UNIVERSIDADE FEDERAL DO CEARÁ
INSTITUTO UFC VIRTUAL
PROGRAMA DE GRADUAÇÃO EM SISTEMAS E MÍDIAS DIGITAIS

PAULO ROGÉRIO MATHEUS BONFIM LEITÃO

DESENVOLVIMENTO NATIVO VS REACT NATIVE: UMA ANÁLISE COMPARATIVA NA CODIFICAÇÃO DE UMA APLICAÇÃO PARA FITNESS

FORTALEZA

2019

PAULO ROGÉRIO MATHEUS BONFIM LEITÃO

DESENVOLVIMENTO NATIVO VS REACT NATIVE: UMA ANÁLISE COMPARATIVA NA CODIFICAÇÃO DE UMA APLICAÇÃO PARA FITNESS

Relatório Técnico apresentada ao Curso de Sistemas e Mídias Digitais do Instituto UFC Virtual da Universidade Federal do Ceará, como requisito parcial à obtenção do título de Bacharel em Sistemas e Mídias Digitais.

Orientador: Prof. Dr. Windson Viana de Carvalho.

FORTALEZA

2019

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

L549d Leitão, Paulo Rogério Matheus Bonfim.
Desenvolvimento nativo vs React Native : uma análise comparativa na codificação de uma aplicação para fitness / Paulo Rogério Matheus Bonfim Leitão. – 2019.
54 f.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Instituto UFC Virtual, Curso de Sistemas e Mídias Digitais, Fortaleza, 2019.
Orientação: Prof. Dr. Windson Viana de Carvalho.

1. React Native. 2. Experiência do usuário. 3. Ferramentas de desenvolvimento multiplataforma. I. Título.

CDD 302.23

PAULO ROGÉRIO MATHEUS BONFIM LEITÃO

NATIVO VS REACT NATIVE: UMA ANÁLISE COMPARATIVA NO DESENVOLVIMENTO DE UMA APLICAÇÃO PARA FITNESS

Relatório Técnico apresentada ao Curso de Sistemas e Mídias Digitais do Instituto UFC Virtual da Universidade Federal do Ceará, como requisito parcial à obtenção do título de Bacharel em Sistemas e Mídias Digitais.

Aprovada em: ___/___/_____.

BANCA EXAMINADORA

_____
Prof. Dr. Windson Viana de Carvalho
Universidade Federal do Ceará (UFC)

_____
Prof. Dr. Edgar Marçal de Barros Filho
Universidade Federal do Ceará (UFC)

_____
Prof. Dr. Marcio Espíndola Freire Maia
Universidade Federal do Ceará (UFC)

AGRADECIMENTOS

Meus agradecimentos para todos aqueles que me ajudaram, não só para aqueles que de alguma forma contribuíram para a minha vida acadêmica, mas também para todos que me ajudaram em algum momento da minha vida.

Ao meu orientador que não desistiu de mim depois de todos esses semestres.

RESUMO

Atualmente o mundo dos smartphones é composto por dois sistemas operacionais, o Android e o iOS, que juntos possuem o domínio quase completo do mercado. Apesar de serem apenas dois sistemas operacionais, desenvolver duas versões de um mesmo aplicativo gera muitos gastos e problemas como bugs, além do custo de manutenção desses aplicativos. Para amenizar esse problema, os desenvolvedores utilizam as ferramentas de desenvolvimento multiplataforma. Porém, o uso dessas ferramentas traz desvantagens que afetam negativamente a experiência do usuário, o que acaba tornando os aplicativos gerados por essas ferramentas inferiores aos nativos. Para investigar se é possível um aplicativo gerado por essas ferramentas multiplataforma poder alcançar o mesmo nível de usabilidade de um aplicativo nativo, foram desenvolvidos dois aplicativos de atividade física como provas de conceito: um utilizando o Android e outro utilizando o framework React Native. Ao fim do desenvolvimento, foram coletadas métricas das versões e foi realizado um Teste de Usabilidade com estudantes do curso de Sistemas e Mídias Digitais, em que foi aplicada uma entrevista estruturada e um questionário SUS. Por fim, foi feita uma discussão sobre os resultados encontrados e as versões geradas.

Palavras-chave: React Native. Experiência do Usuário. Ferramentas de Desenvolvimento Multiplataforma.

ABSTRACT

Today's smartphone world is composed of two operating systems, Android and iOS, which together have nearly complete market dominance. Although they are just two operating systems, developing two versions of the same application generates many costs and issues such as bugs, as well as the cost of maintaining these applications. To alleviate this problem, developers use cross-platform development tools. However, the use of these tools has disadvantages that negatively affect the user experience that makes the applications generated by these tools inferior to the native ones. To investigate whether an application generated by these multi-platform tools can achieve the same level of usability as a native application, two fitness applications were developed as a proof of concept: one using Android and the other using the React Native framework. At the end of development, version metrics were collected and a Usability Test was conducted with students from the Systems and Digital Media course where a structured interview and a SUS questionnaire were applied. Finally, there is a discussion about the results found and the generated versions.

Keywords: React Native. User Experience. Cross-Platform Tools.

LISTA DE FIGURAS

LISTA DE GRÁFICOS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Program Interface*
BLE	Bluetooth Low Energy
CPT	*Cross-platform Tools*
MFA	*Mobile Fitness Applications*
OHA	*Open Handset Alliance*
POC	Prova de Conceito
PWA	*Progressive Web Apps*
SO	Sistema Operacional
UUID	Identificadores de Números Únicos

SUMÁRIO

1 INTRODUÇÃO

Atualmente o mundo dos *smartphones* é dominado por dois sistemas operacionais, o Android e o iOS, que juntos possuem 99.7% da fatia de mercado dos *smartphones* Corporation (2017), fazendo com que empresas e pesquisadores foquem seus esforços nessas duas plataformas. Os dois sistemas operacionais podem ser encontrados em relógios inteligentes, *smart* TV, carros inteligentes, *tablets*, entre outros dispositivos, além dos já tradicionais *smartphones*. Apesar de serem apenas dois sistemas, desenvolver duas versões de um mesmo aplicativo gera muitos gastos e problemas como *bugs*, além do custo de manutenção desses aplicativos.

Segundo El-kassas et al. (2017), desenvolver um aplicativo de forma nativa, ou seja, utilizando as SDK (Software Development Kit) fornecidas pelas empresas proprietárias baseia-se em duas estratégias. A primeira consiste em manter equipes diferentes para que um mesmo aplicativo seja lançado em todos os sistemas operacionais, já a segunda estratégia consiste em uma única equipe desenvolver diversas versões dos aplicativos obrigando os desenvolvedores a terem o conhecimento técnico de diversas plataformas. As duas estratégias possuem desvantagens como aumentar tempo de desenvolvimento, custo e manutenção.

Para amenizar esse problema, os desenvolvedores utilizam as Ferramentas de Desenvolvimento Multiplataforma, ou *Cross-Platform Tools* (CPT), que permitem que o desenvolvedor crie um aplicativo uma única vez e o distribua para múltiplas plataformas (EL-KASSAS ET AL., 2017), além de cortar custos e facilitar a manutenção (ÂNGULO; FERRE, 2014).

O uso de CPT também traz problemas pois muitas vezes não são respeitadas as convenções de cada plataforma, no caso do Android, o Material Design[1] e, para o *iOS*, o Human Interface Guidelines[2], pois a maioria das CPT geram o mesmo aplicativo que possui o mesmo comportamento afetando negativamente a Experiência do Usuário (UX) (ÂNGULO; FERRE, 2014).

1 Material Design – https://material.io/design/. Acessado em 27 nov. 2019
2 Human Interface Guidelines – https://developer.apple.com/design/human-interface-guidelines/. Acessado em 27 nov. 2019

Segundo Xanthopoulos e Xinogalos (2013), “o objetivo máximo do desenvolvimento de aplicativos *cross-platform* é ter o desempenho de um aplicativo nativo e rodar em quantas plataformas forem possíveis”. O segundo objetivo já foi alcançado tendo em vista o domínio das plataformas Android e iOS, que são o grande foco das CPT, enquanto o primeiro objetivo, que impacta diretamente a UX, é o de algumas dessas ferramentas e o tema desta pesquisa: é possível alcançar o mesmo nível de usabilidade de um aplicativo nativo utilizando uma CPT?

O React Native foi o framework escolhido neste trabalho, foi escolhido por sua proposta em entregar uma interface nativa com o uso de componentes que se adaptam a cada plataforma possuindo as respectivas interações. Essa característica auxilia o React Native no objetivo de analisar a usabilidade de um aplicativo gerado por uma CPT em relação à sua versão nativa, além da sua ampla aceitação no mercado e na comunidade JavaScript fazendo dele uma das ferramentas mais famosas da atualidade.

2 REFERENCIAL TEÓRICO

O mercado de celulares está crescendo cada vez mais, chegando ao número de mais de 3 bilhões de pessoas com celulares (LECHETA, 2014). Nos últimos 10 anos, os Sistemas Operacionais (SO) Android e iOS se estabeleceram e dominaram o mercado de *smartphones*, como mostrado na Tabela 1, fazendo com que os desenvolvedores e pesquisadores focassem seus estudos e esforços nessas duas plataformas.

Tabela 1 – Venda global de *smartphones* em 2016 para usuários finais por SO no quarto trimestre de 2016 em milhares de unidades

Sistema Operacional	**4º trimestre de 2016 – Unidades**	**4º trimestre de 2016 – Participação (%)**	**4º trimestre de 2015 – Unidades**	**4º trimestre de 2015 – Participação (%)**
Android	352.669,9	81.7	325.394,4	80.7
iOS	77.038,9	17.9	71.525,9	17.7
Windows	1.092,2	0.3	4.395,0	1.1
BlackBerry	207,9	0	906,9	0.2
Outros SO	530,4	0.1	887,3	0.2
Total	**431.539,3**	**100.0**	**403.109,4**	**100.0**

Fonte: Gartner (2016).

2.1 Plataforma Android

Para Lecheta (2014, p. 22), o Android "consiste em uma nova plataforma de desenvolvimento para aplicativos móveis, baseado em um sistema operacional Linux". O Android foi criado pela Android Inc. em meados de 2005, empresa que era financiada pela Google e que mais tarde foi comprada por ela, mas que atualmente é mantida pela Open Handset Alliance (OHA)[3], que é um consórcio entre empresas de tecnologia e telefonia que decidem o futuro do Android.

Atualmente o Android se encontra na versão 9.0, também chamado de Android Pie.

3 Open Handset Alliance – https://www.openhandsetalliance.com/. Acessado em 27 nov. 2019

Além dos Smartphone, o Android pode ser encontrado em diversos tipos de dispositivos como Tablets, *wearables*, *Smart TV* e automóveis. A maioria da plataforma e sistema Android está sob a licença Apache 2.0 enquanto partes do kernel do Linux estão sob licença GPLv2, dessa forma, não é necessário pagar nada para desenvolver aplicativos Android, mas é necessário pagar uma taxa única de 25 dólares para publicar na loja de aplicativos da Google, a Google Play.

O ambiente de desenvolvimento padrão é o Android Studio, que está disponível para Windows, MacOS e distribuições Linux. As linguagens de programação suportadas pelo Android são Java, C++ e Kotlin, sendo o Java a linguagem oficial da plataforma.

2.2 Plataforma iOS

Segundo Lecheta (2015, p. 30), o "sistema operacional iOS da Apple se destaca como uma das principais plataformas de desenvolvimento mobile do mercado" sendo famoso por sua integração entre o SO e o hardware e entregando para o usuário um ótimo desempenho e experiência de uso. O primeiro iPhone foi lançado em 2007 durante o Apple Worldwide Developers Conference (WWDC), evento organizado anualmente pela Apple para anunciar as principais novidades da empresa.

O iOS se encontra na versão 13.1 e, assim como seu concorrente, pode ser encontrado em Smart TV (tvOS) e smartwatch (watchOS). O ambiente de desenvolvimento padrão do iOS é o Xcode, que é exclusivo para o MacOS. As linguagens de programação suportadas pelo iOS são o Objective-C e o Swift, sendo o Swift a linguagem oficial. Para desenvolver para iOS, não é cobrada nenhuma taxa, mas, para divulgar um aplicativo na App Store, é necessário pagar uma taxa anual no valor de 99 dólares, sendo essa a única forma de distribuição dos aplicativos.

2.3 O que é uma Ferramenta de Desenvolvimento Multiplataforma Para Dispositivos Móveis

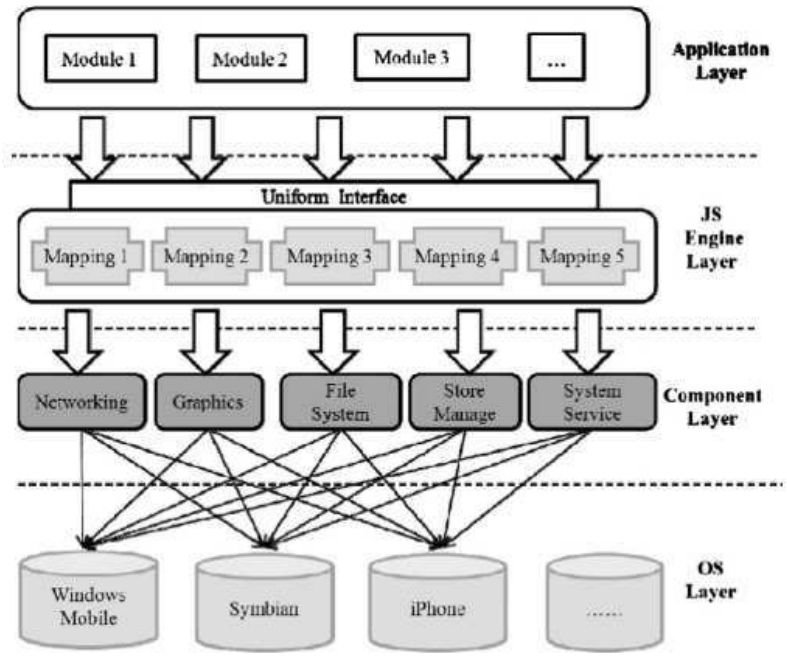
Para Bernades e Miyake (2016, p. 1892), "a abordagem de desenvolvimento nativo não é muito atraente devido ao custo, à quantidade de recursos e aos tempos associados

com as atividades de desenvolvimento para cada plataforma". Uma solução para esse problema seria codificar apenas uma vez e publicar o aplicativo para todas as plataformas com o uso de CPT (EL-KASSAS et al, 2017).

Como citado anteriormente, desenvolver aplicativos nativos é muito custoso, e por isso muitos desenvolvedores recorrem ao uso de CPT. Segundo El-kassas et al. (2017), existem 3 tipos aplicativos móveis, que são: Aplicativos Web, Aplicativos Híbridos e Aplicativos Nativos, além de 6 abordagens adotadas pelas ferramentas multiplataforma, que são: Compilação, Baseada em Componentes, Interpretada, Modelagem, Cloud-Based e Merged. Dessas abordagens, o React Native poderia ser classificado como uma abordagem Baseada em Componentes.

El-kassas et al. (2017) classificam a Abordagem Baseada em Componentes como pacotes ou módulos que contêm um conjunto de funções ou dados possuindo a mesma interface para ambas as plataformas, mas com diferentes implementações internas.

Figura 1: Modelo *framework* baseado em componentes.



Fonte: El-kassas et al. (2017).

2.4 React Native

Anunciado em 2015, React Native é um framework da empresa Facebook, de código-aberto, para criação de aplicativos nativos, ou incremento de aplicações nativas já

existentes, que “não comprometem a experiência do usuário” (FACEBOOK, 2019a). Foi construído em cima da biblioteca React, também do Facebook, que é uma biblioteca para criação de interface de usuário (FACEBOOK, 2019b). React Native ainda não foi oficialmente lançado encontrando-se atualmente na versão 0.61.0, e, por não se tratar de uma versão definitiva do framework, bugs são esperados.

2.4.1 Criação de interfaces no React Native

Para criação de interfaces no React, normalmente é utilizado o JavaScript XML (JSX), que é uma extensão da sintaxe JavaScript que se assemelha bastante ao HTML, em que cada tag é um elemento ou componente React. O resultado visual do JSX vai depender de qual biblioteca de renderização está sendo utilizada; para a web, é utilizado o ReactDOM, e é renderizado código HTML. No React Native, o JSX vai exibir o elemento visual referente a cada plataforma.

O JSX pode conter expressões JavaScript como chamadas de funções e após compilado se transforma numa função JavaScript tradicional. Não existe o uso de HTML ou CSS na criação de interfaces no React Native fazendo com que a estilização se dê exclusivamente através de objetos JavaScript.

2.4.2 Componentes React Native

Componentes em React “permitem você dividir a UI em partes independentes, reutilizáveis e pensar em cada parte isoladamente” (FACEBOOK, 2019b), assim, componentes em React Native possuem o mesmo comportamento, tendo sua implementação invisível ao desenvolvedor na maioria das vezes, enquadrando-se no que foi descrito na seção 2.3.

O React Native possui componentes padrões em sua API para a criação de interface, além de permitir a criação de novos componentes que podem utilizá-los ou que se comunicam diretamente com cada plataforma através dos Módulos Nativos.

Um componente tem um entendimento específico para cada SO, por exemplo, o componente View, no Android, é renderizado como uma View e no iOS é renderizado uma UIView. Porém há casos em que não existe uma equivalência de componentes entre as plataformas levando o desenvolvedor a utilizar diferentes componentes para a interação do

usuário.

3 METODOLOGIA DA PESQUISA

Trata-se de uma pesquisa da área das Ciências Exatas e da Terra abordando o Desenvolvimento de Softwares, mais especificamente o Desenvolvimento de Aplicativos Móveis Utilizando Tecnologias Híbridas. Para Gil (1991), “as pesquisas descritivas têm como objetivo primordial a descrição das características de determinada população ou fenômeno”, que é exatamente no que se focou esta pesquisa, análise da usabilidade de um aplicativo gerado por CPT em relação ao aplicativo nativo ressaltando as diferenças no desenvolvimento desses aplicativos e uma breve comparação dos aplicativos gerados. A pesquisa foi conduzida de forma experimental nos Laboratórios de Pesquisa & Desenvolvimento do curso de Sistemas e Mídias Digitais na Universidade Federal do Ceará, onde foi possível realizar a coleta e análise dos dados de forma quantitativa e qualitativa dos aplicativos desenvolvidos.

3.1 Etapas da Pesquisa

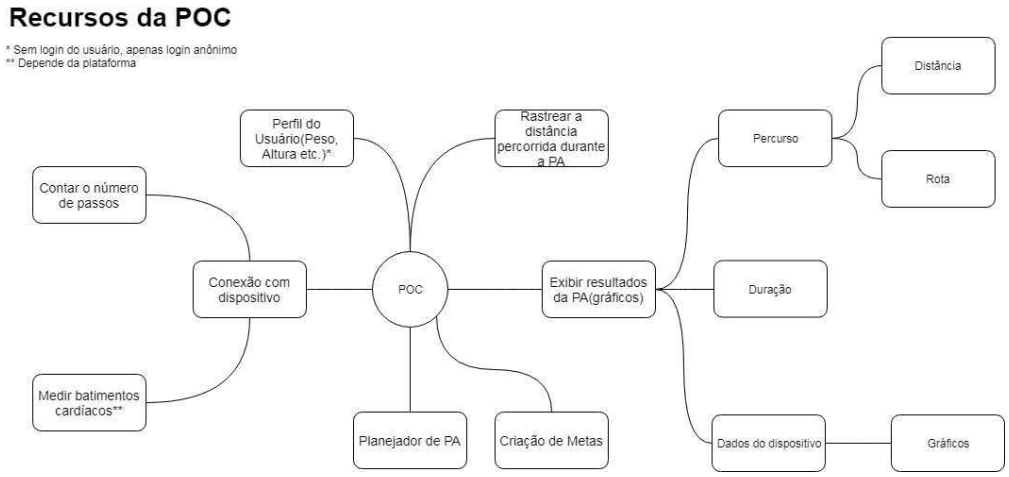
O sistema operacional escolhido para este estudo foi o Android, que é o sistema para dispositivos móveis mais comum da atualidade (CORPORATION, 2017). A pesquisa foi dividida em três etapas, num primeiro momento, foram definidas as funcionalidades e desenvolvidos dois aplicativos de Prova de Conceito (PoC), sendo um deles de forma nativa, ou seja, utilizando ferramentas disponibilizadas pelas empresas responsáveis pelos sistemas operacionais citados anteriormente, assim, servindo como grupo de controle enquanto o outro aplicativo foi desenvolvido utilizando o React Native. Num segundo momento, esses dois aplicativos foram analisados, sendo coletadas métricas de código e informações sobre os aplicativos gerados. Por último foi realizado um teste de usabilidade com usuários que responderam uma entrevista estruturada e um formulário System Usability Scale (SUS).

3.2 Prova de Conceito

Como PoC foi desenvolvida uma aplicação de auxílio a atividades físicas, também conhecida como Mobile Fitness Applications (MFA), que possui duas versões sendo uma desenvolvida em Android e outra desenvolvida em React Native. Rabin e Bock (2011) destacam uma série de funcionalidades interessantes para o desenvolvimento de uma MFA, funcionalidades que são observadas em Gregoski et al. (2012), Leijdekkers e Gay (2013), Consolvo et al. (2006) e Schobel et al. (2013). A escolha das funcionalidades da PoC levou

em consideração o acesso a diversas funcionalidades do smartphone para simular uma MFA do mundo real. As funcionalidades escolhidas em um primeiro momento foram: 1) conexão a um sensor externo como em Gregoski et al. (2012), Leijdekkers e Gay (2013) e Schobel et al. (2013), feita através do Bluetooth, e o dispositivo escolhido foi uma smart band do modelo Xiaomi MiBand 2; 2) Contagem dos passos evidenciados em Leijdekkers e Gay (2013), Rabin e Bock (2011) e Consolvo et al. (2006); 3) Medição do batimento cardíaco como em Gregoski et al. (2012), Leijdekkers e Gay (2013); 4) Visualização dos dados coletados como em Leijdekkers e Gay (2013) e Schobel et al. (2013); 5) Agendamento de atividades físicas como em Leijdekkers e Gay (2013) e Schobel et al. (2013). As funcionalidades podem ser visualizadas na Figura 2.

Figura 2: Mapa mental com as funcionalidades das provas de Conceito.



Fonte: Elaborado pelo autor.

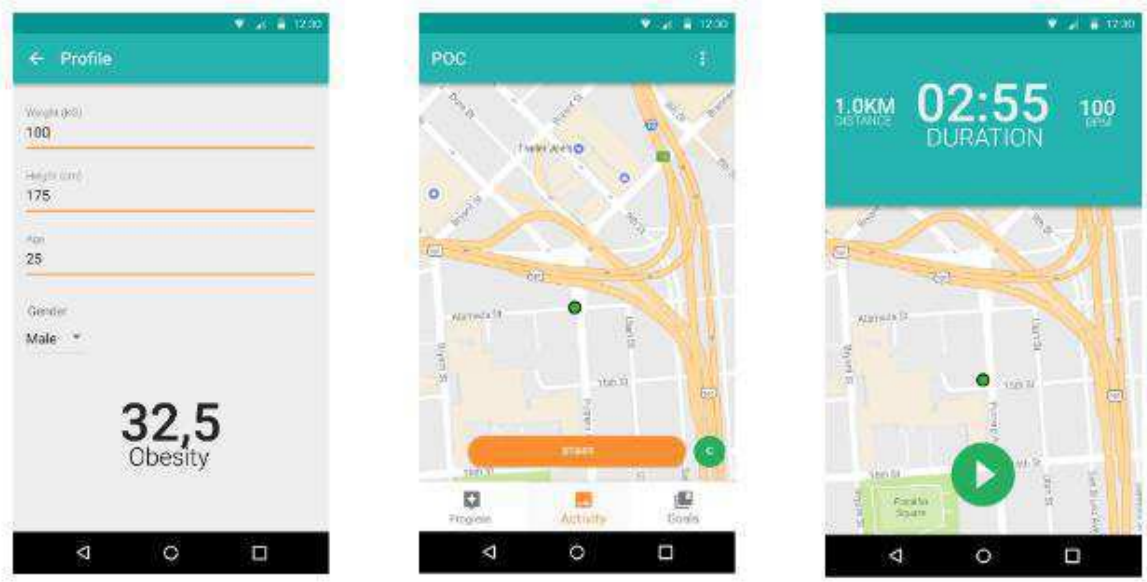
A partir do mapa do mental, é possível listar as principais funções do aplicativo, que são:

- Cadastro de usuário;
- Conexão com dispositivo externo via Bluetooth;
- Obtenção de dados do dispositivo externo;
- Exibição de mapa;
- Exibição da localização do usuário no mapa;
- Exibição de rotas no mapa;
- Criação de atividades físicas;
- Criação de objetivos;

- Rastreamento do usuário;
- Histórico de atividades;
- Detalhe de atividades;
- Exibição de todas as atividades.

Depois de definidas as funcionalidades, o protótipo navegável das telas foi projetado utilizando a versão gratuita do Figma[4], um software para prototipação de telas, totalizando um total de seis telas.

Figura 3: Protótipo de telas I.

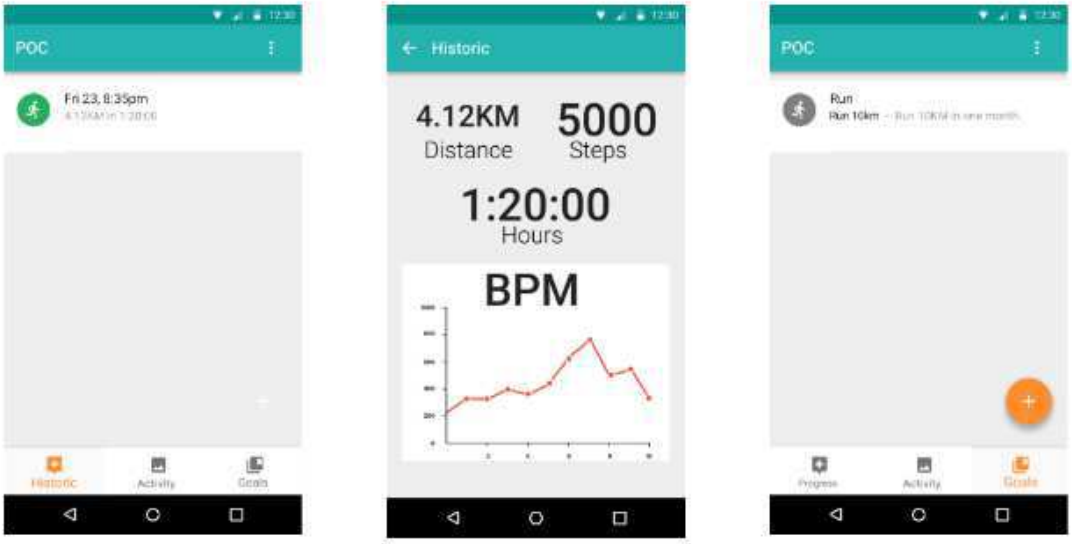


(a) Perfil do Usuário (b) Tela Principal. (c) Tela de Atividade

Fonte: Elaborado pelo autor.

4 Figma – https://www.figma.com. Acessado em 27 nov. 2019.

Figura 4: Protótipo de telas II.



(a) Tela de Histórico (b) Tela de Detalhes (c) Tela de Objetivos

Fonte: Elaborado pelo autor.

3.3 Procedimento

Para a avaliação dos aplicativos, os seguintes procedimentos foram seguidos.

1. Desenvolvimento das provas de conceito;
2. Elaboração da avaliação com usuários;
3. Coleta de métricas de código;
4. Teste com usuários;
5. Sintetização dos resultados.

4 IMPLEMENTAÇÃO DAS PROVAS DE CONCEITO

Este capítulo tem como objetivo descrever o processo de desenvolvimento de cada uma das versões das PoC. Como foi dito, apenas os aplicativos para Android serão testados e por isso apenas essas versões foram desenvolvidas, além de serem projetadas para seguirem as convenções de design do Android.

As versões foram desenvolvidas separadamente e por uma única pessoa, e, tendo em vista essa limitação de recursos, o uso de bibliotecas de terceiros foi bastante explorado. Levou em média um mês e meio para o desenvolvimento de cada versão, e, devido a complicações no desenvolvimento e no tempo do projeto, as funções definidas no capítulo 3.2 de medir os batimentos cardíacos e agendar atividades físicas foram removidas.

Para versionamento do projeto, foram utilizados dois repositórios privados do Github[5] durante todo o período de desenvolvimento e testes por motivos de segurança devido a possível uso mal intencionado da chave do Google Maps; após a conclusão do desenvolvimento dos aplicativos e dos testes, os repositórios foram modificados para serem públicos.

As duas versões possuem as mesmas funcionalidades e também o mesmo número de telas, que são: 1) *Splash Screen*, 2) Cadastro de usuário, 3) Tela Principal, 4) Tela de atividade física, 5) Histórico, 6) Detalhe de atividade e 7) Informações gerais.

4.1 Versão Android

Esta seção tem como objetivo explicar o processo de desenvolvimento da versão Android nativa sendo a linguagem escolhida para o desenvolvimento o Kotlin que, apesar de não ser a linguagem oficial, possui interoperabilidade com o Java e não deve apresentar incompatibilidade ao utilizar as bibliotecas já existentes.

O primeiro passo foi estabelecer comunicação com a Mi Band 2 e para isso foi buscada uma biblioteca proprietária para conexão da própria pulseira ou de terceiros. A Xiaomi, empresa responsável pela Mi Band, não disponibiliza nenhum SDK para conexão do dispositivo, e as bibliotecas de terceiros procuradas em repositórios GitHub estavam todas desatualizadas, além de não funcionarem corretamente, mas forneceram os Identificadores de Números Únicos (UUID) dos serviços da Mi Band. Para tal conexão, foi utilizada a classe

5 Github – https://github.com/. Acessado em 27 nov. de 2019.

BluetoothGatt[6], que permite a conexão, a descoberta de dispositivo e o escaneamento de serviços de dispositivos Bluetooth Low Energy (BLE).

Figura 5: Função para conexão com a pulseira.

```
private fun connect() {
    bluetoothDevice = bluetoothAdapter.getRemoteDevice(MIBAND_ADDRESS)

    Log.v( tag: "PAA:connect",  msg: "Connecting to ${MIBAND_ADDRESS}")
    Log.v( tag: "PAA:connect",  msg: "Device name ${bluetoothDevice.name}")

    bluetoothGatt = bluetoothDevice.connectGatt( context: this,  autoConnect: true, bluetoothGattCallback)
}
```

Fonte: Elaborado pelo autor.

Apenas se conectar ao dispositivo não é o suficiente, sendo necessário o conhecimento dos UUID para ter acesso aos serviços da pulseira, como o número de passos por exemplo. Os UUID foram obtidos a partir de uma biblioteca JavaScript encontrada no GitHub chamada “miband-js”, e a lista pode ser encontrada na Figura 3. Por último era necessário definir as características de cada serviço e como esses serviços seriam chamados, no caso, apenas o pedômetro foi utilizado, como pode ser visto nas Figuras 4 e 5.

Figura 6: Função para conexão com a pulseira.

```
class CustomBluetoothProfile {
    class Basic {
        val service: UUID = UUID.fromString( name: "0000fee0-0000-1000-8000-00805f9b34fb")
        val batteryCharacteristic: UUID = UUID.fromString( name: "00000006-0000-3512-2118-0009af100700")
    }

    class Pedometer {
        var characteristicSteps: UUID = UUID.fromString( name: "00000007-0000-3512-2118-0009af100700")
    }

    class AlertNotification {
        val service: UUID = UUID.fromString( name: "00001802-0000-1000-8000-00805f9b34fb")
        val alertCharacteristic: UUID = UUID.fromString( name: "00002a06-0000-1000-8000-00805f9b34fb")
    }

    class HeartRate {
        var service: UUID = UUID.fromString( name: "0000180d-0000-1000-8000-00805f9b34fb")
        var measurementCharacteristic: UUID = UUID.fromString( name: "00002a37-0000-1000-8000-00805f9b34fb")
        var descriptor: UUID = UUID.fromString( name: "00002902-0000-1000-8000-00805f9b34fb")
        var controlCharacteristic: UUID = UUID.fromString( name: "00002a39-0000-1000-8000-00805f9b34fb")
    }
}
```

Fonte: Elaborado pelo autor.

6 Classe BluetoothGatt – https://developer.android.com/reference/kotlin/android/bluetooth/BluetoothGatt/. Acessado em 27 nov. 2019.

Figura 7: Chamada do número de passos

```
private fun getSteps() {
    try {
        val bchar: BluetoothGattCharacteristic = bluetoothGatt.getService(CustomBluetoothProfile.Basic().service)
                .getCharacteristic(CustomBluetoothProfile.Pedometer().characteristicSteps)
        // The characteristic can be read in BluetoothGattCallback.onCharacteristicRead ...
        if (!bluetoothGatt.readCharacteristic(bchar)) {
            runOnUiThread {
                val toast = Toast.makeText(baseContext,  text: "Failed to get pedometer info", Toast.LENGTH_LONG)
                toast.show()
            }
        }
    } catch (e: Exception) {
        Log.d( tag: "getSteps", e.message)
        e.printStackTrace()
    }
}
```

Fonte: Elaborada pelo autor.

Figura 8: Leitura de passos.

```
override fun onCharacteristicRead(gatt: BluetoothGatt, characteristic: BluetoothGattCharacteristic, status: Int) {
    super.onCharacteristicRead(gatt, characteristic, status)
    val data = characteristic.value
    Log.d( tag: "onCharacteristicRead", characteristic.uuid.toString())
    if (characteristic.uuid == CustomBluetoothProfile.Pedometer().characteristicSteps) {
        Log.d( tag: "onCharacteristicRead",  msg: "pedometer")
        val steps = Pedometer(characteristic.getIntValue(BluetoothGattCharacteristic.FORMAT_UINT16,  offset: 1).toInt())
        if (initialSteps == -1)
            initialSteps = steps.steps

        val currentSteps = steps.steps - initialSteps
        runOnUiThread {
            tvSteps.text = currentSteps.toString()
        }
        updateActivity(mapOf("steps" to currentSteps))
        Log.d( tag: "onCharacteristicRead", currentSteps.toString())

        if (data.size ≥ 8) steps.distance = characteristic.getIntValue(BluetoothGattCharacteristic.FORMAT_UINT32,  offset: 5).toInt()
        if (data.size ≥ 12) steps.calories = characteristic.getIntValue(BluetoothGattCharacteristic.FORMAT_UINT32,  offset: 9).toInt()
    }
}
```

Fonte: Elaborada pelo autor.

Depois de implementada a conexão com a pulseira, as telas definidas no capítulo 3.2 começaram a ser implementadas. A primeira tela a ser implementada foi a de cadastro do perfil do usuário, onde são coletados dados como peso, altura, idade e sexo biológico do usuário, e, logo em seguida, é exibido o Índice de Massa Corpórea (IMC), do inglês Body Mass Index (BMI), a sua implementação pode ser vista na Figura 9, e o resultado da tela pode ser visto na Figura 10. Os dados são salvos no Firebase e serão utilizados para o cálculo de gasto calórico das atividades posteriormente.

Figura 9: Implementação do cálculo IMC.

```
fun calculateBMI() {
    val heightInMeters: Float = this.height.toFloat() / 100F
    val bmi: Float = this.weight / (heightInMeters * heightInMeters)

    view?.findViewById<TextView>(R.id.tvBmi)?.text = String.format("%.2f", bmi)

    when {
        bmi ≤ 18.5 → view?.findViewById<TextView>(R.id.tvBmiCategory)?.text = "THIN"
        bmi in 18.6..24.9 → view?.findViewById<TextView>(R.id.tvBmiCategory)?.text = "HEALTHY"
        bmi in 25.0..29.9 → view?.findViewById<TextView>(R.id.tvBmiCategory)?.text = "OVERWEIGHT"
        else → view?.findViewById<TextView>(R.id.tvBmiCategory)?.text = "OBESE"
    }
}
```

Fonte: Elaborado pelo autor.

Figura 10: Tela de cadastro de usuário.

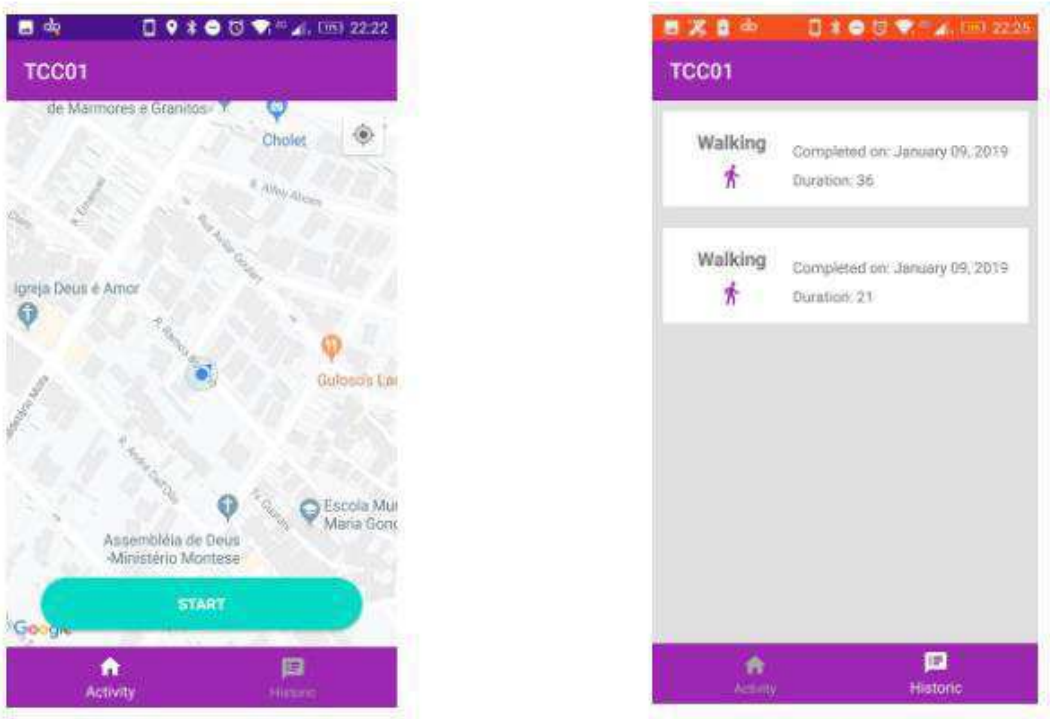
Fonte: Elaborado pelo autor.

A segunda tela a ser desenvolvida foi a Tela Principal. A partir dela, é possível iniciar uma atividade física e consultar o histórico de atividades, Figura 11. A tela de início de atividade mostra a posição do usuário no Google Maps[7] e tem como intuito iniciar uma

7 Google Maps SDK – https://developers.google.com/maps/documentation/android-sdk/intro?hl=pt-br. Acessado em 28 nov. 2019.

atividade física. A tela de histórico exibe todas as atividades salvas no Firebase realizadas pelo usuário ordenadas pela data.

Figura 11: Tela principal da versão Android.



(a) Tela Principal (b) Tela de Histórico.

Fonte: Elaborado pelo autor.

A terceira tela desenvolvida foi a de Atividade Física, provavelmente a tela mais importante e complexa de toda a aplicação. Nela o aplicativo se conecta ao dispositivo inteligente para a contagem de passos, rastreamento do usuário, cálculo da distância percorrida e cronometragem da atividade.

O rastreamento do usuário foi feito utilizando a interface LocationListener[8], que observa a mudança de posição do usuário; a cada mudança, a latitude e a longitude do usuário eram salvas num vetor chamado “locations”, que são salvos no banco de dados. Para o cálculo de distância, o vetor “locations” era percorrido e, a cada interação, a distância entre dois pontos era calculada usando a classe Location[9] e incrementada à distância total. No fim da atividade, o resultado da atividade é salvo no banco de dados, os dados salvos são: 1) duração,

8 Classe LocationListener – https://developer.android.com/reference/android/location/LocationListener. Acessado em 28 nov. 2019.

9 Classe Location – https://developer.android.com/reference/android/location/Location. Acessado em 28 nov. 2019.

2) data de realização da atividade, 3) localizações e 4) quantidade de passos.

Figura 12: Função executada ao usuário se movimentar.

```
override fun onLocationChanged(location: Location?) {
    if (mLocationPermissionGranted && location != null) {
        moveCamera(LatLng(location.latitude, location.longitude))
        if (started && isRunning) {
            locations.add(mapOf("latitude" to location.latitude, "longitude" to location.longitude))
            updateActivity(mapOf("locations" to locations))
            tvDistance.text = "${String.format("%.2f", calcDistance())} KM"
        }
    }
}
```

Fonte: Elaborado pelo autor.

Figura 13: Cálculo de distância.

```
private fun calcDistance(): Double {
    var distance = 0.0
    if (locations.size == 0) return distance

    for ((index, value) in locations.withIndex()) {
        if(index + 1 < locations.size ) {
            val start = Location( provider: "")
            val dest = Location( provider: "")

            start.latitude = value["latitude"] ?: error("")
            start.longitude = value["longitude"] ?: error("")

            dest.latitude = locations[index+1]["latitude"] ?: error("")
            dest.longitude = locations[index+1]["longitude"] ?: error("")


            distance += start.distanceTo(dest)
        }
    }

    Log.d( tag: "calcDistance", (distance / 1000).toString())
    return distance / 1000
}
```

Fonte: Elaborado pelo autor.

A quarta tela implementada foi o detalhe de atividades exibidas no histórico, e nessa tela são exibidos a data em que a atividade foi realizada, a quantidade de passos, o tempo total de atividade, a quantidade de calorias gastas e a rota do usuário. Um botão que leva à tela de informações gerais também foi adicionado. Para o cálculo de calorias, foi simulada a atividade de uma caminhada leve, para a qual são necessários a duração da atividade em minutos, a altura do usuário e o tempo da atividade e uma constante relacionada ao tipo de atividade.

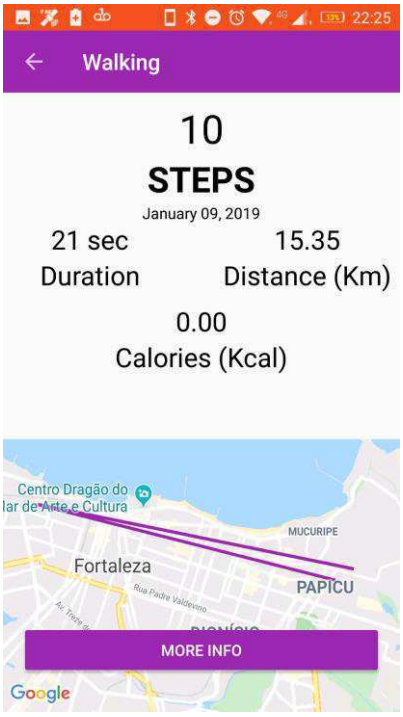
Figura 14: Cálculo de calorias.

```
private fun calcCalories(): Double {
    val durationInMinutes = physicalActivity.duration / 60
    val caloriesSpent = (0.5 * user!!.weight) * durationInMinutes

    return caloriesSpent
}
```

Fonte: Elaborada pelo autor.

Figura 15: Tela de detalhes do histórico de atividade.

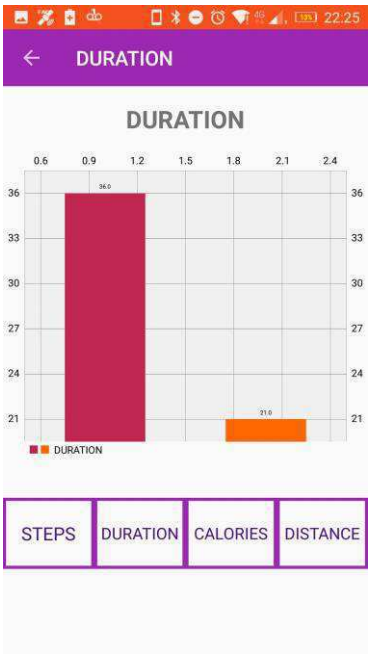


Fonte: Elaborado pelo autor.

Por fim, a última tela a ser implementada foi a de informações gerais das atividades. Ela é responsável por agregar todas as informações de todas as atividades e exibi-las através de gráficos que podem ser filtrados em quatro categorias diferentes: passos,

duração (da atividade), calorias gastas e distância percorrida.

Figura 16: Tela de informações gerais.



Fonte: Elaborado pelo autor.

4.1 Versão React Native

A versão do React Native foi criada usando o React Native CLI, e a linguagem escolhida foi o JavaScript com Flow[10], que "é um verificador de tipos estáticos para o seu código JavaScript", segundo o site do React Native, foi utilizado como IDE o Visual Studio Code[11]. Na versão React Native, foi seguida a mesma ordem de desenvolvimento da versão Android, tendo como primeiro passo a conexão com a Mi Band 2 tentando utilizar uma biblioteca proprietária ou de terceiros, mas novamente não foi obtido êxito na busca.

Para conexão Bluetooth, foi utilizada a biblioteca React Native BLE PLX pois não existe suporte do próprio React Native para esse tipo de conexão. Todos os UUID foram reaproveitados da versão Android, e uma classe chamada "BluetoothHandler", que utiliza o padrão de projeto Singleton, foi criada para gerenciar a conexão com o dispositivo.

10 Flow – https://pt-br.reactjs.org/docs/static-type-checking.html. Acessado em 28 nov. 2019.
11 VS Code – https://code.visualstudio.com/. Acessado em 28 nov. 2019.

Figura 17: Classe responsável pela comunicação Bluetooth

```
// classe responsável por se conectar a Mi Band 2
export default class BluetoothHandler {
    static instance: BluetoothHandler = null;
    manager: BleManager = null;
    device: ?Device = null;
    isConnected: boolean = false;

    constructor() {
        this.manager = new BleManager();
        this.startScan = this.startScan.bind(this);
        this.deviceConnect = this.deviceConnect.bind(this);
        this.stopScan = this.stopScan.bind(this);
        this.isDeviceConnected = this.isDeviceConnected.bind(this);
    }

    static getInstance() {
        if (BluetoothHandler.instance === null) {
            BluetoothHandler.instance = new BluetoothHandler();
        }

        return BluetoothHandler.instance
    }
```

Fonte: Elaborado pelo autor.

Figura 18: Conexão com dispositivo

```
async deviceConnect(device: Device) {
  const deviceConnected = await device.isConnected();

  if (deviceConnected) {
    await device.cancelConnection();
  }

  if (!deviceConnected) {
    device
      .connect()
      .then(device => {
        this.device = device;
        this.isConnected = true;
        return new Promise((resolve, reject) => {
          resolve(device);
        });
      })
      .catch(console.warn);
  }
}
```

Fonte: Elaborado pelo autor.

A segunda tela desenvolvida foi a tela de formulário de cadastro do usuário. As telas foram planejadas para serem semelhantes entre as versões de forma que não apenas o visual mas também a lógica fosse reaproveitada, como o cálculo de IMC por exemplo. Todos os dados do usuário são salvos no Firebase.

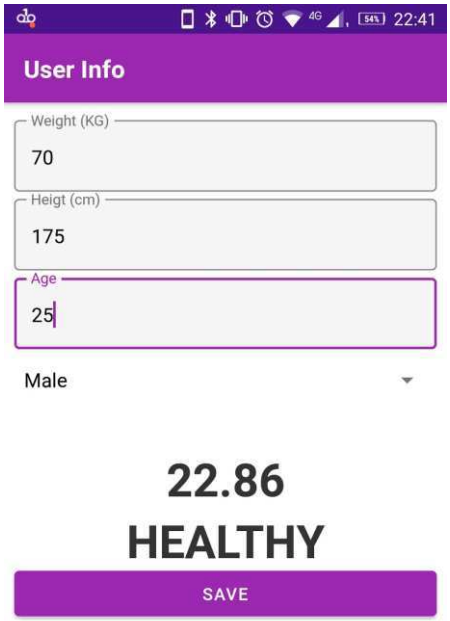
Figura 19: Cálculo do IMC.

```
calculateBMI = () => {
  const { weight, height } = this.state;
  const weightNumber = parseFloat(weight);
  const heightNumber = parseFloat(height) / 100;

  return (weightNumber / (heightNumber * heightNumber)).toFixed(2);
};
```

Fonte Elaborado pelo autor.

Figura 20: Formulário de cadastro React Native.



Fonte: Elaborado pelo autor.

A terceira tela a ser desenvolvida foi a tela principal, assim como no Android, possuindo as mesmas funcionalidades. Para exibição do mapa na versão RN, foi utilizado o componente React Native Maps, que é responsável por exibir o mapa relativo a cada plataforma, no caso do Android, o Google Maps e, no caso do iOS, o MapKit. É possível exibir o Google Maps no iOS, mas não o contrário.

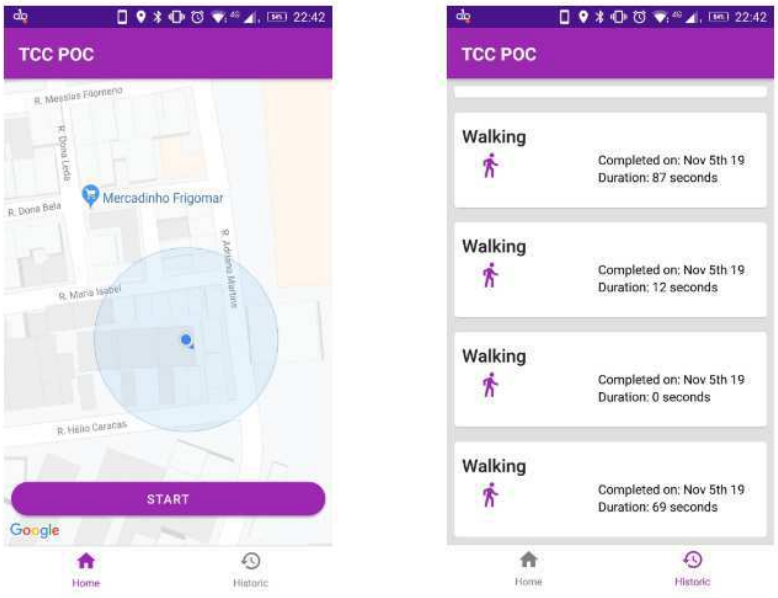
Figura 21: Componente responsável por exibir o mapa.

```
export default class Map extends PureComponent<P, S> {
    static defaultProps = {
        mapPos: {
            latitude: -3.74259555,
            longitude: -38.500003998674366,
            latitudeDelta: 0.001,
            longitudeDelta: 0.001,
            minZoomLevel: 15,
        },
        street: "TCC POC React Native",
        height: "100%",
        showsUserLocation: false,
        followsUserLocation: false,
        liteMode: false,
        markerColor: 'red',
    };

    render() {
        const { showsUserLocation, followsUserLocation, height, mapPos } = this.props;
        return (
            <MapView
                minZoomLevel={0}
                showsUserLocation={showsUserLocation}
                followsUserLocation={followsUserLocation}
                cacheEnabled
                loadingEnabled
                style={{
                    height,
                    width: '100%',
                }}
                region={mapPos}
            />
        );
    }
}
```

Fonte: Elaborado pelo autor.

Figura 22: Tela inicial da versão React Native.



(a) Tela de Início. (b) Tela de Histórico

Fonte: Elaborada pelo autor.

A quarta tela desenvolvida foi a de atividade física, novamente o design da tela foi mantido tendo como grande mudança o cálculo da distância. O vetor “locations”, que salva a latitude e a longitude, foi reutilizado e, para calcular a distância, foi utilizada a Fórmula de

Haversine, provida pela biblioteca Haversine[12]. Para obtenção de passos, foi utilizada a classe BluetoothHandler, que fornece o método para obtenção de passos.

Figura 23: Cálculo de distância.

```
calcDistance = () => {
  const { locations } = this.state;

  let distance = 0;
  if (locations.length > 0) {
    for (let i = 0; i < locations.length - 1; i++) {
      distance += haversine(locations[i], locations[i+1]);
    }

    return distance.toFixed(2);
  } else {
    return  distance;
  }
}
```

Fonte: Elaborado pelo autor.

Figura 24: Obtenção de passos.

```
readPedometer(callback) {
    const { device } = this;
    device.discoverAllServicesAndCharacteristics()
    .then(results => {
        results.readCharacteristicForService(
            CustomBluetoothProfile.basic.service,
            CustomBluetoothProfile.pedometer.characteristicSteps
        )
        .then((characteristic) => {
            if (characteristic.value) {
                const data = Buffer.from(characteristic.value, "base64");
                const steps: number = data.readUInt16LE(1);
                // console.warn(`Bluetooh.Step: ${steps}`);

                callback(steps);
            }
        });
    })
    .catch(error => console.warn(error));
}
```

Fonte: Elaborado pelo autor.

Na tela de detalhes da atividade física, foi reaproveitado o cálculo de calorias e, para a tela de informações gerais, quarta tela, foi feito o mesmo sistema de filtros do Android.

12 Haversine – https://github.com/njj/haversine. Acessado em 28 nov. 2019.

4.1 Avaliação Comparativa das Versões

Esta seção tem como objetivo listar de forma mais clara as principais diferenças entre a versão Android e a React Native. Na Tabela 2, é possível ter uma noção do tamanho do código gerado em cada PoC através das Linhas de Códigos, no inglês Line of Codes (LOC), e do Número de Métodos, no inglês Number of Functions (NOF). Enquanto na Tabela 3 é possível ter noção dos aplicativos gerados comparando suas apks, foi avaliado o tamanho da apk, a memória ocupada ao instalar essa apk e o tempo de inicialização.

Para contagem das LOC, foi utilizado o Cloc para a contagem dos dois projetos. No projeto Android, foi contabilizado um total de 1351 linhas de código em Kotlin, mais 1312 linhas de XML, totalizando 2663 linhas, 406 linhas em branco e 147 linhas de comentário, enquanto no projeto RN foi contabilizado um total de 2142 linhas, 234 linhas em branco e 213 linhas de comentários.

Tabela 2: Linhas de Códigos e Número de Funções de cada versão.

Versão	**LOC**
Android	2663
React Native	2142

Fonte: Elaborado pelo autor.

O tempo médio de inicialização foi calculado de forma manual seguido o seguinte procedimento: 1) Iniciação do aplicativo e, de forma simultânea, da contagem de tempo, 2) Espera do carregamento da tela de cadastro, 3) Pausa do cronômetro, 4) Registro do tempo corrido. O procedimento foi repetido 10 vezes para cada versão, e, no final de cada interação, o aplicativo era fechado na memória.

Tabela 3: Comparativo entre aplicativos

Versão	**Tamanho Apk (MB)**	**Tamanho instalado (MB)**	**Tempo Médio de inicialização**
Android	4	12,06	2,867s
React Native	24	47	4,092s

Fonte: Elaborado pelo autor.

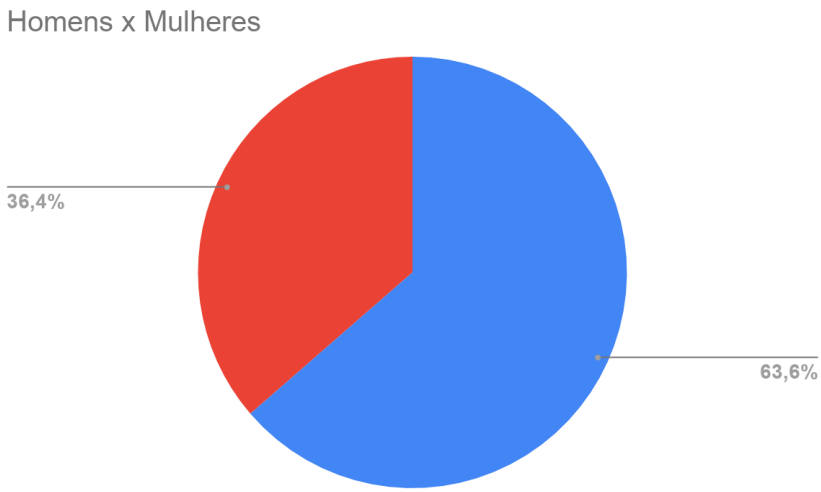
5 AVALIAÇÃO DE USABILIDADE

Neste capítulo é descrito como foi feita a avaliação com usuários, materiais e métodos utilizados, além de um breve resumo dos resultados.

5.1 Perfil dos participantes

A pesquisa foi realizada com estudantes do curso de Sistemas e Mídias Digitais e *por conveniência* foram selecionados alunos que se encontravam no Laboratório de Pesquisa e Desenvolvimento I tendo como pré-requisito falarem inglês básico e serem usuários Android. Foram entrevistados um total de 11 participantes, divididos em 7 homens e 4 mulheres com idades de 20 a 29 anos. Os participantes foram divididos em 2 grupos, 6 participantes avaliaram a versão Android e 5 usuários avaliaram a versão React Native. Os usuários não sabiam qual versão estavam avaliando.

Gráfico 1: Quantidade de homens e mulheres.



Fonte: Elaborado pelo autor.

5.2 Materiais e métodos

As atividades para o teste de usabilidade foram seguidos como os descritos em Barbosa e Silva (2010). Na fase de preparação, foram definidas as atividades realizadas pelo usuário durante a coleta de dados, assim como a preparação dos aplicativos e equipamentos.

As atividades propostas aos participantes foram: 1) Equipar a pulseira, 2) Criar o cadastro preenchendo todos os campos, 3) Navegar pelo app e iniciar uma atividade, 4) Parar a atividade e sair da tela de atividade, 5) Ir para a tela de histórico e selecionar uma das atividades do histórico, 6) Abrir tela de informações gerais, 7) Selecionar os diferentes filtros, 8) Fechar o aplicativo. O smartphone utilizado foi um Lenovo Vibe K6 com processador Snapdragon 430 Qualcomm SDM430, 2GB de memória RAM e com a versão do Android Nougat 7.0. A smart band utilizada foi uma Mi Band 2. Depois da preparação das atividades e equipamentos, foi realizado um teste piloto para cronometrar o tempo médio da coleta de dados e identificar possíveis problemas nos aplicativos que não foram identificados durante a fase de desenvolvimento.

5.3 Procedimento da avaliação

Cada participante realizou a avaliação de forma voluntária e teve de preencher um Termo de Consentimento antes de iniciar o procedimento. Após a realização das atividades, foi realizada a coleta de dados e para isso foi utilizado um questionário SUS (ANEXO A) e uma entrevista estruturada (APÊNDICE A), sendo a entrevista aplicada antes do questionário. A entrevista teve como objetivo identificar melhor o perfil dos usuários entrevistados e identificar a captação de possíveis problemas no aplicativo.

5.4 Resultados

Durante os testes, a versão React Native apresentou um problema e não voltou a funcionar, por isso o número de participantes é inferior à versão Android, que não apresentou falhas. Os usuários, de maneira geral, mostraram-se bastantes motivados a participar da pesquisa e a maioria deles, mesmo não sendo pedido, realizaram pequenas caminhadas para testar o aplicativo com a pulseira.

Foi constatado que mais da metade dos participantes, 63%, não realizam atividades físicas com frequências e que nenhum utilizou dispositivos vestíveis no auxílio de atividades físicas. Em contrapartida, 81% dos usuários já utilizaram, em algum momento, aplicativos para o auxílio de atividades físicas.

A pontuação do SUS vai de 0 a 100 de forma que quanto maior esse valor, melhor

o sistema que está sendo avaliado. Bangor, Kortum e Miller (2009) mostram que sistemas com uma nota maior que 70 são bons, maiores que 85 são excelentes e 100 é o melhor possível, dessa forma, a versão Android pode ser classificada como boa pois obteve uma média de 80,83 enquanto a versão React Native pode ser classificada como excelente pois obteve uma média de 91,5. A pontuação do SUS, diferentemente do que era esperado, teve como vencedora a versão React Native, que apresentou uma média superior à versão Android. Um problema comum às duas versões foi o ato de iniciar uma atividade física, e três usuários não entenderam com clareza como iniciar uma atividade física pelo aplicativo e precisaram do auxílio do pesquisador para completar a atividade.

Tabela 4: Síntese das respostas da entrevista.

Pergunta	**Android**	**React Native**	**Comentários Android**	**Comentários RN**
1	66.66% não praticam	60% não praticam	Nenhum	Nenhum
2	100% nunca utilizaram	100% nunca utilizaram	Nenhum	Nenhum
3	83% já usaram	80% já usaram	Nenhum	Nenhum
4	100% são experientes	80% são experientes	Nenhum	Nenhum
5	50% que não	60% que não	Crash na versão React, “Travamento na transição de abas”	“Demora na contagem de passos”
6	83% que sim	100% que sim	“Estranhou os gráficos, nunca tinha visto gráfico parecido em aplicações android”	Nenhum
7	100% que sim	80% que sim	“Gráficos confusos”	“Gráficos, confuso ao ler o gráfico.”

	“Botão de start sem contraste”	"Confuso ao iniciar a atividade"
	“Tempo no histórico incorreto”	"Bug visual no gráfico"
	“Não entendeu de cara qual atividade iniciar”	
	“Contraste das abas, parece que a aba não selecionada está desativada”	
	“Na tela de mais informações não ficou claro de quais informações se tratava”	

Fonte: Elaborado pelo autor.

6 DISCUSSÃO

A partir da pergunta definida no início deste trabalho, “é possível alcançar o mesmo nível de usabilidade de um aplicativo nativo utilizando uma CPT?”, é possível dizer que, sim, é possível alcançar o mesmo nível de usabilidade de um aplicativo nativo através de uma CPT diferentemente do que apontam Angulo e Ferre (2014), porém faltou a comparação na plataforma iOS, e a pesquisa foi realizada com um número muito pequeno de participantes. Rieger e Majchrzak (2019) propõem um framework para avaliação de CPT e nele listam diversos critérios de avaliação que são também avaliados neste trabalho, que são: 1) Look and feel em que elementos que são nativos devem apresentar comportamentos comuns da plataforma; 2) Dispositivos conectados representam o suporte à conexão de dispositivos externos como sensores ou dispositivos vestíveis.

As Tabela 2 e 3 mostram as diferenças entre as duas versões, as LOC da versão JavaScript são superiores à da versão Kotlin por causa do JSX que estão no mesmo arquivo da lógica. A disparidade no tamanho das APK se deve ao grande número de dependências que o React Native utiliza e que impacta diretamente o tempo de inicialização pois é necessário carregar o JavaScript e o próprio aplicativo. Para contornar esse problema de inicialização, é recomendado pela equipe React Native o uso do Hermes, “uma *engine* JavaScript otimizada para rodar aplicativos JavaScript no Android” (HERMES, 2019). Não é informado se o Hermes é habilitado por padrão no iOS.

A Tabela 4 apresenta o resumo da entrevista e nela é possível observar que a maioria dos participantes não é praticante de exercícios físicos e nunca utilizou nenhum tipo de dispositivo para realizar atividades físicas, fazendo com que o aplicativo mostrado durante os testes não fosse do interesse dos participantes.

O fato de a média do SUS ser diferente se deve provavelmente ao fato de os participantes acharem os gráficos da tela de Mais Informações confusos, 66% dos participantes da versão Android acharam isso, e a demora para atualização dos passos enquanto uma atividade era realizada, apenas 40% dos usuários da versão RN reclamaram do entendimento dos gráficos.

6.1 Dificuldades de Programação

Não houve muita dificuldade para implementação das funcionalidades propostas para a POC com exceção da conexão com o dispositivo externo, que foi um desafio em ambas

as versões devido à falta de documentação da Mi Band 2 e que foi alcançada graças a pesquisas na internet. Devido a essas dificuldades e ao limite de tempo do projeto, não foi possível implementar o serviço de frequência cardíaca na versão React Native fazendo com que essa funcionalidade fosse removida da versão Android com o intuito de deixar as versões o mais parecidas possível.

6.2 Ameaças à Validade

Durante os testes, os participantes não usaram os seus *smartphones*, o que pode ou não ter influenciado a percepção dos usuários; pontos mostrados na Tabela 3 como o tamanho do aplicativo instalado e o tempo de inicialização não foram levados em consideração. Os usuários também não chegaram a realizar uma atividade física de fato, além de utilizarem o aplicativo por pouco tempo. Outro ponto a ser considerado é que as aplicações foram desenvolvidas por uma única pessoa, o que levou a replicação de erros como o da tela de informações ou a demora na implementação na conexão com a smartband, que acabou atrapalhando todo o projeto.

6.3 Trabalhos Relacionados

A Tabela 5 lista os trabalhos relacionados mostrando os *frameworks* avaliados e o foco de cada um desses trabalhos. Este trabalho se assemelha ao de Angulo e Ferre (2014) por comparar a UX, porém a versão nativa iOS não foi avaliada. É desenvolvida uma POC como em Sommer e Krusche (2013), porém a POC desenvolvida acessa diversos recursos do smartphone e se conecta a um dispositivo externo.

Tabela 5 – Trabalhos relacionados ao desenvolvimento de aplicativos utilizando CPT.

Ano	**Autor**	***Frameworks* avaliados**	**Foco**
2017	Majchrzak e Grønli (2017)	React Native, Ionic Framework e Fuse	Desenvolve uma PoC utilizando os três frameworks mas não mede

			o desempenho dos aplicativos
2016	Oliveira et al. (2016)	Não se aplica	Analisa o consumo de energia de aplicativos nativos e híbridos obtidos a partir do F-Droid[13]
2015	Willcox et al. (2015)	Xamarin e Phonegap	Comparar o desempenho de aplicativos nativos e gerados por CPT mas não desenvolve uma PoC
2013	Sommer e Krusche (2013)	Phonegap, Rhodes e Titanium	Desenvolve uma PoC utilizando os três *frameworks* que basicamente realiza chamadas de Web Services mas não mede o desempenho dos aplicativos
2013	Xanthopoulos e Xinogalos (2013)	Titanium	Avalia a processo de desenvolvimento de um aplicativo de *feeder* RSS[14] mobile utilizando CPT
2014	Angulo e Ferre (2014)	Titanium	Desenvolve um app em Titatanium e compara a UX da versão nativa Android e iOS

Fonte: Elaborada pelo autor.

13 F-Droid – https://f-droid.org/. Acessado em 28 nov. 2019.
14 RSS – https://pt.wikipedia.org/wiki/RSS. Acessado em 28 nov. 2019.

7 CONCLUSÃO

Este trabalho analisou uma aplicação PoC de atividade física que possuía duas versões, uma desenvolvida utilizando CPT e uma desenvolvida de forma nativa. Essa aplicação acessa sensores, recursos multimídia e se conecta a um dispositivo externo via Bluetooth. Foi desenvolvido com foco em uma única plataforma móvel, Android, permitindo que os padrões de interação dessa plataforma fossem mais facilmente alcançados.

Buscou-se saber se os aplicativos gerado por CPT conseguiam ter o mesmo nível, ou próximo, de usabilidade de um aplicativo nativo comparando-os utilizando o SUS. O resultado não só foi alcançado como superou a aplicação nativa apresentando uma média no SUS superior e contrariando o que é comumente encontrado na literatura.

Como trabalho futuro, pretende-se testar a PoC no iOS, e para isso é necessário: implementar a versão nativa do iOS, além de testar a versão do RN no iOS; utilizar todos os sensores da Mi Band 2; testar com o público-alvo.

REFERÊNCIAS

ANGULO, Esteban; FERRE, Xavier. A case study on cross-platform development frameworks for mobile applications and UX. In: **Proceedings of the XV International Conference on Human Computer Interaction**. ACM, 2014. p. 27

BARBOSA, Simone Diniz Junqueira; SILVA, Bruno Santana da. **Interação Humano-Computador**. Rio de Janeiro: Elsevier Editora Ltda., 2010.

BANGOR, Aaron; KORTUM, Philip; MILLER, James. Determining What Individual SUS Scores Mean: Adding an Adjective Rating Scale. **Journal Of Usability Studies**, [s. L.], v. 4, n. 3, p.114-123, maio 2009. Disponível em: <http://uxpajournal.org/wp-content/uploads/sites/8/pdf/JUS_Bangor_May2009.pdf>. Acesso em: 29 nov. 2019.

BERNARDES, Tatiana Freitas; MIYAKE, Mario Yoshikazu. Cross-platform Mobile Development Approaches: A Systematic Review. **Ieee Latin America Transactions**, [s.l.], v. 14, n. 4, p.1892-1898, abr. 2016. Institute of Electrical and Electronics Engineers (IEEE). GIL, Antonio C. **Como Elaborar Projetos de Pesquisa**. São Paulo: Atlas 1991.

CONSOLVO, Sunny et al. Design requirements for technologies that encourage physical activity. **Proceedings Of The Sigchi Conference On Human Factors In Computing Systems - Chi '06**, [s.l.], p.457-466, 2006. ACM Press.

CORPORATION, International Data. **Smartphone OS Market Share, 2017 Q1**. 2017. Disponível em: <https://www.idc.com/promo/smartphone-market-share/os>. Acesso em: 26 nov. 2017.

DHILLON, Sunny; MAHMOUD, Qusay H.. An evaluation framework for cross-platform mobile application development tools. **Software: Practice and Experience**, [s.l.], v. 45, n. 10, p.1331-1357, 27 ago. 2014. Wiley-Blackwell.

EL-KASSAS, Wafaa S. et al. Taxonomy of Cross-Platform Mobile Applications Development Approaches. **Ain Shams Engineering Journal**, [s.l.], v. 8, n. 2, p.163-190, jun. 2017. Elsevier BV.

ENERGY. **UX Test SUS**. Disponível em: <https://www.energy.gov/sites/prod/files/2015/09/f26/ux_test_sus.docx>. Acesso em: 16 nov. 2019.

FACEBOOK. **React – Uma biblioteca JavaScript para criar interfaces de usuário**. 2019. Disponível em: <https://pt-br.reactjs.org>. Acesso em: 16 nov. 2019.

FACEBOOK. **React Native · A framework for building native apps using React**. 2019. Disponível em: <https://facebook.github.io/react-native/>. Acesso em: 15 nov. 2019.

GARTNER. **Gartner Says Worldwide Sales of Smartphones Grew 7 Percent in the Fourth Quarter of 2016**. Disponível em: <http://www.gartner.com/newsroom/id/3609817>. Acesso em: 16 nov. 2017.

GIL, Antonio C. **Como Elaborar Projetos de Pesquisa**. São Paulo: Atlas 1991.

GREGOSKI, Mathew J. et al. Development and Validation of a Smartphone Heart Rate Acquisition Application for Health Promotion and Wellness Telehealth Applications. **International Journal Of Telemedicine And Applications**, [s.l.], v. 2012, p.1-7, 2012. Hindawi Limited.

HERMES. **Hermes - JavaScript engine optimized for React Native**. Disponível em: <https://hermesengine.dev/>. Acesso em: 24 nov. 2019.

LECHETA, Ricardo R.. **Desenvolvendo para iPhone e iPad: Aprenda a desenvolver aplicativos utilizando iOS SDK**. 3. ed. São Paulo: Novatec, 2017, p. 23-36.

LECHETA, Ricardo R.. **Google Android: Aprenda a criar aplicações para dispositivos móveis com o Android SDK**. 3. ed. São Paulo: Novatec, 2013, p. 21-30.

LEIJDEKKERS, Peter; GAY, Valerie. Mobile apps for chronic disease management: lessons learned from myFitnessCompanion®. **Health And Technology**, [s.l.], v. 3, n. 2, p.111-118, 15 fev. 2013. Springer Nature.

MAJCHRZAK, Tim; GRØNLI, Tor-morten. Comprehensive Analysis of Innovative Cross-Platform App Development Frameworks. **Proceedings Of The 50th Hawaii International Conference On System Sciences (2017)**, [s.l.], p.6162-6171, 2017. Hawaii International Conference on System Sciences.

OLIVEIRA, Wellington et al. Native or Web? A Preliminary Study on the Energy Consumption of Android Development Models. **2016 IEEE 23rd International Conference On Software Analysis, Evolution, And Reengineering (saner)**, [s.l.], p.589-593, mar. 2016.

RABIN, Carolyn; BOCK, Beth. Desired Features of Smartphone Applications Promoting

Physical Activity. **Telemedicine And E-health**, [s.l.], v. 17, n. 10, p.801-803, dez. 2011. Mary Ann Liebert Inc.

RIEGER, Christoph; MAJCHRZAK, Tim A.. Towards the definitive evaluation framework for cross-platform app development approaches. **Journal Of Systems And Software**, [s.l.], v. 153, p.175-199, jul. 2019. Elsevier BV. http://dx.doi.org/10.1016/j.jss.2019.04.001.

SCHOBEL, Johannes et al. Using Vital Sensors in Mobile Healthcare Business Applications Challenges, Examples, Lessons Learned. **9th Int'l Conference On Web Information Systems And Technologies**. Alemanha, Aachen, p. 16-25. maio 2013.

SOMMER, Andreas; KRUSCHE, Stephan. Evaluation of cross-platform frameworks for mobile applications. **In Proceedings Of The 1st European Workshop On Mobile Engineering**, [s.l], v. 1, n. 1, p.363-376, 2013. Anual.

WILLOCX, Michiel; VOSSAERT, Jan; NAESSENS, Vincent. A Quantitative Assessment of Performance in Mobile App Development Tools. **2015 IEEE International Conference On Mobile Services**, [s.l.], p.454-461, jun. 2015. IEEE.

APÊNDICE A – INSTRUMENTO DE COLETA DE DADOS

ENTREVISTA ESTRUTURADA

1. **Você pratica exercícios físicos com frequência?**

2. **Você já praticou exercícios físicos com o auxílio de dispositivos?**

3. **Você já utilizou algum aplicativo no auxílio de atividades físicas?**

4. **Você se considera um usuário experiente do Android?**

5. **O app apresentou travamentos?**

6. **O app apresentou comportamentos comuns ao Android?**

7. **Alguma interação com o app lhe causou estranheza?**

APÊNDICE B – TERMO DE CONSENTIMENTO

Eu, PAULO ROGÉRIO MATHEUS BONFIM LEITÃO, aluno do curso de Graduação em Sistema e Mídias Digitais da Universidade Federal do Ceará, gostaria de convidá-lo(a) para participar desta pesquisa que objetiva avaliar um aplicativo de prova de conceito, denominado apenas como POC.

Para o aplicativo POC, a proposta é monitorar atividades físicas através do o aplicativo com o auxílio de um dispositivo vestível.

O (a) senhor(a) irá acessar o aplicativo a ser analisado, assim que concordar com este documento. Sua participação é importante, porém você não deve participar contra a sua vontade.

Leia atentamente as informações abaixo e faça qualquer pergunta que desejar, para que todos os procedimentos desta pesquisa sejam esclarecidos.

Será mantido sigilo em relação ao seu nome e/ou quaisquer outros aspectos que possam vir a identifica-lo(a). As informações utilizadas nesta pesquisa possuirão a única finalidade de colaborar com o trabalho de conclusão de curso.

Não será oferecido nenhum valor ao (a) senhor (a) para participar desse estudo. Portanto, nesta pesquisa, sua participação é totalmente voluntária.

O tempo estimado por sessão é de aproximadamente 25 minutos, dividido em duas etapas. Em um primeiro momento serão propostas atividades no aplicativo e depois você irá responder um formulário e uma entrevista.

Após interagir com o aplicativo, será respondido um formulário e uma entrevista que serão de grande importância para o avanço da pesquisa.

Caso possua alguma dúvida, entre em contato comigo pelo telefone: (85) 99936 1580, ou no meu endereço: Rua Hélio Caracas 26 – Apto 301 – Bairro Jacarecanga. CEP: 60010-430

O Comitê de Ética em Pesquisa da Universidade Federal do Ceará encontra-se disponível para esclarecer dúvidas e/ou reclamações em relação à sua participação no referido estudo: Rua Coronel Nunes de Melo, 1127 – Rodolfo Teófilo, telefone: (85) 33668344.

Espero poder contar com sua inestimável cooperação e desde já agradeço pela atenção.

Paulo Rogério Matheus Bonfim Leitão

TERMO DE CONSENTIMENTO LIVRE E ESCLARECIDO

Eu,

_____,

RG/CPF:_____, declaro que tomei conhecimento do estudo **ESPECIFICAÇÃO E DESENVOLVIMENTO DE SOFTWARES EDUCATIVOS EM DISPOSITIVOS MÓVEIS PARA A PRÁTICA DE EXECÍCIOS FÍSICOS**, realizado pelo pesquisador Paulo Rogério Matheus Bonfim Leitão, aluno do curso de Graduação em Sistemas e Mídias Digitais – Instituto UFC Virtual da Universidade Federal do Ceará.

Compreendi perfeitamente tudo que me foi informado sobre minha participação no mencionado estudo e estando consciente dos meus direitos, das minhas responsabilidades, dos riscos e dos benefícios que minha participação implica.

Concordo em dele participar e dou meu consentimento sem que para isso tenha sido forçado(a) ou obrigado(a), participando voluntariamente dessa pesquisa.

Todas as informações que citarei são verdadeiras, podendo ser juntadas às outras que compõem esta pesquisa. Atesto também que recebi uma cópia assinada deste termo.

Fortaleza, _____ de _____ de 2019.

_____

Assinatura do Voluntário

_____

Assinatura do Ledor

_____

Assinatura do Pesquisador

ANEXO A – SYSTEM USABILITY SCALE (SUS)

Nome do participante: _____

System Usability Scale (SUS)

Este é um questionário padrão que mede a usabilidade geral de um sistema. Selecione a resposta que melhor expressa como você se sente sobre cada afirmação depois de usar o site hoje.

	Discordo fortemente				Concordo plenamente
1. Eu acho que gostaria de usar esse sistema com frequência.	☐	☐	☐	☐	☐
2. Eu acho o sistema desnecessariamente complexo.	☐	☐	☐	☐	☐
3. Eu achei o sistema fácil de usar.	☐	☐	☐	☐	☐
4. Eu acho que precisaria de ajuda de uma pessoa com conhecimentos técnicos para usar o sistema.	☐	☐	☐	☐	☐
5. Eu acho que as várias funções do sistema estão muito bem integradas.	☐	☐	☐	☐	☐
6. Eu acho que o sistema apresenta muita inconsistência.	☐	☐	☐	☐	☐
7. Eu imagino que as pessoas aprenderão como usar esse sistema rapidamente.	☐	☐	☐	☐	☐
8. Eu achei o sistema atrapalhado de usar.	☐	☐	☐	☐	☐
9. Eu me senti confiante ao usar o sistema.	☐	☐	☐	☐	☐
10. Eu precisei aprender várias coisas novas antes de conseguir usar o sistema.	☐	☐	☐	☐	☐

Fonte: (ENERGY, 2019) e traduzido pelo autor.