no item 3.6.5). Atualmente esse diretório é utilizado para armazenar fotos JPEG temporárias de pacientes capturadas pela aplicação de Câmera do sistema. Essas fotos são utilizadas no momento da inserção de um novo paciente pelo fragmento AddPatientFragment.

**database/**

Armazena o arquivo do banco de dados TreatmentsDatabase.

**files/diagnosis_results**

Armazena resultados de diagnósticos. Esses resultados estão no formato CSV.

**files/patient_images**

Armazena imagens de pacientes. Essas imagens estão no formato JPEG, com dimensões mínimas de 400x800 pixels.

**files/patient_thumbnails**

Armazena miniaturas de imagens de pacientes. Essas miniaturas estão no formato JPEG, com dimensões de 160x160 pixels. O uso de miniaturas ao invés das imagens originais busca melhor performance em algumas situações.

**files/therapy_results**

Armazena resultados de terapias. Esses resultados estão no formato CSV.

### 3.6.5 FileProvider

O content provider FileProvider é responsável por fornecer acesso temporário à arquivos no diretório interno *cache/shared/* para componentes de aplicação externos. A comunicação entre o FileProvider e componentes externos é feitas por URIs, do mesmo modo que o content provider TreatmentsProvider, porém ao invés de retornar dados relacionais, será retornado streams de arquivos.

As correspondências entre as possíveis URIs e os arquivos compartilhados estão descritos a seguir:

| URI | Seleção |
|---|---|
| content://com.huff.tg2.sharedcache/ sharedcache/temp_patient_image.jpeg | Arquivo de imagem temporária de paciente. Acessado pela aplicação de Câmera do sistema durante a adição de um novo paciente. |

Tabela 3.6: Possíveis URIs do FileProvider

## 3.7 Organização do Código-Fonte

O código fonte está agrupado em pacotes java, listados a seguir:

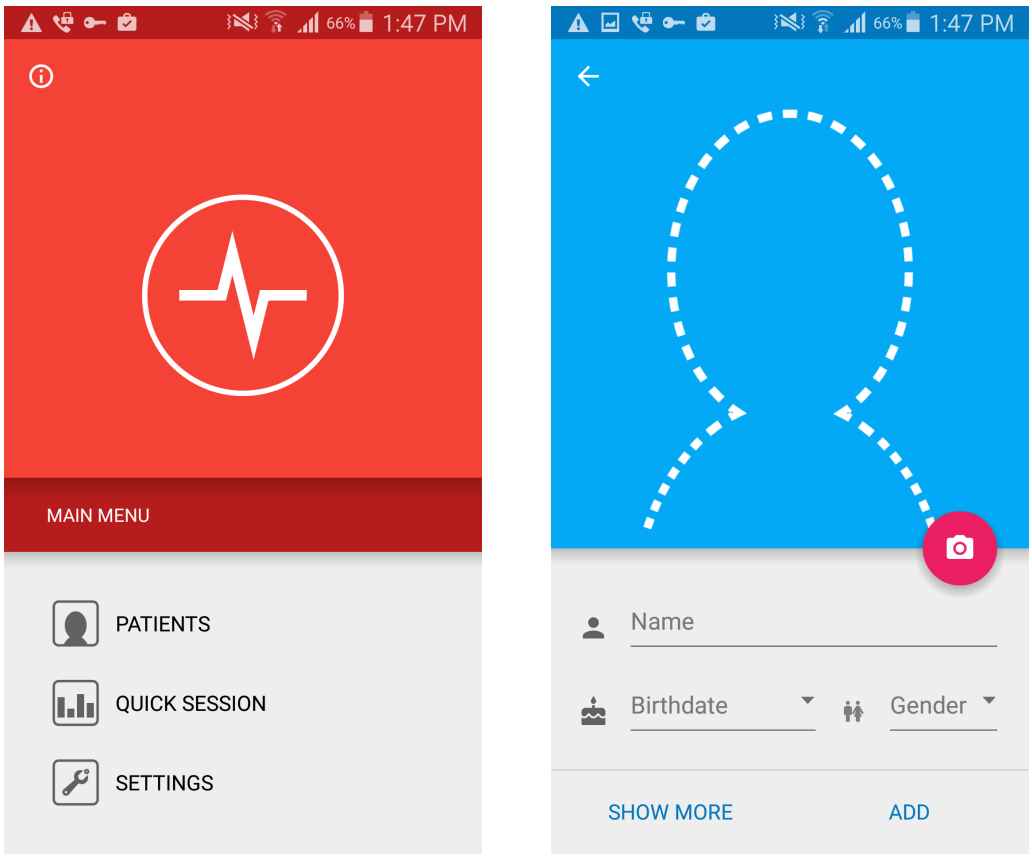
| Pacote | Descrição |
|---|---|
| com.huff.tg2.activities | Contém implementações de Activities |
| com.huff.tg2.files | Contém definições e métodos para a manipulação dos arquivos da aplicação |
| com.huff.tg2.fragments | Contém implementações de Fragmentos |
| com.huff.tg2.models | Contém classes que implementam conceitos abstratos utilizados pela aplicação |
| com.huff.tg2.models.protocol | Contém definições e métodos que implementam o protocolo de comunicação com o estimulador |
| com.huff.tg2.provider | Contém definições do banco de dados TreatmentsDatabase e do content provider TreatmentsProvider |
| com.huff.tg2.services | Contém implementações de Serviços |
| com.huff.tg2.widgets | Contém componentes gráficos criados dentro da aplicação, tais como botões animados, etc |
| com.huff.utils | Contém classes auxiliares que fornecem métodos e constantes para realização de tarefas miscelâneas, não só da aplicação desenvolvida, mas utilizáveis em qualquer contexto |

Tabela 3.7: Pacotes Java da aplicação

## 3.8 Interface Gráfica

Durante o desenvolvimento, foram implementados alguns elementos do Design Material, tais como uso das cores padronizadas nas normas, iconografia, elevação, animações e uso de APIs novas, como por exemplo a classe *RecyclerView* da biblioteca *com.android.support:recyclerview-v7:22.0.0* para a implementação de listas.

Um dos desafios foi implementar alguns desses elementos em versões mais antigas do sistema operacional, já que a versão mínima definida no projeto é a API 14, fazendo com que muitas das




(a) MainActivity (b) AddPatientActivity

Figura 3.5: Desenvolvimento da Interface Gráfica

APIs que facilitariam as implementações estivessem indisponíveis.

No início do desenvolvimento, havia uma preocupação muito grande em criar interfaces gráficas elaboradas e condizentes com as normas do Design Material. Posteriormente, por questões de conveniência, a prioridade passou a ser a funcionalidade da aplicação. No entanto, as primeiras Activities geradas tiveram interfaces mais elaboradas, conforme o ilustrado na figura 3.5.

Os resultados gerados estão descritos em mais detalhes no Capítulo 4.

## 3.9 Protocolo de Comunicação

Atualmente, a comunicação entre a aplicação e o estimulador é feita via Bluetooth Clássico. A aplicação deve estabelecer conexões com dispositivos bluetooth com o identificador CoD (*Class of Device*) definido atualmente como *0x80500* (dispositivo da classe maior *Saúde* e da classe menor *Indefinido*, que oferece serviços de *Informação*), utilizando um identificador de serviço UUID também definido no projeto do estimulador igual à *71c28172-ece1-4df5-b6e6-1e507649c813*.

O protocolo utilizado foi desenvolvido durante o projeto e está dividido em 2 camadas: a *Camada de Mensagem* e a *Camada de Dados*, descritas a nessa seção. O processamento da informação


Figura 3.6: Protocolo de Comunicação

está ilustrado na figura 3.6.

## Camada de Mensagem

O objetivo principal da Camada de Mensagem é extrair o tipo da mensagem enviada. Existem 3 tipos de mensagem:

- *Comando* - Mensagem enviada pela aplicação para o estimulador, com o objetivo de se obter uma *Resposta*. Enviar um Comando implica necessariamente em receber uma Resposta.
- *Resposta* - Mensagem enviada pelo estimulador para a aplicação em resposta a um *Comando*. Uma Resposta pode ser enviada se e somente se um Comando foi anteriormente recebido.
- *Mensagem de Stream* - Mensagem enviada pelo estimulador para a aplicação sem a necessidade de que um Comando tenha sido recebido. Esse tipo de mensagem é usado durante Operações, nas quais o estimulador envia dados periodicamente para a aplicação.

A Camada de Mensagem também é responsável por verificar a validade de pacotes por meio de

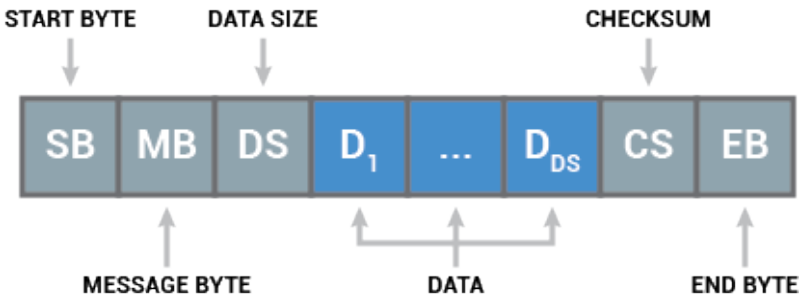



Figura 3.7: Pacote da Camada de Mensagem

redundâncias.

A estrutura de um pacote na Camada de Mensagem é ilustrada na figura 3.7, e possui os seguintes campos:

### START BYTE

Byte único que identifica o tipo da mensagem. Pode assumir 3 valores: 0x23 para Comandos, 0x24 para Respostas e 0x25 para Mensagens de Stream.

### MESSAGE BYTE

Byte único, a ser tratado pela Camada de Dados.

### DATA SIZE

Byte único, identificando o tamanho em bytes do campo *DATA*.

### DATA

Conjunto de bytes, com o tamanho identificado pelo campo *DATA SIZE*, a ser tratado pela Camada de Dados.

### CHECKSUM

Byte único, indicando o valor da soma dos campos *MESSAGE BYTE* , *DATA SIZE* e *DATA* módulo 256. Redundância criada para aumentar a confiabilidade.

### END BYTE

Último byte do pacote, sempre igual a 0x3B.

## Camada de Dados

O objetivo principal da Camada de Dados é extrair a informação contida no pacote através da análise dos campos *MESSAGE BYTE* e *DATA* abstraídos da camada de Mensagem. Essa informação é obtida através da determinação do subtipo da mensagem (usado em conjunto com o tipo), determinado pelo campo *MESSAGE BYTE*, e dos seus argumentos, encapsulados no campo *DATA*. Os possíveis subtipos estão listados a seguir, agrupados em tipos:

### Comandos

#### Start Diagnosis Command (subtipo 0x20)

Inicia uma operação de diagnóstico. Deve receber uma resposta *Start Diagnosis Response*.

Deve passar os seguintes argumentos no campo *DATA*:

```
DATA = (DIAGNOSIS TYPE) : (CHANNEL NUMBERS)[]
```

Onde:

*DIAGNOSIS TYPE* é o tipo de diagnóstico, podendo assumir 3 valores: 0x01 (Cronaxia), 0x02 (Reobase) ou 0x03 (Acomodação).

*CHANNEL NUMBERS* é uma sequência de bytes contendo os números dos canais a serem utilizados durante o diagnóstico.

#### Start Therapy (subtipo 0x30)

Inicia uma operação de terapia. Deve receber uma resposta *Start Therapy Response*.

Deve passar os seguintes argumentos no campo *DATA*:

```
DATA = (THERAPY TYPE) :

(CHANNEL 1 PARAMS)[32] :  (CHANNEL 1 NUMBERS) :

(CHANNEL 2 PARAMS)[32] :  (CHANNEL 2 NUMBERS) :

...

(CHANNEL N PARAMS)[32] :  (CHANNEL N NUMBERS)
```

Onde:

*THERAPY TYPE* é o tipo de terapia, podendo assumir 2 valores: 0x01 (Malha Aberta) ou 0x02 (Malha Fechada).

*CHANNEL X PARAM*S são parâmetros de terapia de um canal a ser estimulado, onde *X* é a indexação desse canal no corpo da mensagem. São 8 floats representando, respectivamente, frequência (1 - 100 Hz), amplitude (0 - 100 mA), largura de pulso (0 - 400µs), duração total (0 - 3600 segundos), tempo de repouso (0 - 300 segundos), temp de estimulação (0 - 60 segundos), rampa de subida (0 - 0.5) e rampa de descida (0 - 0.5).

*CHANNEL X NUMBER* indica o número de um canal sendo estimulado, onde *X* é a indexação desse canal no corpo da mensagem.

#### Finish Operation (subtipo 0xF0)

Finaliza uma operação. Deve receber uma resposta *Finish Operation Response*. Não possui argumentos.

### Respostas

#### Start Diagnosis Response (subtipo 0x20)

Resposta ao comando *Start Diagnosis Command*. Indica se o diagnóstico especificado no comando pôde ser iniciado ou não, indicando os motivos em caso negativo.

Deve passar os seguintes argumentos no campo *DATA*:

```
DATA = (RESULT) : (RESULT ARGS)[]
```

Onde:

*RESULT* indica o resultado do pedido realizado pelo comando, podendo assumir 4 valores: 0x00 (Ok), indicando que a operação foi iniciada, e o estimulador deve começar a enviar Mensagens Stream do tipo diagnóstico; 0x01 (Unsupported Operation), indicando que o tipo de diagnóstico não é suportado; 0x02 (Unsupported Channels), indicando que o estimulador não possui alguns dos canais passados no comando; ou 0x03 (Bad Channels), indicando que alguns dos canais passados no comando não estão corretamente conectados.

RESULT ARGS contém informações complementares para descrever resultados. O seu formado depende do resultado descrito no campo *RESULT*. Para resultados iguais a 0x00 (Ok) ou 0x01 (Unsupported Operation), esse campo é inexistente; para resultados iguais a 0x02 (Unsupported Channels), será uma sequência de bytes indicando o número dos canais não suportados pelo estimulador; para resultados iguais a 0x03 (Bad Channels), será uma sequência de bytes indicando o número dos canais não conectados corretamente.

**Start Therapy Response (subtipo 0x30)**

Resposta ao comando *Start Therapy Command*. Indica se a terapia especificada no comando pôde ser iniciada ou não, indicando os motivos em caso negativo.

Deve passar os seguintes argumentos no campo *DATA*:

```
DATA = (RESULT) : (RESULT ARGS)[]
```

Onde:

*RESULT* indica o resultado do pedido realizado pelo comando, podendo assumir 4 valores: 0x00 (Ok), indicando que a operação foi iniciada, e o estimulador deve começar a enviar Mensagens Stream do tipo terapia; 0x01 (Unsupported Operation), indicando que o tipo de diagnóstico não é suportado; 0x02 (Unsupported Channels), indicando que o estimulador não possui alguns dos canais passados no comando; ou 0x03 (Bad Channels), indicando que alguns dos canais passados no comando não estão corretamente conectados.

*RESULT ARGS* contém informações complementares para descrever resultados. O seu formado depende do resultado descrito no campo RESULT. Para resultados iguais a 0x00 (Ok) ou 0x01 (Unsupported Operation), esse campo é inexistente; para resultados iguais a 0x02 (Unsupported Channels), será uma sequência de bytes indicando o número dos canais não suportados pelo estimulador; para resultados iguais a 0x03 (Bad Channels), será uma sequência de bytes indicando o número dos canais não conectados corretamente.

**Finish Operation Response (subtipo 0xF0)**

Resposta ao comando *Finish Operation Command*. Indica o resultado da tentativa de finalização de uma operação.

Deve passar os seguintes argumentos no campo *DATA*:

```
DATA = (RESULT)
```

Onde:

*RESULT* indica o resultado do pedido realizado pelo comando, podendo assumir 3 valores: 0x00 (Ok), indicando que a operação em progresso foi finalizada; 0x01 (No Operation), indicando que não havia operação em progresso; 0x02 (Error), indicando que a operação em progresso não pôde ser finalizada.

### Mensagem de Stream

#### Diagnosis Stream Message (subtipo 0x10)

Enviada durante operações de diagnóstico, depois que uma resposta *Start Diagnosis Response* com o resultado (0x00 - Ok) foi enviada. Encapsula dados e informações relativos à operação e aos canais estimulados.

Deve passar os seguintes argumentos no campo *DATA*:

```
DATA = (DIAGNOSIS TYPE) : (STATUS) :

(CHANNEL 1 NUMBER) : (CHANNEL 1 STATUS) : (CHANNEL 1 DATA)[4] :

(CHANNEL 2 NUMBER) : (CHANNEL 2 STATUS) : (CHANNEL 2 DATA)[4] :

...

(CHANNEL N NUMBER) : (CHANNEL N STATUS) : (CHANNEL N DATA)[4]
```

Onde:

*DIAGNOSIS TYPE* o tipo de diagnóstico em progresso. Pode assumir os mesmos valores do campo *DIAGNOSIS TYPE* do comando *Start Diagnosis Command*.

*STATUS* contém o estado do diagnóstico. Pode assumir 3 valores: 0x00 (Stimulating), indicando que a operação está em progresso; 0x01 (Finished), indicando que a operação foi finalizada; ou 0x02 (Error), indicando que houve um erro durante a operação.

*CHANNEL X NUMBER* indica o número de um canal sendo estimulado, onde $X$ é a indexação desse canal no corpo da mensagem.

*CHANNEL X STATUS* é um byte único que indica o estado do canal de indexação $X$. Pode assumir 3 valores: 0x00 (Stimulating), indicando que o canal está sendo estimulado; 0x01 (Finished), indicando que a estimulação do canal foi finalizada; ou 0x02 (Not Stimulated), indicando que o canal não pode ser estimulado por algum motivo.

*CHANNEL X DATA* é um float que quantifica a estimulação do canal de indexação $X$ no momento do envio da mensagem.

#### Therapy Stream Message (subtipo 0x20)

Enviada durante operações de terapia, depois que uma resposta *Start Therapy Response* com o resultado (0x00 - Ok) foi enviada. Encapsula dados e informações relativos à operação e aos canais estimulados.

Deve passar os seguintes argumentos no campo *DATA*:

```
DATA = (THERAPY TYPE) : (STATUS) :

(CHANNEL 1 NUMBER) : (CHANNEL 1 STATUS) : (CHANNEL 1 DATA)[4] :

(CHANNEL 2 NUMBER) : (CHANNEL 2 STATUS) : (CHANNEL 2 DATA)[4] :

...

(CHANNEL N NUMBER) : (CHANNEL N STATUS) : (CHANNEL N DATA)[4]
```

Onde:

*THERAPY TYPE* o tipo de terapia em progresso. Pode assumir os mesmos valores do campo *THERAPY TYPE* do comando *Start Therapy Command*.

*STATUS* contém o estado do diagnóstico. Pode assumir 3 valores: 0x00 (Stimulating), indicando que a operação está em progresso; 0x01 (Finished), indicando que a operação foi finalizada; ou 0x02 (Error), indicando que houve um erro durante a operação.

*CHANNEL X NUMBER* indica o número de um canal sendo estimulado, onde $X$ é a indexação desse canal no corpo da mensagem.

*CHANNEL X STATUS* é um byte único que indica o estado do canal de indexação $X$. Pode assumir 3 valores: 0x00 (Stimulating), indicando que o canal está sendo estimulado; 0x01 (Finished), indicando que a estimulação do canal foi finalizada; ou 0x02 (Not Stimulated), indicando que o canal não pode ser estimulado por algum motivo.

*CHANNEL X DATA* é um float que quantifica a estimulação do canal de indexação $X$ no momento do envio da mensagem.

## 3.10 Serviços de Estimulação

Atualmente a aplicação declara 2 serviços: *DiagnosisService* e *TherapyService*. Esses serviços são responsáveis por abrir canais de comunicação com estimuladores e utilizar o protocolo de comunicação descrito na seção 3.9 para realizar operações em pacientes. Durante as operações, os dados dos canais enviados por meio de Mensagens de Stream são coletados e posteriormente armazenados pela aplicação, seguindo a organização descrita na seção 3.6.

Como ambos os serviços possuem muitos pontos em comum, a sua implementação em Java foi feita através da extensão de uma classe abstrata criada dentro da aplicação chamada *StimulationService*. Essa seção descreve a implementação dessa classe abstrata, bem como as particularidades inseridas pelas extensões *DiagnosisService* e *TherapyService*.

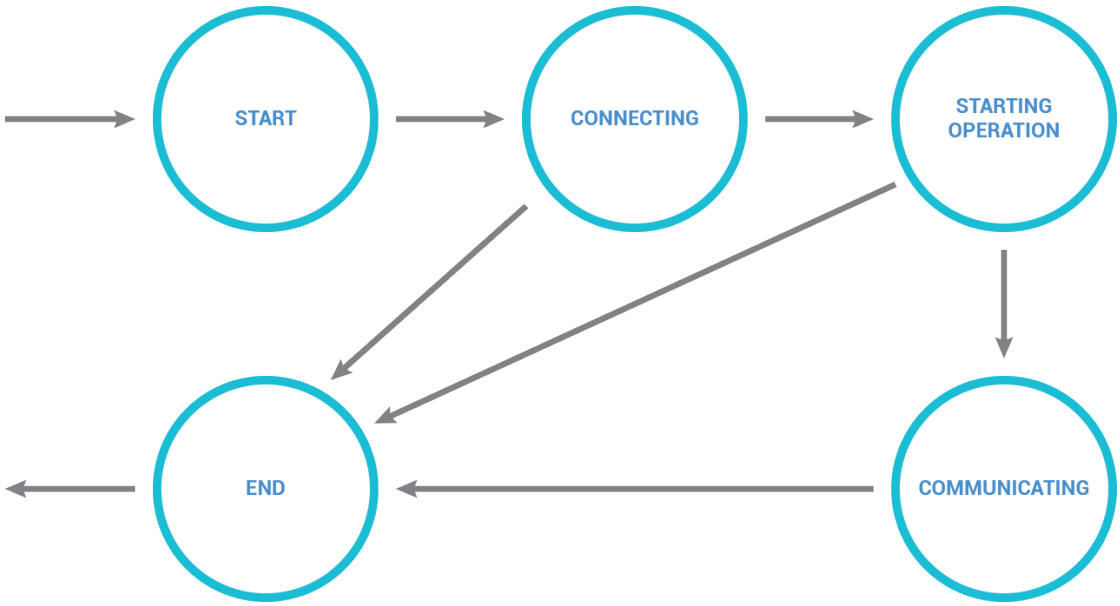



Figura 3.8: Máquina de Estados do Serviço

### 3.10.1 Argumentos

O serviço *StimulationService* precisa receber 2 informações para que seja corretamente executado. Essas informações estão encapsuladas em objetos da classe *StimulationConnection*, do pacote *com.huff.tg2.models* da aplicação, e estão descritas a seguir:

#### Dados do estimulador

Atualmente representados por objetos da classe *BluetoothDevice*, da API de Bluetooth do sistema. Usado para iniciar uma conexão com o estimulador.

#### Dados de canais

Representados por objetos da classe *Channels*, do pacote *com.huff.tg2.models* da aplicação. Contém um grupo de pares canal-músculo que indicam quais canais serão estimulados e quais músculos estão conectados à esses canais.

### 3.10.2 Máquina de Estados

O ciclo de vida do serviço StimulationService pode ser representado por uma máquina de estados (ilustrada na figura 3.8). Os seus estados estão descritos a seguir.

#### Estado START

Estado inicial no qual o serviço entra ao ser iniciado.

Pode apenas transitar para o estado *CONNECTING*.

#### Estado CONNECTING

Estado no qual o serviço utiliza os dados do estimulador para estabelecer uma conexão com o estimulador.

Pode transitar para o estado *END*, caso a conexão seja cancelada pelo usuário ou alguma falha ocorreu; ou para o estado *STARTING OPERATION*, caso a conexão foi estabelecida com sucesso.

#### Estado STARTING OPERATION

Estado no qual o serviço envia um comando ao estimulador com o objetivo de iniciar uma operação. Subclasses devem especificar detalhes dessa operação através da definição do comando que será enviado. A extensão *DiagnosisService* envia um comando do tipo *Start Diagnosis Command*, enquanto a extensão *TherapyService* envia um comando do tipo *Start Therapy Command*, passando parâmetros de terapia passados na criação do serviço.

Pode transitar para o estado *END*, caso a requisição de conexão seja cancelada pelo usuário, ou caso o estimulador retorne algum resultado negativo, ou caso alguma falha ocorra; ou para o estado *COMMUNICATING*, caso a operação seja iniciada com sucesso.

#### Estado COMMUNICATING

Estado no qual a operação está em progresso, e o serviço recebe informações por meio de Mensagens de Stream. O processamento dessa informação deve ser implementado por subclasses. A extensão *DiagnosisService* coleta os dados dos canais a cada instante, sobre-escrevendo dados antigos por dados novos, enquanto a e extensão *TherapyService*, apesar de também coletar os dados a cada instante, armazena todos esses dados, junto ao instante em que foram coletados, de modo a armazenar a evolução de cada canal no tempo.

Transita para o estado *END*, caso a conexão seja cancelada pelo usuário, ou caso o estimulador indique que a operação deve ser finalizada, ou caso alguma falha ocorra.

#### Estado END

Estado final no qual o serviço é finalizado.

### 3.10.3 Eventos

"Eventos" podem ser definidos como eventos externos ao serviço que podem resultar em transições de estados, e estão listados a seguir:

**Evento SERVICE STARTED**

Ocorre quando o serviço é iniciado.

Caso o serviço esteja no estado *START*, irá transitar para o estado *CONNECTING*.

**Evento USER CANCEL**

Ocorre quando o usuário deseja cancelar a operação do serviço.

Caso o serviço esteja no estado *CONNECTING*, irá transitar para o estado *END*.

Caso o serviço esteja no estado *STARTING OPERATION*, irá transitar para o estado *END*.

Caso o serviço esteja no estado *COMMUNICATING*, irá transitar para o estado *END*.

**Evento CONNECTION FAILED**

Ocorre quando a tentativa de conexão com o estimulador falhou por algum motivo interno.

Caso o serviço esteja no estado *CONNECTING*, irá transitar para o estado *END*.

**Evento CONNECTION SUCCESS**

Ocorre quando uma conexão com o estimulador é estabelecida com sucesso.

Caso o serviço esteja no estado *CONNECTING*, irá transitar para o estado *STARTING OPERATION*.

**Evento READ WRITE EXCEPTION**

Ocorre quando há problema interno de leitura ou escrita no canal de comunicação estabelecido com o estimulador.

Caso o serviço esteja no estado *COMMUNICATING*, irá transitar para o estado *END*.

**Evento RESPONSE RECEIVED**

Ocorre quando a aplicação recebe uma mensagem do tipo Resposta do estimulador. Sub-classes devem implementar o processamento dessa resposta, bem como se ela deve resultar em transições, e quais as seriam essas transições.

Na extensão *DiagnosisService*, caso o serviço esteja no estado S*TARTING OPERATION*, irá transitar para o estado *COMMUNICATING*, se e somente se a resposta recebida for do tipo *Start Diagnosis Response* com um resultado igual a 0x00 (OK).

Na extensão *TherapyService*, caso o serviço esteja no estado *STARTING OPERATION*, irá

transitar para o estado *COMMUNICATING*, se e somente se a resposta recebida for do tipo *Start Therapy Response* com um resultado igual a 0x00 (OK).

#### Evento STREAM MESSAGE RECEIVED

Ocorre quando a aplicação recebe uma mensagem do tipo Mensagem de Stream do estimulador. Sub-classes devem implementar o processamento dessa mensagem de stream, bem como se ela deve resultar em transições, e quais as seriam essas transições.

Na extensão *DiagnosisService*, caso o serviço esteja no estado *COMMUNICATING*, irá transitar para o estado *END*, se e somente se a mensagem de stream recebida for do tipo *Diagnosis Stream Message* com um status igual a 0x01 (Finished) ou 0x02 (Error).

Na extensão *TherapyService*, caso o serviço esteja no estado *COMMUNICATING*, irá transitar para o estado *END*, se e somente se a mensagem de stream recebida for do tipo *Therapy Stream Message* com um status igual a 0x01 (Finished) ou 0x02 (Error).

### 3.10.4 Threads

A execução do serviço é feita em 3 threads diferentes: *Thread Principal*, *Thread de Conexão* e *Thread de Comunicação*, descritas a seguir.

#### Thread Principal

Como o nome já diz, é a thread principal do serviço, na qual as operações principais, tais como mudanças de estados e manipulação de observadores, são feitas.

Responsável por detectar o evento *SERVICE STARTED*.

#### Thread de Conexão

Existente durante o estado *CONNECTING*. Responsável por estabelecer uma conexão com o estimulador.

Responsável por detectar os eventos *CONNECTION FAILED* e *CONNECTION SUCCESS*.

#### Thread de Comunicação

Existente durante os estados *STARTING OPERATION* e *COMMUNICATING*. Responsável por ler e escrever no canal de comunicação aberto pela *Thread de Conexão*, fazendo a tradução entre bytes e a abstração de mensagens. A leitura é feita armazenando os bytes lidos no canal em *buffers* e os convertendo em Respostas ou Mensagens de Stream quando possível. A escrita é feita convertendo Comandos em bytes e os escrevendo no canal.

Responsável por detectar os eventos *READ WRITE EXCEPTION*, *RESPONSE RECEIVED* e *STREAM MESSAGE RECEIVED*.

### 3.10.5 Interface com o Usuário

Serviços não interagem diretamente com o usuário. Para que isso seja feito, é necessário que seja implementada uma interface de binding. Essa interface possibilitará a interação do serviço com uma activity. Por sua vez, essa activity exibirá visualmente dados do serviço para o usuário.

O serviço *StimulationService* é conectado a activity *StimulationActivity* por binding, de modo que essa possa exibir dados da operação e dos seus canais para o usuário durante a operação.

Isso foi implementado da seguinte maneira: o serviço declara uma interface chamada *Observer*, que possui *callbacks* que podem ser chamados pelo serviço, e também disponibiliza métodos na classe de *binding* que fazem acesso direto a informações como estado atual e dados dos canais. Quando a *StimulationActivity* se conecta ao serviço por binding, essa recebe uma instância da classe de *binding*, ou seja, ganha acesso a propriedades do serviço, e adiciona um objeto do tipo Observer em uma lista de *Observers* do serviço, que tem seus callbacks chamados pelo próprio serviço durante transições de estado, recebimento de mensagens do estimulador ou outros eventos. Tendo acesso a essas informações, a *StimulationActivity* pode então exibí-las para o usuário através de uma interface gráfica.

# Capítulo 4

# Resultados e Discussão

*Ao final do desenvolvimento, foram produzidas cerca de 15 mil linhas de código gerado manualmente, em Java e XML. Nesse capítulo serão descritos os resultados obtidos com o desenvolvimento da aplicação, sob os pontos de vista de interface com o usuário e o armazenamento dos dados.*

## 4.1 Interface com o Usuário

Essa seção busca avaliar o nível de interatividade da aplicação desenvolvida para com usuários potenciais. Para isso, inicialmente serão apresentados e descritos os resultados finais das telas referentes as Activities da aplicação. Posteriormente serão relatado os resultados do uso da aplicação por usuários que nunca a haviam utilizado.

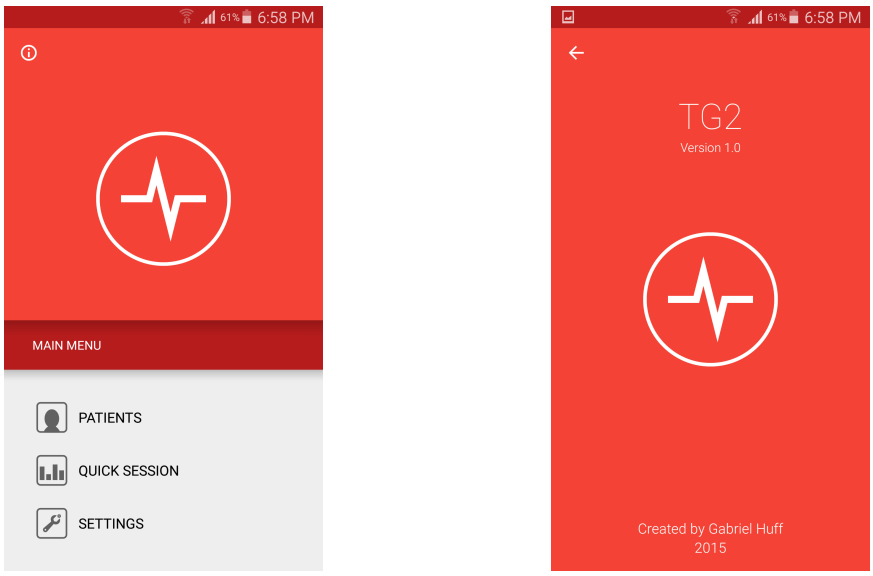
### 4.1.1 Telas da Aplicação

Este item descreve as telas da aplicação, ou seja, os possíveis estados da interface gráfica das Activities desenvolvidas do ponto de vista do usuário.

#### MainActivity

A MainActivity possui dois modos: um que apresenta um menu principal, no qual o usuário pode selecionar uma opção básica disponível (Pacientes, Sessão Rápida ou Configurações) e outro que fornece detalhes gerais da aplicação. Esses modos estão ilustrados na figura 4.1.

No modo de menu principal, as opções são representadas por botões com ícones animados. Quando selecionada, uma opção sofrerá uma variação de cor para indicar a seleção, e seu ícone será animado momentaneamente, além disso esta sofrerá um efeito de *Ripple* (padrão definido no Design Material que consiste no crescimento de um círculo transparente levemente escurecido originado do local da seleção até o preenchimento total da opção, para reforçar a seleção). O comportamento dos ícones está ilustrado na figura 4.2. O usuário poderá transitar para o modo




(a) Menu principal

(b) Detalhes da aplicação

Figura 4.1: Telas da MainActivity

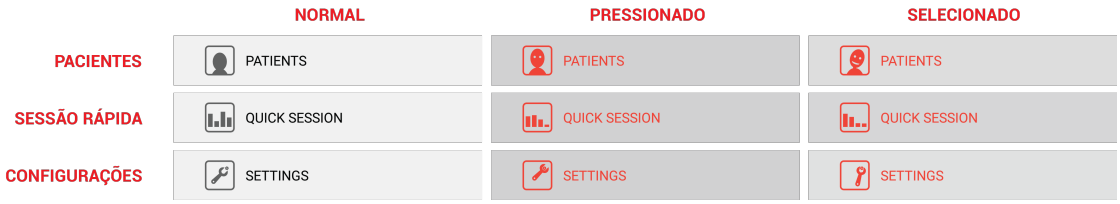


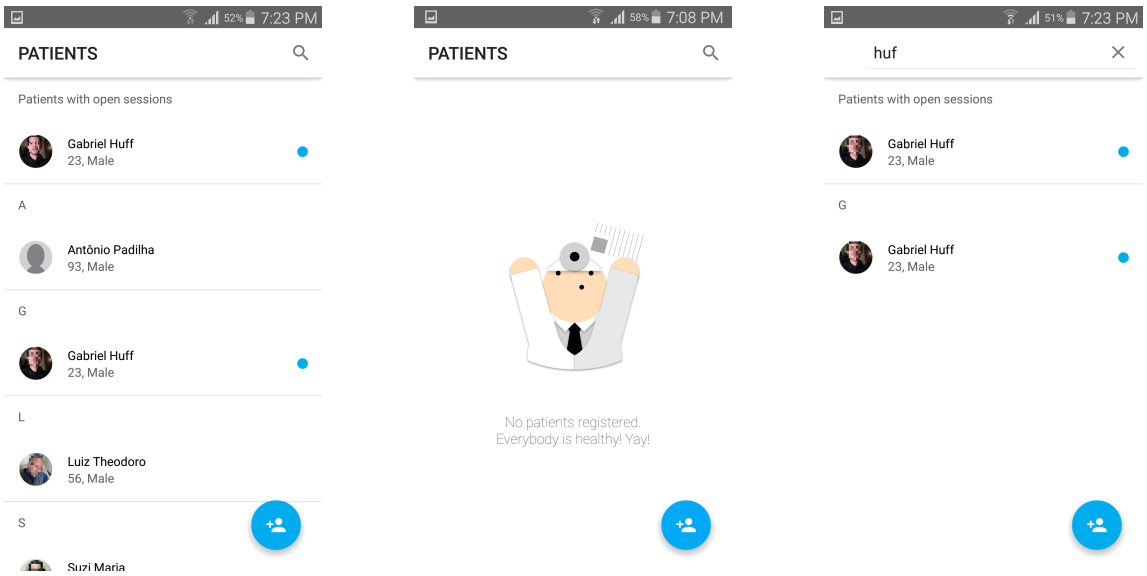

Figura 4.2: Ícones do menu principal em seus 3 estados possíveis: normal, pressionado e selecionado.

de visualização de detalhes da aplicação de dois modos: clicando no ícone de informação no canto superior esquerdo da tela, que se transformará em uma seta para retorno ao menu principal, ou arrastando o dedo de cima para baixo a partir do ícone da aplicação, como se estivesse fechando uma janela.

No modo de visualização de detalhes da aplicação, o usuário terá acesso a informações básicas da aplicação, tais como nome, versão e desenvolvedor. Para retornar ao menu principal, há 3 opções: selecionar o ícone de retorno no canto superior esquerdo, que se transformará novamente no ícone de informação; arrastar a tela de baixo para cima, como se o usuário estivesse abrindo uma janela, ou pressionar o botão *Back* do Android.

### PatientsActivity

A PatientsActivity possui uma lista de pacientes cadastrados na aplicação. Cada elemento dessa lista possui um ícone com uma miniatura da foto do paciente (ou uma foto padrão para pacientes sem foto cadastrada), o nome do paciente, detalhes (nome e idade) e um indicador representado




(a) (b) (c)

Figura 4.3: Telas da PatientsActivity

por um pequeno círculo azul na parte direita indicando que o paciente possui uma sessão aberta. Pacientes antigos possuem uma transparência de 60%. A lista é ordenada em ordem alfabética. Além disso, pacientes com sessões abertas reaparecem no início da lista.

Quando nenhum paciente está cadastrado, uma imagem minimalista de um médico feliz será exibida, e um texto indicando que nenhum paciente está cadastrado será mostrado.

Também possui uma Toolbar para pesquisa de pacientes por nome. O texto de entrada será usado como filtro para a pesquisa, que acontecerá simultaneamente com a inserção de cada caracter do filtro. No caso de nenhum resultado encontrado, será mostrado um texto indicando tal fato.

Por último possui um FAB (*Floating Action Button*, botão circular flutuante, definido nas especificações do Design Material) de cor viva contrastante utilizado para adicionar um novo paciente. Ao selecionado, o FAB sofrerá uma leve variação de elevação, além de um efeito *Ripple*.

A configuração descrita está descrita na figura 4.3.

### AddPatientActivity

A *AddPatientActivity* apresenta detalhes de um paciente a ser adicionado. Possui uma *Toolbar* expandida contendo a foto temporária do novo paciente, ou uma foto padrão quando nenhuma foto foi capturada. Possui um FAB para a captura de uma nova foto temporária. Ao selecionar o FAB, a aplicação de câmera do sistema será chamada para a captura de uma nova foto temporária.

Abaixo da *Toolbar*, estão os campos de entrada de características do paciente, compostos por ícones de acordo com as regras de iconografia do Design Material e widgets de entrada (*EditTexts* para textos personalizados, como por exemplo Nome e Observações, *DatePicker* para a Data de Nascimento e *Spinners* para opções pré-estabelecidas, como por exemplo Gênero).

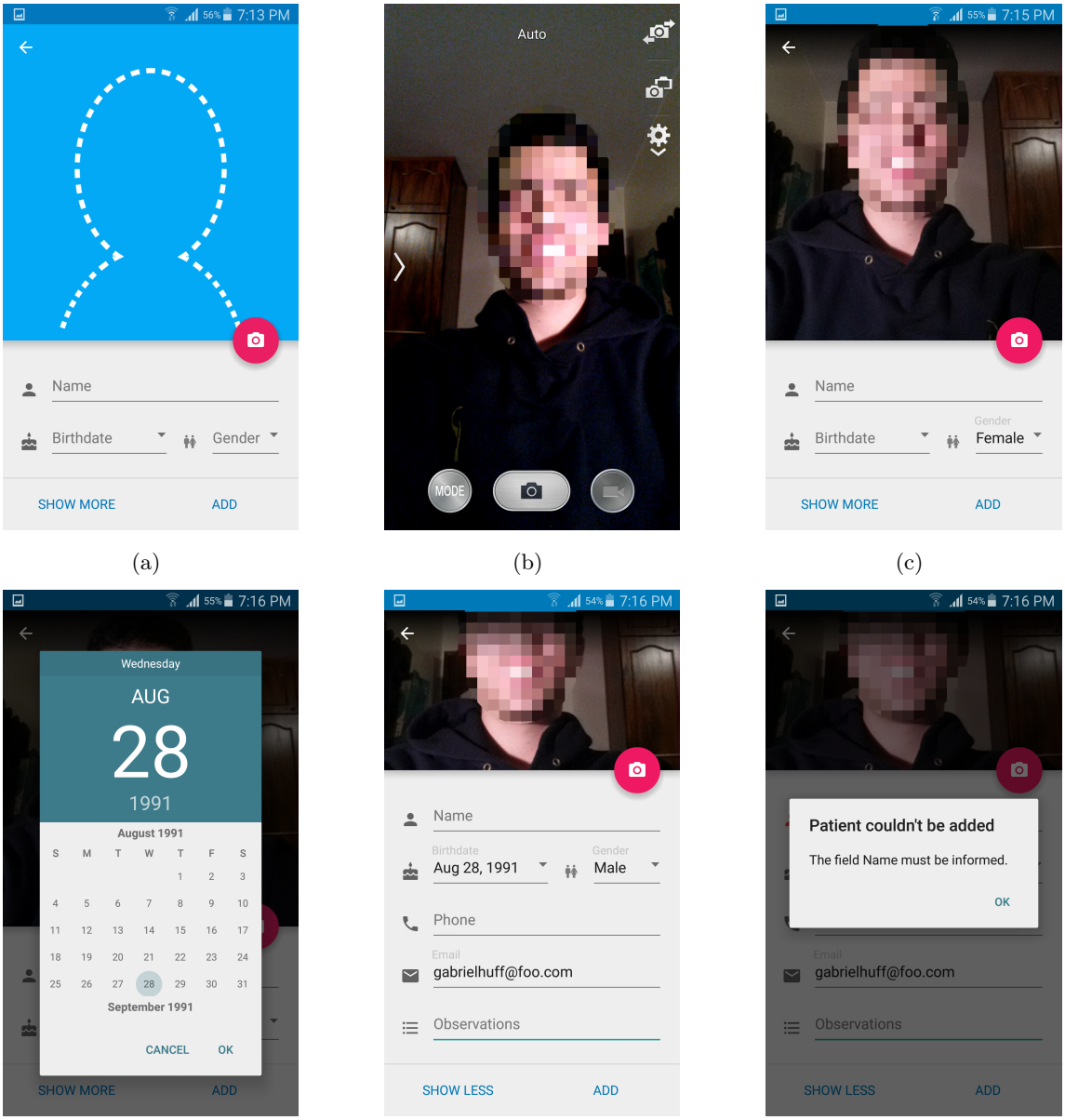



Figura 4.4: Telas da AddPatientsActivity

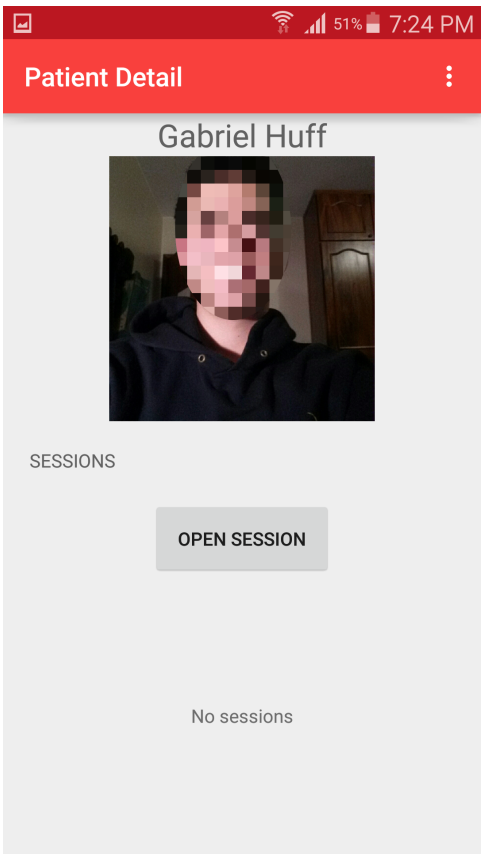



Figura 4.5: Tela da PatientDetailActivity

Há um botão que irá alternar entre mostrar ou não mostrar características secundárias do paciente. Quando essas características são mostradas ao usuário, a *Toolbar*, e consequentemente a imagem do paciente, serão comprimidos.

A operação de adição de paciente pode ser cancelada a qualquer momento ao selecionar o botão de retorno no canto superior esquerdo ou apertando o botão *Back* do Android.

O novo paciente pode ser inserido através do botão de confirmação no canto inferior direito. Caso alguma característica primária não tenha sido definida, um diálogo será exibido de modo a informar o usuário.

Esses possíveis cenários estão ilustrados na figura 4.4.

### SettingsActivity

A SettingsActivity fornece as atuais configurações da aplicação e possibilita a alteração das mesmas pelo usuário.

#### PatientDetailActivity

Apresenta as informações de um determinado paciente. Atualmente, essa *activity* não está completamente implementada. Apenas exibe o nome do paciente, a sua imagem, uma lista de sessões realizadas e um botão. Esse botão pode ter dois comportamentos, dependendo do fato de o paciente ter ou não uma sessão aberta. Para um paciente com uma sessão aberta, esse botão será usado para visualizar essa sessão, enquanto para pacientes sem sessões abertas ele será usado para iniciar uma nova sessão.

A figura 4.5 ilustra essa tela.

#### SessionDetailActivity

A SessionDetailActivity exibe uma lista de operações realizadas na sessão. Para identificação rápida pelo usuário, cada tipo de operação é identificada por um círculo com cores específicas.

Na parte superior há um botão indicando o status da conexão com o estimulador. Esse botão ficará vermelho caso algum detalhe da conexão (dispositivo bluetooth selecionado ou lista de pares canal-músculo) não esteja definido, ou verde caso contrário. Ao clicar nesse botão, o usuário poderá editar os detalhes da conexão.

Também possui um FAB no canto inferior direito que, ao selecionado, abrirá um diálogo de seleção de nova operação. O usuário poderá selecionar um tipo de operação (Cronaxia, Reobase, Acomodação, Terapia em Malha Aberta ou Terapia em Malha Fechada).

A figura 4.6 ilustra os cenários descritos nos parágrafos anteriores.

#### StimulatorConfigActivity

A StimulatorConfigActivity possibilita a seleção do estimulador e a definição dos pares canal-músculo a serem utilizado nas operações de uma sessão.

Possui um botão para a seleção do estimulador, que exibe o estimulador atualmente selecionado ou um texto indicando que nenhum estimulador foi selecionado. Quando selecionado, esse botão abrirá um diálogo de seleção de estimulador. Esse diálogo exibirá uma lista de estimuladores próximos encontrados durante um processo de busca. Durante esse processo, um indicador circular de atividade será mostrado. Ao final do processo, o usuário poderá reiniciar a busca por dispositivos. Ao encontrar um novo estimulador, um novo elemento será adicionado a lista, contendo uma imagem referente ao modelo do estimulador (atualmente só o modelo Estimulador v1.0 está definido). Esse elemento poderá ser clicado para selecionar um novo estimulador.

Possui um botão para a seleção de pares canal músculo, que exibe a atual configuração dos canais utilizados. Ao clicado, esse botão abrirá um diálogo de manipulação de canais. O usuário poderá selecionar um canal e um músculo em *Spinners*, ou seja existem opções pré-definidas. Para adicionar um novo par, o usuário deverá selecionar o botão de adição ao lado dos *Spinners*. Isso irá adicionar um novo elemento à uma lista de pares canal-músculo. Esse elemento será composto pelo

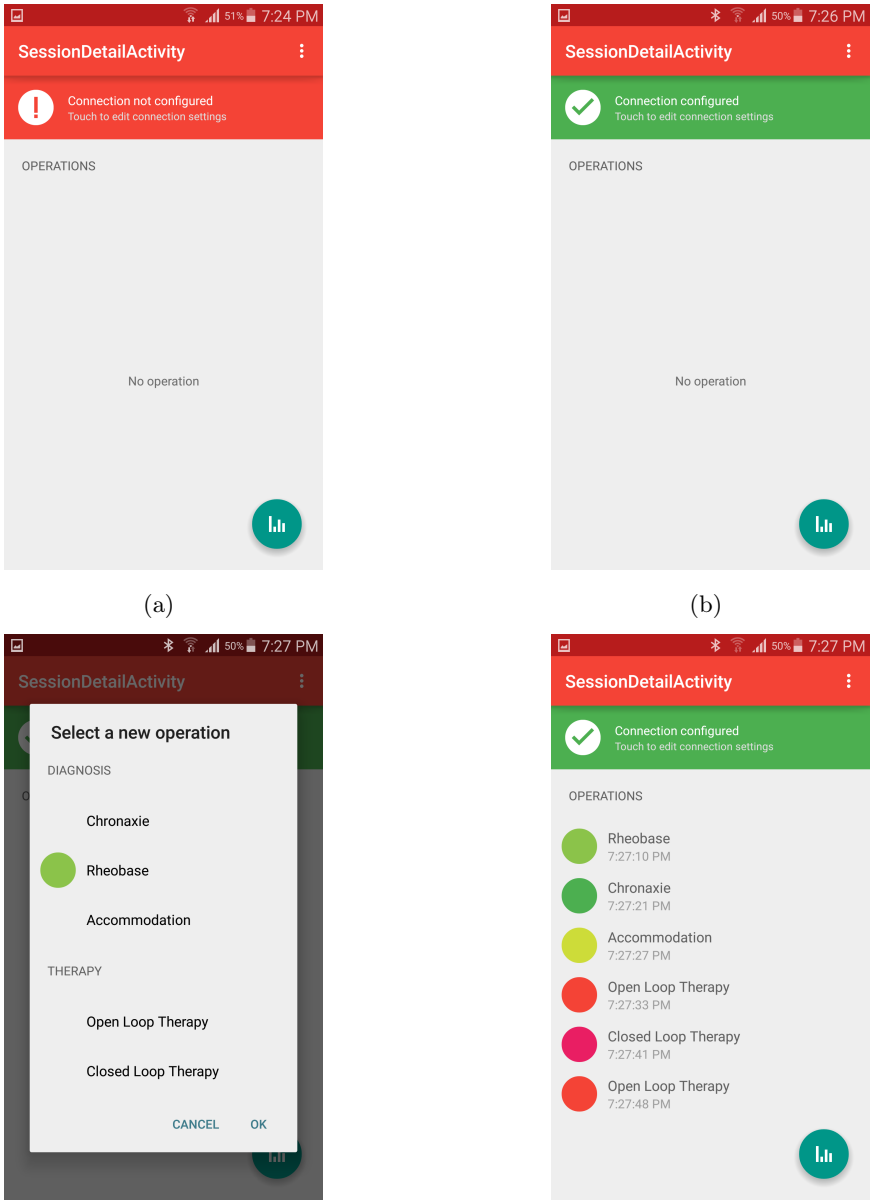
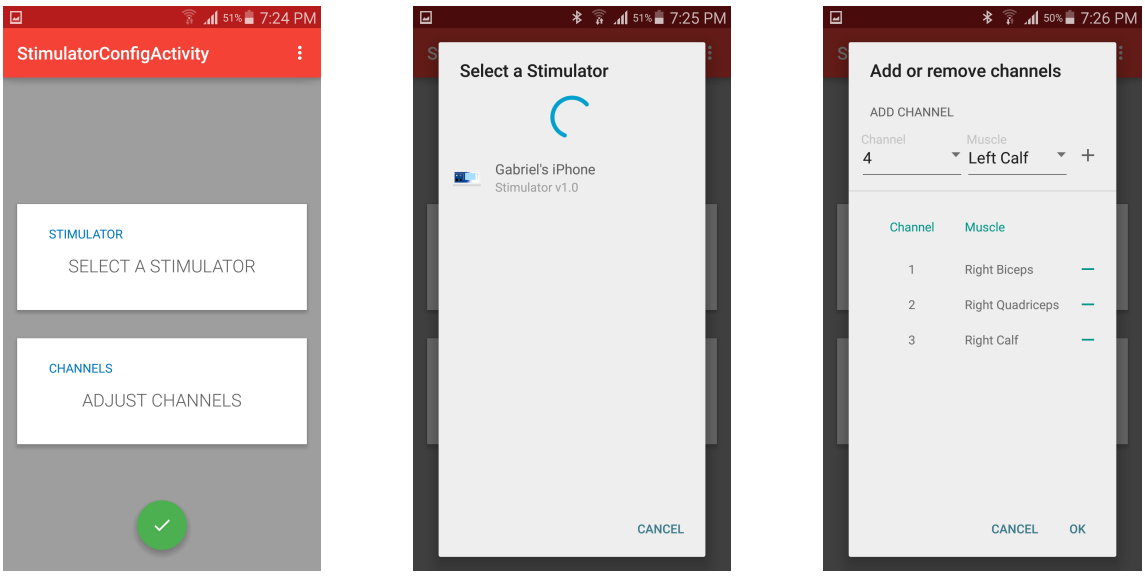



Figura 4.6: Telas da SessionDetailActivity




(a) (b) (c)

Figura 4.7: Telas da StimulatorConfigActivity

número do canal, o nome do músculo e um botão para remoção do par. Ao clicar nesse botão, o par será removido da lista. Para confirmar as alterações realizadas, o usuário deverá pressionar o botão de confirmação, no canto inferior direito do diálogo. Para cancelar as alterações, pode-se clicar no botão de cancelamento na parte inferior do diálogo, ou pressionar o botão *Back* do Android.

Para confirmar as modificações feitas nos detalhes de conexão, pode-se selecionar um FAB na parte inferior da tela. Para cancelar as modificações, pode-se apertar o botão *Back* do Android.

Os possíveis cenários estão ilustrados na figura 4.7.

#### OperationDetailActivity

A OperationDetailActivity permite a visualização dos resultados de uma operação.

A implementação dessa *activity* está pendente.

#### StimulationActivity

A StimulationActivity permite a visualização dos resultados parciais de uma operação, i.e o acompanhamento do progresso de uma operação em tempo real.

A implementação dessa *activity* está pendente.

### 4.1.2 Primeiras impressões de Usuários

Foram realizados testes nos quais usuários eram introduzidos a aplicação desenvolvida, com o objetivo de verificar se a interface gráfica era didática o suficiente para possibilitar um aprendizado

rápido por parte do usuário.

Entre os usuários, foram selecionados participantes do projeto de desenvolvimento do estimulador, pessoas sem conhecimento do projeto e pessoas menos familiarizadas com dispositivos móveis e interfaces homem-máquina.

De modo geral, percebeu-se que os participantes do projeto tiveram muita facilidade na adaptação ao uso da aplicação, o aprendizado foi praticamente instantâneo.

As pessoas sem conhecimento do projeto, mas familiarizadas com dispositivos móveis e interfaces homem-máquina tiveram imensa facilidade na familiarização com a parte do cadastro de pacientes e na visualização de pacientes cadastrados, já que os mecanismos utilizados para esses procedimentos são similares a aplicações de agendas dos dispositivos móveis atuais. Porém tiveram dificuldades com as outras funcionalidades da aplicação, o que era esperado, já que elas não possuíam conhecimentos acerca do projeto.

Um resultado interessante foi que o grupo referente às pessoas com menor familiarização com dispositivos móveis e interfaces homem-máquina, tiveram uma adaptação relativamente rápida aos mecanismos de adição e visualização de pacientes, o que é um indício de que a interface gráfica desenvolvida possui características que facilitariam o aprendizado.

## 4.2 Armazenamento de Dados

A organização de dados proposta na seção 3.6 foi implementada com sucesso durante o desenvolvimento, nos três níveis de abstração desejados: Banco de Dados, Content Providers e Sistema de Arquivos.

A verificação da funcionalidade do banco de dados TreatmentsDatabase foi feita utilizando-se a seguinte metodologia:

- Criou-se uma versão debug da aplicação. Essa versão foi instalada no dispositivo de teste.
- Fez-se diversas manipulações do banco de dados através da interface gráfica do programa, como por exemplo adicionar novos pacientes, sessões e operações.
- Utilizando a ferramenta ADB (Android Debugging Bridge), executou-se o comando "*adb shell*". Esse comando inicia uma sessão remota no dispositivo alvo.
- Com a sessão remota iniciada, executou-se o comando "*run-as com.huff.tg2*". Esse comando faz com que seja iniciada uma sessão com o usuário UNIX da aplicação (válido apenas para pacotes do tipo debug). Com isso, pode-se ter acesso ao diretório interno da aplicação, que contém todos os seus arquivos privados, incluindo o arquivo de banco de dados *TreatmentsDatabase.db*.
- Copiou-se o arquivo *TreatmentsDatabase.db* para a máquina de desenvolvimento.

- Utilizando-se a ferramenta *sqlite3*, verificou-se os conteúdos do banco de dados, incluindo tabelas e gatilhos. O motivo de se usar o *sqlite3* na máquina de desenvolvimento e não no dispositivo é simples: o dispositivo não possuía esse binário.

Ao final do processo verificou-se que os dados armazenados no banco TreatmentsDatabase refletiam as alterações feitas a partir da interface gráfica.

De maneira similar, foi feita a verificação do Sistema de Arquivos da aplicação, ou seja, a estrutura de seu diretório interno. Verificou-se que a organização estava de acordo com o proposto. Só há uma pendência: é necessário implementar um mecanismo que apague arquivos relacionados a pacientes e operações, ou seja, imagens de pacientes e resultados de operações, quando os dados aos quais eles se referem são removidos do banco de dados.

Para a verificação das funcionalidades dos Content Providers (TreatmentsProvider e FileProvider) foram utilizadas as ferramentas de debug do AndroidStudio. Inserindo-se *breakpoints* no código-fonte e criando-se uma versão debug da aplicação foi possível verificar as chamadas dos métodos das implementações, o que facilitou a correção de eventuais bugs para se atingir o comportamento desejado para as URIs suportadas.

# Capítulo 5

# Conclusão

*Nesse capítulo serão interpretados os resultados obtidos, descritos no capítulo 4, afim de se formular considerações finais acerca da funcionalidade da aplicação desenvolvida, além de apontar pendências a serem resolvidas em trabalhos futuros.*

## 5.1 Considerações Finais

O EDE e a NMES são metodologias eficientes e de baixa complexidade utilizadas no diagnóstico e tratamento da PNMDC, um quadro que se torna cada vez mais frequente em pacientes internados em UTIs. Infelizmente, problemas como a inexistência de equipamentos específicos para a realização dessas operações e a falta de profissionais qualificados dificultam a implementação dessas técnicas. Com o objetivo de superar essas barreiras, uma equipe de professores e alunos da UnB criou um projeto intitulado *Prototipagem de um Estimulador Neuromuscular e Transferência de Tecnologia Para Uso Durante a Internação Hospitalar*, que consiste na criação de um eletroestimulador específico para a realização do EDE e da NMES e na qualificação de equipes de fisioterapeutas para utilizar o equipamento desenvolvido. No entanto, era o necessário a criação de uma interface entre os fisioterapeutas e o estimulador. A criação dessa interface foi o foco principal desse trabalho.

A interface foi implementada através do desenvolvimento de uma aplicação para a plataforma Android, utilizando-se as APIs mais recentes e obedecendo-se as boas práticas, com o objetivo de simplificar a manutenção do código gerado em trabalhos futuros.

O produto desenvolvido fornece uma interface específica para a interação do usuário com o estimulador da UnB, o que minimiza a complexidade da utilização do equipamento pelo usuário. Tal fator contribui não somente para a redução do tempo de realização de operações de eletroestimulação, mas também para simplificação do processo de aprendizado e familiarização do usuário com o equipamento. Tal propriedade pôde ser observada através de testes realizados com usuários potenciais.

Além disso, houve uma preocupação em criar interfaces amigáveis e intuitivas, do ponto de vista do usuário, através do uso de elementos descritos nas normas do Design Material, o que

também contribui para a simplificação do processo de aprendizado. Vale a pena ressaltar que nem todos os elementos gráficos da aplicação desenvolvida estão de acordo com as normas do Design Material, como por exemplo algumas medidas e tamanhos de fonte. Isso pode ser justificado em parte como uma tentativa de criar uma identidade visual própria do produto final. Contudo, alguns elementos da aplicação ainda podem ser melhorados do ponto de vista gráfico, como por exemplo a PatientDetailActivity e a SettingsActivity.

O protocolo de comunicação criado possuiu um alto grau de confiabilidade, fato que pôde ser alcançado pela utilização de reundâncias. Tal característica é muito importante em equipamentos médicos, nos quais há uma maior sensibilidade a falhas. Caso haja no futuro, por algum motivo, a necessidade de aumentar a taxa de troca de informações, pode-se remover algumas dessas redundâncias. No entanto, estima-se que isso não será necessário, já que a velocidade atual do protocolo é relativamente alta.

## 5.2 Trabalhos Futuros

Algumas funcionalidades podem ser otimizadas ou implementadas no projeto em trabalhos futuros. Essas funcionalidades estão descritas a seguir:

- Aprimoramento da Interface Gráfica de algumas Activities, como por exemplo a PatientDetailActivity, que atualmente não exibe as características do paciente e a SettingsActivity, que ainda não está implementada.
- Adição das seguintes características secundárias aos pacientes: Peso, Altura e IMC (Índice de Massa Corporal).
- Atualização do protocolo de comunicação para suportar o envio de parâmetros de Diagnóstico.
- Portabilidade para tablets.
- Tradução para a língua portuguesa.

# REFERÊNCIAS BIBLIOGRÁFICAS


[1] HATCH, R. Improving long-term recovery after critical ilness in the uk: The intensive care outcome network (icon) study & the intensive care aftercare network - i-canuk. *ICU Management*, v. 11, n. 3, p. 17–9, 2011.

[2] HERMANS, G. et al. Impact of intensive insulin therapy on neuromuscular complications and ventilator dependency in the medical intensive care unit. *Am J Respir Crit Care Med*, v. 175, n. 5, p. 480–9, 2007.

[3] LATRONICO, N. et al. Critical illness myopathy and neuropathy. *Lancet*, v. 347, n. 9015, p. 1579–82, 1996.

[4] KHAN, J. et al. Early development of critical illness myopathy and neuropathy in patients with severe sepsis. *Neurology*, v. 67, n. 8, p. 1421–5, 2006.

[5] BEDNARIK, J.; LUKAS, Z.; VONDRACEK, P. Critical illness polyneuromyopathy: the electrophysiological components of a complex entity. *Intensive Care Med*, v. 29, n. 9, p. 1505–14, 2003.

[6] BOLTON, C. F. et al. Polyneuropathy in critically ill patients. *J Neurol Neurosurg Psychiatry*, v. 47, n. 11, p. 1223–31, 1984.

[7] ROUTSI, C. Electrical muscle stimulation prevents critical illness polyneuromyopathy: a randomized parallel intervention trial. *Crit Care*, v. 355, n. 16, p. R74, 2010.

[8] DURAND, D. Electric stimulation of excitable tissue. *The Biomedical Engineering HandBook*, 2000.

[9] RUSSO, T. L. et al. Electrical stimulation based on chronaxie reduces atrogin-1 and myod gene expressions in denervated rat muscle. *Muscle Nerve*, v. 35, n. 1, p. 87–97, 2007.

[10] RUSSO, T. L. et al. Electrical stimulation increases matrix metalloproteinase-2 gene expression but does not change its activity in denervated rat muscle. *Muscle Nerve*, v. 37, n. 5, p. 593–600, 2008.

[11] CUMMINGS, J. P. Conservative management of peripheral nerve injuries utilizing selective electrical stimulation of denervated muscle with exponentially progressive current forms. *J Orthop Sports Phys Ther*, v. 7, n. 1, p. 11–5, 1985.

[12] BÓ, A. *Compensation active de tremblements pathologiques des membres supérieurs via la stimulation électrique fonctionnelle*. Tese (Doutorado) — University of Montpellier II, 2011.

[13] GOOGLE. *More About Android Studio*. Julho 2015. Disponível em: <developer.android.com/sdk>.

[14] GOOGLE. *Material Design*. Julho 2015. Disponível em: <google.com/design/spec/material-design>.

[15] HOOVERS. *Bluetooth Traveler*. Abril 2010. Disponível em: <hoovers.com>.