Universidade de Évora - Escola de Ciências e Tecnologia

Mestrado em Engenharia Informática

Dissertação

Sistema de sensores com recolha e análise de indicadores em tempo real para o acompanhamento à distância de grupos de pacientes cardíacos

João Francisco Silva Caçador Pereira Rouxinol

Orientador(es) | José Saias

Évora 2025

Universidade de Évora - Escola de Ciências e Tecnologia

Mestrado em Engenharia Informática

Dissertação

Sistema de sensores com recolha e análise de indicadores em tempo real para o acompanhamento à distância de grupos de pacientes cardíacos

João Francisco Silva Caçador Pereira Rouxinol

Orientador(es) | José Saias

Évora 2025

A dissertação foi objeto de apreciação e discussão pública pelo seguinte júri nomeado pelo Diretor da Escola de Ciências e Tecnologia:

Presidente	|	Teresa Gonçalves (Universidade de Évora)
Vogais	|	José Saias (Universidade de Évora) (Orientador) Vitor Beires Nogueira (Universidade de Évora) (Arguente)

Évora 2025

À minha família, especialmente ao seu mais novo membro, o pequeno António.

Agradecimentos

A escrita desta secção marca o fim de um marco pessoal e académico, que só foi alcançado graças ao apoio de um conjunto de pessoas a quem se torna imperativo expressar a minha gratidão.

Em primeiro lugar, gostaria de agradecer ao Professor José Saias pela sua orientação ao longo de todo o processo, sendo o seu apoio, compreensão e disponibilidade essenciais para a conclusão do trabalho apresentado. Para além da ajuda na dissertação, os ensinamentos, como o rigor e atenção ao detalhe, transmitidos pelo Professor ao longo de todo o meu percurso académico foram essenciais para o sucesso profissional alcançado até aqui.

À *Plexus Tech*, em especial ao Eduardo Flores Belo, pela empatia, compreensão e flexibilidade dadas sempre que precisei de conciliar a dissertação com as minhas responsabilidades profissionais.

À minha namorada, Margarida, o meu profundo obrigado por me apoiar desde o primeiro dia desta jornada, sendo não apenas um pilar essencial da minha vida académica, mas também o alicerce da minha vida pessoal. Foste quem nunca me deixou desistir, mesmo nos momentos mais desafiadores.

Por fim, gostaria de terminar com um agradecimento especial à minha família, cujo apoio incondicional foi a base que me permitiu chegar aqui. À minha irmã, cuja experiência académica me orientou não só na escrita, mas também na forma de equilibrar a gestão emocional. Aos meus pais, as minhas maiores referências, por acreditarem em mim mesmo quando eu duvidava, e por me motivarem sem hesitação ao longo destes 28 anos. Se esta dissertação é a materialização do meu esforço, é também o reflexo do vosso amor e sacrifício.

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Acrónimos

API Application Programming Interface

AVC Acidente Vascular Cerebral

BD Base de dados

BLE Bluetooth Low Energy

BPM Batimentos por minuto

DCV Doenças cardiovasculares

ECG Eletrocardiograma

HTTP Hyper Text Transfer Protocol

HTTPS Hyper Text Transfer Protocol Secure

IDE Integrated Development Environment

IoMT Internet das Coisas Médicas

IoT Internet das Coisas

MVC Model-View-Controller

MVP Model-View-Presenter

MVVM Model-View-View Model

REST Representational State Transfer

RMSSD Root Mean Square of Successive Differences

SDNN Standard Deviation of NN Intervals

SGBD Sistema de Gestão da Base de Dados

SMTP Simple Mail Transfer Protocol

SSE Server-Sent Events

SSL Secure Socket Layer

UI User Interface

Resumo

As doenças cardiovasculares constituem a principal causa de mortalidade global, sendo responsáveis por mais de 18 milhões de óbitos anuais, evidenciando a necessidade de soluções inovadoras de monitorização cardíaca. O controlo da frequência cardíaca (FC) e da variabilidade da frequência cardíaca (VFC) através da análise de um eletrocardiograma (ECG) permite a avaliação contínua da condição cardíaca. Nesta dissertação, propõe-se um sistema integrado de monitorização remota para pacientes cardíacos, composto por sensores *Movesense* para aquisição de sinais ECG, uma aplicação móvel iOS para visualização e partilha de dados em rede, um serviço web baseado em *FastAPI* para comunicação paciente-tutor, e uma base de dados *SQLite* para armazenamento do histórico cardíaco de pacientes. O sistema permite monitorização coletiva em tempo real, com validação experimental a demonstrar precisão do mesmo ($<$5% de erro em comparação a medições manuais). A solução visa democratizar o acesso a cuidados cardíacos, facilitando a deteção precoce e acompanhamento personalizado.

Palavras chave: doenças cardiovasculares, monitorização cardíaca, frequência cardíaca, variabilidade da frequência cardíaca , internet das coisas

Abstract

Sensor system with real-time collection and analysis of parameters for remote monitoring of groups of cardiac patients

Cardiovascular diseases are the leading cause of global mortality, responsible for over 18 million deaths annually, demonstrating the need for innovative cardiac monitoring solutions. Controlling heart rate (HR) and heart rate variability (HRV) through electrocardiogram (ECG) analysis enables continuous evaluation of cardiac condition. In this dissertation, we propose an integrated remote monitoring system for cardiac patients, composed of Movesense sensors for ECG signal acquisition, an iOS mobile application for data visualization and network sharing, a FastAPI-based web service for patient-tutor communication, and an SQLite database for storing patient's cardiac history. The system enables collective real-time monitoring, with experimental validation demonstrating its accuracy ($<$5% error compared to manual measurements). This work aims to democratize access to cardiac care, facilitating early detection and personalized follow-up.

Keywords: cardiovascular diseases, cardiac monitoring, heart rate, heart rate variability, internet of things

1

Introdução

Este capítulo tem como principal objetivo apresentar o contexto, motivação e objetivos do trabalho desenvolvido, bem como a estrutura da dissertação.

1.1 Contexto

Estima-se que, em 2023, existissem mais de quinhentos milhões de portadores de algum tipo de doença cardíaca a nível global [Wor23], o que representa aproximadamente 6% da população mundial. Associada à sua elevada prevalência, as doenças cardiovasculares constituem a principal causa de morte no mundo. Apesar dos números apresentados serem alarmantes, uma percentagem significativa destas patologias pode ser prevenida através da adoção de medidas simples, como a cessação do consumo de tabaco e álcool, a prática regular de exercício físico, a redução da massa gorda e a adoção de uma alimentação saudável.[Wor21].

Para além da redução de fatores de risco, a monitorização da saúde cardíaca é fundamental, não apenas para a prevenção, mas também para o acompanhamento da progressão da doença em indivíduos já diagnosticados. A monitorização constante permite a deteção precoce de possíveis problemas cardíacos, fase em que estes são mais facilmente resolúveis e apresentam melhor prognóstico. A avaliação contínua contribui ainda para o ajuste dos planos de tratamento, com base na eficácia das intervenções. Deste modo, a monitorização cardíaca assume um papel central na prestação de cuidados personalizados e eficazes ao doente cardíaco [Cen23].

Com a evolução tecnológica sentida no século XXI, a Internet das Coisas (IoT, do inglês *Internet of Things*) surgiu como uma das áreas mais inovadoras, nomeadamente no domínio da saúde. A IoT engloba uma vasta gama de dispositivos físicos equipados com sensores, *software* e capacidades de conectividade, que

permitem a comunicação e a partilha de dados através da rede. Estes dispositivos possuem aplicações multifacetadas, abrangendo desde casas inteligentes e agricultura de precisão até à área da saúde, onde têm provocado transformações profundas em diversas especialidades médicas [Ora24a].

No contexto da medicina, a IoT tem permitido a introdução de um número crescente de dispositivos dedicados à monitorização de diversos parâmetros de saúde, sendo estes dispositivos habitualmente conhecidos como *wearables*. Estes dispositivos, que integram sensores, atuadores e *software* específico, incluem, entre outros, monitores de glicemia, termómetros digitais e sensores de frequência cardíaca. A sua capacidade de recolher, processar e transmitir dados em tempo real tem revolucionado a forma como são prestados os cuidados de saúde.

De entre as inúmeras aplicações da IoT na saúde, destaca-se a monitorização remota de pacientes. Esta prática consiste na recolha contínua de métricas de saúde, através de *wearables*, por parte de indivíduos que não se encontram fisicamente em unidades de saúde, permitindo que os dados sejam acessíveis e analisados por profissionais de saúde à distância. O principal objetivo destas soluções é assegurar uma vigilância constante do estado de saúde do paciente, independentemente da sua localização geográfica. Além de promover uma maior comodidade para os doentes, esta abordagem contribui para a redução de custos associados a internamentos hospitalares e para a diminuição da afluência aos serviços de saúde, otimizando assim a utilização de recursos médicos [Kaj24]. Esta transformação, impulsionada pela IoT, não só reforça a eficiência dos sistemas de saúde, como também abre caminho para um paradigma de medicina mais personalizada, preventiva e centrada no paciente.

Neste contexto, o trabalho apresentado nesta dissertação propõe a implementação de um sistema de monitorização à distância de pacientes cardíacos. Para isso, foi desenvolvida uma aplicação móvel que, aliada a sensores para medição de frequência cardíaca, movimento e eletrocardiograma (ECG), permite a monitorização em tempo real da frequência cardíaca (FC) e da variabilidade da frequência cardíaca (VFC) através de uma interface *user-friendly* e acessível remotamente.

Além da criação da interface, o objetivo do trabalho incluiu o desenvolvimento de um sistema completo que permita a transmissão de dados entre o utilizador e o profissional de saúde. Para alcançar este propósito, foi implementado um sistema de comunicação através da rede, garantindo que a comunicação ocorre de forma eficiente e independente da localização do utilizador. Devido à necessidade de gerir múltiplos utilizadores e os seus perfis, foi também desenvolvida uma base de dados para armazenar e organizar os dados recolhidos de forma segura e estruturada.

A solução desenvolvida visa colmatar lacunas no acompanhamento contínuo de pacientes cardíacos, facilitando a monitorização à distância e promovendo uma assistência mais eficiente. Um dos seus principais casos de uso consiste na realização de sessões de prática desportiva ou fisioterapia, em conjunto com uma aplicação de chamadas de vídeo, como o Zoom. Durante estas sessões, o profissional de saúde responsável pode observar em tempo real o comportamento cardíaco dos participantes e a sua variabilidade durante a prática de exercício físico.

1.2 Motivação

1.2.1 Saúde Cardiovascular: Um problema iminente.

Segundo dados da Organização Mundial de Saúde, as doenças cardiovasculares são a principal causa de morte a nível mundial, sendo responsáveis por cerca de 18 milhões de mortes anualmente (ou seja, uma morte a cada 2 segundos), com 80% destas resultantes de enfartes do miocárdio e acidentes vasculares cerebrais (AVC) [Wor]. Para além do número elevadíssimo de mortes, estimam-se também que, todos

os anos, sejam reportados mais de 35 milhões de eventos cardiovasculares agudos anualmente [Wor24]. A literatura atual indica que as mortes por doença cardiovascular são, na sua maioria, causadas pelos seguintes fatores de risco [Cen24]:

- **Hipertensão arterial**: a principal causa de doença cardiovascular. É muitas vezes denominada de "assassina silenciosa" pelo facto de não apresentar sintomas até que os seus valores estejam demasiado elevados, podendo assim já ter causado danos durante muitos anos [FW20].;
- **Alimentação**: uma má alimentação, aliada ao sedentarismo, pode causar obesidade, frequentemente acompanhada de um aumento significativo da pressão arterial;
- **Ingestão de álcool**: o consumo excessivo de álcool contribui para o aumento da pressão arterial;
- **Tabagismo**: o consumo de tabaco, especialmente através de cigarros, pode causar danos na corrente sanguínea, aumentando significativamente o risco de enfarte. Adicionalmente, a nicotina presente nos cigarros contribui para a elevação da pressão arterial.
- **Sedentarismo**. A ausência de exercício físico é outro fator que contribui não só para o aumento da massa gorda de um indivíduo, como também a sua resistência à insulina, sendo assim um fator de risco para Diabetes tipo II [HHZ14]. A Diabetes tipo II é um causador direto de hipertensão arterial.

Apesar dos números apresentados anteriormente serem assustadores, estima-se que cerca de 80% de todas as doenças cardiovasculares são preveníveis através do controlo dos fatores acima descritos [Wor24].

Apesar da prevenção ser o principal ponto de ação, após diagnosticada, a doença cardiovascular não deve ser ignorada, sendo que o diagnóstico da mesma é importantíssimo e deve ser feito o mais rápido possível, pois a deteção precoce pode significar um prognóstico bastante melhor.

1.2.2 O papel da monitorização cardíaca.

Apesar da doença cardíaca apresentar vários sintomas, tais como dor peitoral, falta de ar, tosse excessiva e cansaço [Mound], estes são por norma sentidos apenas em fases mais avançadas, sendo que, em fases iniciais da doença cardiovascular os pacientes são, na sua maioria, assintomáticos. Isto leva a que grande parte das pessoas apenas seja diagnosticada tardiamente, muitas vezes apenas em apenas devido a um evento extremo, como um enfarte ou AVC, o que pode não só causar danos irreversíveis, como até levar à morte do indivíduo [Nat24]. Assim, é importante tomar um papel proativo no diagnóstico precoce, procurando encontrar formas de detetar anomalias que possam estar associadas à doença cardiovascular. Uma das formas mais simples e eficazes é a monitorização cardíaca. A monitorização cardíaca consiste na utilização de algum tipo de equipamento que faça uma leitura da atividade do coração. Esta leitura tem como objetivo principal a visualização de alguns parâmetros da atividade cardíaca, como a pressão arterial ou frequência cardíaca, que, após interpretação, podem ser utilizados para identificar irregularidades que coincidam com algum tipo de doença cardíaca. A monitorização cardíaca é um procedimento simples e sem efeitos adversos, mas com diversas vantagens, tais como:

- **Deteção precoce**. A monitorização cardíaca é um auxílio indispensável na deteção precoce de condições cardíacas, permitindo que as mesmas sejam diagnosticas em períodos onde ainda não apresentam sintomas, e são mais facilmente tratadas.
- **Auxílio no tratamento**. Para além da ajuda no diagnostico, a monitorização cardíaca é especialmente útil em pacientes em tratamento, pois permite perceber a eficácia do plano delineado.

- **Identificação de comportamentos de risco**. Com a monitorização cardíaca, é possível isolar comportamentos do paciente, para perceber quais têm uma influência mais negativa, o que reduzir a progressão da doença.

1.2.3 Democratização da monitorização

Apesar da importância inquestionável da monitorização de pacientes cardíacos, esta monitorização tem de ser pensada de uma forma que apresente uma relação **custo-benefício** aceitável, dado que realizar a monitorização cardíaca constante em ambiente hospitalar levanta algumas questões, tal como o custo em material hospitalar, a necessidade de pessoal qualificado, bem como a comodidade do paciente. Assim, o ideal seria limitar a monitorização cardíaca em ambiente hospitalar apenas em doentes com quadro clínico mais reservado (AVC, enfartes do miocárdio ou outros tipos de condições cardíacas graves).

Para que a promoção da saúde cardiovascular não seja apenas limitada aos pacientes que se encontrem em unidades de saúde, torna-se bastante interessante a criação de alternativas que permitam a monitorização cardíaca remotamente, nomeadamente, sistemas de **Internet das Coisas**.

Na área da saúde, utilização de IoT possui inúmeros casos de uso, tendo-se consolidado nos últimos tempos como uma das tecnologias mais transformadoras do setor, tendo originado um ramo específico do ramo, conhecido como **Internet das coisas médicas** (IoMT, de *Internet of Medical things*). A IoMT engloba qualquer tipo de dispositivo acoplado a sensores, que pode ser utilizado para a melhoria de vários processos hospitalares, tais como:

- **Gestão de equipamentos hospitalares**. A integração da IoMT pode fazer-se através da instalação de sensores em equipamentos hospitalares, sendo que estes sensores podem fazer uma monitorização contínua do estado dos equipamentos, facilitando o processo de deteção de anomalias e manutenção, o que pode resultar numa redução de custos significativa. Adicionalmente, podem ser integrados sensores que facilitem a utilização dos equipamentos, fornecendo *feedback* em tempo real ao profissional de saúde.
- **Monitorização de pacientes**. Atualmente, a integração de sistemas da IoMT permite uma otimização substancial no que toca à monitorização de indicadores de saúde. O desenvolvimento de dispositivos deste tipo permite não só a recolha contínua de dados, como também a disponibilização dos mesmos, facilitando assim a sua análise. [Gle24]. Alguns exemplos deste tipo de dispositivos são sensores de glucose e leitores de pressão arterial inteligentes, bem como todo o tipo de ***wearables***.

Um *wearable* consiste num tipo de dispositivo eletrónico que deve ser utilizado no corpo humano e tem como principal objetivo recolher métricas do utilizador, como a sua frequência cardíaca, número de passos diário ou outros indicadores vitais. Graças à sua facilidade de uso e apresentação de resultados intuitiva, estes dispositivos tornaram-se ferramentas valiosas na prevenção e monitorização de doenças, especialmente no contexto da saúde cardiovascular. A capacidade destes em fornecer dados em tempo real e de forma acessível ao utilizador permite que um indivíduo tenha uma maior consciência do seu estado de saúde, incentivando a adoção de hábitos mais saudáveis e a deteção precoce de potenciais problemas. Outra das grandes vantagens dos *wearables* é o seu custo relativamente baixo quando comparado com a monitorização em ambiente hospitalar. Enquanto a monitorização contínua em unidades de saúde envolve a utilização de equipamentos de custo elevado e intervenção pessoal especializado, os *wearables* oferecem uma alternativa acessível e conveniente para a recolha de indicadores de saúde. Atualmente, a evolução da tecnologia permitiu o desenvolvimento de *wearables* com sensores mais precisos, capazes de realizar a medição de indicadores como a FC ou VFC de forma extremamente eficaz. Isto é fundamental para democratizar o

acesso à monitorização cardíaca, permitindo que um maior número de pessoas possa acompanhar a sua saúde de forma regular, sem a necessidade de recorrer a serviços médicos dispendiosos ou deslocações frequentes a unidades de saúde.

1.3 Objetivos

1.3.1 Desenvolvimento de um sistema de monitorização cardíaca

O objetivo principal deste trabalho consiste no desenvolvimento de um sistema de IoMT que, através da utilização de dispositivos *wearables*, permita a medição de indicadores cardíacos críticos para o diagnóstico e acompanhamento de pacientes com patologias cardíacas. Estes dados, uma vez recolhidos, devem ser partilhados de forma remota com profissionais de saúde, facilitando a monitorização contínua e a intervenção médica atempada. Paralelamente, para garantir que os pacientes tenham uma perceção clara do seu estado de saúde, o sistema deve incluir uma interface gráfica intuitiva e de fácil interpretação. O caso de uso principal do sistema centra-se na sua aplicação durante sessões terapêuticas realizadas à distância, integrado com plataformas de videoconferência, como o Zoom, permitindo uma interação eficaz entre pacientes e profissionais de saúde. A análise de indicadores cardíacos, tais como a FC e a VFC, assume um papel crucial no diagnóstico e na avaliação da gravidade da condição clínica dos pacientes. Para tal, o sistema recorrerá a sensores de ECG, cujo *output* gerado permitirá o cálculo da FC e VFC.

1.3.2 Criação de um sistema intuitivo

Dado o caráter remoto do sistema, é imperativo que a sua utilização seja simples e acessível a todos os pacientes, independentemente do seu nível de familiaridade com tecnologias digitais. Para tal, será desenvolvida uma interface gráfica de fácil compreensão e navegação, totalmente integrada com os sensores utilizados, que permita não só a transmissão eficiente dos dados para os profissionais de saúde, mas também a visualização clara e imediata desses dados pelos utilizadores. Adicionalmente, será criada uma interface dedicada aos profissionais de saúde, igualmente intuitiva, que possibilite a monitorização em tempo real dos dados dos pacientes, garantindo uma experiência de utilização fluida e eficaz para ambas as partes.

1.3.3 Criação de um sistema seguro

O terceiro objetivo prende-se com o nível de segurança do sistema. Um sistema desta natureza envolve a transmissão e armazenamento de dados sensíveis, que incluem não apenas indicadores de saúde, como a FC e a VFC, mas também informações pessoais dos pacientes, tais como nome, idade e endereço de email. A natureza sensível destes dados exige a implementação de medidas robustas de segurança, de modo a garantir a proteção, privacidade e integridade das informações. Assim, serão pensadas várias aplicadas várias medidas de segurança durante o desenvolvimento do sistema, tais como protocolos de encriptação, controlo de acessos e outras práticas recomendadas para assegurar a conformidade com as normas de proteção de dados.

1.3.4 Desenvolvimento de um sistema flexível e escalável

Embora o foco inicial do sistema seja a monitorização remota de pequenos grupos de pacientes cardíacos, a sua arquitetura deve ser concebida para ser escalável, permitindo a expansão não só para a monitorização

de grupos de grandes dimensões, como também a possibilidade de monitorização de vários grupos em simultâneo. Adicionalmente, o sistema foi projetado com flexibilidade, de modo a facilitar a integração futura de novas funcionalidades, como a adição de sensores adicionais ou a medição de outros indicadores de saúde relevantes para a avaliação da condição cardíaca. Para garantir estas características, todos os componentes do sistema foram desenvolvidos segundo uma arquitetura modular.

1.4 Estrutura do documento

Esta dissertação está organizada de forma a apresentar de maneira clara e coerente o desenvolvimento do trabalho realizado, contendo uma separação clara entre os capítulos de fundamento teórico e os que apresentam detalhes concretos sobre a implementação do sistema, sendo o trabalho composto por seis capítulos. De seguida, é feita uma breve síntese ao conteúdo de cada um destes capítulos.

1.4.1 Capítulo I - Introdução 1

Apresenta uma contextualização do problema encontrado, bem como uma enumeração de todas as motivações e objetivos do trabalho proposto. Adicionalmente, é apresentada a estrutura da dissertação.

1.4.2 Capítulo II - Enquadramento Teórico 2

Aborda os conceitos teóricos relevantes relacionados com a monitorização cardíaca através de uma revisão bibliográfica ao trabalho já realizado dentro da temática, tentando apresentar as melhores práticas para a monitorização cardíaca, bem como lacunas nos sistemas existentes.

1.4.3 Capítulo III - Metodologia 3

Apresenta propostas para a estrutura do sistema, bem como técnicas utilizadas para o desenvolvimento do mesmo. Inclui em detalhe o processo de análise e escolha de tecnologias para cada uma das camadas do sistema, bem como a definição de critérios para avaliação do mesmo.

1.4.4 Capítulo IV - Implementação 4

Descreve todos os processos e caminhos tomados na implementação do sistema, sendo dividido em duas partes fundamentais. Na primeira, é descrito o processo de desenvolvimento, onde são abordados temas como a arquitetura do sistema e outras escolhas feitas. Na segunda parte, são descritas as funcionalidades principais do sistema final.

1.4.5 Capítulo V – Análise dos resultados 5

Neste capítulo, são apresentados os testes realizados para avaliar o desempenho, a precisão e a fiabilidade do sistema. Inclui-se a descrição dos métodos de teste, os resultados obtidos e a sua análise crítica, comparando-os com os objetivos inicialmente definidos.

1.4.6 Capítulo VI – Conclusões e recomendações 6

Apresenta as principais conclusões retiradas da análise realizada no capítulo anterior. Adicionalmente, são referidos pontos importantes de melhoria para trabalho futuro.

2

Estado da Arte

O enorme crescimento e adoção da IoMT permitiu o desenvolvimento de soluções cada vez mais inovadoras para a monitorização de pacientes, em todas as áreas da medicina. No contexto da doença cardíaca, a necessidade de monitorização contínua e precisa impulsionou a criação de sistemas de monitorização remota cada vez mais avançados. Este capítulo tem como principal objetivo a realização de uma revisão ao trabalho atualmente disponível no contexto de sistemas de monitorização cardíaca, abordando os avanços tecnológicos, tecnologias utilizadas, resultados obtidos e desafios presentes atualmente. Esta revisão tem como objetivo não só traçar um plano para o sistema a desenvolver, contextualizando o mesmo, como também tentar destacar a relevância do projeto proposto.

2.1 Análise bibliográfica

2.1.1 Métricas de análise

A análise da atividade cardíaca pode ser efetuada de várias formas, podendo, através de diferentes métricas, obter diferentes indicadores para a saúde cardíaca. Para desenvolver um sistema de monitorização cardíaca a ser utilizado em sessões terapêuticas, é necessário primeiro analisar quais os parâmetros a monitorizar.

Em ***"Wearable sensors and devices for real-time cardiovascular disease monitoring"*** [LFZ+21], é efetuada uma análise sobre algumas das técnicas mais utilizadas para monitorização cardíaca, destacando-se as seguintes:

- ***Pulse Waves***. Consiste na medição da elasticidade arterial, sendo útil e importante na deteção de condições específicas como a arteriosclerose, por sua vez, na deteção das patologias mais comuns, não é a mais indicada.

- ***SCG (Seismocardiogram)***. Reflete a vibração nas paredes do coração causado pelo batimento cardíaco. É uma solução de monitorização não invasiva, ainda assim, a execução de movimentos pode introduzir ruído no sinal SCG, o que faz a sua utilização para um sistema de monitorização durante a atividade física não ser a mais indicada.
- ***BCG (Ballistocardiogram)***. Consiste na medição das forças de resposta à ejeção de sangue do coração para os vasos sanguíneos. Apesar de ser um método não invasivo, a sua medição é por norma realizada a pacientes quando estão deitados (em repouso), o que torna a não apropriada para utilização durante a atividade física.
- ***PCG (Phonocardiogram)***. Consiste no registo de todos os sons emitidos pelo coração durante um ciclo cardíaco. É um método rápido e fácil de efetuar, ainda assim, dado utilizar o som como métrica, é sujeito a ruído.
- ***ACG (Acoustic Cardiograph)***. Reflete alterações de vibração entre cada período do ciclo cardíaco. É possível efetuar a sua medição sem contacto cutâneo, o que o torna ideal bastante útil para medições a longo prazo.
- ***ECG (Electrocardiography)***. Consiste na medição da atividade elétrica do coração, sendo estes sinais utilizados para calcular indicadores importantes do coração, como a frequência cardíaca ou variabilidade da frequência cardíaca (sendo que este último pode refletir várias condições cardíacas). É considerado o método standard para o diagnostico de condições cardíacas (como arritmias ou condições mais graves), sendo regularmente utilizado em ambiente hospitalar. A sua medição requer que sensores (elétrodos) estejam sempre em contacto com o paciente. Outro ponto a ter em conta é que o sinal ECG é suscetível a interferência do ambiente externo, sendo que isto pode ser contornado selecionando sensores que possuam baixa impedância.

Concluindo, este artigo apresenta reflexões importantes sobre os tipos mais comuns de monitorização cardíaca, apresentando soluções que monitorizem a atividade mecânica do coração (como o SCG) e a atividade elétrica (como o ECG), descrevendo alguns casos de uso, bem como as principais vantagens e desafios de cada uma delas. Tendo em conta todos os pontos apresentados, bem como as motivações apresentadas, é útil para criar um sistema que faça a monitorização de ECG, pois com isto é possível calcular a **frequência cardíaca** e a **variabilidade da frequência cardíaca**, que são os dois melhores previsores da doença cardiovascular.

O primeiro estudo analisado evidenciou, de forma robusta, que a avaliação contínua dos sinais de ECG de um indivíduo constitui uma das abordagens mais promissoras para o desenvolvimento de sistemas de monitorização destinados à utilização durante a prática de atividade física. No segundo estudo analisado, ***"Wearable Devices for Remote Monitoring of Heart Rate and Heart Rate Variability - What We Know and What Is Coming"*** [AAR22], é novamente destacada a relevância do desenvolvimento de sistemas integrados que combinem a monitorização da FC com a VFC. A frequência cardíaca assume um papel de extrema importância, não apenas como indicador do esforço físico do paciente, mas também como parâmetro fundamental de avaliação da adaptação do sistema cardiovascular. Por sua vez, a VFC emerge como um marcador preditivo de eventos cardiovasculares adversos, tais como arritmias, insuficiência cardíaca e, em casos mais graves, enfartes do miocárdio ou AVC. Além disso, a VFC revela-se um indicador valioso para a avaliação de outros parâmetros fisiológicos, nomeadamente a capacidade de recuperação pós-esforço e a resposta ao stress fisiológico. Neste estudo, é também realizada uma avaliação crítica aos dispositivos de monitorização cardíaca disponíveis no mercado à data da publicação do mesmo (2022), destacando os desafios persistentes associados à aquisição de sinais frequentemente comprometidos pela presença de ruído. Para mitigar esta problemática, são propostas abordagens que integram tecnologias na ótica da aprendizagem automática, tais como redes neuronais artificiais, com o objetivo de melhorar

a interpretação dos sinais de ECG. Estas técnicas têm o potencial de aumentar a precisão dos cálculos da VFC, proporcionando resultados mais fiáveis e realistas, que se assemelhem aos obtidos em contexto hospitalar. Em síntese, a análise deste estudo reforça que a utilização combinada da frequência cardíaca e da variabilidade da frequência cardíaca representa uma via estratégica para o desenvolvimento do sistema idealizado, alinhando-se com as tendências mais recentes no domínio da monitorização cardiovascular.

Dada a importância dada à VFC nos primeiros dois estudos analisados, foi realizada uma revisão da literatura associada à mesma. Em ***"Heart rate variability: Measurement and emerging use in critical care medicine"*** [JBJKW20], é realizada uma análise aprofundada sobre o papel da monitorização da VFC no contexto da saúde. A VFC é obtida através do cálculo da variação temporal entre batimentos cardíacos consecutivos, regularmente denominados **intervalos RR**. No âmbito da saúde cardiovascular, a VFC assume uma posição de destaque como um dos parâmetros mais relevantes para a avaliação da função cardíaca, destacando-se não apenas como um previsor robusto de morte súbita, mas também como um indicador valioso para a monitorização de pacientes, sendo inclusivamente utilizada para a identificação do estado clínico da doença, avaliação da eficácia de intervenções terapêuticas, bem como ao acompanhamento na recuperação pós-evento cardíaco.

Adicionalmente, este estudo apresenta vários métodos para o cálculo da VFC, uma vez que este parâmetro pode ser quantificado através de diferentes abordagens. Entre os métodos destacam-se:

- **Medidas no domínio do tempo**. Caracterizadas pela sua simplicidade de cálculo, estas medidas exigem, contudo, que os dados sejam estacionários, uma vez que são sensíveis a artefactos, tais como ruído ou batimentos cardíacos irregulares.
- **Medidas no domínio da frequência**. Estas técnicas envolvem a análise da VFC em diferentes bandas de frequência, utilizando a transformada rápida de Fourier. Apesar da sua utilidade, também são suscetíveis à presença de artefactos.
- **Medidas não lineares**. Técnicas como a análise de entropia destacam-se pela sua robustez face a artefactos e pela capacidade de processar dados não estacionários. No entanto, a sua complexidade computacional é superior em comparação com os métodos anteriores.

Adicionalmente, o estudo aborda os desafios inerentes à análise da VFC, com particular ênfase nas limitações técnicas associadas ao processamento de sinais. A implementação de algoritmos eficazes de pré-processamento de dados é identificada como um requisito fundamental para assegurar a precisão do cálculo da VFC, um aspeto crítico para a credibilidade e fiabilidade dos sistemas de monitorização cardíaca que integram este parâmetro. Em síntese, o trabalho em análise não só reforça a importância da VFC como ferramenta essencial para a monitorização cardíaca remota, como também oferece perspetivas valiosas sobre as abordagens metodológicas e os desafios técnicos que devem ser considerados no desenvolvimento de sistemas inovadores nesta área. Estas reflexões contribuem para uma compreensão mais aprofundada das vantagens da VFC e delineiam caminhos promissores para a sua aplicação futura em contextos clínicos.

Em ***"Heart Rate Variability Measurement through a Smart Wearable Device: Another Breakthrough for Personal Health Monitoring?"*** [LCMR+23], são descritas mais algumas vantagens da utilização da VFC na prevenção e monitorização da doença cardiovascular, nomeadamente o facto de ser um indicador muitas vezes monitorizado através de métodos não invasivos, tornando a obtenção do mesmo um processo mais cómodo para o paciente. Ainda assim, são apresentadas algumas considerações a ter em conta na monitorização da mesma, dado ser um indicador cardíaco que pode apresentar variações significativas derivadas de fatores quotidianos comuns, como medicações ou stress, mesmo que estas variações não signifiquem condição cardíaca debilitada. Com isto, os autores do estudo mostram preferência por sistemas de monitorização a longo prazo, sendo os valores de VFC obtidos nos mesmos mais indicativos da saúde

cardíaca do indivíduo em estudo. Adicionalmente, é feita uma revisão das soluções para a monitorização da VFC disponíveis no mercado, sendo as duas principais tecnologias de captação utilizadas atualmente, sendo elas **ECG** (analisada anteriormente) e **PPG**. **PPG** (*Photoplethysmogram*) consiste numa tecnologia que recorre à luz para calcular as variações volumétricas da circulação sanguínea, efetuando as medições através do cálculo dos intervalos de tempo entre pulsações, sendo uma pulsação detetada através da reflexão da luz emitida pelos sensores nos tecidos, sendo um método habitualmente considerado menos fiável de monitorização, pois é mais facilmente afetado por fatores externos, como o tom de pele do indivíduo ou as condições de luminosidade do ambiente. Este tipo de tecnologia é atualmente utilizada em dispositivos como *smartwatches*, sendo a sua principal vantagem a simplicidade de uso. Em suma, o estudo apresenta fortes indicadores que, através da utilização de sensores **ECG**, é possível desenvolver um sistema capaz de monitorizar a VFC de forma semelhante à realizada em ambiente hospitalar.

2.1.2 Sistemas de monitorização

A análise realizada permitiu não só a perceção da importância da VFC na prevenção e monitorização cardíaca, como também as melhores práticas para o cálculo da mesma. De seguida, foi analisada a literatura relacionada com sistemas de monitorização cardíaca à distância já disponíveis, com o objetivo de encontrar as melhores práticas, bem como identificar funcionalidades necessárias no sistema a desenvolver. Em ***"A Real-Time Health Monitoring System for Remote Cardiac Patients Using Smartphone and Wearable Sensors"*** [KTK15] é apresentado um sistema que consiste numa aplicação móvel aliada a um conjunto de sensores *wearables* para a monitorização em tempo real de pacientes cardíacos, tendo como principais objetivos oferecer uma solução que permita a monitorização cardíaca em tempo real e remotamente, sem comprometer a eficácia quando comparada com sistemas tradicionais presentes em instituições de saúde. A arquitetura do sistema proposto é constituída pelas três seguintes camadas:

1. **Sensores**. Neste sistema, a monitorização cardíaca é feita através de três parâmetros: frequência cardíaca, pressão arterial e temperatura corporal. A escolha destes parâmetros prende-se com o facto dos três serem geralmente utilizados na monitorização em clima hospitalar. A transmissão dos dados pelos sensores é realizada através da tecnologia **BLE** (*Bluetooth Low Energy*).
2. **Aplicação móvel**. Os dados recolhidos são transmitidos para uma aplicação móvel, desenvolvida especificamente para o caso de uso do sistema, sendo que neste sistema em específico, foi desenvolvida apenas para o sistema *Android*.
3. **Interface para monitorização**. Para permitir a monitorização dos pacientes por parte dos profissionais de saúde, foi desenvolvida uma interface *web* que comunica com a aplicação dos pacientes (através de rede *Wi-Fi* ou 3G), recebendo as informações cardíacos dos mesmos. Para facilitar a deteção de potenciais problemas, o sistema conta também com um sistema de alertas, sendo estes acionados sempre que for detetado algum tipo de anomalia (como uma arritmia ou hipertensão arterial).

Após realizados alguns testes ao sistema, foi possível demonstrar que o sistema permite a um profissional de saúde monitorizar com eficácia até vinte e cinco pacientes em simultâneo, sendo que, quando este processo é intercalado com outras tarefas hospitalares, o número de pacientes se situa nos doze. Estes resultados são promissores, mostrando não só a viabilidade da monitorização remota, como também o potencial deste tipo de sistemas na redução da pressão nas instituições de saúde, sem comprometer o bem-estar dos utentes. Ainda assim, ao desenvolver sistemas deste tipo, é importante relembrar que os mesmos envolvem a transmissão de dados sensíveis, como os dados pessoais ou indicadores de saúde dos pacientes, tornando-se essencial pensar em protocolos de comunicação seguros.

Em ***"An Automated Remote Cloud-Based Heart Rate Variability Monitoring System"*** [HkBF+18], é apresentado mais um sistema de monitorização cardíaca, tendo o mesmo a particularidade de funcionar através de *Cloud Computing*, sendo o principal argumento para a utilização desta tecnologia o facto de esta poder ser uma alternativa viável para a criação de um sistema de baixo custo, disponível através de um *Smartphone*. Este sistema utiliza sensores BLE para a captura dos sinais ECG, sendo o argumento para a escolha deste tipo de métrica o facto da mesma permitir o cálculo da VFC, considerada o método *standard* para a análise da condição cardíaca de um indivíduo, sendo um excelente previsor de várias condições. Apesar disto, foram identificados alguns desafios comuns na implementação deste tipo de sistemas, sendo que estes se prendem principalmente com o cálculo correto da VFC, dado que este cálculo pode ser adulterado por fatores como a utilização de um mau algoritmo de cálculo ou até ruído no sinal captado. São ainda apresentadas algumas soluções para lidar com estes problemas, como a aplicação de pré-processamento aos intervalos de RR obtidos do sinal ECG. Os testes realizados ao sistema apresentaram resultados bastante positivos, obtendo valores de precisão acima dos 95%, contudo, é importante salientar que os testes foram realizados através de dados disponíveis numa base de dados, pelo que este valor pode ser mais baixo quando realizado num ambiente em tempo real.

Em **"Portable heart rate measurement for remote health monitoring system"** [MMZ+15], é descrito o funcionamento de um sistema portátil de monitorização da frequência cardíaca, sendo que o mesmo faz parte de um projeto de nome *Home-smart Clinic*. O objetivo deste sistema é oferecer um sistema de monitorização contínua, que permita que profissionais de saúde tenham acesso aos sinais vitais (como a frequência cardíaca ou a temperatura corporal) de pacientes, enquanto estes estão nas suas casas. O sistema consiste em três componentes principais, o primeiro consiste num módulo de recolha de dados, onde estão incluídos sensores de frequência cardíaca e temperatura corporal, estando os mesmos conectados a um controlador *Arduino* (*Arduino* consiste numa plataforma *open-source* baseada numa placa com um circuito integrado que pode ser programado para controlar vários tipos de sensores [Ard18]), que transmite os dados para o computador do paciente. O segundo componente foca-se no processamento dos dados, e consiste numa base de dados *MySQL*, onde os dados são armazenados para visualização posterior. Por último, existe um módulo de comunicação, que permite a transmissão de dados em tempo real para um profissional de saúde, através de *Ethernet Shield*, que permite que um controlador *Arduino* se conecte à rede. A visualização dos dados pelos profissionais de saúde realiza-se através de uma aplicação *web* simples.

Os testes realizados ao sistema demonstraram valores de frequência cardíaca muito precisos quando comparados à medição manual, o que mostra potencial para a criação de uma alternativa válida à medição em ambiente hospitalar baseada neste sistema (ainda que seja necessária uma validação com uma equipa médica para que tal aconteça). Apesar dos resultados obtidos o tornarem uma opção interessante, as dimensões e ergonomia do sistema tornam-no uma solução pouco viável para a prática de exercício físico, sendo assim uma solução mais adequada à monitorização em repouso.

Em ***"IoT Based System for Heart Monitoring and Arrhythmia Detection Using Machine Learning"*** [CCMMGGOG23], é apresentado um sistema que combina *IoT* e inteligência artificial de modo a detetar arritmias, que consistem em irregularidades no batimento cardíaco de um indivíduo, podendo estas irregularidades consistir em batimentos demasiado lentos, rápidos ou de ritmo irregular [Nat22], devendo as mesmas ser resolvidas o mais rapidamente possível. A arquitetura do sistema consiste nas três camadas, sendo a primeira constituída por um sensor ***Polar H10***, capaz da captação de sinais ECG. A segunda camada consiste numa aplicação *móvel* onde são enviados os dados do sensor, sendo esta responsável pelo envio. A última camada é constituída pela *backend*, onde são aplicados algoritmos de aprendizagem automática como *kNN*, ***Random forest*** ou **Redes neuronais**, de forma a analisar os dados cardíacos automaticamente. A comunicação entre as camadas é feita através de *BLE* entre o sensor e a aplicação móvel e protocolos *HTTP*, a aplicação e a *backend*. Foram ainda realizados testes a cada algoritmo com o objetivo de encontrar o mais preciso, sendo que, em todas as situações testadas, foi o algoritmo **kNN** o

que obteve a precisão mais elevada (acima de 80%).

O trabalho apresentado demonstra o potencial claro na aplicação de inteligência artificial nos sistemas de monitorização cardíaca, especialmente na automatização de processos de análise de dados e sinais em tempo real, como no caso da deteção de arritmias. No caso do sistema a implementar, seria interessante a utilização de algum algoritmo deste tipo para a deteção automática dos intervalos RR, processo que poderia melhorar o cálculo da VFC. Ainda assim, não existem indicações sobre o desempenho do sistema na monitorização de vários pacientes em simultâneo, pelo que o desenvolvimento de uma solução deste tipo pode não funcionar de uma forma tão satisfatória nessas situações.

O trabalho apresentado em ***"Internet of things-based health monitoring system for early detection of cardiovascular events during COVID-19 pandemic"*** [Dam22] delineia o desenvolvimento de um sistema que utiliza IoT para a monitorização cardíaca remota. Este sistema foi concebido com o propósito específico de ser implementado durante a pandemia de COVID-19, período marcado pela adoção de medidas rigorosas de confinamento que, no âmbito das patologias cardíacas, resultaram na interrupção de consultas e exames de rotina essenciais para a avaliação do estado clínico dos pacientes. O sistema proposto visa facilitar a identificação precoce de eventos cardiovasculares, mediante a utilização de sensores ECG.

O funcionamento do sistema assemelha-se a outros já analisados anteriormente, baseando-se na captação de sinais ECG, os quais são posteriormente processados para calcular a FC e a VFC, sendo estas depois transmitidas aos profissionais de saúde, permitindo uma avaliação contínua e em tempo real. Para a análise e deteção de eventos cardiovasculares, o sistema recorre a modelos de aprendizagem automática, destacando-se a utilização de redes neuronais do tipo **LSTM** (*Long Short-Term Memory*), uma arquitetura eficaz no estudo de dados contínuos ou séries temporais [17], o que a torna ideal para este contexto. Quando detetado um evento cardiovascular, é emitido um alerta tanto na aplicação móvel destinada ao paciente, como numa plataforma *web* acessível aos profissionais de saúde.

Nos testes realizados, o sistema demonstrou uma precisão superior a 95%, indicando um desempenho promissor. Contudo, é importante salientar que estes resultados foram obtidos em ambientes experimentais controlados, pelo que a precisão em cenários reais poderá mostrar-se inferior. Não obstante, os resultados alcançados sugerem que soluções desta natureza representam uma alternativa viável, de baixo custo e elevada eficácia, face aos métodos tradicionais de monitorização em instituições de saúde, podendo este tipo de abordagem revelar-se determinante em contextos extremos, onde o acesso a cuidados de saúde se encontre limitado ou restrito.

Em ***"Wearable heart rate monitoring intelligent sports bracelet based on Internet of things"*** [XYH20], é retratado o funcionamento de um sistema de monitorização cardíaca à distância através de um *wearable*, sendo o principal objetivo o desenvolvimento de um sistema ergonómico e adequado à utilização durante a prática de atividade física. Para atingir esta ergonomia, é proposta a utilização de uma bracelete onde estão acoplados sensores capazes de efetuar medições de FC. Em termos de arquitetura, o mesmo funciona de forma semelhante a alguns apresentados anteriormente, onde os sensores comunicam com uma aplicação móvel através de *Bluetooth* ou *Zigbee*, sendo esta aplicação responsável pelo processamento e transmissão dos dados para uma plataforma em *cloud*, onde podem ser analisados posteriormente. Apesar do trabalho apresentado ser um exemplo de como é possível desenvolver soluções de monitorização cardíaca remota ergonómicas, o facto de apenas efetuar medições de FC, bem como o sensor utilizado efetuar medições PPG, que são apresentadas como soluções menos precisas em comparação com ECG, tornam-no uma solução pouco fiável quando comparado com outros sistemas analisados.

Por fim, em ***"Contemporary Advances in Cardiac Remote Monitoring: A Comprehensive, Updated Mini-Review"*** [PFT+24], é realizada uma análise aos avanços atuais no que toca a sistemas de monitorização cardíaca remota, destacando a capacidade deste tipo de sistemas em substituir a monitorização cardíaca presencial, levando à redução do tempo de avaliação de eventos cardíacos, referindo também o

papel da inteligência artificial não só na deteção, como também na previsão de condições cardíacas através da aplicação de algoritmos de aprendizagem automática em sinais ECG, sendo que esta análise demonstra claramente o enorme potencial da aplicação de inteligência artificial em sistemas de monitorização.

2.2 Sumário

Após uma análise detalhada da literatura, foi possível identificar pontos fundamentais que orientam o desenvolvimento do sistema idealizado. Em relação aos indicadores a monitorizar, a utilização de sinais ECG revelou-se a abordagem mais adequada, permitindo o cálculo de indicadores críticos como a FC e a VFC da forma mais precisa, sendo estes parâmetros essenciais para uma avaliação correta da saúde cardíaca de um indivíduo, sendo a FC um bom indicador do esforço cardiovascular, enquanto a VFC atua como um previsor de eventos cardíacos. Contudo, é crucial adotar estratégias eficazes para o processamento dos sinais obtidos pelos sensores ECG, dado que estes são suscetíveis a ruído e interferências. A seleção de algoritmos de cálculo robustos e precisos é, portanto, um aspeto central para garantir a fiabilidade das medições. Adicionalmente, a aplicação de inteligência artificial pode ser uma estratégia útil para o processamento dos sinais obtidos pelos sensores. No que diz respeito ao desenho do sistema, três pontos surgem como fulcrais:

- **Ergonomia**. A escolha de sensores que ofereçam liberdade de movimentos aos utilizadores é essencial, uma vez que o sistema será utilizado durante atividades físicas ou terapêuticas que exigem mobilidade.
- **Aceitação dos Utilizadores**. A interface do sistema deve ser intuitiva e de fácil utilização, tendo em conta o público-alvo, que pode incluir idosos ou indivíduos com pouca familiaridade com tecnologias digitais.
- **Privacidade e Segurança dos Dados**. Dado o caráter sensível dos dados de saúde, o sistema deve ser concebido com mecanismos robustos de proteção de dados, garantindo a confidencialidade e integridade das informações.

3

Metodologia

A definição de uma metodologia clara e robusta é fundamental para o desenvolvimento de qualquer sistema, garantindo que o mesmo seja capaz de responder aos requisitos funcionais e não funcionais. Neste capítulo, detalha-se a abordagem metodológica adotada para a conceção e implementação do sistema, incluindo pontos como a definição de casos de uso, bem como a descrição do processo de seleção para cada uma das tecnologias utilizadas.

3.1 Especificação

Dado que o objetivo principal do sistema apresentado é criar uma solução eficiente para a monitorização de pacientes cardíacos à distância durante a prática de sessões terapêuticas, o mesmo é composto por um sensor *wearable*, uma interface móvel para o paciente conectada ao sensor, uma interface móvel para o tutor e um serviço web que serve como plataforma de comunicação entre o utilizador e o tutor.

3.1.1 Arquitetura do sistema

A arquitetura do sistema é constituída pelas quatro seguintes camadas:

- **Sensores wearables**. Um sensor que faça a leitura dos sinais ECG e frequência cardíaca do paciente que e seja ergonómico, para que possa ser utilizado durante a prática de exercícios nas sessões terapêuticas.

- **Serviço Web**. Um protocolo de comunicação através da rede que permita não só a transmissão de informações cardíacas, como também sistemas de autenticação e gestão de perfis.
- **Interface para pacientes**. Uma aplicação móvel, conectada ao sensor, para que o paciente possa não só monitorizar os seus sinais vitais e acompanhar o seu histórico de saúde, como também enviar os mesmos para o serviço web.
- **Interface para tutores/professores**. Uma aplicação móvel que comunique com o serviço web de forma a receber as informações cardíacas dos pacientes em tempo real.

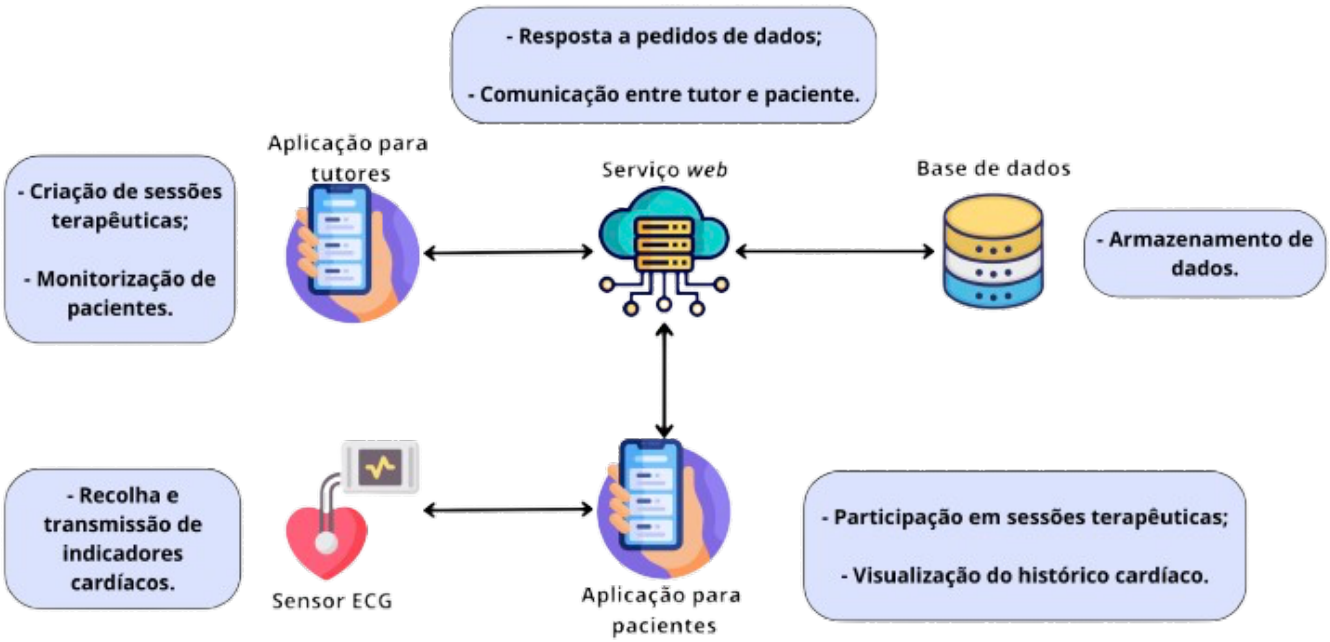

Figura 3.1: Diagrama das camadas do sistema

Quanto à comunicação entre as camadas do sistema, a transmissão de dados entre as mesmas é efetuada da seguinte forma:

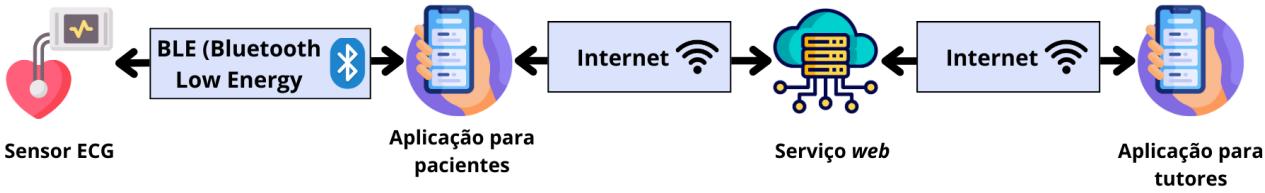

Figura 3.2: Comunicação entre as camadas do sistema

3.1.2 Requisitos funcionais

Para que o sistema possua a capacidade de executar tudo o que foi projetado durante a definição dos objetivos, o mesmo deve permitir as seguintes funcionalidades:

- **Monitorização da FC e VFC em tempo real**. O sistema deve ser capaz de fazer a medição dos sinais ECG em tempo real, transmitindo os mesmos para a interface do paciente, sendo que nesta os sinais devem ser processados rapidamente, e os valores da frequência cardíaca e variabilidade da frequência cardíaca calculados corretamente e transmitidos para a aplicação web continuamente.
- **Armazenamento do histórico dos pacientes**. Após cada sessão terapêutica, o sistema deve armazenar dados importantes relacionados com a monitorização efetuada, tais como a variabilidade da frequência cardíaca média. Todo este histórico deve ser visualizado para o paciente de forma clara e intuitiva.
- **Acesso remoto aos dados**. Os pacientes devem ser capazes de participar numa sessão independentemente da sua localização, desde que possuam conexão à rede. O mesmo se deve passar com os tutores no caso do acesso aos dados cardíacos dos pacientes.
- **Autenticação e gestão de perfis**. O sistema apenas deve permitir que utilizadores registados participem em sessões terapêuticas. No caso de ser tutor numa sessão, devem existir perfis com essas características.
- **Sistema completo**. O sistema estar estruturado de forma que os pacientes consigam executar todas as operações necessárias sem necessidade a plataformas externas, ou seja, deve ser possível que o paciente se registe na plataforma, se inscreva em sessões, participe nas sessões, se conecte aos sensores e envie os seus dados para a plataforma web tudo através da interface disponível, sendo a única exceção a plataforma de videoconferência utilizada para as sessões.

3.1.3 Requisitos não funcionais

Para além da funcionalidade do sistema, é importante também que o sistema cumpra com os seguintes requisitos não funcionais:

- **Segurança**. Todos os dados sensíveis dos utilizadores devem estar protegidos, sendo que funcionalidades como a implementação de criptografia devem estar presentes para garantir a privacidade dos utilizadores.
- **Desempenho**. Dado que, para ter leituras consistentes da FC e VFC é necessário fazer leituras constantes, é fulcral que o sistema seja capaz de não só processar dados rapidamente, como também transmitir os mesmos através do serviço web com velocidade semelhante. Assim, é importante projetar um sistema que seja capaz de receber comunicações a cada segundo.
- **Escalabilidade**. Para que o sistema seja útil, é necessário que o mesmo permita não só a realização de sessões terapêuticos com vários utilizadores em simultâneo, como também a realização de várias sessões em simultâneo. Assim, o serviço deve ser capaz de atender a um grande número de pedidos a cada segundo.
- **Aceitação e usabilidade**. Para que o sistema sirva o seu propósito, é necessário que o mesmo seja aceite pelos utilizadores, assim, é fulcral criar interfaces percetíveis e de fácil utilização, compatível com os dois sistemas principais de *smartphones* da atualidade (iOS e Android).

3.1.4 Utilizadores alvo

Dado o caso de uso principal do sistema ser a utilização do mesmo para realizar a monitorização cardíaca à distância durante sessões terapêuticas, podem ser considerados os dois seguintes tipos de utilizadores-alvo:

- **Pacientes cardíacos**, que utilizam o sistema para aceder a sessões terapêuticas e partilhar os seus dados cardíacos.
- **Tutores/professores**, que utilizam o sistema para ao orientar uma sessões terapêutica, monitorizando os indicadores cardíacos dos pacientes. Podem também ser considerados administradores do sistema como utilizadores, ainda assim, o objetivo principal é que o sistema seja gerido de uma forma quase automática, sendo apenas necessário a intervenção de um administrador em situações adversas.

3.1.5 Casos de uso

Para garantir a correta estrutura do sistema, foi necessário identificar todas as interações possíveis entre os atores e as funcionalidades existentes, sendo que este processo levou à criação de um conjunto de casos de uso, disponíveis no **Anexo A**.

Através dos casos de uso, foi possível desenvolver o seguinte **diagrama de casos de uso**:

3.1.6 Desafios

Após o estudo realizado no estado da arte e tendo em conta os desafios encontrados em outros tipos de sistemas de monitorização cardíaca, as maiores dificuldades prendem-se não só com a segurança e privacidade dos dados (dada a sensibilidade dos mesmos), como com garantir que o sistema funciona de forma aceitável para grandes grupos de pacientes. Adicionalmente, dado ser uma aplicação que poderá eventualmente ser utilizada por idosos com pouca familiaridade com tecnologia, desenvolver uma interface que permita que esse tipo de utilizador navegue de forma independente pode tornar-se bastante desafiante.

3.2 Tecnologias e protocolos utilizados

Para desenvolver um sistema que vá de acordo com os requisitos especificados anteriormente (e onde seja possível executar todos os casos de uso identificados), foi necessário fazer ponderações sobre quais as tecnologias e protocolos a utilizar. Após isto, em conjunto com a literatura analisada, o sistema final foi desenhado da seguinte forma:

3.2.1 Sensores *wearable*

Ao escolher o tipo de sensor a utilizar, a primeira questão prendeu-se com a escolha do tipo de dados a recolher do utilizador. A análise realizada no capítulo anterior permitiu concluir que para um sistema desta natureza, a escolha mais acertada passa por escolher um sensor que consiga monitorizar sinais ECG de pacientes, dado que estes permitem o cálculo de métricas como a FC e da VFC, métricas que, segundo a literatura, são consideradas dos indicadores mais importantes para a avaliação da saúde cardíaca de um indivíduo. A segunda questão é relacionada com tipo de conexão estabelecida entre os sensores e a interface do utilizador. As duas principais tecnologias de conexão regularmente adotadas por este tipo de dispositivos são o ***Bluetooth Classic*** e o **BLE**. A primeira destaca-se pela sua elevada taxa de transmissão de dados, que varia entre 1 Mb/s e 3 Mb/s, tornando-a particularmente adequada para aplicações que envolvam a transferência de conjuntos de dados de grandes dimensões, como a transmissão de áudio ou a conexão com periféricos [Blu24]. Contudo, esta elevada capacidade de transmissão implica, por norma, um consumo energético mais significativo, tornando esta uma opção pouco apropriada em conexões de longa duração

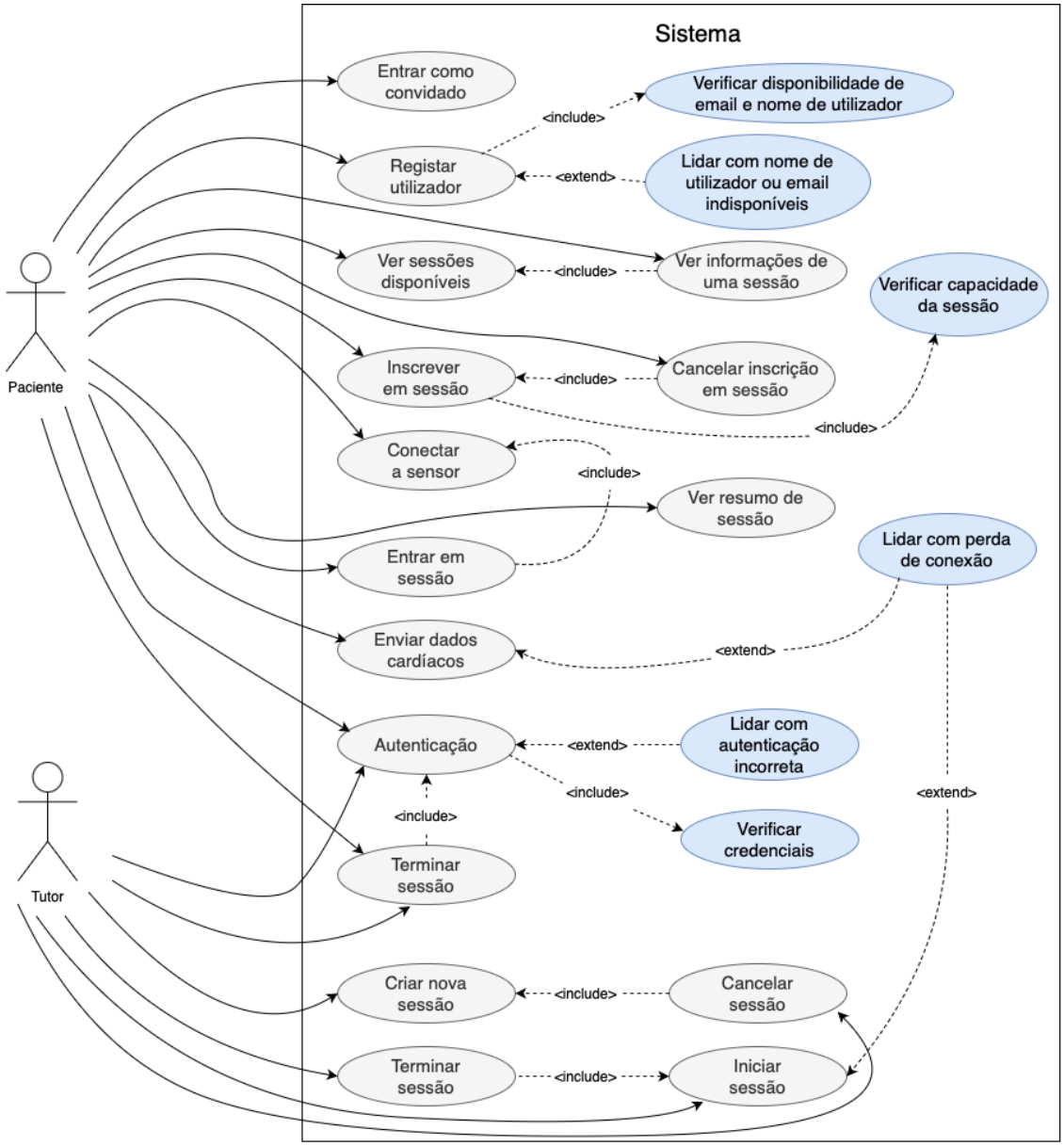

Figura 3.3: Diagrama de casos de uso proposto

e contínuas. Por outro lado, a tecnologia BLE caracteriza-se por taxas de transmissão consideravelmente mais reduzidas, entre 125 Kb/s e 2 Mb/s, o que a torna substancialmente mais eficiente em termos de consumo energético [Blu24], o que faz com que a mesma seja amplamente utilizada em dispositivos IoT, onde a conservação de energia é um fator fulcral. Tendo em consideração estes aspetos, a seleção do sensor adequado recaiu sobre um modelo que suporte a tecnologia BLE.

De modo a selecionar o sensor mais apropriado para o sistema, foi feita uma análise da literatura disponível relativa a sensores de medição de ECG com transmissão de dados através de BLE. Em [RSG22], é efetuada uma comparação entre dois tipos de sensores BLE: o Polar P10 e o *Movesense* ECG. Ainda o estudo se foque na avaliação da frequência respiratória, são utilizados os intervalos de RR obtidos em cada um dos sensores para o cálculo da mesma, sendo estes habitualmente utilizados para o cálculo da FC e VFC. A análise efetuada no estudo concluiu que que os sinais originados pelo sensor *Movesense* são mais detalhados e precisos. Estes resultados são um bom indicador de que a escolha pode passar por sensores *Movesense*

ECG.

Por fim, em [RSC+22], é apresentado um sistema de monitorização cardíaca utilizado durante períodos de atividade física incrementais (ou seja, a intensidade da atividade aumenta durante o tempo), de modo a entender a adaptação cardíaca dos pacientes ao esforço, sendo que o controlo foi feito através da análise da frequência cardíaca e variabilidade da frequência cardíaca dos intervenientes, sendo estes dados calculados através de um sensor *Movesense Medical*, que recolhe sinais ECG. Os resultados obtidos demonstram que o sensor obtém valores de precisão bastante altos durante a atividade física, o que, aliado à facilidade de utilização e certificação médica do mesmo, o tornam uma excelente opção para aplicações deste tipo. Dado isto, a escolha final passou pela utilização de sensores ***Movesense Medical***.

Os sensores *Movesense* são sensores *wearable* capazes de monitorizar vários parâmetros importantes relacionados com a saúde de um indivíduo, estando inclusivamente alguns dos produtos disponíveis certificados para utilização médica profissional. Dotados de vários mecanismos de deteção diferentes, como giroscópio ou acelerómetro, os sensores *Movesense* são capazes de recolher vários tipos diferentes de dados, tais como sinais ECG ou frequência cardíaca, podendo transmitir os mesmos através de tecnologia BLE [Mov23]. Para além de tudo isto, uma das principais vantagens da utilização deste sensor prende-se no elevado número de ferramentas disponíveis para o desenvolvimento de aplicações que utilizem o mesmo, existindo inclusivamente um conjunto de bibliotecas para *iOS* e *Android* disponíveis em [Mov24], com métodos que facilitam a recolha e tratamento de dados do sensor.

Quanto à portabilidade, em [Web25], é analisado o comportamento de um sensor *Movesense* em contexto de exercício de alta intensidade, demonstrando-se capaz de manter uma elevada precisão (99,6%) e sensibilidade (99,7%) mesmo em sessões de corrida ou ciclismo, onde os sistemas tradicionais de monitorização de ECG enfrentam desafios, dadas as suas dimensões. Os resultados descritos no estudo tornam-no uma opção atrativa para a utilização do mesmo no sistema proposto.

3.2.2 Aplicação móvel

Para garantir que o sistema é de fácil acesso, foi idealizada uma interface em forma de uma aplicação móvel para *smartphone*, para que tanto pacientes como terapeutas possam interagir com todo o sistema de forma simples e intuitiva. Dada a necessidade de comunicação com os sensores, é essencial que a aplicação tenha acesso ao *Bluetooth* do dispositivo móvel. Adicionalmente, para agilizar processos como a autenticação ou até o envio de registos cardíacos, é útil que a aplicação possua acesso a outras funcionalidades nativas do dispositivo, como a autenticação biométrica (como o sistema *FaceID* da Apple), ou até acesso total aos contactos telefónicos. É importante ter em consideração os pontos referidos quando decidido o tipo de desenvolvimento a realizar. Em [Soe24], é efetuada uma visão geral sobre os principais tipos de desenvolvimento de aplicações móveis:

- **Desenvolvimento nativo**, onde é criada uma aplicação para cada sistema operativo na linguagem nativa do mesmo. Este tipo de desenvolvimento tem como principal vantagem o acesso a todas as funcionalidades e hardware do dispositivo, permitindo desenvolver aplicações com melhor desempenho. Apesar disto, a necessidade de desenvolver uma aplicação para cada sistema operativo resulta habitualmente em maiores custos no desenvolvimento.

- **Desenvolvimento híbrido**, onde é desenvolvida uma só aplicação numa tecnologia que funciona em vários sistemas operativos, o que permite por norma uma redução significativa no tempo de desenvolvimento, conseguindo desempenho semelhante a uma aplicação nativa em grande parte dos casos. Ainda assim, a utilização de algumas funcionalidades nativas do sistema operativo podem

gerar a necessidade de introduzir lógica adicional complexa, o que pode dificultar o processo de desenvolvimento.

- **Web apps**, que consistem em aplicações baseadas em *browser*, sendo a principal vantagem destas a sua versatilidade, dado que o desenvolvimento de uma aplicação é suficiente para que seja utilizada em qualquer dispositivo com acesso a um browser (computadores, dispositivos *iOS* ou *Android*). Ainda assim, dado se tratar de uma aplicação browser, existem limitações de acesso a algumas funcionalidades nativas do sistema operativo, o que no caso específico deste sistema pode dificultar a comunicação com os sensores, tornando-a a pior opção de entre as apresentadas.

Para o sistema proposto, a prioridade principal prende-se em escolher uma metodologia de desenvolvimento que facilite a conexão com os sensores *Movesense*, dado que qualquer problema na integração dos mesmos pode levar a erros na monitorização cardíaca. Ao analisar os recursos disponibilizados pelo fabricante do sensor em [Mov24], é possível verificar-se que as únicas bibliotecas com suporte oficial são destinadas às plataformas *iOS* e *Android* nativas. Diante desse cenário, para a implementação do sistema proposto, serão elaboradas duas aplicações distintas: uma desenvolvida na linguagem *Swift*, especificamente projetada para dispositivos *iOS*, e outra na linguagem *Kotlin*, voltada para dispositivos *Android*. Essa abordagem assegura a compatibilidade e a otimização do sistema em ambas as plataformas.

No âmbito do desenvolvimento de aplicações nativas, é fundamental considerar diversos aspetos técnicos, entre os quais se destacam as *frameworks* dedicadas à construção de interfaces gráficas.

No ecossistema *iOS*, existem duas opções nativas para o desenvolvimento de interfaces: ***UIKit*** e ***SwiftUI***, sendo que ambas representam abordagens distintas, sendo a primeira uma abordagem que utiliza programação imperativa, ao contrário de *SwiftUI*, que recorre à programação declarativa para o desenvolvimento de componentes visuais. Em [WSP21], é efetuada uma análise às duas bibliotecas, destacando como pontos fortes do *UIKit* a sua maturidade. Ainda assim, a sua natureza imperativa torna o desenvolvimento mais lento e complexo, sendo que a sintaxe declarativa utilizada em desenvolvimento com *SwiftUI* torna a criação de componentes visuais mais intuitiva e rápida, sendo este processo ainda mais rápido dada a existência de uma funcionalidade de pré-visualização, não disponível em *UIKit*, que permite a visualização rápida dos componentes visuais desenvolvidos, sem necessidade de compilação. Adicionalmente, a utilização de *SwiftUI* é geralmente aliada a práticas de programação reativa, sendo este tipo de metodologia ideal para lidar com aplicações com um elevado número de operações assíncronas [Uğ24], cenário altamente provável no sistema, principalmente nas sessões de monitorização, onde são recebidos dados do sensor e enviados para o serviço *web* em simultâneo.

Em relação ao ecossistema *Android*, existem também duas opções nativas para o desenvolvimento de interfaces: *XML*, mais antiga e baseada em programação imperativa, e *Jetpack Compose*, que representa uma alternativa moderna e recorre à programação declarativa. Em [MR23], é efetuada uma análise ao processo de desenvolvimento de vários componentes comuns em aplicações em *XML* e *Jetpack Compose* como listas de elementos ou barras de navegação, concluindo-se que em todos os cenários analisados a utilização de *XML* resulta quase sempre num processo mais complexo e demorado quando comparado com *Jetpack Compose*, o que, aliado ao facto de esta tecnologia apresentar bastantes semelhanças a *SwiftUI*, a torna a escolha mais apropriada para a aplicação *Android*.

Sendo o maior desafio associado ao desenvolvimento de duas aplicações nativas o elevado tempo de desenvolvimento, foi efetuado um levantamento de possíveis soluções capazes de fazer a adaptação entre plataformas, ou seja, através de uma *iOS* obter uma aplicação *Android* funcional, ou vice-versa. No trabalho apresentado em [Bel24], é proposta a utilização de *Large Language Models* (LLM) para efetuar a conversão automática entre tecnologias. Como esperado, foram encontrados problemas comuns em ambos os casos, principalmente relacionados com navegação entre ecrãs ou funcionalidades obsoletas, ainda as-

sim, o modelo foi capaz de converter várias componentes complexas com eficácia, como vistas complexas ou animações, mostrando um caminho promissor neste nível. Dado isto, foi feito um levantamento de soluções semelhantes já disponíveis no mercado, de forma a encontrar uma ferramenta capaz de efetuar a adaptação do sistema de forma eficaz. Após uma análise detalhada, foram identificadas duas ferramentas para potencial utilização. A primeira, ***Kotlin Multiplatform***, consiste numa ferramenta que permite o desenvolvimento de código em linguagem *Kotlin*, sendo este posteriormente utilizado para gerar aplicações *iOS* e *Web* [Kot24], tendo como principal vantagem o facto de ser uma ferramenta desenvolvida pela *JetBrains*, empresa responsável pelo ambiente de desenvolvimento oficial do sistema *Android*, o que lhe garante um elevado nível de documentação e suporte. A segunda ferramenta encontrada, ***Skip Tools***, é uma ferramenta criada com o objetivo de criar aplicações multiplataforma através de código *Swift*, sendo que consiste num *plugin* que converte código *SwiftUI* em *Jetpack Compose* em tempo real, permitindo o desenvolvimento de aplicações para os dois sistemas operativos, sendo a sua principal vantagem a criação de um projeto *Android* com o código convertido [Ski24]. Em relação aos custos associados, a primeira ferramenta é disponibilizada de forma totalmente gratuita, sendo que a segunda possui uma versão gratuita disponibilizada a pequenas equipas de desenvolvimento e projetos pequenos ou sem fins lucrativos, onde se enquadra o sistema. Apesar da primeira opção contar com um maior nível de suporte e maior maturidade, a possibilidade de ter acesso ao projeto *Kotlin* gerado pela plataforma *Skip* torna esta ferramenta uma escolha mais apropriada. Adicionalmente, considerando que esta ferramenta foi disponibilizada apenas em 2023, verifica-se a inexistência de qualquer tipo de experiência prévia com o sistema, tornando a sua utilidade prática e o seu funcionamento atualmente desconhecidos. Essa lacuna representa uma oportunidade única para a documentação detalhada de todo o processo de utilização e análise de resultados, não apenas no que diz respeito à validação da ferramenta, mas também como contribuição de *feedback* para o seu aprimoramento futuro.

3.2.3 Serviço *web*

Para facilitar a comunicação entre pacientes e professores ou tutores, identificou-se a necessidade de desenvolver um serviço que funcione como uma ponte de comunicação entre as demais camadas do sistema, ou seja, uma **API** (*Application Programming Interface*). Para que seja possível realizar todos os casos de uso descritos na secção anterior, é essencial que o serviço web opere de duas formas distintas. A primeira deve seguir um protocolo do tipo pedido-resposta (*request-response*), que corresponderá à grande maioria das interações realizadas, como autenticação, visualização de sessões, entre outras, sendo a segunda uma comunicação contínua, necessária exclusivamente para o tutor durante o processo de monitorização cardíaca, uma vez que este necessita de receber dados constantemente sem efetuar novos pedidos, sempre que um utilizador entrar ou sair de uma sessão ativa ou enviar os seus dados cardíacos.

Dos tipos de API mais utilizadas em aplicações web, destacam-se as seguintes [Ama24b]:

- **API SOAP.** Consiste num protocolo onde a comunicação é feita exclusivamente em XML, sendo utilizado maioritariamente em sistemas *legacy* [Ama24c].
- **API Websocket.** Que compreende um tipo moderno de protocolo de comunicação bidirecional (ou seja, cada dispositivo conectado pode enviar e receber mensagens independentemente) [Ama24b]. É um protocolo de comunicação contínua.
- **SSE (Server-Sent Events)**. Consiste num protocolo onde a comunicação é feita de forma unidirecional (apenas do servidor para o cliente). Este protocolo é ideal para aplicações onde exista necessidade de enviar consistentemente notificações ao utilizador [MDN24]. É um protocolo de comunicação contínua.

- **API REST.** É um tipo de protocolo de comunicação que segue um modelo de pedido-resposta (ou seja, o cliente faz um pedido de informação ao servidor, sendo essa informação retornada na resposta), sendo que neste tipo de API a comunicação é sempre iniciada pelo cliente [Ama24c].

Com base nas necessidades do servidor, para a comunicação do tipo pedido-resposta, podem ser utilizados os protocolos SOAP ou REST. A escolha para o sistema recaiu sobre o protocolo REST, devido ao seu melhor desempenho [Ama24c]. No que diz respeito à comunicação contínua, uma vez que apenas é necessária uma comunicação unidirecional (do servidor para a interface do tutor), a utilização de *Server-Sent Events* (SSE) torna-se mais apropriada. A simplicidade deste protocolo garante um desempenho superior, aliada ao facto de funcionar sobre o protocolo HTTP, o que facilita a sua integração com o restante sistema [Ży24]. A escolha da tecnologia e linguagem de programação utilizadas para implementar esta camada do sistema foi fundamentada nos seguintes aspetos:

- **Simplicidade.** Para agilizar o processo de desenvolvimento, é importante escolher uma tecnologia cuja linguagem de programação seja de sintaxe simples e de rápido entendimento. Adicionalmente, uma sintaxe mais simples pode facilitar também o processo de testes e escrita de documentação.
- **Flexibilidade.** Dada a necessidade de implementação de API que suporte REST e SSE, é importante que a tecnologia escolhida possua um vasto conjunto de ferramentas para o desenvolvimento de APIs.
- **Desempenho.** Para que o servidor web seja eficiente, é importante que o mesmo possua bom desempenho, ou seja, consiga não só responder rapidamente aos pedidos dos clientes, como também ser capaz de processar vários pedidos simultaneamente. Assim, é importante escolher uma tecnologia com suporte para programação assíncrona.

Em [Par24], é apresentada uma síntese das linguagens de programação mais utilizadas no desenvolvimento de serviços web, destacando-se ***JavaScript*** (através de ferramentas como o *Node.js*), ***Python*** (com recurso a bibliotecas como *Flask* ou *FastAPI*), ***PHP***, ***Ruby*** e ***Java*** (utilizando tecnologias como o *Spring Boot*). De entre as opções mencionadas, as que melhor se adequam às necessidades do sistema são *Python* e *Java*, sendo que a primeira se destaca pela sua simplicidade e facilidade de leitura, enquanto a segunda apresenta semelhanças com as linguagens *Swift* e *Kotlin*, utilizadas na aplicação móvel do sistema, o que pode agilizar o processo de desenvolvimento.

Em [Poc24], é efetuada uma comparação entre *Java* e *Python* no contexto de desenvolvimento de serviços web, concluindo-se que serviços desenvolvidos em *Java* beneficiam de maior fiabilidade, desempenho e robustez, algo que torna a utilização desta linguagem ideal para sistemas complexos e de grandes dimensões. Por outro lado, *Python* oferece um processo de desenvolvimento mais ágil, graças à sua sintaxe simplificada, que permite uma escrita e manutenção de código mais rápida e eficiente, além de versatilidade e escalabilidade, sendo mais adequado para projetos de menor dimensão e equipas reduzidas.

Por fim, em [Pyt25], são descritos alguns pontos-chave sobre o desenvolvimento de serviços web, mais especificamente APIs em *Python*, destacando-se a simplicidade e facilidade de escrita de código, bem como o excelente desempenho em operações assíncronas, o que torna esta linguagem uma escolha ideal para o sistema proposto. Além disso, o papel das *frameworks* no desenvolvimento de APIs em *Python* é abordado, destacando-se a sua capacidade de auxiliar no funcionamento assíncrono, gestão de tráfego, segurança, entre outras funcionalidades. De entre as *frameworks* disponíveis, destacam-se as seguintes:

- **Flask**, que se destaca por ser extremamente leve, sendo ideal para projetos de dimensão reduzida. Apesar disto, a sua leveza deve-se ao facto de não incluir funcionalidades (como a integração com bases de dados) que outras opções apresentadas possuem [Fla25].

- **Django**, que é uma opção desenvolvida para a criação de serviços web de alto nível e complexidade, que segue uma ideologia de "*batteries included*", ou seja, inclui várias ferramentas e funcionalidades (como sistemas de autenticação ou protocolos de segurança) [Dja25]. Apesar da sua elevada versatilidade, o extenso leque de possibilidades faz com que seja uma *framework* mais complexa que as restantes opções apresentadas (sendo utilizada maioritariamente em projetos de grandes dimensões).
- **FastAPI**, projetada para o desenvolvimento rápido e eficiente de APIs, sendo uma das suas principais valências [Fas25a]. Através da programação assíncrona, esta *framework* permite a criação de serviços web de baixa latência (ou seja, capazes de responder a vários pedidos em simultâneo).

De entre as soluções analisadas acima, a simplicidade de desenvolvimento e elevado desempenho assíncrono proporcionados pela *framework* ***FastAPI*** tornam-na a opção mais viável para o sistema. Adicionalmente, esta é capaz de gerar documentação automaticamente, o que pode resultar em uma poupança temporal significativa.

3.2.4 Base de dados

Dada necessidade da persistência de alguns tipos de dados tais como perfis de pacientes e tutores, sessões e inscrições nas mesmas ou até dados cardíacos de pacientes, surgiu a necessidade de criar uma base de dados para o sistema. A primeira decisão a ser tomada envolveu a escolha do tipo de base de dados a implementar: **relacional** ou **não relacional**. Embora as bases de dados não relacionais ofereçam um elevado nível de flexibilidade e escalabilidade [Ama24a], foi realizado um levantamento exaustivo dos dados a serem armazenados no sistema, o que permitiu identificar a existência de múltiplas relações entre os diferentes tipos de dados. Exemplos dessas relações incluem a inscrição de pacientes em sessões ou o registo cardíaco de um utilizador numa sessão específica, sendo que a presença deste tipo de relacionamento entre os vários dados torna qualquer tipo de consulta numa base de dados não relacional significativamente mais complexa, o que inviabiliza a sua utilização neste contexto. Por outro lado, os conceitos fundamentais das bases de dados relacionais, como chaves primárias e estrangeiras, facilitam a identificação e gestão dessas relações, promovendo um sistema mais consistente e garantindo maior integridade e segurança dos dados — um aspeto prioritário para o sistema em questão [IBM21]. Assim, embora as bases de dados não relacionais sejam mais facilmente escaláveis, o desempenho de uma base de dados relacional convencional revela-se suficiente para o caso de uso real do sistema proposto. Após tomada a decisão de utilizar uma base de dados relacional, o próximo passo focou-se na escolha do SGBD SQL mais adequado. De entre as mais conhecidas, destacam-se as seguintes:

- **Oracle Database.** É o SGBD mais conhecido e utilizado atualmente [Red24], sendo reconhecido pelo seu desempenho excecional, aliado a um elevado nível de segurança, tornando-o ideal para empresas e projetos de larga escala que necessitem de uma solução com elevada disponibilidade [Ora24b]. Ainda assim, dado não ser uma solução gratuita e acarretar custos bastante elevados (ponto importante dada a natureza não lucrativa do sistema a desenvolver), esta pode não ser a opção torna-se pouco indicada para o sistema a desenvolver.
- **Microsoft SQL Server.** Dado ser uma ferramenta da Microsoft, esta oferece integração com outras ferramentas do ecossistema da empresa, tais como o Microsoft Azure, uma plataforma em nuvem, entre outros [RCW24]. Apesar de existir uma versão gratuita da ferramenta, não é usada mais nenhuma ferramenta do ecossistema Microsoft, pelo que não é necessário escolher algo de complexidade tão elevada, podendo-se assim optar por uma SGBD mais leve.
- **MySQL.** Trata-se de um SGBD disponibilizado em código aberto, sendo atualmente o segundo mais utilizado mundialmente [Red24]. A sua popularidade deve-se principalmente ao seu elevado

potencial de escalabilidade, desempenho e simplicidade [Eri24]. Em termos de desafios na utilização desta ferramenta, as prendem-se apenas com casos de concorrência extremamente alta que envolvam vários milhares de conexões em simultâneo, o que, no caso de uso específico do sistema projetado, não é uma limitação real, pelo que não é um fator eliminatório.

- **PostGreSQL.** Trata-se de outro SGBD de código aberto, com destaque principal na sua robustez e segurança. Contando com mais de 35 anos de existência, possui uma comunidade que contribui ativamente na identificação e rápida resolução de problemas [Pos24]. Ainda assim, existem alguns desafios aliados à sua utilização, tais como a complexidade de configuração do ambiente, o que se deve ao facto deste ser um sistema tão flexível, que permite a afinação de vários parâmetros para otimização de desempenho. Ainda assim, para a base de dados projetada para o sistema, recorrer a uma ferramenta tão complexa pode significar um custo temporal de desenvolvimento muito mais elevado, algo que deve ser evitado, pelo que esta opções como MySQL se tornam mais apropriadas.
- **SQLite.** É atualmente a décima SGBD mais utilizada mundialmente [Red24], sendo conhecida maioritariamente pela sua simplicidade, eficiência, portabilidade e distribuição em código aberto. Ao contrário de todas as descritas anteriormente que operam em forma cliente/servidor, o SQLite é integrado diretamente nas aplicações, sendo a base de dados alocada diretamente num ficheiro. É uma SGBD que não requer qualquer tipo de configuração ou servidor, sendo apenas necessário acesso a um binário para a sua utilização [SQL24]. Adicionalmente, sendo a base de dados guardada num ficheiro, a base de dados pode ser movida de dispositivo facilmente. Ainda assim, dado ser um sistema que se foca tanto na simplicidade e leveza, possui algumas adversidades, principalmente no que toca a problemas de concorrência na escrita, sendo que não é possível executar duas operações de escrita simultaneamente (existindo uma fila onde são colocadas todas as operações em espera) [SQL24]. Este aspeto pode tornar-se um problema se existirem várias operações de escrita em simultâneo.

De entre as opções descritas acima, podem primeiramente descartar-se as opções pagas, ou seja, a *Oracle Database* e a *Microsoft SQL Server*. Em segundo lugar e por se tratar de um sistema de configuração mais complexa, a utilização de *PostGreSQL* pode não ser a melhor opção, pois pode resultar num aumento significativo no tempo de desenvolvimento. Assim, a decisão final prendeu-se pela escolha entre *MySQL* e *SQLite*. Para decidir entre estas, foram primeiro pensadas em todas as operações de escrita a fazer sobre a base de dados, com base nos casos de uso descritos na secção anterior:

Operações de leitura:

- Autenticação.
- Consulta de sessões.
- Consulta de histórico cardíaco;
- Registo (verificação se nome de utilização ou email em uso);
- Visualização de dados do utilizador.

Operações de escrita:

- Adição de um novo utilizador;
- Criação de uma nova sessão;

- Remoção de uma sessão;
- Mudança de palavra-passe;
- Inscrição numa sessão;
- Adição de registo cardíaco.

Dado que nenhum dos sistemas (*MySQL* e *SQLite*) possui restrições na realização de operações de leitura em simultâneo, não é necessário tecer reflexões sobre as mesmas. Quanto à escrita, onde existem restrições de execução unitária de operações no sistema *SQLite*, a necessidade de executar muitas inserções ao mesmo tempo pode criar quedas de desempenho. Ainda assim, com base na utilização projetada para o sistema, o número de operações de escrita a realizar em simultâneo não é elevado o suficiente para que seja sentida redução notória no desempenho do sistema.

4

Implementação

A implementação do sistema proposto envolveu a criação de três componentes distintos: uma aplicação móvel, um serviço *web* e uma base de dados. Nesta secção são descritos alguns dos pontos mais importantes do processo, tais como as decisões tomadas, principais funcionalidades de cada um dos componentes, dificuldades encontradas e respetivas soluções.

4.1 Aplicação móvel

4.1.1 Arquitetura de desenvolvimento

O processo de implementação da aplicação iniciou-se na escolha da arquitetura de desenvolvimento, sendo as habitualmente mais utilizadas as seguintes:

- **MVC**, sendo as suas principais vantagens a sua simplicidade e rapidez de desenvolvimento;
- **MVP**, que oferece uma maior separação dos componentes em relação ao anterior;
- **MVVM**, que faz uma separação total entre os componentes visuais e toda a lógica, tornando o código-fonte mais organizado.

Apesar da sua popularidade em desenvolvimento de aplicações móveis, a arquitetura MVC origina habitualmente projetos muito pouco modulares, dado que a implementação de funcionalidades não planeadas se torna um processo muito mais complexo que nas restantes arquiteturas apresentadas, tornando-a uma opção pouco atrativa. Em relação à arquitetura MVP, apesar de mais modularizada que a arquitetura

MVC, o elevado número de tarefas pelas quais o *presenter* é responsável, tornam o código-fonte do mesmo muitíssimo mais extenso.

Em [IKP23], são apresentados alguns pontos chave do desenvolvimento de aplicações com arquitetura MVVM, tais como o suporte de programação reativa e *data binding*, que permite efetuar atualizações na vista automaticamente consoante mudanças no *View Model*, reduzindo assim as dependências diretas entre as várias camadas, tornando-se esta arquitetura ideal para aplicações com interfaces dinâmicas, como ecrãs com atualizações frequentes de dados.

Algumas vistas pensadas para a aplicação requerem componentes com atualizações frequentes, tais como o ecrã de sessão do paciente, onde se preveem atualizações da interface a cada medição de FC ou VFC efetuada, ou, no caso do ecrã de sessão do tutor, atualizações sempre que seja recebido um dado cardíaco de um paciente. A necessidade de comportamentos desta natureza levou à escolha da arquitetura **MVVM**.

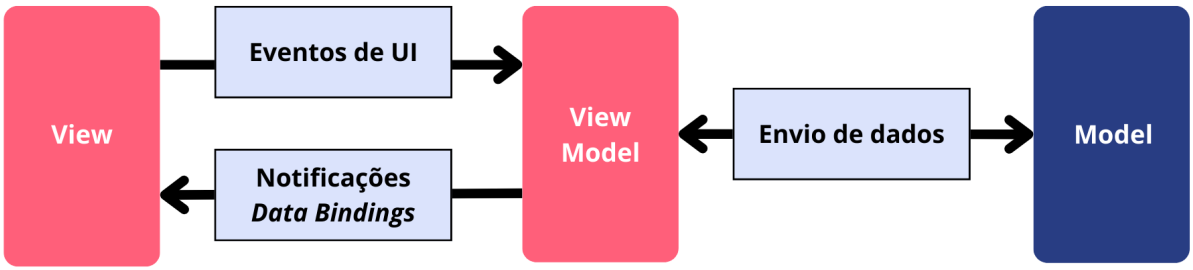

Figura 4.1: Funcionamento da arquitetura MVVM

4.1.2 Comunicação com o sensor

Para o desenvolvimento de uma camada de comunicação com o sensor, foi analisada a documentação disponível para *developers*. Em [Mov25], é escrita a arquitetura de comunicação entre o sensor e as aplicações móveis, sendo a mesma constituída pelos dois seguintes componentes:

- **Camada de software do sensor**, que consiste numa *framework* de microsserviços denominada de *Whiteboard*, utilizada para a comunicação, sendo que a mesma funciona através de pedidos do tipo REST.
- **Camada de software aplicacional**, onde são realizadas operações como a conexão ao sensor ou pedido de dados.

Para o desenvolvimento da camada aplicacional, deve ser tomada uma das seguintes decisões: desenvolver um cliente integralmente, algo que garante controlo absoluto da comunicação, ou optar pela biblioteca nativa disponibilizada para desenvolvimento, *Movesense-mobile-lib*, algo que segue uma ideologia de "*batteries included*", poupando tempo de desenvolvimento.

No caso específico da aplicação a desenvolver, onde é apenas necessário ter acesso aos sinais ECG para efetuar o cálculo da FC e VFC, não é justificado o desenvolvimento de uma camada de comunicação na íntegra.

Ainda assim, utilização de uma biblioteca externa (ou seja, não desenvolvida especificamente para o projeto) levantou as seguintes questões:

- ***Future proofing***. O desenvolvimento de novo hardware e novas funcionalidades para os sensores cria a hipótese de que a biblioteca utilizada sofra também alterações, tendo estas a possibilidade de quebrar o funcionamento da aplicação.
- ***Flexibilidade***. A utilização desta biblioteca obriga á utilização exclusiva de sensores da marca *Movesense*. Apesar destes serem sensores de alta precisão, esta decisão pode limitar o sistema a evolução futura do sistema.

Numa tentativa de solucionar estes dois pontos, foi criada uma camada intermédia que funciona como um gestor entre a aplicação e a biblioteca *Movesense*, sendo responsável pelo recebimento e tratamento de dados do sensor, convertendo os mesmos para um tipo de dados desenvolvido para a utilização na aplicação. A criação desta camada garante não só a fácil resolução de problemas causados por mudanças na biblioteca, como também a possibilidade de, no futuro, ser criado suporte para sensores de vários fabricantes.

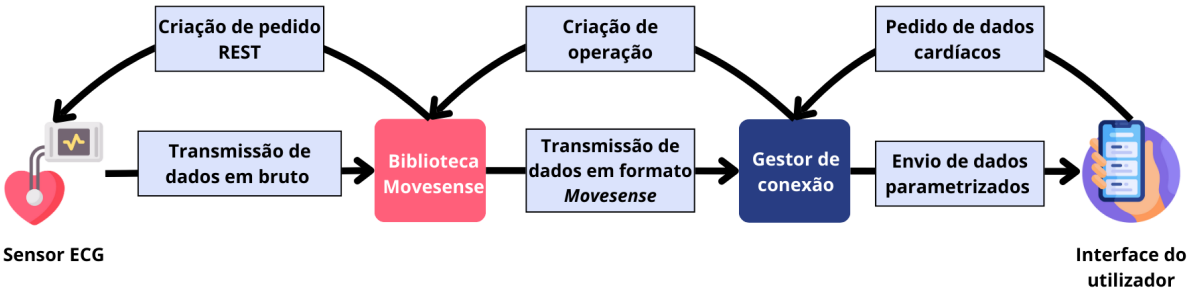

Figura 4.2: Funcionamento da comunicação entre o sensor e a aplicação utilizando o gestor de conexão

4.1.3 Desenho da interface

Sendo o público-alvo da aplicação pacientes cardíacos, sendo esta uma doença mais prevalente com a idade, é natural que muitos dos utilizadores da mesma sejam idosos. Em [Kli18], são descritos alguns dos desafios mais comuns da utilização de aplicações móveis por idosos, sendo o principal descrito a complexidade de algumas interfaces atuais.

De modo a que a aplicação desenvolvida seja de fácil navegação por utilizadores com menor proficiência tecnológica, foram considerados os seguintes pontos:

1. **Navegação intuitiva**. A existência de demasiados fluxos e caminhos possíveis no mesmo ecrã pode tornar a navegação muito complexa, algo que pode causar a desorientação de alguns utilizadores.
2. **Funcionalidades limitadas**. Apesar da implementação de mais funcionalidades poder resultar numa aplicação mais completa, optar por uma abordagem minimalista pode evitar a sobrecarga de informação.
3. **Interface simples**. A presença de demasiados componentes visuais pode sobrecarregar o utilizador, levando à frustração do mesmo.

Com base nestes pontos, foram tomadas algumas decisões em relação à interface. Em relação à navegação da aplicação, foi criada uma estrutura bastante simples, onde toda a navegação é efetuada de forma simples, através de toques únicos em áreas de contacto de grandes dimensões. Adicionalmente, foram adicionados alertas de dupla confirmação para ações de grande impacto no sistema, como o *logout*. As

funcionalidades implementadas focaram-se apenas em cumprir os casos de uso definidos no **Capítulo III**, sendo que para qualquer outra funcionalidade pensada, foi necessário efetuar uma análise ao impacto da mesma na complexidade do sistema.

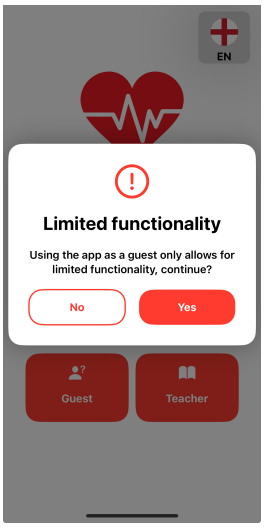

Figura 4.3: Exemplo de um alerta de dupla confirmação presente na aplicação

4.1.4 Monitorização da FC e VFC

Cálculo da FC

Para a obtenção da FC, o software do sensor oferece uma operação *heartrate*, cuja resposta inclui o valor médio da FC a cada instante de medição. A utilização desta operação elimina a necessidade de calcular esta métrica através da análise do sinal ECG, análise essa que se pode tornar demasiado complexa e com pouca precisão.

Cálculo da VFC

Quanto à análise da VFC, esta pode ser realizada de várias formas, existindo várias métricas para a análise da mesma. Dado que a VFC envolve o estudo dos intervalos RR ao longo do tempo, é possível utilizar os mesmos para o cálculo da VFC. Os intervalos RR são habitualmente obtidos através da análise dos sinais ECG, processo que por norma envolve a aplicação de algoritmos de análise. Ainda assim, este passo tornou-se desnecessário, dado que é possível obter os valores para os intervalos RR diretamente do sensor.

Quanto à escolha da métrica de análise da VFC, em [SG17], são analisadas várias métricas de cálculo da VFC, sendo que se podem dividir em dois grupos: **métricas do domínio do tempo** e **métricas do domínio da frequência**. As métricas do domínio do tempo são caracterizadas por um processo de cálculo mais simplificado, ainda que considerado menos preciso. Quanto às métricas do domínio da frequência, apesar de um cálculo mais complexo, estas resultam numa análise mais detalhada, sendo a sua aplicação mais direcionada a sistemas de monitorização contínua de longo prazo. Dado que o objetivo é a monitorização durante a participação de sessões de curta duração, torna-se mais útil a escolha de uma métrica do domínio do tempo, mesmo que menos detalhada.

De entre os principais tipos de métricas do domínio do tempo, destacam-se as seguintes:

- **SDNN**. Trata-se de uma medida calculada através do desvio padrão dos intervalos NN, estando valores mais altos da mesma associados a uma melhor saúde cardiovascular, sendo que é bastante utilizada em ambiente hospitalar.
- **RMSSD**, uma medida habitualmente utilizada para monitorização de curto prazo, sendo uma das suas principais utilizações a avaliação da recuperação ao esforço e relaxamento do indivíduo.

Dada a sua utilização em contexto hospitalar, a utilização da métrica SDNN é crucial para efetuar uma melhor avaliação dos pacientes, sendo esta a métrica escolhida. O cálculo de SDNN é dado pela seguinte fórmula:

$$\mathrm{SDNN} = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(\mathrm{NN}_i - \overline{\mathrm{NN}})^2}$$

A obtenção do valor dos intervalos NN pode ser obtida através da normalização dos intervalos RR, ou seja, um intervalo NN nada mais é que um intervalo RR após removido o ruído.

Para o cálculo dos intervalos NN, foi desenvolvido um algoritmo de filtragem que funciona da seguinte forma:

1. Em [Med24], são descritos como intervalos RR normais todos aqueles que se encontrem entre os 600ms e 1200ms. Dado que o sistema tem como utilizador alvo utilizadores com patologias cardíacas, que podem resultar em intervalos RR fora do normal, foi escolhida uma amplitude mais abrangente, entre os 300ms e os 1500ms, sendo que a mesma resultaria do aumento em 100% da amplitude do intervalo;
2. São removidos todos os intervalos RR que não se encontrem na amplitude definida;
3. De forma a remover anomalias locais que não sejam altas o suficiente para serem detetadas no passo anterior, foi criada uma *sliding window*, com o objetivo de criar o desvio padrão de cada um dos elementos em relação aos elementos vizinhos. Adicionalmente, foi definido um desvio padrão máximo de 20% entre vizinhos;
4. São removidos todos os intervalos RR cujo desvio padrão em relação aos vizinhos seja maior que o valor definido.

Envio dos dados cardíacos

Para a atualização dos dados cardíacos durante uma sessão, quer no ecrã do paciente quer no envio para o serviço *web*, foi necessário pensar numa estratégia que permita uma monitorização informativa o suficiente, sem que esta utilize demasiados recursos da aplicação a nível de memória e rede.

Quanto à transmissão de dados por parte do sensor, os testes realizados permitiram concluir que é recebida uma medição de FC e intervalo RR por segundo.

Dado que a FC é um indicador influenciado por fatores como o esforço físico, pequenos intervalos de tempo são suficientes para a deteção de alguns eventos cardíacos, sendo importante a visualização constante de variações. A necessidade de análise de todas as alterações da FC fez com que a sua atualização e transmissão sejam efetuadas a cada segundo.

Em relação à VFC, esta requer uma janela temporal mais alargada para fornecer resultados importantes, principalmente quando são utilizadas métricas como SDNN. Adicionalmente, dado que o cálculo da mesma requer a conversão dos intervalos RR em NN, é importante diminuir a frequência com que a mesma é calculada. Dado isto, foi utilizado um intervalo de tempo de um minuto para cada novo cálculo da VFC média.

4.1.5 Sessões terapêuticas

Interface do paciente

Após entrar numa sessão, é apresentada uma interface onde o utilizador pode visualizar todos os dados relativos aos seus indicadores cardíacos, sendo a sua FC apresentada através de um gráfico que apresenta as últimas cinco medições efetuadas, para além dos valores máximo, médio e mínimo. Quanto à apresentação da VFC, esta é acompanhada de *feedback* visual (através de uma imagem), sendo este negativo quando os valores obtidos sejam considerados não saudáveis (em [Mednd] é considerado não saudável todo o paciente cuja VFC se encontre abaixo de 50 ms).

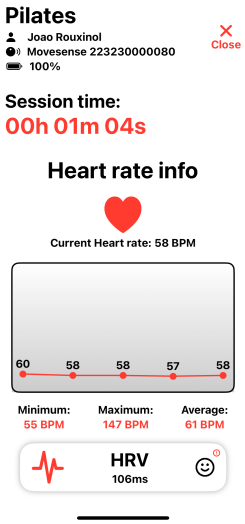

Figura 4.4: Ecrã de sessão para um paciente

Interface do tutor

Após iniciar uma sessão, o tutor é redirecionado para o ecrã de sessão, onde é possível, ao longo de toda a sessão, observar os indicadores cardíacos de cada um dos participantes na mesma. Dadas as limitações dos dispositivos móveis em relação às dimensões do ecrã, foi necessário criar uma lista com *scroll* vertical, para que o sistema mantenha a funcionalidade normal em sessões com vários participantes.

Em relação ao desenho da interface, é criado um componente visual para cada utilizador que participe na sessão. Em cada um destes componentes, o tutor pode observar as seguintes informações:

- Nome do paciente;
- Valores de FC (valor atual, médio, máximo e mínimo)

- VFC média;

Foi ainda desenvolvido um sistema de alertas, onde o tutor é notificado através de um pequeno *popup* sempre que um utilizador entra ou sai da sessão.

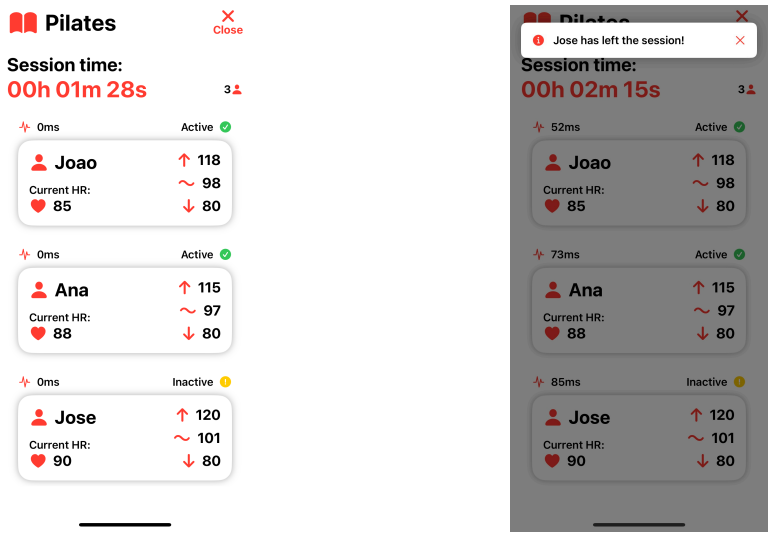

Figura 4.5: Interface do tutor durante uma sessão (à esquerda). Alerta de saída da sessão (à direita)

Após terminada uma sessão cardíaca, o tutor é redirecionado para um ecrã de sumário, onde é gerado um resumo das métricas cardíacas obtidas ao longo da sessão para cada utilizador. Para além das informações cardíacas, existe ainda a possibilidade do aparecimento de alertas para alguns pacientes, caso os valores máximos de FC obtidos sejam considerados demasiado elevados. Segundo [Ame24], o **valor de FC máximo** considerado saudável para um indivíduo é calculado da seguinte forma:

$$FCmax = 220 - idade$$

A criação do alerta é feita através da adição de um ícone informativo que, ao interagir com o componente relativo à sessão do respetivo paciente, apresenta um alerta no ecrã indicando o valor máximo saudável para o mesmo, dada a sua idade.

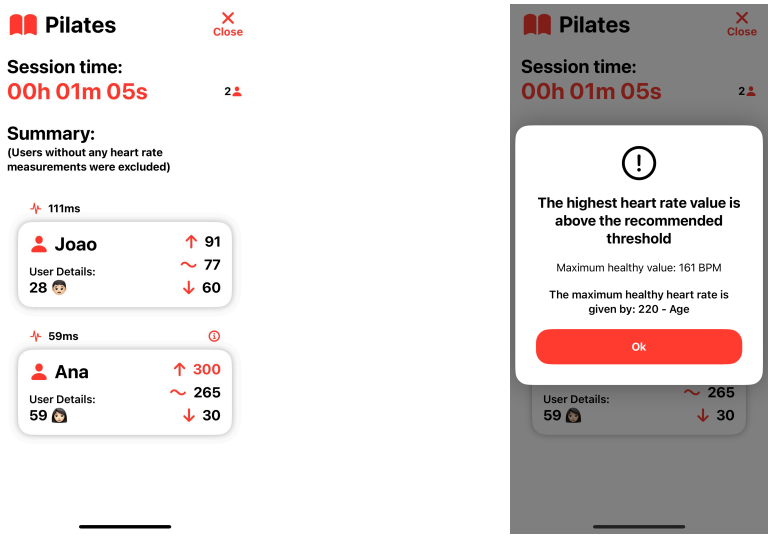

Figura 4.6: Sumário da sessão (à esquerda). Alerta de FC acima do saudável (à direita).

4.1.6 Histórico do paciente

Após participar numa sessão, o utilizador é redirecionado para um ecrã onde são apresentadas estatísticas importantes da sessão, tais como os seus indicadores cardíacos relativos à FC e VFC. Adicionalmente, foi desenvolvido um ecrã que permite ao utilizador visualizar o histórico de sessões nas quais já tenha participado. Para além da visualização do seu histórico cardíaco, é possível gerar uma imagem e partilhar a mesma através da aplicação com familiares ou até terapeutas. Esta funcionalidade foi desenvolvida com o principal objetivo de fornecer métricas a longo prazo da condição cardíaca do paciente, permitindo ao mesmo acompanhar a progressão da doença.

O sumário gerado é semelhante ao apresentado na figura abaixo:

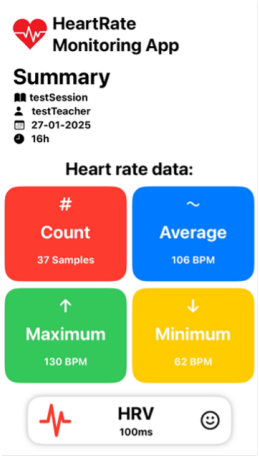

Figura 4.7: Exemplo de sumário gerado após uma sessão terapêutica

4.1.7 Adaptação para *Android*

Após terminado o desenvolvimento da aplicação *iOS*, foi iniciado o processo de adaptação da mesma para o sistema *Android*, através da *framework Skip*.

Na documentação oficial da ferramenta, é recomendada a criação de um novo projeto para que a utilização do *Skip* seja eficaz. Ainda que o projeto iOS já estivesse concluído, foram seguidos os seguintes passos:

1. **Criação de um novo projeto iOS**. Foi criado um projeto vazio, onde foi automaticamente aplicada a ferramenta *Skip*;

2. **Migração incremental**. Foram copiados os módulos do projeto completo para o novo de forma incremental, começando pelos ficheiros mais simples como pequenos componentes reutilizáveis, de modo a perceber as limitações da ferramenta.

Desafios encontrados

Durante o processo de migração, foram em vários momentos encontradas funcionalidades da linguagem *Swift* que a ferramenta não possui a capacidade de converter em código *Android* com sucesso.

De entre as várias funcionalidades onde o *Skip* apresentou dificuldades, a mais notória foi a biblioteca *Combine*, sendo esta uma biblioteca nativa do *Swift* e responsável pela manipulação de eventos assíncronos. Dado que esta foi a biblioteca utilizada para a implementação dos *data bindings* da arquitetura MVVM, todos os ecrãs da aplicação estão desenhados de forma a utilizar a mesma. Dado que não existe suporte para a conversão da biblioteca *Combine* para uma equivalente em *Android*, a conversão do código utilizando a ferramenta *Skip* tornaria obrigatória a realização de várias alterações no código.

Após se tornar evidente que a ferramenta não seria capaz de efetuar a adaptação do sistema de forma completa, foi então feita uma segunda experiência: utilizar a ferramenta *Skip* apenas para a conversão dos componentes visuais e ecrãs. Para isto, foi necessário isolar todos os componentes visuais do sistema de qualquer tipo de lógica, sendo que esta tarefa foi altamente facilitada pela utilização da arquitetura *MVVM*.

Analisando as figuras acima que correspondem aos resultados obtidos através da conversão da aplicação desenvolvida de forma nativa para iOS numa aplicação *Android*, ainda que tenha sido apenas possível a conversão de ecrãs e outros componentes visuais. Os resultados obtidos são bastante satisfatórios, sendo apenas notadas diferenças no que toca a alguns espaçamentos, tonalidades de cor e a ícones, sendo esta última relacionada com o facto de na aplicação iOS os ícones utilizados pertencerem ao próprio sistema operativo, enquanto que os ícones para a aplicação *Android* necessitaram ser obtidos através de repositórios de recursos visuais disponíveis na rede.

Globalmente, os resultados obtidos na utilização do *Skip* não podem ser considerados satisfatórios, dado que a ferramenta não foi capaz de efetuar a adaptação da aplicação diretamente devido a problemas de compatibilidade com bibliotecas Swift, tornando especialmente difícil a conversão de aplicações já desenvolvidas. Ainda assim, os resultados obtidos na conversão das componentes visuais do sistema mostraram-se bastante satisfatórios.

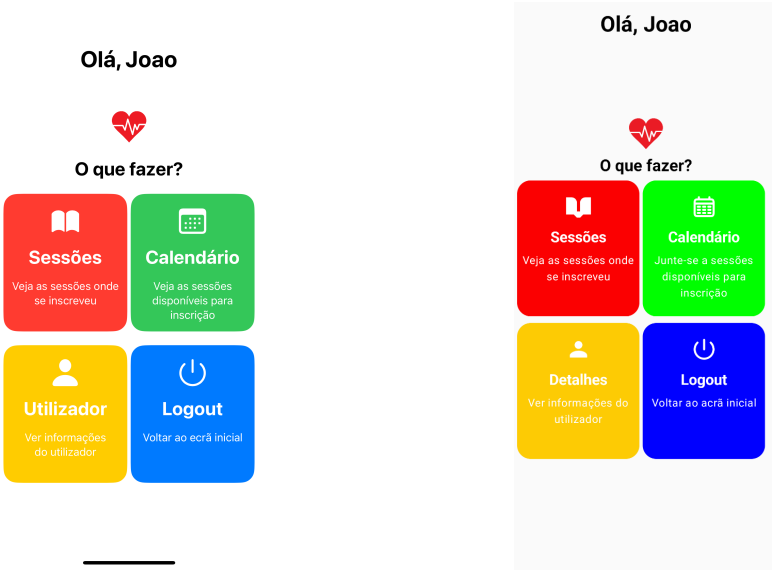

Figura 4.8: Aplicação desenvolvida para iOS e aplicação após adaptação para *Android*.

4.1.8 Funcionalidades adicionais

Suporte para múltiplos idiomas

A fim de ampliar a acessibilidade do sistema a um maior espectro de pacientes cardíacos, considerou-se essencial a implementação da aplicação em várias línguas. Desta forma, desenvolveu-se a aplicação com capacidade integral de funcionamento na língua inglesa e portuguesa, permitindo a alternância entre idiomas através de um mecanismo intuitivo no menu principal, com persistência da preferência do utilizador mesmo após o encerramento do aplicativo.

O suporte para múltiplos idiomas funciona através de um sistema de literais textuais organizadas em ficheiros isolados do código-fonte. Cada elemento textual é associado a um identificador único, sendo esse identificador utilizado no código-fonte para que o sistema de traduções consiga devolver a literal correspondente no idioma correto.

Para garantir a legibilidade do código, estabeleceu-se uma padronização na nomenclatura desses identificadores, estruturando-os conforme o ecrã e a funcionalidade específica a que correspondem.

Esta solução apresenta vantagens significativas no que toca à escalabilidade do sistema. A inclusão de novos idiomas no sistema torna-se um processo bastante simples, tornando-se apenas necessária a adição das correspondentes traduções no ficheiro de literais, sem que sejam necessárias alterações na base de código ou na lógica de funcionamento da aplicação.

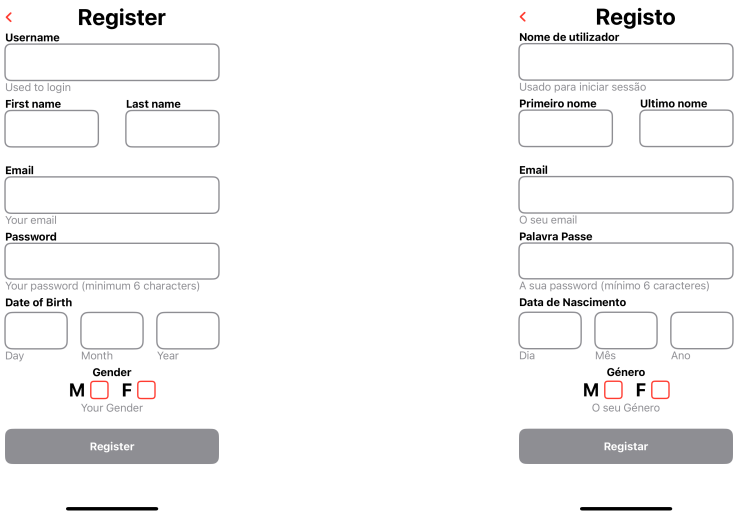

Figura 4.9: Exemplo do suporte para múltiplos idiomas no ecrã de registo de utilizador

4.1.9 Ferramentas utilizadas

Para o desenvolvimento da aplicação iOS, foi utilizado o IDE XCode, sendo o mesmo essencial para a compilação de aplicações iOS. Para o desenvolvimento dos componentes visuais em *Android*, foi utilizado o IDE *Android Studio* e um simulador de dispositivo *Android*.

4.2 Serviço *web*

4.2.1 Arquitetura de desenvolvimento

A primeira etapa no processo de implementação do serviço web consistiu na escolha da arquitetura de desenvolvimento a adotar. Após uma pequena análise, foram identificadas as duas principais utilizadas no desenvolvimento de REST APIs: microsserviços e monólito.

A arquitetura de microsserviços consiste na divisão do sistema num conjunto de serviços, sendo cada um deles responsável por funções distintas, o que resulta numa aplicação modular mais flexível, escalável e testável. Ainda assim, a criação e integração de vários serviços de dimensões é um processo mais complexo, pelo que o tempo inicial de desenvolvimento é por norma mais elevado, sendo esta uma arquitetura por norma mais indicada a equipas de grandes dimensões, onde é possível alocar diferentes membros a diferentes serviços.

A arquitetura de monólito consiste no desenvolvimento do sistema de forma unificada, onde todas as funcionalidades estão implementadas como um serviço único, algo que garante um processo de desenvolvimento inicial mais simples e rápido, dada a baixa complexidade do sistema. Apesar da sua simplicidade, o crescimento do sistema torna a implementação de novas funcionalidades um processo progressivamente mais complexo, tornando esta arquitetura ideal para projetos de menores dimensões.

Considerando que o serviço será desenvolvido simultaneamente com a aplicação móvel e ambos serão apenas desenvolvidos por uma só pessoa, a maior flexibilidade oferecida pela arquitetura de microsserviços não é suficiente para justificar o desenvolvimento de um sistema que leva substancialmente mais tempo a desenvolver, pelo que foi decidido prosseguir com o desenvolvimento de uma arquitetura monólito.

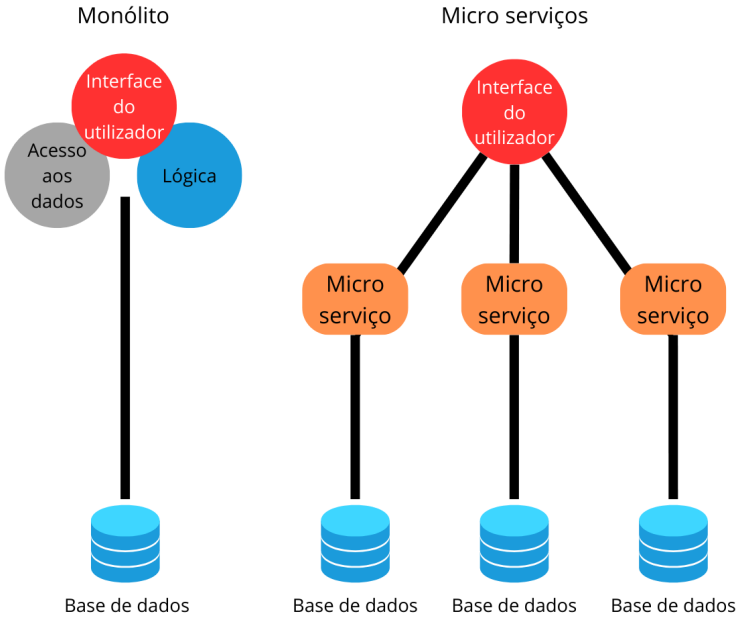

Figura 4.10: Monólito vs Microsserviços

4.2.2 Funcionalidades principais

Perfis

O serviço foi idealizado a pensar nos dois seguintes tipos de utilizador:

- **Paciente**, cujas ações permitem ver, inscrever-se e cancelar a sua inscrição em sessões futuras, participar numa sessão ativa e visualizar o seu histórico cardíaco.
  - O serviço permite a adição de um novo paciente à base de dados caso o nome de utilizador e email passados no corpo do pedido não estejam em uso.
- **Tutor**, cujas ações permitem criar, cancelar, iniciar e terminar sessões.
  - O registo de novos utilizadores com este perfil apenas deve ser efetuado por administradores do sistema.

Autenticação

De forma a criar um sistema de autenticação seguro, foi necessário pensar numa forma para que não existam palavras-passe expostas na base de dados. Na documentação disponível em [Fas25c] é referido que, por defeito, a forma correta de autenticação nos sistemas *FastAPI* é a utilização de algoritmos que criem um *hash* das palavras-passe antes de qualquer inserção na base de dados. Assim, foi utilizada a biblioteca *Argon2*, pois é considerada uma das bibliotecas Python com algoritmos de *hashing* mais seguros. Assim, sempre que um utilizador efetua um registo, é gerado um *hash* da sua palavra-passe, sendo esse o valor

guardado na base de dados, sendo utilizado para comparação sempre que o utilizador efetua uma tentativa de autenticação.

Monitorização cardíaca

Dada a necessidade de atualizar o tutor responsável por uma sessão sempre que um paciente envia os seus dados cardíacos para o serviço, foi utilizado o protocolo de comunicação SSE para criar uma forma de comunicação contínua.

Para perceber as melhores práticas de implementação de protocolos de comunicação SSE, foram analisados alguns protocolos de comunicação deste tipo implementados utilizando *FastAPI*, sendo possível observar que quase a totalidade dos protocolos analisados foi desenvolvida utilizando uma classe da *framework FastAPI*, ***StreamingResponse***, sendo esta responsável pelo envio de dados divididos em blocos, sendo idealmente utilizada para a transferência de ficheiros ou o envio de dados em tempo real.

O controlo do tipo de envio efetuado pela classe *StreamingResponse* é efetuado através de uma variável `media_type`, sendo esta variável responsável pelo cabeçalho `Content-type` do serviço HTTP gerado.

Nos pedidos HTTP comuns, é habitualmente colocado `application/json` no cabeçalho `Content-type`, dada a estrutura chave-valor do JSON ser amplamente suportada por aplicações *front-end*. Ainda assim, este cabeçalho não possui suporte nativo para a reconexão automática após resposta, algo que impossibilita o envio de dados continuamente, tornando-o inutilizável para a implementação do serviço SSE. Dado isto, foi utilizado o `Content-type text/event-stream`, sendo que este ativa um protocolo com suporte a conexão contínua, permitindo assim a um cliente receber dados ao longo do tempo, necessitando apenas de se conectar uma vez.

Ao contrário da comunicação HTTP comum, onde são habitualmente transmitidos dados em formato JSON, num protocolo SSE a informação é transmitida em forma de eventos de texto simples, sendo o formato de cada evento o seguinte:

```
"data: text_to_send\n\n"
```

Dada a natureza simples da informação transmitida, tornou-se essencial pensar numa padronização da informação transmitida pelo protocolo SSE desenvolvido, de modo a simplificar o processo de análise de cada evento recebido na aplicação do tutor.

Para definir um tipo de resposta adequado, foram inicialmente pensados todos os eventos em que um tutor deve ser notificado:

1. Um paciente entra na sessão.
2. Um paciente sai da sessão;
3. Um paciente envia uma medição de FC;
4. Um paciente envia uma medição de VFC.

De seguida, foram identificados todos os parâmetros necessários para que a aplicação do tutor consiga analisar cada um dos eventos pensados descritos acima.

- **Nome de utilizador**, sendo este essencial quem foi o emissor do evento, de todos os utilizadores a participar na sessão;
- **Operação realizada**, de modo a interpretar corretamente os dados recebidos, e evitar erros (como uma medição de FC processada como uma medição de VFC incorretamente).
- **Valor**, que representa o valor da medição efetuada. Este atributo deve ser opcional, dado não ter utilidade nas operações de saída da sessão;
- **Identificador de sessão**, sendo este necessário para que seja permitida a existência de várias sessões ativas em simultâneo, sendo utilizado pelo servidor para reencaminhar cada evento para o respetivo tutor.

Após a identificação dos atributos, foram definidos códigos para cada operação realizada, tendo estes o objetivo de facilitar a interpretação de cada evento.

Código de operação	Evento
`ENTER_SESSION`	Um utilizador entrou na sessão
`LEAVE_SESSION`	Um utilizador saiu da sessão
`HEARTRATE`	Foi recebida uma medição de FC
`HRV`	Foi recebida uma medição de VFC

Tabela 4.1: Códigos de operação para a comunicação SSE

Para a transmissão dos dados para a aplicação do tutor, foi criado um objeto `JSON` que contém os atributos definidos acima, sendo este posteriormente transformado numa `String` e enviado como texto simples, sendo a resposta transmitida ao cliente semelhante à do seguinte exemplo:

```
"data: {"timestamp" = 12345,
"sessionId" = "1",
"username" = "test123",
"event" = "HRV",
"value" = "70"}\n\n"
```

Envio de dados cardíacos

Para o envio de dados cardíacos do paciente durante uma sessão terapêutica, foram criados dois *endpoints* de método `POST`, sendo cada um dedicado a uma das métricas cardíacas suportadas pelo sistema.

O primeiro, `"/heartrate-info"`, é utilizado para o envio de dados relativos à FC, cujo corpo do pedido deve seguir uma estrutura semelhante à seguinte:

```
{
 "sessionId": "1",
 "username": "username123",
 "heartRate": 72,
 "timeStamp": 1698765432
}
```

Quanto ao segundo endpoint, `"/hrv-info"`, é utilizado para o envio de dados relativos à VFC, devendo o corpo do mesmo seguir uma estrutura semelhante à seguinte:

```
{
 "sessionId": "1",
 "username": "username123",
 "hrv": 50,
 "timeStamp": 1698765432
}
```

Para que um envio de dados cardíacos seja válido, são seguidos os seguintes passos:

1. O cliente envia um pedido para o endpoint correspondente à métrica a compartilhar com o sistema;
2. Ao receber o pedido, é validada a inscrição do utilizador na sessão correspondente;
3. Caso o utilizador esteja inscrito na sessão indicada e a mesma se encontre ativa, são os dados cardíacos recebidos são transmitidos ao tutor.
4. Caso uma das condições não se verifique, é retornada uma mensagem de erro.

Sumário de sessão

Para a criação do sumário de uma sessão, foi criado um *endpoint* que pode ser utilizado para enviar um conjunto de dados cardíacos relativos a uma sessão. Para que seja criado um sumário válido, devem ser enviados os seguintes parâmetros no corpo do pedido:

- Nome de utilizador;
- Identificador de sessão;
- Lista de registos de FC;
- Média de VFC obtida durante a sessão.

Após recebida uma requisição neste *endpoint*, são calculados os valores mínimos, médios e máximos da lista de registos da FC e armazenados na base de dados.

Inicialmente, a criação do sumário foi pensada através dos valores da FC e VFC enviados pelos utilizadores ao longo da sessão. Ainda assim, este processo levaria à criação de estruturas de dados mais complexas no serviço, pelo que foi tomada a decisão da criação de um *endpoint* próprio para o efeito.

Sistema de notificações

Foi implementado um sistema de envio de mensagens com o propósito de notificar pacientes nos dois seguintes cenários:

- O cancelamento de uma sessão na qual o paciente esteja inscrito;

- Uma sessão onde o paciente esteja inscrito tenha sido iniciada.

Para a implementação desta funcionalidade foi utilizado o protocolo SMTP (*Simple Mail Transfer Protocol*), permitindo o envio automático de emails sempre que um dos cenários mencionados ocorre. Adicionalmente, a utilização deste protocolo permitiu a adição dos dados da chamada Zoom (ID de reunião e palavra-passe) correspondente à sessão no email enviado.

A utilização de mensagens email como notificações deu-se pelas seguintes razões:

- **Independência da aplicação móvel**. Para além do protocolo SMTP, foram analisadas soluções como notificações *push*, sendo estas vistas como uma alternativa menos flexível, dada a necessidade de configuração adicional do lado do *front-end*;
- **Simplicidade de utilização**. A linguagem *Python* possui bibliotecas de comunicação SMTP de fácil integração com um serviço *FastAPI*, algo que tornou a implementação desta funcionalidade um processo rápido.

Segurança

Para que o sistema funcione de forma segura, principalmente quando são transmitidos dados importantes dos pacientes, como os seus dados pessoais ou registos cardíacos, é essencial que sejam implementadas medidas de segurança, o que fez com que, durante o processo de desenvolvimento do serviço *web* tenham sido analisadas as melhores práticas de segurança em aplicações deste tipo. Uma das questões mais importantes prendeu-se com o protocolo de comunicação a utilizar, sendo que, por defeito, ao executar um serviço num servidor *Uvicorn*, o mesmo utiliza o protocolo HTTP para a comunicação. Apesar de mais eficiente, a comunicação por HTTP não é dotada um elevado nível de segurança, pelo que foi tomada a decisão de utilizar HTTPS para a comunicação, que utiliza certificados SSL (*Secure Socket Layer*) para transmitir os dados criptografados, ao contrário do protocolo HTTP convencional, que transmite as respostas em forma de texto simples [Ama25].

A segunda questão relacionada com a segurança prende-se com a existência de um sistema de autorização robusto, sendo que existe um protocolo de autorização adotado pela generalidade de serviços web, OAuth 2.0, sendo este um protocolo que funciona através de *tokens* de acesso, onde um utilizador só pode ter acesso a certas funcionalidades com um *token* válido, podendo este ter uma validade, ou seja, expirar após uma janela de inatividade [Aut25]. Sendo este o protocolo de autorização recomendado na documentação da *FastAPI*, foi criada uma implementação do mesmo, que funciona da seguinte forma:

1. O serviço possui dois dicionários, sendo o primeiro "*sessionTokens*", onde cada chave corresponde a um nome de utilizador e o respetivo valor ao seu *token* de autorização. O segundo dicionário tem denomina-se "*tokenExpireTime*", onde cada chave corresponde a um nome de utilizador e o respetivo valor ao instante temporal onde a sua chave se torna inválida;
2. Quando um utilizador efetua a autenticação com sucesso, é gerado um *token* de autenticação, sendo mesmo adicionado ao dicionário "*sessionTokens*". Adicionalmente, é adicionado ao dicionário "*tokenExpireTime*" o instante onde o *token* se torna inválido;
3. Sempre que um utilizador executa um pedido ao serviço que necessite de autenticação, é verificado o dicionário onde estão os instantes onde expiram os *tokens*, sendo eliminados todos os já expirados;

4. Sempre que um utilizador faz um pedido com um *token* válido, o mesmo é respondido com sucesso e o instante de tempo onde o este expira é renovado;

5. Sempre que um utilizador faz um pedido para um *endpoint* que necessite de autenticação, sem ter nenhum *token* válido, o serviço responde com um erro explicativo do sucedido.

4.2.3 Desafios

Após desenvolver o módulo de autenticação do serviço, foi pensada na possibilidade de um utilizador se esquecer da sua palavra-passe, algo que poderia não só impedir o utilizador de aceder ao sistema temporariamente, como obrigaria à intervenção de um administrador do sistema para a recuperação do acesso. A ausência temporária de um administrador pode levar a que utilizadores não consigam aceder ao sistema, algo que pode causar frustração, levando ao abandono da plataforma. Assim, tornou-se necessário o desenvolvimento de um processo de recuperação de palavra-passe automático. Os métodos mais comuns de recuperação de palavra-passe incluem a verificação por SMS, autenticação multi-fator e recuperação por email [Uni23]. Dada a já presente o serviço de envio automático de emails, utilizado na notificação de utilizadores para sessões iniciadas ou canceladas, a implementação de um método de recuperação por email possibilita a reutilização de código. Após a escolha do método de recuperação, foi desenvolvido um sistema de recuperação de acesso que funciona da seguinte forma:

1. É enviado um pedido para o *endpoint* "/*send-recovery-email*", cujo corpo do pedido contem o nome de utilizador;

2. É realizada uma busca na base de dados pelo nome de utilizador enviado no corpo do pedido efetuado;

3. Caso o utilizador seja encontrado, é gerado um código numérico de cinco dígitos, e enviado para o email associado no registo;

4. Após enviado o email, o utilizador possui uma janela temporal de quatro minutos para enviar um pedido para o *endpoint* "/change-password", cujo corpo deve conter o nome de utilizador, o código numérico e a nova palavra-passe;

5. É efetuada uma verificação do pedido, com o objetivo de perceber se o código enviado corresponde ao utilizador enviado, bem como se o mesmo ainda está válido;

6. Caso a nova palavra-passe não seja igual à anterior, é gerado um *hash* para a nova palavra-passe, sendo o mesmo guardado no sistema.

A janela temporal de quatro minutos foi desenvolvida através de um dicionário que guarda o instante temporal em que é gerado o código, algo semelhante ao descrito anteriormente nos *tokens* de sessão.

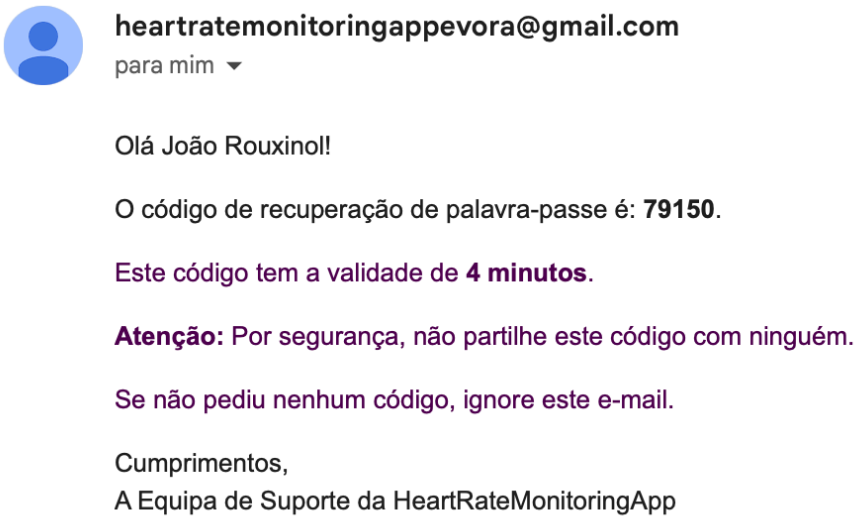

Figura 4.11: Exemplo de um email de recuperação de palavra passe

4.3 Base de dados

4.3.1 Identificação das entidades do sistema

O desenvolvimento da base de dados passou por vários passos, sendo que o primeiro consistiu na análise dos requisitos do sistema para proceder à identificação de cada uma das entidades, o que resultou nas seguintes:

- **Entidade utilizador (*user*)**, que representa um paciente. Esta necessita de conter as informações básicas do utilizador.
- **Entidade professor (*teacher*)**, que representa um profissional de saúde. Esta necessita de informações de autenticação do professor bem como um nome, para que seja identificado pelos pacientes na aplicação.
- **Entidade sessão (*session*)**, que representa uma sessão terapêutica, lecionada por um professor e onde participam utilizadores.

4.3.2 Definição de atributos

Após definidas as entidades da base de dados, foi necessário definir os atributos de cada entidade. Através da análise dos casos de uso definidos no **Capítulo III**, foi possível reunir o seguinte conjunto de atributos:

- **Utilizador:** nome de utilizador, nome, email, data de nascimento, palavra-passe, género.
- **Professor:** nome de utilizador, nome, palavra-passe.
- **Sessão:** identificador de sessão, nome, responsável, descrição, data, hora, número de vagas.

4.3.3 Criação do diagrama E-R

Após a identificação das entidades e atributos, foram definidas as relações e restrições entre os dados, sendo que foram pensadas nas seguintes:

1. Um utilizador pode inscrever-se em zero ou várias sessões.
2. Uma sessão pode ter zero ou vários utilizadores inscritos.
3. Um professor pode estar associado a zero ou várias sessões.
4. Uma sessão necessita sempre de um professor associado.
5. Uma sessão não pode ter mais que um professor associado.

Utilizando o conjunto de relações e restrições definidas acima aliado aos conceitos fundamentais das bases de dados relacionais (chave primária e chave estrangeira), foi possível estruturar o seguinte diagrama de Entidade-Relação (ER):

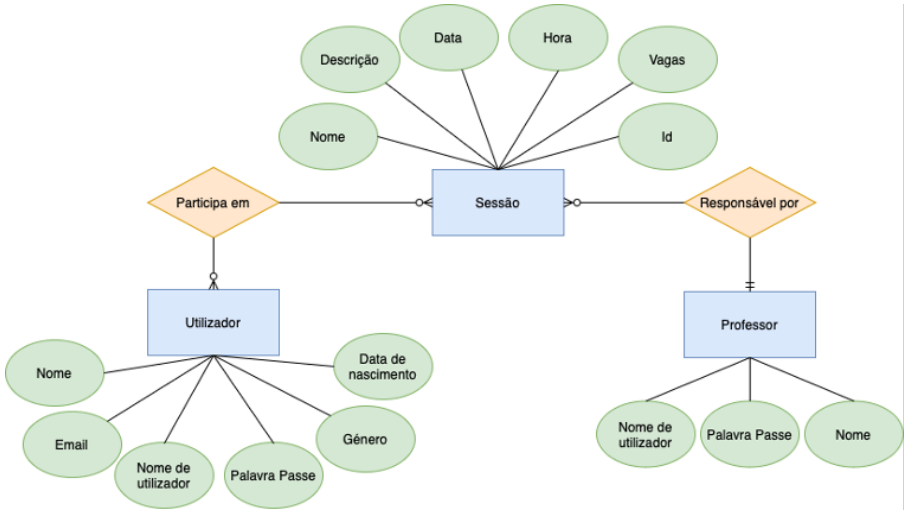

Figura 4.12: Diagrama E-R proposto

4.3.4 Criação da base de dados

O desenvolvimento do diagrama ER permitiu uma melhor perceção das relações existentes entre as entidades, o que permitiu constatar que, para além das entidades, existe a necessidade de criação de duas tabelas adicionais, a primeira, **inscrição em sessão (*sessionSigning*)**, representa uma inscrição numa sessão por parte de um paciente, sendo a segunda **histórico de sessão (*sessionSummary*)**, que representa o resumo dos indicadores cardíacos do paciente numa sessão em que participou. Assim, o desenho final da base de dados irá contar com as seguintes tabelas:

- **Utilizador**, constituída pelos seguintes atributos: nome de utilizador, nome, email, data de nascimento, palavra-passe, género. A chave primária desta tabela é "nome de utilizador".
- **Professor**, constituída pelos seguintes atributos: nome de utilizador, nome, palavra-passe. A chave primária desta tabela é "nome de utilizador".

- **Sessão**, constituída pelos seguintes atributos: id de sessão, nome, responsável, descrição, data, hora, vagas. A chave primária desta tabela é "id de sessão". Existe ainda uma chave estrangeira, correspondente ao atributo "responsável", que referencia o atributo "nome de utilizador" da tabela Professor.
- **Inscrição em sessão**, constituída pelos seguintes atributos: id de sessão, nome de utilizador. A chave primária desta tabela é o conjunto "id de sessão" e "nome de utilizador". Esta tabela conta com duas chaves estrangeiras, a primeira, "id de sessão" referencia a tabela Sessão, sendo a segunda "nome de utilizador", que referencia a tabela Utilizador.
- **Dados Cardíacos de sessão**, constituída pelos seguintes atributos: id de sessão, nome de utilizador, dados da frequência cardíaca (valor mínimo, valor máximo, valor médio e número de medições) e variabilidade da frequência cardíaca. A chave primária desta tabela é o conjunto "id de sessão" e "nome de utilizador". Existem ainda duas chaves estrangeiras, a primeira, "id de sessão" referencia a tabela Sessão, sendo a segunda "nome de utilizador", que referencia a tabela Utilizador.

4.3.5 Normalização da base de dados

Antes de proceder à criação das tabelas ou à escrita de código SQL, foi essencial efetuar o processo de normalização da base de dados, com o intuito de eliminar redundâncias e assegurar a integridade dos dados. Para tal, a base de dados foi normalizada, com o objetivo de efetuar a normalização até à Terceira Forma Normal (3NF).

Na Primeira Forma Normal (1NF), é necessário que todos os atributos de uma tabela consistam em valores atómicos, ou seja, indivisíveis. No caso da base de dados em questão, como não existem listas ou estruturas complexas nos atributos, verificou-se que a mesma já se encontrava em conformidade com a 1NF.

Para a Segunda Forma Normal (2NF), é necessário que todas as tabelas possuam uma chave primária, sendo que os demais atributos devem depender exclusivamente dessa chave. No caso de chaves primárias compostas por múltiplos atributos, todos os outros atributos da tabela devem depender da totalidade da chave primária. Nas tabelas da base de dados em análise, todas as chaves primárias compostas cumprem este requisito, uma vez que os restantes atributos só podem ser obtidos através da totalidade da chave primária. Dessa forma, confirmou-se que a base de dados está em conformidade com a 2NF.

Por fim, para que uma base de dados esteja na Terceira Forma Normal (3NF), é imperativo que não existam dependências transitivas, ou seja, nenhum atributo pode depender de outro que não seja a chave primária. Na base de dados desenvolvida, todos os atributos das tabelas são diretamente dependentes das respetivas chaves primárias, não havendo qualquer caso de dependência transitiva. Assim, conclui-se que a base de dados está efetivamente normalizada até à 3NF.

4.3.6 Criação do diagrama relacional

Após a identificação das tabelas necessárias, bem como a normalização das mesmas, foi possível chegar ao seguinte diagrama relacional:

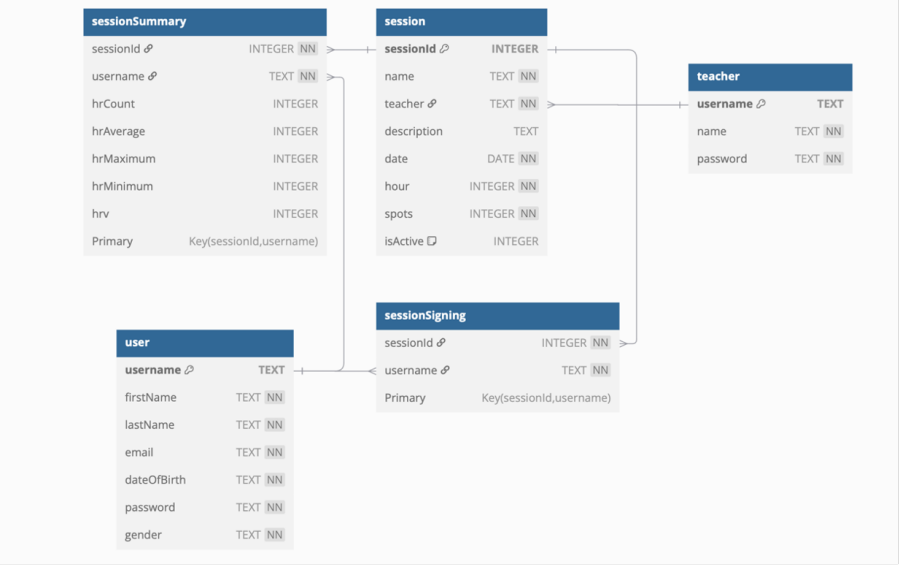

Figura 4.13: Diagrama relacional proposto

Por fim, as tabelas foram adicionadas à base de dados através de comandos SQL, sendo possível consultar o código utilizado no Anexo B.

4.3.7 Desafios

O processo de desenvolvimento da base de dados suscitou diversas questões, sendo quase todas relacionadas com o tipo de dados cardíacos a armazenar. Sendo um dos objetivos a criação de uma funcionalidade de histórico de sessão, onde o utilizador pudesse aceder às estatísticas cardíacas mais relevantes de cada sessão em que participou. Contudo, a definição da estrutura ideal para o armazenamento destes dados revelou-se um desafio. Inicialmente, foi pensada a criação das duas seguintes tabelas para o registo cardíaco:

- **Registo FC**, com os atributos: identificador de sessão, nome de utilizador, instante e valor da frequência cardíaca.
- **Registo VFC**, com os atributos: identificador de sessão, nome de utilizador, instante e valor da variabilidade da frequência cardíaca.

O propósito destas tabelas consistia na execução de inserções nas mesmas sempre que recebida uma medição de FC ou VFC de um utilizador. No entanto, após pensar no funcionamento do sistema durante uma sessão terapêutica, cada utilizador envia uma medição da frequência cardíaca por segundo e uma medição da variabilidade da frequência cardíaca a cada 10 segundos, o que resultaria em 65 operações de escrita por minuto, por utilizador, cenário que, em situações com múltiplos utilizadores, poderia comprometer o desempenho do sistema. Apesar da possibilidade de ter todas as medições efetuadas durante a sessão permitir o desenvolvimento de um ecrã de sumário mais informativo, onde poderiam ser criados gráficos ou outro tipo de insights, a carga excessiva sobre a base de dados tornou esta abordagem inviável.

Para mitigar este problema, exploraram-se alternativas que visassem a inserção de todas as medições apenas no final da sessão em forma de lista, reduzindo significativamente o número de operações de escrita. Contudo, a implementação desta solução enfrentou limitações técnicas, particularmente em *SQLite* e até *MySQL*, que não suportam o armazenamento de listas numéricas de forma nativa (ao contrário de sistemas

como *PostGreSQL*). Uma das alternativas consideradas foi a inserção de listas de números em formato de texto, utilizando caracteres especiais (como vírgulas) como separadores. No entanto, esta abordagem foi rapidamente descartada pelas seguintes razões:

1. Violação da **Primeira Forma Normal (1FN)**, que exige que todos os atributos de uma base de dados normalizada sejam constituídos por valores atómicos.
2. A utilização de texto para armazenar números aumenta desnecessariamente o espaço ocupado na base de dados quando comparadas com valores inteiros, um fator crítico quando se considera que uma sessão de 30 minutos geraria um valor próximo de duas mil medições de FC.

O principal desafio consistiu na identificação de uma forma eficiente de armazenamento que não comprometa a experiência do utilizador. Após análise, concluiu-se que, embora o acesso a todos os registos de FC permita a deteção de variações abruptas, o que pode ser utilizado para a monitorização de condições como arritmias, a observação de valores como a média, o máximo e o mínimo ao longo da sessão seria suficiente para fornecer ao utilizador uma visão geral da sua atividade cardíaca. No caso da VFC, dado que este indicador é idealmente calculado a longo prazo, o valor apresentado no sumário corresponde à média obtida ao longo da sessão.

Esta decisão levou à criação da tabela **Dados Cardíacos de Sessão (*sessionSummary*)**, onde os cálculos do valor mínimo, máximo e médio são realizados na aplicação móvel no final da sessão. Desta forma, é feita apenas uma inserção na base de dados por utilizador por sessão, reduzindo significativamente a quantidade de informação armazenada e garantindo que a base de dados mantenha dimensões mais reduzidas, mesmo quando utilizada por grupos de maiores dimensões. Esta solução equilibra a eficiência do sistema com a utilidade para o utilizador, cumprindo o objetivo principal da funcionalidade de histórico de sessão.

4.3.8 Ferramentas utilizadas

Todo o desenvolvimento da base de dados do sistema foi realizado com o auxílio de apenas uma ferramenta, ***TablePlus***, sendo esta uma interface gráfica (GUI) para gestão de bases de dados. A escolha desta deveu-se ao facto da mesma permitir a execução de *queries* SQL e visualização de tabelas de forma intuitiva, contando adicionalmente com suporte para *SQLite*.

4.4 Distribuição do sistema

A gestão de todo o código-fonte foi realizada através da tecnologia *Git*, selecionada devido à sua natureza *open source* e facilidade de utilização, sendo que foram criados os três seguintes repositórios:

- **"HeartRateMonitoringAPI"**: Contém todos os ficheiros associados ao serviço web, bem como o código *SQL* necessário para a criação das tabelas da base de dados.
- **"HeartRateMonitoringAppiOS"**: Inclui o projeto da aplicação para *iOS*, juntamente com as bibliotecas fornecidas pelo fabricante dos sensores.
- **"HeartRateMonitoringAppAndroid"**: Contém o projeto da aplicação *Android*, embora ainda em fase de desenvolvimento.

Todos os repositórios foram hospedados na plataforma *GitHub* como repositórios públicos, permitindo acesso livre e transparente aos mesmos. Os links para cada repositório estão disponíveis em [Rou25].

4.4.1 Documentação

Dado que o sistema foi distribuído publicamente, a criação de documentação clara e abrangente tornou-se essencial, com o objetivo de facilitar a compreensão do código e fornecer contexto sobre a lógica implementada para todos os utilizadores que acedam aos repositórios do projeto. O processo de documentação consistiu nas seguintes práticas:

- **Comentários no código-fonte**: Nos ficheiros de código-fonte de cada repositório, foram inseridos comentários detalhados, com o intuito de explicar a lógica implementada e simplificar a compreensão do funcionamento do sistema.
- **Ficheiros *README***: Para auxiliar na utilização do sistema, foram criados ficheiros *README* em cada repositório, contendo informações essenciais sobre a execução, estrutura e funcionalidades de cada componente.
- **Documentação *FastAPI***: A utilização da ferramenta *FastAPI* permitiu a geração automática de documentação interativa para o serviço web desenvolvido, tornando a sua utilização mais acessível e eficiente. Adicionalmente, a inclusão de comentários estruturados no código enriqueceu essa documentação, tornando-a mais informativa e completa.

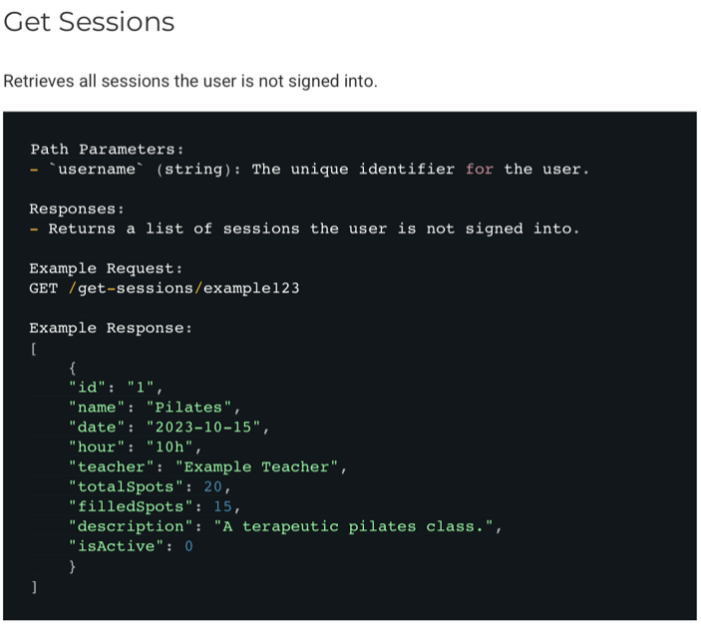

Figura 4.14: Documentação *FastAPI* gerada para um *endpoint* do serviço *web*

4.4.2 Ferramentas utilizadas

A gestão dos repositórios *Git* foi realizada através da interface gráfica *Fork*, que agilizou a gestão de *branches* e facilitou a resolução de conflitos.

5

Resultados

De forma a avaliar o sistema proposto, os principais componentes do mesmo foram submetidos a testes de validação à sua funcionalidade e desempenho, com o objetivo de confirmar se estes estão alinhados com o definido anteriormente.
Para executar todos os testes, foi utilizado o seguinte *hardware*:

- **Computador:** MacBook Pro, 2021;
  - **Processador:** Apple M1;
  - **Memoria RAM:** 16 GB.
- ***Smartphone:*** iPhone 12 Mini;
  - **Processador:** *Apple A14 Bionic*;
  - **Memoria RAM:** 4 GB.
- **Sensor cardíaco:** Suunto Smart Movesense Sensor.

5.1 Testes à monitorização cardíaca

5.1.1 Comparação com dispositivos IoT

De forma a validar as medições de FC e VFC obtidas pelo sensor utilizado no sistema, procedeu-se à comparação dos dados recolhidos com os valores registados por outros dispositivos vestíveis disponíveis no

mercado. Para comparação, foram utilizados três dispositivos diferentes, sendo a escolha baseada no facto de os mesmos representarem três alternativas à monitorização cardíaca disponíveis de forma acessível.

O primeiro dispositivo utilizado para comparação foi o *smartwatch* ***Xiaomi Watch Active S1***, um equipamento que monitoriza a frequência cardíaca através de fotopletismografia, também conhecida por PPG. PPG consiste numa técnica que mede as variações de volume sanguíneo na microcirculação, calculando assim uma estimativa da FC. Dado que este dispositivo não permite a monitorização da VFC, não será possível utilizar o mesmo para validar os dados obtidos pelo sistema em relação a esse parâmetro, pelo que este *smartwatch* foi utilizado apenas para a validação da frequência cardíaca.

O segundo dispositivo utilizado nos testes consiste em um medidor de pressão arterial de pulso, mais especificamente o **Sanitas SBC 15**, sendo este destinado à monitorização de pacientes nas suas habitações. Sendo que este equipamento também não possui suporte para a monitorização da VFC, o mesmo será apenas utilizado para a validação da VFC.

O último dispositivo utilizado foi o ***Apple Watch Series 9***, um *smartwatch* que combina sensores PPG com ECG, permitindo não só a monitorização da FC, como também da VFC. Desta forma, nos testes realizados com o *Apple Watch*, foi possível efetuar comparações a ambos os indicadores cardíacos, proporcionando uma validação a todas as métricas recolhidas pelo sensor utilizado no sistema.

Foram realizados dois testes: no primeiro teste, o utilizador de teste permaneceu imóvel durante 10 minutos, de forma a analisar a capacidade do sistema de monitorização durante o repouso. No segundo teste realizado, que teve como principal objetivo perceber o comportamento do sistema durante a atividade física, o utilizador realizou uma corrida a 8 km/h durante 10 minutos. Dado que os diferentes dispositivos possuem diferentes formas de calcular a FC, foi adotado o seguinte protocolo de monitorização:

- Para os dispositivos que possuem monitorização da FC média durante um período de tempo, como é o caso do *smartwatch Xiaomi*, *Apple watch* e o sistema desenvolvido, foi utilizada a média no final de cada um dos testes.
- No caso do medidor de pressão arterial, o mesmo só apresenta os valores de FC instantâneos relativos à medição efetuada, pelo que foi necessário efetuar três medições de FC intercalares durante os testes, sendo no final calculada a média das medições efetuadas.
- Para a VFC, foi apenas utilizado o valor médio obtido ao longo do período do teste.

Os resultados dos testes foram os seguintes:

Dispositivo utilizado	**FC em repouso**	**FC em atividade**
Apple Watch	68 BPM	108 BPM
Xiaomi S1 Active	64 BPM	102 BPM
Sanitas SBC 15	65 BPM	95 BPM
Sistema proposto	67 BPM	103 BPM

Tabela 5.1: Comparação da FC em repouso e em atividade em relação a outros dispositivos.

Dispositivo utilizado	**VFC em repouso**	**VFC em atividade**
Apple Watch	84 ms	74 ms
Sistema proposto	82 ms	70 ms

Tabela 5.2: Comparação da VFC em repouso e em atividade em relação a outros dispositivos.

Foram de seguida calculadas as diferenças percentuais das medições dos dispositivos em relação às obtidas no sistema:

Dispositivo utilizado	VPR da FC em repouso	VPR FC em atividade
Apple Watch	1.47%	4.63%
Xiaomi S1 Active	4.69%	0.98%
Sanitas SBC 15	3.08%	8.42%

Tabela 5.3: Diferença percentual da FC em relação ao sistema.

Dispositivo utilizado	VPR da VFC em repouso	VPR da VFC em atividade
Apple Watch	2.38%	5.41%

Tabela 5.4: Diferença percentual da VFC em relação ao sistema.

5.1.2 Validação dos dispositivos utilizados

De forma a avaliar a importância dos resultados obtidos na validação do sistema, foi necessário analisar a literatura relativa aos dispositivos utilizados, de forma a perceber se algum deles possui credibilidade científica suficiente para ser considerado um substituto a um equipamento hospitalar.

Em relação ao *smartwatch Xiaomi S1 Active* e ao medidor de tensão arterial *Sanitas SBC 15* não foi possível encontrar qualquer tipo de validação do mesmo na literatura disponível, fazendo com que os mesmos não ofereçam credibilidade suficiente para obter conclusões. Quanto ao *Apple Watch Series 9*, em [KNE+17] é concluído que os valores de FC obtidos pelo *smartwatch* são diretamente afetados pela intensidade do exercício, resultando em valores altamente precisos durante o repouso ou prática ligeira de exercício, sendo esta precisão progressivamente mais baixa à medida que a intensidade do exercício é aumentada. Adicionalmente, em [OLB+24], foi encontrada uma variância de 10 ms nas medições de VFC obtidas pelo *Apple Watch Series 9*, sendo esta mais notória à medida que a intensidade do exercício aumenta.

A análise realizada permite concluir que, independentemente dos resultados obtidos com os dispositivos escolhidos, é importante procurar outras formas de validação do sistema, para que o mesmo obtenha mais credibilidade.

5.1.3 Validação com medição manual

Ainda que os resultados obtidos sejam, na sua generalidade, positivos, em termos científicos, a comparação do sistema com dispositivos sem qualquer tipo de certificação hospitalar não tem qualquer tipo de credibilidade. Assim, tornou-se essencial a execução de algum tipo de validação do sistema em relação a técnicas com algum tipo de validade médica, de modo a aumentar a credibilidade dos valores obtidos. Assim, procedeu-se à comparação dos valores da FC dados pelo sistema com os obtidos através de técnicas de medição manual, sendo a escolha deste método influenciada pelos seguintes fatores:

- **Independência de tecnologia**. Todos os dispositivos utilizados anteriormente recorrem a diferentes tecnologias para efetuar a monitorização cardíaca, podendo existir variações causadas pelos algoritmos de cálculo ou tipo de sensores utilizados.

- **Validação clínica**. A medição manual da frequência cardíaca é um método utilizado em ambiente hospitalar, sendo que o cenário em que os valores obtidos pelo sistema se alinhem com os obtidos manualmente pode ser um forte argumento da validade do sistema para utilização hospitalar.

Os testes realizados seguiram as mesmas diretrizes dos realizados anteriormente: um primeiro teste onde a medição cardíaca é efetuada após um período de dez minutos em repouso e um segundo teste após uma atividade de intensidade moderada durante dez minutos.

Quanto à forma de medição manual utilizada, foram utilizados dois métodos: a palpação da artéria radial no pulso e a artéria carótida no pescoço, sendo estes os métodos recomendados em [LeW24]. O procedimento de cálculo manual foi o seguinte:

1. Foram identificados os pontos de medição para cada um dos métodos;
2. Foi observado o número de batimentos cardíacos nos primeiros quinze segundos;
3. O número de batimentos obtido foi multiplicado por quatro, de modo a obter o número de batimentos por minuto (BPM).

Os resultados obtidos para a monitorização da FC foram os seguintes:

Método utilizado	**VFC em repouso**	**VFC em atividade**
Artéria radial (pulso)	64 BPM	104 BPM
Artéria carótida (pescoço)	64 BPM	104 BPM
Sistema proposto	63 BPM	101 BPM

Tabela 5.5: FC em repouso e em atividade comparativamente com medição manual.

Foram ainda calculadas as diferenças percentuais, obtendo os seguintes resultados:

Método utilizado	**VPR da FC em repouso**	**VPR da FC em atividade**
Artéria radial (pulso)	1.59%	2.97%
Artéria carótida (pescoço)	1.59%	2.97%

Tabela 5.6: Diferença percentual da FC em relação ao sistema comparativamente com medição normal.

Quanto à VFC, a ausência de métodos de cálculo manuais que não envolvam a análise de intervalos RR ou sinais ECG tornaram a sua validação impossível sem a presença de um profissional de saúde.

5.1.4 Análise dos resultados

A comparação dos dados de FC e VFC obtidos pelo sistema com os valores registados por dispositivos disponíveis no mercado permitiu confirmar a precisão e fiabilidade do sistema desenvolvido. Em relação à FC, as diferenças percentuais obtidas revelaram diferenças quase sempre inferiores a 5%, com a única exceção registada na medição do Sanitas SBC 15, durante a atividade física. Contudo, estas diferenças podem ser justificadas tanto pelo facto de este dispositivo exigir o cálculo manual da média como pela menor precisão inerente à medição por oscilómetro, utilizada em dispositivos de monitorização de pressão arterial, especialmente durante a atividade física.

Quando comparado com métodos manuais, considerados mais precisos e com relevância a nível hospitalar, o sistema demonstrou uma concordância ainda mais elevada. As medições por palpação da artéria radial

e carótida apresentaram diferenças abaixo dos 3%, resultados que reforçam a validade e credibilidade do sistema.

No que diz respeito à medição da VFC, não foram identificadas disparidades significativas entre o sistema proposto e o *Apple Watch*, utilizado como referência, com diferenças percentuais gerais inferiores a 6%.

Em suma, os resultados obtidos constituem fortes indícios da viabilidade do sistema em termos de precisão das medições. No entanto, seria importante realizar uma validação mais robusta por parte de uma equipa médica, particularmente no que toca à validação da VFC. Adicionalmente, seria relevante testar o comportamento do sistema com utilizadores portadores de diversas patologias cardíacas em diferentes estágios de desenvolvimento, de modo a compreender o desempenho do sistema nessas condições específicas. Este tipo de validações permitiria consolidar a confiança no sistema, garantindo a sua adequação para aplicação clínica.

5.2 Testes à aplicação móvel

Após terminado o desenvolvimento da aplicação móvel, foi feito um levantamento das melhores métricas de análise de desempenho para aplicações *iOS*. Em [Mon21] é apresentado um conjunto constituído pelas métricas de validação mais comuns. De entre as apresentadas, foram selecionadas as que foram consideradas mais apropriadas para a validação do projeto no estado atual.

5.2.1 Tempo de inicialização da aplicação

O tempo de inicialização é um aspeto crucial no que toca à experiência do utilizador, dado este ser o primeiro contacto do mesmo com a aplicação, sendo que um tempo de inicialização demasiado longo pode levar à redução da taxa de aceitação. Em iOS, são considerados três tipos de inicialização:

- ***Cold launch***, que ocorre quando a aplicação não está carregada na memória;
- ***Warm launch***, que ocorre quando a execução da aplicação foi terminada recentemente, estando ainda parcialmente carregada na memória;
- ***Resume***, que ocorre quando a aplicação está suspensa e completamente carregada na memória.

Para obter o tempo necessário para cada um dos tipos de inicialização de forma precisa, foram desenvolvidos testes automáticos para cada um dos cenários. Em cada um dos cenários, foram executadas dez iterações de inicialização da aplicação, sendo o valor de tempo obtido o valor da média temporal de todos os testes. Os resultados obtidos foram os seguintes:

Teste realizado	**Média de tempo**
Cold launch	2.423 s
Warm launch	1.715 s
Resume	0.094 s

Tabela 5.7: Resultados dos testes de inicialização da aplicação.

Os de automação descritos foram desenvolvidos na linguagem *Swift*, utilizando *XCTest*, a biblioteca oficial da *Apple* para o desenvolvimento de testes unitários, de UI e automáticos a aplicações iOS, sendo possível aceder ao código-fonte dos mesmos no Anexo C.

5.2.2 Testes de utilização de memória

Nos *smartphones* atuais, a memória disponível é habitualmente partilhada entre vários processos executados em background, tais como operações do sistema operativo ou aplicações suspensas, o que a torna um recurso limitado, cuja gestão deve ser pensada. Com isto, foi necessário efetuar uma avaliação à quantidade de memória utilizada pela aplicação em vários cenários de utilização diferentes, sendo os cenários pensados os seguintes:

- **Utilização base**, ou seja, a memória utilizada pela aplicação num ecrã simples, sem que estejam a ser executadas operações em paralelo, como comunicação com o sensor ou o serviço web. Este cenário serve principalmente como ponto de comparação com os seguintes, para obter o impacto de cada operação na utilização de memória;
- **Ecrã de sessão do paciente**, dado que a obtenção de dados cardíacos leva a várias mudanças visuais no ecrã, algo que pode causar um aumento significativo na utilização da memória.
- **Ecrã de sessão do tutor**, onde cada novo aluno a participar numa sessão origina uma nova vista, algo que pode influenciar a utilização de memória. Esta análise é especialmente importante para identificar possíveis limitações no número de utilizadores por sessão.

Para efetuar os testes, foram utilizadas as ferramentas de análise de desempenho disponíveis no *XCode*, sendo estas capazes de fornecer informações importantes não só sobre a utilização de memória, como também sobre o impacto da aplicação na memória disponível. Os resultados obtidos foram os seguintes:

Utilização base

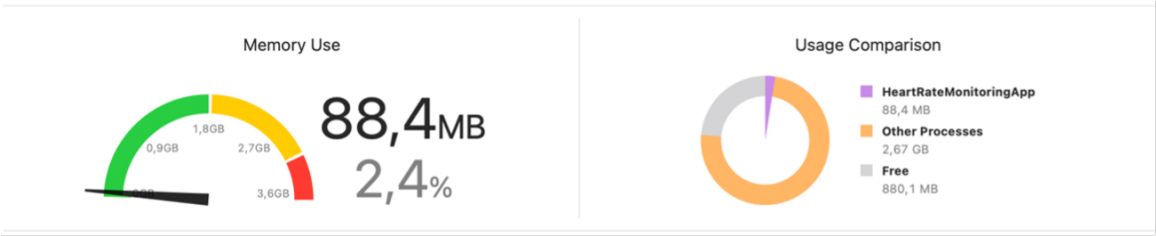

Figura 5.1: Análise da utilização base da memória no sistema

Ecrã de sessão do paciente

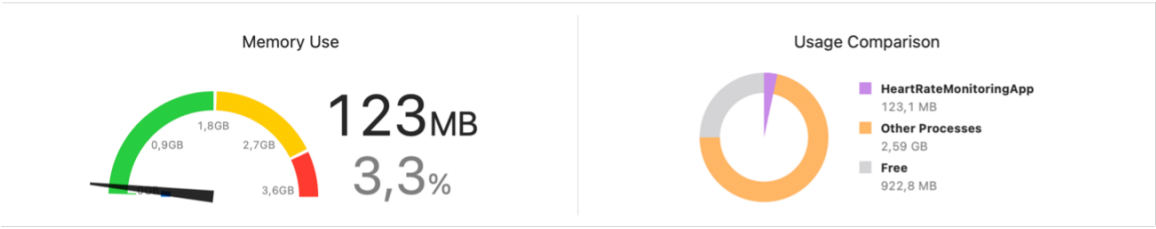

Figura 5.2: Análise da utilização de memória no ecrã de sessão do paciente

Ecrã de sessão do tutor

Para testar o impacto do número de participantes numa sessão no consumo de memória, foram testados dois cenários distintos. No primeiro, Dado que existe a hipótese da influência do número de pacientes no consumo de memória, foram testadas duas situações distintas:

1. **Sessão com zero pacientes:**

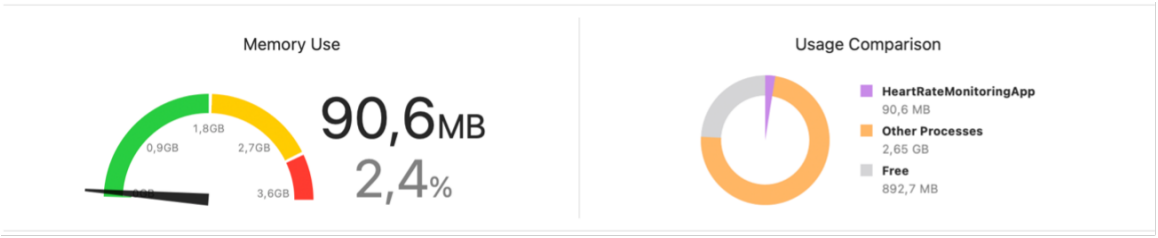

Figura 5.3: Análise da utilização de memória no ecrã de sessão do tutor com zero pacientes

2. **Sessão com trinta pacientes:**

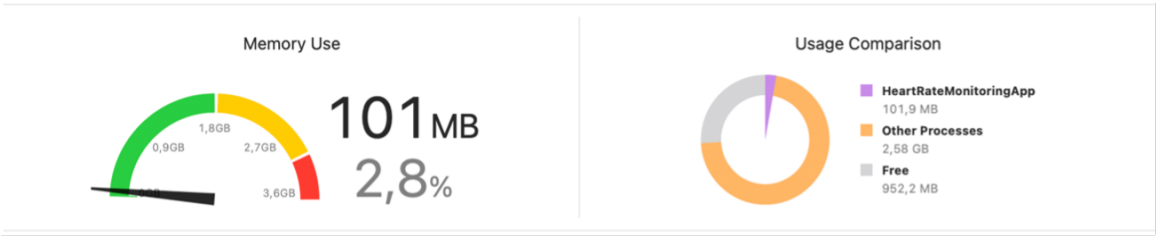

Figura 5.4: Análise da utilização de memória no ecrã de sessão do tutor com trinta pacientes

5.2.3 Testes de utilização da rede

Por último, foram analisadas as métricas relativas à utilização de rede, associadas à comunicação com o serviço *web*, de modo a perceber o consumo de dados correspondente à utilização da aplicação por um paciente. Para este teste, foi simulada uma sessão cardíaca de alguns minutos, onde foram obtidas e analisadas as taxas médias de transmissão e recepção de dados por segundo, sendo possível obter as seguintes informações:

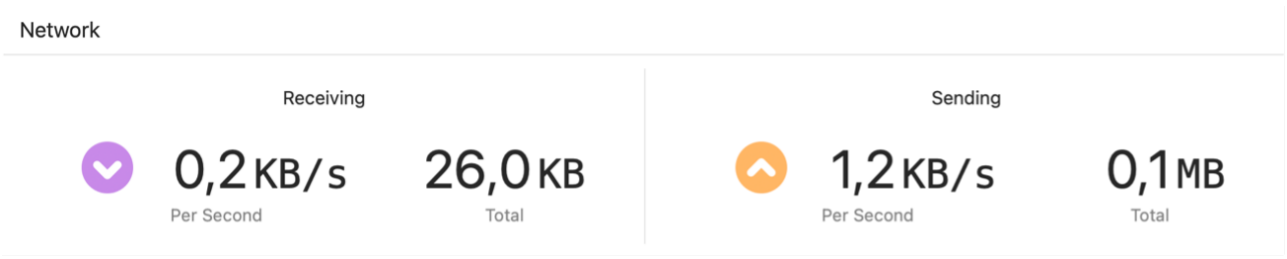

Figura 5.5: Análise da utilização de memória no ecrã de sessão do tutor com trinta pacientes

5.2.4 Análise de resultados

Na situações testadas, a aplicação desenvolvida obteve resultados geralmente positivos, principalmente no que toca à eficiência da mesma. Nos testes efetuados à inicialização da aplicação, foram verificados tempos médios de *resume* muito baixos, ainda assim, a obtenção de um tempo médio de *cold launch* acima dos dois segundos pode causar frustração ao utilizador, sendo a principal causa deste resultado o facto da biblioteca *Movesense* utilizada para a conexão com os sensores ser bastante pesada. Quanto à utilização de memória, foi possível observar percentagens de utilização sempre abaixo dos 5%, valor que demonstra uma aplicação na sua generalidade eficiente e capaz de funcionar com poucos recursos, sendo que o cenário onde foi observada maior utilização de memória foi o ecrã de sessão do paciente, algo facilmente justificável pelo número de operações efetuadas em simultâneo, como a recolha de dados cardíacos, processamento de intervalos RR, envio de dados para o serviço web e atualização dos componentes visuais. Por fim, os valores médios obtidos nos testes à transmissão de dados na rede demonstram uma elevada eficiência no que toca à comunicação, permitindo a utilização do sistema em cenários onde a velocidade da rede não seja elevada.

5.3 Testes ao serviço *web*

Para que o sistema consiga monitorizar continuamente dezenas de pacientes, é necessário que o serviço *web* desenvolvido seja capaz de responder a um número elevado de pedidos em simultâneo. Para tentar perceber e delinear os limites do sistema no que toca à capacidade de resposta a diferentes volumes de tráfego, foram desenvolvidos dois diferentes tipos de testes: **testes de concorrência**, onde são executados vários pedidos em simultâneo para perceber até onde é que o serviço consegue responder com sucesso a todos. Adicionalmente, foram ainda desenvolvidos **testes de desempenho**, com o objetivo de calcular o tempo total de execução de todos os pedidos e o tempo médio por pedido. Para que seja possível perceber o quão escalável é o serviço desenvolvido, foram efetuados testes com respostas de diferentes dimensões.

5.3.1 Condições de teste

- **Endpoint utilizado: *"/get-sessions/username"***. Esta escolha baseia-se no facto de a resposta a este pedido ser uma lista de elementos, sendo assim possível forçar uma resposta com vários elementos e perceber o comportamento do sistema nesse cenário.
- **Testes com diferentes números de pedidos em simultâneo**. Foram escolhidos três casos distintos:
  - 50 pedidos em simultâneo, para simular uma situação normal de utilização;
  - 100 pedidos em simultâneo, para simular uma situação de alto tráfego;
  - 500 pedidos em simultâneo, para simular uma situação irrealista, onde o objetivo é perceber o comportamento do sistema em casos extremos.
- **Testes com respostas de diferentes dimensões**. Sendo a resposta dos testes uma lista, foram testados cenários com listas de diferentes dimensões, para testar a influência do tamanho da resposta no tempo de resposta do serviço. Os três casos testados foram os seguintes:
  - Uma lista vazia, que corresponde a uma resposta de dimensão 2 bytes.
  - Uma lista com dez elementos, que corresponde a uma resposta de dimensão de aproximadamente 1600 bytes.

– Uma lista com cem elementos, que corresponde a uma resposta de dimensão de aproximadamente 16000 bytes.

Quanto aos testes desenvolvidos, foi desenvolvido um *script* na linguagem *Python* que simula a execução de vários pedidos em simultâneo para o endereço escolhido. Após retornada uma resposta a todos os pedidos, são calculadas métricas como o tempo total de execução, o tempo médio de execução, o número de pedidos com sucesso e o número de pedidos falhados. O código dos testes está disponível no Anexo C. Os resultados obtidos foram os seguintes:

Primeiro teste:

Para o primeiro teste realizado, foi simulada uma resposta do servidor constituída por uma lista de zero elementos. Os resultados obtidos foram os seguintes:

Número de pedidos	Pedidos com sucesso	Pedidos falhados	Taxa de sucesso	Tempo total	Tempo médio por pedido
50	50	0	100%	0.13 s	0.10 s
100	100	0	100%	0.33 s	0.19 s
500	248	252	49.6%	2.53 s	1.31 s

Tabela 5.8: Resultados dos testes ao serviço *web* para uma lista de resposta com zero elementos.

Segundo teste:

Para o segundo teste realizado, foi simulada uma resposta do servidor constituída por uma lista de dez elementos. Os resultados obtidos foram os seguintes:

Número de pedidos	Pedidos com sucesso	Pedidos falhados	Taxa de sucesso	Tempo total	Tempo médio por pedido
50	50	0	100%	0.30 s	0.24 s
100	100	0	100%	0.54 s	0.38 s
500	244	256	48.8%	5.22 s	1.03 s

Tabela 5.9: Resultados dos testes ao serviço *web* para uma lista de resposta com dez elementos.

Terceiro teste:

Para o terceiro teste realizado, foi simulada uma resposta do servidor constituída por uma lista de cem elementos. Os resultados obtidos foram os seguintes:

Número de pedidos	Pedidos com sucesso	Pedidos falhados	Taxa de sucesso	Tempo total	Tempo médio por pedido
50	50	0	100%	1.35 s	1.07 s
100	100	0	100%	2.82 s	1.78 s
500	226	274	45.2%	5.23 s	2.02 s

Tabela 5.10: Resultados dos testes ao serviço *web* para uma lista de resposta com cem elementos.

5.3.2 Análise dos resultados

Os testes de desempenho e de stress realizados no servidor demonstraram resultados positivos para cargas de trabalho normais e acima da média, especificamente para 50 e 100 pedidos em simultâneo. Nestes cenários, todos os pedidos foram concluídos com sucesso, atingindo uma taxa de sucesso de 100%. Contudo, ao submeter o sistema a uma carga extremamente elevada — 500 pedidos em simultâneo —, observou-se uma taxa de sucesso ligeiramente abaixo dos 50%, indicando que o sistema apresenta limitações sob condições de stress intenso. A variação do tamanho da resposta resultou numa influência marginal no número de pedidos falhados no cenário de 500 requisições simultâneas, com uma variação inferior a 5%. Embora seja plausível que um aumento exponencial no tamanho da resposta possa levar a uma redução mais significativa na taxa de sucesso, tais cenários são considerados improváveis e fora do âmbito de utilização prática do sistema. Portanto, esta variável não foi considerada crítica para o desempenho geral. No que diz respeito aos testes de desempenho, o sistema demonstrou-se robusto para os casos de uso reais, respondendo a todos os pedidos em menos de 1 segundo para cenários de até 100 requisições simultâneas e respostas com listas de até 10 elementos. No entanto, observou-se um aumento significativo no tempo de resposta à medida que o tamanho da resposta cresce. Por exemplo, o tempo de resposta mais que duplicou para o mesmo número de pedidos em simultâneo quando o tamanho da lista aumentou. Este comportamento sugere que o tempo de processamento e transmissão de dados é diretamente influenciado pelo volume de informação retornada. No contexto de escalabilidade, o sistema mostrou potencial para lidar com cargas maiores, embora com um aumento proporcional no tempo de resposta. Por exemplo, no cenário de respostas com listas de 100 elementos e 100 pedidos em simultâneo, o tempo médio de resposta foi de aproximadamente dois segundos, mas todos os pedidos foram atendidos com sucesso. Este resultado indica que, com melhorias de infraestrutura, como a utilização de um servidor dedicado com maior capacidade de processamento e memória, o desempenho do sistema poderá ser otimizado para suportar cargas mais elevadas, como as 500 requisições em simultâneo, com uma taxa de sucesso mais elevada.

5.3.3 Introdução de melhorias

Para cumprir o objetivo proposto no Capítulo I, onde foi definida a meta da criação de um sistema altamente escalável, torna-se essencial a melhoria do desempenho do serviço web em situações onde a taxa de sucesso foi menor que 50%. Em [Tom24] são apresentadas algumas ações possíveis para a melhoria do desempenho de serviços baseados em *FastAPI*, destacando-se a utilização e otimização do servidor ASGI (no caso do sistema utilizado, *Uvicorn*). Uma das possíveis melhorias propostas consiste no aumento do número de *workers*. Um *worker* consiste em um processo independente que responde a requisições de forma paralela, sendo que a utilização de vários *workers* resulta na capacidade de resposta a um maior pedido em simultâneo, ponto importante para o sistema, dada a sua queda de desempenho ser mais notada neste tipo de situação. Por defeito, a execução padrão de uma aplicação *FastAPI* resulta na criação de um serviço que funciona com apenas um *worker*, o que, mesmo que seja suficiente para situações de desenvolvimento, pode tornar-se incapaz de responder ao tráfego normal de um sistema em contexto real. Em [Gun25], é indicado

que a maioria dos cenários são exequíveis com um conjunto de entre quatro e doze *workers*, sendo que a quantidade recomendada depende do hardware utilizado, e pode ser obtida através da seguinte fórmula:

$$\textit{Número de Workers} = 2 \times \textit{Número de Núcleos do Processador} + 1$$

Dado que o computador utilizado para executar o servidor é constituído por *Macbook Pro* de 2021, sendo este equipado com um processador M1 que, de acordo com as especificações fornecidas pelo fabricante em [App25], é um processador de oito núcleos, sendo quatro deles núcleos de alto desempenho e os quatro restantes para eficiência. Através da fórmula acima, é obtido um valor recomendado de dezassete *workers*, contudo, é importante realçar que, apesar de ser um processador de oito núcleos, os quatro núcleos de eficiência possuem desempenho significativamente inferior aos restantes, pelo que a utilização do valor obtido através do cálculo pode não ser o mais correto. Assim, foi tomada a decisão de utilizar apenas doze *workers*, sendo este o valor máximo necessário para responder com sucesso a milhares de pedidos em simultâneo, segundo [Gun25]. A adição de *workers* ao serviço é bastante simples, sendo apenas necessário adicionar o parâmetro `"-workers x"`, onde `"x"` corresponde ao número de *workers*, ao comando anteriormente utilizado para executar o serviço, resultando no seguinte:

```
"uvicorn yourAppName:app --host 0.0.0.0 --port 8000 --workers 12"
```

5.3.4 Repetição de testes

Após efetuadas as alterações necessárias para executar o servidor com um maior número de *workers*, foram repetidos os testes realizados inicialmente, com o objetivo de analisar o impacto da alteração no desempenho do sistema. Os resultados obtidos foram os seguintes:

Primeiro teste:

Número de pedidos	Pedidos com sucesso	Pedidos falhados	Taxa de sucesso	Tempo total	Tempo médio por pedido
50	50	0	100%	0.08 s	0.05 s
100	100	0	100%	0.35 s	0.12 s
500	248	252	49.6%	2.38 s	1.21 s

Tabela 5.11: Resultados da repetição dos testes para uma lista de resposta com zero elementos.

Segundo teste:

Número de pedidos	Pedidos com sucesso	Pedidos falhados	Taxa de sucesso	Tempo total	Tempo médio por pedido
50	50	0	100%	0.21 s	0.16 s
100	100	0	100%	0.39 s	0.24 s
500	248	252	49.6%	3.32 s	1.60 s

Tabela 5.12: Resultados da repetição dos testes para uma lista de resposta com dez elementos.

Terceiro teste:

Número de pedidos	Pedidos com sucesso	Pedidos falhados	Taxa de sucesso	Tempo total	Tempo médio por pedido
50	50	0	100%	0.66 s	0.49 s
100	100	0	100%	1.37 s	1.08 s
500	248	252	49.6%	5.13 s	3.12 s

Tabela 5.13: Resultados da repetição dos testes para uma lista de resposta com cem elementos.

5.3.5 Resultados finais

Comparando os resultados obtidos após a adição de mais *workers*, no que toca à redução de pedidos falhados nos casos de concorrência mais elevada, foram notórias melhorias no terceiro cenário (onde são simulados pedidos de resposta de grandes dimensões), verificando-se uma redução de mais de vinte pedidos falhados. Ainda assim, foi no tempo de resposta onde foram registadas melhorias mais notórias, verificando-se em todos os testes reduções no tempo médio e total de resposta para pelo menos metade do observado nos testes iniciais, o que se torna especialmente evidente nos cenários de resposta de maiores dimensões (onde a redução do tempo total de resposta ultrapassou a marca de um segundo, em alguns casos). Estes resultados são um forte indicador de que a utilização de um número mais elevado de *workers* pode conceder ao sistema a capacidade de dar resposta a um número de utilizadores muito elevado de forma mais rápida, principalmente quando necessário transmitir respostas de grandes dimensões.

5.4 Testes à base de dados

Como referido na secção relativa às tecnologias e protocolos utilizados, a escolha de *SQLite* em detrimento de *MySQL* deu-se principalmente pelo facto de a mesma ser de implementação mais simples, o que agilizou o processo de desenvolvimento. Ainda assim, esta decisão poderia impactar negativamente o sistema, dado que existem restrições de escrita unitária, ou seja, sempre que existam duas ou mais operações de escrita em simultâneo, é criada uma fila para que cada operação seja executada individualmente. Para testar o impacto destas restrições no desempenho do sistema, foram desenvolvidos e executados alguns testes na base de dados, de forma a perceber o tempo necessário para executar um grande número de operações em simultâneo. Os testes foram desenvolvidos em *Python* através da biblioteca *sqlite3*, sendo que os resultados foram tratados e transformados num histograma através da biblioteca *matplotlib*. Os resultados foram os seguintes:

5.4.1 Primeiro teste:

Neste teste foram efetuadas mil operações de escrita em simultâneo, obtendo os seguintes resultados:

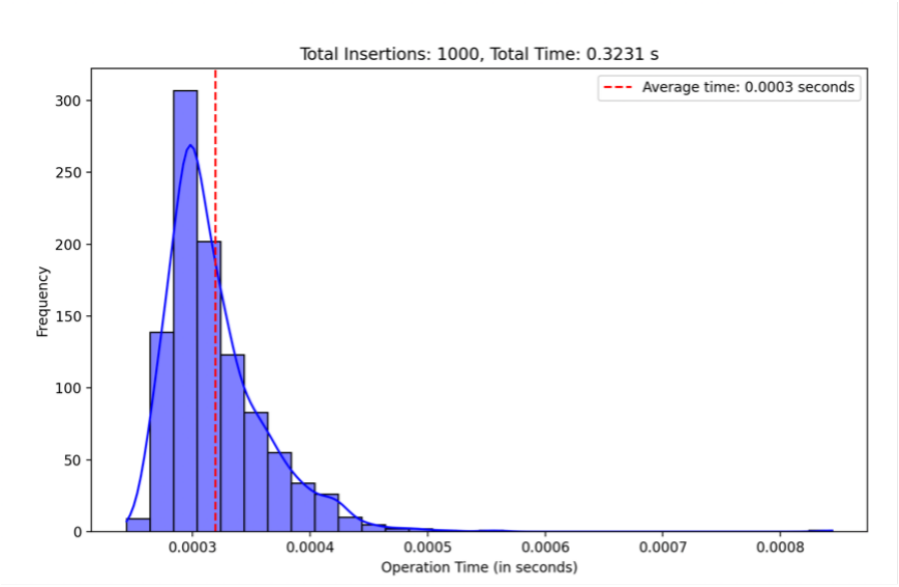

Figura 5.6: Análise do tempo necessário para efetuar 1000 operações de escrita na BD

5.4.2 Segundo teste:

Neste teste foram efetuadas dez mil operações de escrita em simultâneo, obtendo os seguintes resultados:

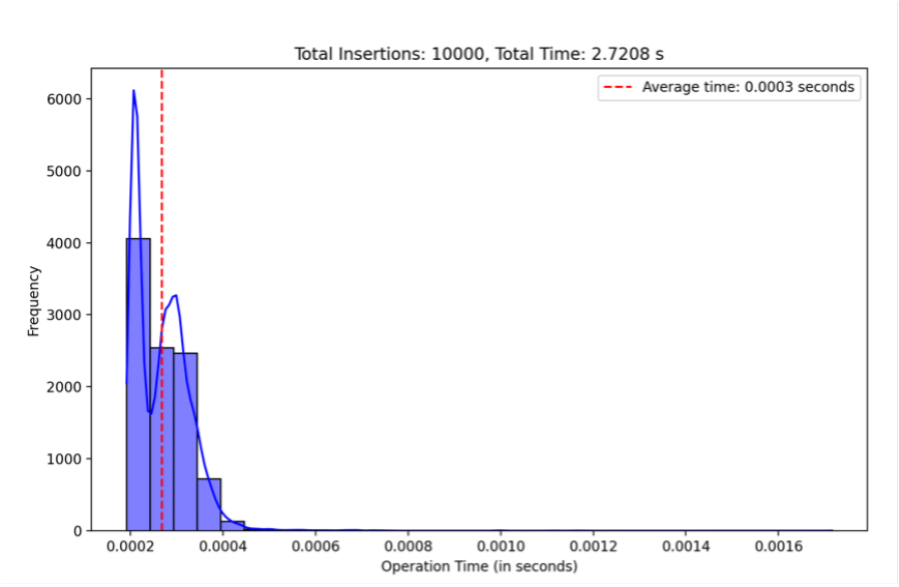

Figura 5.7: Análise do tempo necessário para efetuar 10000 operações de escrita na BD

5.4.3 Terceiro teste:

Neste teste foram efetuadas mil operações de leitura em simultâneo, obtendo os seguintes resultados:

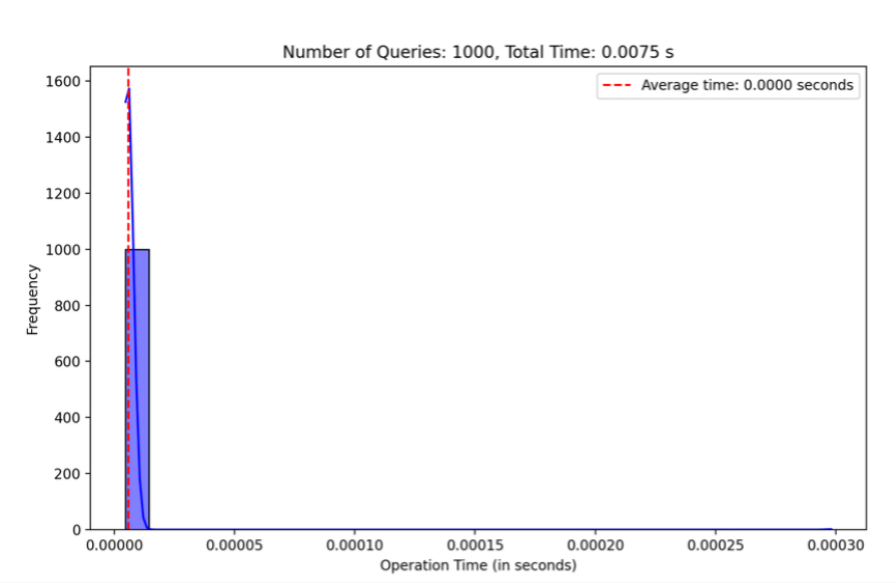

Figura 5.8: Análise do tempo necessário para efetuar 1000 operações de leitura na BD

5.4.4 Quarto teste:

Neste teste foram efetuadas dez mil operações de leitura em simultâneo, obtendo os seguintes resultados:

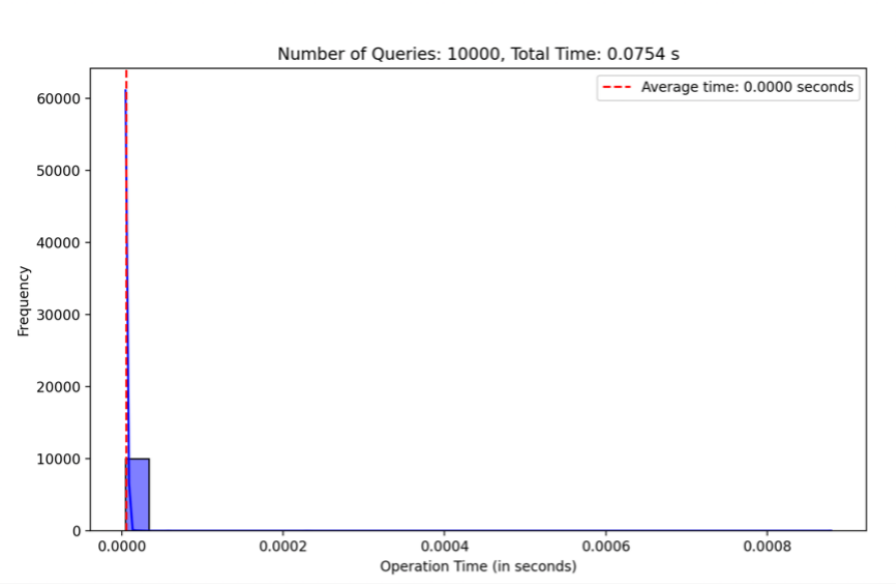

Figura 5.9: Análise do tempo necessário para efetuar 10000 operações de leitura na BD

Os resultados obtidos em todos os testes podem ser analisados na tabela abaixo:

5.4.5 Análise de resultados

Através da média de tempo obtida nos testes efetuados, é possível constatar que, em ambos os tipos de operações (inserção e escrita), são necessárias milhares de operações em simultâneo para que exista

Tipo de operação	Número de *queries*	Tempo total	Tempo médio
Inserção	1000	0.3231 s	0.0003 s
Inserção	10000	2.7208 s	0.0003 s
Consulta	1000	0.0075 s	<0.0001 s
Consulta	10000	0.0754 s	<0.0001 s

Tabela 5.14: Resumo dos resultados obtidos nos testes à base de dados.

um atraso de mais de um segundo. No caso específico do sistema proposto, em utilização normal, não são previstos casos onde existam centenas de operações em simultâneo, sendo que qualquer quebra de desempenho causada não será notada pelos utilizadores. Com base em tudo isto, a utilização de *SQLite* neste sistema não causa impactos significativos de desempenho no mesmo, quando este é utilizado em situações normais.

5.5 Sumário

Os testes efetuados permitiram validar os vários componentes criados, algo útil não só para justificar as escolhas efetuadas no Capítulo 3, como também para identificar possíveis más decisões ou abordagens durante o processo de desenvolvimento. Em relação à monitorização de FC e VFC, os valores obtidos enquadraram-se com as medições efetuadas nos dispositivos de teste utilizados. Ainda que isto seja um ponto positivo, os valores obtidos carecem de uma melhor validação, nomeadamente no que toca à VFC, dada a aplicação de algoritmos de pré-processamento e cálculo desenvolvidos sem a consulta de uma equipa médica. Quanto à aplicação desenvolvida, ainda que esta seja muito eficiente em recursos durante a sua inicialização, os resultados referentes ao tempo necessário para a inicialização da aplicação revelam potencial para melhoria, principalmente no que toca ao *cold launch*, sendo um dos possíveis pontos de ação o desenvolvimento de uma biblioteca para a comunicação com o sensor baseada na disponibilizada pelo fabricante, eliminando todas as funcionalidades não necessárias, de modo a tornar a mesma mais leve. Em relação ao serviço *web*, apesar de existirem cenários onde foram obtidas taxas de insucesso superiores a 50%, os mesmos são simulações de casos impensáveis para o sistema projetado, pelo que não devem ser considerados como problemáticos. Por fim, apesar das restrições de escrita síncrona existentes em *SQLite*, os testes realizados demonstram que o seu desempenho é suficiente para o sistema proposto, não sendo justificada uma mudança para outro SGBD.

6

Conclusão

Este capítulo tem como objetivo apresentar uma análise a todo o trabalho efetuado, tecendo algumas considerações sobre o mesmo. Adicionalmente, são apresentados alguns pontos de trabalho futuro.

6.1 Considerações gerais

O trabalho apresentado descreve o processo de desenvolvimento de um sistema de monitorização de pacientes cardíacos à distância, com a finalidade de ser utilizado durante a participação em sessões terapêuticas, sendo o mesmo projetado para permitir a recolha, transmissão e análise em tempo real de dois indicadores cardíacos, FC e VFC, considerados fulcrais para a avaliação da condição cardiovascular de um indivíduo.

Foi desenvolvida uma aplicação móvel *iOS* utilizando *Swift*, garantindo operabilidade intuitiva tanto para pacientes como para tutores, sendo que, após desenvolvida, foi explorada a ferramenta *Skip* com o objetivo de efetuar a adaptação da aplicação para sistemas *Android*, processo este que se mostrou difícil, dada a incompatibilidade da ferramenta com várias tipologias da linguagem *Swift*, ainda assim, foi possível utilizar o *Skip* para a conversão dos componentes visuais, algo que facilitará o processo de conversão completa da aplicação para *Android* no futuro.

Quanto à comunicação entre tutor e paciente, foi desenvolvida uma *API* utilizando *FastAPI*, sendo esta capaz de comunicação *REST*, responsável pela quase totalidade das comunicações do sistema e *SSE*, que permite a comunicação contínua de forma simples e leve, utilizada para o envio dos dados cardíacos dos pacientes para o tutor durante uma sessão.

Quanto à base de dados, foi utilizado *SQLite*, sendo o principal motivo de escolha da mesma a sua simplicidade e leveza, bem como a ausência de necessidade de um servidor dedicado à base de dados.

Apesar da ausência de testes com grupos de pacientes reais, foram realizados testes aos vários componentes do sistema isoladamente, de forma a comprovar a possibilidade de utilização dos mesmos.

A aplicação desenvolvida mostrou-se altamente eficiente em termos de recursos, mantendo o seu consumo abaixo dos 5% do dispositivo utilizado em todas as situações testadas, inclusivamente em interfaces complexas e com vários elementos, como a monitorização de trinta pacientes em simultâneo. Estes resultados são importantíssimos pois demonstram não só o baixo impacto da mesma no consumo total de memória, como também validam a capacidade de monitorizar grupos com vários pacientes.

Quanto ao serviço web desenvolvido, apesar de observadas taxas de sucesso algo reduzidas para elevados números de pedidos em simultâneo (cerca de 500), a probabilidade de uma situação semelhante acontecer num contexto real é quase nula. Para números de pedidos mais realistas, o sistema conseguiu não só responder a todos com sucesso, como também apresentar velocidades de resposta altamente satisfatórias. Após a introdução de mais *workers*, os tempos de resposta melhoraram significativamente, algo que indicia que as principais limitações do serviço estão no *hardware* em que foi executado.

Em relação ao sistema de base de dados utilizado, surgiram inicialmente algumas dúvidas em relação à utilização do mesmo, principalmente dado às suas limitações nas operações de escrita em simultâneo. Ainda assim, a validação realizada permitiu constatar que não existiu degradação significativa do desempenho mesmo quando são executadas milhares de operações de escrita em simultâneo.

Por fim, de modo a validar os indicadores cardíacos obtidos no sistema, foi efetuada uma comparação dos mesmos com os obtidos através de outros dispositivos IoT, observando-se diferenças percentuais bastante baixas, algo que pode ser interpretado como um bom indicador da precisão do sistema. Ainda assim, dado que nenhum dos dispositivos IoT utilizados possui qualquer tipo de validação formal para uso clínico, foi realizada uma comparação da FC do sistema com valores obtidos através de medições manuais, onde foi possível constatar valores de FC semelhantes.

Assim, é possível afirmar que o sistema demonstra potencial para ser utilizado como alternativa ao acompanhamento de pacientes cardíacos de forma segura, intuitiva e remota, cumprindo com os objetivos propostos inicialmente.

6.2 Trabalho futuro

Apesar do sistema desenvolvido se mostrar capaz de cumprir com todos os objetivos propostos inicialmente, existem vários aspetos a melhorar de forma a que o sistema se possa considerar um produto pronto para utilização em larga escala. Os principais pontos de ação futura são os seguintes:

6.2.1 Implementação de Acessibilidade

O sistema foi concebido com o propósito de ser utilizado por pacientes cardíacos, um grupo que, devido à natureza crónica e progressiva da patologia, é constituído por uma percentagem significativa de indivíduos idosos. No contexto atual, onde a acessibilidade assume um papel de cada vez maior destaque, reforçado por diretrizes internacionais como as *Web Content Accessibility Guidelines* (WCAG), torna-se importante garantir que o sistema pode ser utilizado por qualquer tipo de utilizador, independentemente do mesmo possuir algum tipo de incapacidade. Neste sentido, propõe-se a realização de um estudo exaustivo de boas práticas no desenvolvimento de interfaces acessíveis, com o intuito de integrar funcionalidades que promovam a inclusão, com o objetivo final de desenvolver uma aplicação que se enquadre dentro das normas WCAG 2 [Web25], alinhando o sistema com os padrões globais de acessibilidade e contribuindo para um

dos objetivos iniciais, a democratização da saúde.

6.2.2 Desenvolvimento de aplicação *Android* nativa

Durante o desenvolvimento do projeto, explorou-se a adaptação da aplicação *iOS* nativa para a plataforma *Android*, recorrendo a uma solução externa, o *Skip tools*. Contudo, os resultados obtidos revelaram limitações técnicas, algo que não permitiu o desenvolvimento da aplicação *Android* integralmente. Assim, propõe-se a continuação do desenvolvimento da versão *Android* da aplicação, garantindo que um maior número de utilizadores possa utilizar o sistema.

6.2.3 Validação do Cálculo da Variabilidade da Frequência Cardíaca

Apesar da biblioteca fornecida pelo fabricante do sensor utilizado no sistema permitir o acesso simplificado às leituras da frequência cardíaca, os valores médios da VFC são calculados mediante a análise dos intervalos RR devolvidos pelo sensor, sendo depois aplicado um algoritmo para o cálculo da VFC através da métrica SDNN. Apesar de funcional, este algoritmo carece de validação por parte de uma equipa médica especializada. Dado que não foi possível efetuar qualquer tipo de validação com credibilidade médica à VFC, torna-se imperativo que profissionais de saúde qualificados avaliem a precisão do sistema. Este processo permitirá identificar eventuais lacunas e implementar melhorias necessárias.

6.2.4 Disponibilização da Aplicação em *App Store*.

Apesar do sistema no seu estado atual se encontrar operacional, o mesmo só pode ser utilizado eficientemente após disponibilizado através das lojas de aplicações de ambos os sistemas operativos (*App Store* da *Apple* e *Google Play Store*). Para que o mesmo seja possível, é necessário um processo de preparação de ambas as aplicações, onde é necessário garantir que as mesmas cumpram com todos os requisitos técnicos e legais, de modo a que sejam aprovadas e lançadas nas em ambas as *App Stores*. Este processo é especialmente importante na aplicação iOS, dado que os critérios de aprovação são altamente exigentes.

Concluindo, o sistema desenvolvido surge como uma alternativa aos sistemas de monitorização cardíaca comuns, permitindo que tutores ou profissionais de saúde consigam controlar indicadores cardíacos de vários utilizadores em simultâneo, através de uma aplicação móvel intuitiva e remota, cumprindo assim com os objetivos delineados inicialmente. Ainda assim, para que este sistema possa ser utilizado em contexto real, é necessário que alguns dos pontos definidos como trabalho futuro sejam concluídos, tais como a disponibilização da aplicação em *App Stores*.

A

Casos de uso do sistema

Neste anexo estão descritos os casos de uso definidos no processo de especificação do sistema.

A.1 Ator: Paciente

- **"Entrar como convidado".**
  - O utilizador acede à aplicação sem *login*.
  - Fluxo Principal:
    - O sistema permite a consulta das sessões disponíveis.
- **"Registar conta".**
  - O utilizador cria uma conta.
  - Condições:
    - O nome de utilizador e email escolhidos não estão em utilização.
  - Fluxo Principal:
    - O sistema valida os dados e cria a conta.
  - Fluxo Alternativo:
    - Se o nome de utilizador ou email já estiverem em uso, o sistema exibe uma mensagem de erro.
- **"*Login*".**

  - O utilizador efetua o *login*.
  - Condições:
    * A combinação `nome de utilizador` e `palavra passe` estão corretos.
  - Fluxo Principal:
    * O sistema autentica o utilizador, redirecionando o mesmo para o menu de funcionalidades disponíveis.
  - Fluxo Alternativo:
    * Se a combinação inserida estiver incorreta, o sistema exibe uma mensagem de erro.

- **"*Logout*".**
  - O utilizador realiza o *logout*.
  - Condições:
    * O utilizador efetuou o *login*.
  - Fluxo Principal:
    * O sistema encerra a sessão do utilizador e redireciona o mesmo para a página inicial.

- **"Visualizar sessões disponíveis".**
  - O utilizador visualiza uma lista de sessões disponíveis para inscrição.
  - Condições:
    * O utilizador efetuou o *login*.
  - Fluxo Principal:
    * O sistema exibe uma lista de sessões.

- **"Visualizar informações da sessão".**
  - O utilizador visualiza as informações de uma sessão.
  - Condições:
    * O utilizador efetuou o *login*.
  - Fluxo Principal:
    * O sistema exibe detalhes da sessão, incluindo:
      · Descrição;
      · Professor responsável;
      · Data e hora;
      · Número de vagas disponíveis.

- **"Inscrever numa sessão".**
  - O utilizador inscreve-se numa sessão para participar no futuro.
  - Condições:
    * O número de participantes inscritos na sessão não iguala o limite máximo;
    * O utilizador efetuou o *login*.
  - Fluxo Principal:
    * O sistema registra a inscrição e atualiza o número de vagas disponíveis.

- **"Cancelar inscrição numa sessão".**
  - O utilizador cancela a inscrição numa sessão.
  - Condições:
    * O utilizador está inscrito na sessão;
    * O utilizador efetuou o *login*.
  - Fluxo Principal:
    * O sistema remove a inscrição e atualiza o número de vagas disponíveis.
- **"Conectar a um sensor".**
  - O utilizador pode conectar-se a um sensor próximo.
  - Condições:
    * O utilizador está inscrito numa sessão;
    * O utilizador pode entrar na sessão;
    * O utilizador efetuou o *login*.
  - Fluxo Principal:
    * O sistema estabelece a conexão com o sensor e entra na sessão.
- **"Participar numa sessão".**
  - O utilizador entra numa sessão.
  - Condições:
    * O utilizador está inscrito numa sessão;
    * O utilizador está conectado a um sensor;
    * O utilizador efetuou o *login*.
  - Fluxo Principal:
    * O utilizador entra na sessão e inicia a monitorização cardíaca.
- **"Enviar dados cardíacos".**
  - O utilizador envia os seus dados cardíacos para o servidor.
  - Condições:
    * O utilizador participa numa sessão;
    * O utilizador está conectado a um sensor;
    * O utilizador efetuou o *login*.
  - Fluxo Principal:
    * O sistema recebe e armazena os dados cardíacos.
  - Fluxo Alternativo:
    * Se a conexão com o sensor for perdida, o sistema exibe uma mensagem de erro e redireciona o utilizador para um ecrã de fim de sessão.
- **"Consultar dados de sessões anteriores".**
  - O utilizador pode consultar dados cardíacos sobre uma sessão onde participou.
  - Condições:
    * O utilizador efetuou o *login*.
  - Fluxo Principal:
    * O sistema exibe os dados cardíacos da sessão selecionada.

A.2 Ator: Tutor:

- **"*Login*".**
  - O tutor efetua o *login*.
  - Condições:
    - O nome de utilizador e password estão corretos.
  - Fluxo Principal:
    - O sistema autentica o tutor e concede acesso às funcionalidades.
  - Fluxo Alternativo:
    - Se o nome de utilizador ou password estiverem incorretos, o sistema exibe uma mensagem de erro.
- **"*Logout*".**
  - O tutor realiza o *logout*.
  - Condições:
    - O tutor efetuou o *login*.
  - Fluxo Principal:
    - O sistema encerra a sessão do tutor e redireciona para a página inicial.
- **"Criar uma sessão".**
  - O tutor cria uma sessão.
  - Condições:
    - O tutor efetuou o *login*.
  - Fluxo Principal:
    - O sistema valida os dados e cria a sessão.
- **"Cancelar uma sessão".**
  - O tutor cancela uma sessão.
  - Condições:
    - O tutor criou a sessão;
    - O tutor efetuou o *login*.
  - Fluxo Principal:
    - O sistema remove a sessão e notifica os participantes.
- **"Iniciar uma sessão".**
  - O tutor inicia uma sessão.
  - Condições:
    - O tutor criou a sessão;
    - O tutor efetuou o *login*.
  - Fluxo Principal:
    - O sistema inicia a sessão e permite o acesso dos participantes.

- **"Terminar uma sessão".**
  - O tutor termina uma sessão em funcionamento.
  - Condições:
    * O tutor iniciou a sessão;
    * O tutor efetuou o *login*.
  - Fluxo Principal:
    * O sistema encerra a sessão e gera um relatório de todos os pacientes.

B

Código-fonte para criação da base de dados

Neste anexo consta todo o código-fonte desenvolvido para a criação das tabelas utilizadas no sistema.

B.1 Código Fonte

```
-- Table to store user information
CREATE TABLE IF NOT EXISTS user (
   username TEXT PRIMARY KEY,
   firstName TEXT NOT NULL,
   lastName TEXT NOT NULL,
   email TEXT NOT NULL,
   dateOfBirth DATE NOT NULL,
   password TEXT NOT NULL,
   gender TEXT NOT NULL
);

-- Table to store teacher information
CREATE TABLE IF NOT EXISTS teacher (
   username TEXT PRIMARY KEY,
   name TEXT NOT NULL,
   password TEXT NOT NULL
```

```
);

-- Table to store session information
CREATE TABLE IF NOT EXISTS session (
   sessionId INTEGER PRIMARY KEY AUTOINCREMENT,
   name TEXT NOT NULL,
   teacher TEXT NOT NULL,
   description TEXT,
   date DATE NOT NULL,
   hour INTEGER NOT NULL,
   spots INTEGER NOT NULL,
   isActive INTEGER DEFAULT 0,
   FOREIGN KEY (teacher) REFERENCES teacher(username)
);

-- Table to store user sign-ups for sessions
CREATE TABLE IF NOT EXISTS sessionSigning (
   sessionId INTEGER NOT NULL,
   username TEXT NOT NULL,
   PRIMARY KEY (sessionId, username),
   FOREIGN KEY (username) REFERENCES user(username),
   FOREIGN KEY (sessionId) REFERENCES session(sessionId)
);

-- Table to store summary data for user sessions
CREATE TABLE IF NOT EXISTS sessionSummary (
   sessionId INTEGER,
   username TEXT,
   hrCount INTEGER,
   hrAverage INTEGER,
   hrMaximum INTEGER,
   hrMinimum INTEGER,
   hrv INTEGER,
   PRIMARY KEY (sessionId, username),
   FOREIGN KEY (username) REFERENCES user(username),
   FOREIGN KEY (sessionId) REFERENCES session(sessionId)
);
```

B.2 Documentação

Adicionalmente, foi criada alguma documentação para ajudar na compreensão do código-fonte.

B.2.1 Tabela *user*

A documentação desenvolvida para a tabela *user* foi a seguinte:

Atributo	Tipo de Dados	Descrição	Informação Adicional
username	TEXT	Identificador único do utilizador.	Chave Primária.
firstName	TEXT	Primeiro nome do utilizador.	-
lastName	TEXT	Apelido do utilizador.	-
email	TEXT	Endereço de e-mail do utilizador.	-
dateOfBirth	DATE	Data de nascimento do utilizador.	Formato DD-MM-YYYY.
password	TEXT	Senha do utilizador.	-
gender	TEXT	Gênero do utilizador.	M = Masculino, F = Feminino.

Tabela B.1: Documentação da tabela user.

B.2.2 Tabela *teacher*

A documentação desenvolvida para a tabela *teacher* foi a seguinte:

Atributo	Tipo de Dados	Descrição	Informação Adicional
username	TEXT	Identificador único do professor.	Chave Primária.
name	TEXT	Nome completo do professor.	-
password	TEXT	Senha do professor.	-

Tabela B.2: Documentação da tabela teacher.

B.2.3 Tabela *session*

A documentação desenvolvida para a tabela *session* foi a seguinte:

Atributo	Tipo de Dados	Descrição	Informação Adicional
sessionId	INTEGER	Identificador único da sessão.	Chave Primária Incremento automático
name	TEXT	Nome da sessão.	-
teacher	TEXT	Professor responsável pela sessão.	Chave Estrangeira
description	TEXT	Descrição da sessão.	Opcional
date	DATE	Data da sessão.	Formato YYYY-MM-DD.
hour	INTEGER	Hora da sessão.	Formato 24 horas
spots	INTEGER	Número de vagas totais da sessão.	-
isActive	INTEGER	Status da sessão.	0 = Inativa 1 = Ativa -1 = Terminada

Tabela B.3: Documentação da tabela session.

B.2.4 Tabela *sessionSigning*

A documentação desenvolvida para a tabela *sessionSigning* foi a seguinte:

Atributo	Tipo de Dados	Descrição	Informação Adicional
sessionId	INTEGER	Identificador da sessão.	Chave Estrangeira
username	TEXT	Identificador do utilizador.	Chave Estrangeira

Tabela B.4: Documentação da tabela sessionSigning.

B.2.5 Tabela *sessionSigning*

A documentação desenvolvida para a tabela *sessionSummary* foi a seguinte:

Atributo	Tipo de Dados	Descrição	Informação Adicional
sessionId	INTEGER	Identificador da sessão.	Chave Estrangeira
username	TEXT	Identificador do utilizador.	Chave Estrangeira
hrCount	INTEGER	Número total de leituras de frequência cardíaca.	-
hrAverage	INTEGER	Média da frequência cardíaca.	Valor em BPM
hrMaximum	INTEGER	Frequência cardíaca máxima.	Valor em BPM
hrMinimum	INTEGER	Frequência cardíaca mínima.	Valor em BPM
hrv	INTEGER	Variabilidade da frequência cardíaca média.	Valor em ms

Tabela B.5: Documentação da tabela *sessionSummary*.

B.3 Exemplos

Por fim, foram criados alguns exemplos de possíveis entradas em cada tabela na base de dados.

B.3.1 Tabela *user*

Atributo	Exemplo
username	joao.rouxinol
firstName	Joao
lastName	Rouxinol
email	joao.rouxinol@example.com
dateOfBirth	23-09-1996
password	exemplo_12345
gender	M

Tabela B.6: Exemplo de entrada na tabela *user*.

B.3.2 Tabela *teacher*

Atributo	Exemplo
username	joao.silva
name	Joao Silva
password	exemplo_67890

Tabela B.7: Exemplo de entrada na tabela *teacher*.

B.3.3 Tabela *session*

Atributo	Exemplo
sessionId	1
name	Pilates
teacher	joao.silva
description	Aula de pilates para iniciantes.
date	25-04-2025
hour	18
spots	50
isActive	0

Tabela B.8: Exemplo de entrada na tabela *session*.

B.3.4 Tabela *sessionSigning*

Atributo	Exemplo
sessionId	1
username	joao.rouxinol

Tabela B.9: Exemplo de entrada na tabela *sessionSigning*.

B.3.5 Tabela *sessionSummary*

Atributo	Exemplo
sessionId	1
username	joao.rouxinol
hrCount	120
hrAverage	75
hrMaximum	110
hrMinimum	60
hrv	100

Tabela B.10: Exemplo de entrada na tabela *sessionSummary*.

C

Código-fonte dos testes efetuados

Neste anexo é apresentado todo o código-fonte utilizado para os testes realizados aos vários componentes do sistema.

C.1 Testes à aplicação

Para a execução de testes à aplicação *iOS*, foi criado um pequeno conjunto de testes automáticos na linguagem *Swift* que testam os diferentes tipos de inicialização da aplicação.

C.1.1 Código Fonte

```
import XCTest

class LaunchTimeTests: XCTestCase {
    func testColdLaunchTime() {
        let app = XCUIApplication()
        measure {
            app.launch()
        }
    }

    func testWarmLaunchTime() {
```

```
        let app = XCUIApplication()
        app.launch()
        app.terminate()

        measure {
            app.launch()
        }
    }

    func testResumeTime() {
        let app = XCUIApplication()
        app.launch()

        XCUIDevice.shared.press(.home)
        sleep(2)

        measure {
            app.activate()
        }
    }
}
```

C.2 Testes ao serviço *web*

Para a execução de testes ao serviço *web*, foi criado um pequeno *script* na linguagem *Python* que permite a execução de um número arbitrário de pedidos *REST* em simultâneo.

C.2.1 Código Fonte

```
import asyncio
import httpx
import time
import numpy as np
from pydantic import BaseModel
from typing import List, Optional, Union

class LoadTestResult(BaseModel):
    total_requests: int
    passed: int
    failed: int
    total_time: float
    average_time_per_request: float

async def make_request(client, endpoint):
    try:
        start_time = time.time()
        response = await client.get(endpoint)
```

```
        end_time = time.time()
        return response.status_code, end_time - start_time
    except Exception as e:
        return str(e), None

async def main() -> LoadTestResult:
    limits = httpx.Limits(max_connections=500)
    async with httpx.AsyncClient(
    base_url="http://localhost:8000",
    limits=limits
    ) as client:
        num_requests = 50

        test_start_time = time.time()
        tasks = [
            make_request(client,
            "/get-sessions/Guest") for _ in range(num_requests)
            ]
        results = await asyncio.gather(*tasks)
        test_end_time = time.time()
        total_time = test_end_time - test_start_time

        # Cálculo das metricas de resposta
        response_times = [
            time_taken for status_code,
            time_taken in results if time_taken is not None
        ]
        avg_time_per_request = np.mean(response_times) if response_times else 0
        passed = sum(1 for status_code, _ in results if status_code == 200)
        failed = len(results) - passed

        return LoadTestResult(
            total_requests=num_requests,
            passed=passed,
            failed=failed,
            total_time=total_time,
            average_time_per_request=avg_time_per_request
        )

# Example usage
if __name__ == "__main__":
    result = asyncio.run(main())
    print(result.json())
```

C.2.2 Exemplo de *output*

```
{
    "total_requests": 500,
    "passed": 248,
    "failed": 252,
    "total_time": 2.38,
    "average_time_per_request": 1.21
}
```

C.3 Testes à base de dados

Para a execução de testes à base de dados, foi criado um pequeno *script* na linguagem *Python* que permite a execução de um número arbitrário de *queries SQL* em simultâneo.

C.3.1 Código-Fonte para um teste de inserção

```
import sqlite3
import time
from pydantic import BaseModel
from typing import List

class InsertTestResult(BaseModel):
    total_inserts: int
    total_time: float
    average_time_per_insert: float

def testInsert(db_path: str, num_inserts: int) -> InsertTestResult:
    conn = sqlite3.connect(db_path)
    cursor = conn.cursor()

    insert_query = """
    INSERT INTO session
    VALUES (?, 'name', 'teacher', 'description', '18-03-2025', '16', 100, 0);
    """

    total_start_time = time.time()
    individual_times = []

    for session_id in range(1, num_inserts + 1):
        start_time = time.time()
        cursor.execute(insert_query, (session_id,))
        conn.commit()

        end_time = time.time()
        insert_time = end_time - start_time
        individual_times.append(insert_time)
```

```
    total_end_time = time.time()
    total_elapsed_time = total_end_time - total_start_time
    average_time_per_insert = sum(individual_times) / len(individual_times)

    conn.close()

    return InsertTestResult(
        total_inserts=num_inserts,
        total_time=total_elapsed_time,
        average_time_per_insert=average_time_per_insert
    )

# Example usage
if __name__ == "__main__":
    DB_PATH = 'HeartRateMonitoring.sqlite3'
    NUM_INSERTS = 10000
    result = testInsert(DB_PATH, NUM_INSERTS)
    print(result.json())
```

C.3.2 Exemplo de *output* para um teste de inserção

```
{
    "total_inserts": 10000,
    "total_time": 2.7208,
    "average_time_per_insert": 0.0003
}
```

C.3.3 Código-Fonte para um teste de consulta

```
import sqlite3
import time
from pydantic import BaseModel
from typing import List

class SelectTestResult(BaseModel):
    total_selects: int
    total_time: float
    average_time_per_insert: float

def testSelect(db_path: str, num_selects: int) -> SelectTestResult:
    conn = sqlite3.connect(db_path)
    cursor = conn.cursor()

    select_query = """
    SELECT *
    FROM session
```

```
    """

    total_start_time = time.time()
    individual_times = []

    for session_id in range(1, num_inserts + 1):
        start_time = time.time()
        cursor.execute(select_query)
        conn.commit()

        end_time = time.time()
        select_time = end_time - start_time
        individual_times.append(select_time)

    total_end_time = time.time()
    total_elapsed_time = total_end_time - total_start_time
    average_time_per_select = sum(individual_times) / len(individual_times)

    conn.close()

    return SelectTestResult(
        total_selects=num_selects,
        total_time=total_elapsed_time,
        average_time_per_insert=average_time_per_select
    )

# Example usage
if __name__ == "__main__":
    DB_PATH = 'HeartRateMonitoring.sqlite3'
    NUM_SELECTS = 10000
    result = testInsert(DB_PATH, NUM_SELECTS)
    print(result.json())
```

C.3.4 Exemplo de *output* para um teste de consulta

```
{
    "total_selects": 10000,
    "total_time": 0.0754,
    "average_time_per_insert": 0.0000
}
```

Bibliografia

[AAR22] Naveen Alugubelli, Hussam Abuissa, and Archana Roka. Wearable devices for remote monitoring of heart rate and heart rate variability-what we know and what is coming. *Sensors (Basel)*, 22(22):8903, 2022. Published: 17 November 2022.

[Ama24a] Amazon Web Services (AWS). O que é nosql? `https://aws.amazon.com/pt/nosql/`, 2024. Accessed: 23 December 2024.

[Ama24b] Amazon Web Services (AWS). O que é uma api (interface de programação de aplicações)? `https://aws.amazon.com/pt/what-is/api/`, 2024. Acedido em: 26 dezembro 2024.

[Ama24c] Amazon Web Services (AWS). Qual é a diferença entre o soap e o rest? `https://aws.amazon.com/pt/compare/the-difference-between-soap-rest/`, 2024. Acedido em: 26 dezembro 2024.

[Ama25] Amazon Web Services (AWS). Qual é a diferença entre http e https? `https://aws.amazon.com/pt/compare/the-difference-between-https-and-http/`, 2025. Acedido em: 14 fevereiro 2025.

[Ame24] American Heart Association. Target heart rates chart, 2024. Disponível em: `https://www.heart.org/en/healthy-living/fitness/fitness-basics/target-heart-rates`.

[App25] Apple Inc. Macbook pro (13-inch, m1, 2020) - technical specifications. `https://support.apple.com/en-us/111893`, 2025. Accessed: 10 February 2025.

[Ard18] Arduino. Introduction to arduino. `https://www.arduino.cc/en/Guide/Introduction/`, 2018. Last revision: 5 February 2018. Accessed: 22 December 2024.

[Aut25] Auth0. What is oauth 2.0? `https://auth0.com/intro-to-iam/what-is-oauth-2`, 2025. Accessed: 14 February 2025.

[Bel24] Matteo Beltante. Assessing llm capabilities in mobile ui code conversion: A swiftui and jetpack compose comparative study. `https://hdl.handle.net/10589/227660`, 2024. Published: 10 October 2024. Accessed: 29 December 2025.

[Blu24] Bluetooth SIG. Bluetooth technology overview. https://www.bluetooth.com/learn-about-bluetooth/tech-overview/, 2024. Accessed: 23 December 2024.

[CCMMGGOG23] Rafael E. Cañón-Clavijo, Carlos E. Montenegro-Marin, Paulo A. Gaona-Garcia, and Juan Ortiz-Guzmán. Iot based system for heart monitoring and arrhythmia detection using machine learning. *Journal of Healthcare Engineering*, 2023:6401673, 2023. Published: 8 February 2023.

[Cen23] Centre Cardiologique Laval. The crucial role of cardiac health monitoring. https://www.centrecardiolaval.com/en/2023/11/16/the-crucial-role-of-cardiac-health-monitoring/, 2023. Published: 16 November 2023. Accessed: 20 December 2024.

[Cen24] Centers for Disease Control and Prevention. Heart disease risk factors. https://www.cdc.gov/heart-disease/risk-factors/index.html, 2024. Posted: 2 December 2024. Accessed: 20 December 2024.

[Dam22] Sabri Dami. Internet of things-based health monitoring system for early detection of cardiovascular events during covid-19 pandemic. *World Journal of Clinical Cases*, 10(26):9207–9218, 2022. Published: 2022.

[Dja25] Django Software Foundation. Django documentation. https://docs.djangoproject.com/en/5.1/, 2025. Accessed: 14 January 2025.

[Eri24] Jeffrey Erickson. Mysql: Understanding what it is and how it's used. https://www.oracle.com/mysql/what-is-mysql/, 2024. Published: 29 August 2024. Accessed: 26 December 2024.

[Fas25a] FastAPI. Alternatives, inspiration and comparisons. https://fastapi.tiangolo.com/alternatives/, 2025. Accessed: 14 January 2025.

[Fas25b] FastAPI Documentation. Custom response - html, stream, file, others. https://fastapi.tiangolo.com/advanced/custom-response/#streamingresponse, 2025. Accessed: 14 February 2025.

[Fas25c] FastAPI-Users Documentation. Password hashing configuration. https://fastapi-users.github.io/fastapi-users/latest/configuration/password-hash/, 2025. Accessed: 14 February 2025.

[Fla25] Flask Documentation. Flask user's guide. https://flask.palletsprojects.com/en/stable/, 2025. Accessed: 14 January 2025.

[FW20] Flavio D. Fuchs and Paul K. Whelton. High blood pressure and cardiovascular disease. *Hypertension*, 75(2):285–292, 2020. Accessed: 20 December 2024.

[Gle24] Glenegales Hospitals. Understanding the internet of medical things (iomt) and its benefits. https://www.gleneagleshospitals.co.in/blogs/understanding-internet-medical-things-iomt-benefits, 2024. Accessed: 20 December 2024.

[Gun25] Gunicorn Documentation. Gunicorn architecture design documentation. https://docs.gunicorn.org/en/latest/design.html, 2025. Accessed: 10 February 2025.

[HHZ14] Marc T. Hamilton, David G. Hamilton, and Theodore W. Zderic. Sedentary behavior as a mediator of type 2 diabetes. *Medicine and Sport Science*, 60:11–26, 2014. Accessed: 20 December 2024.

[HkBF+18] Ahmed Faeq Hussein, N. Arun kumar, Marlon Burbano-Fernandez, Gustavo Ramírez-González, Enas Abdulhay, and Victor Hugo C. De Albuquerque. An automated remote cloud-based heart rate variability monitoring system. *IEEE Access*, 6:77055–77064, 2018.

[IBM21] IBM. What is a relational database? `https://www.ibm.com/think/topics/relational-databases`, 2021. Published: 20 October 2021. Accessed: 26 December 2024.

[IKP23] Deri Indrawan, Dana Kusumo, and Shinta Puspitasari. Analysis of the implementation of mvvm architecture pattern on performance of ios mobile-based applications. *JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)*, 8:59–65, 02 2023.

[JBJKW20] Ben W. Johnston, Richard Barrett-Jolley, Anton Krige, and Ingeborg D. Welters. Heart rate variability: Measurement and emerging use in critical care medicine. *Journal of Intensive Care Society*, 21(2):148–157, 2020. Published: 2020.

[Kaj24] Kajeet. Advantages of iot in remote patient monitoring. `https://www.kajeet.com/en/blog/advantages-of-iot-in-remote-patient-monitoring`, 2024. Accessed: 20 December 2024.

[Kli18] Blanka Klimova. Acceptance and use of mobile devices and apps by elderly people. In *17th Conference on e-Business, e-Services and e-Society (I3E)*, pages 30–36, Kuwait City, Kuwait, oct 2018. Springer.

[KNE+17] Alqassim Khushhal, Simon Nichols, William Evans, Rebecca Goulding, Sarah Jane Hobbs, Mark Kelson, Mark Lyons, Rebecca A. Needham, Adam Runacres, and Kelly A. Mackintosh. Validity and reliability of the Apple Watch for measuring heart rate during exercise. *Sports Medicine International Open*, 1(6):E206–E211, oct 2017.

[Kot24] Kotlin Documentation. Kotlin multiplatform documentation. `https://kotlinlang.org/docs/multiplatform.html`, 2024. Accessed: 29 December 2024.

[KTK15] Priyanka Kakria, N. K. Tripathi, and P. Kitipawang. A real-time health monitoring system for remote cardiac patients using smartphone and wearable sensors. *International Journal of Telemedicine and Applications*, 2015:373474, 2015. Published: 2015.

[LCMR+23] Kevin Li, Carlos Cardoso, Alejandro Moctezuma-Ramirez, Abdelmotagaly Elgalad, and Emerson Perin. Heart rate variability measurement through a smart wearable device: Another breakthrough for personal health monitoring? *International Journal of Environmental Research and Public Health*, 20(24):7146, 2023. Published: 6 December 2023.

[LeW24] Howard E. LeWine. Want to check your heart rate? here's how. Online, 4 2024. Chief Medical Editor, Harvard Health Publishing; Editorial Advisory Board Member, Harvard Health Publishing.

[LFZ+21] Jian Lin, Rumin Fu, Xinxiang Zhong, Peng Yu, Guoxin Tan, Wei Li, Huan Zhang, Yangfan Li, Lei Zhou, and Chengyun Ning. Wearable sensors and devices for real-time cardiovascular disease monitoring. *Cell Reports Physical Science*, 2(8):100541, 2021.

[MDN24] MDN Web Docs. Using server-sent events. https://developer.mozilla.org/en-US/docs/Web/API/Server-sent_events/Using_server-sent_events, 2024. Accessed: 27 December 2024.

[Med24] Medscape. Normal electrocardiography (ecg) intervals. https://emedicine.medscape.com/article/2172196-overview?form=fpf, 2024. Accessed: February 16, 2024; Published: January 30, 2025; Accessed: January 30, 2025.

[Mednd] Caring Medical. Heart rate variability, n.d. Accessed February 21, 2025.

[MMZ+15] Hasmah Mansor, Siti Sarah Meskam, Nasiha Sakinah Zamery, Nur Quraisyia Aqilah Mohd Rusli, and Rini Akmeliawati. Portable heart rate measurement for remote health monitoring system. In *2015 10th Asian Control Conference (ASCC)*, pages 1–5, 2015.

[Mon21] Corbin Montague. Monitoring app performance on ios. https://medium.com/expedia-group-tech/monitoring-app-performance-on-ios-7a48fb25cfc2, 2021. Published: 23 March 2021. Accessed: 10 February 2025.

[Mound] Mount Sinai Health System. Warning signs and symptoms of heart disease, n.d. Accessed: 2 December 2024.

[Mov23] Movesense. Movesense medical sensor data sheet spec-1. https://www.movesense.com/wp-content/uploads/2023/04/Movesense-Medical-Spec-Sheet-2.0-03-2023.pdf, 2023. Version 2.0, March 2023. Accessed: 23 December 2024.

[Mov24] Movesense. Movesense mobile library. https://bitbucket.org/movesense/movesense-mobile-lib/src/master/, 2024. Accessed: 23 December 2024.

[Mov25] Movesense. *Movesense API Reference*. Suunto Oy, jan 2025. Accessed: January 10, 2025.

[MR23] Edon Milla and Mirjana Radonjić. Analysis of developing native android applications using xml and jetpack compose. *Balkan Journal of Applied Mathematics and Informatics*, 6(2):167–178, 2023. Accessed: 29 December 2024.

[Nat22] National Heart, Lung, and Blood Institute. What is an arrhythmia? https://www.nhlbi.nih.gov/health/arrhythmias, 2022. Last updated: 24 March 2022. Accessed: 22 December 2024.

[Nat24] National Institute on Aging. Heart health and aging. https://www.nia.nih.gov/health/heart-health/heart-health-and-aging, 2024. Content reviewed: 22 July 2024. Accessed: 20 December 2024.

[OLB+24] Ben O'Grady, Rory Lambe, Maximus Baldwin, Tara Acheson, and Cailbhe Doherty. The validity of apple watch series 9 and ultra 2 for serial measurements of heart rate variability and resting heart rate. *Sensors*, 24(19), 2024.

[Ora24a] Oracle. Internet of things (iot). https://www.oracle.com/pt/internet-of-things/, 2024. Accessed: 20 December 2024.

[Ora24b] Oracle. Why choose oracle database for all your data needs? https://www.oracle.com/a/ocom/docs/database/why-choose-oracle-database-infographic.pdf, 2024. Accessed: 26 December 2024.

[Par24] Paris Business School. The 5 most frequently used web programming languages. https://www.edcparis.edu/en/school-news/5-most-frequently-used-web-programming-languages, 2024. Published: 3 September 2024. Accessed: 27 December 2024.

[PFT+24] Alberto Preda, Raffaele Falco, Chiara Tognola, Marco Carbonaro, Sara Vargiu, Michela Gallazzi, Matteo Baroni, Lorenzo Gigli, Marisa Varrenti, Giulia Colombo, Gabriele Zanotto, Cristina Giannattasio, Patrizio Mazzone, and Fabrizio Guarracini. Contemporary advances in cardiac remote monitoring: A comprehensive, updated mini-review. *Medicina*, 60(5), 2024.

[Poc24] Radu Poclitari. Python vs java for app development: The war of fierce backend rivals. https://www.index.dev/blog/python-vs-java-for-app-development-the-war-of-fierce-backend-rivals, 2024. Published: 15 February 2024. Accessed: 27 December 2024.

[Pos24] PostgreSQL Global Development Group. About postgresql. https://www.postgresql.org/about/, 2024. Accessed: 26 December 2024.

[Pyt25] Python Central. How python simplifies api development: A guide for scalable web solutions. https://www.pythoncentral.io/how-python-simplifies-api-development-a-guide-for-scalable-web-solutions/, 2025. Published: 14 January 2025. Accessed: 20 January 2025.

[RCW24] Mike Ray, Saisang Cai, and Randolph West. What is sql server? https://learn.microsoft.com/en-us/sql/sql-server/what-is-sql-server?view=sql-server-ver16, 2024. Created: 4 September 2024. Accessed: 26 December 2024.

[Red24] Redgate Software. Db-engines ranking. https://db-engines.com/en/ranking, 2024. Accessed: 26 December 2024.

[Rou25] Rouxinelo. Github repository. https://github.com/Rouxinelo, 2025. Accessed: 10 February 2025.

[RSC+22] Bruce Rogers, Marcelle Schaffarczyk, Martina Clauß, Laurent Mourot, and Thomas Gronwald. The movesense medical sensor chest belt device as single channel ecg for rr interval detection and hrv analysis during resting state and incremental exercise: A cross-sectional validation study. *Sensors*, 22(5), 2022.

[RSG22] Bruce Rogers, Marcelle Schaffarczyk, and Thomas Gronwald. Estimation of respiratory frequency in women and men by kubios hrv software using the polar h10 or movesense medical ecg sensor during an exercise ramp. *Sensors*, 22(19), 2022.

[SG17] Fred Shaffer and J. P. Ginsberg. An overview of heart rate variability metrics and norms. *Frontiers in Public Health*, 5:258, sep 2017. Published 2017 Sep 28.

[Ski24] Skip Tools. Skip tools: Compare. https://skip.tools, 2024. Accessed: 29 December 2024.

[Soe24] Mads Soegaard. Native, web or hybrid app: Which one is better? *Interaction Design Foundation - IxDF*, 2024. Published: 22 January 2024. Accessed: 29 December 2024.

[SQL24] SQLite. Appropriate uses for sqlite. https://www.sqlite.org/whentouse.html, 2024. Accessed: 26 December 2024.

[Tom24] Tom. Optimizing fastapi for high performance: A comprehensive guide. https://tomtalksit.itsupportpro.uk/optimizing-fastapi-for-high-performance-a-comprehensive-guide-1e08c16924b3, 2024. Published: 22 August 2024. Accessed: 10 February 2025.

[Uni23] Uniqkey. Password recovery methods: The do's & don'ts. https://blog.uniqkey.eu/password-recovery-methods/, 2023. Published: 9 August 2023. Accessed: 14 February 2025.

[Uğ24] Gaye Uğur. What is reactive programming in swift? https://gayeugur.medium.com/what-is-reactive-programming-in-swift-c35fd9fd4af0, 2024. Published: 9 May 2024. Accessed: 29 December 2024.

[Web25] Web Accessibility Initiative (WAI). How to meet wcag. https://www.w3.org/WAI/WCAG22/quickref/, 2025. Accessed: 10 March 2025.

[Wor] World Health Organization. Cardiovascular diseases. https://www.who.int/health-topics/cardiovascular-diseases. Accessed: 2 December 2024.

[Wor21] World Health Organization. Cardiovascular diseases (cvds). https://www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds), 2021. Published: 11 June 2021. Accessed: 20 December 2024.

[Wor23] World Heart Federation. World heart report 2023. https://world-heart-federation.org/wp-content/uploads/World-Heart-Report-2023.pdf, 2023. Accessed: 20 December 2024.

[Wor24] World Heart Federation. Prevention. https://world-heart-federation.org/what-we-do/prevention/, 2024. Accessed: 20 December 2024.

[WSP21] Paweł Wiertel and Maria Skublewska-Paszkowska. Comparative analysis of uikit and swiftui frameworks in ios system. *Journal of Computer Sciences Institute*, 20:170–174, 2021. Accessed: 29 December 2024.

[XYH20] Ningning Xiao, Wei Yu, and Xu Han. Wearable heart rate monitoring intelligent sports bracelet based on internet of things. *Measurement*, 164:108102, 2020.

[Ży24] Bartłomiej Żyliński. Sse vs websockets: Comparing real-time communication protocols. https://softwaremill.com/sse-vs-websockets-comparing-real-time-communication-protocols/, 2024. Published: 26 February 2024. Accessed: 27 December 2024.

Contactos:
Universidade de Évora
Escola de Ciências e Tecnologia
Colégio Luis António Verney, Rua Romão Ramalho, nº59
7000 - 671 Évora | Portugal
Tel: (+351) 266 745 371
email: geral@ect.uevora.pt