IPS Instituto Politécnico de Setúbal

Escola Superior de Tecnologia de Setúbal

Pedro Filipe Fernandes da Costa

KBIT: mHealth Framework

PROJETO DE MESTRADO E PÓS-GRADUAÇÃO

Framework para aquisição distribuída de biossinais e aplicação Móvel para Dinamometria

Trabalho de Projeto submetida como requisito parcial para obtenção do grau de **Mestre em Engenharia de Software**

Júri

Presidente (Doutor, Cláudio Miguel Garcia Loureiro dos Santos Sapateiro, Escola Superior Tecnologia de Setúbal)

Orientador (Doutor, Hugo Humberto Plácido da Silva, IT – Instituto de Telecomunicações)

Vogal (Doutor, Ivan Miguel Serrano Pires, Escola Superior de Tecnologia e Gestão de Viseu)

11 de Fevereiro de 2021

Agradecimentos

Em primeiro lugar gostaria de agradecer ao Instituto de Telecomunicações (IT) e ao Professor Hugo Silva por todo o apoio, supervisão e disponibilidade durante todas as etapas do projeto, bem como pelo material facultado para o desenvolvimento do projeto.

Gostaria de agradecer também ao Luís Marques e ao Prof. João Valente, da empresa BrainAnswer, pelo rápido apoio e esclarecimento de dúvidas acerca do funcionamento da plataforma.

Resumo

Até 2050, a população mundial com 60 anos ou mais deve totalizar 2000 milhões, acima dos 900 milhões em 2015. Associado ao envelhecimento da população está a fragilidade, sendo que esta é considerada altamente predominante com o aumento da idade e acarreta riscos elevados relacionados com o declínio do estado de saúde, incluindo mortalidade, quedas e hospitalização. Estudos recentes demonstram que a força de preensão é um parâmetro que reflete a resistência muscular em pessoas idosas em diferentes condições clínicas. A dinamometria é um dos processos possíveis para obtenção de medidas quantificáveis da força muscular, potência e/ou resistência. Com este trabalho tenta-se obter uma solução de forma a que a esta população mais envelhecida tenha um maior e mais fácil acompanhamento com uma alternativa de baixo custo para avaliação da fragilidade por via de dinamometria, com tecnologias comummente presente no nosso dia a dia. Descreve-se a criação de uma *framework* que permite o rápido e fácil desenvolvimento para ambiente móvel, focando-se na incorporação de sensores para medição da força de preensão e com potencial para medição de outros sinais fisiológicos. Esta possibilita a aquisição, processamento, visualização, gravação e comunicação dos dados em tempo real para plataformas *cloud*. Com a capacidade das tecnologias *cloud*, estes dados são depois integrados com outras ferramentas. Foram realizados ensaios às ferramentas de desenho de gráficos e analisada a usabilidade de várias opções de layout da aplicação, tendo-se adotado a solução que proporciona melhor desempenho. Como prova de conceito e foco, realizou-se um estudo mais aprofundado ligado à dinamometria, fazendo a monitorização da força de preensão do utilizador recorrendo a um dinamómetro ligado a um dispositivo BITalino sendo os dados registados no smartphone. Assim é possível obter uma leitura em tempo real do teste realizado no smartphone guardar o mesmo e, posteriormente, analisar estes dados.

Palavras-chave: Framework, Smartphone, Dinamometria, mHealth, Android.

Abstract

By 2050, the world population aged 60 and over is expected to total 2000 million, up from 900 million in 2015. Associated with population aging is fragility, which is considered highly prevalent with increasing age and entails high risks related to declining health status, including mortality, falls and hospitalization. Recent studies show that grip strength is a parameter that reflects muscle resistance in elderly people in different clinical conditions. Dynamometry is one of the possible processes for obtaining quantifiable measures of muscle strength, power and/or resistance. With this work we try to obtain a solution so that this older population has a greater and easier follow-up with a low-cost alternative for the evaluation of fragility by dynamometry, with technologies commonly present in our daily lives. It describes the creation of a framework that allows the fast and easy development for mobile environment, focusing on the incorporation of sensors for measuring grip force and with potential for measuring other physiological signals. This allows the acquisition, processing, visualization, recording and communication of data in real time for cloud platforms. With the capability of cloud technologies, this data is then integrated with other tools. Tests were performed on the graph design tools and the usability of various layout options of the application was analyzed, adopting the solution that provides the best performance. As a proof of concept and focus, a more in-depth study was carried out linked to dynamometry, monitoring the user's grip force using a dynamometer connected to a BITalino device and the data recorded on the smartphone. This way it is possible to obtain a real time reading of the test performed on the smartphone, save it, and later analyze this data.

Keywords: Framework, Smartphone, Dynamometric, mHealth, Android.

Índice

Lista de Figuras

Lista de Tabelas

Lista de Siglas e Acrónimos

AAL	Ambient Assisted Living
API	Application Programming Interface
CAALYX	Complete Ambient Assisted Living Experiment
ECG	Electrocardiogram
CSS	Cascading Style Sheets
D3.js	Data-Driven Documents
DOM	Document Object Model
DPL	Data Processing Language
eCAALYX	Enhanced Complete Ambient Assisted Living Experiment
FP6	Sixth Framework Programme
GB	Gigabyte
HTML	Hypertext Markup Language
Hz	Hertz
IoT	Internet of Things
JSON	Javascript Object Notation
JVM	Java Virtual Machine
LCD	Liquid-Crystal Display
LED	Light-Emitting Diode
MAC	Media Access Control
mHealth	Mobile Health

microFET	Micro Force Evaluation and Testing
mmHg	Millimetre of Mercury
MVC	Model-View-Controller
MVP	Model-View-Presenter
RAM	Random-Access Memory
REST	Representational State Transfer
SDK	Software Development Kit
SVG	Scalable Vector Graphics
UI	User Interface
UML	Unified Modeling Language
URL	Uniform Resource Locator
WFM	Workflow Manager
XML	Extensible Markup Language

Capítulo 1
Introdução

Neste capítulo são resumidas as motivações e principais áreas de enquadramento do presente trabalho. É descrita a motivação que levou à realização do mesmo e feita introdução e explicação sobre a dinamometria e mHealth.

1.1. Motivação

Em todo o mundo a esperança média de vida é mais longa, e o envelhecimento da população é cada vez maior. Novas proteções sociais, melhorias na qualidade dos cuidados de saúde e melhores condições de vida e trabalho, levam as pessoas a viver mais do que no passado. Pela primeira vez na história, a maioria das pessoas pode esperar viver até os sessenta anos e além, e a velhice está-se a tornar uma realidade para um número cada vez maior de pessoas. Juntamente com a diminuição da mortalidade, o declínio nas taxas de fertilidade é um fator preponderante no envelhecimento da população.

Até 2050, a população mundial com 60 anos ou mais deve totalizar 2000 milhões (Figura 1), acima dos 900 milhões em 2015. Atualmente, 125 milhões de pessoas têm 80 anos ou mais, e até 2050, 80% de todos os idosos viverão em países com baixos e médios rendimentos. Embora a mudança na distribuição da população de um país para as idades mais avançadas - conhecida como envelhecimento da população - tenha começado em países de elevados rendimentos (por exemplo, no Japão, onde 30% da população já tem mais de 60 anos), agora são os países com rendimentos baixos e médios que estão a passar pela maior mudança. As expectativas dos idosos estão a mudar juntamente com a crescente esperança de vida, no entanto, há pouca evidência que sugira que as pessoas mais velhas hoje passem os últimos anos com uma saúde melhor do que a dos seus pais. Embora as taxas de incapacidade severa tenham diminuído nos países desenvolvidos nos últimos trinta anos, não existiram mudanças significativas na incapacidade leve a moderada no mesmo período [1].

Associado ao envelhecimento da população está a fragilidade, sendo que esta é considerada altamente predominante com o aumento da idade e acarreta riscos elevados

relacionados com o declínio do estado de saúde, incluindo mortalidade, quedas e hospitalização. Uma definição clínica de fragilidade de acordo com Fried [2] inclui exaustão, atividade física reduzida, velocidade de caminhada lenta, força de preensão reduzida e perda de peso não intencional.

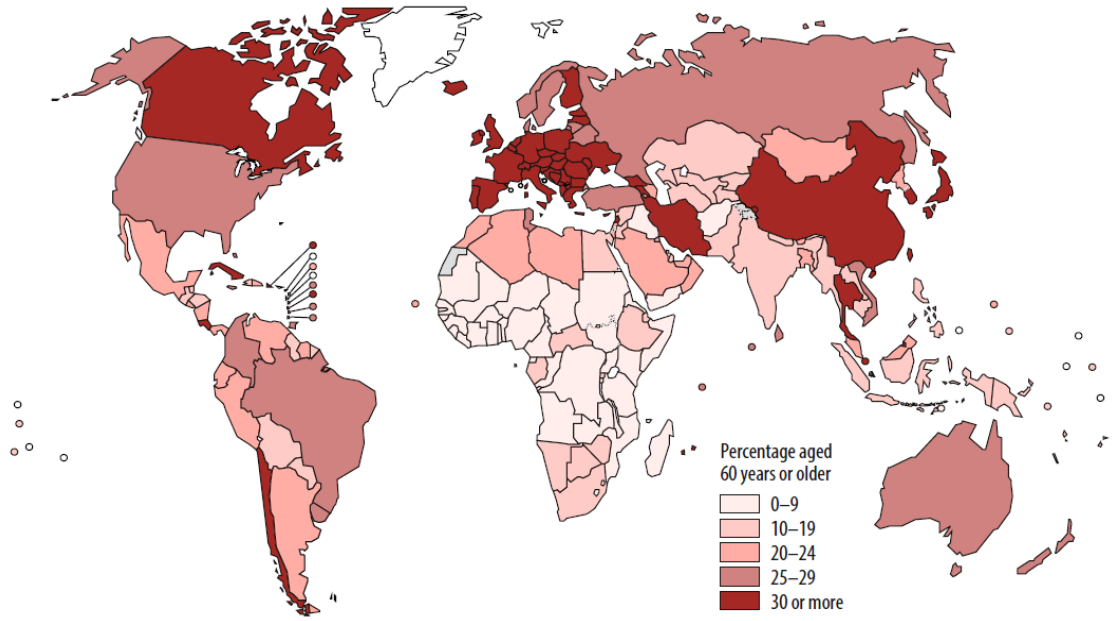

Figura 1-1 - Proporção da população com 60 anos de idade ou mais, por país (projeções para 2050) [1].

1.2. Dinamometria

A avaliação da força de preensão, em particular a medição da força voluntária máxima da mão, é um dos métodos mais simples para avaliação da função muscular [2], tendo já sido demonstrada como uma medida válida, confiável e viável em várias populações. Esta faz a mediação da força muscular geral dos membros superiores [3], mas pode ser correlacionada com a força dos membros inferiores [4]. Além disso, esta técnica demonstrou ser uma forma de triagem confiável, rápida e válida na avaliação do risco de fragilidade em internamento hospitalar. Para mais, este é um marcador importante na avaliação da diminuição drástica da massa muscular e da força (um fenômeno chamado sarcopenia) [4], estado nutricional [5] e fragilidade [6].

A força de preensão é ainda um preditor de mortalidade prematura, início precoce da incapacidade, complicações pós-operatórias, aumento do tempo de internamento [3], fraturas e declínio cognitivo em idosos [7]. Além disso, dados da literatura tendem a

apoiar o fato de que a força de preensão pode ser um bom preditor de depleção de massa celular, diminuição funcional durante a hospitalização [2], complicações pós-operatórias e mortalidade [3]. Essa previsão é importante para a identificação de indivíduos que correm o risco de eventos futuros indesejáveis e para determinar metas apropriadas para os esforços de redução de riscos [8]. Portanto, pode ser valioso introduzir essa medida como marcador de sarcopenia, estado nutricional, fragilidade e aptidão física. Devido à sarcopenia, pessoas idosas ou doentes funcionarão mais perto do limite de força máxima [9].

I. Bautmans [9] desenvolveu um estudo com o objetivo de validar a estimativa de resistência muscular e examinar as relações com dependência e inflamação em idosos com uma população de 91 pacientes idosos (83 ± 5 anos), 100 pacientes de controlo idosos (74 ± 5 anos) e 100 pacientes de controlo jovens (23 ± 3 anos). A força de preensão foi medida continuamente mantendo a força máxima até a exaustão, onde a fadiga era expressada quando a força de preensão estava abaixo dos 50% do máximo. Com isto chegou-se à conclusão de que a força de preensão é um parâmetro que reflete a resistência muscular em pessoas idosas em diferentes condições clínicas. A dinamometria é um dos processos possíveis para obtenção de medidas quantificáveis da força muscular, potência e/ou resistência.

1.3. mHealth

mHealth é a área que explora a saúde em contexto de mobilidade como prática médica e de saúde pública suportada por dispositivos móveis, como smartphones, dispositivos de monitorização de pacientes, assistentes digitais pessoais e outros dispositivos sem fios [10]. Este é um componente do eHealth, que é um campo emergente, desde cerca de 2000, na interseção de informática médica, saúde pública e negócios, referindo-se a serviços de saúde e informações fornecidas ou aprimoradas pela Internet e tecnologias relacionadas [11].

Entre 2013 e 2014, nas economias emergentes e em desenvolvimento, cerca de um quarto das pessoas já possuem smartphone, ou seja, um dispositivo móvel com capacidade de aceder à Internet e a aplicações. Já em 2017, a quantidade de pessoas com um smartphone subiu para cerca de 40%, sendo que nas economias avançadas este valor ultrapassa atualmente os 70%. Apesar de grande parte destes utilizadores serem de gerações mais novas, as gerações mais idosas tendem cada vez mais a dispor de

smartphones e, portanto, a fazerem parte desta percentagem [12]. Em países em desenvolvimento, o uso de tecnologias de eHealth juntamente com o uso de smartphones para apoiar intervenções de saúde e saúde pública, notadamente recolha de dados para investigação em saúde, tem vindo a demonstrar de forma convincente que estes dispositivos podem ser muito eficazes para melhorar o tempo e a qualidade na recolha [13]. Num outro estudo, também em países em desenvolvimento, foi destacado o uso bem sucedido de smartphones para apoiar os cuidados com a saúde remota e a telemedicina, facilitando a interação entre o paciente e o prestador, de forma a prestar um diagnóstico médico fora do ambiente clínico [16, 17].

Portanto, o potencial do uso de dispositivos móveis, como forma de transformar a assistência médica e a intervenção clínica na comunidade é tremendo.

1.4. Objetivos

Com este trabalho pretendeu-se desenvolver uma *framework* que permite o desenvolvimento rápido e fácil de aplicações para dispositivos móveis, com foco particular na aquisição de dados a partir de sensores para medição de sinais fisiológicos e dinamometria. O principal objetivo foi proporcionar uma metodologia *no-code*, propondo-se nesta tese a construção de aplicações baseadas em blocos, de forma a obter uma maior abstração permitindo assim alterar o comportamento da aplicação consoante as necessidades do utilizador.

Com a abordagem proposta, a estruturação da aplicação pode ser feita com base num ficheiro descritor, permitindo assim adaptar a aplicação para diferentes alternativas. Desta forma é obtida uma solução de mHealh especialmente desenvolvida de forma a poder proporcionar à população mais envelhecida uma solução para que esta tenha um maior e mais fácil acompanhamento, permitindo obter um historial e acompanhamento remoto do utente. Com isto a solução apresentada é uma alternativa de baixo custo para a avaliação da fragilidade por via de dinamometria, utilizando tecnologias comummente presentes no nosso dia a dia.

1.5. Estrutura do Documento

O documento está organizado em sete capítulos. O primeiro capítulo é dedicado a uma introdução ao problema, enquadramento do projeto com os principais temas, quais

as lacunas encontradas e como as tentar resolver. No segundo capítulo é feito um resumo do estado da tecnologia e ferramentas utilizadas atualmente. No terceiro capítulo são indicadas as tecnologias e ferramentas utilizadas para o desenvolvimento da aplicação. No quarto capítulo é apresentada uma visão geral de como foi desenvolvida e arquitetura utilizada na aplicação, bem como os principais componentes da aplicação. No quinto capítulo são demonstrados os testes realizados durante o desenvolvimento, de forma a obter a ferramenta e metodologia mais indicada para o projeto. No sexto capítulo é explicada a implementação do projeto, como as diferentes partes foram concebidas e como estão interligadas. E por último, o sétimo capítulo é dedicado à conclusão.

Capítulo 2
Estado de Arte

Neste capítulo são apresentadas algumas ferramentas, métodos e soluções existentes e que são utilizadas atualmente para solucionar problemas análogos ao presente trabalho. Relativamente à dinamometria, são demonstrados vários dinamómetros utilizados atualmente em diferentes exames. São também apresentadas algumas *frameworks* ligadas à saúde e demonstrado como e o que estas pretendem alcançar.

2.1. Dinamometria

Existem atualmente vários dispositivos portáteis utilizados para a medição de força, que geralmente consistem num dispositivo de aperto com um uma agulha ou mostrador LCD para apresentar os resultados das medições. Soluções com a possibilidade de transmitir os dados sem fios para o computador ou telemóvel são relativamente recentes e encontram-se pouco desenvolvidas.

2.1.1. Dinamómetros Analógicos

Os dinamómetros analógicos são os mais comummente utilizados na medicina e apresentam apenas os valores medidos instantaneamente.

2.1.1.1. DINAMÓMETRO JAMAR

O dinamómetro Jamar (Figura 2-1) é a referência em equipamentos de teste de força de preensão e o mais comummente utilizado para medição clínica. Fornece resultados precisos, que podem ser facilmente verificados e repetidos de modo a que sejam comparáveis em toda a literatura. Este é um requisito importante nas clínicas que pretendem estabelecer as bases de força e acompanhar o progresso do paciente. Além disso, o dinamómetro Jamar possui valores padrão caracterizados para adultos e crianças, que podem ser facilmente utilizados clinicamente.

Como forma de utilização, enquanto os dedos puxam as alças em direção à palma

da mão e ao polegar, o dinamômetro mede a força entre as duas alças da sua estrutura unidireccionalmente, de maneira comparável a um elemento de mola [16]. Este permite testar diferentes grupos musculares, intrínseco versus extrínseco, sendo sugerida a sua validade concorrente no teste da força muscular [17].

Figura 2-1 - Dinamómetro de Jamar.

2.1.1.2. MÉTODO RECORRENDO A ESFIGMOMANÓMETRO

Outro dos métodos utilizados para a força de preensão é feito recorrendo a um esfigmomanómetro. Com este é possível medir a força de uma forma simples e objetiva. Em primeiro lugar o esfigmomanómetro é enrolado em forma de cilindro de modo a que o paciente o consiga segurar confortavelmente.

Este é depois insuflado com ar até uma pressão de 20 mmHg, a partir da qual o paciente terá de aplicar a força máxima. Consoante os valores obtidos é possível caracterizar funcionalmente o paciente. No exemplo da Figura 2-2 é possível observar que com a mão esquerda foi aplicada uma pressão de cerca de 250 mmHg e de apenas 110 mmHg na direita.

Figura 2-2 - Dinamómetro recorrendo a esfigmomanómetro [18].

Esta é uma técnica conveniente que foi utilizada previamente e demostra valores confiáveis quando comparado ao dinamómetro de Jamar [18]

2.1.1.3. MARTIN VIGORIMETER

O Martin Vigorimeter (Figura 2-3) consiste numa borracha em forma de pera, ligada através de um tubo de borracha a um manômetro. Este foi projetado especificamente para pacientes com artrite de modo a permitir uma avaliação adequada da força de preensão, evitando stress excessivo nas articulações fracas ou dolorosas. Em comparação ao dinamómetro de Jamar, este dispositivo apresenta uma solução mais confortável e ergonómica para o paciente, pois os requisitos em relação à posição da mão são mínimos e, portanto, é possível aplicar uma força máxima de uma posição confortável.

Figura 2-3 - Martin Vigorimeter.

2.1.2. Dinamómetros Digitais

Mais recentemente têm vindo a surgir os dinamómetros digitais, no entanto ainda são menos utilizados que os analógicos. Estes já apresentam formas de gravar digitalmente os valores medidos para análise posterior.

2.1.2.1. DINAMÓMETRO JAMAR SMART HAND

O dinamómetro Jamar tem também uma versão digital sem fios. Esta comunica com uma aplicação móvel, tornando assim o exame mais rápido e simplificando a análise dos resultados devido à interface gráfica. Os dados relativos à força de pressão obtidos durante o exame poderão ser exportados em formado PDF, sendo que adicionalmente estão disponíveis valores estatísticos como a média, desvio padrão e coeficiente de variação [19].

2.1.2.2. MICROFET

A microFET disponibiliza na sua oferta diferentes dinamômetros e inclinómetros desenvolvidos para eliminar a natureza subjetiva dos testes músculo-esqueléticos que podem ser utilizados para diversos tipos de exames. Estes comunicam com um software clínico disponível para Windows, que realiza automaticamente cálculos e verificações de validade, gera tabelas e gráficos a partir dos dados, cria os relatórios das sessões, orienta o clínico através de testes selecionados ou protocolos inteiros e fornece imagens incorporadas para demonstrar o calibre adequado e posicionamento do paciente [20].

Com os diferentes dispositivos criados pela microFET é possível não só realizar exames de dinamometria, mas também medir a amplitude de movimento, força de aperto (medida através da força de aperto do polegar com o dedo indicador) e realização da dinamometria para avaliação de diferentes músculos.

2.2. mHealth

Existem inúmeras aplicações relacionadas com mHealth disponíveis para smartphones, no entanto cerca de 30% das aplicações são destinadas ao uso por profissionais da área da saúde, incluindo médicos, enfermeiros, estudantes de medicina. [21]. Nesta subsecção, descrevemos algumas das soluções existentes no estado da arte.

2.2.1. CAALYX: Complete Ambient Assisted Living Experiment

O CAALYX foi um projeto de dois anos financiado pela Comissão Europeia ao abrigo do FP6 (Sixth Framework Programme). Este foi desenvolvido com o objetivo de monitorizar o estado de saúde de utilizadores idosos 24 horas por dia, sete dias por semana, de modo a prever/detetar quaisquer condições adversas de saúde em desenvolvimento e prevenir complicações antes que elas se desenvolvam, respeitando a privacidade e a vida pessoal dos utilizadores necessidades.

O objetivo do CAALYX é desenvolver um dispositivo vestível e leve, capaz de medir sinais vitais do utilizador e comunicar automaticamente em casos de emergências onde quer que o utilizador esteja. As principais tarefas passaram por desenvolver um dispositivo para detetar sinais e padrões para determinar estados críticos na saúde do utilizador, capaz de detetar quedas do utilizador quer este esteja em casa ou não e obter a posição geográfica de forma a saber a posição exata do utilizador. Desta forma,

permite a monitorização dos utilizadores e o cuidador decidirá se deve acionar o serviço de emergência. Quando o utilizador está em casa os sensores estão ligados via Bluetooth ao computador, permitindo assim o registo dos dados. Quando o utilizador se encontra fora de casa, a monitorização e registo de dados continua a ser feita via smartphone, com a adição do GPS do dispositivo móvel para obter a localização exata do utilizador.

O CAALYX está dividido em três áreas diferentes (Figura 2-4): o "Sistema de Monitorização de Roaming", o "Sistema de Monitorização Doméstico" e o "Sistema Central de Atendimento e Monitorização".

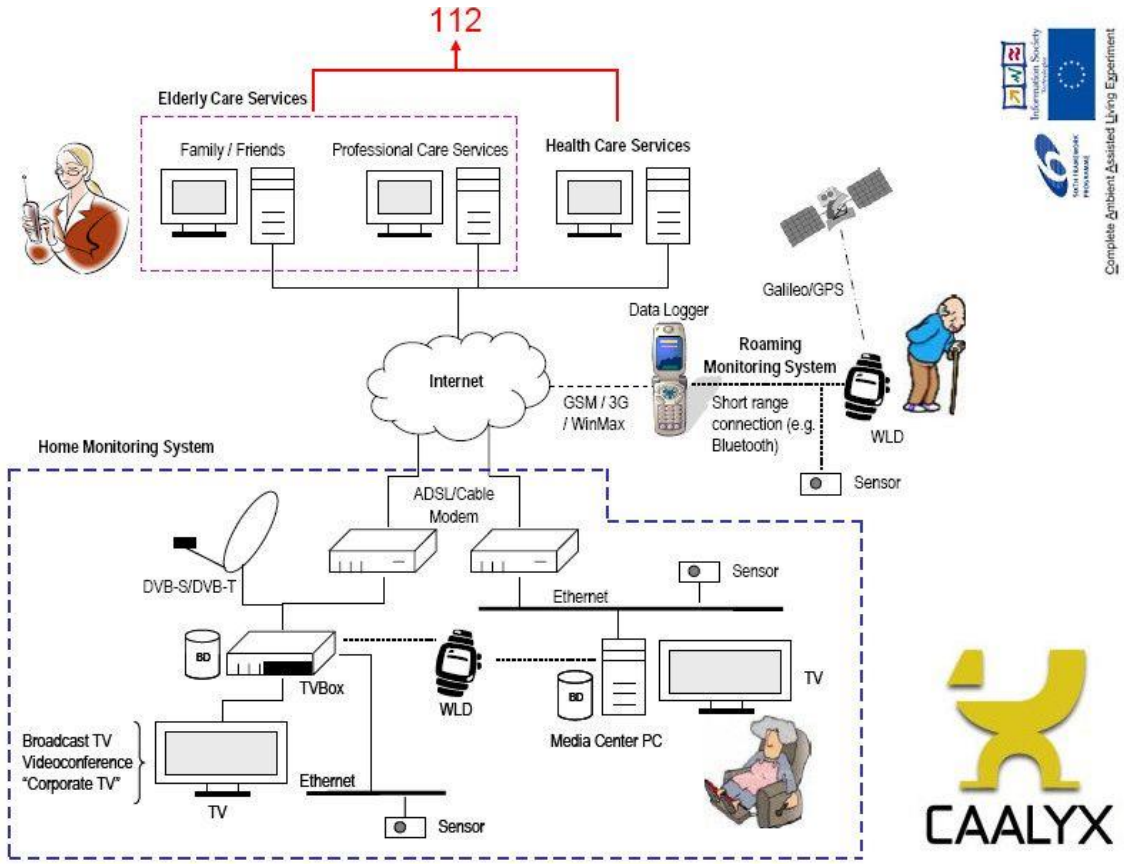

Figura 2-4 - Arquitetura do sistema CAALYX [22].

- O **Sistema de Monitorização de Roaming** tem o objetivo de monitorizar o utilizador nas suas atividades diárias, quer este esteja em casa ou não. Em caso de urgência este envia um pedido de ajuda ao Serviço Central de Atendimento, bem como alguns sinais vitais.
- O **Sistema de Monitorização Doméstico** tem o objetivo de monitorizar o utilizador em casa, integrando os diferentes dispositivos e sensores disponíveis. É também suportado um sistema de voz ou vídeo chamada que pode ser usado

para monitorização remota ou prestação de serviços, bem como a inclusão social no caso de videoconferência com amigos ou familiares.

- O **Sistema Central de Atendimento e Monitorização** recebe os alertas de todos os utilizadores inscritos. O cuidador avalia os dados recebidos e consoante a gravidade destes fica encarregue de alertar, ou não, o serviço de emergência, que caso seja alertado, recebe também a posição geográfica e os dados relativos à emergência. Além deste serviço é também fornecida a possibilidade de videoconferência com o ambiente doméstico, consoante os pedidos do utilizador. Inclui ainda outros serviços como lembretes para tomar comprimidos, lembretes de atividades e visitas programadas, visitas eletrônicas etc.

Este projeto foi continuado com o eCAALYX (Enhanced Complete Ambient Assisted Living Experiment), também financiado pela Comissão Europeia ao abrigo do programa Ambient Assisted Living (AAL). Em comparação com o anterior, a diferença que este apresenta a possibilidade de fazer a monitorização da saúde de idosos com múltiplas condições crónicas, em casa ou fora desta. Sendo que o anterior não cobria a monitorização e gestão de utilizadores com presença de uma ou mais condições adicionais co-ocorrendo [22].

2.2.2. CogKnow

O CogKnow, tal como o CAALYX, foi um projeto de três anos financiado pela Comissão Europeia ao abrigo do FP6. O objetivo foi desenvolver um dispositivo protético cognitivo validado por um utilizador e serviços associados para pessoas idosas com demência leve. Este projeto teve a finalidade de entender as necessidades de pacientes com demência leve do tipo Alzheimer e obter soluções portáteis e configuráveis para ajudar estas pessoas a navegar durante o seu dia. Especificamente, os protótipos foram desenvolvidos para ajudar os pacientes a lembrarem-se, manter contato social, realizar atividades quotidianas e de lazer, e aumentar o sentimento de segurança.

De maneira a obter os dados, o utilizador tinha disponível em casa um dispositivo responsável por recolher a informação das atividades que realizava. Quando este se encontra fora de casa, terá consigo um dispositivo responsável por fazer a monitorização e realizar as mesmas tarefas que o dispositivo de casa. O dispositivo móvel tem também a responsabilidade de levar o utilizador de volta a casa caso este se

encontre desorientado [23].

O COGKNOW está dividido em três componentes principais (Figura 2-5), estes são:

- **Home Hub** disponibiliza os serviços dentro da própria casa. Este está disponível num tablet que funciona como hub para os sensores disponíveis em casa.
- **Cognitive Assistant** é um dispositivo móvel com recursos para chamadas e com o objetivo de obter a localização do utilizador quando este está fora de casa.
- **Servidor** que armazena as informações relacionadas aos utilizadores e controla configurações detalhadas de serviço, como lembretes. As informações armazenadas no servidor são automaticamente sincronizadas com o Cognitive Assistant e o Home Hub [24].

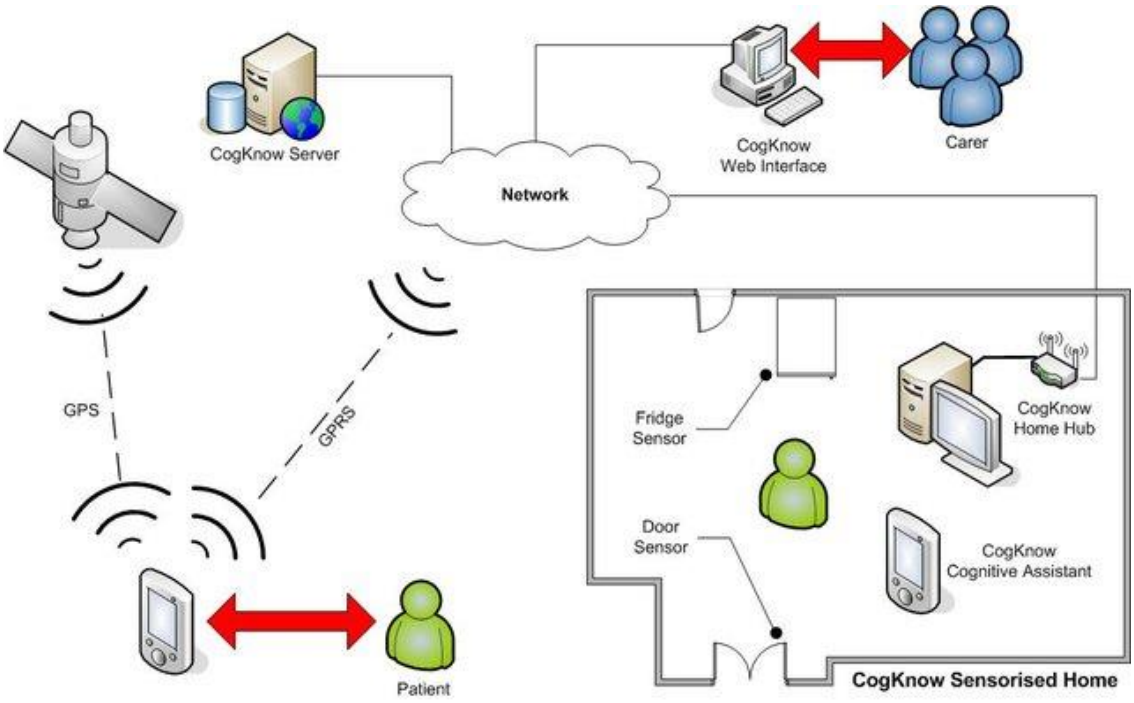

Figura 2-5 - Arquitetura do sistema COGKNOW [24].

2.2.3. ICT4Depression

O ICT4Depression foi um projeto financiado pela Comissão Europeia ao abrigo do programa FP7. Calcula-se que até 2030 a depressão seja o distúrbio de saúde com a maior carga nos países mais desenvolvidos. Este projeto foi desenvolvido com o objetivo de aliviar esta carga fazendo uso de tratamentos da depressão e das inovações em tecnologias de informação e comunicação. Tem o propósito de incluir tratamentos de autoajuda para a depressão, avaliação automática recorrendo a um smartphone ou via web, sensores biomédicos vestíveis para monitorizar diversas atividades quotidianas e

indicadores eletrofisiológicos, métodos computacionais para apoio à avaliação do estado de um paciente e se existe risco de recaída. Foi criado como um sistema flexível de forma a ser possível apoiar as pessoas recorrendo a observações e feedback contínuos via smartphone e web.

Neste projeto o uso do smartphone tem duas utilidades, ter de forma portátil e resumida os mesmos módulos disponíveis via web e fazer a monitorização do progresso do paciente de forma a dar a este o apoio adequado consoante o progresso da terapia. O smartphone recolhe também informação via avaliação momentânea ecológica em tempo real sobre contexto, comportamento e humor em ambientes naturais. Adicionalmente, são também medidos os níveis de atividade do paciente, recorrendo aos sensores do smartphone.

O ICT4Depression recorre ao uso de sensores biomédicos para fazer uma monitorização continua do paciente. No entanto estes dados são recolhidos e analisados posteriormente, não havendo um feedback em tempo real [25].

O sistema ICT4Depression é composto por uma coleção de módulos e subsistemas (Figura 2-6), unindo-se na forma de uma arquitetura orientada a serviços. De uma perspectiva de alto nível, o sistema pode ser visto como uma coleção de ferramentas para psicoterapias e farmacoterapias, que recebem informações de vários subsistemas de monitorização, a fim de apresentar melhor formas de tratamento para o paciente.

- Os **Therapeutic modules** ajudam o paciente a avaliar a gravidade dos problemas com informações gerais sobre depressão e estabelecimento de metas. Desta forma o paciente é ajudado a recuperar o controlo da sua vida e a tratar a depressão através das técnicas indicadas.
- Os componentes **Cross-therapy** têm o principal objetivo de avaliar o progresso na recuperação do paciente. Estes recolhem informação acerca do paciente e desta forma os pacientes têm uma melhor visão do progresso de sua terapia, permitindo assim aumentar sua eficácia.
- O **Reasoning Engine** tem objetivo de fornecer orientação pessoal ao paciente através da monitorização continua, e desta forma fornecer um feedback quase instantâneo [26].

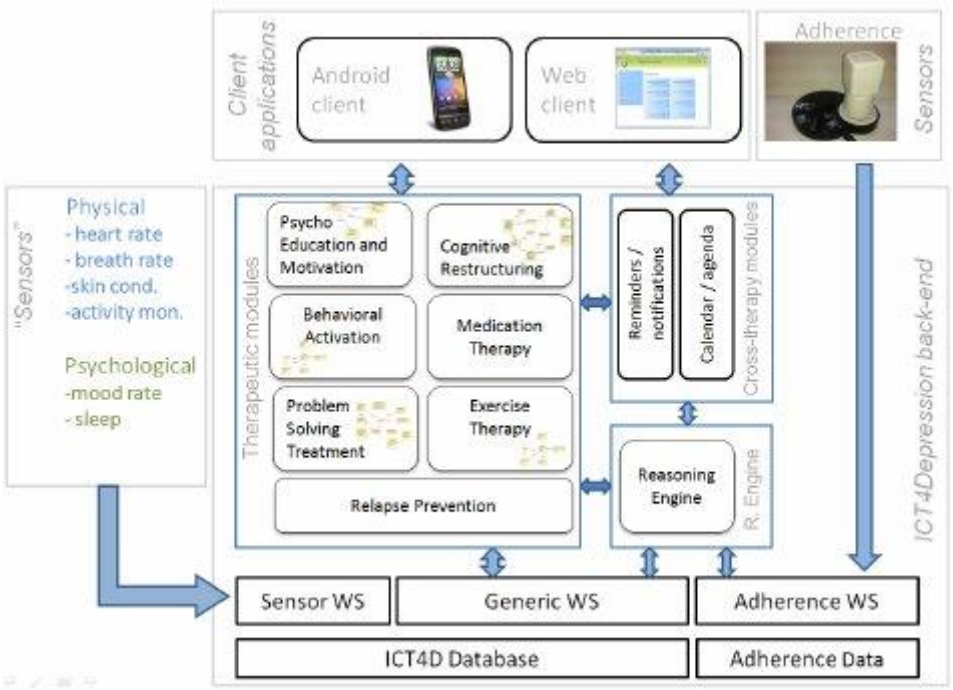

Figura 2-6 - Visão geral do sistema ICT4Depression [26].

2.2.4. iPROGNOSIS

O iPROGNOSIS foi um projeto financiado ao abrigo do programa Horizon 2020, com o objetivo de criar uma abordagem inteligente recorrendo a tecnologias de informação e comunicação para a deteção e intervenção precoce dos sintomas da doença de Parkinson e na vida quotidiana do idoso, promovendo o envelhecimento saudável e ativo, e introduzindo novas ferramentas ligadas à saúde.

A deteção de Parkinson é feita em duas fases. Numa primeira fase os dados são recolhidos através da interação do utilizador com seu smartphone ou smartwatch com a finalidade de capturar dados que possam estar relacionados com sintomas iniciais de Parkinson. Os dados são recolhidos a partir da forma como a pessoa utiliza as funcionalidades quotidianas do seu smartphone, como por exemplo, fazer chamadas telefónicas, escrever mensagens ou mesmo em como este o segura. Numa segunda fase o foco passa não só para a forma como o utilizador interage com o seu smartphone ou smartwatch mas também muitos outros dispositivos ligados à internet (Internet of Things - IoT). Desta forma é possível uma maior recolha de dados que poderá estar correlacionada com a deteção da doença de Parkinson [27].

2.2.5. MobileBIT

O MobileBIT é uma ferramenta criada para simplificar o processo de prototipagem de aplicações destinadas à aquisição, processamento, gravação, comunicação e visualização de sinais biomédicos em tempo real, recorrendo a plataformas móveis ou web. Foi desenvolvido com o objetivo de disponibilizar a diferentes profissionais ligados à saúde uma ferramenta que possa ser facilmente modificada para atender as diferentes necessidades de cada caso, tendo a vantagem de poder fazer o processamento de sinais de uma forma móvel.

Esta *framework* pode ser utilizada para a monitorização móvel, e melhorar a qualidade do atendimento e a vida do utilizador. O MobileBIT apresenta uma arquitetura versátil, construída por blocos, que o utilizador ou os profissionais podem facilmente combinar e trocar. Este utiliza uma interface web que permite uma maior flexibilidade e facilidade de configuração [28].

2.3. Discussão

Os dinamómetros analógicos apresentados anteriormente, são utilizados em diferentes áreas ligadas à saúde, dai a necessidade de estes estarem altamente validados, serem universalmente aceites e estarem testados por protocolos. A elevada precisão destes instrumentos é importante para exames de força de preensão, onde se pretende saber a força máxima, pois a análise detalhada das curvas de produção de força permite observar a convergência ou não de um utilizador para um cenário de fragilidade. Contudo, num exame onde se pretende analisar a evolução da força ao longo de um período de tempo, a falta de registo digital constitui uma limitação. As alternativas analógicas apenas permitem o registo de alguns dados, observados e anotados manualmente por um especialista. Desta forma, dados mais detalhados dos domínios estatístico, temporal ou espectral serão dificilmente obtidos.

Por sua vez, as soluções digitais e sem fios disponíveis e descritas anteriormente, apesar de extensamente validadas, apresentam algumas desvantagens. O dinamómetro de Jamar, apesar de disponibilizar uma aplicação móvel para apresentar e guardar os dados, tornando assim a solução completamente portátil, funciona apenas como um mostrador digital. Para além disso, destina-se exclusivamente para o uso com o dinamómetro disponível e, portanto, não apresenta flexibilidade para a inclusão de variáveis adicionais na análise, por via da ligação a outros dispositivos. A solução da

microFET, além de bastante dispendiosa, já contem várias opções de dispositivos, bem como a possibilidade de realizar diferentes tipos de análise, no entanto todos estes estão relacionados com a medição pontual da força e, portanto, não há flexibilidade para analisar os sinais de força fora deste tipo de enquadramento. Contendo apenas o software para Windows, esta não é uma solução completamente portátil devido à necessidade de um computador para obter os dados.

Em relação as aplicações de mHealth, grande parte das opções disponíveis nas lojas de aplicações (Google Play store, Apple store) foram desenvolvidas para tarefas simples e não relacionadas diretamente com dinamometria. As aplicações são essencialmente para uso próprio e maioritariamente para acompanhamento do utilizador, tal como medir peso, calcular índice de massa corporal, etc. Soluções mais complexas como as acima apresentadas são altamente especializadas e com o objetivo de prevenir ou detetar agravamentos na saúde do utilizador onde é necessária a validação de um especialista.

Nas soluções CAALYX e CogKnow o dispositivo móvel é apenas utilizado para fazer o registo dos dados quando o utilizador não se encontra em casa e, devido ao facto de serem projetos antigos, ainda não previam a utilização de smartphones. Já nas soluções ICT4Depression e iPROGNOSIS, existe um maior uso dos smartphones recorrendo a diversas funcionalidades que estes apresentam. No caso do ICT4Depression, é utilizado não só para o registo dos dados recolhidos de questionários dos utilizadores, mas também para disponibilizar os dados disponíveis na versão web da plataforma, possibilitando também a utilização de sensores para medir níveis de atividade. No iPROGNOSIS todos os dados são recolhidos pela utilização quotidiana do smartphone ou smartwatch recorrendo também aos sensores.

No geral, o estado da arte divide-se entre soluções adaptadas para dinamometria, mas baseadas em tecnologias legadas com baixo nível de digitalização e conectividade, e arquiteturas avançadas, criadas para servir as necessidades de outras patologias que não as abordadas no presente trabalho, e fechadas. Com este trabalho propõe-se a criação de uma solução *user-friendly* para avaliação da força de preensão baseada em smartphones e IoT, com vista a suportar a análise do declínio funcional em população idosa.

Como forma de ampliar o leque de aplicações do trabalho desenvolvido, a solução assenta numa nova *framework* proposta no contexto deste projeto, denominada KBIT, e que pretende oferecer uma base de código e metodologia para criação de aplicações móveis que usam dispositivos eletrónicos, tornando a sua utilização completamente

portátil, e incluindo ainda a criação simplificada de interfaces gráficas para visualização e registo dos dados no smartphone. No entanto, contrariamente a outras *frameworks* existentes, pretende-se que seja flexível de adaptar para permitir a conexão com diferentes dispositivos e que permita realizar diversos testes, sendo a principal vantagem a possibilidade de compor aplicações criadas autonomamente por um utilizador sem formação especializada em programação móvel, e num contexto médico.

Contudo é pretendido também que as soluções finais sejam disponibilizadas com um baixo custo. O KBIT foi criado com base no projeto MobileBIT, com o objetivo de ser fácil de utilizar e permitir ao utilizador adaptar as aplicações às suas necessidades. O KBIT faz uma atualização às ferramentas utilizadas, facilitando algumas tarefas ao utilizador, nomeadamente modificar as funcionalidades de forma mais simples e adicionando diferentes opções para armazenamento de dados na *cloud*.

Capítulo 3
Background

Neste capítulo são apresentadas as ferramentas e tecnologias utilizadas para o desenvolvimento da aplicação. Durante o desenvolvimento foram utilizadas tecnologias atuais e populares, mas já maduras. Desta forma é facilitado o futuro da utilização do trabalho desenvolvido.

3.1. Android

O Android é um sistema operativo móvel baseado em Linux, lançado sob a licença Apache de código aberto, tornando-o aberto e gratuito. A principal vantagem do uso do Android é que oferece uma abordagem unificada para o desenvolvimento. Desta maneira as aplicações podem ser executadas em dispositivos diferentes, independentemente do fabricante. Atualmente, o Android é o sistema operativo mais utilizado mundialmente, após ter ultrapassado o Windows no início de 2017. Tendo assim Android, Windows e iOS uma percentagem de utilizadores de cerca de, respetivamente, 39%, 35% e 15% [29]. Por último, uma das grandes vantagens é a disponibilidade de dispositivos de baixo custo.

Para o desenvolvimento foi utilizada a API na versão 23. Com esta versão a aplicação abrange um total de 74.8% dos utilizadores. Além de ser abrangida uma grande quantidade de utilizadores, também estão disponíveis novas funcionalidades da API. Qualquer dispositivo disponível no mercado já se encontra com a versão de Android acima da 6.0 e as marcas tendem a disponibilizar atualizações cada vez mais cedo, sendo que a disponibilidade apenas tende a aumentar [30].

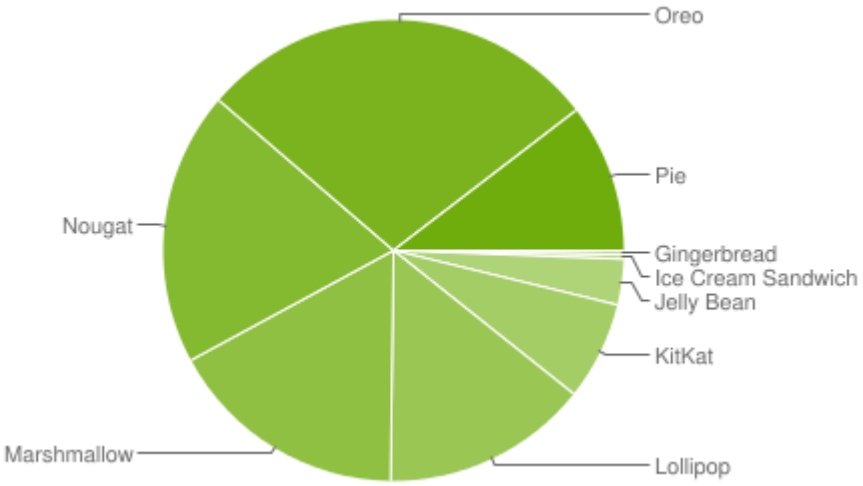

Figura 3-1 - Distribuição da percentagem de utilizadores por versão [30].

3.1.1. Kotlin

Kotlin é uma linguagem de programação *open source* e de uso geral para a JVM e Android, que combina recursos de programação funcional e orientada a objetos. É focada na interoperabilidade, segurança, clareza e suporte a ferramentas e *frameworks* como por exemplo Swing, Spring, JavaFX, etc. Kotlin teve origem na JetBrains em 2010 e é *open source* desde 2012 [31].

Em Maio de 2019, Kotlin foi anunciado como a linguagem de preferência para desenvolvimento de aplicações Android por parte da Google, tendo cada vez mais conteúdo disponível por parte da mesma [32]. Não só esta é a linguagem preferida pela Google, mas também por parte dos programadores, sendo que, em 2018, ficou em segundo lugar no top das linguagens de programação preferidas e, em 2019, em quarto lugar [33].

Este é por isso um dos motivos de escolha desta linguagem para o desenvolvimento da aplicação, pois desta forma é garantida a utilização de uma linguagem com futuro e apoio da comunidade.

3.2. Tecnologias Web

De entre as ferramentas mais poderosas atualmente existentes para criar uma boa interface com o utilizador e ter algum nível de facilidade de design, podem-se destacar as tecnologias web e os browsers. Existem muitas tecnologias atualmente capazes de criar e interagir com conteúdo online, nomeadamente o HTML, o CSS e o JavaScript.

As várias tecnologias existentes trabalham em conjunto para criar diferentes tipos de interfaces tal como se pode ver por toda a internet, desde simples websites pessoais até a plataformas de grande dimensão como o Facebook e o YouTube. O CSS serve para definir como os elementos HTML serão apresentados através dos seus chamados "estilos" (*styles*). Estes estilos podem ser guardados e editados mais tarde de modo a alterar a aparência de várias ou até mesmo todas as páginas de um website de forma centralizada. Por outro lado, o JavaScript foi criado com o intuito de permitir uma maior interatividade do utilizador com as páginas HTML e é atualmente utilizado em larga escala, sendo a linguagem de *scripting* mais popular na internet.

Os standards mais recentes para as tecnologias mencionadas são, respetivamente, o HTML5 [34] , CSS3 [35] e algumas bibliotecas como o jQuery [36] no caso do JavaScript. Nos dias de hoje, é possível criar interfaces simples, intuitivas e altamente funcionais com algum nível de facilidade por parte dos criadores devido aos avanços tecnológicos. No que toca ao Android OS, existe o componente WebView que permite que as páginas Web sejam mostradas como parte do layout das aplicações nativas o que, em conjunto com a possibilidade de ligar código Java/Android ao JavaScript, permite desenhar interfaces completamente baseadas em tecnologias Web. Além de facilitar o design em si e dar mais liberdade ao programador, isto permite que haja reutilização de código entre aplicações de desktop e móveis devido à união dos diferentes meios.

3.2.1. D3.js

D3.js (Data-Driven Documents) é uma biblioteca de JavaScript para produzir métodos de visualização de dados em navegadores web de forma dinâmicas, interativas e personalizável. Foi desenvolvida por Mike Bostock, é de código aberto, e utiliza as tecnologias Web como o SVG, HTML e CSS. Devido ao D3 se basear nos padrões da web, beneficia de todos os recursos dos navegadores modernos sem a necessidade de uma *framework* proprietária, combinando poderosos componentes de visualização e uma abordagem orientada a dados para a manipulação do DOM.

O D3 é usado em de milhares de sites, desde para a criação de gráficos interativos para sites de notícias on-line, painéis de informações para visualização de dados ou produção de mapas. Devido, a natureza exportável do formato SVG permite que os gráficos criados sejam usados em aplicações dinâmicas [37].

3.2.2. Flot

O Flot é uma biblioteca de desenho de gráficos em JavaScript para aplicações científicas e de engenharia. A fonte de distribuição do Flot é compilado para o ES5, sendo portanto compatível com todos os principais *browsers*. Este permite o desenho de variados tipos de gráficos tais como gráfico de linhas, gráfico circular, gráfico de barras, gráfico de áreas, gráfico empilhado, etc [38].

3.2.3. Google Charts

O Google Charts é uma biblioteca de gráficos baseada em JavaScript, criada para aprimorar aplicativos Web, adicionando recursos para a criação de gráficos interativos. Este fornece uma grande variedade de tipos de gráficos. O gráfico já vem com um estilo pré-definido, mas pode ser facilmente alterado. Os gráficos são renderizados utilizando HTML5/SVG, sendo desta forma compatíveis com uma grande parte dos browsers [39].

3.3. Aplicações Híbridas

As aplicações híbridas enquadram-se numa categoria especial de aplicações web, que utilizam um ambiente baseado na web através do uso de APIs de plataforma nativas disponíveis num determinado dispositivo. No caso do presente trabalho, a aplicação possui uma interface gráfica baseada em tecnologias Web e faz uso das APIs nativas da plataforma Android.

Ao contrário das aplicações da Web, alojadas num servidor e acessíveis através de um URL, as aplicações híbridas são normalmente instaladas por meio de uma loja de aplicativos e ficam disponíveis localmente no dispositivo como se tratasse de qualquer outra aplicação. Isto significa que os utilizadores precisam seguir o mesmo procedimento para instalar aplicações híbridas ou aplicações nativas.

Estas aplicações desempenham um papel crítico na ponte entre os recursos Web e os recursos protegidos do dispositivo (ex: acesso ao sistema de ficheiros, utilização da interface Bluetooth, entre outras), permitindo a criação de aplicações que beneficiam do melhor dos dois mundos [40].

3.4. JSON

JSON é um formato leve de dados originado a partir do JavaScript e que tem como

maior vantagem o facto de ser altamente legível tanto para humanos como para máquinas. Isto significa que existe uma maior facilidade por parte tanto dos humanos como das máquinas de interpretar e produzir dados neste tipo de formato. Em comparação com o XML (Figura 3-2), é visivelmente mais legível e simples para dados correspondentes.

Apesar da sua relação com o JavaScript, o JSON é um formato de texto que é completamente independente da linguagem de programação ou do ambiente utilizado. Uma das suas grandes vantagens como linguagem de troca de dados é que utiliza convenções que são muito familiares à família de linguagens baseadas no C, tais como o próprio C, C++, C#, Java, JavaScript, Python e muitas mais.

Esta notação tem como base duas estruturas: uma é baseada em pares de chaves e valores, tal como os dicionários em algumas das linguagens mais conhecidas, e a outra é uma lista ordenada de valores que pode ser comparada às listas ou vetores dessas mesmas linguagens. Valores estes que podem ser do tipo `string, number, boolean, null, object ou array.` Estes valores podem ainda estar *nested* [41].

JSON

```
{
    "Pages": {
        "Login":false ,
        "Results":true ,
    },
    "Source": {
        "Bitalino": {
            "MACAddress" : "20:15:05:29:21:00",
            "SampleRate" : 100,
            "Channels" :[0],
            "Port" : 0
        }
    },
    "Processing" : {
        "InvertValue": true
    },
    "Sink" : {
        "Graph": {
            "Type" : "speedMeter"
        }
    }
}
```

XML

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
   <Pages>
      <Login>false</Login>
      <Results>true</Results>
   </Pages>
   <Processing>
      <InvertValue>true</InvertValue>
   </Processing>
   <Sink>
      <Graph>
         <Type>speedMeter</Type>
      </Graph>
   </Sink>
   <Source>
      <Bitalino>
         <Channels>
            <element>0</element>
         </Channels>
         <MACAddress>20:15:05:29:21:00</MACAddress>
         <Port>0</Port>
         <SampleRate>100</SampleRate>
      </Bitalino>
   </Source>
</root>
```

Figura 3-2 - Comparação entre JSON e XML

3.5. BITalino

O BITalino é uma placa de desenvolvimento *low cost* para a aquisição de dados, que permite que se criem projetos usando sensores para medição de sinais biomédicos

[42]. Existe também a ferramenta OpenSignals, que permite em tempo real o controle total do dispositivo e a visualização de dados on-line e off-line. Para o desenvolvimento recorrendo ao BITalino estão disponíveis APIs para diferentes tecnologias como por exemplo, Java, Python, Matlab, etc.

O projeto foi desenvolvido com o BITalino como principal ferramenta para a obtenção de dados devido à facilidade de utilização e implementação, quantidade de sinais possíveis de obter, custo e acessibilidade. A placa utilizada foi a (r)evolution Board Kit BT[1], que permite taxas de amostragem e 1, 10, 100 ou 1000Hz, tem 2 entradas digitais, 2 saídas digitais, 6 entradas analógicas, 1 entrada auxiliar para a bateria, e 1 saída analógica. Podem ser integrados diferentes sensores tais como, eletromiografia, eletrocardiografia, atividade eletrodérmica, eletroencefalografia e acelerômetro e atuadores como *buzzer*, conversor digital-analógico, LED, ou motores de vibração [43].

3.6. Cloud

Como um paradigma cada vez mais popular de tecnologia e negócios, a computação na *cloud* tem vindo a conquistar a computação comercial. Esta trouxe uma grande mudança na forma em como a informação é armazenada e como a utilização de programas é feita. Em vez de executar programas ou armazenar dados num computador individual, tudo é alojado na "nuvem" - um conjunto de computadores e servidores que são acedidos via Internet. A computação *cloud* permite o acesso em vários dispositivos, como computadores ou smartphones em qualquer parte do mundo. Com a computação *cloud*, o objetivo é ocultar a complexidade da gestão da infraestrutura aos utilizadores. Ao mesmo tempo, estas plataformas oferecem escalabilidade massiva, alta confiabilidade, elevado desempenho e configurabilidade especificável, sendo estes recursos fornecidos a custos relativamente baixos em comparação com infraestruturas dedicadas. Quanto à segurança, como várias cópias dos dados são mantidas continuamente, mesmo se uma ou mais máquinas apresentarem problemas, o serviço continua a funcionar normalmente [44].

[1] https://bitalino.com/en/board-kit-bt

3.6.1. Dropbox

O Dropbox é um serviço de alojamento de ficheiros que oferece armazenamento em nuvem, sincronização de arquivos, nuvem pessoal e software para os diferentes dispositivos que o utilizador poderá utilizar. Este serviço reúne ficheiros de forma centralizada, criando uma pasta especial no computador do utilizador. O conteúdo dessas pastas é sincronizado automaticamente com os servidores da Dropbox e com outros computadores ou dispositivos em que o utilizador instalou a Dropbox, mantendo os arquivos atualizados em todos os dispositivos [45].

A Dropbox disponibiliza uma API que permite a ligação à conta do utilizador e desta forma carregar e descarregar ficheiros de uma forma fácil. No entanto para utilizar esta funcionalidade o utilizador deverá autorizar a aplicação de forma a utilizar o OAuth 2. Após autorizar, será disponibilizado um *token* de acesso que concede a capacidade da API de fazer chamadas para aceder aos ficheiros [46].

A Dropbox apresenta diversos pontos positivos por ser uma ferramenta amplamente utilizada e, portanto, os utilizadores estão habituados à mesma. Utiliza um modelo de negócios *freemium*, no qual é oferecida aos utilizadores uma conta gratuita com um espaço de armazenamento de 2 gigabytes.

3.6.2. BrainAnswer

A plataforma BrainAnswer é um sistema que permite o armazenamento remoto, navegação e visualização de dados para diferentes tipos de sensores. Atualmente a BrainAnswer permite a armazenar dados para os sensores Optris PI, g.Nautilus, BioSense, Eye Tracking, Emotiv Epoc+, Mouse Tracking. Após armazenados, os dados estão acessíveis numa base de dados on-line, permitindo que o utilizador interaja com os mesmos a partir de uma interface visual HTML5 acessível a partir de qualquer navegador da Web.

Nesta interface é possível fazer anotações, descarregar os dados, visualizar graficamente os dados e partilhá-los. Estes são organizados por estudos, cada estudo contém os dados referentes a cada teste associado a um participante, sendo que este depois contem os resultados para as diferentes tarefas. Como podemos observar na Figura 3-3, cada linha representa um teste associado a um participante. Na primeira coluna temos o nome dos participantes, na segunda o sensor associado ao teste e nas restantes os resultados para cada parte do teste [47].

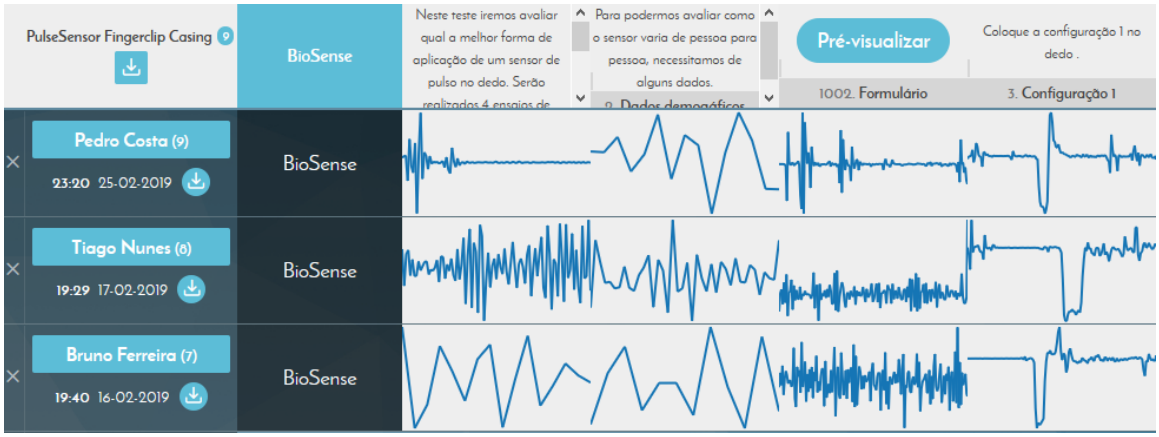

Figura 3-3 - Dashboard BrainAnswer

A BrainAnswer disponibiliza uma API REST, através da qual é possível obter e carregar dados sobre os estudos associados ao utilizador. Desta forma é possível carregar os dados diretamente da aplicação para a BrainAnswer sem haver preocupação por parte do utilizador em configurar e carregar os dados manualmente. Para aceder a esta é apenas fazer a autenticação no qual se obterá um *loginToken*. Com este *token* é possível comunicar com a BrainAnswer através da API [48].

3.6.3. RepoVizz

O repoVizz [2] é um sistema online integrado capaz de formatação estrutural, armazenamento remoto, navegação, troca, anotação e visualização de dados multimodais alinhados temporalmente.

Dada a crescente necessidade de investigação colaborativa orientada a dados, o repoVizz visa resolver as dificuldades encontradas na partilha e na navegação de grandes coleções de dados multimodais. No estado atual, o repoVizz (Figura 3-4) está projetado para armazenar fluxos de dados heterogêneos alinhados no tempo: áudio, vídeo, captura de movimento, sinais fisiológicos, descritores extraídos, anotações etc. Os dados são estruturados por meio de arquivos XML personalizados, permitindo ao utilizador organizar dados multimodais de qualquer maneira hierárquica, pois a estrutura XML contém apenas metadados e ponteiros para os arquivos de dados. Os conjuntos de dados são armazenados numa base de dados on-line, permitindo que o utilizador interaja com os dados remotamente através de uma interface visual HTML5 acessível a partir de qualquer navegador da Web. Esse recurso pode ser considerado um

[2] https://repovizz.upf.edu/repo/Home

aspeto essencial do repoVizz, pois os dados podem ser explorados, anotados ou visualizados em qualquer local ou dispositivo [49].

O repoVizz disponibiliza também uma API através da qual é possível, entre outras operações, fazer o carregamento direto de dados para a conta de utilizador associada, ficando estes disponíveis para a visualização na Web. Desta forma é possível fazer uma integração com aplicações de forma simples, onde o utilizador apenas teria de fazer a autenticação recorrendo à `api_key` associada à conta do mesmo [50].

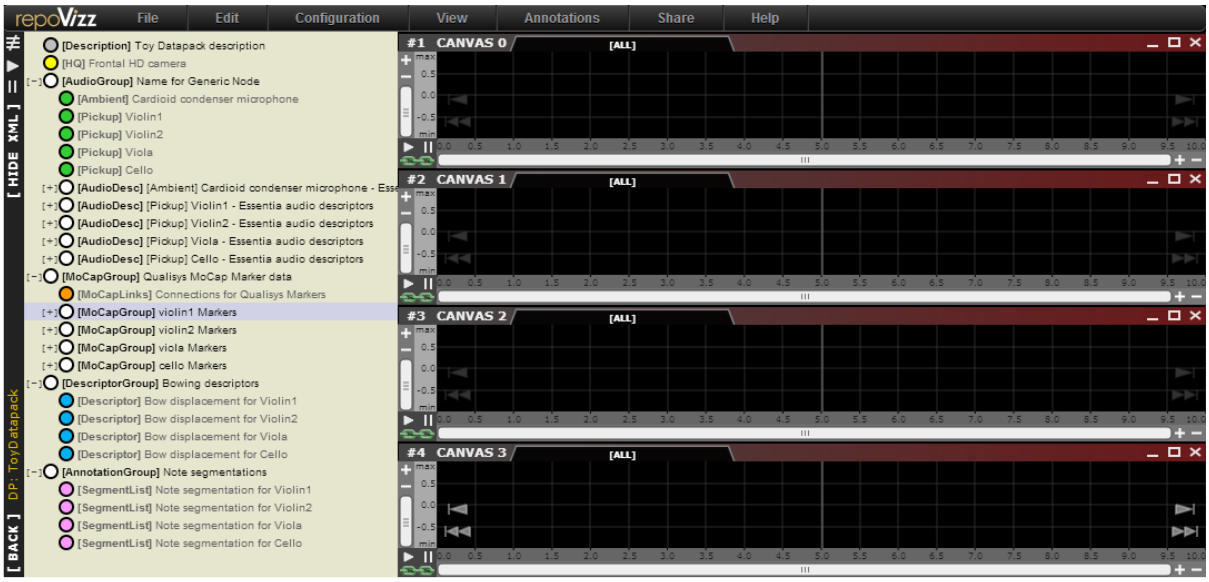

Figura 3-4 - Dashboard repoVizz

Com estas capacidades o repoVizz é uma ferramenta conveniente para a integração com o presente projeto. A capacidade de permitir diferentes formatos de fluxos de dados e estes poderem ser estruturados por meio de arquivos XML personalizados. Infelizmente este projeto parece ter sido abandonado e, apesar de ainda existir, já não existe suporte para upload de novos dados.

Capítulo 4
Planificação

4.1. Arquitetura Geral

Para o desenvolvimento da aplicação será utilizada a plataforma Android permitindo a flexibilidade e adaptação para qualquer dispositivo deste tipo. Representado na Figura 4-1, aplicação irá fazer a conexão com um dispositivo através de Bluetooth e, caso necessário, os dados serão armazenados num servidor *cloud* utilizando os serviços REST disponibilizados.

A aplicação vai ser desenvolvida recorrendo ao padrão de desenvolvimento MVP (Model - View - Presenter). Este, como estrutura, permite fazer a separação entre a informação apresentada ao utilizador e a lógica por trás desta. Com esta separação os componentes Model e Presenter serão implementas na linguagem Kotlin, facilitando o acesso aos componentes do Android e o componente View será em grande parte implementado numa WebView e recorrendo as tecnologias de Web.

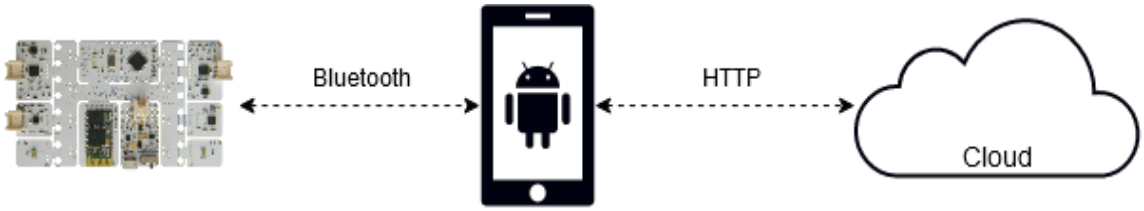

Figura 4-1 - Arquitetura geral KBIT

4.2. Metodologia

Durante o desenvolvimento do projeto recorreu-se a uma metodologia iterativa, com reuniões efetuadas semanalmente nos momentos mais importantes quando possível, de forma a obter um feedback constante e assim realizar as alterações necessárias o mais cedo possível. Com este método, existe a possibilidade de revisitar fases trabalhadas anteriormente de forma a fazer os ajustes necessários. Além desta metodologia, foi seguido também um modelo incremental, que consiste na entrega progressiva de módulos, facilitando as alterações que terão de ser efetuadas e tornando

mais fácil efetuar os testes da aplicação final. Desta forma será utilizado o modelo iterativo incremental (Figura 4-2), um modelo clássico de desenvolvimento de software.

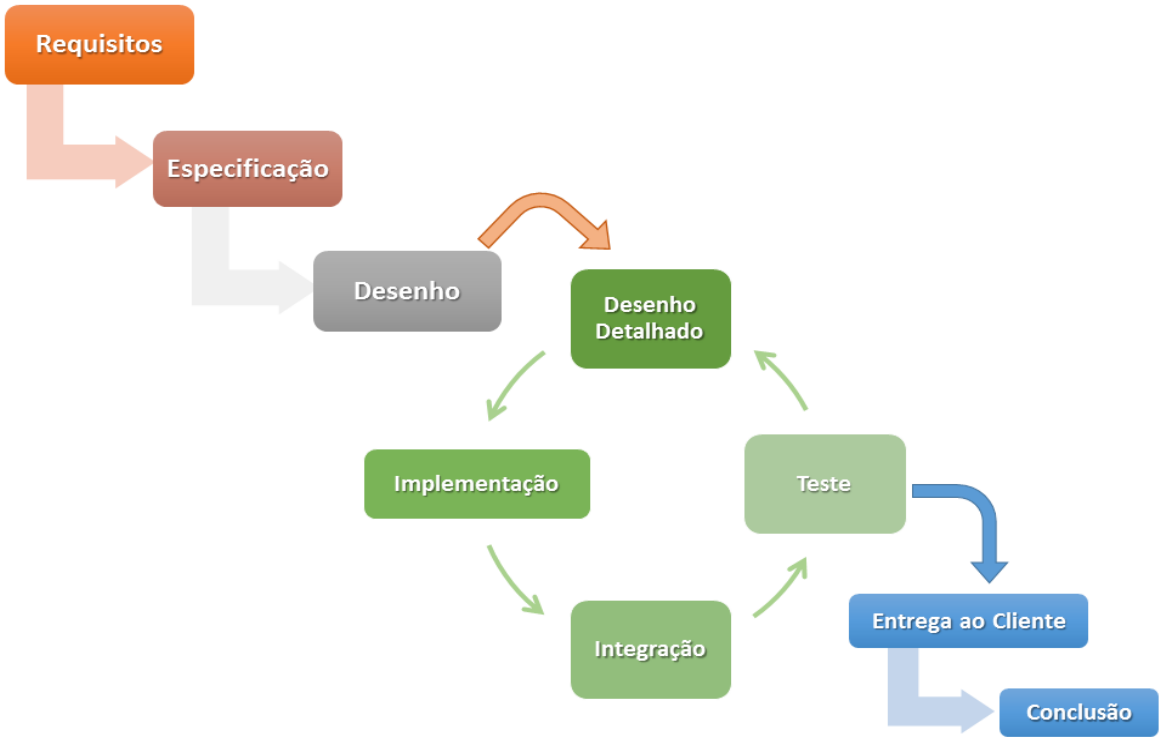

Figura 4-2 - Metodologia Iterativo Incremental

4.3. Módulos

A separação por módulos divide o projeto nas diferentes partes que o compõem. Desta forma é possível diferenciar e descrever cada parte do mesmo.

Tabela 4-1 - Tabela de módulos

Módulo	Descrição
Teste	Responsável por dispor em tempo real os valores obtidos graficamente.
Resultados	Permite ao utilizador ver todos os dados registados no exame efetuado, bem como alguns dados importantes (por exemplo, máximo, mínimo, etc.)
Autenticação	Todas as ações associadas à autenticação tal como, própria autenticação, criação de conta e recuperação de palavra-chave.
Blocos Funcionais	Separação da lógica por blocos funcionais
Workflow Manager	Responsável por a inicialização da aplicação e gerir a ligação entre

	os diferentes blocos.
Dispositivos	Conexão e obtenção de dados dos diferentes dispositivos.
Cloud	Armazenamento dos dados obtidos do teste num serviço cloud escolhido por o utilizador.
Armazenamento	Armazenamento dos dados localmente.

4.4. Requisitos Funcionais

Os requisitos funcionais definem as funções que o sistema deverá realizar. Para a atribuição de prioridades foi utilizada a nomenclatura **MoSCoW**, que consiste numa técnica de classificação usada em gestão de desenvolvimento de software para chegar a um entendimento comum com as partes interessadas sobre a importância que colocam na entrega de cada requisito do sistema.

Tabela 4-2- Tabela de Requisitos

Requisito	Prioridade
Aplicação	
Utilização de webview para interface com o utilizador	Must
Blocos Funcionais	
Desenvolvimento por Blocos Funcionais	Must
Divisão em três diferentes tipos de blocos	Must
Workflow Manager	
Workflow Manager responsável configuração inicial	Must
Workflow Manager responsável por gestão dos blocos	Must
Configuração através de ficheiro de DPL	Must
Ficheiro DPL em formato JSON	Should
Dispositivos	
Ligação Bluetooth à placa BITalino	Must
Ligação a diferentes dispositivos	Want
Armazenamento	
Gravação de dados em ficheiros	Must
Gravação em diferentes formatos de ficheiro	Want
Cloud	
Ligação com plataforma cloud	Must

Diferentes plataformas cloud disponíveis	Could
Teste	
Disposição dos dados graficamente	Must
Diferentes tipos de gráficos	Should
Resultados	
Gráfico para disposição de dados geral	Must
Disposição de valores importantes	Must
Disposição de lista de exames prévios	Could
Autenticação	
Autenticação através do método email e palavra-chave	Must
Autenticação com a associação da conta Google	Could
Recuperação de palavra-chave	Must

4.5. Mockup

Um *mockup* é um protótipo de um sistema que permite testar rapidamente um projeto. Desta forma as modificações a realizar serão mais facilmente alteradas.

Representado na Figura 4-3, encontra-se o *mockup* realizado com base nos requisitos levantados. Neste podem-se ver quais os ecrãs necessários, uma representação básica de como serão compostos e em que ordem se encontrarão.

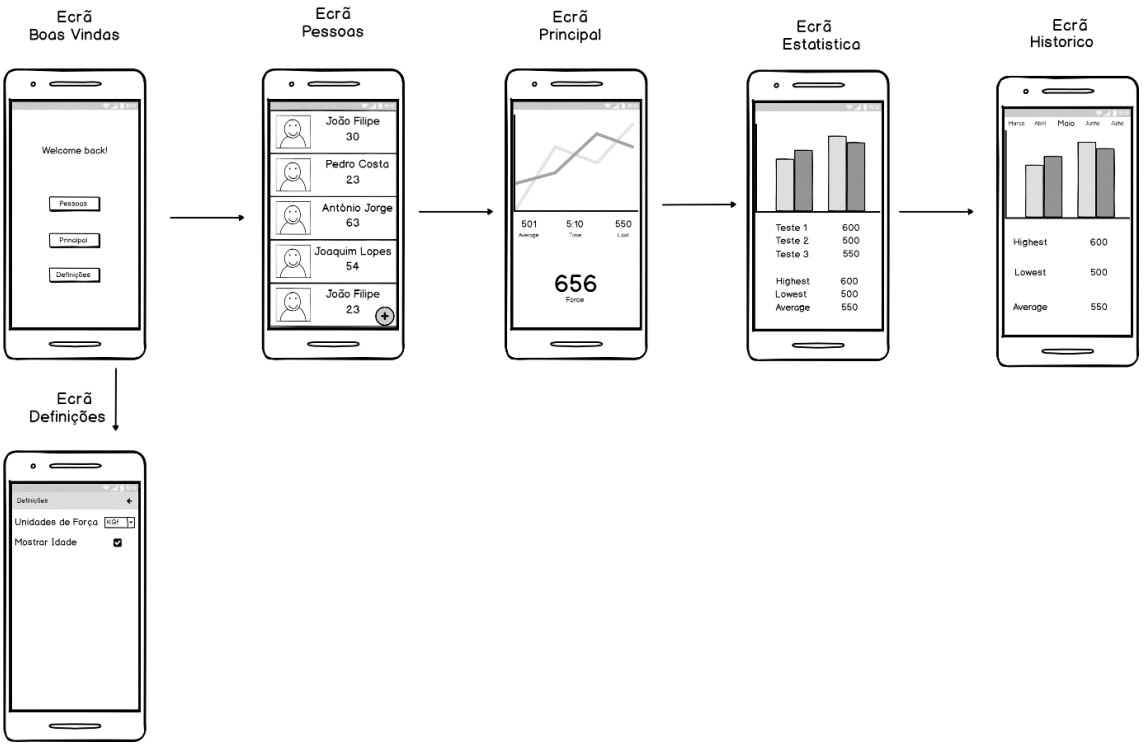

Figura 4-3 - Mockup inicial da aplicação

Capítulo 5
Arquitetura do Sistema

Neste capítulo é explicada a arquitetura sobre a qual o projeto se baseou e a forma como o mesmo funciona. São explicados os principais módulos da aplicação, qual o seu objetivo e como estes funcionam.

5.1. Arquitetura Geral

As funcionalidades da aplicação estão divididas em componentes de software chamados de Blocos Funcionais, que podem ser reutilizados e combinados de diferentes maneiras na aplicação. Os Blocos Funcionais são organizados por uma camada designada por Workflow Manager, que permite a troca de dados entres os vários componentes. Esta metodologia permite que os desenvolvedores possam conceber as aplicações numa perspetiva de mais alto nível, sem se preocuparem com os problemas mais básicos relacionados com a programação dos sensores em si, por exemplo.

O comportamento de cada um dos Blocos Funcionais é definido através de uma linguagem de descrição baseada em JSON, designada por Data Processing Language (DPL). Podem-se definir vários parâmetros como os sensores que são necessários ou as tarefas a serem realizadas. A interação com o utilizador é feita em grande parte com base em tecnologias Web como HTML, JavaScript e CSS, sendo que apenas a parte da autenticação é feita diretamente em Android.

A aplicação segue um modelo de aplicação híbrida no sentido em que combina uma camada Web com uma camada nativa. No que toca aos sensores, a aplicação esconde a manipulação dos dados na camada nativa e permite operações especificas à camada Web. Desta forma, podem-se criar aplicações Web com as funcionalidades pretendidas a partir de Blocos Funcionais previamente definidos.

Existem várias vantagens deste modelo híbrido, como por exemplo:

- Os problemas de programar os sensores físicos são abstraídos para os programadores devido ao uso dos Blocos Funcionais. A preocupação do programador foca-se apenas na aplicação como um todo e o resto fica tratado por cada bloco individual.

- A camada Web facilita o desenvolvimento de interfaces com o utilizador para quem as desenvolve, podendo escolher blocos que se pretendem ou não utilizar de uma forma muito mais simplificada e sem se preocuparem com a programação Android.
- A interface da aplicação pode ser mais facilmente alterada e adaptada as necessidades do utilizador devido às inúmeras *frameworks* de JavaScript que podem ser utilizadas.

No entanto, apesar das vantagens, uma aplicação híbrida tem por norma uma performance inferior a uma nativa, o que pode ser um problema. Nestas situações, existe a possibilidade de criar aplicações nativas Android evitando o modelo híbrido, permitindo-lhes acesso aos dados dos sensores diretamente. Uma vista completa da arquitetura pode ser vista na Figura 5-1.

Resumindo, existem quatro componentes importantes que tornam a aplicação útil no desenvolvimento de aplicações móveis baseadas no uso de sensores:

- Blocos Funcionais
- Data Processing Language
- Workflow Manager
- User Interfaces baseadas em Web.

Para aplicações baseadas nesta metodologia, a camada Web pode conter a informação relevante, que irá depois ser entregue à camada de JavaScript quando novos eventos acontecerem. Desta forma, o programador pode escolher o que fazer com esta informação no contexto destas aplicações

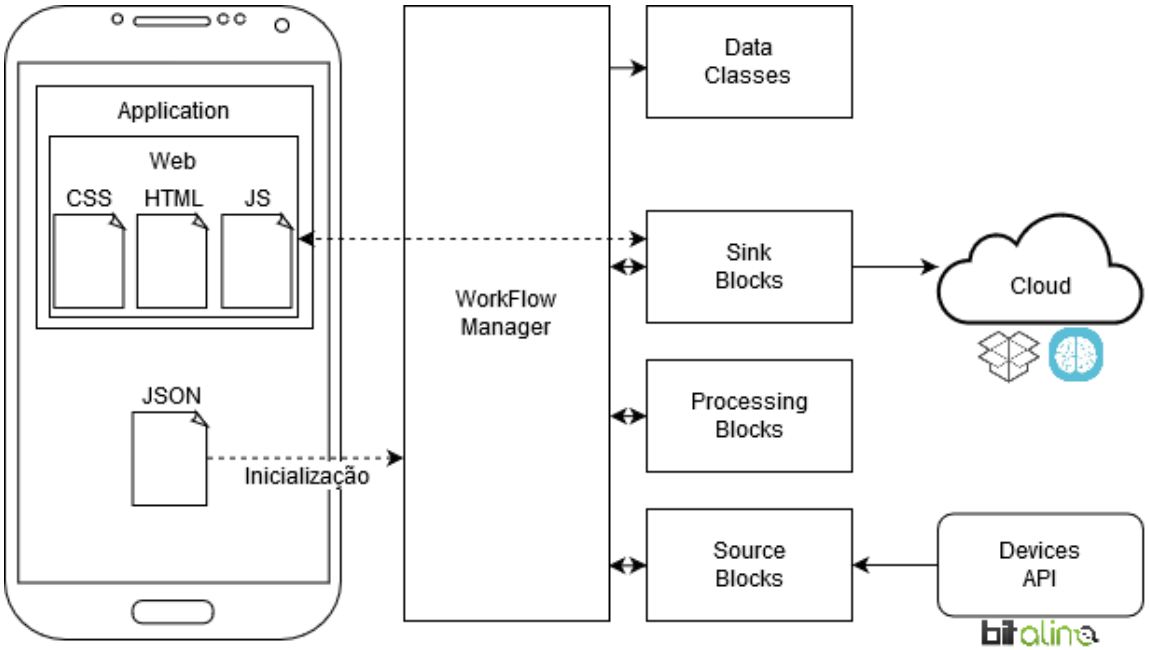

Figura 5-1 - Arquitetura da aplicação

O seguinte exemplo (Figura 5-2), representa como pode ser composta uma aplicação de forma a ser possível realizar um teste de dinamometria. O lado A demonstra o ficheiro de Data Processing Language. O lado B representa os blocos funcionais associados.

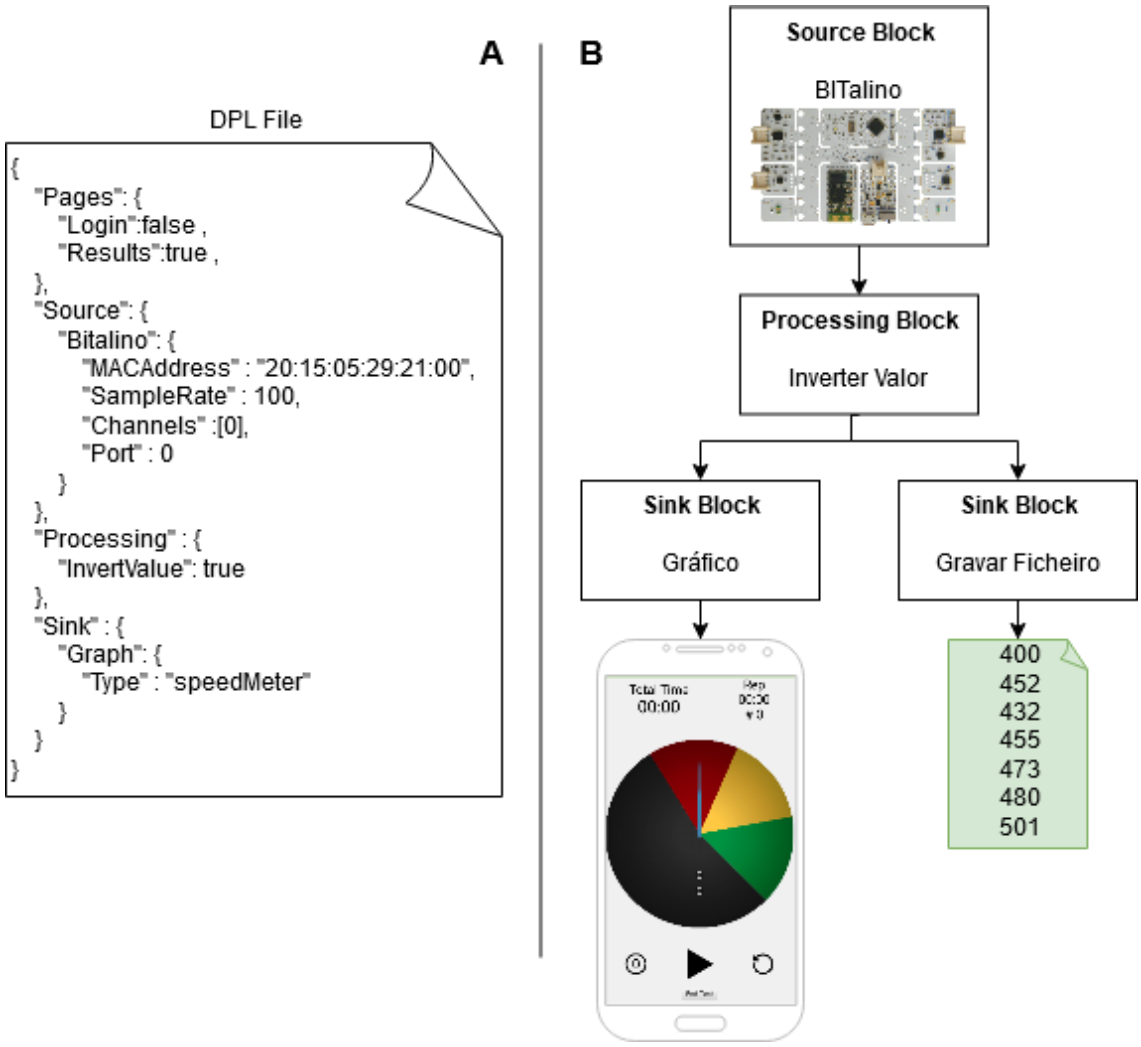

Figura 5-2 - Exemplo de blocos para aplicação de dinamometria.

5.2. Data Processing Language

Para a configuração e conexão feita inicialmente entre os respetivos blocos de entrada e saída, foi utilizado um documento baseado em JSON e designado como Data Processing Language (DPL), um exemplo deste pode ser observado do lado A da Figura 5-2. Neste documento de configuração é também definido que atividades se irão encontrar disponíveis na aplicação.

```
{
	"Pages": {
		"<PARAMETER>";"<VALUE>"
	},
	"<TYPE>": {
		"<LABEL>" : {
			"<PARAMETER>":"<VALUE>"
		}
}
```

Neste documento os blocos são definidos da seguinte maneira:

- **Label**. Cada bloco é identificado recorrendo ao nome da classe que este representa.
- **Parâmetros de configuração**. Cada bloco contém os dados para a configuração do mesmo. Por exemplo no caso de um "*Source Block",* este vai ter a informação critica para a configuração como o endereço MAC, porta de ligação, etc. No caso de um "*Sink Block"* este contem a configuração como por exemplo o nome do ficheiro, os dados para acesso à cloud, etc. No caso de um *"Processing Block"*, este tem o nome do bloco e caso seja aplicável um valor associado.
- ***Pages***. Esta secção é responsável pela configuração das páginas que serão apresentadas. Por exemplo, no caso da página de autenticação ter o valor a falso, esta não será apresentada.
- ***Type***. Estes blocos podem ser de três tipos diferentes, estes podem ser blocos *Sink, Processing* ou *Source*. Este bloco é composto por um ou vários blocos que contem o *Label* que define a sua classe e dentro destes contem a configuração referente ao mesmo.

5.3. Blocos Funcionais

Blocos funcionais são os componentes base da aplicação, onde cada componente é visto como um bloco que contém as tarefas correspondentes ao seu tipo. Os blocos são diferenciados em três tipos "*Source block", "Sink block"* e "*Processing block"*

- *Source Blocks* são os responsáveis por aquisição de dados dos diferentes dispositivos (por exemplo, BITalino), têm a responsabilidade de estabelecer a conexão (Bluetooth, Wi-Fi, etc.) com os respetivos dispositivos e obter os dados dos mesmos.
- *Sink Blocks* são os responsáveis por "consumir" os dados em operações como gravação, comunicação e visualização dos mesmos.
    - No caso do armazenamento de dados estes são usados para criar registos no armazenamento do dispositivo.
    - No caso de comunicação estes são utilizados para fazer o armazenamento remoto ou para fornecer dados para o serviço *cloud*.

    - No caso da visualização de dados, estes são responsáveis por fornecer os dados à camada Web que fica depois responsável por a correta disposição dos mesmos.
- *Processing Blocks* são os responsáveis pelo processamento dos dados, estes são os blocos intermédios o que significa que recebem os dados dos *source blocks*, fazem eventuais passos de processamento necessários e encaminham para os *sink blocks.* Estes podem estar ligados a vários *source blocks* no caso da receção de dados de mais do que um dispositivo e podem também estar ligados a vários *sink block* por exemplo no caso da visualização e armazenamento de dados.

O exemplo representado na Figura 5-2, mostra como pode ser composto o conjunto de blocos de forma a criar um teste de dinamometria. Existem quatro blocos diferentes no sistema:

- BITalino (*Source Block*) – Bloco encarregue por fazer a conexão ao BITalino, obter os valores e enviar para o WFM.
- Inverter Valor (*Processing Block*) – Bloco responsável por inverter os valores recebidos. Bloco necessário no caso dos valores recebidos terem uma progressão inversamente proporcional à grandeza física correspondente.
- Gravar Ficheiro (*Sink Block*) – Bloco responsável por guardar os valores num ficheiro após processados.
- Gráfico (*Sink Block*) – Bloco responsável por processamento necessário para os valores serem representados graficamente.

5.4. Workflow Manager

O Workflow Manager é o núcleo da estrutura, este é responsável por gerir e fazer a ligação entre os diferentes blocos.

Assim que a aplicação é iniciada, um ficheiro JSON é analisado pelo Workflow Manager. Este, através de uma biblioteca de processamento JSON, realiza as instanciações, configurações e ligações dos blocos necessários. Devido à importância do ficheiro JSON na inicialização da aplicação, é indispensável que a estrutura do mesmo se encontre válida. Caso os atributos não estejam bem definidos, será apresentada uma mensagem de erro ou quando possível é considerado um valor padrão.

Independentemente da configuração do descrita no ficheiro DPL quando a

aplicação é inicializada é feita a configuração de cada bloco que este contem. Após feita a configuração e a ligação lógica dos diferentes blocos a aplicação ficará pronta a utilizar. Com o exemplo da Figura 5-2, podemos ver que com o ficheiro de DPL, representado do lado A da figura, será processado pelo Workflow Manager, de forma a criar as ligações representadas no lado B da figura.

Capítulo 6
Testes de Ferramentas

Neste capítulo são apresentados dois testes realizados durante o desenvolvimento da aplicação. Estes foram realizados com o intuito de encontrar a ferramenta ou metodologia mais indicada para utilizar durante o desenvolvimento da aplicação.

6.1. Ferramentas de Desenho de Gráficos

Com o objetivo de selecionar a ferramenta de desenho de gráficos mais adequada, foi realizado um teste onde são comparadas algumas alternativas, indicando as vantagens e desvantagens de cada uma. A ferramenta selecionada deverá ter uma ampla diversidade de gráficos, resposta rápida em grandes volumes de dados, bem como apoio à utilização da ferramenta, como tutoriais ou suporte da comunidade.

Nesta comparação foram consideradas três ferramentas de desenho de gráficos, Flot, D3.js e Google Charts.

6.1.1. Condições dos Testes

Especificações do computador: Intel Core i5 6600K 3.50GHz, 8GB RAM DDR4, AMD Radeon R9 200 4GB, Windows 10 Education 64-bit.

Especificações do telemóvel: Marca e modelo Oneplus One, Snapdragon 801, Quad-core 2.5 GHz Krait 400, Adreno 330, 3 GB RAM, Android 7.1 (Nougat).

Condições do gráfico: Em ambos os gráficos testados apenas foi desenhado o essencial, isto é, os eixos e a linha ou barras. As bibliotecas foram descarregadas (quando possível) para a utilização local evitando desta maneira eventuais atrasos adicionais resultantes da ligação à internet. O ficheiro HTML apenas contem o mais básico para o seu funcionamento.

6.1.2. Vantagens e Desvantagens

Com os resultados destes testes foram realizados um conjunto de tabelas e gráficos de forma a resumir e demonstrar os pontos fortes e fracos da cada uma. Na Tabela 6-1

são apresentadas as vantagens e desvantagens das ferramentas. Na Tabela 6-2 é dada uma classificação a diferentes aspetos constituintes das ferramentas, com estas foram construídos os gráficos da Figura 6-1 onde é facilmente visível os pontos em que estes excedem e/ou falham. Na Figura 6-2 é apresentado na mesma figura os três gráficos da Figura 6-1 facilitando assim a comparação das ferramentas.

Tabela 6-1 - Vantagens e desvantagens das ferramentas

Vantagens	**Desvantagens**
Flot	
Fácil utilização	Pouca variedade de gráficos
Fácil configuração	Suporte acabou em 2014
Boa documentação	
D3.js	
Enorme variedade de gráficos	Curva de aprendizagem acentuada
Bastante configurável	
Boa documentação	Necessário o desenho das partes individualmente (Eixos, valores nos eixos e valor no gráfico)
Grande comunidade	
Google Charts	
Fácil utilização	Variedade de gráficos apesar de grande, não é tão extensa como a do D3.js
Facilmente configurável	
Boa documentação	Requer ligação à internet
Grande comunidade	

Tabela 6-2 - Classificação por aspeto das ferramentas

	Flot	**D3.js**	**Google Charts**
Dificuldade de Utilização	Fácil	Difícil	Fácil
Dificuldade de Configuração	Fácil	Difícil	Fácil
Variedade de Configuração	Média	Média	Bastante
Qualidade de Documentação	Boa	Boa	Boa

Variedade de gráficos	Pequena	Média	Grande
Ligação à internet	Não	Não	Sim
Comunidade	Pequena (*)	Grande	Grande
Curva de aprendizagem	Pequena	Pequena	Acentuada

* Suporte acabou em 2014

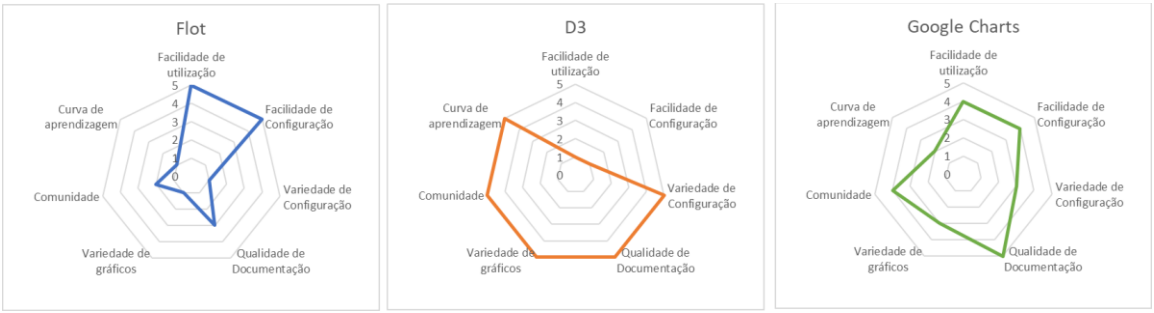

Figura 6-1 - Vantagens e desvantagens das três ferramentas

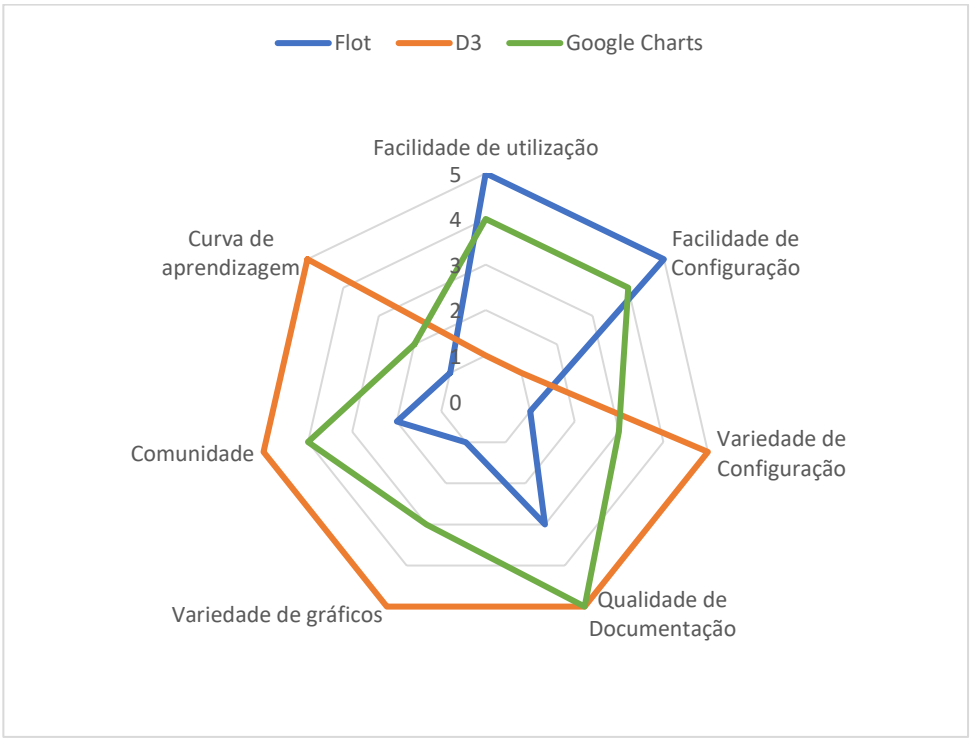

Figura 6-2 - Comparação das vantagens e desvantagens das ferramentas

6.1.3. Conclusão

Após feita a análise das diferentes opções, conclui-se que o D3.js é a ferramenta mais indicada para a utilização no projeto. Apesar da sua bastante acentuada curva de aprendizagem, esta é a ferramenta com a melhor performance, que contem a maior

diversidade de gráficos com grande variedade de customização dos mesmo e contem uma grande e ativa comunidade.

Comparativamente ao D3.js, o Flot tem uma performance relativamente semelhante e a sua utilização é a mais simples dos três, no entanto a diversidade de gráficos é bastante reduzida. O Google Charts é a ferramenta mais equilibrada, a sua utilização é bastante simples, a customização e opções adicionais são facilmente implementadas, a diversidade de gráficos é significativa e a comunidade é ativa. No entanto este perde bastante ao nível de performance sendo o mais lento dos três e tendo também como limitação a sua necessidade de uma ligação à internet para a sua utilização.

6.2. User Interface - Bottom vs Top

Com o KBIT tentou-se criar uma aplicação em que a usabilidade fosse um ponto central. Do ponto de vista do utilizador a usabilidade é importante pois pode facilitar a forma como uma tarefa é concluída com precisão e, desta forma, o utilizador obtém uma sensação agradável ao utilizar aplicação em vez de ser este a adaptar-se à aplicação.

Com o tamanho dos smartphones cada vez maior, as aplicações têm vindo cada vez mais a aproveitar este espaço. No entanto, é inevitável que algumas partes tornem-se inacessíveis (por exemplo) quando os dispositivos são usados apenas com uma mão. Atualmente muitas das aplicações desenvolvidas apresentam um problema em comum que é a acessibilidade móvel. As aplicações de browsers (Figura 6-4) são um bom exemplo, apesar de se aplicar a outros estilos de aplicação, pois muitos dos controlos estão alinhados ao topo do ecrã. Este não é um problema apenas de design, a Google disponibiliza orientações e recursos em que as ações se encontram no topo do ecrã, promovendo e facilitando assim o desenvolvimento e utilização desta abordagem.

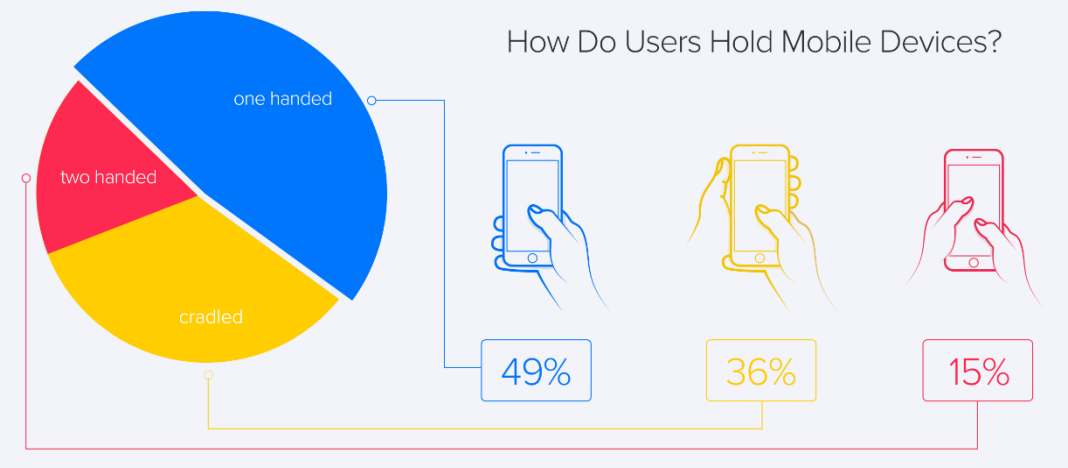

Figura 6-3 - Como os utilizadores usam o smartphone

De acordo com as estatísticas, os utilizadores seguram os telefones de três diferentes maneiras (Figura 6-3). A maioria das pessoas usa apenas uma mão para controlar o seu smartphone. Portanto, o desenvolvimento deve ser guiado tendo em conta as necessidades e hábitos dos utilizadores, não causando desconforto. Além do desconforto que pode causar, os utilizadores poderão estar a realizar outras tarefas como segurar um objecto, interagir com um paciente, ou a realizar outras tarefas que necessitem da outra mão disponível. O objetivo principal é fornecer a melhor experiência ao utilizador sem fazer movimentos adicionais para que este se sinta confortável, de uma maneira que este esteja habituado e poderá realizar outras tarefas simultaneamente.

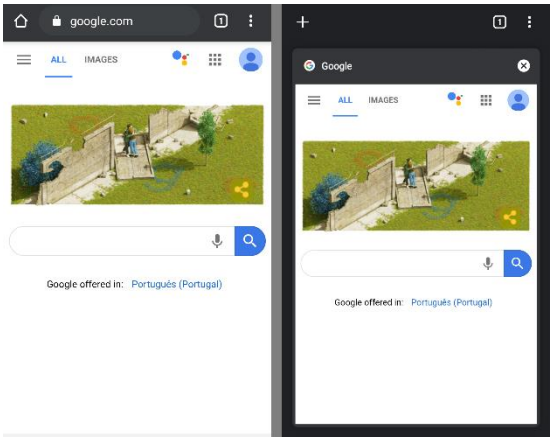

Figura 6-4 - Aplicação Google Chrome

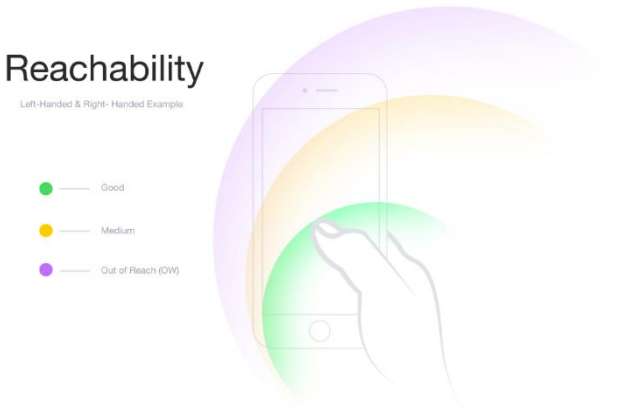

Figura 6-5 - Zonas de acessibilidade

Ao segurar o smartphone, como regra, deverá ser possível operar a aplicação apenas com os polegares. Devido a não ser possível esticar os polegares para alcançar as áreas superiores do ecrã, a experiência do utilizador deve basear-se nisso (Figura 6-5). Deve-se evitar colocar botões úteis, campos de entrada ou outros recursos na parte superior do ecrã (Figura 6-4) [51].

No entanto esta tendência tem vindo a alterar-se e empresas como a Apple, Samsung e HTC já adicionaram modos para facilitar a utilização com apenas uma mão. Quando acionado o modo, a parte superior do ecrã é chegada para baixo de forma a facilitar o uso com apenas uma mão. Aplicações importantes como o teclado também já disponibilizam o modo de uma mão (Figura 6-6), que quando acionado, faz com que o teclado se torne mais pequeno e disposto junto a um dos lados do ecrã.

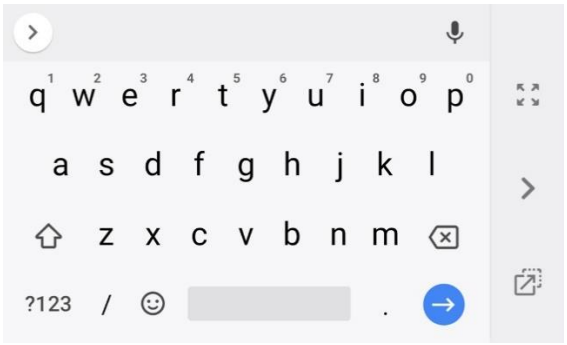

Figura 6-6 - Modo de utilização para uma mão do teclado

6.2.1. Caso de Estudo

Para testar esta hipótese foi desenvolvida uma aplicação simplificada com um conjunto de tarefas que são feitas de forma idêntica, sendo a grande diferença o alinhamento dos recursos a utilizar. Numa versão os recursos estão alinhados à parte inferior do ecrã e noutra estão alinhados à parte superior. Ambas as soluções foram

realizadas com o intuito de:

- Possibilitar o uso apenas com uma mão;
- Constituir uma solução simples;
- Manter um design da interface que o utilizador possa já estar familiarizado.

Em ambas as implementações os objetivos são exatamente os mesmos mas alcançados de forma idênticas. As tarefas são, por ordem:

- Utilizar a ferramenta de pesquisa para encontrar e selecionar o contacto “Pedro Mata”;
- Selecionar a opção de criação de contacto;
- Introduzir um nome e número e gravar o contacto.

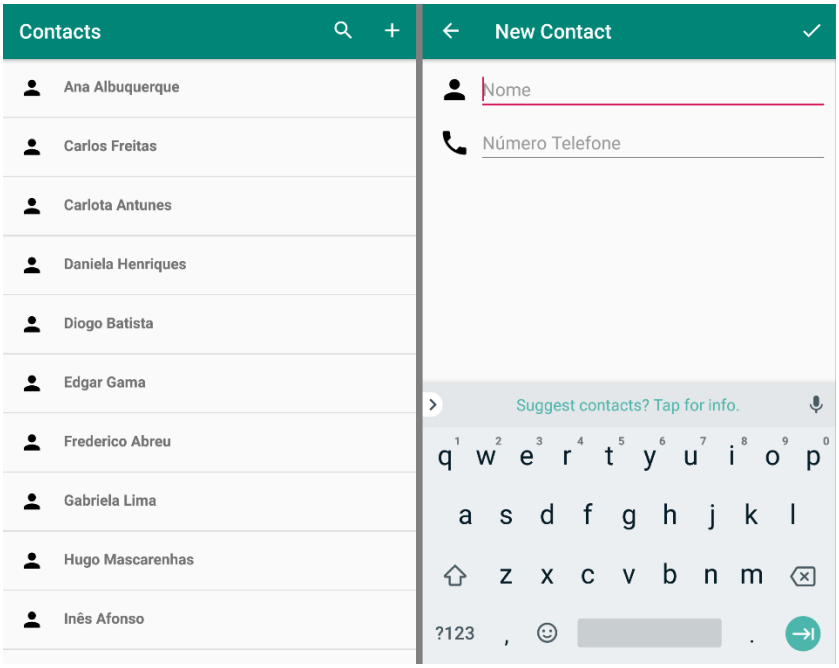

Figura 6-7 - Aplicação com alinhamento ao topo

Como se pode observar pela Figura 6-7, nesta solução os recursos estão alinhados ao topo do ecrã. Esta é uma das soluções mais encontradas em diferentes aplicações e a mais comum no caso de criação de um novo contacto.

Com esta implementação é esperada uma maior dificuldade ou até a impossibilidade de realizar as tarefas recorrendo apenas a uma mão. No entanto, sendo esta uma implementação mais comum, é esperado que os utilizadores consigam concluir a tarefa.

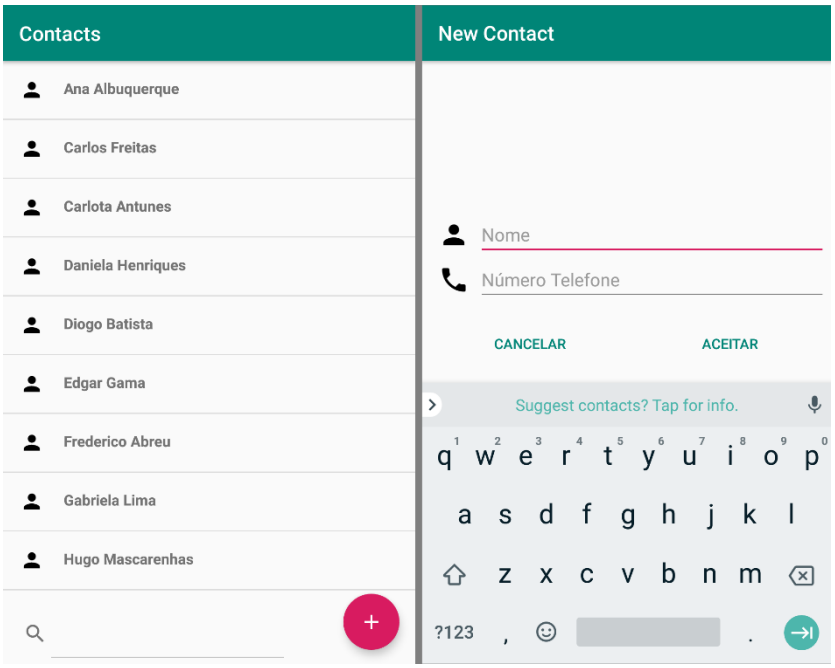

Figura 6-8 - Aplicação com alinhamento à base

Como se pode observar pela Figura 6-8, nesta solução os recursos estão alinhados à base do ecrã. Esta solução não é tão comummente encontrada, no entanto, existem cada vez mais aplicações com o alinhamento à base.

Com esta implementação é esperada uma maior facilidade na realização das tarefas recorrendo apenas a uma mão. No entanto, sendo esta uma implementação menos comum e não se encontrando os botões no primeiro campo de visão onde os utilizadores procuram por controlos, é esperado que os utilizadores tenham uma maior dificuldade de encontrar a tarefa.

6.2.2. Conclusão

Numa experiência realizada com 10 utilizadores, onde foram registados os tempos e observadas as dificuldades em cada tipo de implementação. Para o alinhamento à base obteve-se um tempo de (15 ± 4 segundos) e um tempo de (18 ± 6 segundos) para o alinhamento ao topo. Em relação as dificuldades observadas, quando o alinhamento é feito ao topo encontra-se uma maior diferença nos tempos pois existe uma maior dificuldade no acesso quão menor for o tamanho da mão do utilizador.

Portanto, relativamente à usabilidade, o alinhamento à base do ecrã mostrou ser superior e o favorito para grande parte dos utilizadores. Em relação ao alinhamento ao topo, este tem a vantagem de ser mais apelativo esteticamente e ter mais recursos que facilitam o seu desenvolvimento.

Capítulo 7
Implementação

Neste capítulo é explicada a implementação da aplicação, bem como algumas das ferramentas utilizadas e o intuito, incluindo ainda a lógica geral de organização da aplicação

7.1. Diagrama de Pacotes

Para o desenvolvimento da aplicação foi utilizado padrão arquitetónico Model-View-Presenter. O Model-View-Presenter (MVP) é uma derivação do padrão de arquitetura MVC (Model-View-Controller), usado principalmente para a construção de interfaces. No MVP, o presenter assume a funcionalidade do "intermediário" e toda a lógica de apresentação é enviada para este. O MVP defende a separação da lógica de negócios e persistência da Activity e Fragment. O objetivo é separar em três camadas (Figura 7-1) diferentes que podem funcionar independentemente.

- **View**: as *views* são responsáveis por decidir como os dados serão exibidos para o utilizador. Tipicamente é implementada através de uma Atividade ou Fragmento, ou no caso do KBIT a *webview*. Esta apenas é responsável por o que mostrar ao utilizador quando um evento ocorrer.
- **Presenter**: Toda a lógica de negócios é mantida aqui. É responsável por agir como o intermediário entre a *view* e o *model*. Este recupera dados do *model* e retorna-os formatados para a *view*. Diferentemente do MVC típico, este é responsável pelo que acontece quando existe interação com a *view.*
- **Model**: é o responsável por gerir os dados. As responsabilidades do modelo incluem o uso de APIs, cache de dados, gestão de bases de dados, etc.

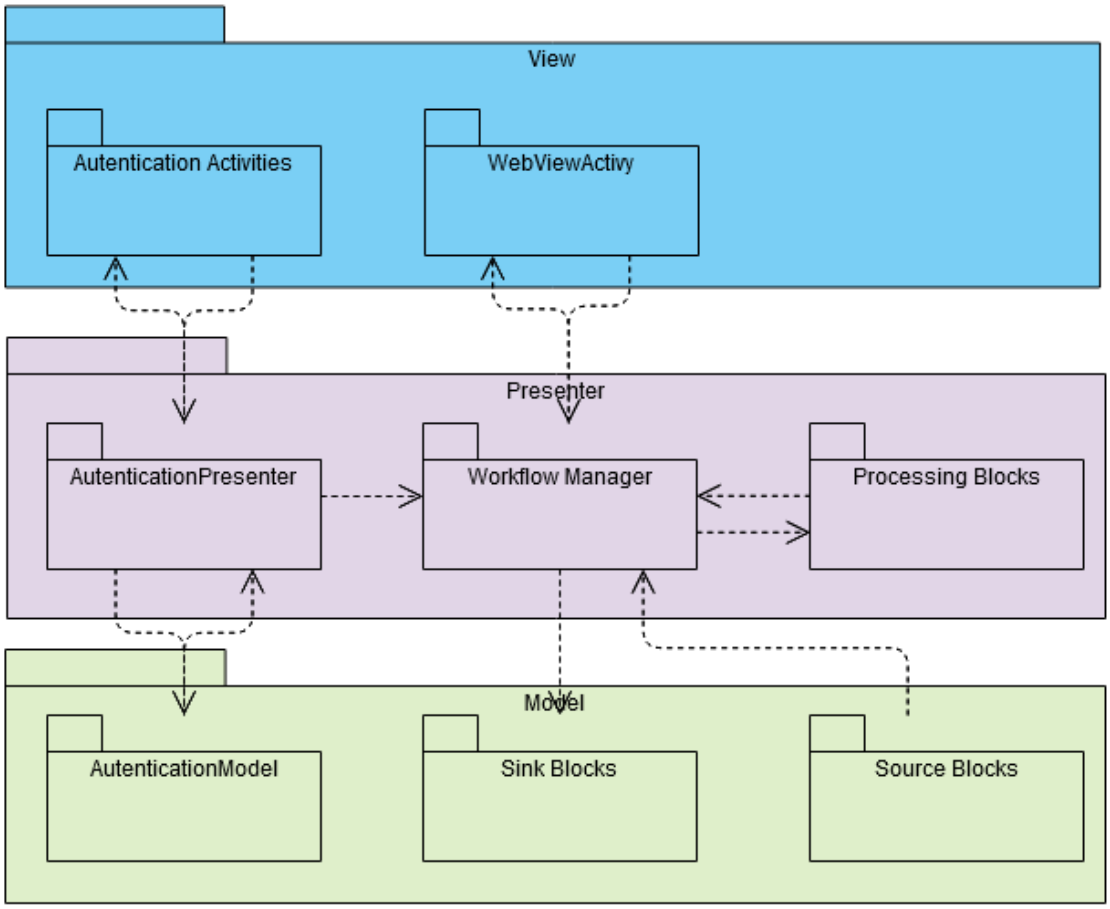

Figura 7-1 – Diagrama de pacotes da aplicação

7.2. Data Processing Language

Sendo a configuração da aplicação definida por um ficheiro JSON quando esta é inicializada, é necessário ter em atenção a validade do mesmo.

Esta é uma das bases da aplicação e é com o conteúdo deste ficheiro que os diferentes blocos funcionais são inicializados e configurados, portanto é necessário analisar o mesmo de forma a fazer a transformação em classes utilizáveis. Adicionalmente, este ficheiro define como é feita a interação com a aplicação, podendo este alterar o fluxo da mesma, como por exemplo se o módulo de autenticação é necessário ou não. No entanto o fluxo não pode ser completamente alterado pois existem módulos que são obrigatórios. Como podemos observar na Figura 7-3 – Ficheiro de DPL, os módulos de autenticação e o de resultados tem o valor de *true*, portanto serão ambos introduzidos no fluxo da mesma.

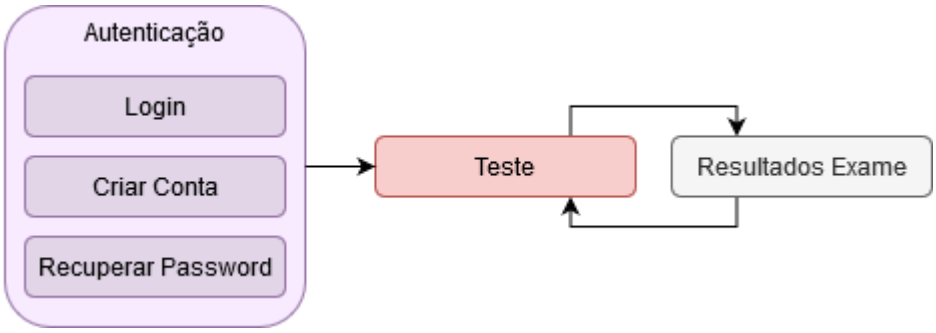

Figura 7-2 - Fluxo da aplicação

```
{
        "Pages": {
                "Login":true ,
                "Results":true ,
        },
        "Source": {
                "Bitalino": {
                        "MACAddress" : "20:15:05:29:21:00",
                        "SampleRate" : 100,
                        "Channels" :[0],
                        "Port" : 0
                }
        },
        "Processing" : {
                "InvertValue": true
        },
        "Sink" : {
                "Graph": {
                        "Type" : "speedMeter"
                }
        }
}
```

Figura 7-3 – Ficheiro de DPL

7.3. Blocos Funcionais

Os blocos funcionais são programados uma vez e depois reutilizados e combinados de diferentes maneiras dependendo do tipo de aplicação que se queira alcançar.

Para esse objetivo, foram desenvolvidas um conjunto de interfaces para criar rápida e corretamente um bloco funcional. O design UML dessas classes é mostrado na

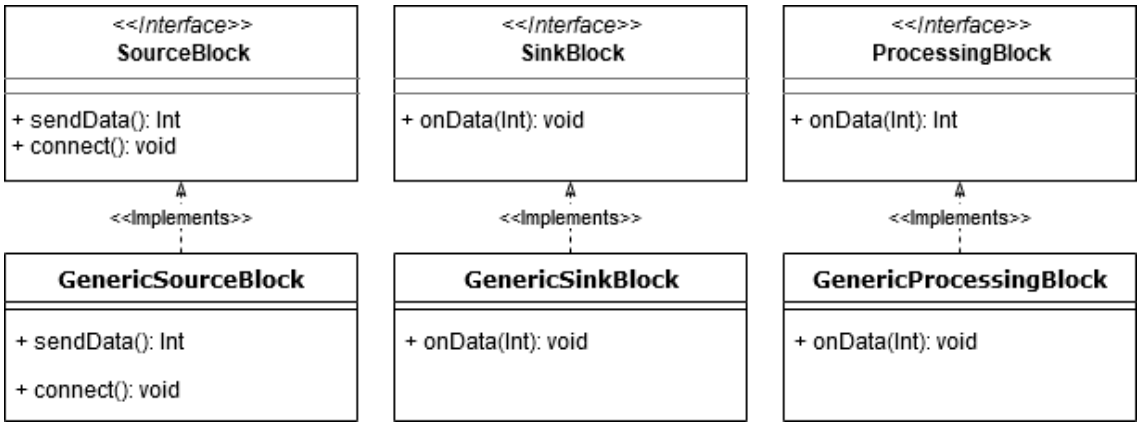

Figura 7-4. O desenvolvimento de novos blocos funcionais é direto usando essas interfaces e estas são suficientemente básicas de forma a permitir inclusão de novos blocos sem problemas e sem perder funcionalidades importantes.

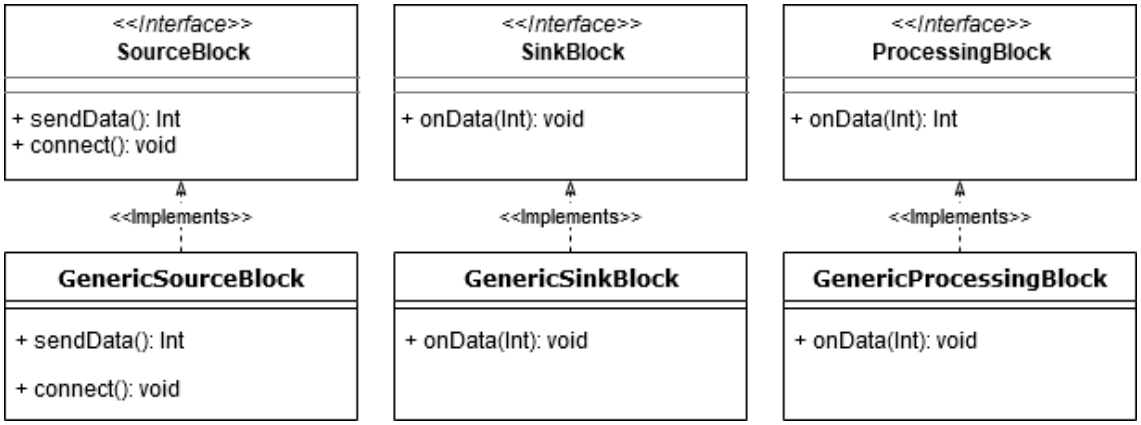

Figura 7-4 - Interfaces para criação de blocos funcionais

7.4. Workflow Manager

O Workflow manager é a peça central do projeto e o responsável pela ligação entre os diferentes blocos. De forma a lidar automaticamente com a ligação dos blocos *sink* foi utilizado o padrão Observer.

O padrão Observer (Figura 7-5) é um padrão de design comportamental que permite definir um mecanismo de assinatura de um para muitos, com o objetivo de notificar vários objetos sobre quaisquer eventos que ocorram no objeto que estão a observar. O objeto principal que é o responsável por receber os dados e distribuí-los é o *observable*. Os objetos que serão atualizados são os *observers*. Neste caso o *workflow*

manager será o *observable*, sempre que este recebe um novo valor proveniente de um bloco *source* irá notificar e atualizar os blocos *sink*, representados como "GenericSink", necessários com este valor.

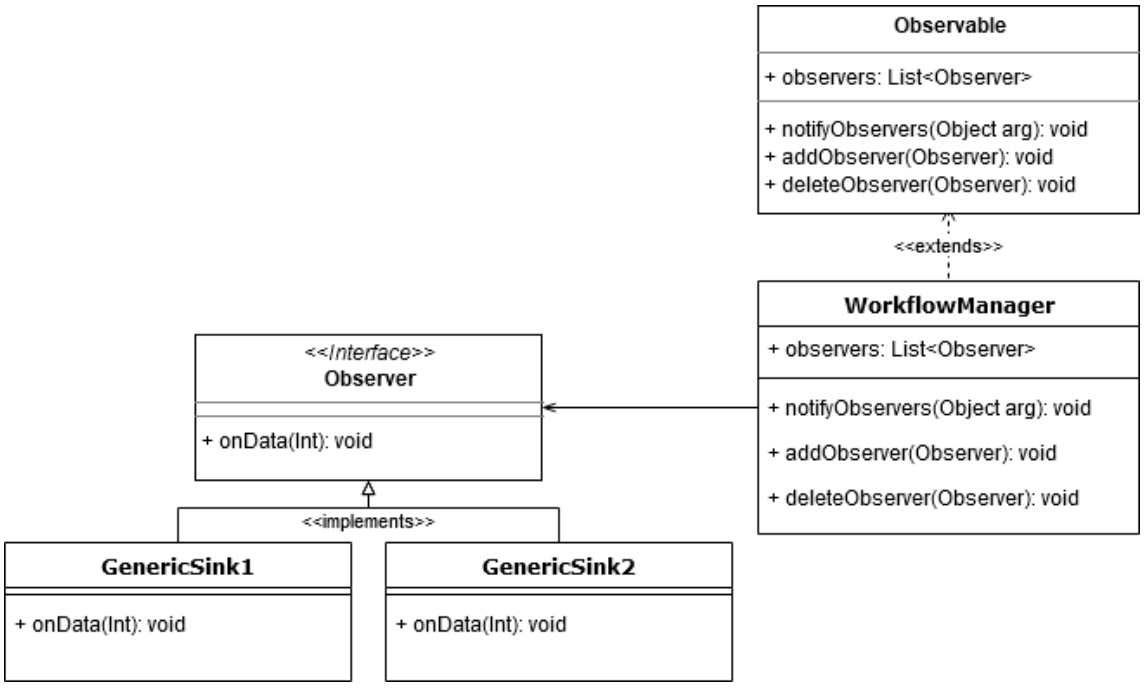

Figura 7-5 - Padrão observer

7.5. Cloud

7.5.1. Dropbox

A Dropbox disponibiliza uma API de forma a facilitar o acesso à sua HTTP-based Core API v2[3]. Para a utilização da API é necessário registar a aplicação na App Console[4], e após feito o registo é obtido o acesso à chave necessária para a comunicação com a API. Adicionalmente, o utilizador pode controlar que acessos são disponibilizados à conta, obtendo assim um maior controlo e segurança pela mesma.

Este necessita da dependência:

```
implementation 'com.dropbox.core:dropbox-core-sdk:3.1.1'
```

Para fazer pedidos à API é necessária uma instância do objeto referente à Dropbox. Nesta instanciação é passada a chave de forma a estabelecer a ligação à conta do

[3] https://www.dropbox.com/developers/documentation/http/documentation

[4] https://www.dropbox.com/developers/apps

utilizador.

```
DbxRequestConfig config =
      DbxRequestConfig.newBuilder("example").build();
DbxClientV2 client = new DbxClientV2(config, ACCESS_TOKEN);
```

Após estabelecida a ligação com a conta do utilizador já é possível fazer o envio de ficheiros para a Dropbox associada. O envio é feito da seguinte forma:

```
try (InputStream in = new FileInputStream("example.txt")) {
    FileMetadata metadata =
            client.files().uploadBuilder("/example.txt")
        .uploadAndFinish(in);
}
```

Desta forma fica disponível mais uma opção de armazenamento *cloud* para o utilizador, em que este apenas terá de gerar uma chave para ter acesso à sua conta. Esta é uma mais valia devido ao grande número de utilizadores que já utilizam a Dropbox, estando assim familiarizado com a ferramenta e não tendo a necessidade de instalar uma nova aplicação.

7.5.2. BrainAnswer

A plataforma BrainAnswer disponibiliza uma REST API e, para comunicar com esta, é necessária uma biblioteca para fazer as comunicações HTTP. Para tratar desta comunicação foi utilizada a biblioteca Fuel[5], entre os diferentes recursos, destaca-se a facilidade e estilo fluente de fazer os pedidos, carregamento de ficheiros e desserialização das respostas.

Este necessita da dependência:

```
implementation 'com.github.kittinunf.fuel:fuel:2.2.0'
```

Para fazer a comunicação com os *endpoints* disponíveis é utilizada a instância `Fuel` ou `FuelManager`, seguido do método necessário para o pedido. Este pedido é feito da seguinte maneira:

```
Fuel.post("https://brainanswer.pt/").responseString(){request, resp
onse, result -> … }
```

[5] https://github.com/kittinunf/fuel

7.6. Autenticação

Com a autenticação podemos saber a identidade de um utilizador, e desta forma é possível armazenar a informação do mesmo de forma a fornecer a mesma experiência mesmo que o utilizador mude de dispositivo.

Para a autenticação recorreu-se ao serviço de *backend* do Firebase. Este disponibiliza um conjunto de SDKs de fácil utilização para diferentes tecnologias, sendo as principais Android, Web e iOS. Este suporta diferentes métodos de autenticação, desde o típico email e password (ou número de telemóvel), até aos provedores de identidade federada como a Google, Facebook, Twitter, etc. Em seguida, as credenciais serão passadas para o Firebase Authentication SDK onde serão verificadas e retornarão uma resposta ao cliente. Caso o pedido seja bem-sucedido, é possível obter informações básicas do perfil do utilizador e controlar o acesso do utilizador aos dados armazenados em outros produtos Firebase.

É necessário adicionar as dependências:

```
implementation 'com.google.firebase:firebase-core:16.0.9'
implementation 'com.google.firebase:firebase-auth:17.0.0'
implementation 'com.google.android.gms:play-services-auth:16.0.1'
implementation 'com.google.firebase:firebase-database:17.0.0'
```

Após adicionadas as dependências o Firebase pode ser utilizado. Para a utilização do mesmo recorre-se a uma instância do FirebaseAuth, onde estão disponíveis todos os métodos necessários às ações associadas à autenticação.

```
mAuth = FirebaseAuth.getInstance()
```

Para a criação de uma conta (Figura 7-6) apenas é necessário fornecer um email e password. Caso estes sejam válidos a criação da conta é realizada. No caso da recuperação da password apenas é necessário fornecer um email e, caso exista no sistema, será enviado um email para proceder à recuperação. No caso de autenticação, se o email e password forem válidos, o utilizador prossegue para a aplicação. Este tem ainda a possibilidade de fazer a autenticação recorrendo à sua conta Google.

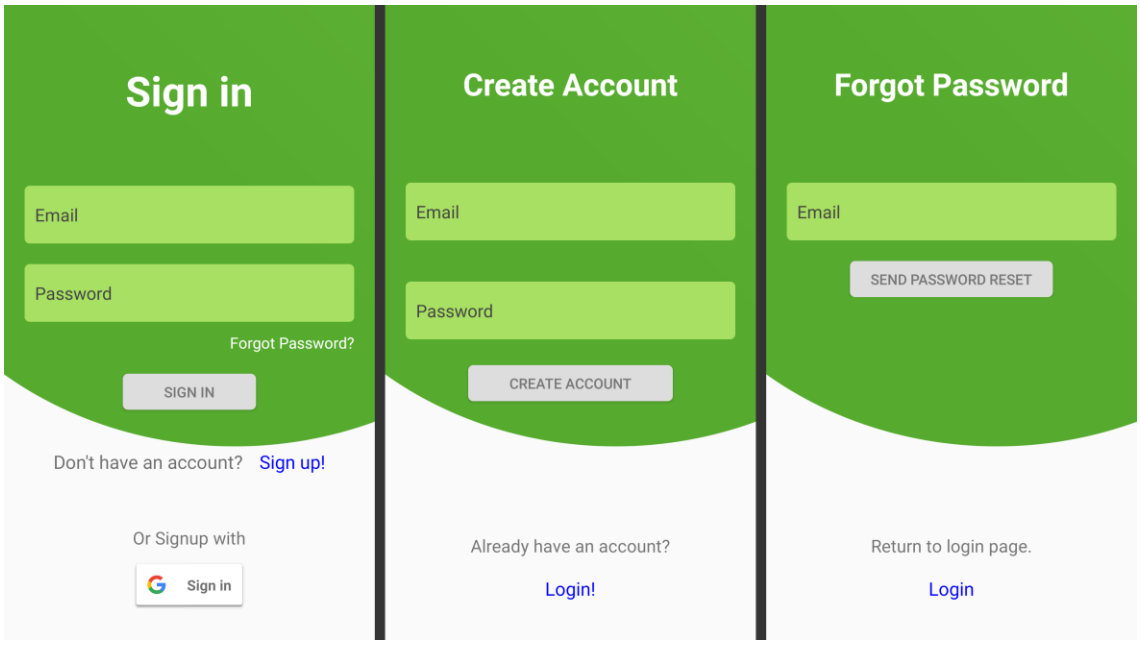

Figura 7-6 - Atividades referentes à autenticação

7.7. Gráficos

A visualização dos dados recebidos é um dos pontos mais importantes no projeto. Sendo este um bloco *sink* responsável por a visualização da informação recebida.

Para a construção dos gráficos foi utilizado o D3.js, uma biblioteca JavaScript projetada para exibir dados digitais na forma de gráficos dinâmicos. Este permite visualizar os dados recorrendo a HTML, SVG e CSS. O D3.js permite um grande controle sobre o resultado visual final.

Para manter a estrutura de módulos, as funções referentes aos gráficos foram desenvolvidas seguindo o padrão de desenvolvimento *Immediately-invoked Function Expressions*. Desta maneira mantem-se um objeto global de desenvolvimento limpo e, de uma maneira simples, é possível isolar declarações de variáveis.

Isto é feito utilizando uma função, transformando-a numa expressão e executando-a imediatamente a seguir a esta ser criada. Recorrendo a este padrão podemos criar objetos onde apenas é possível ter acesso aos métodos e variáveis disponibilizadas.

```
!function () {
    chart = {}
    chart.render = function (values) {
        /* Code */
    }
    chart.update = function (value) {
        /* Code */
    }
    this.chart = chart;
```

```
}();
```

Assim, no objeto global obtemos um objeto referente ao gráfico com os métodos *render* e *update*.

- `.render` – método responsável pela construção e desenho do gráfico. Este pode receber ou não um conjunto de dados; caso receba, os mesmos são desenhados inicialmente no gráfico, caso contrário o gráfico é inicializado sem quaisquer valores desenhados.
- `.update` – método responsável por atualizar ou desenhar no gráfico os novos valores recebidos; este recebe um valor e fará as modificações necessárias ao gráfico de forma a este apresentar o novo valor.

Na Figura 7-7 estão disponíveis os diferentes gráficos disponíveis na aplicação. Os primeiros dois gráficos são para visualização de dados em tempo real, sendo estes atualizados constantemente durante a realização de um teste. O último gráfico é utilizado para a visualização dos resultados do teste realizado, neste estão desenhados todos os valores desde início, portanto não existe atualização do mesmo. Neste último gráfico é possível ainda ampliar as escalas de forma a melhor examinar os valores.

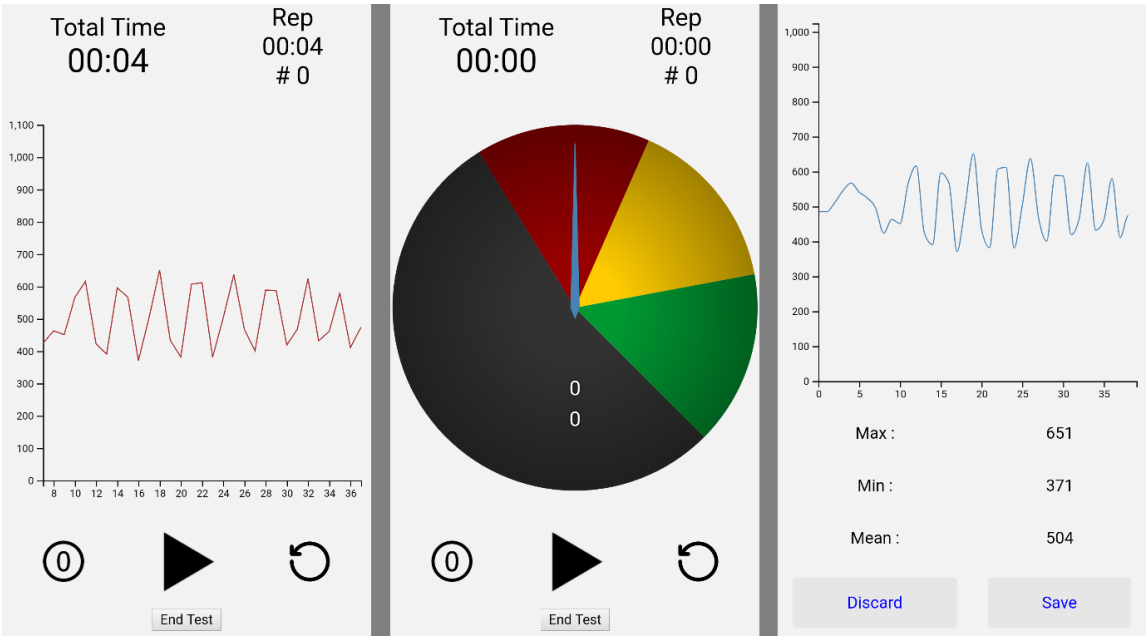

Figura 7-7 - Diferentes gráficos para visualização de dados

Na Figura 7-8 podemos observar a sequencia e a interação que existe entre os diferentes objetos de forma a apresentar os dados no gráfico. A sequencia inicia quando o utilizador dá início à recolha de dados e demonstra as etapas até que estes são

apresentados no gráfico.

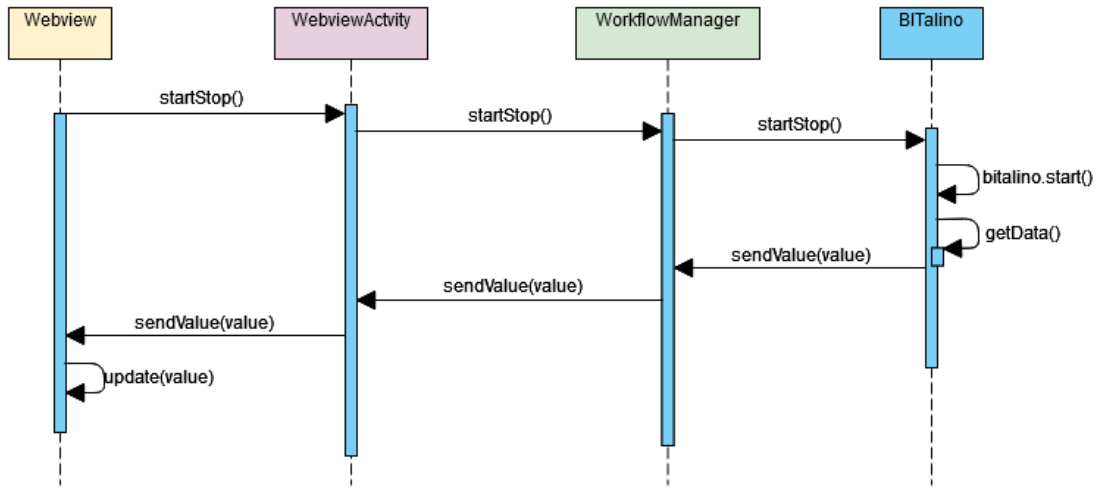

Figura 7-8 – Diagrama de sequência para desenho de dados no gráfico

7.7.1. JSON para Objects

Um dos pontos base do projeto é a inicialização dos blocos através de um ficheiro JSON, portanto é necessário analisar o mesmo de forma a fazer a transformação em classes utilizáveis.

O Klaxon [6] foi a biblioteca utilizada para esta análise. Após adicionar a dependência:

```
implementation 'com.beust:klaxon:5.0.1'
```

Para a serialização apenas é necessário:

```
Klaxon().parse<Class>(jsonFile)
```

Após feita a serialização, obtém-se o conjunto de *Data Classes* referidos no ficheiro de JSON indicado.

7.7.2. Data Classes

Uma *data class* é um conceito que não está ligado a nenhuma linguagem de programação específica; é um padrão para, de uma maneira simples de representar, encapsular e mover informações. Frequentemente, a criação destas classes tem como principal objetivo armazenar dados, pois nestas estão disponíveis algumas

[6] https://github.com/cbeust/klaxon

funcionalidades padrão e funções utilitárias geralmente derivadas dos dados. Estas classes são definidas através da *keyword* `data`:

```
data class User(val name: String, val age: Int)
```

7.8. BITalino API

A interface de programação de aplicações (API) Java do BITalino traz para as aplicações Android todas as funcionalidades destes dispositivos. A API é implementada com base na utilização do padrão de design de fábrica, e fornece uma maneira simples de escolher entre os diferentes tipos de comunicação disponíveis. A classe *BITalinoCommunication* contém os métodos necessários para se comunicar com um dispositivo BITalino, além de receber informações sobre seu estado. Este pode ser acedido através de uma conexão Bluetooth clássica ou usando Bluetooth Low Energy. A classe *BITalinoException* implementa a exceção lançada pela API quando uma condição de erro é atendida ao chamar qualquer uma das funções da API ou métodos de classe.

7.9. Webview

Sendo a aplicação desenhada para tirar proveito da abordagem de aplicações híbridas e desta forma beneficiar da utilização das APIs disponíveis em Android e da facilidade de desenvolvimento de Web, é necessário fazer a ligação entre as duas. Esta ligação é feita a partir de uma *webview* que permite assim incorporar um navegador Web na Aplicação. A *webview* permite a visualização de sites locais que façam uso de JavaScript.

- Configuração da *webview*

Por padrão o JavaScript está desativado numa WebView, no entanto acedendo às configurações este é possível ativar. Isto é feito através do seguinte comando:

```
webView.settings.javaScriptEnabled = true
```

De forma a ajudar no processo de depuração é necessário ativar o mesmo para a *webview*. Além de ativar, reencaminhamos as mensagens obtidas na *webview* para Android para serem posteriormente apresentadas.

```
WebView.setWebContentsDebuggingEnabled(true)
webView.webViewClient = object : WebViewClient() {
    fun onConsoleMessage(consoleMessage: ConsoleMessage){
```

```
        Log.d("Webview: ", consoleMessage.message())
    }
}
```

Para criar a ligação entre Android e JavaScript é necessário criar uma interface para fazer a ligação entre ambos. Após criada a interface, é necessário fazer o `bind` entre estes passando uma instância de classe para ligar ao JavaScript e o nome da interface que no JavaScript possa ser utilizado para aceder à classe. No exemplo a seguir foi utilizado o nome "Android".

```
webView.addJavascriptInterface(this, "Android")
```

Por último é necessário carregar um ficheiro para este ser apresentado na *webview*. Isto é feito da seguinte forma:

```
webView.loadUrl("file:///android_asset/main.html")
```

- Android para Web.

Para chamar funções no lado de JavaScript através de Android apenas é necessário utilizar o método `.loadUrl` na *webView* com o nome da função que queremos executar. No entanto este deve ser executado recorrendo a um Runnable para garantir que é chamado na mesma *thread* que a *webview*.

```
fun toWebView() {
    webView.post {
        run {
            webView.loadUrl("javascript: runOnJavascript();")
        }
    }
}
```

- Web para Android.

Para executar funções em Android a partir de JavaScript, primeiro é necessário recorrer ao nome da interface anterior definida, neste caso "Android" e de seguida o nome da função a executar. Do lado do Android para a função poder ser chamada deverá ter a anotação `@JavascriptInterface`.

```
function toAndroid(message){
    Android.runOnAndroid(message);
}

@JavascriptInterface
fun runOnAndroid(message: String) {
    println(message)
}
```

7.10. Comunicação com UI

Em Android todas as operações feitas na interface de utilizador são realizadas num único *thread*, também conhecido como principal ou UI *thread*. Todos os eventos necessários a serem realizados são processados por uma instância da classe Looper. Caso seja necessário atualizar a UI a partir de outro *thread*, é necessário fazer a sincronização entre ambos. Para garantir esta sincronização utiliza-se à classe Handler. Esta permite a comunicação com o *thread* da UI a partir de qualquer outro *thread* em segundo plano, garantindo assim a sincronização de ambos. O Handler permite enviar e processar objectos do tipo Message e Runnable associados ao um MessageQueue da thread [52].

Para utilizar o Handler é necessário instanciar com base no Looper principal.

```
val handler: Handler = Handler(Looper.getMainLooper()){}
```

Após instanciado, na classe que pretendemos receber os dados deverá ser sobreposto o método handleMessage(). Este método é chamado quando é recebida uma nova mensagem no mesmo thread.

```
handler = object : Handler(Looper.getMainLooper()) {
    override fun handleMessage(inputMessage: Message) {
        // Code ...
    }
}
```

Na classe de onde as mensagens serão enviadas, apenas é necessário recorrer ao método sendMessage(). As mensagens são do tipo Message e deverão conter um Bundle no seu conteúdo.

```
val mss = Message()
val bundle = Bundle()
bundle.putString("data","data")
mss.data = bundle
handler.sendMessage(mss)
```

7.11. Armazenamento de Dados

O Android oferece várias opções para guardar os dados relativos às aplicações. O tipo de armazenamento escolhido deve ser selecionado tendo em conta necessidades específicas, tais como o tipo de dados que é necessário armazenar, quanto espaço estes ocupam e se essas informações devem ser privadas ou acessíveis por outras aplicações e pelo utilizador do dispositivo.

As diferentes opções de armazenamento de dados disponíveis são:

- Armazenamento interno de arquivos: armazena os dados privados da aplicação no sistema de arquivos do dispositivo; desta forma não ficam disponíveis para serem acedidos à exceção da própria aplicação.

- Armazenamento de arquivos externo: armazena os dados no sistema de armazenamento externo compartilhado; desta forma estes ficam acessíveis permitindo assim a partilha entre utilizadores e outras aplicações.
- *Shared preferences*: armazena dados primitivos privados na forma de pares de chave-valor.
- Base de dados: armazena os dados estruturados numa base de dados privada.

Devido à necessidade de aceder a ficheiros e estes estarem disponíveis para o utilizador, o principal método de armazenamento utilizado foi o armazenamento externo. Desta forma, garantimos que o utilizador pode aceder ao ficheiro JSON de configuração de forma simples e que este já estará habituado devido à estrutura de pastas utilizada em qualquer sistema. Além disso, os ficheiros gravados encontram-se também armazenados desta forma, permitindo assim um fácil acesso aos mesmos e a possibilidade de serem acedidos por outras aplicações, como por exemplo aplicações onde é permitida a partilha de ficheiros. Este tipo de armazenamento não é apagado quando a aplicação é desinstalada, portanto deverá apenas conter o que é realmente necessário de forma a não poluir o sistema de ficheiros do utilizador.

Com as diferentes versões de Android, o caminho para o armazenamento pode ser diferente. Para evitar problemas o acesso ao armazenamento externo deve ser feito recorrendo a caminhos relativos. Para aceder desta forma deve ser utilizado o:

```
Environment.getExternalStorageDirectory()
```

O armazenamento de dados requer permissões por parte do utilizador. Quando é permitida a permissão de `WRITE` é também permitida a de `READ`.

As permissões requeridas são as seguintes:

```
<uses-permission
android:name="android.permission.READ_EXTERNAL_STORAGE"/>
<uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

7.12. Desempenho

O desempenho da aplicação é algo bastante importante e, dependendo de quão intensivos são os fluxos de dados, estes ao passar pela cadeia de blocos podem criar problemas no desempenho. Em testes de desemprenho realizados, taxas de amostragem acima dos 100Hz são problemáticas e podem causar problemas de desempenho. Estes

problemas passam pela atualização da interface de aquisição ficar atrasada em relação ao ritmo de envio de dados e, nos casos mais extremos, até bloquear por completo, o que tem um impacto indesejável na usabilidade da aplicação pois os dados recebidos não serão mostrados, havendo assim perdas de informação.

Estes problemas tendem a desaparecer com o aumento da capacidade computacional em dispositivos móveis.

7.12.1. Down sampling

Downsampling ou decimação é o processo que permite reduzir a taxa de amostragem de um sinal para efeitos de processamento. É geralmente utilizado para reduzir o volume de dados com que é necessário lidar [53].

Por vezes, a taxa de amostragem pode não ser possível de alterar ou apenas apresentar valores fixos. No caso do BITalino a taxa de amostragem apenas pode ter os valores de 1, 10, 100 ou 1000 Hz. Segundo o teorema de Nyquist (Figura 7-9) de modo a que não exista perda de informação, um sinal analógico pode ser convertido em digital através de um processo de amostragem periódica, utilizando intervalos de tempo iguais, garantindo que a taxa de amostragem deve ser " superior a duas vezes o componente de frequência mais alta no sinal analógico". Por exemplo, no caso do BITalino, para fazer a amostragem de um sinal de ECG cuja banda passante está compreendida entre os 0.5-40 Hz, seria necessário utilizar pelo menos uma taxa de amostragem de 100Hz. Contudo, esta taxa de amostragem gera um volume de dados superior ao que se consegue representar no ecrã, gerando conteúdo redundante.

Se a informação redundante não for utilizada por qualquer bloco, esta deverá ser enviada para outros blocos. De maneira a evitar um fluxo de dados redundantes, o processo de decimação é feito o mais cedo possível na cadeia de processamento, sendo este ponto o bloco em que os dados são recebidos. Existem diferentes técnicas e algoritmos para realizar a decimação, que podem ser implementadas como blocos normais de processamento.

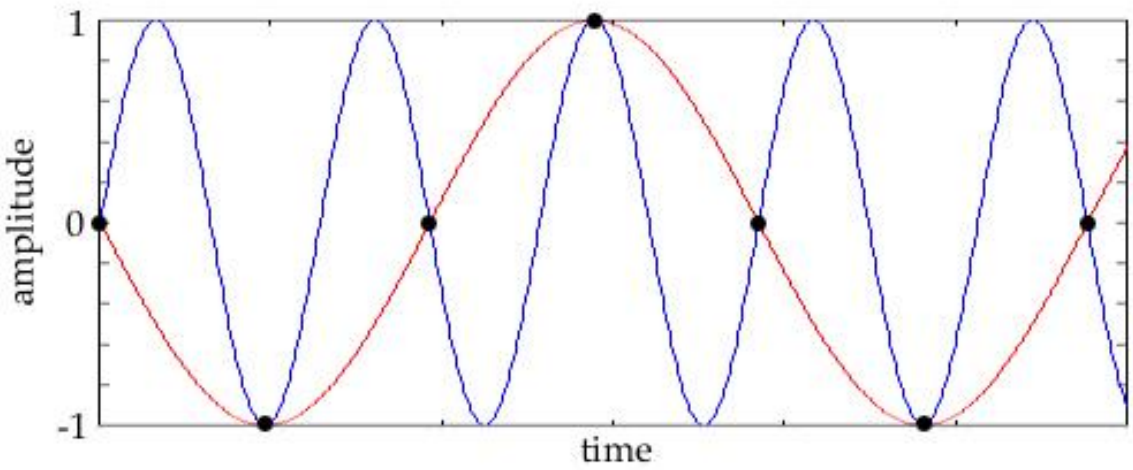

Figura 7-9 – Teorema de Nyquist ilustrado com uma duas ondas seno.

7.12.2. Desenho dos Gráficos

A parte gráfica é o bloco final mais exigente computacionalmente. De forma a evitar o excesso de carga no smartphone, apenas deverão ser enviados para este bloco os valores absolutamente necessários, portanto todos os cálculos feitos com base nos dados deverão ser feitos previamente. Para garantir o maior desempenho no momento da atualização dos dados no gráfico, apenas é feita a alteração no gráfico necessária para representar os novos dados, evitando o redesenho do gráfico por inteiro.

Para evitar um maior peso nos gráficos que necessitam de atualização em tempo real dos valores, estes devem ser os mais simples dentro do possível. Isto é, apenas deverá ser desenhado o necessário e evitar sombras ou outros efeitos que possam aumentar o tempo no desenho do mesmo.

Capítulo 8
Conclusão e Trabalho futuro

Neste capítulo, os resultados obtidos nesta tese são resumidos e discutidos. São destacadas as contribuições do mesmo e quais poderão ser os futuros passos da aplicação.

8.1. Conclusão

Neste relatório apresentou-se o KBIT, uma *framework* para desenvolvimento rápido de aplicações híbridas para dispositivos Android.

A estrutura do KBIT é baseada na arquitetura MVP separando a aplicação em diferentes aspetos lógicos permitindo desenvolvimento rápido e, no caso de utilização por outros desenvolvedores, facilidade de adaptação de desenvolvimento. De forma a se obter uma aplicação do estilo *no-code* com facilidade de configuração recorre-se a um DPL para definir a estrutura da aplicação e inicializar os blocos funcionais. A arquitetura baseada na construção por blocos funcionais permite que o utilizador combine blocos funcionais e defina assim o comportamento da aplicação, obtendo grande versatilidade e abstração dos módulos. O utilizador pode alterar alguns dos componentes da interface permitindo adaptação da mesma para diferentes casos de uso. A interface da aplicação é feita recorrendo a tecnologias web, conseguindo assim uma fácil configuração do aspeto da mesma e permitindo a novos desenvolvedores facilidade na alteração da mesma. Adicionalmente, é permitido o armazenamento de dados num servidor remoto disponível na *cloud* e a integração com ferramentas que permitem o armazenamento e a visualização destes dados na mesma.

Desta forma obteve-se uma *framework* para aplicações híbridas que permite abstração de sensores, facilidade de integração de novas funcionalidades, interface atual e facilmente alterável. Como foco foi ainda obtida uma solução para a avaliação autónoma da fadiga por parte dos utilizadores mais idosos.

8.2. Trabalho futuro

Futuramente pretende-se acrescentar novas funcionalidade de forma a cobrir um

maior número de alternativas que a aplicação poderá apresentar. Certas funcionalidades que poderão ser adicionadas passam por:

- Suporte a maior número de sensores.
- Adicionar blocos de forma a tratar da interação com diferentes sensores.
- Adicionar novos gráficos de forma a cobrir novas necessidades que possam surgir.
- Suporte de notas nos resultados. Permitir que sejam adicionadas notas no gráfico dos resultados e que estas sejam adicionadas com o tempo ao ficheiro gravado.
- Uso de múltiplos sensores simultaneamente. Permitir a conexão Bluetooth, ou outra, com múltiplos sensores e registar os dados dos mesmos.
- Testes de implementação em diferentes smartphones. Devido às diferenças que existem entre os smartphones e marcas, é importante realizar testes com um maior número de dispositivos.

8.3. Contributos

Durante o desenvolvimento deste trabalho foi criada uma aplicação especificamente para dinamometria, denominada de GripIT [54]. Esta aplicação tinha algumas diferenças, tais como, não existia a possibilidade de configuração através do ficheiro DPL, as configurações eram realizadas num menu de definições, apenas continha um gráfico e não tinha a opção de armazenamento cloud.

Utilização da aplicação GripIT no projeto FORTO[7].

Utilização da aplicação GripIT no Hospital de Santa Maria para implementação do protocolo de Ewing[8].

O projeto será disponibilizado em formato open source de forma a continuar o exemplo e apoiar o projeto BITalino. Desta forma novos utilizadores terão disponíveis mais recursos que poderão usar como exemplo no início dos seus projetos.

[7] https://forto-aal.eu/

[8] https://www.hkmj.org/system/files/hkmj1908sp5p9.pdf

Bibliografia

[1] World Health Organization, “World report on ageing and health 2015.”

[2] R. W. Bohannon, “Dynamometer measurements of hand-grip strength predict multiple outcomes,” *Percept. Mot. Skills*, vol. 93, no. 2, pp. 323–328, 2001.

[3] R. W. Bohannon, “Hand-grip dynamometry predicts future outcomes in aging adults,” *J. Geriatr. Phys. Ther.*, vol. 31, no. 1, pp. 3–10, 2008.

[4] A. J. Cruz-Jentoft *et al.*, “Sarcopenia: European consensus on definition and diagnosis,” *Age Ageing*, vol. 39, no. 4, pp. 412–423, 2010.

[5] K. Norman, N. Stobäus, M. C. Gonzalez, J. D. Schulzke, and M. Pirlich, “Hand grip strength: Outcome predictor and marker of nutritional status,” *Clinical Nutrition*, vol. 30, no. 2. pp. 135–142, 2011.

[6] H. Syddall, C. Cooper, F. Martin, R. Briggs, and A. A. Sayer, “Is grip strength a useful single marker of frailty?,” *Age Ageing*, vol. 32, no. 6, pp. 650–656, 2003.

[7] R. Cooper *et al.*, “Objective measures of physical capability and subsequent health: A systematic review,” *Age and Ageing*, vol. 40, no. 1. pp. 14–23, 2011.

[8] R. W. Bohannon, J. Bear-Lehman, J. Desrosiers, N. Massy-Westropp, and V. Mathiowetz, “Average grip strength: A meta-analysis of data obtained with a Jamar dynamometer from individuals 75 years or more of age,” *J. Geriatr. Phys. Ther.*, 2007.

[9] I. Bautmans, O. Onyema, K. Van Puyvelde, S. Pleck, and T. Mets, “Grip work estimation during sustained maximal contraction: Validity and relationship with dependency and inflammation in elderly persons,” *J. Nutr. Heal. Aging*, vol. 15, no. 8, pp. 731–736, 2011.

[10] World Health Organization, *Global strategy and action plan on ageing and health*. 2017.

[11] G. Eysenbach, “What is e-health?,” *Journal of Medical Internet Research*, vol. 3, no. 2, pp. 1–5, 2001.

[12] 1615 L St NW, S. 800Washington, and D. C. 20036USA202-419-4300 \textbar M.-857-8562 \textbar F.-419-4372 \textbar M. Inquiries, “Social {Media} {Use} {Continues} to {Rise} in {Developing} {Countries},” *Pew Res. Center’s Glob. Attitudes Proj.*, 2018.

[13] J. A. Blaya, H. S. F. Fraser, and B. Holt, “E-health technologies show promise in developing countries,” *Health Aff.*, vol. 29, no. 2, pp. 244–251, 2010.

[14] A. W. Martinez, S. T. Phillips, E. Carrilho, S. W. Thomas, H. Sindi, and G. M. Whitesides, “Simple telemedicine for developing regions: Camera phones and paper-based microfluidic devices for real-time, off-site diagnosis,” *Anal. Chem.*, vol. 80, no. 10, pp. 3699–3707, 2008.

[15] W. A. Kaplan, “Can the ubiquitous power of mobile phones be used to improve health

outcomes in developing countries?," *Global. Health*, vol. 2, 2006.

[16] M. Mühldorfer-Fodor *et al.*, "Grip force monitoring on the hand: Manugraphy system versus Jamar dynamometer," *Arch. Orthop. Trauma Surg.*, vol. 134, no. 8, pp. 1179–1188, 2014.

[17] J. Bear-Lehman and B. C. Abreu, "Evaluating the hand: Issues in reliability and validity," *Physical Therapy*, vol. 69, no. 12. pp. 1025–1033, 1989.

[18] K. Denby, G. Nelson, and C. A. Estrada, "Bedside hand grip assessment with the sphygmomanometer," *Journal of General Internal Medicine*, vol. 28, no. 10. p. 1381, Oct-2013.

[19] F. P. Medical, "Jamar ® Smart Hand Dynamometer," 2015. [Online]. Available: https://www.physioparts.co.uk/jamar-smart-hand-dynamometer-wireless. [Accessed: 10-Oct-2019].

[20] "microFET Medical Dynamometers and Inclinometers." [Online]. Available: https://hogganscientific.com/microfet-dynanomenters-inclinometers/. [Accessed: 10-Oct-2019].

[21] A. Kailas, C. C. Chong, and F. Watanabe, "From mobile phones to personal wellness dashboards," *IEEE Pulse*, vol. 1, no. 1, pp. 57–63, 2010.

[22] M. N. K. Boulos *et al.*, "CAALYX: A new generation of location-based services in healthcare," *Int. J. Health Geogr.*, vol. 6, 2007.

[23] F. J. M. Meiland *et al.*, "COGKNOW development and evaluation of an ict-device for people with mild dementia," in *Studies in Health Technology and Informatics*, 2007, vol. 127, pp. 166–177.

[24] M. Mulvenna *et al.*, "Designing & evaluating a cognitive prosthetic for people with mild dementia," in *ECCE 2010 - European Conference on Cognitive Ergonomics 2010: The 28th Annual Conference of the European Association of Cognitive Ergonomics*, 2010.

[25] L. Warmerdam *et al.*, "Innovative ICT solutions to improve treatment outcomes for depression: The ICT4Depression project," *Annu. Rev. CyberTherapy Telemed.*, vol. 10, pp. 339–343, 2012.

[26] A. Rocha *et al.*, "ICT4Depression: Service oriented architecture applied to the treatment of depression," in *Proceedings - IEEE Symposium on Computer-Based Medical Systems*, 2012.

[27] "iPROGNOSIS." [Online]. Available: http://www.i-prognosis.eu/. [Accessed: 10-Oct-2019].

[28] M. Cânovas, H. Silva, A. Lourenço, and A. Fred, "MobileBIT: A framework for mobile interaction recording and display," *Heal. 2013 - Proc. Int. Conf. Heal. Informatics*, pp. 366–369, 2013.

[29] “Operating System Market Share Worldwide,” 2020. [Online]. Available: https://gs.statcounter.com/os-market-share. [Accessed: 14-Sep-2020].

[30] Google, “Android distribution dashboard,” 2018. [Online]. Available: https://developer.android.com/about/dashboards.

[31] “Kotlin FAQ,” 2019. [Online]. Available: https://kotlinlang.org/docs/reference/faq.html. [Accessed: 10-Oct-2019].

[32] F. Lardinois, “Kotlin is now Google’s preferred language for Android app development,” *TechCrunch*, 2019. [Online]. Available: http://social.techcrunch.com/2019/05/07/kotlin-is-now-googles-preferred-language-for-android-app-development/. [Accessed: 10-Oct-2019].

[33] StackOverflow, “Developer Survey Results 2019,” 2019. [Online]. Available: https://insights.stackoverflow.com/survey/2019?utm_source=Iterable&utm_medium=email&utm_campaign=dev-survey-2019. [Accessed: 10-Oct-2019].

[34] M. Pilgrim, *HTML5: Up and Running*. O’Reilly Media, 2010.

[35] D. McFarland, *CSS3: The Missing Manual*. O’Reilly Media, 2012.

[36] C. Lindley, *jQuery Cookbook*. O’Reilly Media, 2009.

[37] “d3.js.” [Online]. Available: https://d3js.org/.

[38] “Flot.” [Online]. Available: https://github.com/flot/flot.

[39] Google, “Google Charts.” [Online]. Available: https://developers.google.com/chart.

[40] N. Gok, Nizamettin; Khanna, *Building Hybrid Android Apps with Java and JavaScript*. O’Reilly Media, 2013.

[41] “JSON.” [Online]. Available: https://www.json.org/json-en.html.

[42] H. P. Da Silva, A. Fred, and R. Martins, “Biosignals for everyone,” *IEEE Pervasive Comput.*, 2014.

[43] PLUX, “BITalino (r)evolution Board Kit Data Sheet.” [Online]. Available: https://bitalino.com/datasheets/REVOLUTION_BITalino_Board_Kit_Datasheet.pdf.

[44] J. Wu, L. Ping, X. Ge, W. Ya, and J. Fu, “Cloud storage as the infrastructure of Cloud Computing,” in *Proceedings - 2010 International Conference on Intelligent Computing and Cognitive Informatics, ICICCI 2010*, 2010.

[45] “Dropbox,” 2019. [Online]. Available: https://www.dropbox.com/. [Accessed: 10-Oct-2019].

[46] S. Marx, “Dropbox generate access token,” 2014. [Online]. Available: https://blogs.dropbox.com/developers/2014/05/generate-an-access-token-for-your-own-account/. [Accessed: 10-Oct-2019].

[47] “BrainAnswer,” 2019. [Online]. Available: https://ds.brainanswer.pt/.

[48] “BrainAnswer Docs,” 2019. [Online]. Available: https://brainanswer.pt/docs/.

[49] O. Mayor and E. Maestre, “REPOVIZZ : A MULTIMODAL ON-LINE DATABASE AND BROWSING TOOL FOR MUSIC PERFORMANCE RESEARCH Music technology Group ,” *Conf. Int. Soc. Music Inf. Retr.*, pp. 1–2, 2009.

[50] “Repovizz API,” 2017. [Online]. Available: https://repovizz.upf.edu/repo/api-doc/index.html. [Accessed: 10-Oct-2019].

[51] A. Kirhenstein, “Mobile Reachability. Rules of Thumb,” 2015. [Online]. Available: https://medium.com/@Draward/mobile-reachability-rules-of-thumb-ce37dd0cd3ad. [Accessed: 10-Oct-2019].

[52] Google, “Handler.” [Online]. Available: https://developer.android.com/reference/kotlin/android/os/Handler#sendMessage(android.os.Message).

[53] M. Parker, *Digital Signal Processing 101: Everything You Need to Know to Get Started*, 2nd ed. Newnes, 2017.

[54] P. Costa, M. Rocha, R. Baptista, and H. Silva, “GripIT: A Mobile Isometric Handgrip Test for Evaluation of Autonomic Cardiovascular Reflexes in Non-Clinical Applications,” 2019.

Anexo I

Resultados de testes de gráficos

No anexo I encontram-se os resultados referentes aos testes realizados no capítulo 6.2 User Interface - Bottom vs Top. No seguinte anexo pode-se encontrar os resultados para as três ferramentas testadas, com os valores para os gráficos de linhas e para o gráfico de barras. Foram registados os tempos de 5 execuções e calculados a média e desvio padrão para os mesmos.

Resultados do Telemóvel

		Flot			D3.js			Google Charts		
	Nº Dados	Tempo (ms)	Média (ms)	Desvio Padrão (ms)	Tempo (ms)	Média (ms)	Desvio Padrão (ms)	Tempo (ms)	Média (ms)	Desvio Padrão (ms)
Gráfico de linhas	100	150.68	148.6538	4.727	35.38	38.315	2.463	429.035	428.4416	4.375
		141.355			40.254			423.366		
		154.301			36.857			424.803		
		148.333			41.381			433.958		
		148.6			37.703			431.046		
	1000	175.769	168.3208	8.807	47.354	50.1244	3.848	903.627	4451.12	19.554
		169.065			56.1			862.688		
		176.385			47.483			913.506		
		165.534			51.931			887.613		
		154.851			47.754			883.686		
	10000	342.565	340.8912	15.795	167.488	170.3606	7.349	3048.973	3063.2362	50.406
		365.434			182.271			3074.567		
		330.189			162.343			3066.517		
		324.25			170.598			2992.989		
		342.018			169.103			3133.135		
	Nº Dados	Tempo (ms)	Média (ms)	Desvio Padrão (ms)	Tempo (ms)	Média (ms)	Desvio Padrão (ms)	Tempo (ms)	Média (ms)	Desvio Padrão (ms)
Gráfico de barras	50	150.343	153.1764	6.796	64.734	65.6038	2.308	392.318	1932.738	11.534
		146.744			66.594			388.084		
		164.659			66.481			370.432		
		152.781			62.08			400.902		
		151.355			68.13			381.002		
	100	154.974	156.2606	4.397	89.588	86.3518	4.664	437.993	449.3542	9.676
		160.911			90.053			453.996		
		153.295			78.583			446.076		
		160.814			85.732			445.321		
		151.309			87.803			463.385		
	1000	577.784	588.1812	14.846	467.682	477.0364	9.537	1134.17	1136.7314	29.589
		593.562			484.863			1163.984		
		568.19			471.466			1093.336		
		604.697			471.673			1164.72		
		596.673			489.498			1127.447		

		Flot			D3.js			Google Charts		
	Nº Dados	Tempo (ms)	Média (ms)	Desvio Padrão (ms)	Tempo (ms)	Média (ms)	Desvio Padrão (ms)	Tempo (ms)	Média (ms)	Desvio Padrão (ms)
Gráfico de linha	100	22.128	24.6228	3.205	5.468	5.9662	0.5	84.317	87.1954	2.56
		22.528			5.561			85.748		
		23.115			5.895			87.069		
		25.48			6.223			87.703		
		29.863			6.684			91.14		
	1000	23.561	28.0992	4.521	6.745	7.6862	0.937	170.122	924.478	8.769
		25.593			7.102			185.784		
		27.499			7.577			185.991		
		28.364			7.819			189.623		
		35.479			9.188			192.958		
	10000	53.682	55.3578	1.582	24.802	25.842	1.05	373.409	409.9146	30.682
		57.736			25.111			401.656		
		54.622			25.779			402.049		
		54.665			26.015			414.806		
		56.084			27.503			457.653		
	Nº Dados	Tempo (ms)	Média (ms)	Desvio Padrão (ms)	Tempo (ms)	Média (ms)	Desvio Padrão (ms)	Tempo (ms)	Média (ms)	Desvio Padrão (ms)
Gráfico de barras	50	23.273	24.713	1.109	6.9	8.1418	0.738	41.289	209.682	0.744
		24.152			8.147			41.322		
		24.73			8.267			41.759		
		25.175			8.679			42.238		
		26.235			8.716			43.074		
	100	27.774	29.1404	1.302	8.725	9.5168	0.68	70.666	72.998	2.008
		28.292			9.126			71.567		
		29.088			9.318			72.745		
		29.375			10.009			74.498		
		31.173			10.406			75.514		
	1000	34.456	36.5684	1.615	34.467	36.8526	1.917	142.916	146.5794	2.521
		35.78			35.655			146.222		
		36.312			36.948			146.467		
		37.762			37.736			147.356		
		38.532			39.457			149.936		