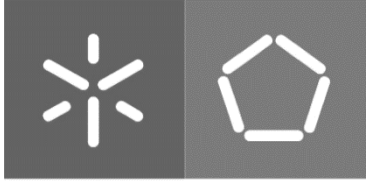

Universidade do Minho
Escola de Engenharia

André Filipe Silva Teixeira

Sistema de monitorização de sinais fisiológicos para pessoas com transtorno mental

Dissertação de Mestrado

Mestrado Integrado em Engenharia

Eletrónica Industrial e Computadores

Trabalho efetuado sob orientação do

Professor Doutor João Miguel Clemente de Sena Esteves

dezembro de 2020

Agradecimentos

Após dar por concluído este trabalho não posso deixar de expressar um sincero agradecimento a todos aqueles que me ajudaram e tornaram possível a sua realização:

Ao meu orientador Professor Doutor João Sena Esteves por todo o acompanhamento, apoio, motivação e disponibilidade demonstrados ao longo do trabalho. Agradeço todas as ideias e sugestões que ajudaram na valorização deste trabalho.

À Professora Doutora Filomena Soares que não estando oficialmente como orientadora desta dissertação acompanhou de igual forma, revelando sempre uma grande disponibilidade e colaboração.

À IPSS Recovery que disponibilizou três *wristband* E4 Empatica que foram utilizadas para a recolha de dados fisiológicos.

Aos Professores Doutores Celina Leão, Vítor Carvalho e Demétrio Matos pelas sugestões e ideias que ajudaram a aperfeiçoar esta dissertação.

Aos meus colegas de laboratório, em especial ao Vinícius Silva e ao Pedro Cunha, por todo o apoio e esclarecimento de dúvidas.

Aos meus pais, irmão, restante família e aos meus amigos por todo o incentivo, carinho e conselhos no decorrer do meu percurso académico, sem a sua ajuda nada disto seria possível.

DECLARAÇÃO DE INTEGRIDADE

Declaro ter atuado com integridade na elaboração do presente trabalho académico e confirmo que não recorri à prática de plágio nem a qualquer forma de utilização indevida ou falsificação de informações ou resultados em nenhuma das etapas conducente à sua elaboração.

Mais declaro que conheço e que respeitei o Código de Conduta Ética da Universidade do Minho.

Resumo

Nos últimos anos, tem-se verificado a vulgarização de *wearables* (dispositivos vestíveis) tais como *smartbands*, cada vez mais evoluídos tecnologicamente e a custos acessíveis. Estes equipamentos permitem fazer a monitorização contínua de sinais fisiológicos, oferecendo um vasto leque de oportunidades para o desenvolvimento de aplicações de monitorização e avaliação de saúde.

No âmbito desta dissertação desenvolveu-se o protótipo de utilização laboratorial de um sistema de monitorização remota *online* de sinais fisiológicos para pessoas com transtorno mental. Para a recolha dos dados fisiológicos foi escolhida a *wristband* E4 da empresa Empatica que, por ser um dispositivo portátil e robusto, de reduzidas dimensões, sem interface para o paciente e discreto se adequa ao público alvo em questão. O sistema desenvolvido permite que os dados recolhidos sejam transmitidos para a *Cloud* e, com auxílio de um *smartphone*, uma aplicação Android funciona como *gateway*. Assim, é possível a receção dos dados por *Bluetooth Low Energy* (BLE) e a sua transmissão por *Wi-Fi* ou rede móvel 4G. A aplicação Android permite, ainda, a seleção dos dados que se pretende transmitir e a sua visualização numérica. Uma aplicação *Web* responsiva permite visualizar, na forma de gráficos, todos os dados enviados para a *Cloud* e também possibilita a consulta de sessões prévias. Desta forma, implementou-se um sistema de monitorização remoto que, para além de ajudar os cuidadores ou familiares a efetuar uma avaliação à distância dos pacientes, permite um atendimento mais rápido e imediato, excluindo a necessidade de deslocações e, consequentemente reduzindo os custos associados.

O sistema desenvolvido foi testado laboratorialmente, de forma a avaliar todo o processo de monitorização desde a aquisição dos dados até à visualização final na interface desenvolvida. Foram realizados testes para avaliar o atraso das comunicações e compararam-se os resultados entre a comunicação por *Wi-Fi* e 4G. Por fim, testou-se a segurança e *performance* do sistema. Os resultados obtidos demonstraram um atraso menor na comunicação por *Wi-Fi* relativamente à comunicação por 4G, verificando-se também que não existiram perdas de dados. No que diz respeito à segurança, confirmou-se que apenas os utilizadores autenticados através do email institucional é que têm acesso à visualização dos dados.

Palavras-Chave: Sistema de monitorização, sinais fisiológicos, *wearables*, Empatica E4 *wristband*, *gateway*, aplicação Android, *Cloud*, aplicação *Web*, *Wi-Fi*, 4G

Abstract

Over the past few years there has been a widespread use of wearable devices such as smartbands, which are getting more evolved from a technological point of view and at affordable costs. These devices allow the continuous monitoring of physiological signals, offering a wide range of opportunities for the development of health monitoring and evaluation applications.

Within the scope of this dissertation, the prototype for laboratory use of a system with the purpose of remote online monitoring physiological signals was developed for people with mental or disorders. In order to perform the collection of physiological data, the E4 wristband from Empatica was chosen. The developed system allows the collected data to be transmitted to the Cloud and, with the help of a smartphone, an Android application works as a gateway. This enables the reception of data via Bluetooth Low Energy (BLE) and its transmission via Wi-Fi or 4G mobile network. The Android application also allows selecting the data to be transmitted and its numerical visualization. Through a responsive Web application, all data sent to the Cloud can be viewed, in graphics, and previous sessions can be accessed. Thus, a monitoring system that helps the caregivers and family members in long distance evaluations of the patients was implemented. This program also allows faster and immediate care, eliminating the majority of travel needs and reducing associated costs. The developed system was tested in a laboratory, in order to evaluate the entire monitoring process, from data acquisition to final visualization in the interface. Tests were carried out to assess the delay in communications and the results were compared between communications over Wi-Fi and 4G. Finally, the safety and performance of the system were tested. The archived results showed a lower delay in Wi-Fi communications comparatively to 4G communications, and no data losses. Regarding security, it was confirmed that only users authenticated through an institutional email have access to the data visualization.

Keywords: Monitoring system, physiological signals, wearables, E4 wristband, gateway, Android application, Cloud, Web application, Wi-Fi, 4G

Índice de Conteúdos

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

ANS	*Autonomic Nervous System*
API	*Application Programming Interface*
APs	*Acess Points*
BaaS	*Backend as a Service*
BAN	*Body Area Network*
BLE	*Bluetooth Low Energy*
Bpm	Batimentos por minuto
BVP	*Blood Volume Pulse*
CSS	*Cascading Style Sheets*
DGS	Direção Geral da Saúde
ECG	Eletrocardiograma
EDA	*ElectroDermal Activity*
EEG	*ElectroEncephaloGram*
FDA	*Food and Drug Administration*
GAP	*Generic Acess Profile*
GATT	*Generic Attribute Profile*
GSR	*Galvanic Skin Response*
HR	*Heart Rate*
HRV	*Heart Rate Variability*
HTM	*Hypertext Markup Languages*
IaaS	*Infraestructure as a Service*
IBI	*Interbeat Interval*
IoT	*Internet of Things*
PaaS	*Platform as a Service*
PSN	*Parasympathetic Nervous System*
SaaS	*Software as a Service*
SC	*Skin Conductance*
SCL	*Skin Conductance Level*
SCR	*Skin Conductance Response*
SDK	*Software Development Kit*

SNS	*Sympathetic Nervous System*
SO	Sistema Operativo
UI	*User Interface*
WAN	*Wireless Area Network*
WLAN	*Wireless Local Area Network*
WPAN	*Wireless Personal Area Network*

1. Introdução

Este capítulo inicia-se com um breve enquadramento e motivação associados a sistemas de monitorização de sinais fisiológicos. De seguida são referidos os principais objetivos estabelecidos, e por fim, é apresentada a estrutura desta dissertação.

1.1. Enquadramento e Motivação

Ao longo dos últimos anos a popularidade e utilização de *wearables* (dispositivos vestíveis), tais como *smartwatches* e *smartbands*, têm registado uma elevada taxa de crescimento e a tendência é para que esse aumento continue (Mück et al., 2019). A utilização destes equipamentos, cada vez mais dotados de diferentes sensores, associados à Internet das Coisas (IoT – Internet of Things) tem vindo a afirmar-se na área da saúde através do desenvolvimento de sistemas de monitorização remota (Mavrogiorgou et al., 2019). A utilização de serviços *Cloud* permite que os dados adquiridos sejam facilmente guardados e posteriormente acedidos pelos médicos ou cuidadores, proporcionando uma monitorização contínua, ao contrário dos métodos tradicionais em que o paciente é obrigado a deslocar-se aos centros de saúde e hospitais.

Os sistemas de monitorização remota e a telemedicina vieram revolucionar a forma como se fornece assistência médica aos utentes (pacientes com insuficiência cardíaca, deficiência, idosos, entre outros) apresentando muitas vantagens na sua utilização. Entre estas destacam-se a possibilidade de o utente ser acompanhado a partir de casa, reduzindo do número de internamentos, a diminuição de custos e uma intervenção mais rápida em caso de emergência. A implementação de sistemas deste tipo é uma necessidade atual visto que, perante um aumento da afluência aos hospitais, pode não ser possível efetuar o atendimento presencial dos pacientes devido à falta de recursos e tempo. A pandemia do COVID-19 agravou ainda mais estas situações (Calton et al., 2020; Ritchey et al., 2020).

Dados da Direção-Geral da Saúde (DGS) revelam que em todo o mundo 12% das doenças são do foro mental e em Portugal este valor atinge um valor anual de 22,9 % (42,7% ao longo da vida) caso de perturbações ligeiras (*Direção-Geral da Saúde*, sem data). Neste âmbito, a utilização de sistemas de monitorização remota e telemedicina pode constituir um complemento dos métodos clássicos de diagnóstico baseados em escalas subjetivas de auto relato ou entrevistas cara a cara (*Can Remote Patient Monitoring Assist Mental Health Services | Trapollo*, sem data)(Abdullah & Choudhury, 2018).

Neste contexto torna-se particularmente relevante o desenvolvimento de sistemas para monitorização remota de pacientes com transtorno mental.

1.2. Objetivos

O principal objetivo desta dissertação é desenvolver o protótipo de um sistema de monitorização remota *online* de sinais fisiológicos para pessoas com transtorno mental. De forma sucinta, as tarefas principais desta dissertação são as seguintes:

- Estudo e seleção dos sinais fisiológicos que melhor se adequam/associam à saúde mental;
- Seleção de um dispositivo que possua sensores capazes de medir os sinais fisiológicos escolhidos e que ao mesmo tempo, tendo em conta o tipo de utilizadores a que se destina, seja um dispositivo robusto, leve, discreto e que não condicione o dia-a-dia dos seus utilizadores;
- Estudo e seleção de um serviço *Cloud*;
- Desenvolvimento de uma aplicação Android que funcione como *gateway* de forma a permitir a comunicação entre dispositivos com diferentes protocolos (BLE e *Wi-Fi*). A aplicação deve ainda possibilitar a consulta e seleção dos dados que se pretende enviar para a *Cloud*;
- Desenvolvimento de uma aplicação *Web* que permita monitorizar em tempo real os sinais fisiológicos adquiridos, através da apresentação de gráficos.

1.3. Estrutura da Dissertação

Esta dissertação encontra-se estruturada em seis capítulos: Introdução, Fundamentos, Estado da Arte, Desenvolvimento do Sistema, Teste e Resultados e, por fim, Conclusões e Trabalho Futuro.

No segundo capítulo, Fundamentos, são apresentados os conceitos e tecnologias relevantes para o desenvolvimento do sistema em evidência. Em particular, apresentam-se sinais fisiológicos relacionados com a saúde mental, alguns serviços *Cloud,* tecnologias de comunicação e o sistema Android.

No terceiro capítulo, Estado da Arte, apresentam-se alguns trabalhos de interesse relacionados com o tema desta dissertação.

No quarto capítulo, Desenvolvimento do Sistema, é apresentada a arquitetura do sistema, descrevendo-se de modo geral os componentes envolvidos. De seguida, especifica-se o dispositivo

selecionado e apresentam-se as configurações dos serviços *Cloud* do Firebase. Nas últimas secções, aborda-se o desenvolvimento da aplicação Android que funciona como *gateway* e da aplicação *Web* onde de procede à visualização gráfica dos dados.

No quinto capítulo, Teste e Resultados, apresentam-se os testes realizados às aplicações desenvolvidas e a discussão dos resultados obtidos.

No sexto e último capítulo, Conclusões e Trabalho Futuro, são apresentadas as conclusões desta dissertação e fazem-se algumas sugestões para trabalhos futuros.

2. Fundamentos

Neste capítulo serão abordados conceitos importantes no contexto desta dissertação, nomeadamente sinais fisiológicos, alguns dispositivos adequados à sua medição e tecnologias relevantes para o desenvolvimento de um sistema de monitorização.

2.1. Sistema Nervoso Autónomo

O transtorno mental ou doença mental refere-se a um conjunto de condições de saúde mental que afetam o humor, pensamento e comportamento humano e que se refletem na atividade do sistema nervoso autónomo. As doenças mentais incluem depressão, transtorno de ansiedade, transtornos psicóticos, distúrbios alimentares e comportamentos de dependência.

O sistema nervoso autónomo (ANS – *Autonomic Nervous System)* é um sistema de controlo que funciona inconscientemente, regulando funções corporais como frequência cardíaca, respiração, temperatura, entre outros, sem qualquer reconhecimento ou ação consciente do organismo. As principais tarefas do ANS passam por manter a homeostase corporal e coordenar as respostas corporais. Este sistema pode ser subdividido em 2 ramos: sistema nervoso simpático (SNS – Sympathetic Nervous System) caracterizado por ser um sistema de resposta rápida e o sistema nervoso parassimpático (PNS – *Parasympathetic Nervous System)* caraterizado por ser um sistema de amortecimento mais lentamente ativado.

O sistema nervoso simpático é responsável pelo controlo involuntário de vários órgãos internos e atua, de modo a preparar o organismo para reagir em situações de medo, *stress* e excitação, adequando o funcionamento de diversos sistemas internos para um elevado estado de prontidão.

O sistema nervoso parassimpático, ao contrário do SNS está encarregue de estimular ações que permitam ao organismo responder a situações de relaxamento e calma. A sua ativação é normalmente mais lenta do que a do SNS.

2.2. Sinais Fisiológicos

De entre um leque abrangente de sinais fisiológicos que se podem medir, a literatura refere alguns que estão relacionados com a saúde mental (Abdullah & Choudhury, 2018), (Agelink et al., 2002; Y. T. Chen et al., 2011; Holzinger et al., 2013; A. Y. Kim et al., 2018; E. Y. Kim et al., 2017; Nahshoni et al., 2004; Osipov et al., 2015; Sarchiapone et al., 2018; Valkonen-Korhonen et al., 2003b; Zeeshan Baig & Kavakli, 2019) e que são apresentados de seguida.

2.2.1. *Atividade* Eletrodérmica

A atividade eletrodérmica (EDA – *Electrodermal Activity*), também chamada resposta galvânica da pele (GSR – *Galvanic Skin Response*), é um termo utilizado para descrever a variação contínua das características elétricas da pele. Estas variações podem ser observadas por alterações na resistência da pele, quando induzida uma pequena corrente elétrica, ou pela diferença de potencial entre diferentes zonas da pele (Boucsein, 1992).

A propriedade mais amplamente estudada é a condutância da pele (SC – *Skin Conductance*) que corresponde ao inverso da resistência da pele. A SC não é de domínio consciente e é modulada autonomamente pela atividade simpática. Se o ramo simpático do sistema nervoso autónomo estiver altamente estimulado, a atividade das glândulas sudoríparas vai ser ampliada, o que, por sua vez, aumentará a condutância da pele. A sua medição é normalmente caracterizada por duas componentes, o nível tónico (*Tonic Skin Conductance Level* – SCL) e a componente fásica (*Phasic Skin Conductance Response* – SCR) (Figura 1).

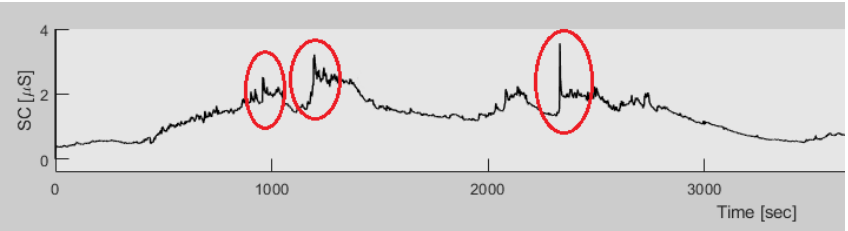

Figura 1: Exemplo de um sinal EDA onde os SCRs se encontram rodeados.

O SCL está geralmente associado a pequenas variações ao longo do tempo, no nível de condutividade basal (medição na ausência de qualquer estímulo) que resulta da regulação do sistema nervoso autónomo, hidratação da pele ou estado psicológico.

Por outro lado, a SCR está normalmente associada a eventos de curta duração, com alterações rápidas e significativas no sinal perante a ocorrência de estímulos. As alterações podem ser facilmente observáveis na forma de "picos" denominados de *Skin Conductance Responses* (SCRs) ou GSR *Peaks* seguidos de um decaimento progressivo (Figura 2) (Braithwaite et al., 2013; Empatica, 2015).

Além disso, os SCRs apresentam algumas características de importância aquando da sua análise como descrito na Tabela 1.

Tabela 1: Características dos SCR (Uday et al., 2018).

Características	Descrição
Latência	Intervalo de tempo entre o início do estímulo e a ocorrência do pico
Amplitude do Pico	Diferença de amplitude entre a amplitude do Pico e o estímulo
Tempo de Subida	Tempo que decorre desde o início do estímulo até atingir o pico
Tempo de Recuperação	Tempo necessário para o sinal recuperar do pico despoletado pelo estímulo

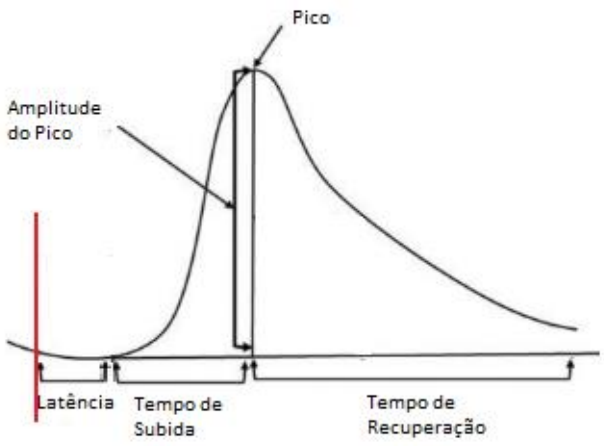

Figura 2: Características dos SCRs de um sinal EDA – adaptado de (Braithwaite et al., 2013).

2.2.2. Variabilidade da Frequência Cardíaca

A variabilidade da frequência cardíaca (HRV – *Heart Rate Variability*) é o termo utilizado para descrever a variação de intervalo de tempo entre batimentos cardíacos, denominado como *interbeat interval* (IBI).

As alterações de HRV foram descritas em muitos transtornos psiquiátricos, como por exemplo transtornos depressivos, esquizofrenia e transtorno bipolar. Estudos prévios revelaram que a HRV é inferior em pacientes com depressão, comparativamente com pessoas normais (Agelink et al., 2002),

(Y. T. Chen et al., 2011). D. S. Quintana (Quintana et al., 2016) realizou um estudo de HRV envolvendo 47 pacientes com esquizofrenia, 33 com transtorno bipolar e 212 como grupo de controlo. Os resultados obtidos demonstraram uma significativa redução do HRV nos dois distúrbios face ao grupo de controlo, contudo não se verificou nenhuma diferença entre dois distúrbios. No caso do estudo realizado por M. Valkonen-Korhonen (Valkonen-Korhonen et al., 2003a), os pacientes com psicose demonstraram uma redução da frequência cardíaca (HR) e não uma variação do HRV na realização de tarefas com carga mental crescente, quando comparados com o grupo de controlo no qual se verificou uma diminuição do HRV.

Na deteção de batimentos cardíacos, usualmente, recorre-se a métodos como o eletrocardiograma (ECG) e o fotoplestimograma (PPG).

2.2.2.1. Eletrocardiograma

O eletrocardiograma (ECG) corresponde à medida da atividade elétrica do coração registada na superfície da pele, com recurso a elétrodos. O IBI é calculado pela medição do intervalo de tempo entre dois pontos R consecutivos (pico mais elevado do eletrocardiograma) (Figura 3). Quanto maior for a distância RR, mais baixa será a frequência cardíaca e vice-versa. Por sua vez, o HR é inversamente proporcional ao IBI e pode ser derivado por:

$$HR = \frac{60}{IBI} \tag{1}$$

com IBI em segundos e HR expresso em batimentos por minuto (bpm)

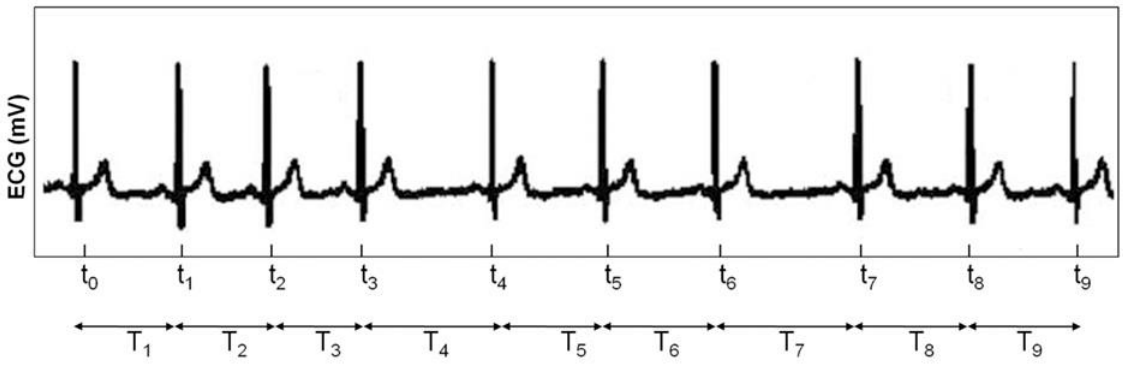

Figura 3: Intervalos RR representados no sinal ECG (Zeeshan Baig & Kavakli, 2019).

2.2.2.2. Fotoplestimograma

A fotopletismografia (PPG) é uma técnica ótica não invasiva de baixo custo empregue na deteção de mudanças no volume de sangue (BVP – *Blood Volume Pulse*) que passa nos tecidos. O sensor PPG deteta mudanças de absorção de luz através da diferença entre a emissão de luz infravermelha (normalmente, através de um díodo) e a quantidade de luz que retorna a um fotodetetor. Atualmente, esta técnica é a forma mais comum de medir a frequência cardíaca em *smartwatches* e *smartbands*.

Tal como acontece no sinal ECG, na técnica PPG, o HR é derivado do sinal bruto do BVP, medindo a distância entre dois picos PP consecutivos (Figura 4).

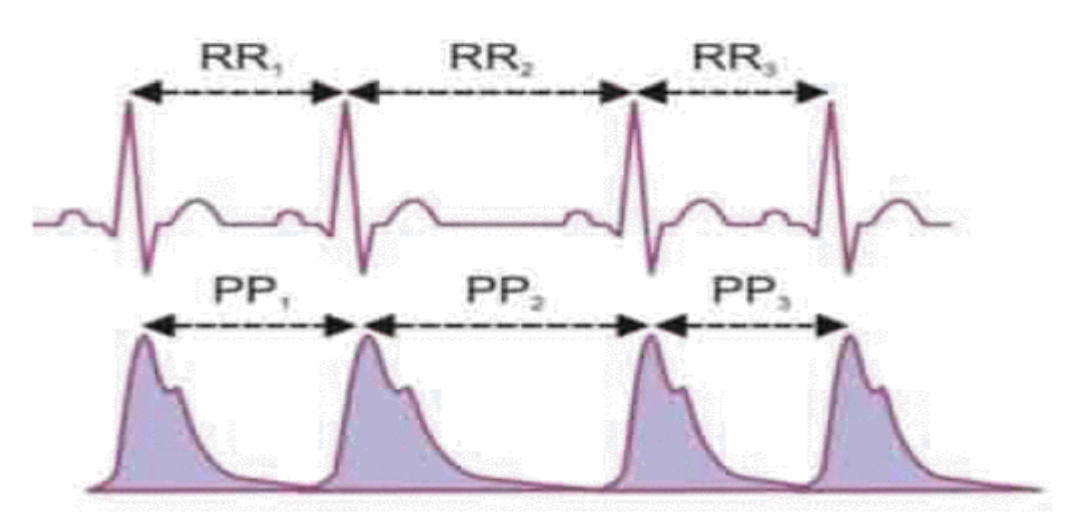

Figura 4: Relação entre o sinal ECG e o BVP (Y. T. Chen et al., 2011).

2.2.3. Eletroencefalograma

As células cerebrais humanas comunicam entre si via impulsos elétricos, sendo o eletroencefalograma (EEG) o sinal resultante da atividade elétrica no cérebro, captada por elétrodos colocados sobre o escalpe. A análise de sinais de EEG é um campo de investigação em curso nas áreas da neurociência e psicologia e, desta forma, o aceleramento do diagnóstico torna-se, algo bastante aliciante. Com isto em mente, um estudo efetuado por A. Khodayari-Rostamabad (Khodayari-Rostamabad et al., 2010) demonstra que, através de métodos de *machine learning*, é possível realizar um processo de diagnóstico autónomo a partir de um eletroencefalograma. Num conjunto de treino de 207 indivíduos composto por 64 utentes com desordem depressiva maior (MDD – *Major Depressive Desorder*), 12 com depressão bipolar, 40 com esquizofrenia crónica e 91 pessoas sem comportamentos depressivos, verificou-se ser possível prever corretamente a doença envolvida, na ordem dos 85%.

Outra aplicação do eletroencefalograma passa por prever os métodos mais eficazes para tratamentos de doenças psiquiátricas. Num estudo levado a cabo pelos autores anteriores (Khodayari-Rostamabad et al., 2013) aplicou-se uma rede neuronal capaz de mapear características extraídas de um pré-eletroencefalograma, de modo a antecipar a resposta de doentes com MDD a tratamentos com SSRI (*Selective Serotonin Reuptake Inhibitor*) quando sujeitas a este exame. Como resultados, os autores obtiveram uma precisão de 87,9%, com 98.76% de confiança, e uma taxa de precisão compreendida no intervalo [75%,100%], indicando que o método proposto tem potencial na previsão da terapia antidepressiva.

2.3. Dispositivos para Medição de Sinais Fisiológicos

Conforme o que se encontra supramencionado, foram estudados alguns dispositivos que possuam sensores capazes de monitorizar os sinais fisiológicos pretendidos.

A *wristband* Embrace (Empatica) é um dispositivo para medição de sinais fisiológicos que se encontra validado pelo *Food and Drug Administration* (FDA) nos Estados Unidos e na Europa encontra-se certificada como um dispositivo médico. Tem como foco a deteção de crises de epilepsia. Esta pulseira recolhe e analisa os dados e perante alguma irregularidade é, imediatamente, enviado um alerta para os cuidadores. O principal sensor responsável pelos resultados obtidos é o sensor EDA incorporado, que permite medir mudanças elétricas na pele associadas ao sistema nervoso simpático que regula situações de excitação ou *stress* (Poh et al., 2012).

Um outro exemplo é a Bitalino, uma placa de desenvolvimento projetada para lidar com a aquisição de sinais corporais. Num dos seus modelos, a placa encontra-se dividida em blocos que permitem ao utilizador ajustá-los da maneira que melhor se adapta às suas ideias de projeto (Plácido Da Silva et al., 2014). Para além disso, este dispositivo integra sensores de eletromiografia, atividade eletrodermal, eletrocardiografia, entre outros que serviram como base para alguns trabalhos. Um exemplo disto é o trabalho realizado por S. Guardiola (Guardiola et al., 2017), onde se efetuou uma avaliação de como o stress na prática de condução influencia a atividade cardíaca do condutor. Esta experiência foi realizada num simulador de condução e verificou-se que os sinais fisiológicos podem refletir a resposta do organismo perante situações de stress.

Os dispositivos da Shimmer (por exemplo Shimmer3 GSR+, Shimmer3 ECG) foram desenvolvidos com o intuito de possibilitar a aquisição de dados biofísicos e cinemáticos de forma simples e eficaz

(*Shimmer | Wearable Sensor Technology*, sem data). O seu *design* pequeno e confortável não causa desconforto ao utente e não afeta qualquer tipo de movimento ou exercício. O Shimmer fornece uma variedade de aplicações de software, quer para Windows como Android que dão suporte à aquisição, armazenamento e exibição de dados. Para além disto, ainda disponibiliza interface de programação de aplicações (APIs) para desenvolvimento de aplicações em C# e Android, bem como drivers para MATLAB e LabVIEW. G. Udovičić, J. Derek, M. Russo, et al. (Udovičić et al., 2017), recorreram ao uso de um dispositivo Shimmer3 GSR+ com o intuito de analisar o comportamento humano e determinar qual a emoção que este sente naquele momento. Para tal, as emoções de 13 indivíduos foram estimuladas através da visualização de 30 fotografias, provenientes de uma base de dados com 730 imagens. No final, verificou-se uma precisão de medição de 81% perante situações de excitação.

Por último, a *wristband* E4 (Empatica) é destinada a investigadores, permitindo a gravação de sinais fisiológicos e a sua posterior análise. Além do mais, é um dispositivo flexível com ferramentas para gravação e gestão de dados e possibilita, ainda, a criação de aplicações próprias, recorrendo à API e ao kit de desenvolvimento de software (SDK) fornecidos. Experimentalmente, este equipamento demonstrou uma qualidade de dados análoga à obtida com dispositivos, atualmente, utilizados na medicina (McCarthy et al., 2016). C. McCarthy, N. Pradhan, C. Redpath, et al. (McCarthy et al., 2016) compararam medições para avaliação de eventos cardíacos alcançadas pela *wristabnd* E4 e por um monitor Holter, equipamento frequentemente utilizado em exames médicos. Este estudo demonstrou que o desempenho de ambos foi consistente para 85% dos segmentos analisados.

Na Tabela 2 são apresentadas algumas características dos dispositivos mencionados anteriormente.

Tabela 2: Comparação entre os diferentes dispositivos mencionados.

Nome	Sensores	Vantagens	Desvantagens
Embrace (249$)	• GSR • Acelerómetro • Giroscópio • Sensor de temperatura periférica	• Dispositivo validado pelo FDA	• Não permite desenvolver aplicações por terceiros
Bitalino (desde 150€)	• ECG • EMG • GSR • ACC • etc	• APIs que permitem desenvolver apps para diversas plataformas, C++, Matlab, Raspberry Pi, Android,	• Dispositivo mais difícil de adaptar ao corpo e pouco confortável devido às suas dimensões
Shimmer3 (desde 529€)	• ECG • EMG • GSR • PPG • IMU	• APIs que permitem desenvolver apps c# e Android	• Inexistência de um módulo que possua o conjunto de todos os sensores
E4 (1690$)	• PPG • GSR • Acelerómetro • Sensor de temperatura periférica	• APIs que permitem desenvolver apps para Android, IOS e Windows	• Custo mais elevado que o dos outros dispositivos apresentados nesta tabela

2.4. Tecnologias de comunicação

Diversas tecnologias de rede podem ser utilizadas no desenvolvimento de sistemas remotos de monitorização ligados à área da saúde. Primeiramente, é necessário recolher os dados sensoriais, seguindo-se a necessidade de se assegurar a transmissão dos dados coletados para um nó central e, por último, enviar os dados para uma estação remota.

2.4.1. Rede de Área Corporal

Uma rede de área corporal (BAN - *Body Area Network*) consiste numa rede composta por um ou mais sensores implantados ou colocados sobre a superfície corporal e uma ou mais redes de comunicação, utilizadas para transportar os dados coletados. Deste modo, vários parâmetros podem ser monitorizados continuamente, tais como eletrocardiograma (ECG), eletroencefalograma (EEG), atividade eletrodérmica (EDA), temperatura, entre outros.

Numa das abordagens de implementação a arquitetura da comunicação BAN é dividida em três camadas (M. Chen et al., 2011; Khan & Pathan, 2018): *intra-BAN*, *inter-BAN* e *beyond-BAN*, como apresentado na Figura 5.

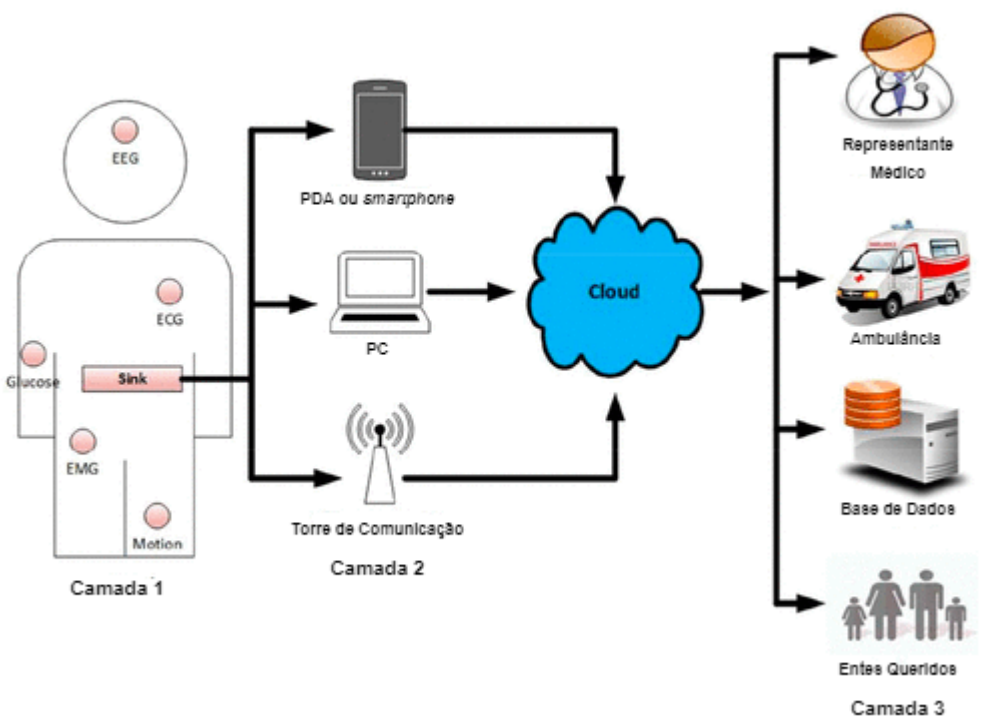

Figura 5: Arquitetura das 3 camadas que se integram na comunicação BAN, adaptado de (Khan & Pathan, 2018).

A primeira camada, *intra-BAN*, refere-se à comunicação com ou sem fios entre os dispositivos sensores e um nó coletor (*sink*) transportado pelo utilizador. Este coletor pode ser um dispositivo desenhado para essa função, bem como um dispositivo genérico como um *smartphone*.

Na segunda camada, encontra-se a *inter-BAN* que engloba a comunicação sem fios entre o *sink* e um ou mais pontos de acesso (APs – *access points*). Esta camada é responsável por garantir a

interconexão entre diferentes tipos de redes, podendo recorrer a tecnologias de redes sem fios como *Wi-Fi*, *Bluetooth*, *Zigbee*, entre outros.

Por último, a terceira camada *beyond-Ban* é projetada para fornecer serviços de *streaming* de dados coletados da BAN e serviços de armazenamento remoto. Desta forma, os médicos ou cuidadores podem ser notificados sobre uma emergência através da Internet ou por SMS, por exemplo através do desenvolvimento de um sistema cliente/servidor para acesso a registos de pacientes.

2.4.2. Redes de Área Pessoal

As redes de área pessoal sem fios (WPAN – *Wireless Personal Area Network*) são redes que conectam equipamentos próximos entre si e centrados em torno de um único individuo, possibilitando comunicações de curto alcance e de baixo consumo energético. *Bluetooth Low Energy* (BLE) e ZigBee (IEEE 802.15.4) são duas das tecnologias que mais se destacam (Pantelopoulos & Bourbakis, 2010).

2.4.2.1. *Bluetooth Low Energy*

O BLE é uma tecnologia de rede sem fios de curto alcance e de baixo consumo de energia que opera na banda de frequência de 2,4GHz. Em comparação com o *Bluetooth* clássico, o BLE destina-se a fornecer um consumo bastante inferior com um alcance de comunicação análogo. Em termos de arquitetura, apresenta uma pilha de camadas projetada para transmitir dados com baixo consumo de energia, sendo assim uma opção a ter em conta para dispositivos alimentados a bateria.

Como mencionado, o BLE opera na gama de frequência dos 2.4GHz que se divide em 40 canais: 37 canais de dados para comunicação bidirecional entre dispositivos e 3 canais para descobrir e estabelecer conexão (Fotouhi et al., 2016). Para estabelecer a conexão e trocar dados é necessário compreender duas camadas da *stack* do BLE, a camada GAP (*Generic Acess Profile*) e a camada GATT (*Generic Attribute Profile*), como representado na Figura 6.

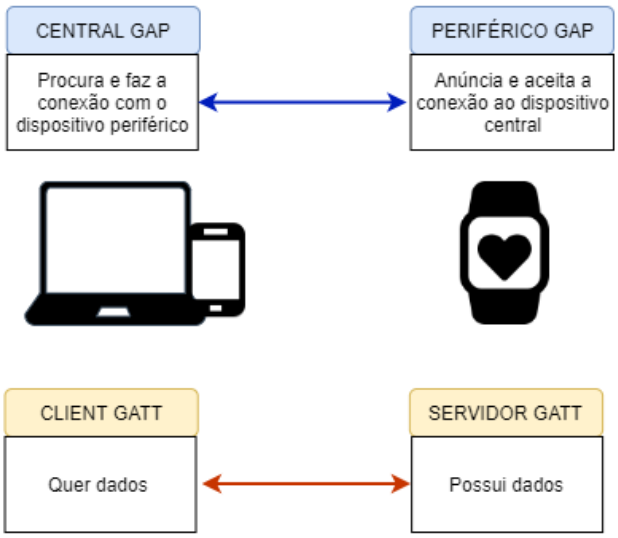

Figura 6: Visão geral da comunicação BLE.

A camada GAP é encarregue das conexões entre dispositivos BLE, tornando os dispositivos visíveis para o mundo exterior e definindo dois tipos de dispositivos: periférico e central. O dispositivo periférico emite um sinal que anuncia a sua presença e aceita a conexão do dispositivo central. Por sua vez, o dispositivo central procura os anúncios emitidos pelo dispositivo periférico e estabelece conexão com eles (*Bluetooth low energy overview | Android Developers*, sem data; Dian et al., 2019) .

Por outro lado, o GATT define como dois dispositivos comunicam entre si após se estabelecer a conexão. Do mesmo modo que a camada GAP, o BLE divide os dispositivos em dois tipos de acordo com a origem dos dados: servidor GATT e cliente GATT. O cliente é o dispositivo que recebe ou acede aos dados do servidor e corresponde tipicamente ao dispositivo periférico. Por sua vez, o servidor é o dispositivo que contém os dados e os remete para o cliente. Os dados encontram-se estruturados no servidor com base numa hierarquia de características e serviços. O serviço corresponde à entidade mais ampla que engloba uma ou mais características e define o comportamento associado à realização de uma função ou recurso do dispositivo. Uma característica é composta por atributos e define a transferência de dados. Para além disso, é constituída por um campo que declara a característica, um campo com um valor e um campo que descreve esse valor. A Figura 7 apresenta, por exemplo, o serviço de temperatura de uma *smartband*, que pode incluir a característica que retorna o valor da temperatura e usa um atributo para descrever o sensor, outro para guardar o valor da temperatura e outro para especificar a unidade de medida da temperatura.

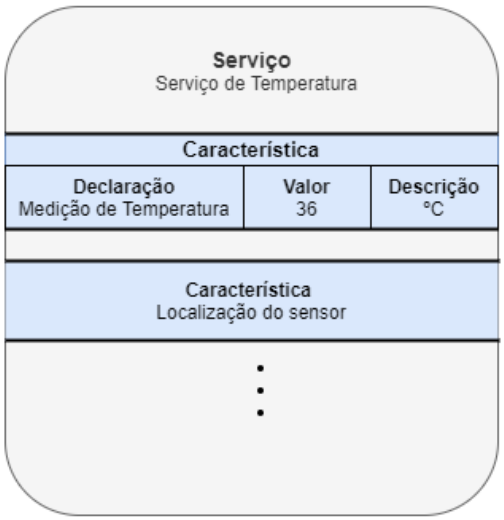

Figura 7: Exemplo da estrutura de dados de um servidor GATT.

2.4.3. Redes de Área Local Sem Fios

Uma rede de área local sem fios (WLAN – *Wireless Local Area Network*) é uma tecnologia de redes sem fios que surgiu da evolução das antigas LAN (*Local Area Network*), redes por cabos como as redes IEEE 802.3/Ethernet. O IEEE 802.11/ *Wi-Fi* é o standard mais utilizado pelas WLAN modernas e permite que os dispositivos se conectem à rede local ou à Internet através de APs. O *Wi-Fi* usa bandas de frequência de 2.4 GHz e 5GHz e têm um alcance médio de 100 metros, dependendo do padrão utilizado. Em comparação com as redes WPAN, estas redes apresentam uma maior taxa de transmissão e alcance de dados, contudo possuem um consumo energético superior.

Além do mais, nos sistemas de monitorização, o *Wi-Fi* é frequentemente utilizado para fazer a ponte entre a camada de processamento e a camada de armazenamento (Nguyen et al., 2017).

2.4.4. Redes de Longa Distância

Uma rede de longa distância sem fio (WWAN – *Wireless Wide Area Network*) é uma tecnologia de rede sem fio que possibilita a troca de dados de longa distância e abrange uma enorme área geográfica, como um país, continente ou o mundo. Um exemplo deste tipo de rede é a própria Internet.

A dimensão superior desta rede, em comparação com a WLAN, requer diferenças de tecnologia. Uma WWAN geralmente difere da WLAN pelo uso de tecnologias de telecomunicação móvel, como 2G, 3G, 4G LTE e 5G para transferir dados. No caso da rede 4G, a largura de banda está compreendida entre os 200Mbps e 1Gbps e, mais recentemente, a tecnologia 5G dispõe de uma largura de banda na ordem

dos Gbps (1Gbs e superior)(Sharma et al., 2016). Estas redes, de um modo geral, possuem também custos associados à sua utilização.

Existem ainda redes de longa distância, mas com pouco consumo energético, as LPWAN (Raza et al., 2017), como por exemplo as LoRa e Sigfox. LoRa foi desenvolvida pela Semtech Corporation e utiliza uma nova técnica de espalhamento espectral, denominada *Spread Spectrum*, o que permite um baixo consumo energético. A taxa de transmissão é relativamente baixa, podendo atingir cerca de 37,5 kbps. O seu alcance está dependente da zona de instalação da antena, do ambiente em redor e da presença de obstáculos, podendo atingir largas centenas de quilómetros (100Km). A tecnologia SigFox foi projetada para efetuar comunicações de longo alcance, fácil conectividade e consumo de energia reduzido. Esta rede pode atingir alcances de cerca de 10km em regiões urbanas e 50km em regiões rurais. A sua taxa de transmissão de dados é extremamente baixa, podendo atingir apenas os 100bps o que poderá ser um inconveniente em diversas situações.

Por norma, este tipo de redes é aplicado na transmissão de dados para a camada de armazenamento em sistemas de monitoração ou aplicações inteligentes de saúde (Zemrane et al., 2019)(Olatinwo et al., 2019).

2.5. Serviços *Cloud*

A *Cloud* ou a computação na *Cloud* refere-se a ampla gama de fornecimento de serviços de computação que inclui servidores, armazenamento, redes, base de dados, software, etc para disponibilizar de forma rápida e flexível estes recursos aos utilizadores sem a necessidade de infraestruturas ou hardware interno (Mell & Grance, 2012; *O que é a computação na cloud? | Microsoft Azure*, sem data). Em suma, a *Cloud* é a expressão geral para o fornecimento de serviços tecnológicos através da Internet.

O modelo da *Cloud* é composto por diferentes modelos de serviço, destacando-se:

- ***Software as a Service* (SaaS):** Este modelo é caracterizado por fornecer aos utilizadores acesso remoto a aplicações de software hospedadas numa infraestrutura *Cloud*, através da Internet. Neste modelo o utilizador não controla a infraestrutura da "nuvem" como servidores, sistemas operativos, rede ou armazenamento, sendo tudo gerido pelo provedor da *Cloud*. Dropbox, Gmail ou Office 365 são exemplos de SaaS.

- ***Platform as a Service* (PaaS):** Este modelo de serviço fornece ao utilizador ferramentas para desenvolver e testar aplicações criadas ou adquiridas por si, usando linguagens de programação, bibliotecas e *frameworks* suportadas pelo provedor. Este modelo permite criar aplicações sem a necessidade de o utilizador se preocupar com a gestão e configuração da infraestrutura de servidores, sistema operativo, armazenamento.
- ***Infraestructure as a Service* (IaaS):** Este modelo fornece recursos, tais como armazenamento, redes, máquinas virtuais e sistemas operativos, onde o utilizador pode implantar e executar software.
- ***Backend as a Service (BaaS)*:** Este modelo disponibiliza APIs que permitem fazer a ponte entre as aplicações *Web* e *mobile* e os serviços disponíveis na *Cloud*. Desta forma, o utilizador pode focar-se no desenvolvimento *front-end* sem ter responsabilidades inerentes à gestão e manutenção de servidores que fica encarregue ao provedor. Este modelo inclui serviços como autenticação de utilizadores, gestão de base de dados, notificações e hospedagem.

2.5.1. Firebase

O Firebase é uma plataforma que surgiu em 2011, primeiramente desenvolvida pela *start-up* Envolve e mais tarde, em 2014, adquirida pela Google. O Firebase é um Baas que abrange diversos serviços, como base de dados em tempo real, autenticação, *hosting*, armazenamento, entre outros e dispõe de um plano gratuito (*Firebase*, sem data).

2.5.1.1. Autenticação

Firebase Authentication é o serviço disponibilizado para proceder à autenticação de utilizadores e fornece serviços de *back-end*, SDKs fáceis de usar e uma interface pré construída (FirebaseUI) que agiliza o processo de implementação. O Firebase Authentication permite que os utilizadores façam *login* nas suas aplicações através de diferentes formas: Email e Senha, número de telefone e através de diferentes provedores externos - *Login* do Google, Facebook, Apple, GitHub, Twitter, etc. Após o *login* através de provedores externos ser efetuado com êxito é possível recuperar o *acess token* do provedor, que pode, de seguida, ser usado para obter dados adicionais usando as APIs deles.

2.5.1.2. Base de Dados em Tempo Real

O Firebase Realtime Database é o serviço mais popular do Firebase e consiste numa base de dados NoSQL que além de permitir o armazenamento de dados, possibilita a sincronização dos mesmos em tempo real para todos os utilizadores ligados em diferentes plataformas como *Web* e mobile. Isto é alcançado graças à tecnologia de *WebSockets* que proporciona uma comunicação contínua e bidirecional através de um único *socket* sem que seja necessário fazer pedidos constantes ao servidor. Existe ainda suporte que atua perante perdas de conexão à Internet e armazena os dados a serem gravados numa *cache* local. Assim que a conexão for restabelecida, os dados são automaticamente enviados e os clientes recebem as alterações perdidas, restabelecendo deste modo a sincronização.

2.5.1.3. *Cloud Functions*

As Funções do Firebase permitem executar respostas automáticas a eventos gerados por recursos do Firebase e do Google *Cloud*, como por exemplo a autenticação e alterações na base de dados sem ser necessário a gestão de servidores proporcionando, portanto, uma manutenção fácil e reduzida. As funções podem ser desenvolvidas em Javascript ou TypeScript e podem ser implantadas automaticamente nos servidores através de um comando apenas.

2.6. Sistema Operativo Android

O Android é um sistema operativo (SO) baseado em Linux que se encontra presente em 2,5 mil milhões de dispositivos móveis ativos. Além de estar maioritariamente presente em *smartphone*s e tablets, atualmente, este também se encontra disponível para TV (Android TV), *smartwatch* (Android Wear) e carros (Android Auto). Inicialmente, foi desenvolvido por uma aliança de empresas, e posteriormente comprado pela Google em 2005 à qual pertence até à data. Neste momento, a versão mais recente é o Android 10. O IDE oficial para desenvolvedores é o Android Studio e encontra-se de momento na versão 3.6.3. Neste subcapítulo serão abordados os principais componentes para o desenvolvimento de aplicações.

2.6.1. Activity

Uma *Activity* representa uma tela/janela de uma aplicação Android. É um componente essencial que serve de interface ao utilizador, permitindo-o interagir com a aplicação. Normalmente a maioria das aplicações contêm mais que um ecrã, ou seja, incluem várias *Activities*. Comummente, o ecrã principal de uma aplicação é denominado de MainActivity, por ser o primeiro a ser exibido ao utilizador.

É necessário salientar que para ser possível utilizar as *Activities* é necessário efetuar, previamente, o seu registo no ficheiro AndridManifest.xml da aplicação Android, para que o sistema operativo as possa reconhecer.

Ao contrário de outras linguagens de programação que iniciam as aplicações através do método main(), o sistema Android inicia numa instância *Activity*. Ao longo do programa, a instância *Activity* vai transitando de estado resultante da invocação de métodos de *callback* correspondentes a estágios específicos do seu ciclo de vida. Na Figura 8 pode ser vista a representação do ciclo de vida de uma *Activity* e os respetivos métodos.

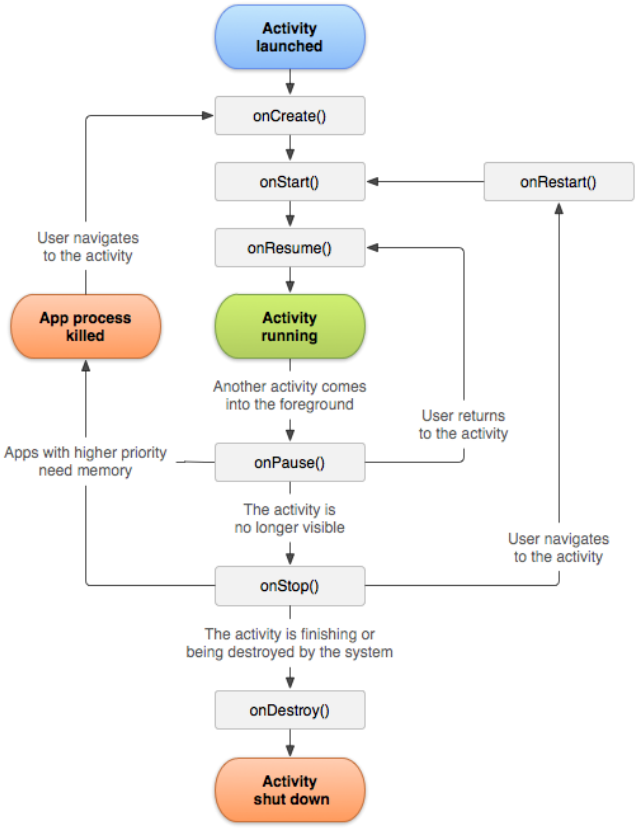

Figura 8: Ciclo de vida de uma Activity (*Understand the Activity Lifecycle | Android Developers*, sem data).

Conforme a *Activity* entra num novo estado, o sistema invoca diversos métodos. De forma resumida, os mais relevantes são:

- **onCreate():** é executado quando a *Activity* é criada e a sua implementação é obrigatória. É neste método que se executa toda a logística de inicialização e se define o *layout* da interface gráfica. Este fenómeno deve acontecer apenas uma vez durante toda a vida útil de uma *Activity*.
- **onStart():** a *Activity* torna-se visível para o utilizador, à medida que a aplicação prepara a sua entrada em primeiro plano e a torna interativa.
- **onResume():** a *Activity* encontra-se em primeiro plano e é neste estado que interage com o utilizador. Este estado é mantido até que algo suceda e desvie o foco da aplicação.
- **onPause():** este método é chamado quando a *Activity* abandona o primeiro plano. É a primeira indicação de que a *Activity* será parada ou destruída.
- **onStop():** é executado quando a *Activity* deixa de estar visível para o utilizador. Pode ocorrer quando a *Activity* vai ser destruída, outra *Activity* se vai sobrepor à atual ou quando uma *Activity* existente é trazida para primeiro plano.
- **onDestroy():** é invocado quando a *Activity* vai ser destruída. Isto pode ocorrer devido à finalização da *Activity* ou como forma de poupar espaço por parte do sistema, destruindo-a temporariamente.

É de realçar que não é obrigatório implementar cada um destes métodos, apenas é importante garantir que a aplicação tenha o desempenho esperado pelo utilizador.

2.6.2. Fragment

Um *Fragment* é um componente independente do Android que pode ser usado por uma *Activity*. É possível combinar vários fragmentos em uma única *Activity*, de modo a separar diferentes partes da interface gráfica, reduzindo assim a responsabilidade exercida sobre ela. Pode ser visto como uma espécie de *subActivity* com o seu próprio *layout* e os seus próprios eventos de *callback* do ciclo de vida. Deve ser sempre hospedado a uma *Activity* e, por isso, o seu ciclo de vida está dependente do ciclo de vida da *Activity* que o hospedou. O ciclo de vida encontra-se apresentado na Figura 9.

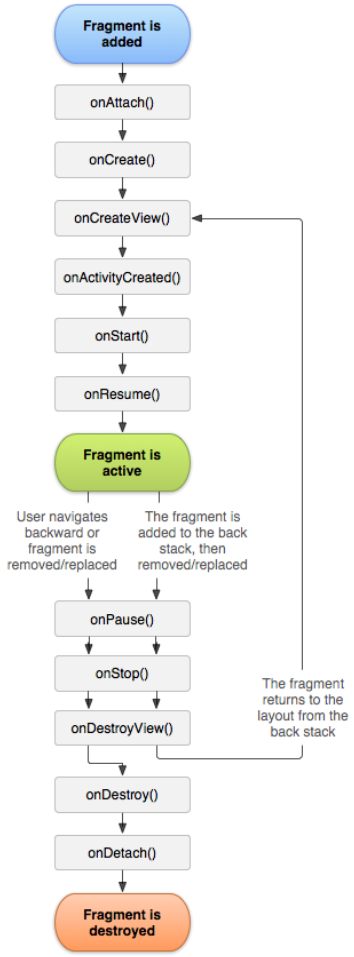

Figura 9: Ciclo de vida de um Fragment (*Fragments | Android Developers*, sem data).

As semelhanças entre o ciclo de vida do *Fragment* e da *Activity* são visíveis, excetuando os seguintes eventos que se encontram apenas no ciclo de vida do *Fragment*:

- **onAttach():** momento em que a instância do *Fragment* se associa à *Activity*. Contudo nenhuma se encontra totalmente inicializada.
- **onCreateView():** o sistema invoca este método na altura de desenhar o *Fragment* e retorna uma *View* que contém o *layout*.
- **onActivityCreated():** este método é chamado quando a *Activity* completou o onCreate e é a partir deste momento que se pode interagir com a *interface*.
- **onDestroyView():** é chamado quando a atividade vai ser destruída, à semelhança do que acontece no método onDestroy() da *Activity*.
- **onDeatach():** momento em que o *Fragment* é removido da *Activity*.

2.6.3. Services

Um *Service* é um componente que é executado em segundo plano sem interação direta com o utilizador. É, tipicamente, utilizado na realização de tarefas que demoram longos períodos de tempo e depois de iniciado, continua em execução mesmo que o utilizador altere de aplicação. Existem 3 tipos distintos de *Services*:

- **Foreground:** este tipo de *Service* realiza operações percetíveis ao utilizador através da exibição de uma notificação. O serviço continua ativo mesmo que não haja interação com a aplicação e é, normalmente, utilizado na transferência de ficheiros ou na reprodução de música.
- **Background:** este serviço realiza operações sem a perceção do utilizador.
- **Bound:** serviço que funciona como servidor numa interface cliente-servidor. Este serviço apenas permanece em execução enquanto um cliente (como uma *Activity*) está conectado a ele. O serviço é destruído quando todos os clientes forem desvinculados.

Como o *Service* não possui interface com o utilizador, não se encontra conectado ao ciclo de vida de uma *Activity* e, portanto, tem o seu próprio ciclo de vida. A Figura 10 apresenta o ciclo de vida de um *Service*, sendo que do lado esquerdo se encontra representado o ciclo de vida de um *Service* do tipo *Foreground* e *Background* e do lado direito do tipo *Bound*.

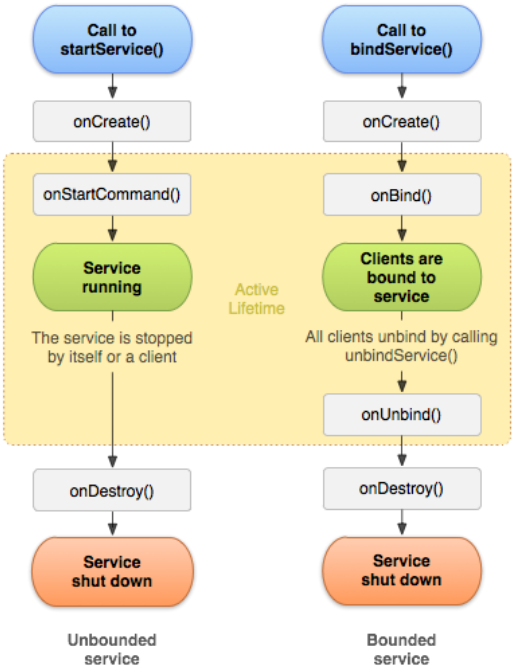

Figura 10: Ciclo de vida de um Service (*Services overview | Android Developers*, sem data).

2.7. Programação *Web*

A arquitetura de uma aplicação *Web* engloba relações e interações entre componentes como interface ao utilizador, *middleware*, base de dados, entre outros, de forma a garantir a robustez, eficiência, escalabilidade e segurança de uma página *Web*. Assim, uma aplicação *Web* é constituída por duas componentes estruturais, o lado do cliente e o lado do servidor(*What is the Importance of Web Application Architecture?*, sem data).

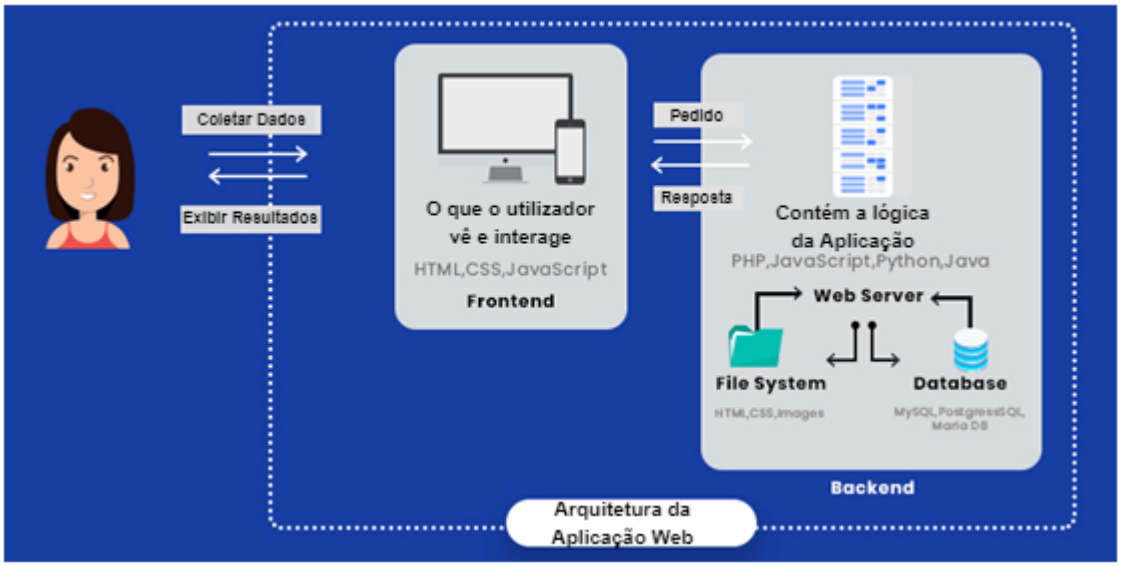

Figura 11: Arquitetura aplicação Web, adaptado de (*What is the Importance of Web Application Architecture?*, sem data) .

O lado do servidor, também denominado *back-end*, representa a parte que não é acessível ao utilizador e é responsável pela criação da página requerida pelo utilizador (HTTP *request*). Normalmente, subdivide-se em duas porções: *Web server* e base de dados, onde se recorre a linguagens de programação como PHP, Java, JavaScript, C#, Phyton, entre outros.

O lado do cliente, ou *front-end*, corresponde à interação do utilizador com a interface gráfica através de um *browser* em contraste com o *back-end*, onde o código é executado no servidor. A combinação das linguagens HTML (*Hypertext Markup Languages*), CSS (*Cascading Style Sheets*) e JavaScript são utilizadas, respetivamente, para estruturar, dar estilo e comportamento às páginas *Web*. A Figura 12 apresenta a estruturação de uma página através de três camadas.

Figura 12: Estrutura de uma página Web, adaptado de (Yi et al., 2018).

2.7.1. HTML

O HTML é o esqueleto base para o desenvolvimento de páginas *Web*, sendo responsável pela construção da camada estrutural sem o qual a página não existiria. Esta é uma linguagem de marcação, ou seja, representa um conjunto de códigos aplicados em documentos de texto e utilizados na definição e apresentação do texto com recurso a *tags* (etiquetas), que, por sua vez, cercam as diferentes partes do conteúdo e orientam o texto para agir de acordo com o objetivo projetado. É a partir desta linguagem que se criam páginas *Web* e que se possibilita a apresentação de informação na Internet.

A estrutura de um documento HTML começa com o elemento <html>, elemento raiz da página. O elemento <head> contem meta informações sobre a página e o elemento <body> é responsável por definir o corpo do documento, servindo de contentor para todo o tipo de conteúdo visível. O documento termina com o fecho do elemento raiz </html> (*Introduction to HTML | w3schools.com*, sem data).

Com a atualização para o HTML5 (versão mais recente) as páginas *Web* apresentam melhor estrutura, com novos elementos a poderem ser incorporados de forma mais simples e intuitiva.

2.7.2. CSS

O CSS é um tipo de linguagem utilizada na alteração e adição de estilos a um documento *Web*, incluindo cores, *layout*, tipos de letra, entre outros. Esta é responsável por criar a camada de apresentação de uma página *Web*, separando o conteúdo da representação visual.

Apesar de o código CSS poder ser executado diretamente nas *tags* ou dentro de *tags* <style>, uma boa prática consiste em criar um ficheiro CSS que contenha todos os estilos necessários. Assim, quando for necessário aplicar alguma alteração basta modificar este arquivo. Esta separação permite, ainda, facilitar a manutenção de *Web*sites, compartilhar estilos entre páginas e personalizar páginas para

diferentes ambientes. Para além disto, o CSS é responsável por tornar uma página responsiva, através da aplicação de diferentes estilos de acordo com a resolução do ecrã, onde a página *Web* é visualizada.

A diferença entre um site que usa CSS e outro que não use este tipo de linguagem é extremamente díspar. Posto isto, tendo em conta que o CSS torna um site esteticamente mais agradável e intuitivo, esta linguagem, ao longo do tempo, deixou de ser um recurso e tornou-se numa necessidade.

O Bootstrap é o *framework Web* mais popular para o desenvolvimento de aplicações *front-end* que permite um desenvolvimento mais rápido graças aos *templates* e estilos que fornece(*Bootstrap*, sem data).

2.7.3. JavaScript

O JavaScript é uma linguagem de programação dinâmica e orientada a objetos que permite criar interatividade com o utilizador, com o conteúdo da linguagem HTML e com a estética desse mesmo conteúdo fornecida pelo CSS. Assim, esta linguagem produz efeitos, apresenta conteúdos dinâmicos, entre outras funcionalidades interativas. O JavaScript constitui a camada de comportamento e através do interpretador do *browser* o código fonte é lido e executado diretamente sem a necessidade de um compilador.

Embora originalmente o JavaScript tenha sido projetado para ser executado nos browsers, atualmente é, também, utilizado do lado do servidor através de ambientes como o node.js.

Os *Frameworks* Java Script como Angular, React e Vue.js oferecem estruturas que possibilitam acelerar o processo de desenvolvimento, minimizando a quantidade de código necessária. Estas estruturas permitem, ainda, configurar aplicações de forma responsiva e, assim, o trabalho com JavaScrip torna-se mais fácil e suave.

3. Estado da Arte

M. Hamim, S. Paul, S. Hoque, et al. (Hamim et al., 2019) implementaram um sistema de monitorização de saúde remota para pessoas idosas ou doentes, de modo a que cuidadores e médicos possam ter uma análise mais facilitada e completa do estado do utilizador (Figura 13). O sistema é constituído por três sensores, nomeadamente, PPG, GSR e temperatura ligados a um Arduino UNO que efetua a leitura e processamento dos dados. Estes são posteriormente enviados para uma Raspberry Pi que se encontra conectada ao Arduino e é responsável pela transmissão dos dados para o sistema de armazenamento *Cloud* do Firebase.

Para fazer a interação com o utilizador, foi desenvolvida uma aplicação Android para apresentar os dados em tempo real enviados para a plataforma *Cloud*. A aplicação possui um sistema de *login* de modo a prevenir violação de privacidade, e apenas permite a consulta dos dados se o ID inserido corresponder ao ID fornecido pelo paciente.

Embora a interface gráfica desenvolvia permita a visualização dos dados em tempo real, apenas permite a consulta do seu valor numérico. A implementação de uma visualização gráfica como nesta dissertação permitiria uma análise ao longo do tempo que acrescentaria uma melhor perceção do estado dos doentes.

Figura 13: Diagrama de blocos do sistema de monitorização de saúde, adaptado de (Hamim et al., 2019).

I. Villanueva-Miranda, H. Nazeran e R. Martinek (Villanueva-Miranda et al., 2018) desenvolveram um sistema de ECG remoto como forma de prevenção de acidentes cardiovasculares. Este sistema é composto por três módulos distintos: o servidor remoto, o servidor local e a aplicação *Web* (Figura 14).

O servidor local engloba o hardware do sistema que é responsável pela recolha dos dados cardíacos do utilizador, pelo seu processamento e pelo seu envio para o servidor remoto. A aquisição do ECG foi realizada através de um sensor de 3 elétrodos conectados ao módulo e-Health Sensor Shield V2.0 para Arduino. Os dados adquiridos através da porta série são enviados através do protocolo de *Websockets* do servidor local (cliente.js) para o servidor (server.js) hospedado no serviço *Cloud* da Amazon (AWS).

A aplicação *Web* desenvolvida faz a ligação com o servidor remoto e proporciona ao utilizador a análise os dados cardíacos em tempo real. A interface é composta pela representação gráfica do sinal ECG, frequência cardíaca e intervalo RR.

Nesta implementação, o sistema desenvolvido não permite guardar os dados que são enviados impossibilitando a sua consulta posterior. O sistema poderia ser melhorado com a utilização de uma base de dados que permitiria obter os dados anteriormente gravados e proceder à sua visualização.

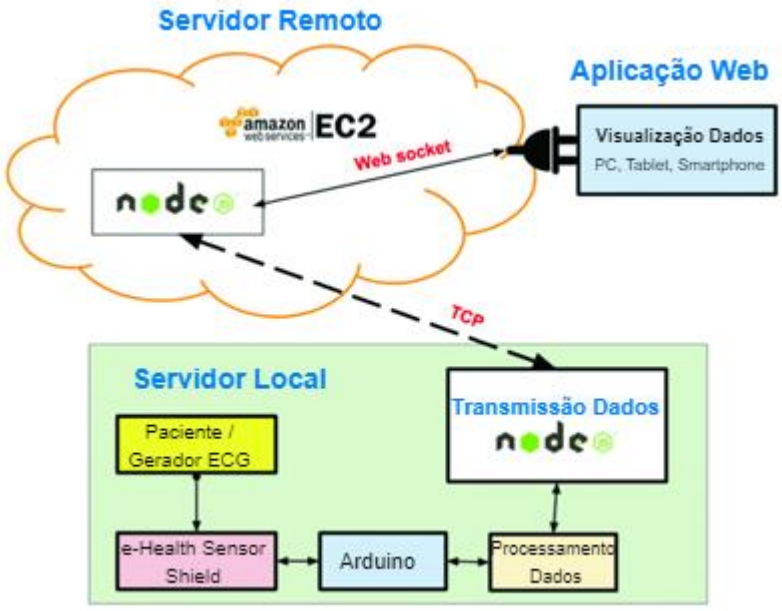

Figura 14: Arquitetura do sistema baseada em Websockets, adaptado de (Villanueva-Miranda et al., 2018) .

H. Yew, M. Ng, S. Ping, et al. (Yew et al., 2020) desenvolveram um sistema de monitorização em tempo real baseado na Internet of Things com aplicação em eletrocardiogramas. O sistema proposto divide-se em quatro módulos – sensor, controlador, módulo de comunicação e servidor *Web*. Primeiramente, os dados são coletados em tempo real através de um Arduino e enviados para o broker MQTT hospedado numa Raspeberry. De seguida, recorrendo ao protocolo MQTT (*Message Queing Telemetry Transport*) é efetuada a transmissão dos dados ECG para um servidor na *Web* acessível através de um *smartphone* ou de um computador para monitorização em tempo real ou para consulta de dados ECG registados anteriormente.

Este sistema foi testado e demonstrou que não ocorreram perdas de pacotes ou erros, tanto em redes locais como redes de longo alcance.

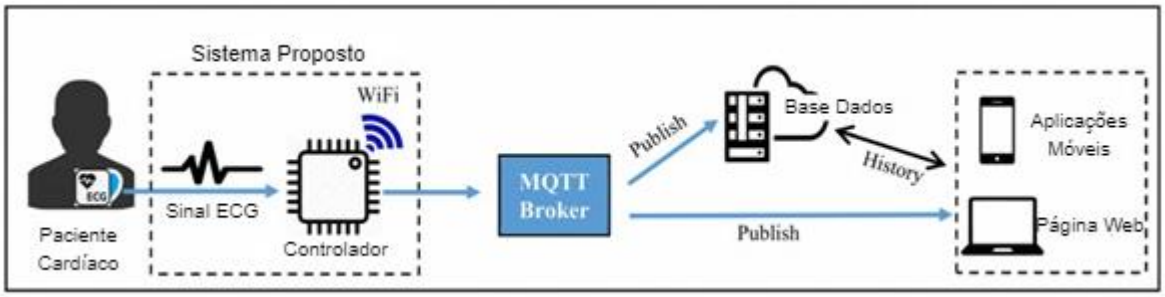

Figura 15: Arquitetura proposta do sistema de monitorização em tempo real, adaptado de (Yew et al., 2020).

4. Desenvolvimento do Sistema

Este capítulo descreve os componentes e procedimentos utilizados no sistema de monitorização proposto. Inicia-se com a apresentação dos requisitos do sistema a implementar, da visão global do sistema, passando depois pela apresentação do dispositivo utilizado para a recolha dos dados e finalizando com o *software* desenvolvido.

4.1. Requisitos do Sistema

Após a análise global do sistema pretendido foram analisados os procedimentos e requisitos necessários à sua implementação.

Em primeiro lugar surgiu a necessidade de encontrar um dispositivo capaz de adquirir os sinais fisiológicos de interesse e com características adequadas ao público alvo em questão. Tendo em conta as opções de mercado disponíveis foi selecionada a *wristband* E4. Este dispositivo destaca-se dos outros estudados por ser o mais discreto e de menores dimensões. Como não tem um ecrã, não constitui um fator de distração para o utilizador nem lhe permite uma possível desconfigurarão do aparelho. O equipamento proporciona ao utilizador uma grande liberdade de movimentos e autonomia, dispensando-o da necessidade de permanecer parado num determinado local para se proceder à monitorização. Além disso, possui certificado CE para dispositivo médico de classe 2a e, ao contrário da *wristband* Embrace, dispõe de ferramentas (SDK e API) que permitem o desenvolvimento de um sistema próprio. De seguida surgiu a necessidade de armazenar os sinais recolhidos e então procurou-se um serviço *Cloud* para esse efeito. Desta forma, a acessibilidade aos dados torna-se mais fácil, com os utilizadores a terem acesso a partir de qualquer lugar com ligação à Internet, em qualquer dispositivo e a qualquer hora. Optou-se pela utilização do serviço *Cloud* do Firebase que se destaca pela sua base de dados que suporta alterações em tempo real, permitindo não só o armazenamento dos dados, mas também a sincronização para todos os utilizadores ligados de cada vez que existe uma alteração na base de dados.

Por fim, para a visualização dos sinais recolhidos optou-se pelo desenvolvimento de uma aplicação *web* por facilitar a visualização gráfica dos dados e também pela possibilidade que oferece de ser consultada não só num computador, mas também num *smartphone* ou num *tablet*.

4.2. Arquitetura do Sistema

O sistema desenvolvido apresenta uma arquitetura remota que permite a visualização de sinais fisiológicos, adquiridos à distância, através de uma *wristband* E4, a partir de qualquer lugar com acesso à Internet. A arquitetura pode ser dividida em duas partes principais: *wristband* E4 e aplicação Android para aquisição e envio de dados para o Firebase; e aplicação *Web* para visualização gráfica dos dados. A Figura 16 apresenta a descrição deste sistema.

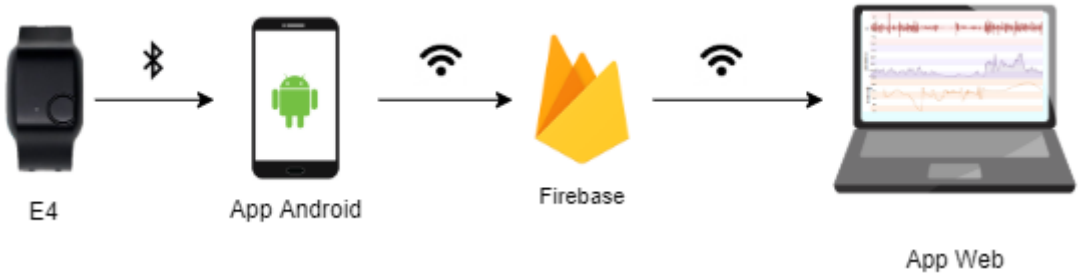

Figura 16: Arquitetura Remota do sistema implementado.

Através da tecnologia BLE é assegurada a transferência de dados entre a *wristband E4* e a aplicação Android onde se poderá consultar os valores numericamente. De acordo com a vontade do utilizador, este pode iniciar a transmissão dos dados para a base de dados do Firebase, que por sua vez são depois visualizados graficamente na aplicação *Web* através de leituras realizadas na base de dados. Como referido anteriormente, tanto para o envio como para a consulta dos dados é necessária conexão à Internet que pode ser assegurada dentro de casa através da rede *Wi-Fi*, ou via dados móveis no exterior.

4.3. Empatica E4 *wristband*

A *wristband* E4 da Empatica (Figura 17) foi o dispositivo escolhido para recolher os dados fisiológicos.

Trata-se de um *wearable* projetado para ser usado no pulso e destinado à recolha contínua de dados fisiológicos em ambientes domésticos ou médicos para uso de profissionais de saúde. Este dispositivo foi desenvolvido pela empresa Empatica e possui certificado CE para dispositivo médico de classe 2a.

Figura 17: Wristband E4 (*E4 wristband | Empatica*, 2020).

Relativamente aos sensores que incorpora, o dispositivo está equipado com quatro sensores – PPG, EDA, Acelerómetro de 3 eixos e termopilha infravermelho – que captam os dados fisiológicos e não fisiológicos do utilizador. As especificações deste sensores (*E4-SP069-B-20150001-Manual | Empatica*, sem data) podem ser vistas na Tabela 3.

Tabela 3: Especificações dos sensores da E4.

Sensores	Frequência de Amostragem	Resolução
PPG	64Hz	0.9nW/Digit
EDA	4HZ	1 digit ~900 pSiemens.
Termopilha IV	4HZ	0.02°C
Acelerómetro	32HZ	8bits

Inicialmente, a E4 pode operar em dois modos distintos:

- No modo gravação (*Recording Mode*) graças a uma memória flash que lhe permite gravar até 60 horas de dados. Os dados podem, posteriormente, ser enviados para a *Cloud* através do programa *desktop* E4 Manager e mais tarde visualizados através da plataforma *Web* E4 Connect. Nesta plataforma é ainda possível descarregar os dados em bruto no formato CSV para futuras análises ou processamento.

Figura 18: Representação do Recording Mode (*E4 wristband | Empatica*, 2020).

- No modo de transmissão (*Streaming Mode*) para ser possível a visualização em tempo real dos dados num *smartphone*. Neste modo, a E4 conecta-se à aplicação E4 realtime através do BLE e assim será possível visualizar todos os dados enviados em formato de texto e visualizar graficamente o EDA e o BVP.

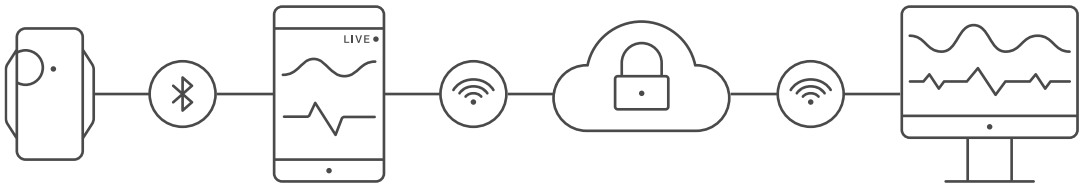

Figura 19: Representação do Streaming Mode (E4 wristband | Empatica, 2020).

A Figura 20 apresenta as aplicações que retratam os dois modos de funcionamento inerentes à E4.

Figura 20: E4 Connect e E4 realtime.

O dispositivo possui um botão, que para além de ligar e desligar o equipamento, permite sinalizar eventos de interesse para o utilizador, que serão representados como uma linha vertical nos gráficos obtidos. Este possui, ainda, um led indicador de estado que informa algumas situações de acordo com a cor que estiver ativa:

- Amarelo – indica que a E4 está a carregar ou tem pouca bateria;
- Verde – indica que o dispositivo iniciou e aguarda conexão *Bluetooth*;
- Roxo – indica que entrou no *Recording Mode*;
- Azul – indica que entrou no *Streaming Mode*;

As especificações técnicas do dispositivo encontram-se mencionadas na Tabela 4.

Tabela 4: Especificações técnicas da E4.

Dimensões	Caixa: 44x40x16mm Bracelete: 110 -190mm
Peso	25g
Autonomia	*Streaming mode*: 24h *Recording Mode*: 32h
Memória	60h de armazenamento
Transferência de Dados	BLE USB 2.0

Além do mais, a Empatica fornece um conjunto de soluções que permite desenvolver aplicativos para Windows, Android e IOS, projetados para a transmissão de dados em tempo real. Para se criar os próprios aplicativos é necessário um API *Key* e SDk(s) para a(s) plataforma(s) escolhida(s) que podem ser obtidos na página pessoal do utilizador.

E4 link é a denominação dada à biblioteca que é utilizada para desenvolver aplicações Android e IOS. Para o Android é necessário ter no mínimo o nível 19 do Android SDK API e para IOS exige-se o mínimo *deployment target* IOS 9.3.

E4 streaming server é a aplicação Windows que funciona como intermediário na comunicação entre a E4 e a aplicação desenvolvida e opera, exclusivamente, com o BlueGiga BLED112 *Bluetooth Smart Dongle.*

4.4. Firebase

Nesta secção é esclarecido todo o processo relativo à utilização e configuração dos serviços pretendidos disponibilizados pelo Firebase.

Previamente à configuração dos serviços pretendidos, foi necessário criar um projeto no Firebase através da consola disponível no *Web*site. Após a criação deste projeto, foi necessário adicionar o Firebase a cada uma das aplicações desenvolvidas (app *Web* e app Android), como será demonstrado mais adiante nas secções respetivas.

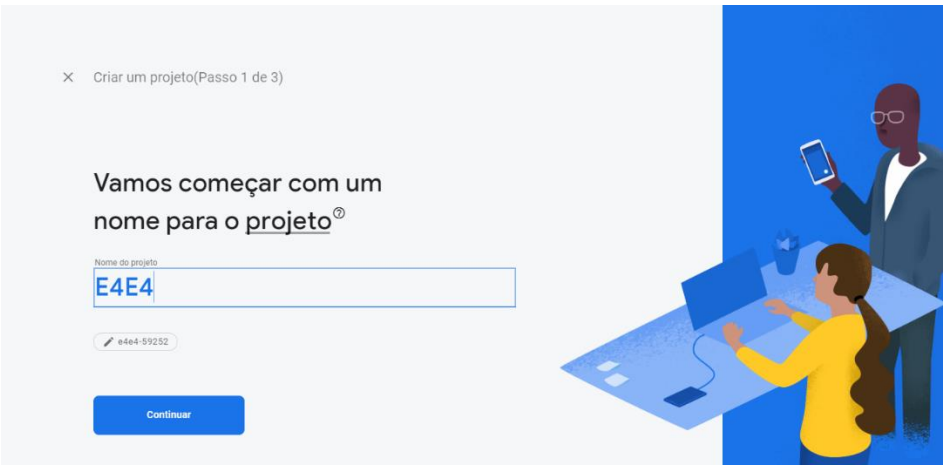

Figura 21: Criação de um projeto no Firebase.

4.4.1. Base de Dados

A base de dados do Firebase é uma base de dados do tipo não-relacional (NoSQL), ou seja, a gravação dos dados não foi efetuada no formato de tabelas. Contrapondo, os dados foram armazenados num ficheiro JSON que contém múltiplos objetos. Os objetos JSON consistem num par chave/valor, onde a chave identifica o objeto e o valor representa os dados guardados. Vários destes objetos formam o que se designa uma árvore JSON.

Tendo em consideração os dados que são enviados pela E4 e a forma como se pretende que estes sejam extraídos da base de dados, organizou-se o ficheiro JSON por sessões, onde cada sessão corresponde a uma transferência de dados entre a E4 e o Firebase ao longo de um intervalo de tempo. Assim, de forma a evitar uma estrutura em ninho, organizou-se o ficheiro JSON da seguinte forma:

- Dispositivo a transmitir;
- Lista de sessões;
- Última sessão;
- Sessão.

```
{
  "Dispositivos": {},
  "Lista Sessoes": {},
  "Sessao no 53": {},
  "Sessao no 54": {},
  "Ultima Sessao": 54
}
```

Esta estruturação é definida na aplicação Android através de classes Java, visto que esta aplicação está encarregue do envio dos dados da E4 para o Firebase. A utilização de classes permite um código mais legível e facilita a sua manutenção. Quando ocorre uma escrita através do SDK da Firebase o objeto da classe a ser escrita é automaticamente mapeado no caminho específico do ficheiro JSON com a mesma estrutura da classe.

O primeiro objeto do ficheiro JSON corresponde a "Dispositivos" e contém uma lista de objetos que identifica as *wristband* E4 pelo seu nome. A esses objetos está associado uma variável do tipo *bool* que indica se o dispositivo se encontra a transmitir, *true*, ou se está indisponível, *false*. Sempre que uma nova E4 for utilizada, esta será adicionada ao objeto "Dispositivos". No caso de uma E4 já constar no ficheiro, o seu conteúdo chave/valor será reescrito.

```
"Dispositivos": {
    "Empatica E4 - A020FE": {
      "ligado": false
    },
    "Empatica E4 - A02106": {
      "ligado": false
    }
  }
```

O campo "Lista Sessões" contém uma lista de identificadores únicos gerados cada vez que uma sessão é iniciada na aplicação Android. Estes identificadores são gerados automaticamente pelo Firebase quando há uma inserção de um novo filho neste local e representam uma sessão. Cada um deles contém a data e hora em que se iniciou a sessão, o nome da respetiva E4 e um número inteiro que vai sendo incrementado sempre que há uma nova sessão.

```
"Lista Sessoes": {
  "-MBK7x8FnPAYkhzrodwn": { // identificador gerado pelo FIrebase
    "data": "03-07-2020 15:45:35",
    "nome": "Empatica E4 - A020FE",
    "numero": 53
  },
  "-MBK91Z3-D73LMxMEcLJ": {
    "data": "03-07-2020 15:50:20",
    "nome": "Empatica E4 - A020FE",
    "numero": 54
  }
},
```

Os dados sensoriais enviados são guardados no campo "Sessao no X", onde X representa o número da sessão que também será guardado em "Lista Sessoes". Cada sensor possui um campo respetivo onde os dados são escritos, como por exemplo no campo "EDA" serão guardadas as informações relativas ao sinal EDA capturado pela E4. Caso os dados de um sensor não sejam transmitidos, não irá constar na base de dados um campo específico para tal. Além dos sensores, existe ainda um objeto que contém informação relativamente aos dados sensoriais que estão a ser enviados, através de variáveis do tipo *bool*. Este objeto foi adicionado com o intuito de facilitar o ajuste automático do gráfico conforme a disponibilidade dos dados dos sensores presentes numa determinada sessão.

```
"Sessao no 53": {},
"Sessao no 54": {
  "EDA": {},
  "Sensores": {
    "acc": false,
    "bvp": false,
    "eda": true,
    "hr": false,
    "temp": true
  },
  "Temp": {}
}
```

O campo correspondente a cada sensor possui uma lista de identificadores que contêm o *timestamp* e o valor obtido pela E4 no momento da obtenção do dado sensorial. O *timestamp* corresponde ao número de segundos passados desde 1 de janeiro de 1970, em *Coordinated Universal Time* (UTC). A utilização dos identificadores neste campo é de particular importância, visto que desta forma é garantida

uma organização cronológica dos eventos gravados e se assegura a impossibilidade de se efetuar a reescrita nesta localização.

```
"EDA": {
      "-MBK91qFmgK1XeZs0KU4": {
        "data": 2.8542990684509277,
        "timestamp": 1593787820.2011025
      },
      "-MBK91qG7P2UzPqWHH4A": {
        "data": 2.8670992851257324,
        "timestamp": 1593787820.4511025
      },
      "-MBK91qG7P2UzPqWHH4B": {
        "data": 2.8734993934631348,
        "timestamp": 1593787820.7011025
      },
      ...
   }

```

No final do ficheiro JSON está definida a chave "Ultima Sessao", em que o valor correspondente indica o valor numérico da última sessão transmitida. Sempre que uma nova sessão é iniciada este valor é incrementado. Este campo é utilizado para que do lado da aplicação Android se saiba qual a última sessão transmitida para que, de seguida, possa ser incrementada a variável na lista de sessões.

```
"Ultima Sessao": 54
```

Por fim, foram definidas as regras de segurança relativas à base de dados através de um separador próprio na consola do Firebase. As regras são definidas no formato JSON e aplicam-se tanto para a leitura, como para a escrita na base de dados.

```
{
  "rules": {
    ".read": " auth.token.email_verified==true && auth.token.email.matches(/.*alunos.uminho.pt$/) ",
    ".write": " auth.token.email_verified==true && auth.token.email.matches(/.*alunos.uminho.pt$/) "
  }
}
```

A propriedade "auth.token.email_verified==true" corresponde a um utilizador autenticado e com email validado, enquanto que a propriedade "auth.token.email.matches(/.*alunos.uminho.pt$/)"

representa um email que corresponda à expressão compreendida entre parênteses. A combinação destas duas propriedades autoriza apenas a utilizadores deste tipo ler e escrever na base de dados. O email institucional da Universidade do Minho foi utilizado para simular um email de uma instituição de cuidados ou de um centro de saúde onde este sistema poderá vir a ser implementado. Desta forma previne-se possíveis adulterações da base dados, como alterações dos dados existentes ou até a sua eliminação. As regras não restringem a leitura apenas ao que foi escrito pelo utilizador, no sentido que cada utilizador só pode ter acesso ao que escreveu, como profissionais de saúde ou cuidadores que podem ter acesso aos dados de todos os dispositivos.

4.4.2. Autenticação

Para assegurar a segurança dos dados que são disponibilizados na aplicação *Web* utilizou-se o serviço de autenticação do Firebase. Este serviço garante uma barreira de acesso aos dados, que, juntamente com a definição de regras de consulta e escrita na base de dados, impedem o acesso direto e rápido a qualquer pessoa que aceda à aplicação.

O método de autenticação escolhido consiste na inserção de um *email* e *password*. Este método foi ativado através do painel lateral da consola do Firebase na opção "Authentication". Depois no separador "Sign-in method" selecionou-se a opção "E-mail/senha".

No separador "Users" encontram-se listadas todas as contas registadas, sendo ainda possível excluir a conta de um utilizador, desativar uma conta impedindo o seu *login* e enviar um *email* para o utilizador redefinir a sua senha de acesso.

Figura 22: Separador "Users" e opções permitidas.

4.4.3. Hospedagem

O Firebase Hosting é mais um dos serviços disponibilizados pelo Firebase que permitiu a hospedagem da aplicação *Web* desenvolvida. É um serviço direcionado para aplicativos *Web front-end* e destaca-se pela sua rapidez e segurança. Os arquivos carregados são armazenados em *cache* em SSDs (*Solid State Drives*) na rede de distribuição de conteúdo (CDN – *Content Delivery Network*) e, posteriormente, são entregues aos visitantes, de acordo com a sua localização geográfica de modo a se efetuar a ligação com o servidor mais próximo e mais rápido, reduzindo, assim, o tempo de transferência dos dados. O certificado SSL (*Secure Sockets Layer)* é incorporado a este serviço para assegurar que o conteúdo seja sempre entregue em segurança.

Através da linha de comandos do Firebase foi possível hospedar os ficheiros da aplicação *Web* no servidor de *hosting*. Toda esta informação encontra-se descrita na subsecção seguinte.

4.4.4. Firebase Command Line Interface

A Firebase Command Line Interface (CLI) dispõem de um conjunto de ferramentas deveras úteis para testar, gerir e implementar um projeto a partir da linha de comandos. Este ambiente permite fazer operações que de outra forma não são possíveis de concretizar na consola da *Web* do Firebase. No caso desta dissertação, a interface foi utilizada para:

- interação direta com a base de dados para remover dados de grande volume;
- executar um servidor *Web* localmente e hospedagem da *Web* app.

Para instalar o Firebase CLI foi, previamente, necessário instalar o Node Package Manager (npm)(*Node.js*, sem data). O npm é o gerenciador de pacotes do Node.js - interpretador de linguagem Javascript orientado para executar scripts do lado do servidor.

Logo após a instalação do Node.js, através da linha de comandos, recorreu-se ao comando npm para instalar o CLI do Firebase.

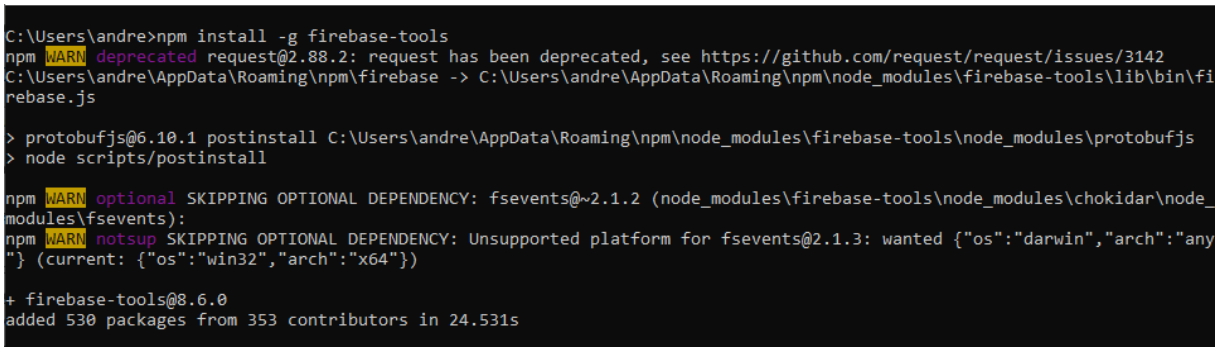

Figura 23: Comando de instalação do CLI.

Com a instalação do CLI dada como concluída pode-se, então, iniciar a interação com o ambiente. Todas as instruções a efetuar no CLI do Firebase são precedidas da palavra "firebase". Para aceder ao projeto foi necessário, em primeiro lugar, proceder à autenticação com o email de registo no Firebase. Depois de confirmada a autenticação, a comunicação entre o computador e o Firebase é estabelecida e o acesso aos projetos fica disponível. Verificou-se que tudo ocorreu corretamente visualizando os projetos existentes através do comando que lista os projetos. No caso apresentado, só se encontrava um projeto disponível pois só um é que tinha sido criado.

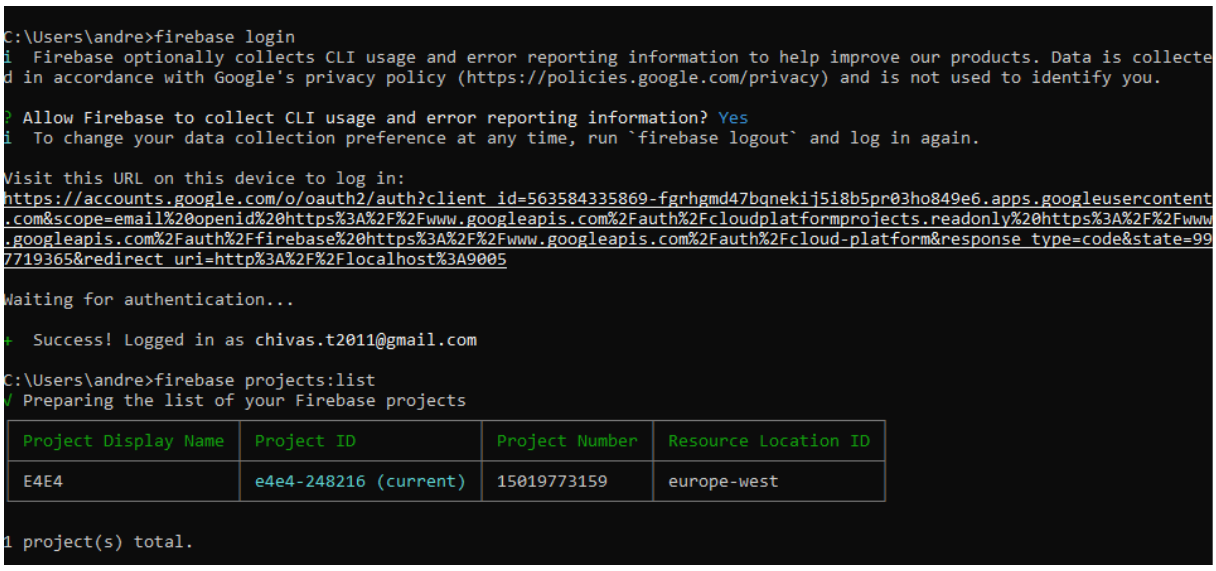

Figura 24: Login realizado com sucesso e lista de projetos disponíveis.

No seguimento do *login* realizado com sucesso, utilizou-se o comando "firebase init" que é necessário para tarefas comuns executadas no CLI, como a implementação de um projeto. Durante a execução deste processo é pedida a escolha de um recurso para configurar no projeto. Esta etapa associa o diretório local escolhido a um projeto Firebase, de modo a que os comandos sejam executados no devido projeto Firebase. No final da execução, para qualquer um dos recursos apresentados, são

adicionados ao diretório um ficheiro de configuração denominado firebase.json e um ficheiro contendo o nome do projeto.

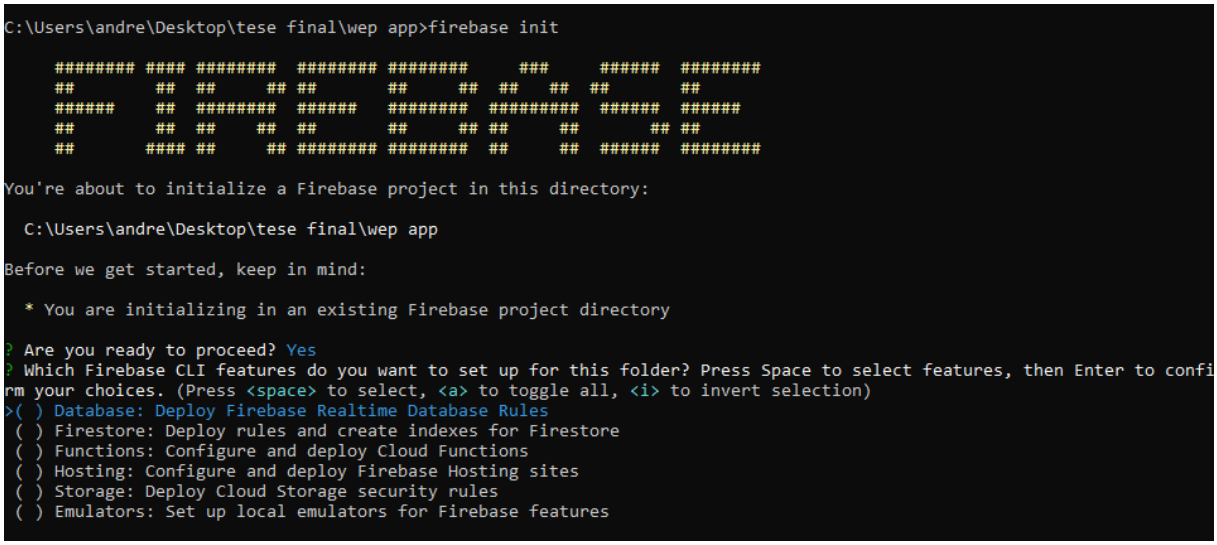

Figura 25: Retorno do comando 'firebase init'.

Como se pretendia usar o serviço de Hosting do Firebase para hospedar a aplicação *Web* desenvolvida, escolheu-se, então, esse recurso no menu apresentado pelo comando 'firebase init'. No final gerou-se um novo diretório denominado public que contém os arquivos index.html e 404.html. Todos os outros arquivos desenvolvidos para a criação da *Web app* foram, posteriormente, colocados nesta pasta.

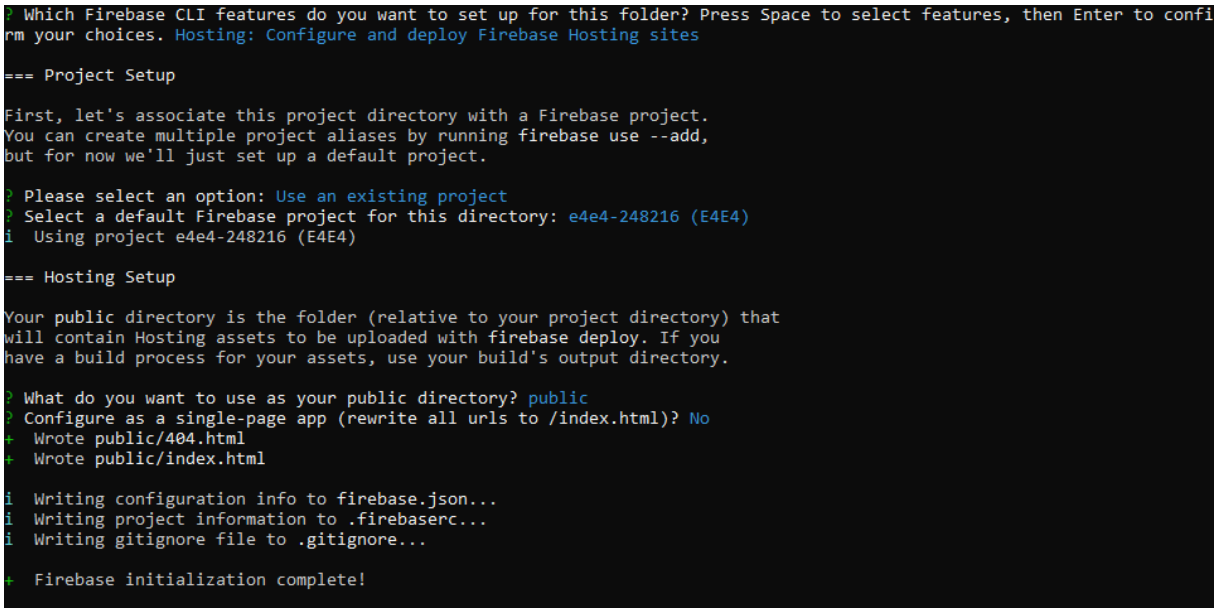

Figura 26: Continuação do retorno do comando 'firebase init' para a escolha do serviço de hosting.

Antes de fazer o upload da pasta public para os servidores, inseriu-se o comando "firebase serve". Este comando hospeda a aplicação na própria máquina, que pode ser consultada através do seguinte link - http://localhost:5000. Desta forma foi possível visualizar e testar a aplicação antes de esta ser hospedada *online*.

```
C:\Users\andre\Desktop\tese final\wep app>firebase serve

=== Serving from 'C:\Users\andre\Desktop\tese final\wep app'...

i  hosting: Serving hosting files from: public
+  hosting: Local server: http://localhost:5000
```

Figura 27: Comando Firebase serve.

Quando a aplicação se encontrava de acordo com o pretendido utilizou-se, finalmente, o comando de deploy que implementa a *Web* app nos sites de hospedagem padrão do Firebase:

- projectID.*Web*.app
- projectID.firebaseapp.com

Na área de Hosting na consola do Firebase alterou-se a configuração de armazenamento para suportar até 3 versões da aplicação *Web* desenvolvida. Assim, aquando da sua necessidade, é possível reverter a app para uma versão anterior. Esta ação é proveitosa, caso a versão em vigor tenha algum problema, e desta forma, é possível reverter para uma versão funcional enquanto se procede à averiguação dos problemas presentes na versão atual.

```
C:\Users\andre\Desktop\tese final\wep app>firebase deploy

=== Deploying to 'e4e4-248216'...

i  deploying hosting
i  hosting[e4e4-248216]: beginning deploy...
i  hosting[e4e4-248216]: found 2 files in public
+  hosting[e4e4-248216]: file upload complete
i  hosting[e4e4-248216]: finalizing version...
+  hosting[e4e4-248216]: version finalized
i  hosting[e4e4-248216]: releasing new version...
+  hosting[e4e4-248216]: release complete

+  Deploy complete!

Project Console: https://console.firebase.google.com/project/e4e4-248216/overview
Hosting URL: https://e4e4-248216.web.app
```

Figura 28: Comando Firebase deploy.

Caso seja necessário desabilitar o *Hosting*, para que a aplicação não seja mais visível *online*, ou até removê-la por completo, utiliza-se o comando de *disable* através do CLI. A partir do momento em que se encerra a execução, a aplicação fica inativa e deixa de estar acessível, contudo ainda permanece na consola do Firebase para o caso de se querer reverter esta ação ou remover a aplicação por completo do servidor.

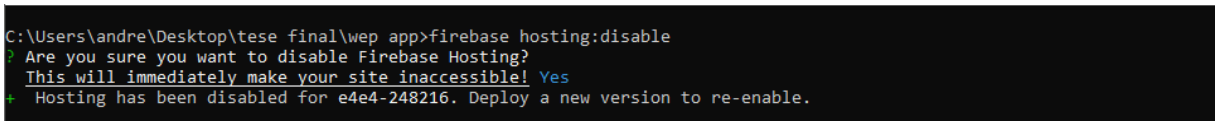

Figura 29: Comando para desabilitar o Hosting.

O Firebase limita a interação com a base de dados através da consola na presença de grandes quantidades de dados. No caso de o utilizador necessitar de excluir largas quantidades de dados ou removê-los totalmente da base de dados, torna-se inviável a realização desta operação tanto pela consola, como pelo API das linguagens de programação disponíveis. Uma solução rápida e simples para eliminar todos os dados passa por importar um ficheiro JSON vazio através da consola do Firebase. Uma outra solução consiste na utilização do comando "database:remove" seguido do caminho específico que se pretende remover. No entanto, é necessário completar o "Project Setup" através do "firebase init" (Figura 30) para este comando funcionar.

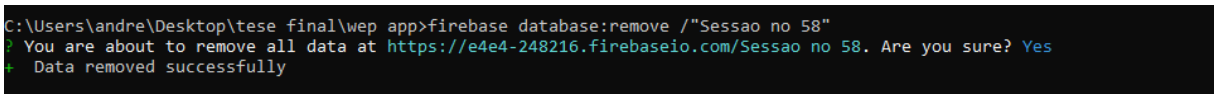

Figura 30: Comando para remover dados numa localização específica.

Para limpar a base de dados completa basta recorrer ao mesmo comando na raiz do projeto, ou seja, executar "firebase database:remove /".

4.5. Aplicação Android

Nesta secção é abordada a aplicação Android desenvolvida que permite receber os dados da *wristband* E4, apresentá-los e enviá-los para o Firebase. Para conectar a E4 via BLE utilizou-se o SDK disponibilizado na página pessoal do *Web*site da Empatica e para o Firebase foi seguida a API disponível. A aplicação foi desenvolvida utilizando o IDE Android Studio.

4.5.1. Dependências

As dependências permitem incluir bibliotecas externas, arquivos JAR ou outros módulos externos ao projeto. Para ser possível trabalhar com a E4, foi necessário adicionar algumas dependências para integrar o Firebase e a biblioteca. Após adicionar as dependências, sincronizou-se o projeto de modo a garantir que todas as dependências têm as respetivas versões necessárias. Na Figura 31 encontram-se todas as dependências utilizadas para o desenvolvimento da aplicação.

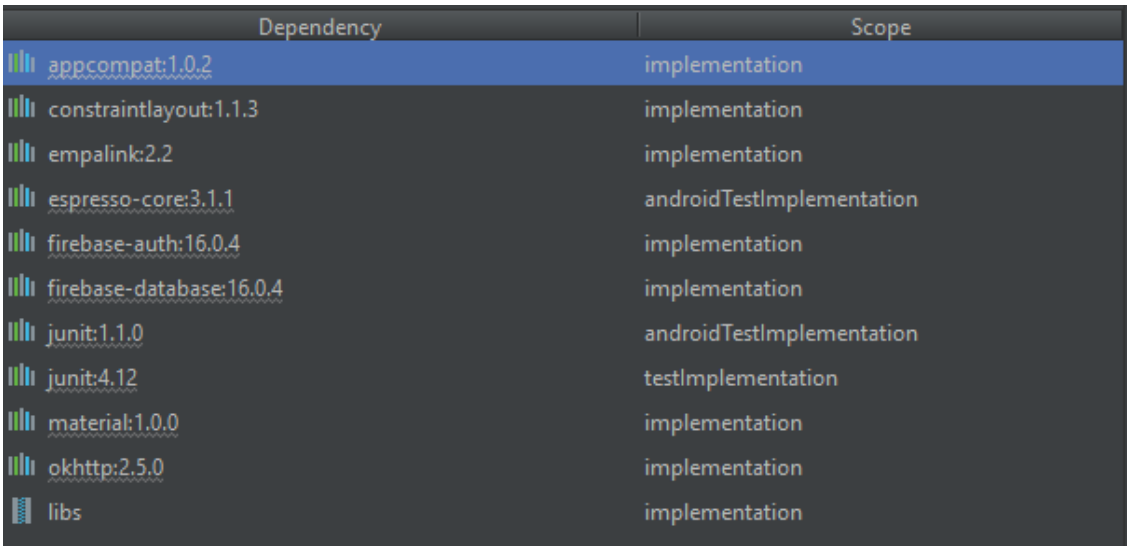

Figura 31: Dependências utilizadas no desenvolvimento da aplicação.

4.5.1.1. Dependências da E4

1. Primeiramente descarregou-se o SDK disponível na área de desenvolvedor do site da Empatica e copiou-se o arquivo E4link-2.2.aar para a pasta libs que se encontra incluída na pasta app.
2. De seguida, no build.gradle (project level) alterou-se as linhas finais para:

```
allprojects {
  repositories {
    jcenter()
    flatDir {
      dirs 'libs'
    }
  }
      }
```

3. Por fim, no app build.gradle adicionou-se ao bloco das dependências as seguintes linhas:

```
implementation 'com.squareup.okhttp:okhttp:2.5.0'
implementation 'com.empatica.empalink:empalink:2.2@aar'
```

4.5.1.2. Dependências do Firebase

1. Adicionar o ficheiro de configuração do Firebase

Quando se registou a app Android logo a seguir à criação do projeto (através da consola do Firebase), descarregou-se o arquivo google-service.json. Este ficheiro contém as configurações do Firebase para o Android e, em seguida, foi movido para dentro da raiz do módulo app.

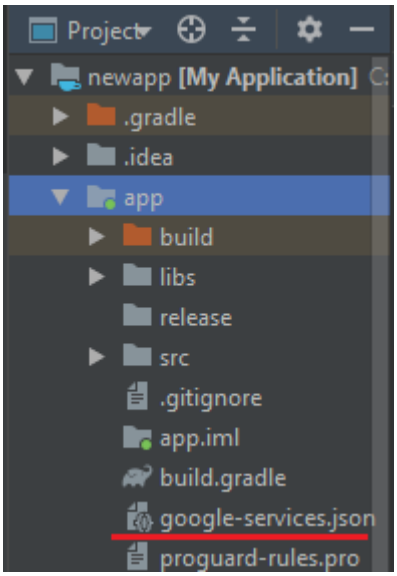

Figura 32: Localização do arquivo google-services.json.

Posteriormente, adicionou-se ao build.gradle(project-level) os plugins do google-services (Figura 33), que são ativados no build.gradle(app-level) através da seguinte linha de código:

```
apply plugin: 'com.google.gms.google-services'
```

```
buildscript {
    repositories {
        google()
        jcenter()

    }
    dependencies {
        classpath 'com.android.tools.build:gradle:3.5.3'
        classpath 'com.google.gms:google-services:4.2.0'

        // NOTE: Do not place your application dependencies here; they belong
        // in the individual module build.gradle files
    }
}

allprojects {
    repositories {
        jcenter()
        flatDir {
            dirs 'libs'
        }
        google()
    }
}
```

Figura 33: Dependências do build.gradle(project-level).

2. Adicionar SDK dos serviços do Firebase

No build.gradle(app-level) adicionou-se, por fim, as dependências correspondentes aos serviços utilizados. No caso desta dissertação, o serviço de autenticação e o serviço da base de dados foram adicionados no bloco dependencies, respetivamente:

```
implementation 'com.google.firebase:firebase-auth:16.0.4'
implementation 'com.google.firebase:firebase-database:16.0.4'
```

4.5.2. Fluxograma da Aplicação

A aplicação desenvolvida é composta por uma única atividade e o seu fluxograma encontra-se representado na Figura 34. Nessa *Activity* referida como "Main Activity" são apresentados os dados recebidos em formato de texto, realiza-se a procura e conexão à E4 e são transmitidos os dados para a base de dados do Firebase.

No início da aplicação no método onCreate() faz-se a verificação de conexão à Internet, condição necessária para se efetuar a autenticação e transmissão dos dados. A autenticação é efetuada sem a perceção do utilizador através de um email e password previamente guardados na área de autenticação do Firebase. Assim, a confidencialidade dos dados e a simplicidade de interação com app foi desenvolvida conforme o público a que se aplica. Caso se verifique a conexão à Internet, é apresentado na tela uma lista dos sensores correspondentes aos sensores da *wristband* E4 e um botão de procura deste dispositivo. Caso haja falhas na ligação o utilizador é notificado e a aplicação é encerrada.

Quando o utilizar seleciona a procura do dispositivo, a aplicação permanece neste estado até que um dispositivo seja encontrado e seja efetuada a ligação. Encontrando-se a ligação estabelecida os dados começam a ser imediatamente transmitidos da E4 para aplicação, onde são exibidos. De seguida, através de caixas de seleção, o utilizador tem a capacidade de selecionar os dados que pretende transmitir para o Firebase ao pressionar os respetivos botões para esse efeito. Caso nenhuma caixa seja selecionada, o envio não acontece e um alerta é apresentado para que o utilizador selecione pelo menos uma caixa. Quando se inicia uma transmissão, o envio de dados acontece de forma contínua até que o utilizador deseje interromper esta ação. Ao parar, a aplicação retorna ao seu estado inicial.

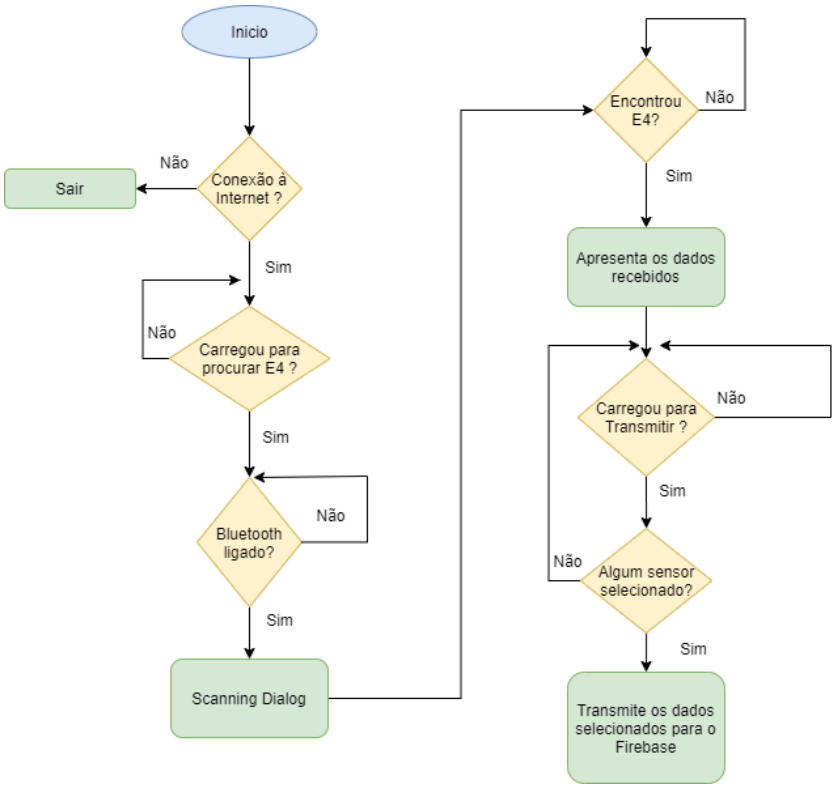

Figura 34: Fluxograma da aplicação desenvolvida.

4.5.3. Biblioteca da Empatica

A biblioteca adicionada à pasta *libs* apresenta um conjunto de classes com as quais se tem de trabalhar para interagir com a *wristband*.

- A EmpaticaDeviceManager é a classe principal que permite interagir com dipositivos E4 e fornece métodos para procura, conexão e desconexão.
- A interface EmpaStatusDelegate disponibiliza métodos para receber informação sobre o estado do DeviceManager.
- A interface EmpaDataDelegate disponibiliza métodos para receber os dados do dispositivo E4. Os métodos disponíveis encontram-se descritos na Tabela 5.

Tabela 5: Métodos java para receber em tempo real os dados do E4.

Retorno	Método	Descrição
void	didReceiveGSR (float gsr, double timestamp)	Invocado quando um novo valor GSR fica disponível
void	didReceiveBVP (float bvp, double timestamp)	Invocado quando um novo valor BVP fica disponível
void	didReceiveIBI (float ibi, double timestamp)	Invocado quando um novo valor IBI fica disponível
void	didReceiveTemperature (float t, double timestamp)	Invocado quando um novo valor de temperatura fica disponível
void	didReceiveAcceleration (int x, int y, int z, double timestamp)	Invocado quando um novo valor de aceleração fica disponível
void	didReceiveBatteryLevel (float level, double timestamp)	Invocado quando um novo valor de bateria fica disponível

4.5.4. Procura e Conexão de Dispositivos

Para utilizar os recursos do *Bluetooth* foi necessário conceder algumas permissões, definidas no ficheiro Android*Manifest.xml* da aplicação como se pode ver na Figura 35. A primeira permissão é essencial para executar qualquer comunicação *Bluetooth* e a segunda é necessária para executar a procura de dispositivos. Foi, ainda, indispensável adicionar permissão para acesso à localização pois os dispositivos detetáveis podem revelar informações sobre a localização do utilizador.

```
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

Figura 35: Permissões necessárias para utilizar o Bluetooth.

Antes de se proceder à procura, é verificado se o *Bluetooth* se encontra ligado. Caso não seja possível efetuar a ligação via *Bluetooth*, é exibido um diálogo que indica ao utilizador a necessidade da sua ativação (Figura 36).

```
if (!mBluetoothAdapter.isEnabled()) {
    Intent enableBtIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
    startActivityForResult(enableBtIntent, REQUEST_ENABLE_BT);
}
```

Figura 36: Verificação se o Bluetooth está ativo.

O primeiro passo que foi realizado para implementar a procura consistiu em instanciar um objeto da classe EmpaDeviceManager, passando-lhe o *context* da aplicação e referências das classes EmpaDataDelegate e EmpaStatusDelegate. A API Key disponível na página pessoal do desenvolvedor deve ser inserida para ativar o DeviceManager e para que seja possível ligar exclusivamente às E4 adquiridas, caso contrário não seria possível efetuar a ligação.

```
deviceManager = new EmpaDeviceManager(getApplicationContext(),  dataDelegate: this,  statusDelegate: this);
deviceManager.authenticateWithAPIKey(EMPATICA_API_KEY);
```

Figura 37: Instância da classe EmpaDeviceManager.

De seguida, através da interface EmpaStatusDelegate é recebido o estado READY via função didUpdateStatus() quando o DeviceManager está pronto a ser utilizado. Assim, será possível iniciar a procura por dispositivos através da função startScanning() do DeviceManager.

```
@Override
public void didUpdateStatus(EmpaStatus status) {
    //...
    if (status == EmpaStatus.READY) {
        //...

        deviceManager.startScanning();
        Toast.makeText( context: MainActivity.this,  text: "Start scanning", Toast.LENGTH_SHORT).show();
        /*...*/
        loadingDialog.setTexto("Scanning...");
        loadingDialog.show(getSupportFragmentManager(), tag: "MyFragLoadDialog");
        /*...*/
```

Figura 38: Procura de dispositivos.

Se um dispositivo se encontrar no alcance, a função de *callback* didDiscoverDevice é chamada. Nesse momento, para-se a procura de dispositivos e faz-se a conexão ao dispositivo se API *Key* corresponder (Figura 39). Caso a conexão seja bem-sucedida, o dispositivo começa a transmitir os dados correspondentes a cada sensor que se encontram acessíveis através dos métodos de *callback* do EmpaDataDelegate.

```
@Override
public void didDiscoverDevice(EmpaticaDevice device, String deviceName, int rssi, boolean allowed) {
..
    if (allowed) {

        deviceManager.stopScanning();
        Toast.makeText( context: MainActivity.this,  text: "Stop scanning", Toast.LENGTH_SHORT).show();
        Toast.makeText( context: MainActivity.this, deviceName, Toast.LENGTH_SHORT).show();
        loadingDialog.dismiss();
        try {
            // Connect to the device
            deviceManager.connectDevice(device);
            device_name=deviceName;
            //...
        } catch (ConnectionNotAllowedException e) {...}
        togglebutton.setEnabled(true);
```

Figura 39: Conexão ao dispositivo.

4.5.5. Transmissão para a Base de Dados

No que diz respeito à transmissão dos dados, recorreu-se ao uso de classes para gravar os dados, tal como se encontra descrito anteriormente na secção do Firebase. A Figura 40 apresenta um esquema das classes utilizadas.

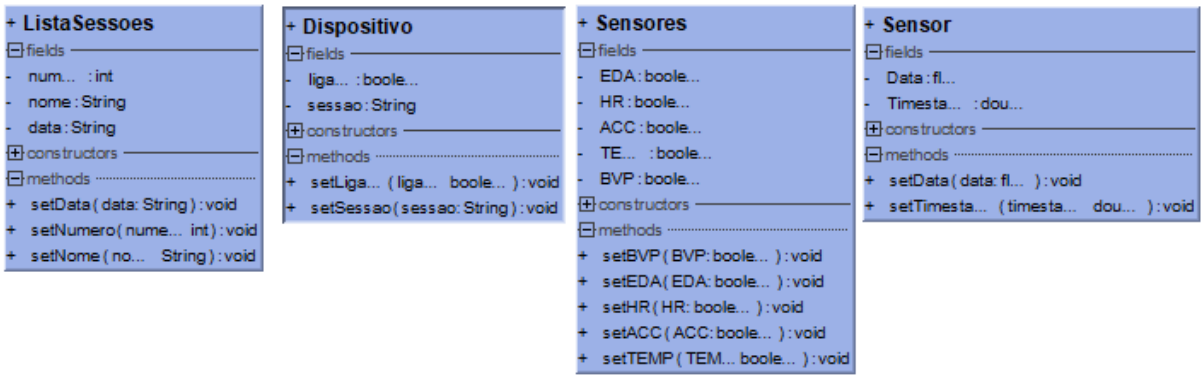

Figura 40: Diagrama das classes utilizadas para a escrita na base de dados.

No entanto, primeiro foi fundamental conhecer o número da última sessão, para que, quando for necessário iniciar uma nova sessão, não se escreva no lugar da anterior. Assim, quando se inicia a procura do dispositivo, efetua-se a leitura do valor da última sessão. Caso seja retornado o valor *null*, significa que ainda não existe nenhuma sessão e, então, guarda-se o valor 1 na variável que conta o número de sessões. No caso de retornar algum valor, incrementa-se esse valor e regista-se o valor incrementado. Assim sendo, foi utilizado o método addLisenerForSingleValueEvent() para ler uma única vez uma referência da base de dados que contém o caminho específico que se quer ler (Figura 41).

```
DatabaseReference sessoesReference = FirebaseDatabase.getInstance().getReference().child("Ultima Sessao");
sessoesReference.addListenerForSingleValueEvent(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
        Toast.makeText( context: MainActivity.this, text: "ULTIMA SESSAO"+dataSnapshot.toString(), Toast.LENGTH_SHORT).show();
        if(dataSnapshot.getValue()==null)
            conta_sessoes=1;
        else {
            try {
                conta_sessoes = Integer.parseInt(dataSnapshot.getValue().toString());
                conta_sessoes = conta_sessoes + 1;
            } catch(NumberFormatException nfe) {
                System.out.println("Could not parse " + nfe);
            }
        }
    }
```

Figura 41: Leitura da última sessão.

No que diz respeito à gravação dos dados, inicia-se com a gravação do valor da última sessão quando se inicia a transmissão através de um botão da aplicação. São também guardados dados para formar uma lista de sessões para futura consulta, nomeadamente o número da sessão, o nome do dispositivo que se encontra a transmitir e a data de início da transmissão. Seguidamente, à medida que os dados vão sendo recebidos nas funções de *callback* do EmpaDataDelegate (Figura 42), é verificado se a *checkbox* relativa ao sensor foi selecionada.

```
@Override
public void didReceiveGSR(float gsr, double timestamp) {
    updateLabel(edaLabel, text: "" + gsr);
    if(togglebutton.isChecked() ) {
        if (edaCheckbox.isChecked())
            writeToDatabase(device_name, SensorName: "EDA", timestamp, gsr);
    }
}
```

Figura 42: Função de callback que recebe os dados relativos ao EDA.

Após verificada a seleção da *checkbox* correspondente, é chamada a função "writeToDatabase". Esta função recebe como parâmetros de entrada o nome do dispositivo que está a transmitir, o nome do sensor, o *timestamp* e os dados. Posteriormente o método setValue() fica encarregue de enviar para o caminho específico da base de dados, um objeto da classe Sensor que contém o valor e o respetivo *timestamp* (Figura 43).

```
private void writeToDatabase(String DeviceName,String SensorName, Double Timestamp, Float Data ){
    sensor.setData(Data);
    sensor.setTimestamp(Timestamp);
    mDatabase= FirebaseDatabase.getInstance().getReference().child("Sessao no "+ String.valueOf(conta_sessoes)).child(SensorName);
    mDatabase.push().setValue(sensor);
}
```

Figura 43: Escrita dos dados na base de dados através de um objeto da classe Sensor.

4.6. Aplicação *Web*

Nesta secção é apresentada a aplicação *Web* que foi desenvolvida com foco na apresentação gráfica em tempo real dos dados enviados pela aplicação Android para o Firebase.

Por questões de segurança criou-se um sistema de registo/*login*, recorrendo ao serviço Authentication do Firebase. Este serviço permite ao utilizador autenticado com sucesso aceder às transmissões em tempo real ou consultar no futuro todos os dados gravados, isto é, após uma sessão terminar. Na Figura 44 é apresentado o diagrama geral da aplicação.

A aplicação foi desenvolvida recorrendo às linguagens de programação *Web*, html, css e javascript e foi utilizado o editor de texto Visual Studio Code.

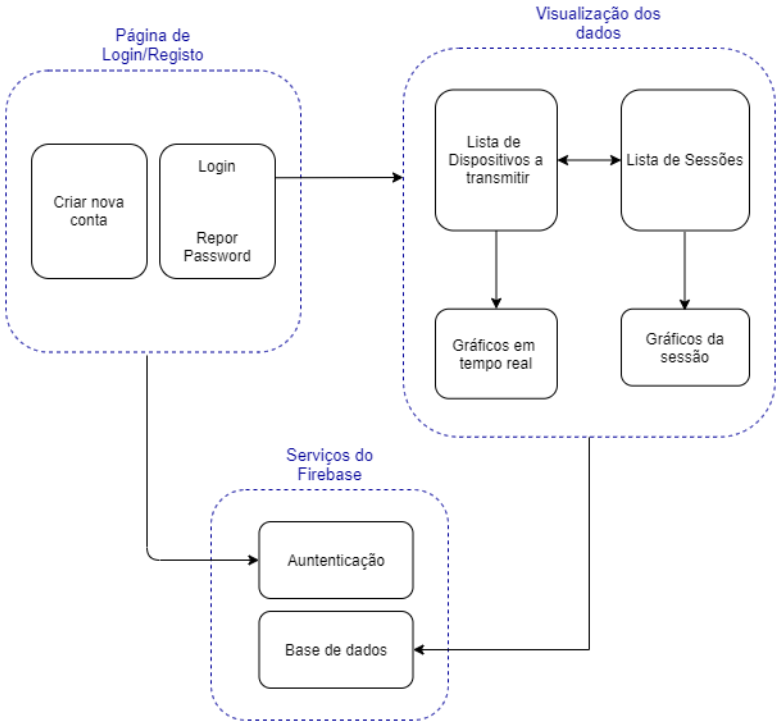

Figura 44: Diagrama geral da aplicação.

Na página de registo/*login* o utilizador pode realizar três operações: efetuar *login*, criar uma nova conta, ou repor a password. Para se repor a password, é necessário aceder-se ao email registado, para o qual foi enviado um *link* para proceder à inserção da nova senha.

Caso o *login* seja efetuado com sucesso, é apresentada uma página onde são listados os dispositivos E4 que se encontram a transmitir. Ao selecionar um desses dispositivos, serão apresentados os gráficos em tempo real correspondentes aos dados que foram selecionados para transmissão na aplicação

Android. O utilizador pode, ainda, optar pela visualização dos dados de sessões passadas, alternando para a página que lista as sessões existentes na barra de navegação criada. Nessa página é permitido visualizar os gráficos da sessão pretendida ou apagar uma sessão permanentemente.

4.6.1. Registo/*Login*

Quando se acede à aplicação, a primeira página apresentada é a página que permite ao utilizador realizar o *login* para, posteriormente, aceder às sessões e às transmissões em tempo, ou criar uma nova conta caso o utilizador não se encontre registado.

Antes de programar o sistema de *login* foi necessário adicionar os SDKs do Firebase à aplicação *Web* (Figura 45). O SDK principal do Firebase que é sempre necessário e os SDKs dos serviços utilizados, que correspondem neste caso, ao serviço de Autenticação e ao da Base de Dados.

```
<!-- Principal SDK do Firebase -->
<script src="https://www.gstatic.com/firebasejs/6.2.0/firebase-app.js"></script>
<!--  SDKs dos serviços utilizados do Firebase -->
<script src="https://www.gstatic.com/firebasejs/6.2.0/firebase-database.js"></script>
<script src="https://www.gstatic.com/firebasejs/6.2.0/firebase-auth.js"></script>
```

Figura 45: SDKs do Firebase necessários.

De seguida, adicionou-se as configurações do projeto Firebase à aplicação *Web* e procedeu-se à sua inicialização (Figura 46). Para isso, através da consola do Firebase, na secção "Visão geral do projeto" selecionou-se a plataforma *Web* e copiou-se o código gerado para um ficheiro javascript da aplicação *Web*.

```
var config= {
    apiKey: "AIzaSyCm-y_rBmSVkDkvILURezrJw_vVdkhhx80",
    authDomain: "e4e4-248216.firebaseapp.com",
    databaseURL: "https://e4e4-248216.firebaseio.com/",
    storageBucket: "e4e4-248216.appspot.com",
};

// Initialize Firebase
firebase.initializeApp(config);
```

Figura 46: Configurações do projeto firebase para a aplicação Web.

Com as configurações do projeto definidas, procedeu-se à implementação do registo do utilizador através da utilização de um email e password passados à posteriori para a função

createUserWithEmailAndPassword() (Figura 47). Nesta função foi implementada a apresentação de avisos aquando da realização do registo ou perante a falha deste. Caso o registo seja criado com sucesso, é apresentada uma mensagem de êxito ao utilizador. Pelo contrário, se ocorrer algum erro é gerada uma mensagem que evidência o motivo para tal ocorrência.

O *login* é semelhante ao registo e apenas se alterou o método utilizado. Neste caso, o método usado foi o *signInWithEmailAndPassword()*. Ambos os métodos recebem como argumentos o email e a password inseridos pelo utilizador.

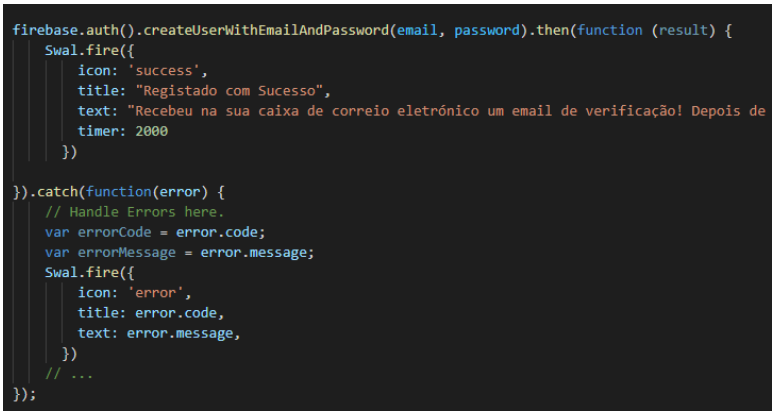

Figura 47: Método createUserWithEmailAndPassword.

Perante um eventual esquecimento da password implementou-se o método *sendPasswordResetEmail()* que recebe o endereço de email para onde será enviado o link de redefinição da password.

Por fim, implementou-se o método mais relevante, método responsável pela verificação da mudança de estado de *login*, ou seja, se o utilizador se encontra conectado ou desconectado. Deste modo, anexou-se um observador ao objeto de autenticação através do método *onAuthStateChanged* (Figura 48). Deste modo, é possível saber o estado de *login* do utilizador uma vez que o observador é chamado sempre que existe uma mudança de estado. É através deste método que se verifica, também, se o email do utilizador que fez *login* é válido.

```
firebase.auth().onAuthStateChanged(function(user) {
  if (user) {

    if (!user.emailVerified) {
      Swal.fire({···

      user.sendEmailVerification().then(function () {//envia email de validação···
      firebase.auth().signOut();// sign-out
      document.getElementById("signup").reset();
    }
    else{
      window.location.replace('transmissao');//redireciona para pagina transmissao
    }

  }
});
```

Figura 48: Método onAuthStateChanged.

No caso de ser válido, o utilizador será redirecionado para a página de transmissão em tempo real. Caso contrário, é enviado um *email* para o *email* de utilizador registado, através do método *sendEmailVerification()*, e o utilizador é obrigado a sair através do método *signOut.* Este último método é também utilizado quando o utilizador deseja terminar a sessão, sendo redirecionado de volta para a página de registo/*login*.

Na Figura 49 encontra-se representado o fluxograma do sistema de registo/*login* que resume a informação acima explicada.

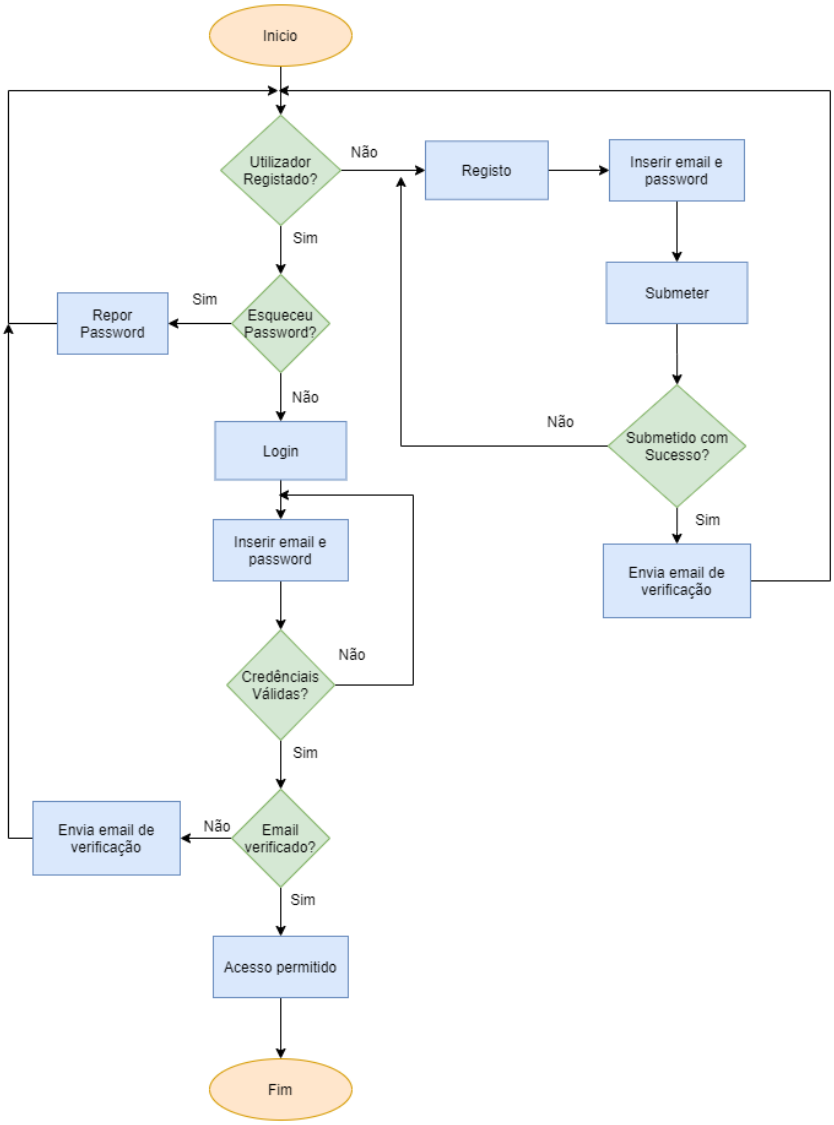

Figura 49:Fluxograma do Sistema de Login/Registo.

4.6.2. Dispositivos a Transmitir

Depois de o *login* ser efetuado com sucesso, o utilizador é redirecionado para a página que apresenta as *wristband* E4 que estão a transmitir dados para o Firebase.

A partir do momento em que se inicia uma transmissão na aplicação Android, imediatamente é escrito na base dados a informação relativa à E4 que se encontra a transmitir, nomeadamente, o seu nome, o número da sessão e uma variável que representa o estado de transmissão. De modo a permitir ao utilizador averiguar e escolher o dispositivo que pretende, foi necessário efetuar do lado da aplicação *Web* uma leitura na base de dados para saber quais os dispositivos que estão a transmitir. Na Figura 50 encontra-se representado fluxograma associado a esta leitura.

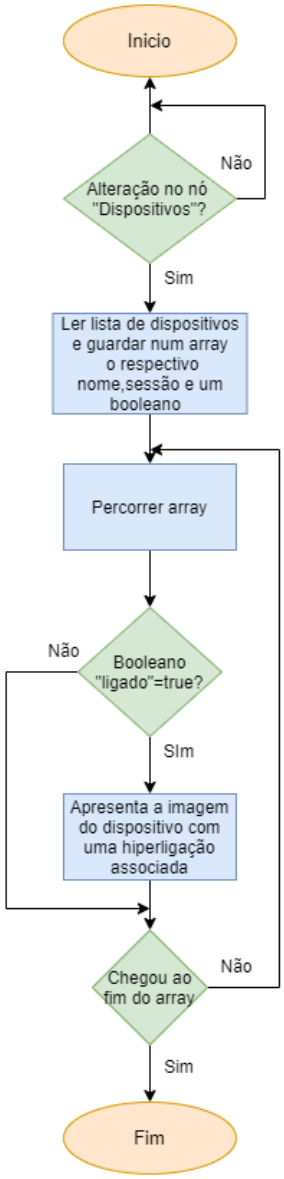

Figura 50: Fluxograma relativo à leitura dos dispositivos a transmitir.

Foi utilizado o método *on()* do *firebase.database.Reference* que possibilita fazer a leitura do conteúdo num determinado caminho ("Dispositivos") através do evento "*value*". Este método é disparado sempre que os dados naquele local são alterados e, para além disso, a função de *callback* do evento recebe um *snapshot* que contém uma lista dos dispositivos a transmitir. De seguida, através de um ciclo *for,* percorre-se a lista e compara-se se o booleano enviado para a base de dados é igual a "true" (Figura 51).

```
var databaseRef = firebase.database().ref().child("Dispositivos");

databaseRef.on('value', function (snapshot) {
    var childData = snapshot.val();
    var DataDispositivos=new Array();
    var i=0;

    snapshot.forEach(function(childSnapshot){
        var dataligado=childSnapshot.val();
        DataDispositivos.push({
            nome: childSnapshot.key,
            transmitir:dataligado.transmitir,
            sessao:dataligado.sessao
        })
        document.getElementById("mydiv").innerHTML = "";

    // Cria imagem do dispositivo com hiperligação asssociada
    if (DataDispositivos[i].transmitir == true) { ···
    });
});
```

Figura 51: Método para a leitura dos dispositivos a transmitir.

Caso o variável *bool* seja verdadeira, gera-se dinamicamente uma imagem do dispositivo a transmitir com uma hiperligação associada, onde é passado como *query string* o número da sessão.

O utilizador ao selecionar a imagem do dispositivo que pretende, será encaminhado para uma página que apresenta os gráficos em tempo real e onde através da *query string* do URL (*Uniform Resource Locator*) é possível efetuar a leitura correta da sessão correspondente.

4.6.3. Gráficos em Tempo Real

Para visualizar os dados recolhidos pela E4 e transmitidos pela aplicação Android para a base de dados do Firebase recorreu-se à biblioteca HighCharts (Highcharts, sem data). Esta é uma biblioteca de gráficos multiplataforma em JavaScript baseada em SVG (Scalable Vector Graphics).

Para ser possível trabalhar com esta biblioteca criou-se um container para o gráfico através da *tag* html e todo o resto da interação foi produzida através da linguagem JavaScript. Em vez de se criar um gráfico para cada sensor, construiu-se um gráfico de linhas com várias séries, onde cada série corresponde aos dados de um sensor, que proporcionou uma melhor análise do conjunto, visto que existe apenas uma linha temporal com as séries sincronizadas.

Na Figura 52 é apresentado o algoritmo desenvolvido para a apresentação do gráfico relativo aos dados que são enviados pela aplicação Android. Ao chegar à página de visualização gráfica, depois de o utilizador escolher o dispositivo associado a uma transmissão de dados, a primeira tarefa a ser realizada consiste na extração do número de sessão através da *query string* do URL.

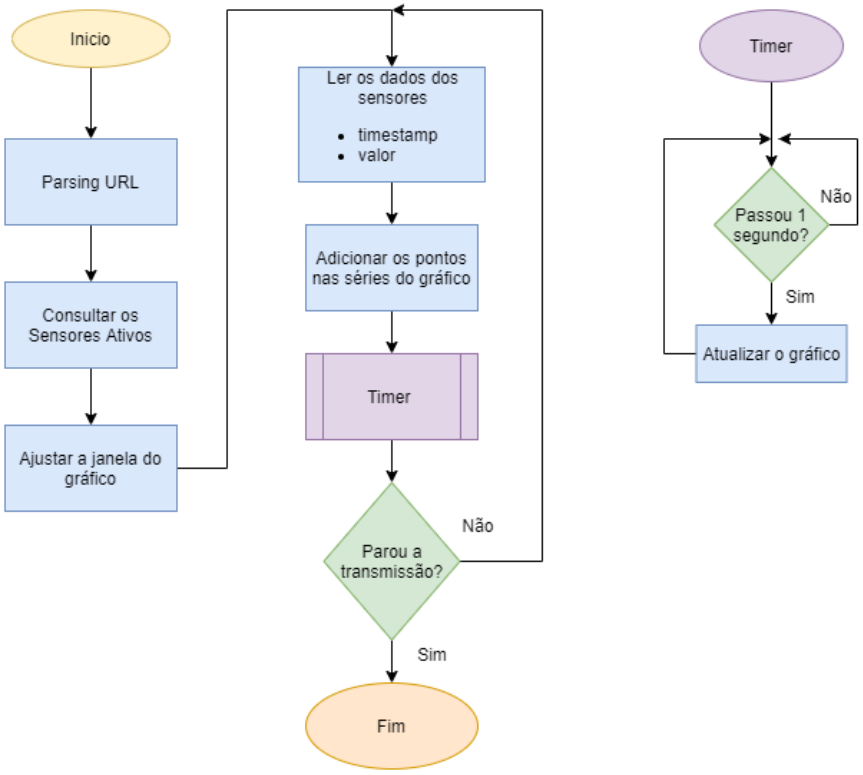

Figura 52: Fluxograma da visualização gráfica em tempo real.

Após obter o número da sessão, procede-se à leitura dos sensores que estão a transmitir na sessão considerada, utilizando o método *once()* ao invés do método *on()*. Este método é disparado apenas uma vez, o que o torna vantajoso, dado que a consulta dos sensores ativos só precisa de ser efetuada uma vez ao entrar na página. Para cada sensor consultado é verificado se o seu valor é igual a "true", o que significa que os seus dados estão a ser guardados na base de dados. Esta verificação foi utilizada de forma a adaptar o desenho dos diferentes sinais no gráfico, isto é, de modo a formatar a altura do gráfico conforme os sensores ativos e impedir a existência de espaços vazios entre sinais, no caso dos sensores não se encontrarem todos disponíveis.

De seguida, realizou-se a leitura dos dados de cada sensor – *timestamp* e valor, utilizando, novamente, o método *on()*. À medida que os dados são recebidos na função de *callback*, estes são imediatamente adicionados às respetivas séries do gráfico através do método *addPoint()*.

```
if (time_counter < 30) {
    EdaChart.series[TEMP].addPoint({
        x: dataFromFirebaseTemp.label * 1000,
        y: dataFromFirebaseTemp.value
    }, false, false);
}
else {
    EdaChart.series[TEMP].addPoint({
        x: dataFromFirebaseTemp.label * 1000,
        y: dataFromFirebaseTemp.value
    }, false, true);
}
```

Figura 53: Método addPoint da biblioteca HighCharts.

O primeiro argumento deste método foi definido como falso, de forma a impedir o redesenhar do gráfico sempre que se inseriu um novo ponto. Este facto impediu que a aplicação ficasse lenta e que perdesse interatividade, caraterísticas associadas à adição de uma grande quantidade de pontos num curto intervalo. Posto isto, implementou-se uma função que a cada 1 segundo chama o método *redraw(),* responsável por redesenhar no gráfico os pontos que foram adicionados. O segundo argumento permite que um ponto adicionado no início seja removido à medida que um novo ponto é adicionado. Este argumento foi definido como "true" quando se verifica que o tempo decorrido desde o início da transmissão alcança os 30s, garantido, assim, uma janela de visualização mais focada num curto espaço de tempo.

Além do que já foi referido, implementou-se um sistema que possibilita ao utilizador definir um *threshold* máximo e mínimo para cada tipo de sinal. Quando esses limites são ultrapassados a cor da linha é atualizada e atua como um alerta para um fenómeno relevante. É através do *array* zones que se define o valor máximo e mínimo pretendidos, de acordo com o input do utilizador, e as respetivas cores. De seguida, o método *update()* é responsável por alterar a cor da série no gráfico.

```
EdaChart.series[0].update({
    zones: [{
        value: minEda,
        color: '#fae505'
    }, {
        value: maxEda,
        color: '#7cb5ec'
    }, {
        color: '#ed2b31'
    }]
})
```

Figura 54: Método para alterar a cor da linha do gráfico.

Por fim adicionou-se um módulo de exportação que permite ao utilizador realizar a transferência do gráfico em PDF ou em diversos formatos de imagem. É, ainda possível a sua impressão sem o resto dos elementos que compõem a página e guardar os dados do gráfico em formato CSV e XLX.

4.6.4. Sessões Disponíveis

A página que lista as sessões foi criada com o intuito de proporcionar ao utilizador uma consulta de todo o historial de sessões realizado, podendo, assim, efetuar uma análise posterior à realizada em tempo real. Nesta página, apresenta-se uma tabela de sessões onde o utilizador pode escolher visualizar uma sessão ou proceder à sua eliminação. A tabela divide-se em 4 colunas, a primeira representa o número de sessão, a segunda o dispositivo que procedeu à transmissão, a terceira a data de início da sessão e, por fim, a última coluna apresenta os botões de visualizar e apagar.

Para preencher a tabela, procedeu-se à criação de uma referência para o caminho "Lista Sessoes" do serviço da base de dados e depois efetuou-se a sua leitura através do método *once()*. Assim, recebe-se um *snapshot* que contém uma lista de objetos com os atributos que se referiu (número de sessão, nome do dispositivo e data de transmissão). Esse *snapshot* é percorrido até ao fim através de um ciclo for, onde em cada iteração se adiciona os atributos e os botões às linhas da tabela. O fluxograma pode ser visualizado na Figura 55.

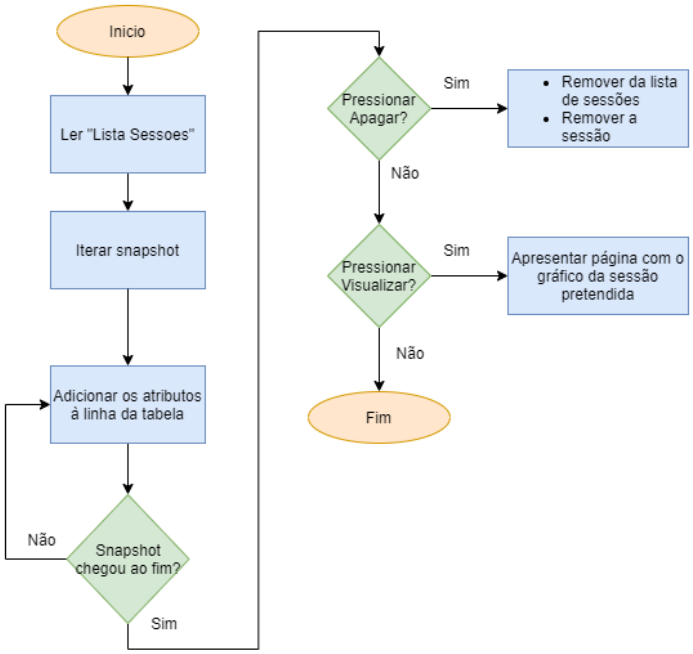

Figura 55: Fluxograma das sessões disponíveis.

O botão "Visualizar" é constituído por uma hiperligação com o número da sessão associado, para que a referência na base de dados seja lida corretamente. O botão de apagar, além de eliminar o nó correspondente na lista de sessões, remove também a sessão em si, ou seja, omite todos os dados transmitidos pela E4. Para efetuar uma remoção, basta criar a referência do caminho que se quer remover e em seguida utilizar o método *remove()* nessa mesma referência (Figura 56).

```
var RefList = firebase.database().ref().child("Lista Sessoes").child(key);
RefList.remove();
var RefSess = firebase.database().ref().child("Sessao no "+id);
RefSess.remove();
```

Figura 56: Remoção da sessão e na lista de sessões.

No final, as sessões encontram-se apresentadas na tabela por ordem crescente, da mais antiga para a mais recente, de acordo com leitura da base de dados. Esta forma de organização pode não ser a mais intuitiva e, por isso, acrescentou-se uma funcionalidade de ordenação da tabela de acordo com o número de sessão e com o nome dos dispositivos. Este facto, permite ao utilizador alternar a visualização cronológica das sessões, apresentando as mais recentes primeiro e as mais antigas no final da tabela e vice-versa, ou agrupar as sessões por dispositivos.

4.6.5. Gráfico Estático

O utilizador, ao selecionar a sessão que pretende visualizar, é encaminhado para uma página que carrega todo o conteúdo da sessão da base de dados e o apresenta num gráfico. Perante este cenário, o utilizador terá acesso à sessão completa, isto é, irá visualizar todo o decorrer da sessão desde o início até ao fim, ao contrário do que ocorre na transmissão em tempo real, onde apenas é possível a visualização de uma janela de 30 segundos em que os dados vão sendo perdidos com o decorrer do tempo.

A leitura da base de dados é realizada através do método *once()*, visto que só é necessário efetuar uma leitura para formar um gráfico estático. O *snapshot* resultante da leitura de cada sensor é iterado de forma a guardar os valores e *timestamps* num *array*.

Para representar os dados do *array* graficamente, usou-se o objeto "options" (Highcharts, sem data) com o objetivo de adicionar a cada série o respetivo *array*, através do método *push().* Este objeto é passado para o construtor do gráfico onde, através do método *reflow()*, se irá ajustar o gráfico ao container .

```
options.series.push(series[Temp]);
chart = new Highcharts.StockChart(options);
chart.reflow();
```

Figura 57: Adição dos valores e timestamps à serie que representa a temperatura.

Do mesmo modo que os gráficos em tempo real, também se efetuou o ajuste automático da área das séries do gráfico para impedir espaços em branco quando os dados de algum sensor não foram transmitidos durante uma sessão.

4.7. Aplicação Windows

Numa primeira fase, em vez das aplicações Android e *web* anteriormente descritas, desenvolveu-se uma aplicação para Windows, utilizando as outras ferramentas disponibilizadas pela Empatica, para comunicar e visualizar graficamente os sinais recolhidos pela E4 wristband.

O sistema implementado necessitou de recorrer ao E4 Streaming Server, aplicação disponibilizada pela Empatica, para interligar a comunicação entre a E4 e um computador através do BlueGiga BLED112 *Bluetooth* Smart Dongle (Figura 58).

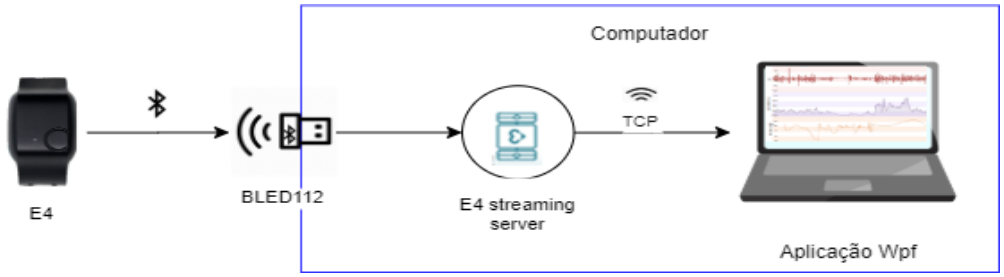

Figura 58: Arquitetura local.

A comunicação entre a *wristband* E4 e aplicação responsável por apresentar graficamente os dados é assegurada pelos protocolos de comunicação BLE e TCP. Numa direção, encontra-se a *wristband* E4 conectada por BLE ao servidor TCP (E4 Streaming Server) através do dongle BLED112. No lado oposto, a aplicação desenvolvida conecta-se ao E4 Streaming Server através de conexão TCP, recebendo os dados provenientes do dispositivo (Figura 59). Para operar o E4 Streaming Server é necessário inserir o *email* da conta criada e uma chave API fornecida pela Empatica quando o equipamento foi adquirido.

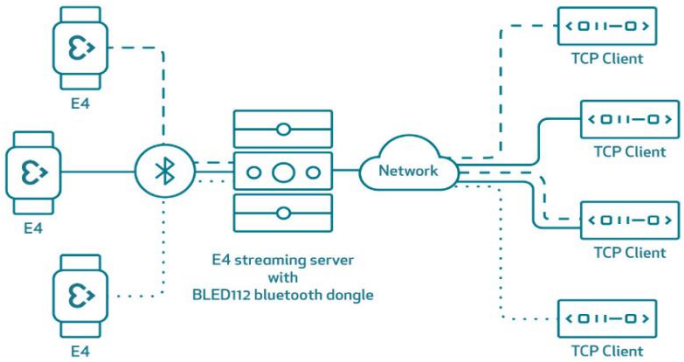

Figura 59: Representação do funcionamento do E4 streaming server (*E4 streaming server*, sem data).

O aplicativo foi desenvolvido sobre a estrutura WPF (*Windows Presentation Foundation*) em linguagem C#, usando o IDE Visual Studio. Este foi desenhado com o intuito de criar uma interface amigável, capaz de produzir gráficos de sinais fisiológicos coletados a partir da E4 e emitir alertas específicos.

Simplificadamente, a estrutura da aplicação pode ser dividida em duas partes: um cliente TCP para estabelecer conexão com o E4 Streaming Server e uma parte de processamento para lidar com os dados e renderizar os gráficos. Relativamente à primeira parte, criou-se um separador que permite ao utilizador

inserir as configurações de IP e porta do servidor por meio de caixas de texto. Após se estabelecer a conexão, é possível iniciar a gravação e leitura do fluxo de dados (Figura 60).

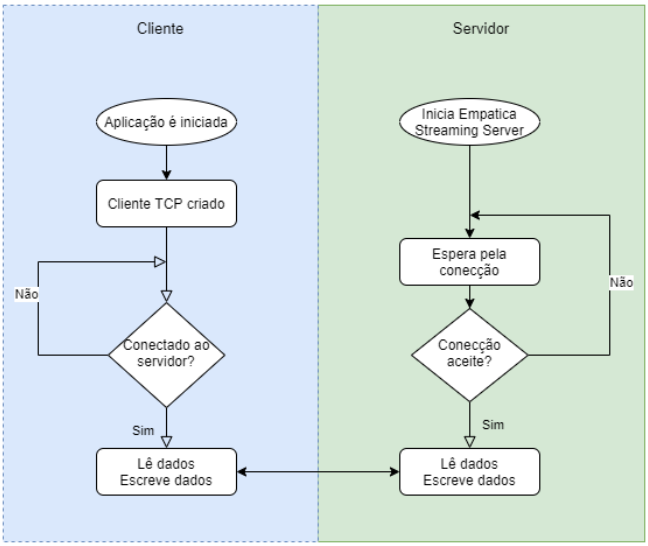

Figura 60: Fluxograma comunicação TCP.

O protocolo da mensagem e a sua especificação encontram-se definidos no site da Empatica (*Data Streaming Packets*, sem data; *Message Protocol*, sem data). O primeiro pacote de dados enviado consiste na solicitação da lista de dispositivos de forma a perceber quais os dispositivos disponíveis. Após a conexão ao dispositivo pretendido, é possível subescrever dados dos sensores o que implicou a necessidade de implementar uma nova *Thread*, de forma a receber continuamente os dados sem causar o bloqueio da interface (UI - *User Interface*).

A etapa de processamento foi desenvolvida tendo em atenção a existência de diversos sensores diferentes. A análise pode ser elaborada considerando que o fluxo de dados segue o formato <STREAM_TYPE> <TIMESTAMP> <DATA> para cada unidade de dados. Por exemplo: E4_Gsr 123345627891.123 3.129 indica que o valor de GSR obtido naquele instante foi de 3,129. Durante a análise, o *timestamp* e o valor sensorial são salvos em arquivos csv., o que impede a perda de dados e permite a realização de análises adicionais futuras. Para visualizar os dados coletados, utilizou-se a versão *trial* da biblioteca SciChart (*SciChart / Charts*, sem data). Os gráficos foram organizados de modo a que o eixo X forneça a data e hora, previamente, convertidas do *timestamp*, e o eixo Y forneça os dados. Os alertas definidos passam por alterar a cor da linha dos gráficos sempre que o valor atual for superior ou inferior aos limites impostos pelo utilizador.

Esta aplicação *Windows* foi posta de parte devido a algumas dificuldades sentidas ao longo da sua implementação, nomeadamente na interação com a biblioteca gráfica.

5. Testes e Resultados

No presente capítulo são apresentados os testes realizados ao sistema de monitorização desenvolvido e os resultados obtidos. Foram, ainda, realizados testes de segurança e atraso de modo a garantir um bom funcionamento do sistema.

5.1. Aplicação Windows

A aplicação WPF inicialmente desenvolvida e mais tarde abandonada encontra-se representada na Figura 61 e está separada em duas áreas: zona gráfica e um menu *Slider* com itens de inserção e seleção.

O menu *Slider* permite que os utilizadores definam configurações de IP e porta, através de *TextBox* editáveis, para possibilitar a conexão ao servidor. Quando a conexão é estabelecida, é possível pesquisar e selecionar os dispositivos disponíveis através da *ComboBox* abaixo. É possível ainda alterar os temas gráficos neste separador, bem como observar informações úteis sobre como trabalhar com o *software*, pressionando o botão de ajuda.

Figura 61: Esquerda: Menu de conexão ao servidor e emparelhamento à E4 do lado; Direita: Visualização Gráfica dos dados enviados pela E4.

A área de gráficos apresenta os dados fornecidos pelo dispositivo E4 ao longo do tempo. Os gráficos são animados e oferecem aos utilizadores opções de *zoom*, deslizamento e *tooltips*. Em cada gráfico, há

ainda um botão *PopUp* que permite aos utilizadores definirem limites superiores e inferiores. Quando os dados excedem os limites, um alerta visual é gerado e a cor da linha é alterada.

Por último, no canto superior direito, há um alerta visual com forma retangular. A partir da sua observação, é possível averiguar quando é que a E4 se encontra conectada (retângulo verde) ou desconectada (retângulo vermelho).

Devido a certos problemas ao longo do desenvolvimento da aplicação, nomeadamente na comunicação por *sockets* e de interação com a biblioteca Scichart, os resultados obtidos não foram totalmente satisfatórios, não sendo possível visualizar graficamente todos os dados de um sinal nem atingir uma *performance* da aplicação de nível razoável.

5.2. Aquisição de Dados

A aquisição dos dados fisiológicos foi realizada através da *wristband* E4, como se encontra referido na secção 4.3. Os testes realizados foram sempre efetuados no modo de *streaming* do dispositivo, sendo este modo visível através da coloração azul do indicador LED.

Figura 62: E4 a funcionar no modo de streaming.

No sistema de monitorização remoto proposto, os dados sensoriais são enviados por BLE para a aplicação Android desenvolvida, onde são apresentados em formato de texto ao mesmo tempo que são enviados para o serviço de base de dados do Firebase. Os dados apresentados são:

- BVP;
- HR;
- EDA;
- Acelerómetro;

- Temperatura da pele;
- Bateria do dispositivo.

O ecrã da aplicação é composto por uma lista desses dados, sendo que a cada um está associado uma *checkbox* que permite ao utilizador selecionar os dados que pretende transmitir para o Firebase. Abaixo da lista podem ser vistos dois botões – um que permite procurar e conectar aos dispositivos e outro que inicia a transmissão. Quando não existe conexão com um dispositivo, a inexistência de dados é representada por um traço horizontal e o botão de transmissão encontra-se desabilitado. Ao pressionar o botão "Procurar Dispositivos" é iniciada a procura e é apresentado um *fragment* com uma *ProgressBar* até que a conexão seja alcançada. Quando a conexão é estabelecida a aplicação começa a receber e a representar os dados e o botão de transmissão fica acessível para que o utilizador possa iniciar a transmissão a qualquer momento. Na Figura 63 e na Figura 64 apresenta-se o ecrã da aplicação nos seus diferentes estados.

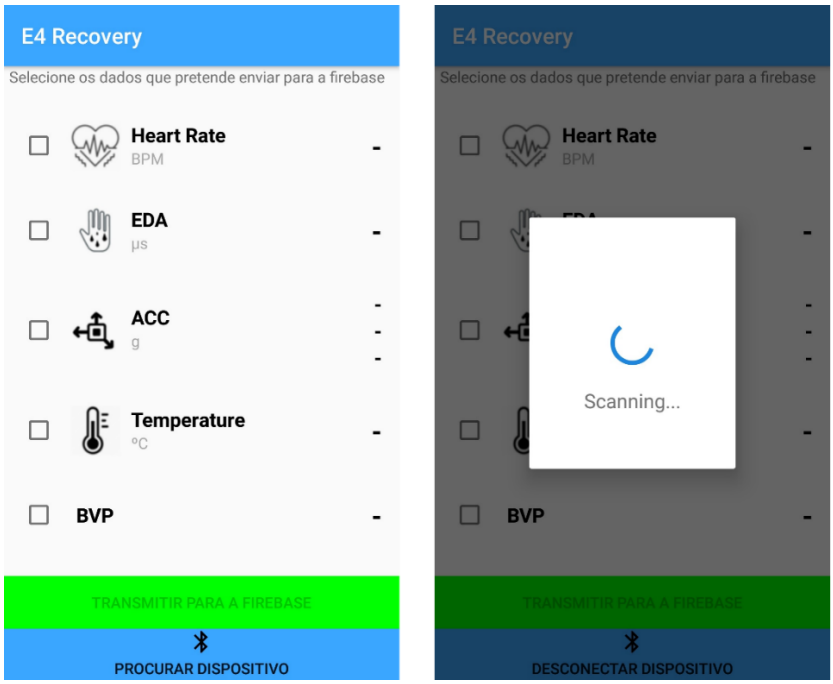

Figura 63: Ecrã da aplicação Android. Esquerda: Antes de iniciar a procura de uma wristband E4. Direita: Depois de iniciar a procura.

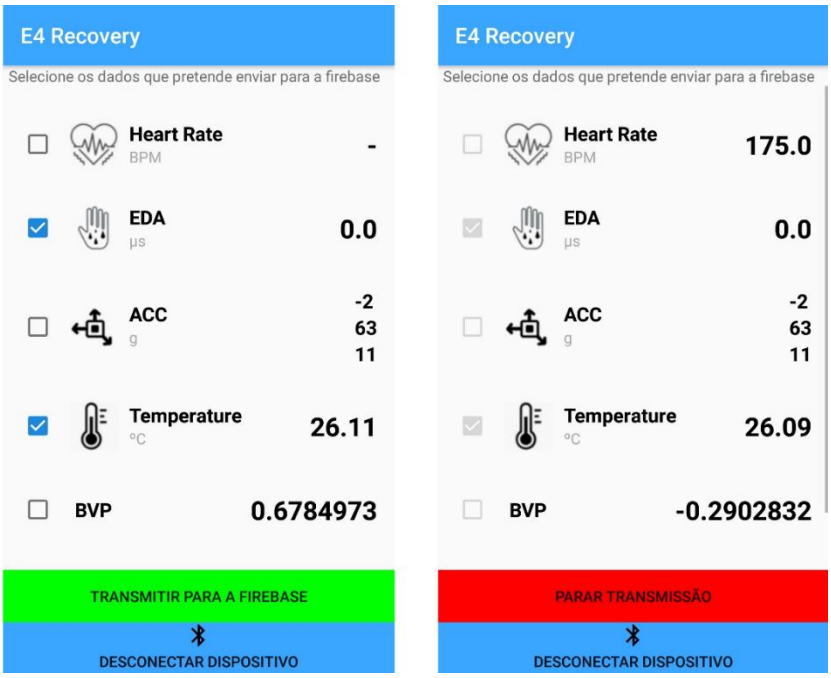

Figura 64: Esquerda: Aplicação encontra-se a receber os dados antes de iniciar a transmissão. Direita: Transmissão iniciada.

5.3. Monitorização através da Aplicação *Web*

Nesta subsecção são apresentados os resultados relacionados com a receção dos dados a partir do Firebase e a representação gráfica dos mesmos. São, ainda, apresentados testes de atraso da comunicação, bem como testes de segurança da aplicação.

5.3.1. Sistema de Registo/*Login*

A página de Registo/*Login* implementada constitui a primeira barreira de acesso aos dados, capaz de impedir que qualquer utilizador não autenticado tenha acesso aos dados presentes na base de dados do Firebase. O sistema é constituído por três formulários, um formulário de *login*, um de registo e outro para fazer *reset* da password (Figura 65, Figura 66 e Figura 67, respetivamente).

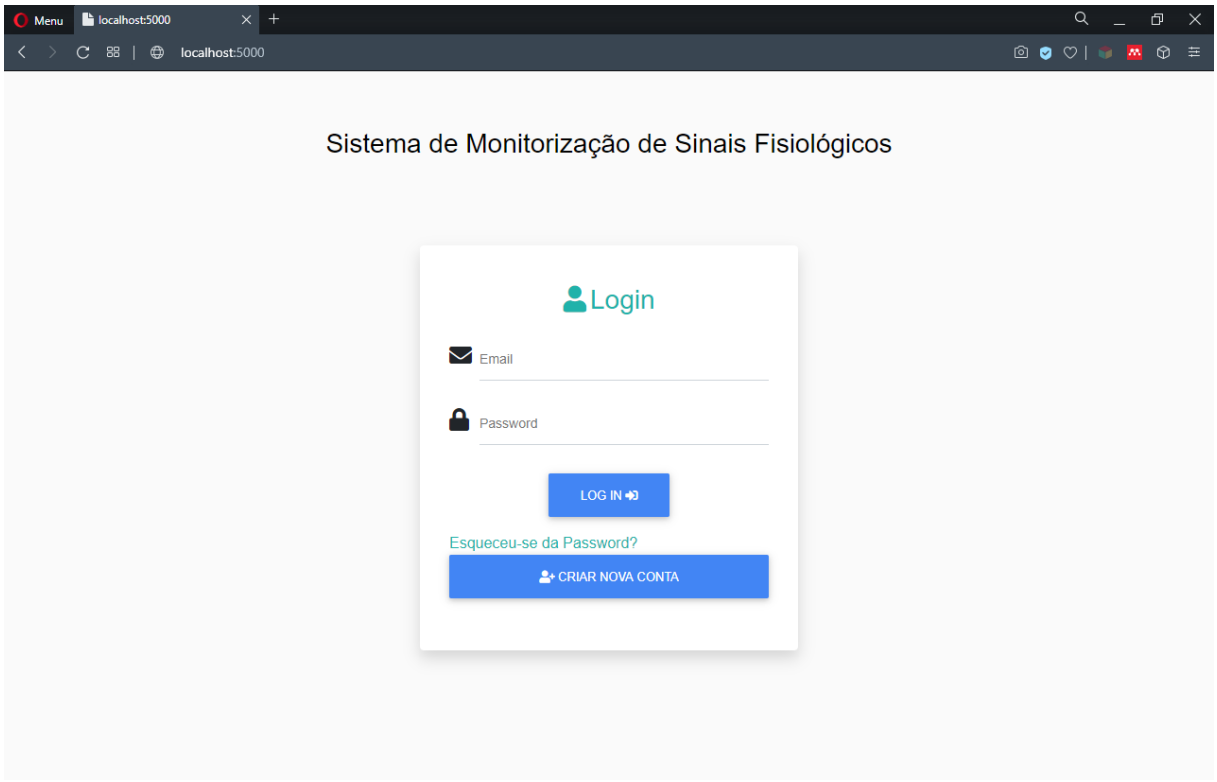

Figura 65: Formulário de login da aplicação Web.

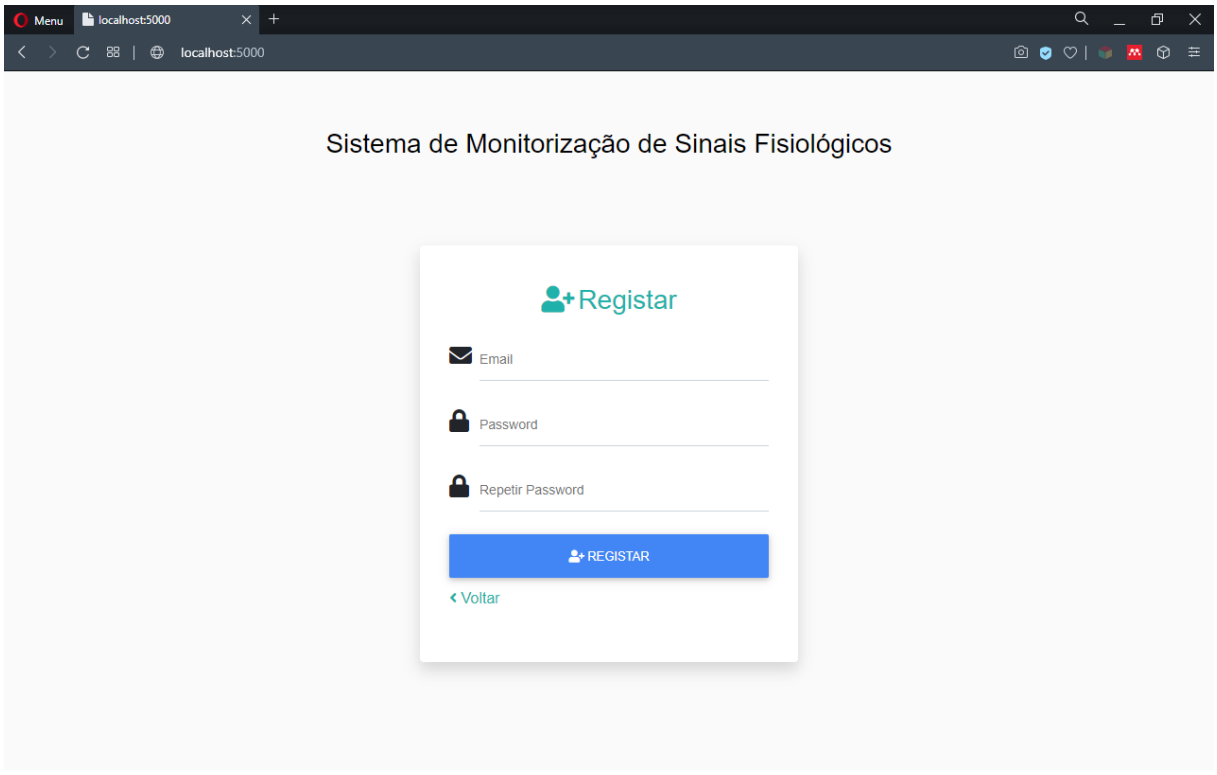

Figura 66: Formulário de registo da aplicação Web.

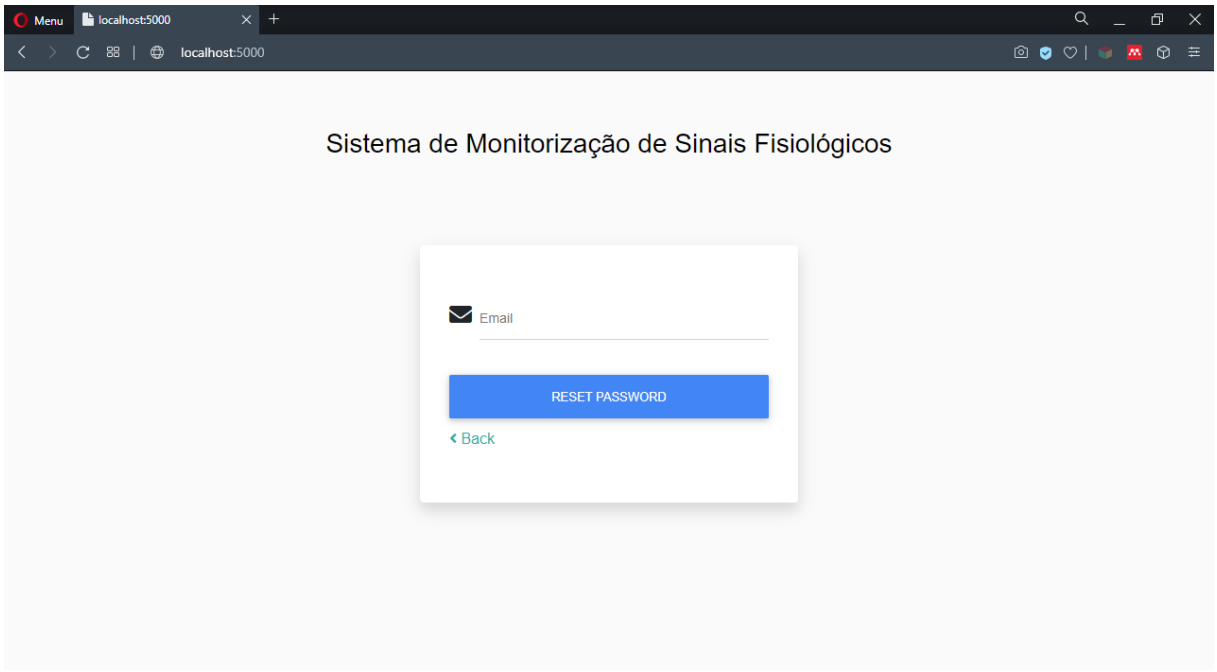

Figura 67: Formulário de reset da password da aplicação Web.

Ao pressionar qualquer um dos botões de submissão dos formulários é apresentado um alerta no caso da ocorrência de algum erro. Os alertas são apresentados através de caixas de alerta customizadas através do uso do plugin sweeralert2 e são constituídos pelo retorno das mensagens de erro geradas pelo sdk de "Authentication" do Firebase. No caso do registo e do *reset* da password também são apresentados alertas de carácter positivo quando o utilizador efetua o seu registo com sucesso e quando o email para redefinir a password é válido. Na Figura 68 encontram-se dois exemplos dos alertas gerados.

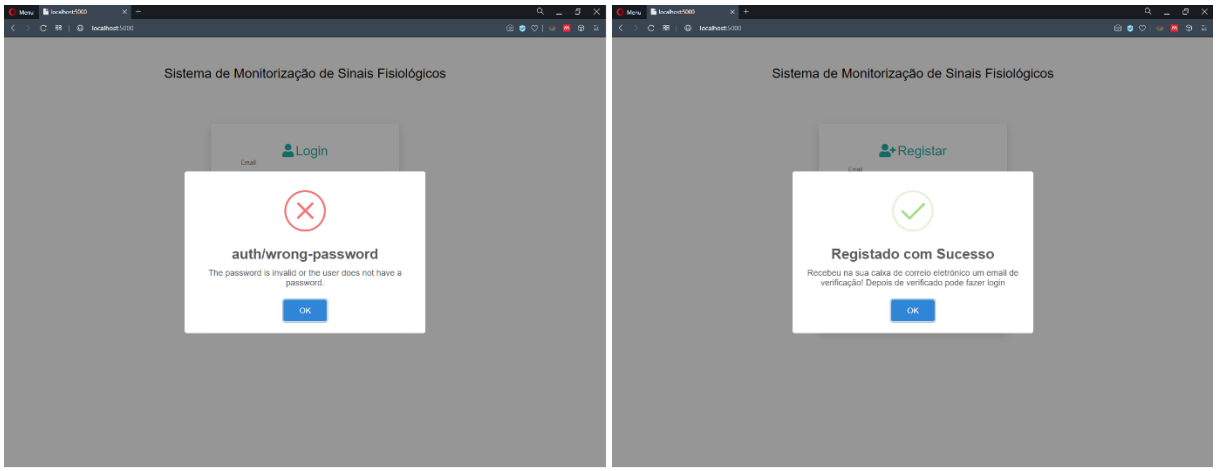

Figura 68: Alertas do sistema registo/login. Esquerda: Alerta negativo por password incorreta. Direita: Alerta positivo de registo realizado com sucesso.

5.3.2. Visualização Gráfica em Tempo Real

Na Figura 69 é apresentada a primeira página visualizada após o utilizador efetuar o *login*. Esta página precede a visualização gráfica em tempo real e permite escolher o dispositivo que está a realizar a transmissão. No caso de nenhum dispositivo estar a transmitir a área de exposição permanece vazia. Para além disso, as imagens dos dispositivos desaparecem e aparecem, de acordo com o estado de transmissão, sem ser necessário atualizar a página devido ao método *on()* do *firebase.database.Reference* como foi explicado na secção 4.6.2.

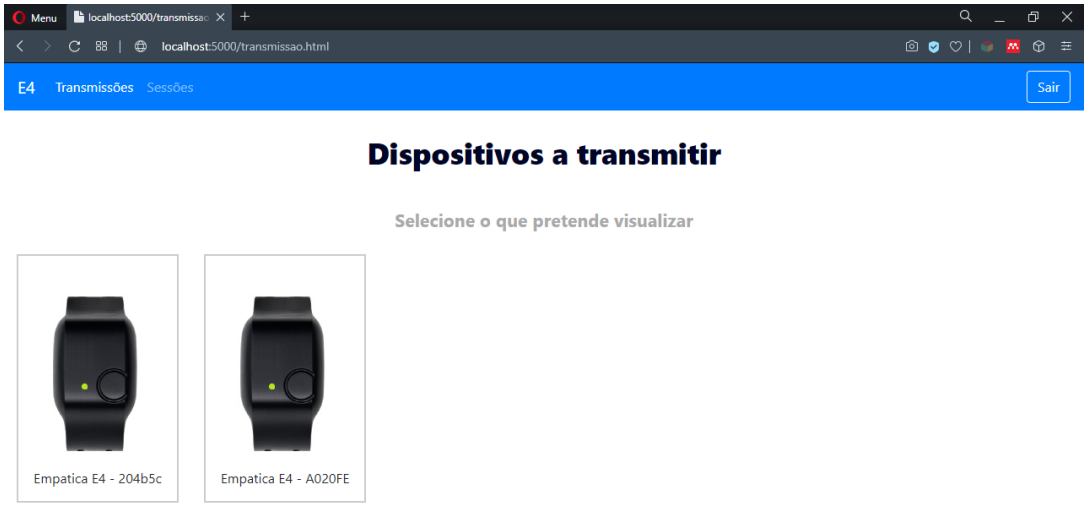

Figura 69: Página Web que permite selecionar o dispositivo a transmitir.

O gráfico em tempo real é aberto noutro separador para que seja possível visualizar mais que um dispositivo se necessário. O gráfico gerado com recurso à biblioteca HighCharts é constituído por um conjunto de eixos alinhados verticalmente, onde se adicionou a cada um uma série que representa um só sinal. Os sinais apresentados são gerados dinamicamente e correspondem aos selecionados na aplicação Android, permitindo assim reduzir os dados armazenados na base dados e o número de leituras efetuadas, resultando na melhoria de possíveis custos de faturação do Firebase. Além do mais, o esforço de computação exigido decresce diretamente com a diminuição de quantidades de dados a ser processada.

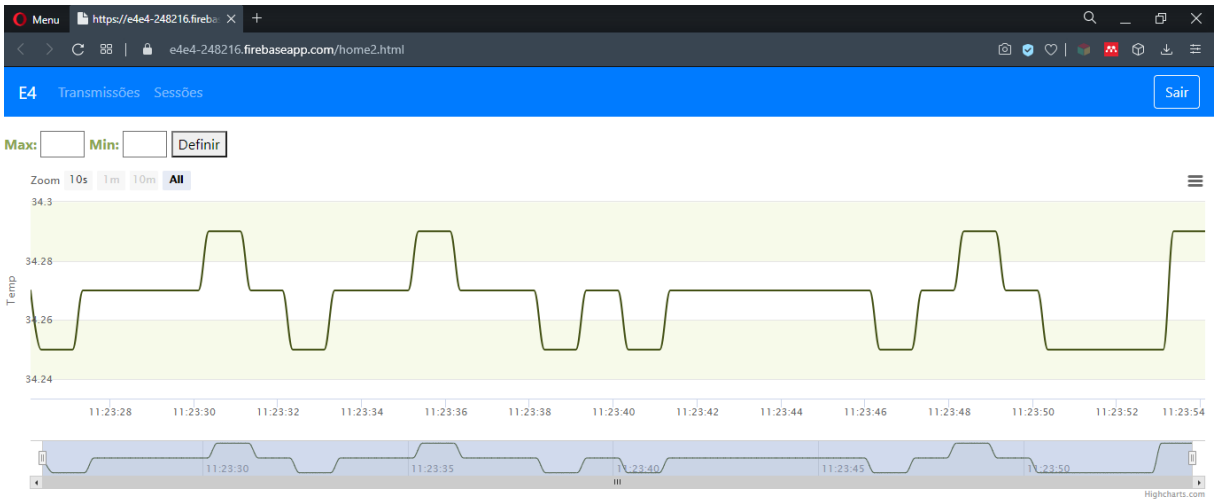

Figura 70:Visualização gráfica de apenas um sinal.

Através de campos de entrada, é possível que o utilizador especifique limites máximos e mínimos às séries. Quando esses limites são ultrapassados, a cor da linha é alterada para vermelho, no caso do valor máximo, e para amarelo, no caso do valor mínimo. Este facto permite ao utilizador ter uma melhor perceção visual ao efetuar a análise do gráfico ou alertá-lo de alguma ocorrência, caso seja usado para esse propósito.

Figura 71: Visualização gráfica em tempo real de 4 sinais. Nas duas primeiras séries foram definidos limites.

5.3.2.1. Medição de atraso da comunicação

Os testes realizados, neste subcapítulo, avaliam o atraso das comunicações desde o momento que se escreve na base de dados, a partir da aplicação Android, até ao momento que se efetua a leitura na aplicação *Web*. Estes testes foram realizados com recurso a um único aparelho - *smartphone* Xiaomi Redmi Note 4x com suporte 4G - de modo a evitar possíveis erros induzidos pelo uso de diferentes dispositivos (E4, *smartphone* e Computador) por não compartilharem o mesmo relógio. Posto isto, foi também utilizado o *timestamp* obtido no momento, em vez do *timestamp* enviado pela E4. O cenário de testes pode ser visualizado na Figura 72.

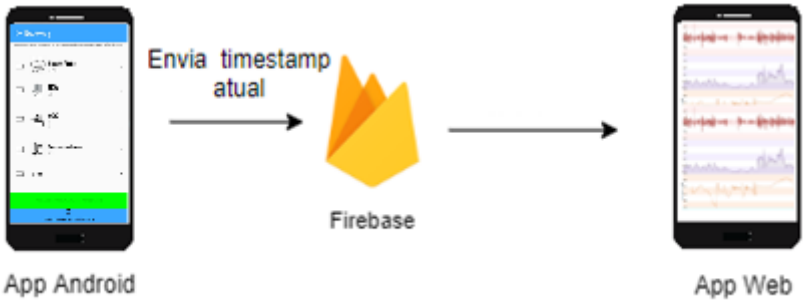

Figura 72: Cenário de teste para o cálculo do atraso da comunicação.

Na aplicação Android, recorreu-se à função *didReceiveGSR(float gsr, double timestamp)* durante a realização dos testes. Esta função é invocada cada vez que um novo valor de GSR fica disponível e é nela que se efetua o envio do *timestamp* atual, obtido através do método *System.currentTimeMillis()*, para a base de dados. Do lado da aplicação *Web*, quando um valor GSR é lido através da função de *callback* do Firebase, é calculado o atraso dado pela diferença entre o valor do *timestamp* atual, obtido pelo método Date().getTime(), e o valor do *timestamp* lido. Resumidamente, acompanha-se o seguinte algoritmo:

Na aplicação android

- Timestamp_inical = timestamp atual;
- Enviar Timestamp_inical para a base de dados;

Na aplicação *Web*:

- Ler Timestamp_inicial da base de dados;
- Timestamp_final = timestamp atual;
- Atraso = Timestamp_final – Timestamp_inicial

Com recurso às ferramentas de desenvolvedor do Chrome (*Get Started with Remote Debugging Android Devices | Chrome DevTools*, sem data), foi possível efetuar a depuração remota da aplicação *Web* no *smartphone* e extrair os tempos de atraso que foram guardados ao longo do tempo num array. Desta forma, calculou-se os atrasos de 1000 amostras tanto para ligação à Internet via *Wi-Fi*, como para dados móveis. Os resultados obtidos para os tempos de atraso, em milissegundos, encontram-se apresentados na Tabela 6.

Tabela 6: Tempos de atrasos obtidos para dois tipos de ligação à Internet.

Ligação à Internet	***Tempo Médio***	***Tempo Máximo***	***Tempo Mínimo***
Wi-Fi	308,252	3968	154
Dados móveis (4G)	394,587	3044	178

Pela interpretação dos resultados, observou-se que foram obtidos melhores resultados perante a aplicação com ligação à Internet por *Wi-Fi*. Na Figura 73 e 74 estão representados, através de histogramas, os tempos de atrasos obtidos para as 1000 amostras. Verificou-se que na ligação por *Wi-Fi* cerca de 90% das amostras registaram um atraso compreendido entre 154 e 304 milissegundos, enquanto que na ligação por dados móveis cerca de 79% das amostras registaram um atraso compreendido entre 178 e 318 milissegundos.

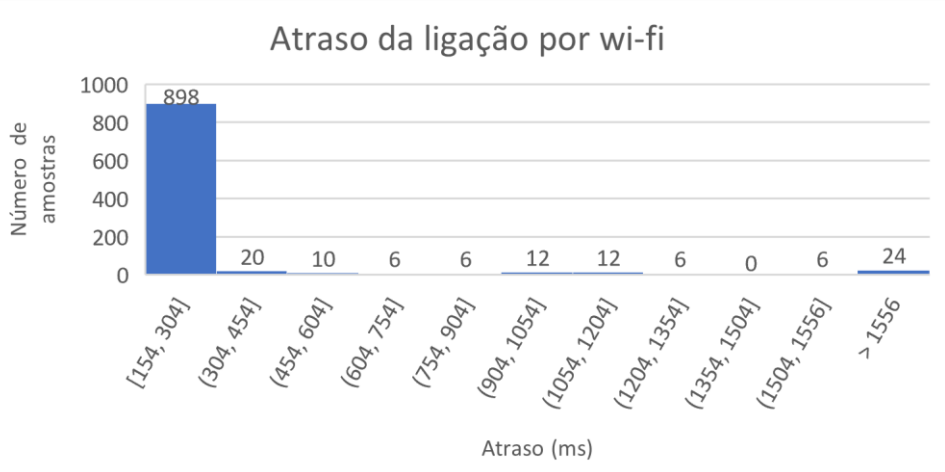

Figura 73: Histograma dos valores de atraso com ligação à Internet por Wi-Fi.

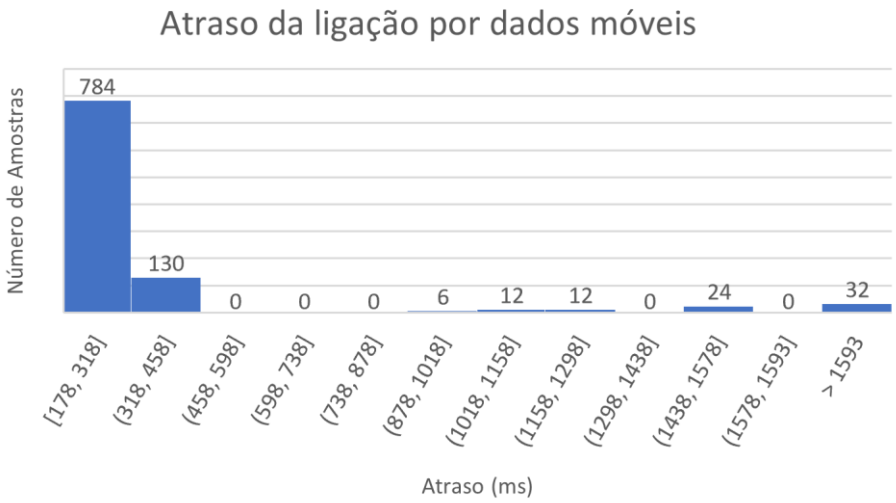

Figura 74:Histograma dos valores de atraso com ligação à Internet por 4G.

Por último, foi possível verificar a existência de valores de atraso mais elevados que a média, nas duas ligações, que podem ter tido origem de alguma latência da rede ou do processo de renderização dos gráficos. Pode-se, ainda, verificar um atraso mais acentuado caso se verifique a perda de ligação, uma vez que o Firebase garante que não ocorre perda de dados e a sua recuperação pela sua inserção na base de dados quando a ligação for retomada. Os resultados obtidos aproximam-se dos resultados obtidos pelos autores da referência (*Get Started with Remote Debugging Android Devices | Chrome DevTools*, sem data).

5.3.3. Visualização das Sessões Realizadas

As sessões previamente realizadas podem ser consultadas selecionando a opção "Sessões" situada na barra de navegação no topo da aplicação. A página que se encontrar ativa permanece destacada com uma cor mais viva e o utilizador pode optar por encerrar a sessão através do botão "Sair".

As sessões realizadas são apresentadas numa tabela onde cada sessão é identificada através de um número, do nome da E4 e da data de início de transmissão. No topo da tabela, mais especificamente nos cabeçalhos "SESSÃO" e "Dispositivo", é possível ordenar a tabela por ordem decrescente/crescente e por dispositivos respetivamente, através da seleção do critério pretendido.

Através do botão "Visualizar" é possível aceder ao gráfico da sessão escolhida e com o botão "Apagar" o utilizador tem a opção de remover os dados de uma sessão da base de dados. Para garantir que uma sessão não é removida por engano, uma janela de confirmação é apresentada de modo a que

o utilizador confirme a sua intenção. Na Figura 75 apresenta-se a lista de sessões e a janela de confirmação quando se tenta apagar.

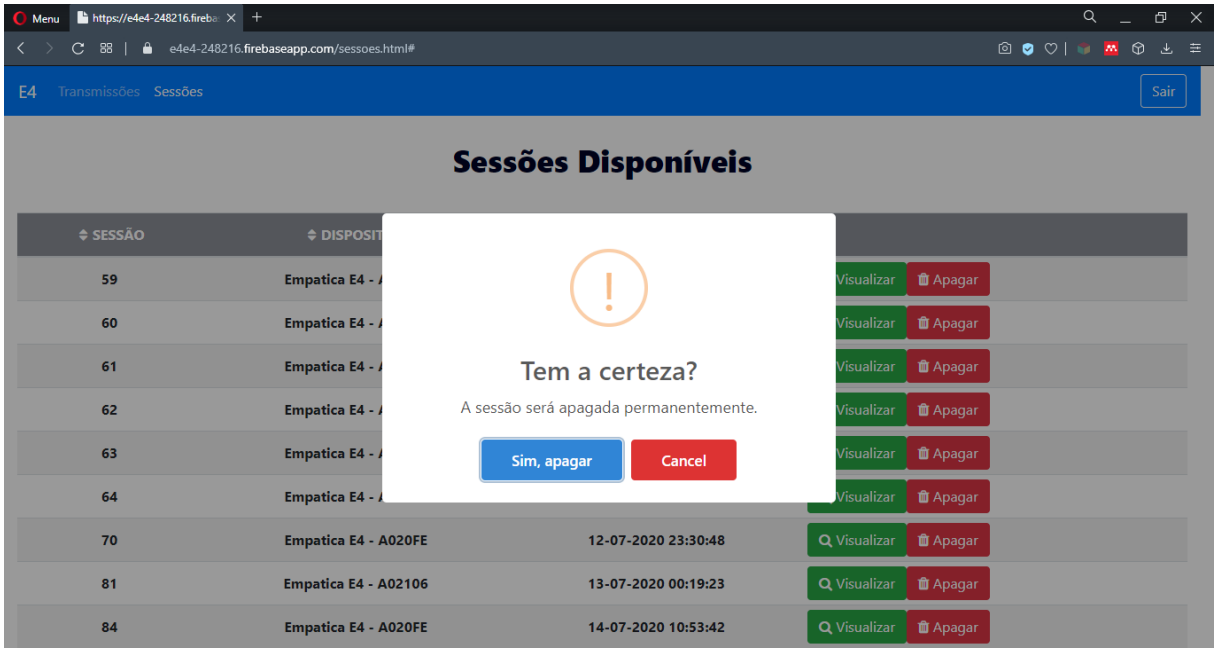

Figura 75: Lista de sessões realizadas.

O gráfico que é apresentado quando se visualiza uma sessão é idêntico ao apresentado em tempo real, contudo, neste caso, tem a particularidade de se poder apresentar todos os dados guardados, exibindo assim todo o histórico dessa sessão. Na Figura 76, observa-se um exemplo de um gráfico de uma sessão. No canto superior direito é possível distinguir a data de início e fim da sessão e um menu que permite imprimir, guardar em diferentes formatos de imagem, entre outros, o gráfico criado. Na parte inferior do gráfico, por baixo do eixo temporal, existe uma barra de navegação que exibe a visão geral da primeira série do gráfico. Poder-se-ia ter optado pela visualização de todas as séries simultaneamente, no entanto optou-se por visualizar apenas uma devido à confusão que se iria criar. A barra de navegação permite aumentar e diminuir o *zoom* e pode ser deslocada para ver apenas um conjunto dos dados selecionados.

Figura 76: Gráfico de uma sessão realizada.

5.3.4. Segurança da Base de Dados

As regras de segurança do Firebase constituem a segunda barreira de acesso e a mais importante, relativamente, à segurança da base de dados. As regras são definidas fora da aplicação, removendo assim a responsabilidade do cliente em garantir a segurança dos dados. Desde que estas se encontram bem definidas, o acesso de utilizadores mal-intencionados que tentem roubar ou adulterar os dados encontra-se impedido.

Inicialmente, no momento de criação do projeto do Firebase as regras de leitura e escrita foram definidas ambas a "true", de modo a que fosse possível executar qualquer escrita e leitura. De seguida, realizou-se a escrita da *string* "dados disponíveis" no caminho "dados" que foi utilizado para os testes que a seguir se apresentam.

No primeiro teste realizado com as regras a "true", mesmo com o sistema de *login* implementado foi possível aceder aos dados sem ser necessário efetuar o *login*. Na Figura 77 é possível ver a indicação

"null" quando se imprimiu o utilizador, realçando que o *login* não foi efetuado, contudo a resposta à leitura foi concluída com a *promise* a retornar "fulfilled" e com a apresentação da string guardada no caminho "dados". Estas regras são as mais vulneráveis que podem ser aplicadas, visto que um utilizador não registado conseguiu aceder aos dados e apenas foram utilizadas na fase inicial do desenvolvimento do projeto.

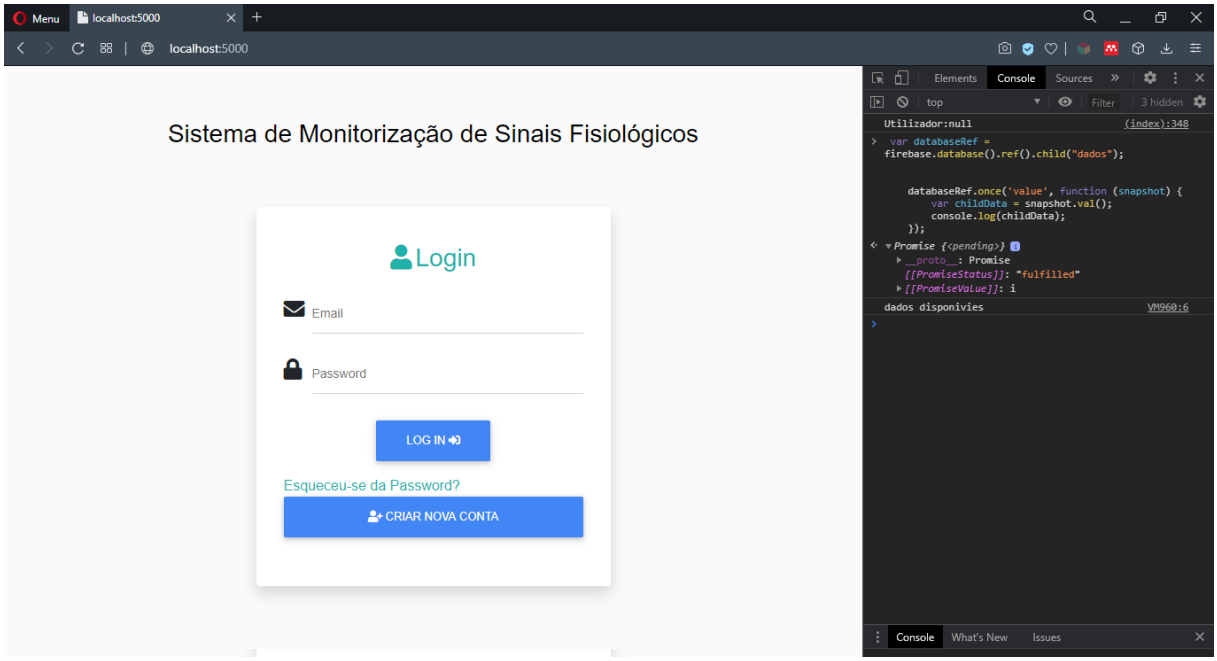

Figura 77: Tentativa de obtenção de dados sem efetuar o login com as regras definidas a true.

No segundo teste, já com as regras finais definidas (apresentadas no final da secção 4.4.1) realizou-se o *login* através de um *email* validado e tentou-se obter a leitura no campo "dados" do mesmo modo que no primeiro teste. Como é possível verificar pela observação da Figura 78, não foi possível concretizar esta operação dado que o *email*, apesar de estar validado, não termina em alunos.uminho.pt como definido nas regras previamente estabelecidas. É ainda possível observar que a lista de sessões não é apresentada de acordo com o que seria espetável.

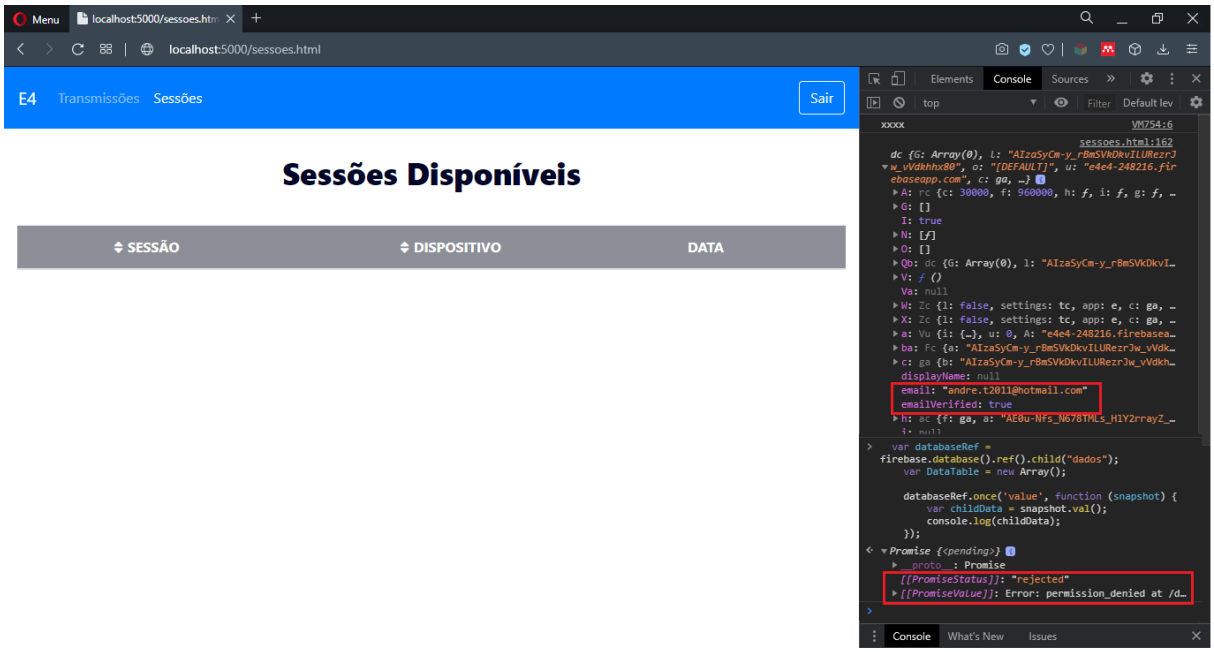

Figura 78: Tentativa de obtenção dos dados através de um utilizador com email validado, mas sem corresponde ao tipo de email permitido e com as regras finais definidas

Por fim, o último teste realizado encontra-se apresentado na Figura 79 e demonstra que com as regras finais definidas, o utilizador só terá acesso aos dados caso essas regras sejam cumpridas. Neste caso, utilizou-se o *email* institucional para efetuar o *login* e pode ver-se que a leitura no caminho "dados" foi realizada com sucesso com a *string* "dados disponíveis" a ser apresentada. A lista de sessões desta vez encontra-se preenchida, o que era espetável, dado que os requisitos de acesso foram cumpridos.

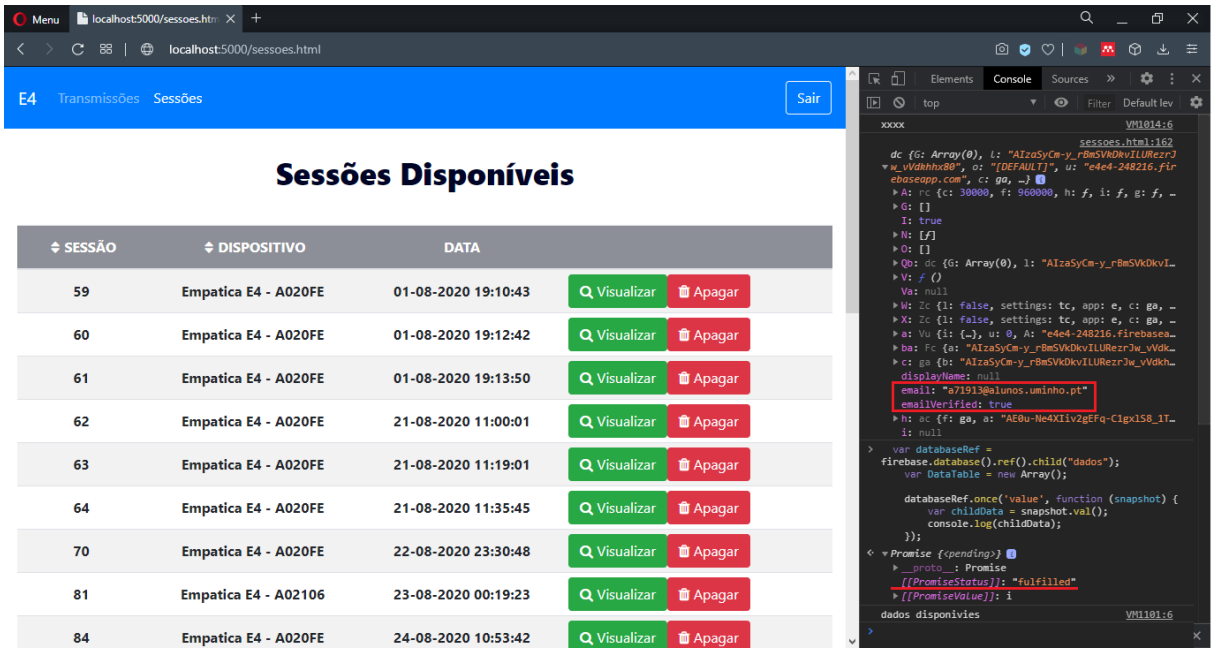

Figura 79: Tentativa de acesso aos dados com o email do utilizador a cumprir todos os requisitos.

6. Conclusões e Trabalho Futuro

No âmbito desta dissertação desenvolveu-se o protótipo de utilização laboratorial de um sistema de monitorização remota *online* de sinais fisiológicos, que no futuro poderá vir a auxiliar o tratamento e o acompanhamento do estado de saúde de utilizadores com transtorno mental.

Inicialmente, realizou-se um estudo para se determinar quais os sinais fisiológicos de interesse e os equipamentos/dispositivos que podem ser utilizados na sua medição. Também se determinou a arquitetura de sistema que mais se adequa às necessidades e objetivos pretendidos.

Escolheu-se a *wristband* E4 da empresa Empatica para a aquisição de diversos sinais: EDA; temperatura; HR e movimento. Trata-se de um dispositivo de reduzidas dimensões, robusto, portátil e sem fios elétricos externos ligados a sensores ou elétrodos, que proporciona ao utilizador uma grande liberdade de movimentos e autonomia, dispensando-o da necessidade de permanecer parado num determinado local para se proceder à monitorização. Além disso, é um dispositivo discreto e que, por não ter um ecrã, não constitui um fator de distração para o utilizador nem lhe permite uma possível desconfigurarão do aparelho. Os dados recolhidos pela E4 são transmitidos para um *smartphone* por BLE e através de uma aplicação Android. De seguida, são enviados por *Wi-Fi* ou rede móvel 4G para a *Cloud* Firebase. Nesta arquitetura, o *smartphone* representa um papel fundamental dado que, para além de fornecer uma interface gráfica e permitir ao utilizador escolher os dados que pretende transmitir através da aplicação Android desenvolvida, também atua como *gateway*. Também se desenvolveu uma aplicação *Web* responsiva que permite visualizar graficamente os dados enviados em tempo real para a *Cloud* ou consultar os dados de sessões anteriores.

Por último, testou-se o sistema desenvolvido em laboratório, de forma a avaliar todo o processo de monitorização desde a aquisição dos dados até à visualização final na interface desenvolvida. Realizaram-se testes para avaliar o atraso das comunicações e comparam-se os resultados entre a comunicação por *Wi-Fi* e 4G. Para finalizar, testou-se a segurança e *performance* do sistema. Foram obtidos atrasos médios de cerca 300 ms, com a comunicação por *Wi-Fi* a apresentar um atraso inferior ao obtido na comunicação por 4G. Não se verificaram perdas de dados ao longo do processo de monitorização. Na aplicação *Web* os dados apenas se encontraram acessíveis com autenticação através de endereço de email institucional, conforme definido nas regras finais de funcionamento da base de dados do Firebase.

Numa fase inicial, elaborou-se uma aplicação para Windows, que devido às dificuldades sentidas com o protocolo de comunicação e com a biblioteca utilizada para desenhar os gráficos, não apresentou o resultado esperado. A abordagem remota posteriormente adotada apresenta uma maior versatilidade e vantagens em relação à essa primeira aplicação, já que não implica o uso obrigatório de um computador para interagir com o sistema e a recolha de dados pode ser efetuada a partir de qualquer lugar.

Apesar de o sistema desenvolvido cumprir plenamente objetivo pretendido, existe sempre a oportunidade de efetuar alguns aperfeiçoamentos. Como possíveis trabalhos futuros sugerem-se os seguintes:

- Melhoramento das interfaces gráficas das aplicações *Web* e Android à medida que tal venha a revelar-se necessário;
- Notificação prévia aos cuidadores quando um dispositivo inicia uma transmissão ou quando um determinado valor de um sinal, por exemplo HR, é ultrapassado;
- Estudo da necessidade de implementação de algoritmos de pré-processamento para filtrar eventuais ruídos que afetem os sinais adquiridos em contexto real de utilização.

Referências

Abdullah, S., & Choudhury, T. (2018). Sensing Technologies for Monitoring Serious Mental Illnesses. *IEEE Multimedia*, *25*(1), 61–75. https://doi.org/10.1109/MMUL.2018.011921236

Agelink, M. W., Boz, C., Ullrich, H., & Andrich, J. (2002). Relationship between major depression and heart rate variability. Clinical consequences and implications for antidepressive treatment. *Psychiatry Research*, *113*(1–2), 139–149. https://doi.org/10.1016/S0165-1781(02)00225-1

Bluetooth low energy overview | Android Developers. (sem data). Obtido 22 de Agosto de 2020, de https://developer.android.com/guide/topics/connectivity/bluetooth-le

Bootstrap. (sem data). Obtido 27 de Agosto de 2020, de https://getbootstrap.com/

Boucsein, W. (1992). Electrodermal Activity. Em *Electrodermal Activity*. https://doi.org/10.1007/978-1-4757-5093-5

Braithwaite, J., Watson, D., Robert, J., & Mickey, R. (2013). A Guide for Analysing Electrodermal Activity (EDA) & Skin Conductance Responses (SCRs) for Psychological Experiments. Em *....* https://doi.org/10.1017.S0142716405050034

Calton, B., Abedini, N., & Fratkin, M. (2020). Telemedicine in the Time of Coronavirus. *Journal of Pain and Symptom Management*, *60*(1), e12–e14. https://doi.org/10.1016/j.jpainsymman.2020.03.019

Can Remote Patient Monitoring Assist Mental Health Services | Trapollo. (sem data). Obtido 10 de Abril de 2020, de https://www.trapollo.com/telehealth-and-remote-patient-monitoring-assist-mental-health-services/

Chen, M., Gonzalez, S., Vasilakos, A., Huasong, ·, Victor, C. ·, Leung, C. M., Chen, M., Gonzalez, · S, Cao, · H, Leung, V. C. M., & Vasilakos, A. (2011). Body Area Networks: A Survey. *Mobile Netw Appl*, *16*, 171–193. https://doi.org/10.1007/s11036-010-0260-8

Chen, Y. T., Hung, I. C., Huang, M. W., Hou, C. J., & Cheng, K. S. (2011). Physiological signal analysis for patients with depression. *Proceedings - 2011 4th International Conference on Biomedical Engineering and Informatics, BMEI 2011*, *2*, 805–808. https://doi.org/10.1109/BMEI.2011.6098461

Data Streaming Packets. (sem data). Obtido 22 de Setembro de 2020, de https://developer.empatica.com/windows-streaming-server-data.html

Dian, F. J., Yousefi, A., & Lim, S. (2019). A practical study on Bluetooth Low Energy (BLE) throughput. *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference, IEMCON 2018*, 768–771. https://doi.org/10.1109/IEMCON.2018.8614763

Direção-Geral da Saúde. (sem data). Obtido 15 de Julho de 2020, de https://www.dgs.pt/paginas-de-sistema/saude-de-a-a-z/programa-nacional-para-a-saude-mental/perguntas-e-respostas.aspx

E4-SP069-B-20150001-Manual | Empatica. (sem data). Obtido 14 de Fevereiro de 2020, de https://empatica.app.box.com/v/E4-User-Manual

E4 streaming server. (sem data). Obtido 22 de Setembro de 2020, de https://developer.empatica.com/windows-streaming-server.html

E4 wristband | Empatica. (2020). Empatica. https://www.empatica.com/en-eu/research/e4/

Empatica. (2015). *What should I know to use EDA data in my experiment?* 1–5. https://support.empatica.com/hc/en-us/articles/203621955-What-should-I-know-to-use-EDA-data-in-my-experiment-

Firebase. (sem data). Obtido 15 de Julho de 2020, de https://firebase.google.com/

Fotouhi, H., Čaušević, A., Lundqvist, K., & Björkman, M. (2016). Communication and Security in Health Monitoring Systems - A Review. *Proceedings - International Computer Software and Applications Conference*, *1*, 545–554. https://doi.org/10.1109/COMPSAC.2016.8

Fragments | Android Developers. (sem data). Obtido 11 de Junho de 2020, de https://developer.android.com/guide/fragments

Get Started with Remote Debugging Android Devices | Chrome DevTools. (sem data). Obtido 13 de Agosto de 2020, de https://developers.google.com/web/tools/chrome-devtools/remote-debugging

Guardiola, S., Girbés, V., Armesto, L., Dols, J., & Tornero, J. (2017). *Physiological Signal Analysis for Driver Stress Detection*. *October*.

Hamim, M., Paul, S., Hoque, S. I., Rahman, M. N., & Baqee, I. Al. (2019). IoT Based remote health monitoring system for patients and elderly people. *1st International Conference on Robotics, Electrical and Signal Processing Techniques, ICREST 2019*, 533–538. https://doi.org/10.1109/ICREST.2019.8644514

Highcharts. (sem data). *Interactive JavaScript charts for your webpage | Highcharts*. Obtido 17 de Julho de 2020, de https://www.highcharts.com/

Holzinger, A., Bruschi, M., & Eder, W. (2013). On interactive data visualization of physiological low-cost-sensor data with focus on mental stress. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *8127 LNCS*, 469–480. https://doi.org/10.1007/978-3-642-40511-2_34

Introduction to HTML | w3schools.com. (sem data). Obtido 29 de Dezembro de 2020, de https://www.w3schools.com/html/html_intro.asp

Khan, R. A., & Pathan, A.-S. K. (2018). The state-of-the-art wireless body area sensor networks: A survey. *International Journal of Distributed Sensor Networks*, *14*(4), 155014771876899. https://doi.org/10.1177/1550147718768994

Khodayari-Rostamabad, A., Reilly, J. P., Hasey, G., Debruin, H., & MacCrimmon, D. (2010). Diagnosis of psychiatric disorders using EEG data and employing a statistical decision model. *2010 Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBC'10*, 4006–4009. https://doi.org/10.1109/IEMBS.2010.5627998

Khodayari-Rostamabad, A., Reilly, J. P., Hasey, G. M., de Bruin, H., & MacCrimmon, D. J. (2013). A machine learning approach using EEG data to predict response to SSRI treatment for major depressive disorder. *Clinical Neurophysiology*, *124*(10), 1975–1985. https://doi.org/10.1016/j.clinph.2013.04.010

Kim, A. Y., Jang, E. H., Kim, S., Choi, K. W., Jeon, H. J., Yu, H. Y., & Byun, S. (2018). Automatic detection of major depressive disorder using electrodermal activity. *Scientific Reports*, *8*(1). https://doi.org/10.1038/s41598-018-35147-3

Kim, E. Y., Lee, M. Y., Kim, S. H., Ha, K., Kim, K. P., & Ahn, Y. M. (2017). Diagnosis of major depressive disorder by combining multimodal information from heart rate dynamics and serum proteomics using machine-learning algorithm. *Progress in Neuro-Psychopharmacology and Biological Psychiatry*, *76*, 65–71. https://doi.org/10.1016/j.pnpbp.2017.02.014

Mavrogiorgou, A., Kiourtis, A., Perakis, K., Pitsios, S., & Kyriazis, D. (2019). IoT in Healthcare: Achieving Interoperability of High-Quality Data Acquired by IoT Medical Devices. *Sensors*, *19*(9), 1978. https://doi.org/10.3390/s19091978

McCarthy, C., Pradhan, N., Redpath, C., & Adler, A. (2016). Validation of the Empatica E4 wristband. *2016 IEEE EMBS International Student Conference: Expanding the Boundaries of Biomedical Engineering and Healthcare, ISC 2016 - Proceedings*, 1–4. https://doi.org/10.1109/EMBSISC.2016.7508621

Mell, P., & Grance, T. (2012). The NIST definition of cloud computing: Recommendations of the National Institute of Standards and Technology. Em *Public Cloud Computing: Security and Privacy Guidelines* (pp. 97–101). https://doi.org/10.6028/NIST.SP.800-145

Message Protocol. (sem data). Obtido 22 de Setembro de 2020, de https://developer.empatica.com/windows-streaming-server-commands.html

Mück, J. E., Ünal, B., Butt, H., & Yetisen, A. K. (2019). Market and Patent Analyses of Wearables in Medicine. Em *Trends in Biotechnology* (Vol. 37, Número 6, pp. 563–566). Elsevier Ltd. https://doi.org/10.1016/j.tibtech.2019.02.001

Nahshoni, E., Aravot, D., Aizenberg, D., Sigler, M., Zalsman, G., Strasberg, B., Imbar, S., Adler, E., & Weizman, A. (2004). Heart Rate Variability in Patients with Major Depression. *Psychosomatics*, *45*(2), 129–134. https://doi.org/10.1176/appi.psy.45.2.129

Nguyen, H. H., Mirza, F., Naeem, M. A., & Nguyen, M. (2017). A review on IoT healthcare monitoring applications and a vision for transforming sensor data into real-time clinical feedback. *Proceedings of the 2017 IEEE 21st International Conference on Computer Supported Cooperative Work in Design, CSCWD 2017*, 257–262. https://doi.org/10.1109/CSCWD.2017.8066704

Node.js. (sem data). Obtido 20 de Março de 2020, de https://nodejs.org/en/

O que é a computação na cloud? | Microsoft Azure. (sem data). Obtido 29 de Outubro de 2020, de https://azure.microsoft.com/pt-pt/overview/what-is-cloud-computing/#cloud-computing-models

Olatinwo, D. D., Abu-Mahfouz, A., & Hancke, G. (2019). A survey on LPWAN technologies in WBAN for remote health-care monitoring. Em *Sensors (Switzerland)* (Vol. 19, Número 23). MDPI AG. https://doi.org/10.3390/s19235268

Osipov, M., Behzadi, Y., Kane, J. M., Petrides, G., & Clifford, G. D. (2015). Objective identification and analysis of physiological and behavioral signs of schizophrenia. *Journal of Mental Health*, *24*(5), 276–282. https://doi.org/10.3109/09638237.2015.1019048

Pantelopoulos, A., & Bourbakis, N. G. (2010). A survey on wearable sensor-based systems for health monitoring and prognosis. Em *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews* (Vol. 40, Número 1, pp. 1–12). https://doi.org/10.1109/TSMCC.2009.2032660

Plácido Da Silva, H., Guerreiro, J., Lourenco, A., Fred, A. L. N., Lourenço, A., Fred, A., & Martins, R. (2014). *BITalino: A Novel Hardware Framework for Physiological Computing Clustering View project PersonAAL(Personalized web application to improve quality of life and remote care for older adults) View project BITalino: A Novel Hardware Framework for Physiologica*. http://openeeg.sourceforge.net/doc/

Poh, M. Z., Loddenkemper, T., Reinsberger, C., Swenson, N. C., Goyal, S., Sabtala, M. C., Madsen, J. R., & Picard, R. W. (2012). Convulsive seizure detection using a wrist-worn electrodermal activity and accelerometry biosensor. *Epilepsia*, *53*(5), e93–e97. https://doi.org/10.1111/j.1528-1167.2012.03444.x

Quintana, D. S., Westlye, L. T., Kaufmann, T., Rustan, O. G., Brandt, C. L., Haatveit, B., Steen, N. E., & Andreassen, O. A. (2016). Reduced heart rate variability in schizophrenia and bipolar disorder compared to healthy controls. *Acta Psychiatrica Scandinavica*, *133*(1), 44–52. https://doi.org/10.1111/acps.12498

Raza, U., Kulkarni, P., & Sooriyabandara, M. (2017). Low Power Wide Area Networks: An Overview. *IEEE Communications Surveys and Tutorials*, *19*(2), 855–873. https://doi.org/10.1109/COMST.2017.2652320

Ritchey, K. C., Foy, A., McArdel, E., & Gruenewald, D. A. (2020). Reinventing Palliative Care Delivery in the Era of COVID-19: How Telemedicine Can Support End of Life Care. *American Journal of Hospice and Palliative Medicine*, *37*(11), 992–997. https://doi.org/10.1177/1049909120948235

Sarchiapone, M., Gramaglia, C., Iosue, M., Carli, V., Mandelli, L., Serretti, A., Marangon, D., & Zeppegno, P. (2018). The association between electrodermal activity (EDA), depression and suicidal behaviour: A systematic review and narrative synthesis. *BMC Psychiatry*, *18*(1). https://doi.org/10.1186/s12888-017-1551-4

SciChart | Charts. (sem data). Obtido 22 de Setembro de 2020, de https://www.scichart.com/

Services overview | Android Developers. (sem data). Obtido 29 de Dezembro de 2020, de https://developer.android.com/guide/components/services

Sharma, T., Ritesh, K., Chauhan, N., & Agarwal, S. (2016). Analogous study of 4G and 5G. *Proceedings of the 10th INDIACom; 2016 3rd International Conference on Computing for Sustainable Global Development, INDIACom 2016*, 2137–2140.

Shimmer | Wearable Sensor Technology. (sem data). Obtido 29 de Setembro de 2020, de http://www.shimmersensing.com/

Uday, S., Jyotsna, C., & Amudha, J. (2018). Detection of Stress using Wearable Sensors in IoT Platform. *Proceedings of the International Conference on Inventive Communication and Computational Technologies, ICICCT 2018*, *Icicct*, 492–498. https://doi.org/10.1109/ICICCT.2018.8473010

Udovičić, G., Derek, J., Russo, M., & Sikora, M. (2017). Wearable Emotion Recognition system based on GSR and PPG signals. *MMHealth 2017 - Proceedings of the 2nd International Workshop on Multimedia for Personal Health and Health Care, co-located with MM 2017*, 53–59. https://doi.org/10.1145/3132635.3132641

Understand the Activity Lifecycle | Android Developers. (sem data). Obtido 10 de Junho de 2020, de https://developer.android.com/guide/components/activities/activity-lifecycle#java

Valkonen-Korhonen, M., Tarvainen, M. P., Ranta-Aho, P., Karjalainen, P. A., Partanen, J., Karhu, J., & Lehtonen, J. (2003a). Heart rate variability in acute psychosis. *Psychophysiology*, *40*(5), 716–726. https://doi.org/10.1111/1469-8986.00072

Valkonen-Korhonen, M., Tarvainen, M. P., Ranta-Aho, P., Karjalainen, P. A., Partanen, J., Karhu, J., & Lehtonen, J. (2003b). Heart rate variability in acute psychosis. *Psychophysiology*, *40*(5), 716–726. https://doi.org/10.1111/1469-8986.00072

Villanueva-Miranda, I., Nazeran, H., & Martinek, R. (2018, Novembro 9). CardiaQloud: A remote ECG monitoring system using cloud services for eHealth and mHealth applications. *2018 IEEE 20th International Conference on e-Health Networking, Applications and Services, Healthcom 2018*. https://doi.org/10.1109/HealthCom.2018.8531164

What is the Importance of Web Application Architecture? (sem data). Obtido 27 de Agosto de 2020, de https://www.excellentwebworld.com/web-application-architecture/

Yew, H. T., Ng, M. F., Ping, S. Z., Chung, S. K., Chekima, A., & Dargham, J. A. (2020). IoT Based Real-Time Remote Patient Monitoring System. *Proceedings - 2020 16th IEEE International Colloquium on Signal Processing and its Applications, CSPA 2020*, 176–179. https://doi.org/10.1109/CSPA48992.2020.9068699

Yi, N., Feng, X., & Li, C. (2018). The design and implementation of the front end of the art play library system. *Proceedings - 7th International Conference on Communication Systems and Network Technologies, CSNT 2017*, 285–288. https://doi.org/10.1109/CSNT.2017.8418553

Zeeshan Baig, M., & Kavakli, M. (2019). A survey on psycho-physiological analysis & measurement methods in multimodal systems. *Multimodal Technologies and Interaction*, *3*(2), 1–20. https://doi.org/10.3390/mti3020037

Zemrane, H., Baddi, Y., & Hasbi, A. (2019). Ehealth smart application of WSN on WWAN. *ACM International Conference Proceeding Series*, *Part F1481*. https://doi.org/10.1145/3320326.3320358

Anexo

Manual de Instruções do Sistema Desenvolvido

Wristband E4

Coloque a *wristband* E4 no pulso da mão não dominante e garanta que fica bem ajustada, para que os elétrodos não alterem a sua posição ao longo da monitorização.

Ligue o dispositivo carregando 2 segundos no botão. O indicador LED irá apresentar a cor verde de forma intermitente durante 60 segundos e é dentro deste período que deve fazer a ligação com a aplicação Android. Caso a ligação não seja estabelecida o dispositivo entrará no modo automático de gravação.

Aplicação Android

Verifique a sua ligação à Internet e abra a aplicação Android.

Ligue o *Bluetooth* e pressione o botão azul que permite procurar um dispositivo. Assim que a ligação seja estabelecida, os dados sensoriais começam a ser apresentados no ecrã e o indicador LED do dispositivo apresentará a cor azul durante alguns segundos.

Selecione os dados que pretende monitorizar à distância e pressione o botão verde de transmissão para dar início ao envio dos dados.

Quando pretender terminar a transmissão pressione o botão vermelho.

Aplicação Web

Registe-se na aplicação através do seu *email* e proceda à confirmação desse email através do *link* que irá receber na sua caixa de correio eletrónico.

Efetue o *login* para ter acesso à aplicação onde pode visualizar os gráficos correspondentes aos dispositivos que estejam a transmitir, ou visualizar as sessões realizadas no passado.

No separador das transmissões serão apresentados os dispositivos que estão a transmitir. Selecione o dispositivo cujos dados pretende visualizar, sendo de seguida apresentados os gráficos dos sinais em função do tempo. Tem a possibilidade de deslocar os gráficos para a esquerda ou para a direita para

visualizar valores anteriores ou posteriores aos que estão a ser apresentados, e fazer zoom, aumentando ou diminuindo o intervalo de tempo de visualização. Pode ainda definir valores máximos e mínimos para que a cor da linha do gráfico mude quando esses valores forem ultrapassados.

No separador das sessões será apresentada uma lista com as sessões anteriormente realizadas. Selecione o botão verde caso pretenda ver os gráficos da respetiva sessão. Selecione o botão vermelho caso pretenda eliminar definitivamente a sessão.

Para sair da aplicação basta pressionar o botão “Sair”.