UNIVERSIDADE D
COIMBRA

José Eduardo Salgado Ramos

iFriend – um sistema IoT para a monitorização de pessoas idosas

Dissertação no âmbito do Mestrado em Engenharia Eletrotécnica e de Computadores, na especialização de Computadores orientada pelo Professor Doutor Jorge Sá Silva e Professor Doutor André Rodrigues e apresentada ao Departamento de Engenharia Eletrotécnica e de Computadores da Faculdade de Ciências e Tecnologias da Universidade de Coimbra.

Setembro de 2022

Agradecimentos

Em primeiro lugar gostaria de agradecer ao Prof. Dr. Jorge Sá Silva e ao Prof. Dr. André Rodrigues por todo o apoio que me deram durante a realização desta dissertação e pela oportunidade de realizar a mesma no âmbito deste projeto. De seguida, agradeço a todos os membros do grupo de trabalho, com atenção especial ao Marcelo Fernandes por toda a ajuda fornecida a solucionar problemas neste projeto. Ainda no âmbito da dissertação queria agradecer a todos os parceiros do projeto: a Universidade de Alcalá, a Universidade de Extremadura e o Hospital Príncipe de Astúrias, que contribuíram positivamente de maneira a tornar este projeto cada vez mais completo.

A nível pessoal, queria agradecer aos meus pais por me darem a oportunidade de estudar na Universidade de Coimbra e por todo o apoio que deram no meu percurso académico. Também gostava de agradecer a todas as pessoas que me acompanharam ao longo destes anos e que sempre estiveram disponíveis para me ajudar nos momentos mais complicados e para celebrar nos melhores momentos.

Resumo

O número de pessoas de 3ª idade está a aumentar, o que leva a que cada vez exista um maior número de pacientes com a necessidade de apoio clínico e que sejam vigiados com mais frequência, de forma que seja possível detetar precocemente certas doenças e situações de perigo como as quedas.

Com a evolução das tecnologias e uma vez que os dispositivos móveis apresentam cada vez mais funcionalidades, tem sido desenvolvido ambientes de Internet das Coisas (IoT). O IoT permite utilizar estes novos dispositivos como sensores, para retirar dados do mundo real, possibilitando uma variedade de aplicações que ajudam a melhorar a qualidade de vida das pessoas.

Assim, usando um *Human in the Loop Cyber-Physical Systems* (HITLCPS) pode-se criar uma aplicação com foco nas pessoas e que estas sejam integradas no sistema. Ou seja, pode-se desenvolver um sistema que permite monitorizar parâmetros físicos e psicológicos, atividades física, qualidade de sono, nível de stress, rotinas, entre outros.

O objetivo desta tese é, então, criar um sistema HITLCPS com o propósito de monitorizar idosos de um lar, com insuficiência renal, de maneira a conseguir avaliar o seu estado de saúde e conseguir fazer uma possível inferência de doenças que poderão estar a desenvolver-se.

Este projeto tem como base o desenvolvimento de uma aplicação para *smartphone*, *smartwatch* e de um sistema de monitorização não-intrusiva através do uso de *routers*. As aplicações do *smartphone* e do *smartwatch* irão ser usadas para obter informações sobre a rotina dos idosos. Os *router*s serão usados, principalmente, para efetuar as medições dos ritmos respiratório e cardíaco. Para mostrar os dados recolhidos, por ambas as tecnologias, aos médicos que cuidam dos idosos, foi criado um *dashboard* que mostra estes dados numa forma tratada de maneira a dar informação relevante aos médicos.

Abstract

The number of elderly people is increasing, which means that there is an increasing number of patients in need of clinical support and in need to be monitored more frequently, in a way that it is possible to detect certain diseases and situations of danger from falls.

With the evolution of technologies and since devices have more and more functionalities, Internet of Things (IoT) environments have been developed. The IoT makes it possible to use these new devices as sensors to extract data from the real world, enabling a variety of applications that help improve people's quality of life.

Thus, using a Human in the Loop Cyber-Physical Systems (HITLCPS), we can create an application focused on people and integrate them into the system. That means, to develop a system that is capable of monitoring patient's physical and psychological parameters, physical activities, sleep quality, stress level, routines, among others.

The objective of this thesis is to create a HITLCPS system with the purpose of monitoring elderly people in a nursing home, with kidney failure, to be able to assess their health status and be able to make a possible inference of diseases that may be developing.

This project is based on the development of an application for smartphone, smartwatch, and a non-intrusive monitoring system using routers. The smartphone and smartwatch application will be used to obtain information about the routine of the elderly. The routers will mainly be used to measure respiratory and heart rates. To show the data collected by both technologies to the doctors who care for the elderly, a *dashboard* was created that shows this data in a processed form to provide relevant information to doctors.

Acrónimos

AFA. *Associação Fazer Avançar*
API. *Application Programming Interface*
BLE. *Bluetooth Low Energy*
BSSID. *Basic Service Set Identifier*
CPS. *Cyber-Physical Systems*
CSI. *Channel State Information*
GEs. *Generic Enablers*
GPS. Global Positioning System
HITLCPS. *Human in the Loop Cyber-Physical Systems*
HTTP. *Hypertext Transfer Protocol*
ID. *IDentification*
IoT. *Internet of Things*
ISABELA. *Iot Student Advisor and BEst Lifestyle Analyzer*
MAC. *Media Access Control*
REST. *Representational State Transfer*
SDKs. *Software Development KIT*
SO. *Sistema Operativo*
UI. *User Interface*
URL. Uniform Resource Locator

Lista de Figuras

Conteúdo

1. Introdução

1.1. Motivação

Devido à evolução da medicina e ao aumento da qualidade de vida na sociedade dos países desenvolvidos, a idade média da população tem vindo a aumentar a cada ano. Uma das consequências é o aumento do número de pessoas afetadas por várias doenças[1], como por exemplo a insuficiência renal, o que leva a que estas pessoas necessitem de ser supervisionadas mais frequentemente. Este aumento da população idosa e a situação pandémica atual, leva a uma sobrecarga do Sistema Nacional de Saúde (SNS) que é um grande problema nos dias em que vivemos.

Grande parte da investigação para o desenvolvimento e integração de sensores médicos em humanos é feita através do campo dos sensores vestíveis, contudo estes são desconfortáveis e exigem que o utilizador tenha mais cuidado com a roupa que usa e com a manutenção do próprio sensor.

Com vista a tentar solucionar este problema, neste projeto usou-se um sistema IoT para fazer uma monitorização constante dos sinais vitais, do estado mental e geral do doente, usando tecnologias como *smartphone*s, *smartwatch*es e redes WiFi.

IoT, *Internet of Things* é uma rede que conecta o mundo físico, ou seja, o espaço em que vivemos, com o mundo digital (Internet), o que proporciona à sociedade uma grande variedade de resolução de problemas. O IoT consiste na implementação de sensores que recebem os dados do exterior que posteriormente irão ser processados e exibidos a partir de um dispositivo.

O objetivo deste projeto é fornecer soluções tecnológicas ao alcance da maioria da população, mais concretamente na faixa etária da terceira e quarta idades. Para isso pretende-se construir uma solução que constantemente consiga detetar precocemente doenças no grupo dos idosos, sem a necessidade de usar equipamento médico ou vestuário sensorizado. Com esta solução pretende-se determinar o estado de saúde da pessoa e mostrar, num *dashboard*, se o seu estado está a piorar ou se surgiu uma nova patologia, usando um rastreio dos sinais vitais a partir da rede WiFi, bem como o comportamento humano a partir do *smartphone* e do *smartwatch*. Este projeto será testado num cenário real com pessoas idosas com insuficiência renal, que estão a ser acompanhadas por um corpo clínico, que poderão fornecer mais informações sobre o estado de saúde do paciente.

Posto isto, o desenvolvimento destes dois métodos de recolha de dados foi repartido, onde fiquei responsável pela construção da aplicação para *smartphone* e da parte do *dashboard* que mostra os dados recolhidos por esta, e o estudante Guilherme Lemos, orientando do Prof. Dr. Jorge Sá Silva e do Prof. Dr. André Rodrigues, ficou encarregue da recolha de sinais vitais através de redes WiFi e da parte do *dashboard* que corresponde a estes dados.

1.2. Estrutura do Documento

Este Projeto de dissertação está dividido em cinco partes: Introdução, Conceitos, Metodologias e Planeamento, Desenvolvimento do sistema iFriend e a Conclusão.

O 1º capítulo contém a motivação do projeto, ou seja, mostra o problema que está a ser abordado, como irá ser resolvido e os principais objetivos.

O 2º capítulo apresenta os principais conceitos e sistemas que foram implementados no projeto.

O 3ª capítulo detalha as metodologias usadas para o desenvolvimento do trabalho, uma descrição das principais tecnologias envolvidas, o estado de Arte e projetos semelhantes a este.

O 4º capítulo aborda o projeto iFriend, fazendo a descrição do projeto, dos parceiros, dos requisitos funcionais e não funcionais e da arquitetura do projeto.

O 5º capítulo mostra o trabalho desenvolvido e os testes realizados nesta dissertação.

No último capítulo apresenta-se a conclusão e o trabalho que se poderá desenvolver no futuro.

1.3. Metodologias

Nesta secção pretende-se descrever a metodologia de desenvolvimento que foi seguida na construção do projeto iFriend.

O iFriend usou uma metodologia *Scrum*, que consistiu em fazer diferentes iterações dentro do projeto de maneira a dar resposta aos requisitos do mercado, consumidores e parceiros. O planeamento do projeto segundo esta metodologia é feito a partir de *sprints*. Um *sprint* é uma iteração que faz parte de um desenvolvimento continuo de um projeto, normalmente dura entre 2 e 4 semanas, onde é realizado uma parte do trabalho que foi atribuído no início desse *sprint*. Todos os aspetos principais do projeto são divididos em múltiplas partes para serem distribuídos pelos *sprints*, de maneira a serem desenvolvidas e revistas, com o objetivo de chegar ao final com o produto concluído [2].

Esta metodologia é indicada para este trabalho porque o projeto contém fatores que estão sujeitos a muitas alterações, pois é um projeto em cooperação com outras entidades, o que leva a troca de opiniões e sugestões entre todas as partes em todas as fases do projeto.

Consequentemente, todas as semanas foram realizadas duas reuniões, a primeira apenas com o grupo que trabalhou no projeto iFriend, tendo como finalidade a atualização do estado de desenvolvimento do trabalho definido para o respetivo *sprint*, bem como a atribuição de trabalho para os próximos *sprints*. Desse modo, os orientadores da presente dissertação, Prof. Dr. Jorge Sá Silva e do Prof. Dr. André Rodrigues, dirigiam as reuniões, o aluno de doutoramento Marcelo Fernandes ajudava a solucionar problemas, sempre que necessário, e que por sua vez desenvolveu a estrutura do FIWARE do projeto ISABELA que foi usada neste projeto. O desenvolvimento do sistema de recolha de sinais vitais, a partir de redes WiFi, ficou a encargo do aluno de mestrado Guilherme Lemos, não obstante recaiu sobre mim o desenvolvimento da aplicação para smartphone. O desenvolvimento do *dashboard* foi realizado em cooperação pelo Guilherme Lemos e por mim.

Por outro lado, a segunda reunião era igualmente conduzida pelo Prof. Dr. Jorge Sá Silva e pelo Prof. Dr. André Rodrigues, no entanto esta era presenciada por todos os orientandos do grupo de trabalho do Prof. Dr. Jorge Sá Silva e do Prof. Dr. André Rodrigues. Deste modo, apresentávamos aos nossos colegas o ponto de situação do projeto, de maneira a mostrar o que estávamos a construir e como estávamos a fazê-lo,

sendo também uma forma de obtermos diferentes opiniões sobre como melhorar e delinear próximos passos para o projeto.

2. Estado de Arte

2.1. Conceitos

2.1.1. Internet of Things

A *Internet of Things* (IoT) é uma das tecnologias mais populares nos dias de hoje. Esta faz a conexão entre os objetos físicos e a Internet, possibilitando a recolha de dados de diferentes ambientes, como por exemplo, as tarefas diárias de uma pessoa. Assim a IoT pode ser aplicada em diferentes tipos de áreas de automação [3], como por exemplo monitorização ambiental, transportes e mobilidade, gestão de resíduos, eficiência energética e *smart grids*, gestão de água, segurança, *eHealth*, entre outros [4], reduzindo assim o esforço feito pelos humanos.

Os sistemas IoT são todos diferentes e têm a sua própria implementação, mas geralmente todos seguem um esquema semelhante. O sistema tem uma componente que recolhe dados do ambiente através de sensores, tendo estes objetos algum tipo de comunicação com o resto do sistema. De seguida temos a componente que trata de mover os dados dos objetos com os sensores para a base de dados, sendo que estes dados, por vezes, recebem um pré-processamento antes de serem guardados. E por fim, temos a componente que vai analisar estes dados de forma a tentar chegar a algum resultado, é nesta fase que são muitas vezes usados mecanismos de *machine learning* e formas de visualizar os dados[5].

Atualmente, a sociedade está rodeada por milhões de dispositivos que operam e interagem entre si para fornecer importantes serviços de computação. Desta maneira, deve-se apreciar a liberdade e a flexibilidade que a IoT proporciona e, o facto de a sociedade fazer parte deste ciclo, deste mundo conectado, torna o conceito ainda mais interessante. Ainda assim, mesmo com o avanço tecnológico, a IoT enfrenta alguns desafios que são principalmente [3]:

- **Segurança** – Como a segurança é um dos pilares da Internet, faz com que a segurança seja importante e o maior desafio para o IoT. Com o número de dispositivos a aumentar, a quantidade de oportunidades de explorar as suas vulnerabilidades aumenta, isto devido a dispositivos mal desenhados, que por exemplo, não protegem devidamente os dados que recolhem.
- **Privacidade** – O IoT cria desafios únicos em termos de privacidade, sendo que a maior parte deles provem da integração de dispositivos em ambientes sem o consentimento de quem faz parte destes. Isto torna-se predominante em dispositivos de uso corrente, como por exemplo, serviços de GPS, emails, dipositivos pessoais entre outros.
- **Compatibilidade** – Devido ao rápido crescimento do IoT, as diferentes tecnologias competem para serem consideradas como o padrão de uso, o que cria dificuldades quando se tenta conectar dispositivos de diferentes áreas. Temos o WiFi, Bluetooth e Z-Wave a competir pelo melhor método de transporte de dados.
- **Complexidade** – Por vezes, os dispositivos não são compatíveis uns com os outros, ou com as tecnologias usadas, então é necessário a construção de uma interface para que seja possível aceder a estes. Se este for o caso para muitos dispositivos a complexidade de gerir este sistema vai subir rapidamente.

- **Gestão de dados** – Estando os dados no centro de todo o esquema do IoT, estes precisam de ser bem geridos, mantidos em segurança e processados com o devido cuidado, para que não aconteça, por exemplo, a perda de dados, ou direcionar os dados para o local errado.
- **Fluxo de dados** – A quantidade enorme de dados produzido por todos os dispositivos podem acabar por consumir uma grande largura de banda, podendo causar excesso de tráfego e perdas de dados. Por isso, alguma estratégia tem de ser utilizada para que não ocorra perda de dados e exista um fluxo constante de dados.
- **Limites energéticos** – Todos os dipositivos necessitam de energia, e muitos deles foram desenhados com o intuito de poupar o máximo de energia, para que possam funcionar o máximo de tempo possível com as reservas de energias limitadas. Muitas vezes os sistemas IoT precisam que estes dispositivos executem várias tarefas que consomem energia, precisando, portanto, de subsistir um balanço entre uso de energia e computação, sendo assim um desafio para o IoT.

2.1.2 Human in the Loop Cyber-Physical Systems

Para se conseguir perceber o conceito *Human in the Loop Cyber-Physical Systems* (HITLCPS) é preciso definir o que são *Cyber-Physical Systems* e o conceito de *Human in the Loop*.

Cyber-Physical Systems (CPS) consistem na monitorização e controlo de ambientes e fenómenos físicos através de redes de dispositivos interconectados que possui sensores, de maneira a conseguirem atuar sobre estes ambientes. CPS representam um aglomerado de robótica, rede de sensores *wireless*, computação móvel e IoT, com o objetivo de chegar a um ambiente altamente monitorizado e facilmente controlado e adaptável [6].

Human in the Loop consiste num modelo onde a interação do humano é necessária para que o sistema funcione corretamente e responda às necessidades de quem o está a usar.

Desta forma, *Human in the Loop Cyber-Physical Systems* refere-se à junção destes dois conceitos numa só tecnologia, ou seja, quando usamos sensores e dispositivos móveis para monitorizar e avaliar a natureza humana, os humanos passam a fazer parte dos CPS. Os *Cyber-physical systems* passam a ter em consideração a interação humana, pois o comportamento e presença humana já não são vistos como um fator desconhecido ou externo, e passam a ser uma componente essencial do sistema [6].

Um HITLCPS abarca 3 componentes principais: aquisição de dados, inferência de estados e atuação. Como se pode ver na figura 1, o sistema começa por recolher dados do ambiente e do humano com os sensores que estão disponíveis, estes dados são depois processados por modelos de inferências de modo a retirar uma conclusão sobre o estado do mundo físico e do estado do humano, e por último o sistema atua sobre estes.

Embora exista uma constante evolução deste conceito, este ainda sofre de algumas limitações, tais como a dificuldade de integrar o humano como uma peça integrante do sistema, assim como o desenvolvimento de sistemas que permaneçam

estáticos, o que torna tudo muito mais controlado, mas ao mesmo tempo menos real. Outra limitação importante a referir é o facto de ainda existirem limitações éticas, isto deve-se ao facto de atualmente ainda existir resistência por parte da sociedade na monitorização em tempo real das atividades do dia a dia [6].

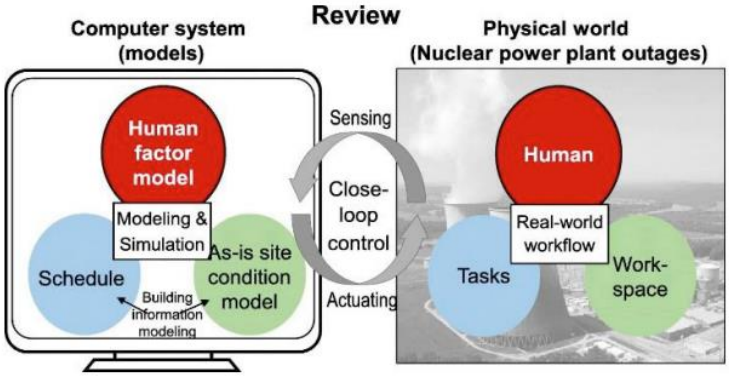

Figura 1- Exemplo de um diagrama de um sistema HITLCPS [7]

2.2. Tecnologias

2.2.1. Android

O Android é um sistema operativo (SO) concebido para dispositivos móveis que começou por ser desenvolvido por uma empresa de *Silicon Valley* chamada de Android Inc. Mais tarde esta foi comprada pela Google que forneceu ao Android uma vantagem no fornecimento de um conjunto completo de software, que incluía o SO principal, *middleware* [8], e aplicações móveis especificas, tornando-o mais tarde num sistema operativo *open-source*.

A linguagem de programação principal é o Java, mas na plataforma não existe *Java Development Machine*, que é o que permite executar códigos nesta linguagem. A Google desenvolveu uma máquina virtual especifica para o Android, denominada de *Dalvik [9]*. Esta foi desenhada para otimizar o uso da bateria e manter as funcionalidades num ambiente com memória e poder de processamento limitados, como acontece com telemóveis, *netbooks* e tablet PCs [10].

Atualmente o Android é importante, pois dá liberdade de criar aplicações novas e criativas, fornecendo uma maneira dinâmica de desenvolver novas *third-party applications*. Algumas funcionalidades importantes do Android são: adaptabilidade, facilidade de implementar *cross-platform*, possibilidade de acesso a permissões para especificar o uso de hardware e software disponível no dispositivo, entre outras funcionalidades. Uma das principais preocupações do Android é a segurança, não permitindo que aplicações externas alterem ficheiros instalados. O sistema possibilita ainda, aos utilizadores, o poder de aceitar ou não as permissões que são solicitadas pelas aplicações. Para além disto, a arquitetura do Android é bastante parecida ao de um computador o que simplifica a resolução problemas de segurança [9].

A arquitetura do Android é composta por 3 *layers*: *Application Layer* - onde as componentes das aplicações são executadas, a *App Framework Layer* - desenhada para permitir aos programadores terem acesso aos principais serviços das aplicações, e por último o Android *Runtime Layer e Linux Kernel* - focam-se principalmente no estado de execução dos processos [9].

Como referido anteriormente, uma aplicação Android é desenvolvida usando a linguagem de programação Java e, através de SDKs (*Software Development KIT*) e APIs, consegue proporcionar uma programação eficiente, de código aberto, com várias funcionalidades e ainda assim tentando aproveitar ao máximo o processador do smartphone. Algumas destas funcionalidades são: um conjunto de *Views*, que inclui botões, listas, caixas de texto, *Content providers* usados para permitir as comunicações *inter-process*; *Resource Manager* que permitem à aplicação aceder a *strings*, *layout files*, etc; *Application Notifications* que possibilitam trabalhar com a barra de notificações; e a *Activity Manager* que gere e analisa a fase do ciclo de vida de uma aplicação [9].

2.2.2. Xamarin

O Xamarin é uma plataforma *open-source* usada para desenvolver aplicações modernas e com bom desempenho para iOS, Android e Windows usando .NET. Faz a gestão da comunicação do código partilhado com o código específico de cada plataforma [11].

O Xamarin permite que 90% do código seja partilhado entre plataformas, possibilitando que os programadores escrevam o código principal do projeto através do Xamarin.Forms [12], mas conseguindo um desempenho como se estivessem a escrever na linguagem nativa [11].

Uma abordagem de *cross-platform* apresenta baixos custos de desenvolvimento em comparação com o desenvolvimento nativo. A manutenção de aplicações fica mais acessível do que se tivéssemos de fazer a manutenção de vários produtos, cada um com a sua linguagem de programação. Pelo facto de se usar a mesma tecnologia no projeto todo, não é necessário termos especialistas em plataformas especificas [13].

A figura 2 mostra a arquitetura geral de uma aplicação *cross-platform* em Xamarin. O Xamarin permite que se crie uma interface de utilizador para cada plataforma e usar o C# para escrever a lógica do programa que é partilhada entre as plataformas [11].

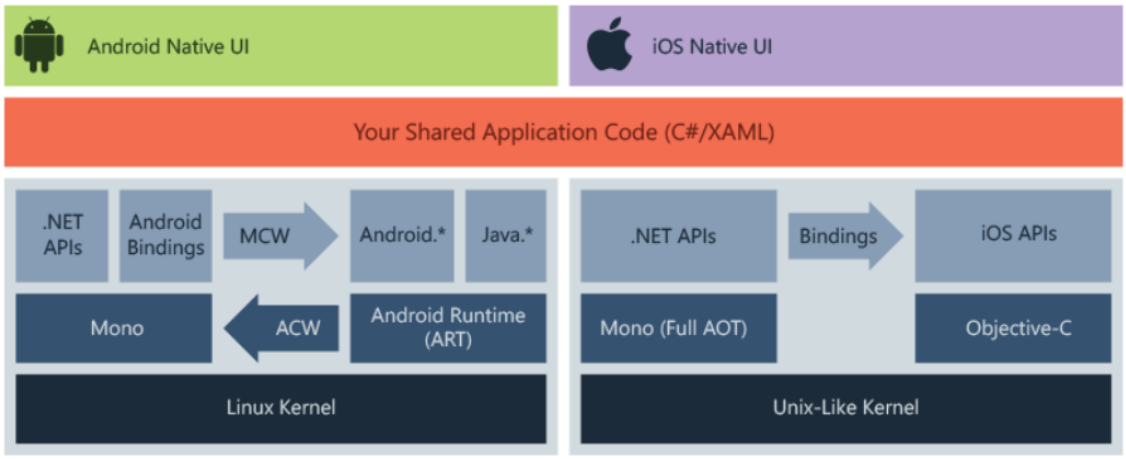

Figura 2- Arquitetura de uma aplicação cross-platform [10]

A arquitetura do Xamarin pode ser dividida em 6 *layers*: o *Data Layer* que guarda os dados no dispositivo móvel mesmo que fique sem bateria; o *Data Access Layer* que permite a interação com a base de dados; o *Business Layer que* permite executar tarefas, como, por exemplo, a autenticação de utilizadores a partir de uma certa localização; o

Service Acess Layer que permite a transferência de dados de uma tecnologia para outra, como, por exemplo, no uso de uma cloud; o *Aplication Layer* necessário para determinar para que dispositivo vai cada código; e por último o *User Interface Layer* que se encarrega da interface do utilizador [14].

2.2.3. Git-Version Control System

Version Control System é um sistema que guarda alterações de ficheiros ao longo do tempo, para que mais tarde seja possível aceder a versões destes ficheiros.

Este sistema é especialmente usado no desenvolvimento de software e *design* gráfico ou web, pois permite guardar todas as versões de uma imagem ou de um *layout*. Permite reverter ficheiros a versões antigas, projetos inteiros a versões antigas, comparar mudanças ao longo do tempo, comparar a versão atual com a versão de um parceiro de trabalho, fazer fusão entre versões e ramificações possibilitando o trabalho simultâneo entre pessoas e o trabalho em funcionalidades novas sem comprometer o estado dos ficheiros principais, ver quem fez a última modificação, identificar erros ou saber quem os identificou e quando, entre outras funcionalidades. Version Control Systems possibilitam recuperar ficheiros caso algo tenha acontecido com os ficheiros com que se trabalha localmente [15].

Neste projeto, o Version Control System usado é uma instância local do GitLab com autenticação, que garante que todos os projetos estejam seguros e sejam privados. Para além de ser uma maneira de guardar os projetos, esta ferramenta serviu como método de organização, permitindo observar o trabalho de todos os colaboradores do projeto, tal como programar o trabalho dos diferentes projetos. Na imagem em baixo podemos ver a arquitetura de um Version Control System [15]

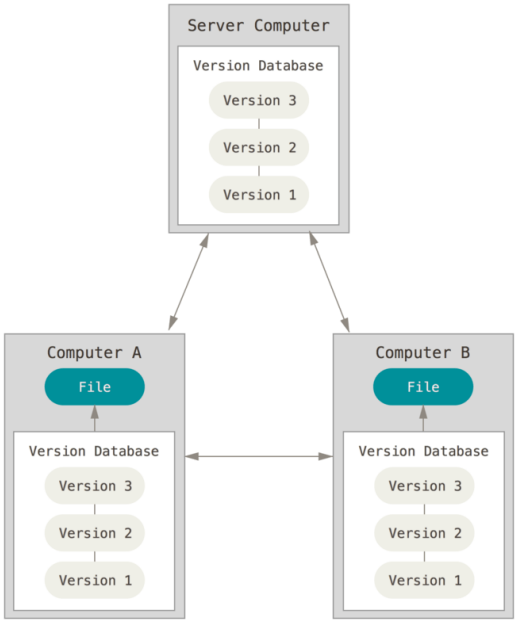

Figura 3 - Exemplo de um Version Control System [15]

2.2.4. FIWARE

FIWARE é um conjunto de especificações e módulos de software de um projeto europeu que providenciam *Application Programming Interface* (API)s, que tornam mais fácil o desenvolvimento de aplicações inteligentes em diferentes setores [16].

É uma *framework open source*, que combina componentes que permitem a comunicação entre múltiplos componentes de IoT, com o uso de APIs para troca e gestão de informação que permite a construção de soluções inteligentes. Da mesma forma, o FIWARE ainda disponibiliza serviços que podem ser integrados em projetos ou em outros serviços já existentes de maneira a melhorá-los e a complementá-los [17].

FIWARE é baseado numa biblioteca de componentes intitulados *Generic Enablers* (GEs) com o objetivo de implementar APIs. A partir das APIs destes GEs, os programadores tem acesso a diversas funcionalidades e conseguem colocar estas funcionalidades em prática, fazendo com que a programação seja muito mais fácil ao combinar os recursos fornecidos pelos GEs [16].

A Fundação FIWARE suporta a comunidade fornecendo recursos e validando tecnologias FIWARE. A comunidade de FIWARE é uma comunidade aberta e independente, com membros dedicados em concretizar a missão do FIWARE que consiste na construção de um ecossistema aberto e sustentável que seja público, sem custos e com padrões para plataformas de software orientados para a implementação que facilitam o desenvolvimento de novas aplicações inteligentes em diferentes setores [17].

2.2.5. Dash

Dash é uma biblioteca *open source* desenvolvida em Python para criar aplicações web reativas, e teve origem num protótipo público no GitHub em 2015 [18].

É uma *framework low-code* para construção rápida de aplicações de dados em Python, R, Julia, C# e MATLAB, construída em cima de Plotly.js e React.js, o que torna o Dash uma ferramenta ideal para quem trabalha com dados, possibilitando a construção e a instalação de aplicações de dados com interfaces de utilizadores adaptáveis[19].

Figura 4 - Exemplo de uma aplicação Dash [20]

As aplicações Dash são reproduzidas no *web browser*. Pode-se instalar a aplicação numa máquina virtual e depois partilhar a *app* a partir de URLs. Pelo facto de usar um *web browser* para ver a aplicação, o Dash é *cross-platform*, estando preparado para ser usado em dispositivos móveis[19]. Na figura 4 temos um exemplo de uma aplicação em dash [20].

2.2.6. Firebase

Firebase, desenvolvida pela Google, é uma plataforma de desenvolvimento de aplicações que é usado por milhões de organizações. Esta plataforma ajuda os desenvolvedores a criar, gerir e crescer as suas aplicações de uma forma fácil, rápida e segura. Não é necessário o uso de programação do lado do Firebase, para além disso, ainda fornece serviços que podem ser usados em aplicações de Android, IOS e Web [21].

Os serviços de autenticação de utilizadores do Firebase fornecem bibliotecas e SDKs fáceis de usar, de modo a autenticar utilizadores em aplicações. Assim, este sistema é usado de forma a que proteja os dados da aplicação [22].

2.2.7. Postman

Postman é uma plataforma para desenvolver, guardar, catalogar e usar APIs, simplificando todos os passos necessários para construir estas, de uma forma mais rápida e eficiente.

É uma aplicação usada para testar APIs, através de pedidos HTTP de forma a analisar a resposta recebida, não necessitando de nenhum trabalho extra para mandar ou receber os pedidos no Postman [23], ou seja, não é necessário escrever código de forma a ter um HTTP *web client*. Em lugar disto, para testar um API cria-se um conjunto de testes, designado de *collections*, e o Postman interage com a API.

2.3. Projetos Semelhantes

2.3.1. ISABELA

ISABELA, IoT Student Advisor and BEst Lifestyle Analyzer, foi um projeto desenvolvido por um grupo de pesquisa na Universidade de Coimbra, com o principal objetivo de implementar o conceito de HITLCPS num sistema, capaz de monitorizar os estudantes e de os ajudar a melhorar o seu desempenho académico[24].

O projeto inclui o desenvolvimento de duas aplicações Android, uma para *smartphone* e outra para *smartwatch*, a utilização da plataforma FIWARE e a utilização de dispositivos embebidos, como o Raspberry Pi e o Arduino. A aplicação de *smartphone* e *smartwatch* foi usada para obter informação sobre a rotina dos estudantes e os dispositivos embebidos para obter dados sobre o ambiente em casa e na universidade. A partir destes dados conseguiram inferir a atividade, localização, sociabilidade e padrões de sono. Para além disso, também usaram sensores virtuais/sociais,

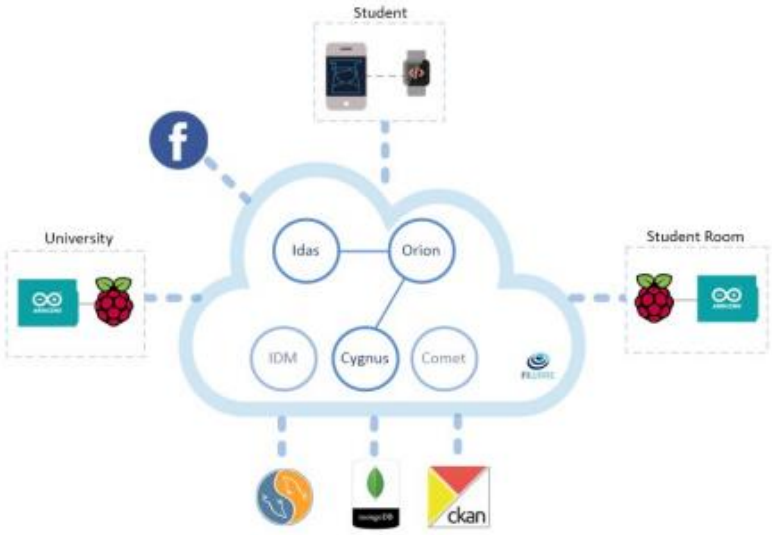

Figura 5 - Arquitetura do projeto ISASBELA [24]

nomeadamente o Facebook[24].

Na figura 5 pode-se ver a arquitetura deste projeto. Para implementar um sistema HITLCPS é preciso implementar três competências, aquisição de dados, inferência de estados e atuação. Neste projeto o *smartphone*, *smartwatch* e os dispositivos embebidos foram usados para recolher dados, o FIWARE como um modelo de *cloud* que implementa a capacidade de guardar os dados e da comunicação entre todos os dispositivos, o *smartphone* é onde acontece toda a inferência e a atuação é feita através de notificações e mensagens [24].

Até ao momento, foram feitos 2 estudos usando este sistema, um deles feito na Universidade de Coimbra com a duração de 30 dias e o outro estudo foi feito na Escuela Politécnica Nacional (EPN), *Quito*, *Ecuador*, com a mesma duração. Com estes testes aperceberam-se que era necessário melhorar as técnicas usadas para fazer a inferência do desempenho dos alunos, pois estas não foram fiáveis. Mas, mais importante que isso seria realizar um teste da aplicação em uma situação real em que os estudantes usassem o sistema ao longo de um semestre, para que existissem dados suficientes para fazer uma boa classificação do desempenho do aluno.

Desta maneira, este projeto foi a base e a inspiração para o iFriend tendo muitos aspetos em comum, nomeadamente o desenvolvimento de uma aplicação para *smartphone* com o intuito de recolher dados sobre a rotina dos utilizadores, o sistema de comunicação, o sistema de guardar dados e o objetivo de implementar um sistema HITLCPS. Por outro lado, os dois projetos têm as suas diferenças, particularmente na

área de atuação do projeto, onde o projeto iFriend é aplicado para a área da saúde e o projeto ISABBELA é aplicado na área do ensino. Ambos os projetos possuem formas distintas de obter parte dos seus dados e, por último, a avaliação que é feita pelo sistema HITLCPS do ISABELA é fornecida aos utilizadores deste sistema, enquanto no iFriend o principal objetivo é que a avaliação feita pelo sistema seja entregue aos médicos que acompanham os idosos do lar que vão servir de estudo a este projeto.

2.3.2. Sensor Glove for Interaction with a Nursing-Care Assistive Robot

Este estudo consistiu no desenvolvimento de uma luva com sensores para o controlo de *nursing-care robotic systems*, com o objetivo de tornar a interação entre o utilizador e os robôs de assistência mais intuitiva e confiável [25].

Para a construção da luva usaram, principalmente, sensores flexíveis, utilizado para detetar movimentos dos dedos do utilizador, e *inertial measurement unit*, que serviu para detetar movimentos do braço do utilizador. Em combinação com a luva, foram usados dois protótipos de *nursing-care robotic systems,* que consiste num *Automated Guided Vehicle* e num *off-the-shelf robot* (YuMi) [25].

Os principais problemas abordados por este estudo são:

1. A criação de um padrão para o *design* das luvas com sensores;
2. Realizar uma avaliação das luvas com os sensores flexíveis;
3. Um método de analisar *human in the loop systems* com a presença da luva com sensores;

Com este estudo, conseguiram desenvolver um *design* para as *Sensor Glove,* que estivesse integrada com os *nursing-care robotic systems,* de forma que esta fosse confiável e refletisse as intenções do utilizador ao controlar os robôs de assistência. A avaliação deste sistema integrado foi validada através dos resultados experimentais, tendo uma boa taxa de sucesso no controlo do robô [25].

Fazendo uma comparação entre os dois projetos, estes são semelhantes no aspeto em que ambos atuam na área da saúde e ambos tentam integrar o ser humano nos *Cyber-Physical Systems*. Por outro lado, este estudo utiliza sensores vestíveis para obter informação sobre o humano, algo que o iFriend não ambiciona fazer.

2.3.3. Happy System Architecture

Este estudo teve como objetivo implementar uma arquitetura que usa sensores para inferir emoções humanas, e usar esta informação para melhorar a vida do utilizador. O dispositivo de recolha de dados usado foi o *smartphone*, pois este é muito difundido e possui muitos sensores capazes de monitorizar o utilizador e o ambiente. Para fazer a inferência das emoções foi usada uma rede neuronal que recebe os dados dos sensores do *smartphone*, assim como recebe o feedback do utilizador de maneira a aumentar a precisão do sistema [26].

Neste contexto foram desenvolvidas 3 aplicações:

- **HappyWalk** – é um sistema de intervenção de mudança de comportamentos, que estima o humor do utilizador, de maneira a melhorar o seu estado físico e mental. Utiliza informação proveniente do microfone, acelerómetro e GPS do *smartphone* como *inputs* da rede neuronal, onde esta vai devolver uma destas 4 emoções: euforia, tédio, calma e ansiedade. O sistema considera o tédio e a

ansiedade como emoções negativas e quando isto acontece este sugere ao utilizador lugares que possivelmente goste[26].

- **HappySpeak** – Esta aplicação tem o intuito de ajudar os imigrantes. E para perceber as necessidades destes, foi realizada uma parceria com a Associação Fazer Avançar (AFA) e o seu programa SPEAK. O SPEAK é um programa cultural com o propósito de juntar as pessoas e promover a aprendizagem de novas línguas. Portanto, HappySpeak replica a plataforma SPEAK, permitindo que os utilizadores se registem e vejam futuros eventos na sua área de residência. Estes também são sugeridos aos utilizadores quando a rede neuronal, que recebe informação sobre mensagens, chamadas e a presença nos eventos SPEAK, deteta que estes se sentem sozinhos[26].
- **WeDoCare** – Esta aplicação tem o propósito de ajudar os refugiados, detetando se estes estão a ser vítimas de ataques violentos. A aplicação para *smartphone* recolhe dados do acelerómetro, GPS e microfone e usa o *Happy system* para fazer a inferência das emoções humanas para tentar prevenir este tipo de ataques, alertando a polícia e cidadãos que estejam nas proximidades quando os ataques são detetados[26].

Em suma, podemos observar que todos os 3 projetos e o iFriend têm aspetos em comum, tais como, o uso de um sistema HITLCPS, o objetivo de inferir e melhorar a saúde, assim como a utilização do *smartphone* para recolher dados. Em contrapartida, o iFriend recolhe dados de mais sensores do *smartphone*, usa redes WiFi para recolher ainda mais dados sobre os humanos, tem diferentes públicos-alvo, e ainda o *feedback* que é obtido pelo sistema iFriend é fornecido aos médicos que cuidam dos idosos e não diretamente às pessoas que usam o sistema.

3. iFriend

3.1. Descrição do projeto

Este projeto consistiu em criar um sistema baseado numa aplicação para *smartphone*, desenvolvida por mim, e num software para *router*s, explorado pelo aluno Guilherme Lemos, para a monitorização do estado de saúde de um idoso usando dispositivos sem fios e a tecnologia de rede WiFi.

As soluções que existem, baseadas em IoT e que medem parâmetros humanos físicos e psicológicos, passam, na sua maioria, por usar sensores vestíveis que são desconfortáveis, intrusivos e requerem que o utilizador tenha um cuidado extra quando usa o sensor, como por exemplo manter posições especiais para dormir e limitar o número de banhos. Por vezes os médicos pedem ainda aos pacientes que tenham um diário da atividade diária.

De outro modo, nos tempos de hoje já possuímos dispositivos que contêm sensores de dimensões reduzidas que possibilitam a sua integração nas atividades diárias. Estes dispositivos, tipicamente chamados de "*Wearables*", conseguem fazer medições do estado físico de quem os utilizar, por exemplo, através da medição da pulsação, da percentagem de oxigénio no sangue, da monitorização do sono, stress e atividade física realizada.

Outro aspeto destas tecnologias é a capacidade de ser possível detetar se a pessoa se encontra em movimento ou em repouso, e fazer medições nesta última situação. Isto é um ponto importante, pois as medições enquanto o utilizador está a movimentar-se são menos precisas. Então para complementar estes dispositivos usou-se uma técnica para obter a condição física do paciente a partir da rede WiFi.

Esta técnica foi desenvolvida para monitorizar a frequência cardíaca e respiratória, usando a informação de estado do canal (CSI) através da deteção de interferências causadas pelo corpo humano nas comunicações na rede WiFi.

Para além disto, recorremos a algumas funcionalidades do projeto ISABELA, desenvolvido na Universidade de Coimbra, e usamo-lo como meio de teste. O ISABELA tem como objetivo criar um sistema ciberfísico com reconhecimento humano (*Human in the Loop Cyber-Physical Systems*) apto a ajudar os estudantes a melhorar o seu desempenho académico. Com base neste sistema, criou-se uma aplicação para *smartphone* com vista a conseguir-se um *feedback* do utilizador e obter-se uma melhor perceção do estado mental e físico do paciente.

Resumindo, o iFrend tem como base dois pilares, a monitorização interativa não intrusiva de parâmetros biológicos de um utente com ou sem problemas de saúde, e a capacidade preditiva da variação destes parâmetros ao longo do tempo. A plataforma funcionará de forma continua e transmitirá informação em tempo real e a baixo custo.

3.2. Parceiros

O Projeto iFriend foi realizado por investigadores da Universidade de Coimbra em cooperação com 3 entidades espanholas, a Universidade de Alcalá, a Universidade de Extremadura e o Hospital Príncipe de Astúrias. Cada uma das entidades tem tarefas específicas para desenvolver, sendo que haverá sempre cooperação e comunicação entre todas as partes.

A Universidade de Coimbra desenvolveu o sistema do iFriend desde a aquisição de dados dos dispositivos, comunicação com a base de dados, dedução do contexto dos dados e do estado de saúde do doente, e a app.

Tanto a Universidade de Coimbra como o Hospital Príncipe de Astúrias e a Universidade de Alcalá irão fazer a validação do sistema, uns com pacientes não idosos e outros com pacientes idosos. Depois da validação laboratorial a Universidade de Alcalá e o Hospital Príncipe de Astúrias irão recolher dados num lar de idosos com insuficiência renal, e por fim a Universidade de Alcalá e a Universidade de Extremadura irão realizar a análise dos dados recolhidos.

3.3 Requisitos funcionais

Numa primeira fase, os principais requisitos funcionais do iFriend foram o registo dos parâmetros frequência cardíaca e frequência respiratória e, a partir destes, a inferência do estado de saúde do paciente. Para isso o desenvolvimento foi divido em dois módulos:

- **Módulo de Aquisição de Dados Humanos** – Utilizou-se os dados provenientes dos telemóveis, "*wearables*", e explorou-se a possibilidade de se usar mecanismos que detetem interferências causadas pelo corpo humano nas comunicações em WiFi. Usou-se uma ferramenta *open source*, o Atheros CSI, que permitia a leitura do "*Channel State Information*" que foi utilizado para obter os dados necessários.
- **Módulo de Inferência de Contexto Humano** – Tarefa que infere que dados estão a chegar ao sistema e como processá-los. Este módulo possibilita uma análise do contexto humano e ambiental de dados, de forma rigorosa e precisa para que se tenha a informação que pretendemos e para tomar decisões de controlo adequadas. Ou seja, depois de receber os dados, estes são filtrados e o sistema faz a verificação dos momentos que a pessoa esteve em movimento e em repouso. Depois, o sistema reencaminhará os dados para o algoritmo adequado para estimar as frequências cardíaca e respiratória.

Para além destes, existem outros aspetos que são importantes que uma aplicação possua, como por exemplo: uma **página inicial** onde se consiga ver o nome do projeto e de todas as entidades que contribuíram para o desenvolvimento da mesma; uma **página de configurações** para permitir que o utilizador altere as configurações; um sistema que **guarda os dados** temporariamente enquanto não há conexão à Internet; e um sistema de envio de dados para uma base de dados.

Estes requisitos podem estar sempre sujeitos a alterações ou à adição de novos requisitos ao longo do desenvolvimento do projeto.

3.4. Requisitos não funcionais

Os requisitos não funcionais são os que geralmente todas as aplicações de telemóvel devem suportar para que estas se apresentem completas e sem falhas.

Tal como foi referido no tópico anterior estes requisitos estão sujeitos a alterações e à adição de novos. Alguns dos requisitos não funcionais são a **segurança e a privacidade** dos dados de cada utilizador, o que é muito importante na área da saúde, a preocupação em fazer um ***design*** **da aplicação apelativo e fácil de se usar** (necessário devido à utilização desta aplicação por pessoas idosas), e a **acessibilidade** da aplicação,

ou seja, se todos os utilizadores são capazes de ter o acesso à aplicação a partir de diferentes plataformas.

3.5. Arquitetura

Nesta secção expõe-se qual é a arquitetura escolhida para o sistema e os seus componentes. Como mencionado anteriormente, este sistema inclui o ser humano no *loop,* como se pode ver na figura 6. Desta forma, o utilizador interage com o *smartphone* e *smartwatch* de uma maneira direta, e de uma maneira indireta através da rede WiFi. De seguida os dispositivos comunicam os dados recebidos com o servidor, que posteriormente são mandados para a Dash Web App e processados para serem

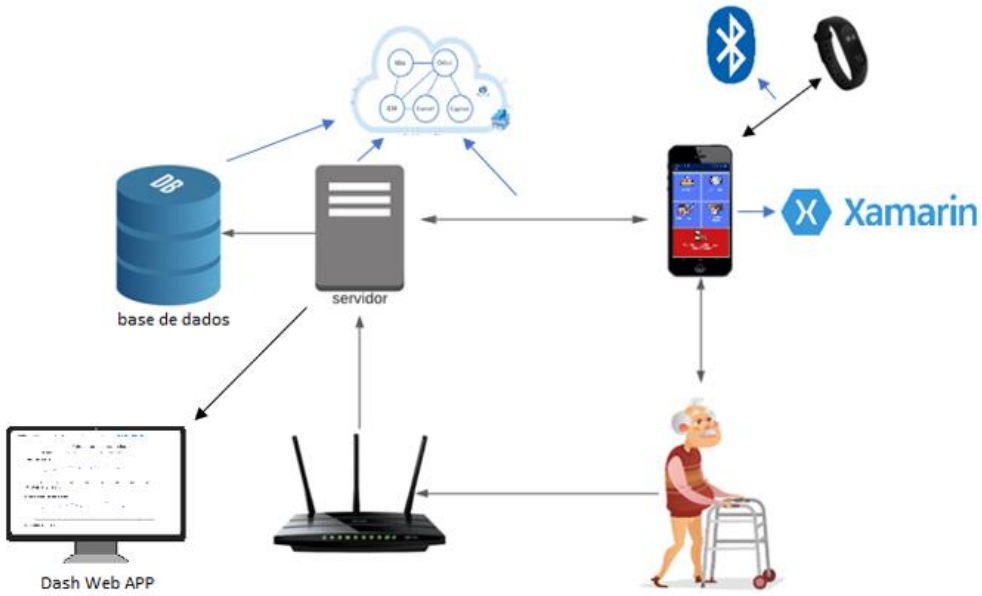

Figura 6 - Arquitetura do projeto iFriend

visualizados pelos médicos que cuidam dos idosos. Este projeto foi divido em partes, onde fiquei responsável por desenvolver a aplicação para smartphone, o estudante da Universidade de Coimbra, Guilherme Lemos, membro do grupo de trabalho, ficou responsável pela recolha de sinais vitais usando redes WiFi, a parte do FIWARE foi usado o sistema do projeto ISABELA desenvolvido pelo aluno de doutoramento da Universidade de Coimbra José Marcelo Fernandes e a parte do Dash Web App foi desenvolvida por mim em parceria com o estudante Guilherme Lemos.

3.5.1. FIWARE

Como especificado anteriormente, foram utilizadas algumas funcionalidades do projeto ISABELA, deste modo o sistema FIWARE montado para este projeto é o mesmo.

O FIWARE é o componente de *background* que funciona como armazenamento de dados e comunicação entre os diferentes dispositivos. Na figura 7 [24], podemos ver a arquitetura do FIWARE usada, que está dividida em vários blocos importantes para o funcionamento deste projeto. Os principais módulos usados neste projeto foram o ORION, o CYGNUS e o COMET.

O ORION é uma API que permite a criação de entidades virtuais para representar objetos do mundo real, que é importante para este sistema para criar uma conexão entre os dados dos sensores e a aplicação que os vai receber. Do mesmo modo, possibilita a gestão do contexto da informação, incluindo atualizações, *queries*, registos e subscrições [27].

CYGNUS é um conector encarregue de manter certas fontes de dados nas configurações de certos armazenamentos de terceiros como o MySQL, criando assim um histórico de dados [28]. Com este módulo consegue-se criar subscrições por atributo e quando uma entidade desse atributo é criada ou atualizada o CYGNUS salva essas alterações [24]. Assim, os dados recolhidos pelo sistema iFriend permanecem mais organizados na base de dados e torna mais fácil o acesso aos dados aglomerados pelos atributos.

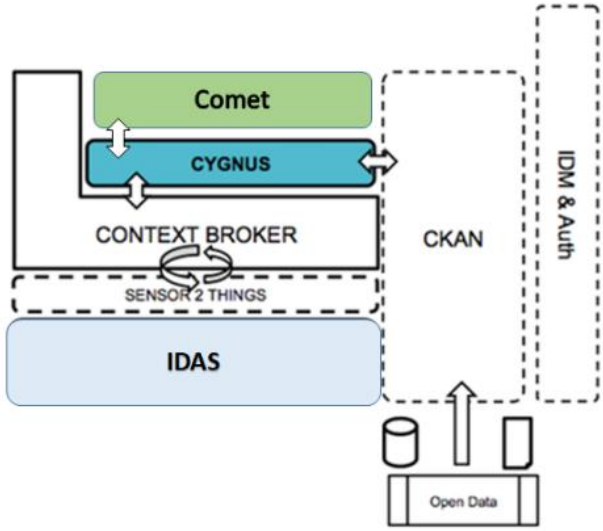

Figura 7 - Arquitetura FIWARE [24]

COMET é um componente que faz a gestão do histórico de dados brutos ou a agregação de dados com um contexto de tempo[29]. A API usada para comunicar com aplicações externas permite agregar os dados por tempo e até mesmo por intervalo de tempo[24]. Este módulo é usado para enviar dados do *smartphone* para o sistema FIWARE e do sistema FIWARE para a Dash Web App para depois estes dados serem processados e mostrados nesse site.

3.5.2. Aplicação de smartphone

A aplicação de *smartphone* é constituída por duas tarefas importantes, uma delas corresponde a tudo o que se relaciona com a parte gráfica da aplicação, *User Interface* (UI), e a outra tarefa são todos os serviços que correm no *background* da aplicação que fazem com que esta funcione como é pretendido.

A tarefa UI não pode estar muito sobrecarregada com trabalho, pois precisa de estar disponível para interagir com o utilizador da aplicação. Desta forma, normalmente está encarregue com pequenas tarefas como mudar definições, mostrar interfaces gráficas e por vezes guardar dados na base de dados.

As tarefas em *background* gerem o trabalho mais complicado, como a recolha e o processamento de dados, pedidos HTTP, envios de dados para servidores, entre outras tarefas que sejam de computação mais intensa.

Esta aplicação tem essencialmente 3 tarefas a correr em background, como se pode ver na figura 8. Tem uma tarefa para ajudar na gestão do UI, a tarefa SocialiteSensors que se encarrega de recolher dados dos sensores do *smartphone* e guarda-os na base de dados, e a última, a tarefa da Comunicação FIWARE que faz o envio dos dados recolhidos pela tarefa SocialiteSensors para o servidor. Todas estas tarefas serão explicadas no capítulo 4.6.

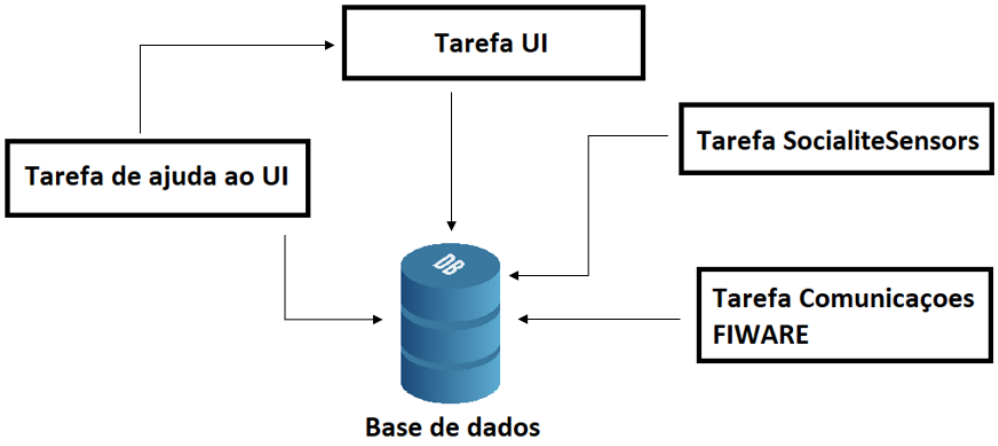

Figura 8 - Arquitetura da Aplicação de Smartphone

3.5.3. Redes WiFi

É possível a utilização de dispositivos WiFi para fazer uma monitorização passiva dos sinais vitais, como o batimento cardíaco e o ritmo respiratório, deixando de ser essencial o uso de vestuário com sensores para o fazer. O sistema implementado examina a informação do estado do canal, CSI, com a finalidade de detetar os

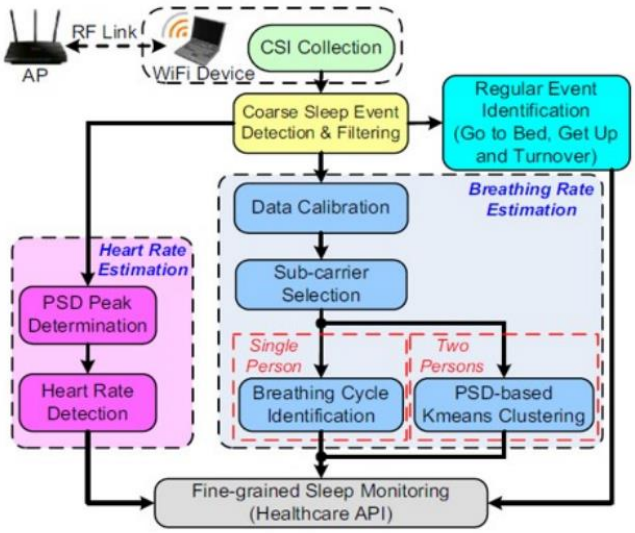

Figura 9 - Esquema do sistema de estimação de sinais vitais [30]

movimentos causados pela respiração e batimento cardíaco [30]. Esta tecnologia baseia-se no efeito que o corpo humano tem na propagação dos sinais utilizados em comunicações WiFi. A presença de objetos estáticos causa a reflexão do sinal. Em contrapartida, o corpo humano origina um aumento nos caminhos de propagação devido ao espalhamento dos sinais [31].

A configuração que foi usada para obter o CSI, foram dois repetidores WiFi, um que está ligado à WiFi local e o outro que funciona como cliente ligado ao primeiro repetidor. Nos testes realizados em laboratório os repetidores estavam a uma distância de 2 metros aproximadamente, mas num cenário real estarão em cantos opostos de uma divisão.

3.5.3. DASH

O Dash tem uma estrutura bastante simples, é composto essencialmente por duas partes. A primeira onde se define a interface gráfica da aplicação com componentes de html de uma biblioteca chamada dash_html_components, e com o uso de componentes da biblioteca dash_core_components é possível fazer facilmente uma interface com gráficos, barras de pesquisa, escolhas de datas entre outras funcionalidades.

```
import pandas as pd

df = pd.read_csv('https://raw.githubusercontent.com/plotly/datasets/master/gapminderDataFiveYear.csv')

app = dash.Dash(__name__)

app.layout = html.Div([
    dcc.Graph(id='graph-with-slider'),
    dcc.Slider(
        id='year-slider',
        min=df['year'].min(),
        max=df['year'].max(),
        value=df['year'].min(),
        marks={str(year): str(year) for year in df['year'].unique()},
        step=None
    )
])


@app.callback(
    Output('graph-with-slider', 'figure'),
    Input('year-slider', 'value'))
def update_figure(selected_year):
    filtered_df = df[df.year == selected_year]

    fig = px.scatter(filtered_df, x="gdpPercap", y="lifeExp",
                     size="pop", color="continent", hover_name="country",
                     log_x=True, size_max=55)

    fig.update_layout(transition_duration=500)

    return fig


if __name__ == '__main__':
    app.run_server(debug=True)
```

Figura 10 - Exemplo de código de uma aplicação em Dash [32]

A segunda parte da aplicação consiste em *callback* que tornam a aplicação reativa à interação do utilizador. As *callbacks* recebem informação dos Inputs e fazem o processamento dos dados para que sejam depois mandados para o devido *Output*.

Nas figuras 10 e 11 pode-se ver um exemplo de uma aplicação feita em Dash com o seu correspondente código[32].

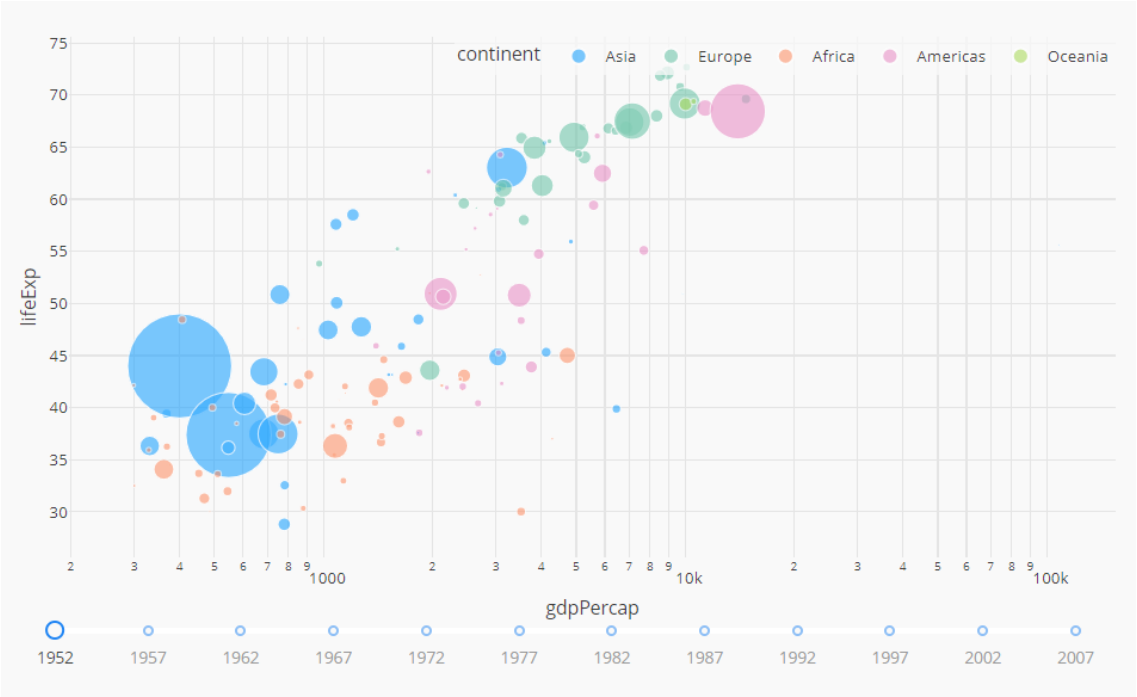

Figura 11 - Exemplo de aplicação Dash [32]

4. Trabalho realizado e considerações

4.1. Trabalho realizado

Uma vez que as tecnologias e software utilizados neste projeto não foram lecionadas nas unidades curriculares, foi necessário um período de adaptação e de aprendizagem de novos conceitos.

Esta aprendizagem foi feita através de cursos online e desafios propostos pelos orientadores, onde se ficou a perceber o funcionamento e como se programa em Android e em Xamarin que foram as bases para desenvolver a aplicação para o *smartphone*, figura 12.

Depois fez-se um orçamento de dispositivos para serem comprados para os idosos usarem no lar, figura 13. Com um limite monetário imposto pelos nossos parceiros tínhamos de comprar um *smartphone*, um *smartwatch* e quatro *routers*. Estas tabelas possuem informações sobre diversos dispositivos, permitindo fazer escolhas consoante as nossas limitações e as características que queremos em cada dispositivo. Assim, os equipamentos escolhidos pelo nosso grupo para os parceiros adquirirem para os idosos foram: o *smartphone* Xiaomi Redmi 9C, o *router* TP-Link Archer C6 V2 e o *smartwach* TicWatch Pro 3 GPS.

No fim deste tempo ainda se criou um *mockup* para a aplicação de smartphone com o objetivo de recolher dados sobre a rotina dos utilizadores, como se pode ver na figura 14. Para o desenvolvimento do *mockup* usou-se a ferramenta Mockitt [33], que permite a criação de protótipos de *User Interface* (UI) em diferentes plataformas. O *mockup* foi apresentado tanto aos orientadores como aos parceiros do projeto, para validação do mesmo.

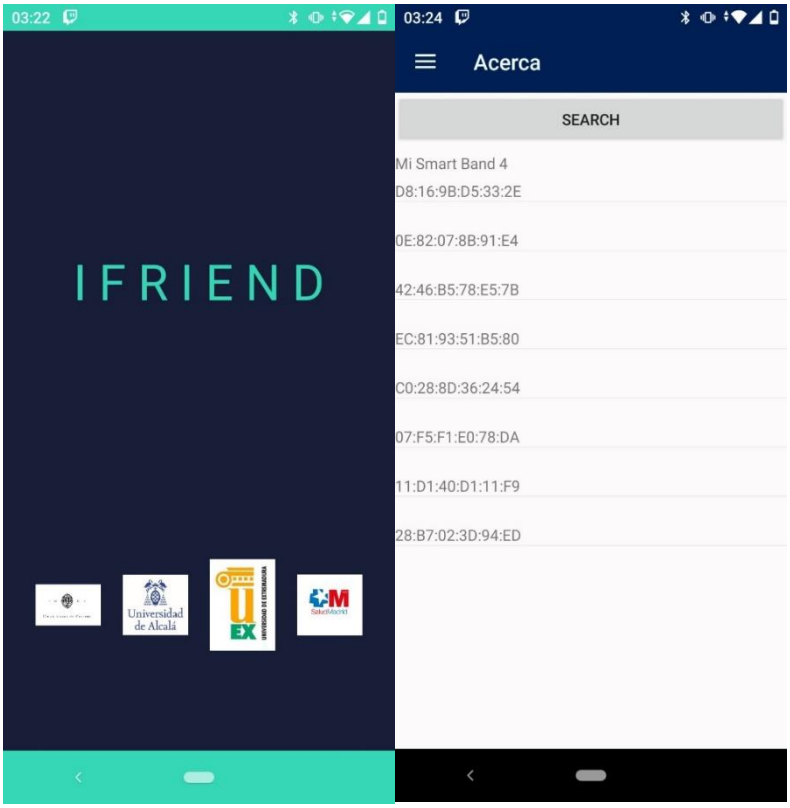

Figura 12 - Aplicação criada durante a aprendizagem

	Sistema Operativo	Suporta Spo2	Hear-rate sensor	Acelerómetro	Giróscopio	GPS	Barómetro	NFC	Bateria	Preço
TicWatch Pro 3 GPS	Wear OS	Sim	Sim	Sim	Sim	Sim	Sim	Sim	72h	299,99 €
Fossil Sport	Wear OS	Não	Sim	Sim	Sim	Sim	Não	Sim	24h	170,99 €
Skagen Falster 2	Wear OS	Não	Sim	Sim	Sim	Sim	Não	Sim	24h	119,60 €
Ticwatch E2	Wear OS	Não	Sim	Sim	Sim	Sim	Não	Não	2 days	152,43 €
Ticwatch S2	Wear OS	Não	Sim	Sim	Sim	Sim	Não	Não	2 days	120,97 €
Huawei GT 2e	HarmonyOs	Sim	Sim	Sim	Sim	Sim	Sim	Não	14 days	141,19 €
Samsung Galaxy Watch Active 2	Tizen-based wearable OS 4.0	Não	Sim	Sim	Sim	Sim	Sim	Sim	131h	220,40 €

Figura 13 - Uma das tabelas criadas para o orçamento

Figura 14 - Mockups da aplicação para smartphone

4.1.1. APP Smartphone

4.1.1.1. Aquisição de dados

Nesta aplicação existem dois tipos de dados a serem recolhidos: dados sobre a rotina/sintomas dos utilizadores e dados dos sensores do smartphone.

Os dados sobre a rotina dos utilizadores foram recolhidos através dos botões do *layout* da aplicação com o objetivo de serem mostrados em mais detalhe na secção 4.6.1.4. Estes botões tem o propósito de serem usados pelo utilizador ao longo do seu dia, carregando no botão correspondente à atividade que estiverem a fazer ou à patologia que estiverem a sentir. Desta forma, teremos acesso à hora em que cada atividade foi realizada, obtendo um registo da sua rotina e possíveis sintomas que ocorram ao longo do dia, que anteriormente estava a ser feita aos idosos do lar, através de questionários impressos, no final do dia.

```
"id": "8OO4qj93yoW0VptBj3j9m3kg5511",
"type": "ifriendevent",
"event": {
    "type": "Text",
    "value": [
        {
            "e": "ChestPain",
            "t": "2022-06-26T04:04:44Z"
        }
    ],
    "metadata": {}
},
```


Figura 15 - Exemplo de dados recolhidos pelos botões da aplicação

Os dados dos sensores do *smartphone* foram recolhidos através da biblioteca SocialiteSensors desenvolvida em conjunto com o grupo de trabalho, que ajudou na criação da biblioteca para o sistema operativo iOS. Esta tem como objetivo a recolha de dados em *background* a partir do maior número de sensores que estejam presentes no *smartphone* e que seja permitido pelo seu sistema operativo.

Dado que, os sistemas operativos IOS e Android possuem diferentes métodos para realizar a recolha de dados dos sensores presentes no *smartphone* e diferentes procedimentos no desenvolvimento e gestão dos serviços em *background*, foi imprescindível programar cada uma destas tarefas em cada um dos sistemas operativos. De forma semelhante, existem bibliotecas como a Xamarin.Essentials [34] que permitem a recolha de dados de alguns sensores, mas o número de sensores é limitado e o controlo sobre o tempo de recolha, quantidade, entre outros parâmetros também são limitados.

Para se conseguir um serviço em *background* que permaneça sempre ativo mesmo quando a aplicação está desligada, foi preciso usar um tipo especial de serviço chamado Foreground Service [35]. Este tipo de serviço consegue funcionar mesmo que a aplicação se encontre minimizada ou até mesmo fechada, tendo recursos do *smartphone* suficientes para que a tarefa seja executada devidamente e que o utilizador tenha um acesso rápido à aplicação, pois, este serviço deve fornecer uma notificação que o Android mostra enquanto o serviço estiver a correr, tendo assim uma prioridade maior que um serviço normal [35].

Como foi referido anteriormente, tentou-se recolher dados do maior número de sensores do *smartphone*. A figura 16 mostra todos os dados dos sensores que foram recolhidos com esta biblioteca. Alguns dos dados foram recolhidos diretamente através de APIs do Android, enquanto outros dados foram recolhidos usando APIs de terceiros.

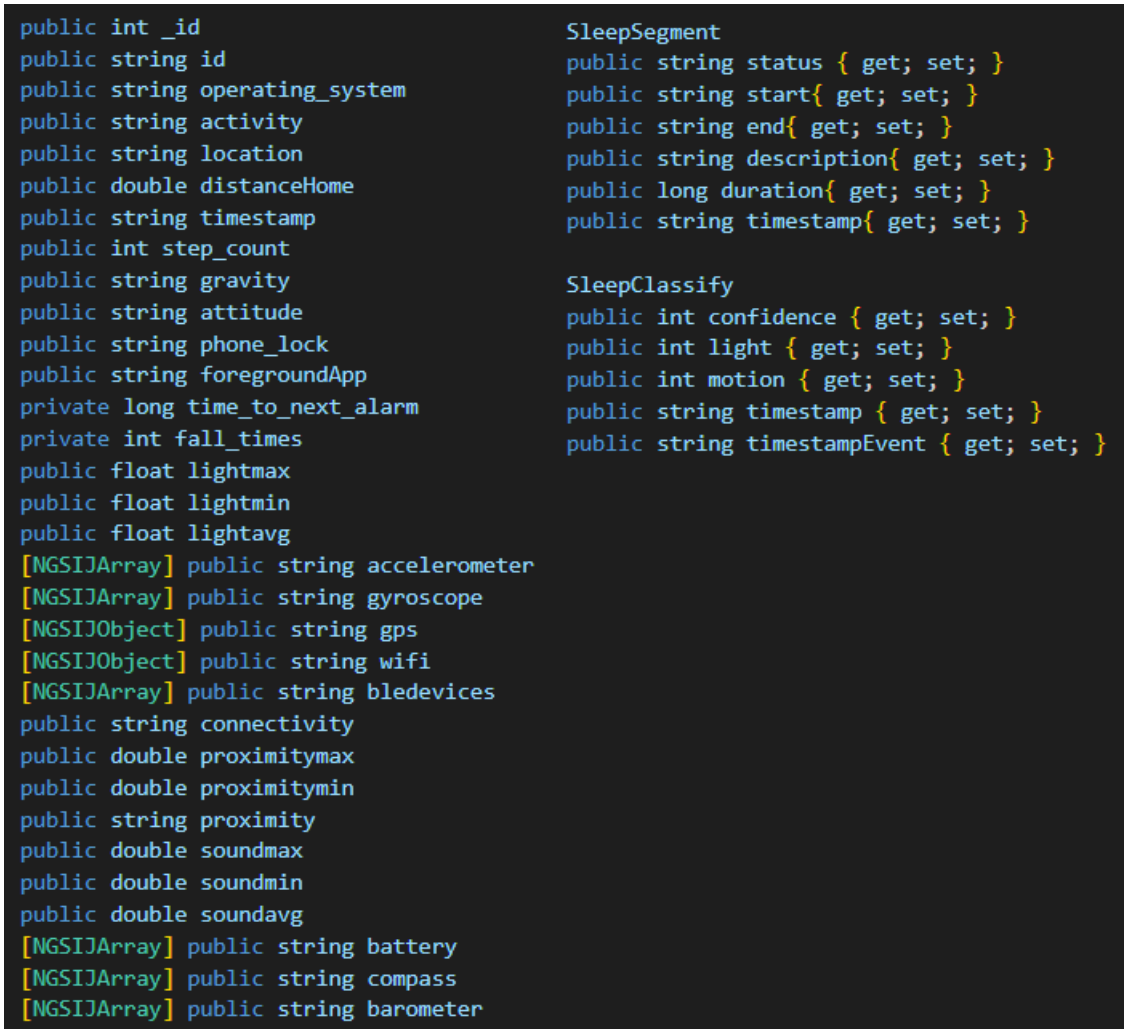

Figura 16 - Dados recolhidos pela biblioteca SocialiteSensors

Para obter o sistema operativo do *smartphone* em que a aplicação está a funcionar, foi utilizado a Device *class* do Xamarin.Forms [36], que contém um número de propriedades e métodos que ajudam a parametrizar o *layout* de acordo com sistema em que a aplicação está a ser instalada. Com esta classe conseguimos determinar se o sistema operativo é IOS ou Android.

A atividade e o sono são recolhidos pela ActivityRecognitionClient *class* que pertence à GoogleAPI desenvolvida pela Google [37]. Para ambas, é feito o pedido para que se façam as devidas atualizações da atividade e do sono, e a API chama um serviço que trata os dados quando existirem novos dados para serem recolhidos.

A localização e a distância à casa são obtidas e calculadas a partir da recolha do WiFi e do GPS, a obtenção destes valores vai ser explicada em mais detalhe na secção 4.6.2. A distância à casa é calculada em metros e a localização pode assumir dois valores "*Home*", quando o utilizador está em casa, ou "*Unknown*", quando o utilizador está fora de casa.

A aquisição do *timestamp* foi conseguida usando a estrutura DateTime que pertence ao Xamarin, que permite a manipulação de datas e horas, assim como a aquisição destas no dispositivo em que estiver a correr a aplicação [38].

A biblioteca Android.Hardware [39] dá-nos acesso à recolha de dados de vários sensores, SensorType [40], onde são posteriormente escolhidos aqueles que seriam

mais importantes, que neste caso seriam a intensidade da luz detetada pelo *smartphone*, a proximidade que o *smartphone* está de um objeto ou pessoa e por último a quantidade de passos que foram dados pelo utilizador e foram registados pelo *smartphone*. Para se obter os dados destes sensores é preciso ir criar uma instância de cada sensor e registá-los com a classe SensorManager [41], quando o serviço recebe dados destes sensores um evento é acionado que é capturado por uma função que processa esses dados.

A informação que indica se o smartphone está bloqueado ou não, foi obtida a partir da KeyguardManager *Class* [42], que contém uma propriedade que informa sobre este estado.

ForegroundApp corresponde ao nome da aplicação que estiver em primeiro plano. Conseguiu-se obter o nome dessa aplicação através do uso da classe UsageStatsManager [43], que fornece acesso ao histórico e estatísticas de uso do *smartphone*. Assim, com esta classe fez-se uma recolha de eventos num certo intervalo de tempo, tendo posteriormente de se verificar qual é o último evento que indique que a aplicação se moveu para primeiro plano.

Apesar do seu uso limitado, decidiu-se usar o Xamarin.Essentials [34] para conseguir obter dados de alguns sensores, pois este permite acesso aos dados de sensores como o giroscópio, acelerómetro, bússola, barómetro, conexão à Internet e leituras da bateria do *smartphone*. Foi necessário registar a função que processa os dados de cada um dos sensores, para que esta seja chamada quando ocorrer um evento desse tipo de sensor e para que comece a monitorização de mudanças dos sensores escolhidos.

A biblioteca SocialiteSensors também faz uso do microfone do *smartphone* para retirar amostras do ruido do ambiente em que o utilizador se encontra. Não é guardado nenhuma amostra do áudio recolhido, só se recolhe a amplitude máxima gravada naquele intervalo de tempo, para se ter perceção se o utilizador está ou não num ambiente com barulho. Para se recolher estes dados foi usada a MediaRecorder *class* [44] da biblioteca base do Android do Xamarin, Xamarin.Android [45], onde é exequível começar e parar a gravação de áudio, ficando guardada numa propriedade da classe o valor da amplitude máxima dessa gravação.

O Xamarin.Android, para além de fornecer a recolha de informação do microfone do *smartphone*, também permite o acesso à classe LocationManager e aos serviços de localização [46], que através da sua implementação permite ter acesso à localização GPS e a atualizações na localização GPS do utilizador.

Através da classe WifiManager [47] é possível fazer um *scan* de todas as redes WiFi que o smartphone consegue encontrar. Quando o *scan* terminar, um BroadcastReceiver [48], que recebe transmissões do sistema operativo Android, vai receber esses dados que depois serão tratados de maneira a obter-se as informações pretendidas, como a WiFi que o *smartphone* estiver conectado, o BSSID e a força de sinal de cada rede.

De forma semelhante à recolha de WiFi, também se faz a recolha dos dispositivos BLE (Bluetooth Low Energy) que o *smartphone* consegue alcançar. Foi usado a biblioteca Puglin.BLE [49] para fazer esta recolha, onde se guarda o nome e o *MAC Address* dos dispositivos BLE.

É de se notar que, pelas políticas do Android e do IOS, é preciso pedir permissão aos utilizadores para utilizar os dados de maior parte destes sensores. A figura 22 na secção 4.6.1.4 é um exemplo da aplicação a pedir ao utilizador permissão para usar a sua localização.

4.1.1.2. Comunicação

4.1.1.2.1. Interna

Nesta secção explica-se a comunicação que acontece dentro da aplicação de *smartphone*. Esta normalmente acontece entre a tarefa de UI e as tarefas em *background*.

Tal como foi mencionado anteriormente, no Xamarin não se consegue desenvolver tarefas em *background* com código que funcione simultaneamente para as duas plataformas, Android e IOS. Desta maneira, as tarefas em *background* tiveram de ser realizadas separadamente, cada uma adaptada as restrições de cada sistema. Isto torna a comunicação entre a tarefa UI e as tarefas em *background* mais complexas.

Com vista a ultrapassar este problema, usaram-se dois métodos para fazer esta comunicação.

O primeiro método envolveu o uso da base de dados para guardar os dados que são recolhidos pela aplicação, que por sua vez vão ser acedidos pelas tarefas em *background* que vão tratar e enviar estes dados para o servidor, como por exemplo os dados que são recolhidos com os botões das rotinas são guardados na base de dados que, em seguida, são acedidos por uma tarefa em *background* que os envia para o servidor.

O segundo método consiste no uso de um *nugget package* chamado Xamarin.Essentials, que fornece APIs para todas a plataformas de aplicativos móveis [34]. Esta biblioteca tem uma classe chamada *Preferences* que armazena valores de preferências do aplicativo num repositório de chave/valor [50]. Um exemplo do uso desta classe na aplicação é o caso dos botões que só podem ser carregados uma vez por dia, estes ficam guardados nas preferências quando são pressionados e no final do dia a tarefa em *background* responsável por dar *reset* aos botões remove estas preferências voltando a ser possível carregar nos botões.

4.1.1.2.2. FIWARE

Os principais usos do FIWARE nesta aplicação foram o armazenamento dos dados recolhidos e a comunicação entre a aplicação e o servidor. Para fazer esta ligação, foi usada a biblioteca RestFiwareAPI desenvolvida por membros do grupo de trabalho.

Esta biblioteca tem como função principal, o envio dos dados, em *background*, que estão armazenados na base de dados do *smartphone* para a FIWARE. Para se conseguir isto, a biblioteca começa por fazer um *GET request*, que é um *REST request* para receber informações, para verificar se a identidade do utilizador já existe. Se não existir é efetuado um *POST request* para criar a identidade.

Posteriormente à verificação de identidade, começa o serviço principal da biblioteca. Este vai buscar os dados armazenados na base de dados do *smartphone*, itera por todos os dados a enviar e faz um *PATCH request* para atualizar os valores das entidades no FIWARE. No fim, apaga os valores enviados da base de dados, mas só depois de receber a confirmação da FIWARE de que estes foram recebidos e guardados.

4.1.1.3. Base de dados

A base de dados do *smartphone* nunca teve o intuito de guardar dados por muito tempo. Esta base de dados tem o objetivo de guardar os dados até que seja possível fazer conexão com o FIWARE, prevenindo assim a perda de dados.

Para criar e aceder à base de dados usamos o *nugget package* sqlite-net-pcl [51], tornando fácil trabalhar com a base de dados. Para isto, foi preciso criar classes com o formato pretendido para guardar os dados na base de dados, figuras 17, 18, 19 e 20, criar os métodos da base de dados, que incluem a criação da base de dados com o caminho da mesma e métodos para ir coletar/salvar/destruir dados nesta. Por fim, só temos de usar estes métodos para criar a base de dados e gerir os dados presentes nesta.

Este projeto tem 4 tipos diferentes de tabelas para os diferentes dados que são recolhidos: SmartphoneData que guarda todos os dados dos sensores que são obtidos na biblioteca SocialiteSensors, figura 17, SleepClassify e SleepSegment que guardam os dados do sono adquiridos a partir da Google API Activity Recognition Sleep, figura 18 e 19, e a DBItem com os dados recebidos dos botões das rotinas, figura 20.

Figura 17 - Dados da tabela SmartphoneData

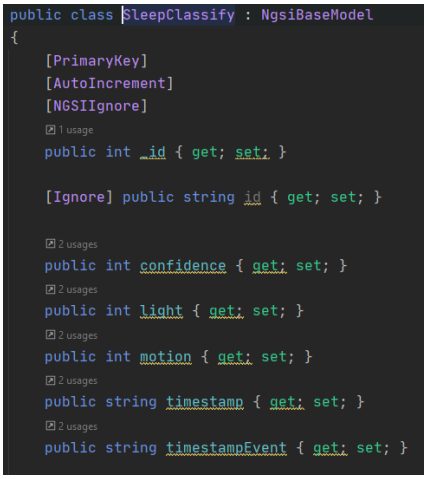

Figura 18 - Dados da tabela SleepClassify

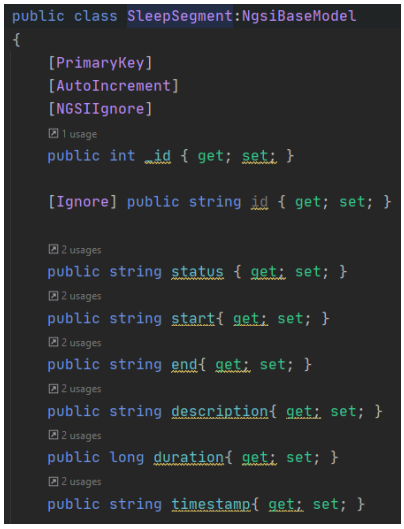

Figura 19 - Dados da tabela SleepSegment

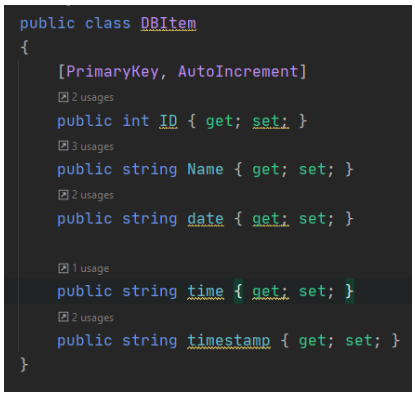

Figura 20 - Dados da tabela DBItem

4.1.1.4. APP design

Esta secção mostra a aparência da aplicação e como funciona a navegação pela mesma. É importante ter em conta que esta aplicação foi desenvolvida com o intuito de ser usada por pessoas com mais idade, por isso tentou-se que a aplicação fosse o mais simples possível usando botões grandes, imagens e textos com letra grande.

Depois do utilizador abrir a aplicação, o primeiro ecrã que aparece é um ecrã de introdução, que dura poucos segundos, onde aparece o nome da aplicação, iFriend, e todas as entidades que fazem parte deste projeto, figura 21. Se for a primeira vez que a aplicação é lançada vão aparecer janelas para aceitar permissões, como por exemplo, a de uso da localização, para que os serviços da aplicação consigam desempenhar as suas funções, figura 22. Para além disso, assim que a aplicação é lançada todos os serviços em *background* que foram especificados anteriormente também começam.

Figura 21 - Página de introdução

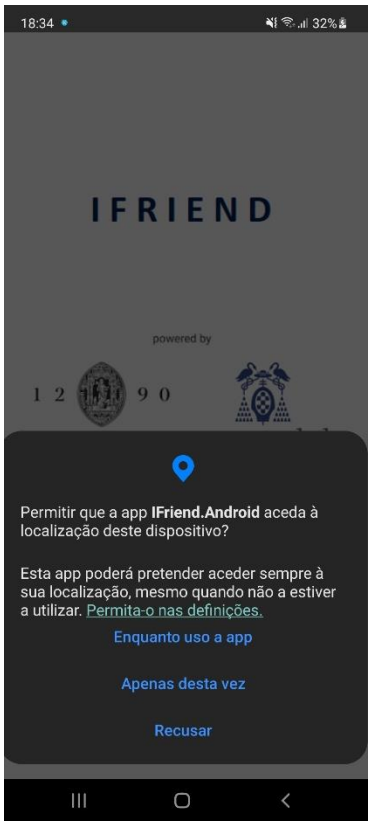

Figura 22 – Permissões da aplicação

Passados alguns segundos o ecrã de introdução desaparece, e dá lugar à página de *login* da aplicação, figura 23. Aqui o utilizador tem de introduzir as suas credenciais para conseguir entrar na aplicação. Se ainda não tiver conta, têm a possibilidade de criar uma conta diretamente na aplicação, figura 24.

Para fazer esta autenticação, usou-se a autenticação do Firebase [52] que disponibiliza uma API que de muitas funções, permite guardar, criar utilizadores e verificar se estes existem.

Assim, no ecrã de *login*, depois do utilizador introduzir as suas credenciais e carregar no botão de *login*, é chamada a função de verificação de utilizador da API do Firebase que recebe como parâmetro de entrada o *email* e a *password* da conta. Caso esta função retorne que os parâmetros estão corretos, a aplicação permite ao utilizador prosseguir para o resto das páginas da aplicação. Caso a função retorne que algo está incorreto a aplicação avisa o utilizador para que este possa corrigir as suas credenciais ou criar uma conta nova caso ainda não exista conta com aquelas credenciais, figura 25. Da mesma forma, a página de registo tem o mesmo esquema que a página de login, só que esta pede à base de dados de utilizadores do Firebase para criar um novo utilizador.

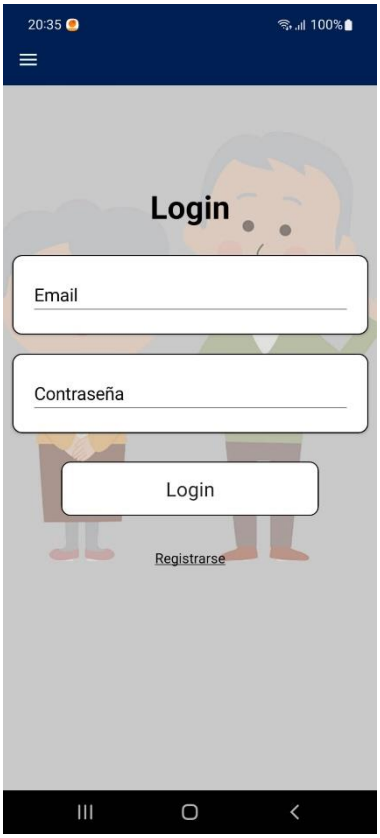

Figura 23 – Página de Login

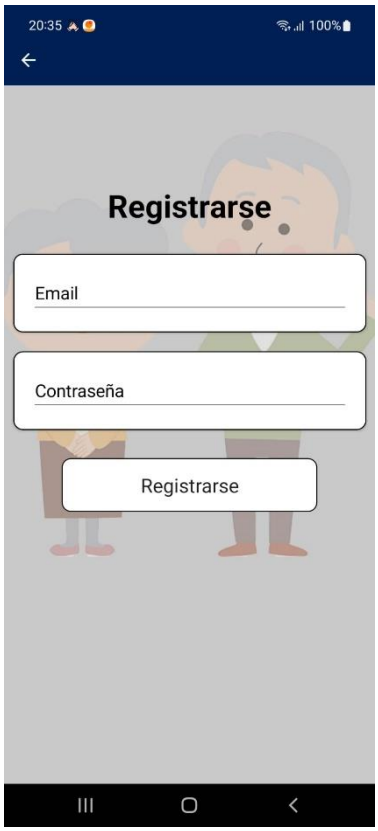

Figura 24 - Página de Registo

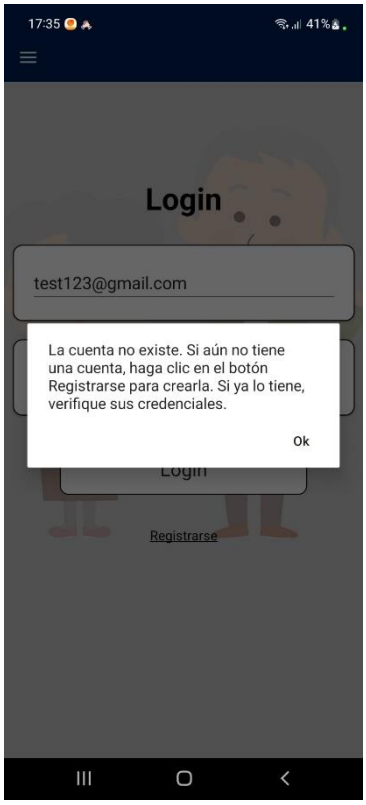

Figura 25 - Mensagem de erro de login

Feito o login na aplicação, o ecrã principal, figura 26, aparece. Esta página serve como meio de navegação para o resto das páginas da aplicação. Como podemos ver na

Figura 26 - Ecrã principal da aplicação

figura 26. Esta página é constituída por 5 botões que representam as principais informações a obter do dia a dia dos idosos.

Logo que este ecrã principal começa, fazemos uma verificação se já obtivemos os dados do WiFi de casa para ser usado no serviço SocialiteSensors, como explicado anteriormente. Se ainda não tivermos esta informação uma janela irá abrir onde se pergunta ao utilizador se pretende nos fornecer esses dados, figura 27.

Figura 27 - Notificação para perguntar sobre a WiFi de casa

Se o utilizador aceitar, este é redirecionado para a página das definições, figura 28, onde nos consegue dar a informação sobre a WiFi de casa, como mudar a linguagem entre português, inglês e espanhol, e especificar se quer enviar os dados para o servidor só com o WiFi ou sempre.

Figura 28 - Página das definições

Ao carregar em cada um dos botões do ecrã principal, o utilizador é redirecionado para a página correspondente àquele botão. Cada um dos botões dentro de cada página recolhe o dia e a hora a que foram carregados, e é assim que fazemos parte da monitorização da rotina dos idosos.

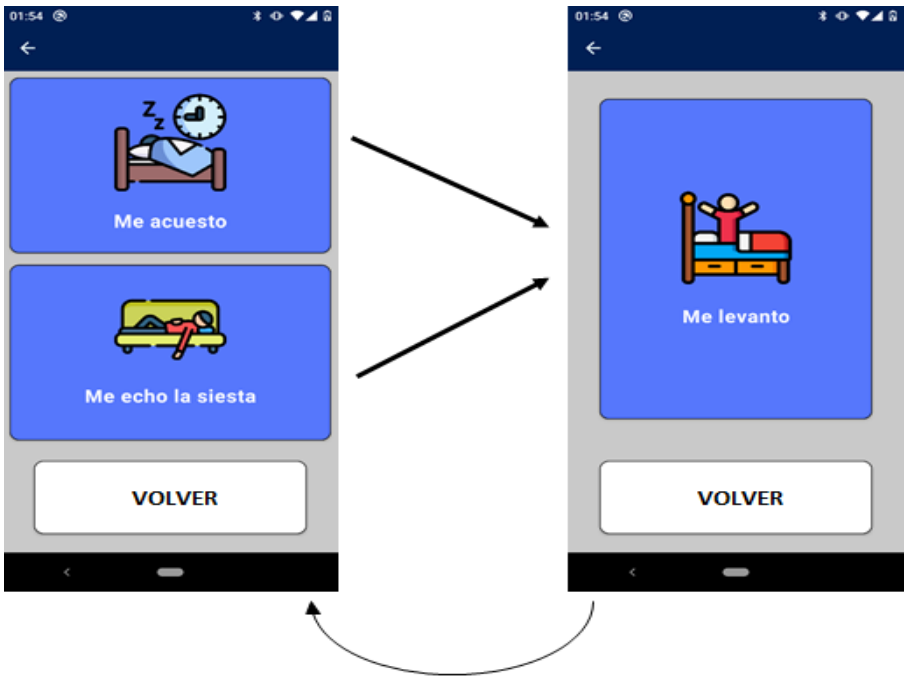

Figura 29 - Design e funcionamento da página do sono

Figura 30 - Página sono depois do primeiro botão ser usado

O primeiro botão encaminha o utilizador à página de sono, onde recolhem a hora de deitar e de levantar do utilizador. Como podemos ver na figura 29 do lado esquerdo tem dois botões, um para ser pressionado quando o utilizador vai dormir à noite e outro quando vão fazer uma cesta. Ambos os botões levam a uma página com um botão para ser carregado quando o utilizador se levanta. O primeiro botão de dormir tem uma diferença em relação ao segundo, que é de só poder ser pressionado 1 vez por dia aparecendo um visto em cima do botão, figura 30.

O botão das Comidas/Refeições leva à página onde recolhemos as horas das 3 refeições mais importantes do dia, como podemos ver na figura 31. Quando o utilizador carrega num botão o visto aparece, mostrando que já fez aquela refeição e não o permite carregar outra vez no mesmo botão.

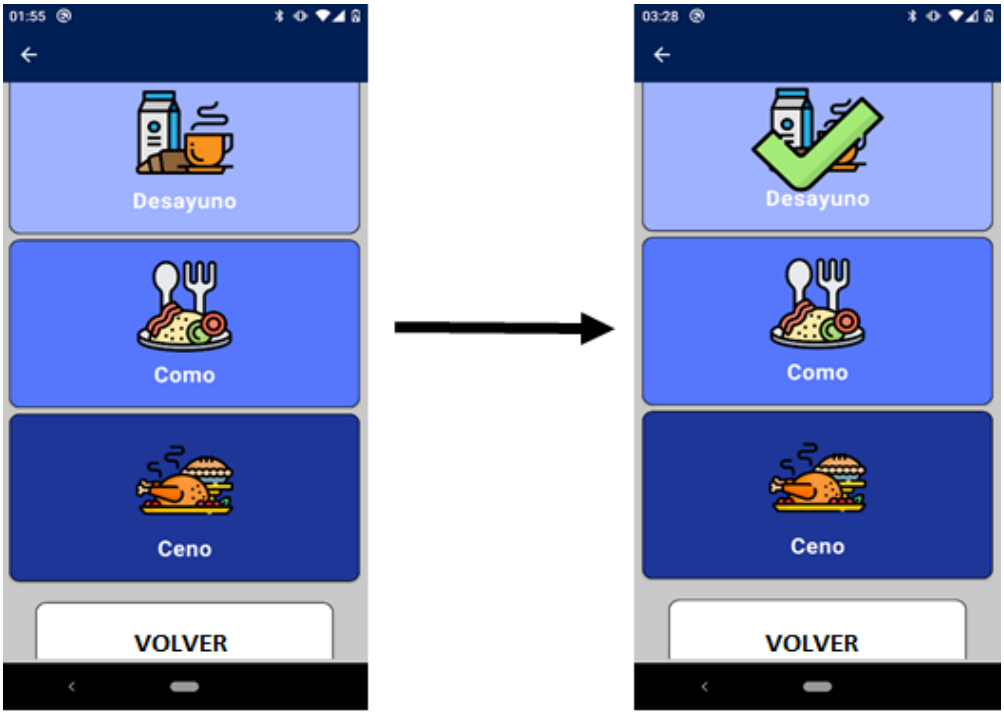

Figura 32 - Design e funcionamento da página das refeições

O botão da medicação redireciona o utilizador para a página onde se recolhe a hora em que tomam os medicamentos, figura 32. Tem 2 botões, um para medicação que tomam ao início do dia e um para o fim do dia. Em termos de funcionamento, tem o mesmo esquema da página anterior, ou seja, quando o botão é pressionado aparece um visto, significando que essa medicação já foi tomada e impossibilitando o uso do mesmo botão nesse dia.

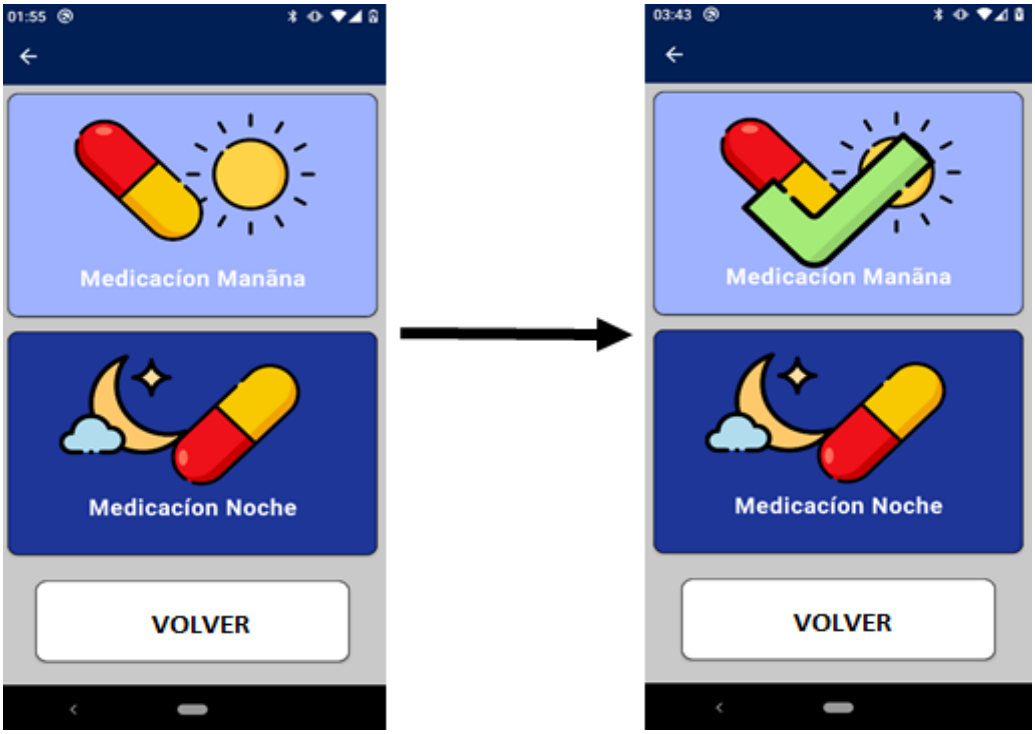

Figura 31 - Design e funcionamento da página dos medicamentos

O botão do Tempo livre mostra uma página com 4 atividades que os idosos podem estar a fazer ao longo do dia, figura 33. Cada uma delas leva a um botão do término dessa atividade.

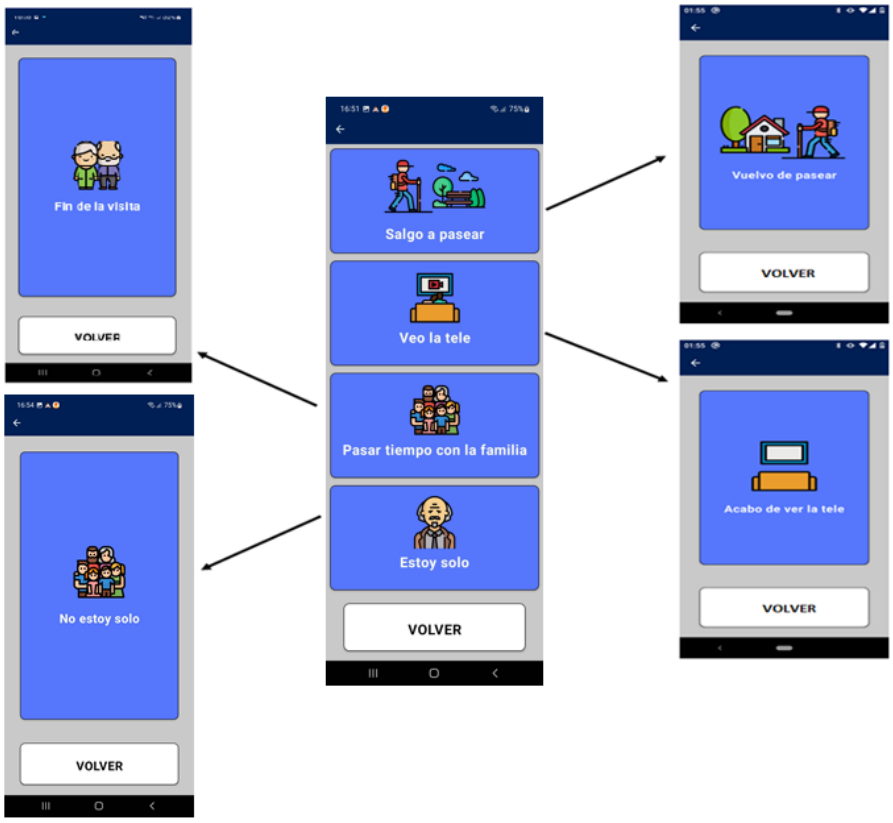

Figura 33 - Design e funcionamento da página de tempo livre

Por último, o botão dos problemas clínicos que direciona para as páginas com sintomas que o utilizador pode ter ao longo do dia. Quando o utilizador tem algum destes sintomas carrega no botão e este fica vermelho, indicando que tem aquele sintoma. Quando deixa de o sentir, carrega de novo no botão e volta a ficar azul indicando que já não tem aquele sintoma, figura 34.

Figura 34 - Design e funcionamento das páginas dos problemas clínicos

4.1.2. Processamento de dados

A aplicação para *smartphone* tem dois tipos de dados para serem processados: a atividade e a localização.

Como referido anteriormente, a atividade é obtida a partir da Google Activity Recognition API [37]. Esta ferramenta devolve uma lista com as atividades, a confiança de cada uma delas e aquela que é a mais provável de ser a atividade do momento. Este projeto só considera e guarda a atividade que é dada como mais provável.

A localização só tem dois valores: ou o utilizador está em casa e o valor é "*Home*", ou o utilizador está fora de casa e o valor é "*Unkown*". Para obter estes valores utilizaram-se dois métodos, ambos usam os dados fornecidos pelo utilizador quando dá a conhecer qual a sua rede WiFi de casa. Quando o utilizador escolhe a rede WiFi de casa de uma lista fornecida pela procura feita pela aplicação, obtém-se duas informações importantes: o nome da WiFi de casa e obtém-se igualmente a localização GPS no momento que ele escolhe o WiFi.

Com o nome do WiFi de casa, sempre que for possível fazer uma procura por redes WiFi [47] que o *smartphone* consegue alcançar, também se consegue o nome da WiFi que o *smartphone* está conectado, se estiver conectado a alguma. Desta maneira, ao comparar os dois nomes, é possível saber se o utilizador está ou não em casa.

Com a localização GPS de casa e com uma frequente atualização da localização GPS [46] pode-se fazer o cálculo da distância que o utilizador está de casa. Considera-se que o utilizador está fora de casa se estiver a uma distância superior a 100 metros.

4.1.3. Dashboard

Esta aplicação *web* tem como objetivo mostrar aos médicos e enfermeiros, que cuidam dos idosos, todos os dados que recolhemos tanto pela aplicação de *smartphone* como os sinais vitais obtidos pelos routers. Desta forma, em conjunto com o estudante Guilherme Lemos do grupo de trabalho do Prof. Dr. Jorge Sá Silva e do Prof. Dr. André Rodrigues, desenvolveu-se esta aplicação, figura 35, 36 e 37. Cada um desenvolveu a parte da aplicação que correspondia aos dados que recolheu com o seu trabalho, ou seja, fiquei encarregue de desenvolver a parte da aplicação que mostrava os dados vindos do *smartphone* e o Guilherme Lemos a que mostrava os dados vindos dos *routers*. Para o desenvolvimento desta *web app* só foram usados componentes da biblioteca Dash [18], *GET Request* para o modulo COMET do FIWARE[29] para obter os dados que vão ser mostrados nos gráficos e também bibliotecas que ajudaram no processamento dos dados recebidos, com as respetivas datas.

A aplicação começa por pedir ao utilizador as credenciais para ter acesso aos dados recolhidos pelas duas tecnologias, figura 35. Para proteger os dados recolhidos dos idosos do lar, as credenciais de acesso só vão ser fornecidas aos médicos que os acompanham. Tal como na aplicação de *smartphone*, foi usado o Firebase para se fazer a autenticação dos utilizadores. Estas contas são criadas diretamente no Firebase, pois a aplicação web não tem como criar conta para que só os médicos tenham acesso a estas.

Depois de fazer o *login*, e como se pode ver nas figuras 36 e 37, o médico consegue pesquisar o paciente pelo ID e pelo dia que pretende consultar os dados. Para além disso existe uma página que mostra, em tempo real, os dados do paciente escolhido. Seguidamente, tem dois botões para mostrar os diferentes dados.

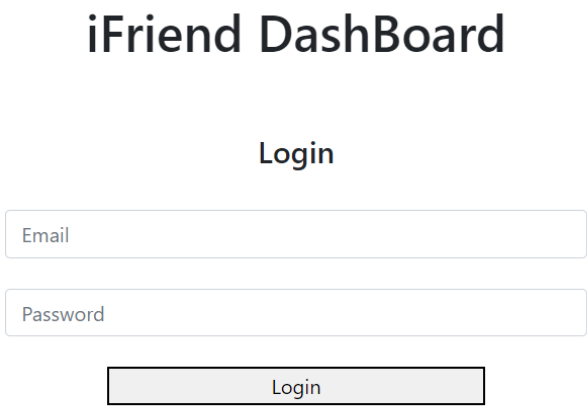

Figura 35 - Login no Dashboard

O primeiro mostra os gráficos do ritmo respiratório e cardíaco, e as médias de cada um num certo intervalo de tempo escolhido.

O outro botão mostra os dados da rotina do utilizador, a localização, atividade, número de passos, e um gráfico com os dados do sono que foram obtidos através da aplicação de smartphone. Estes foram os dados que os médicos solicitaram.

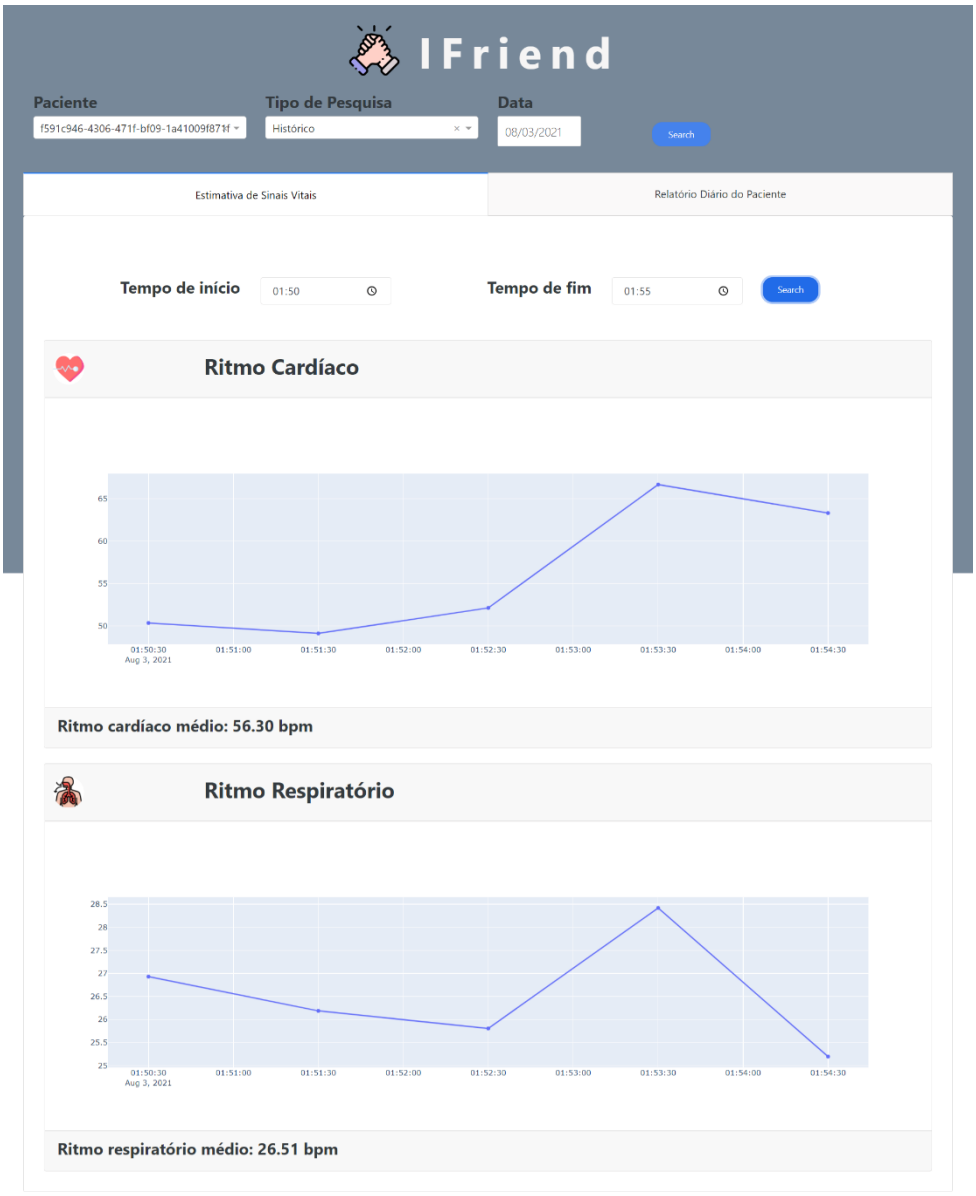

Figura 36 - Dash Web App sinais vitais

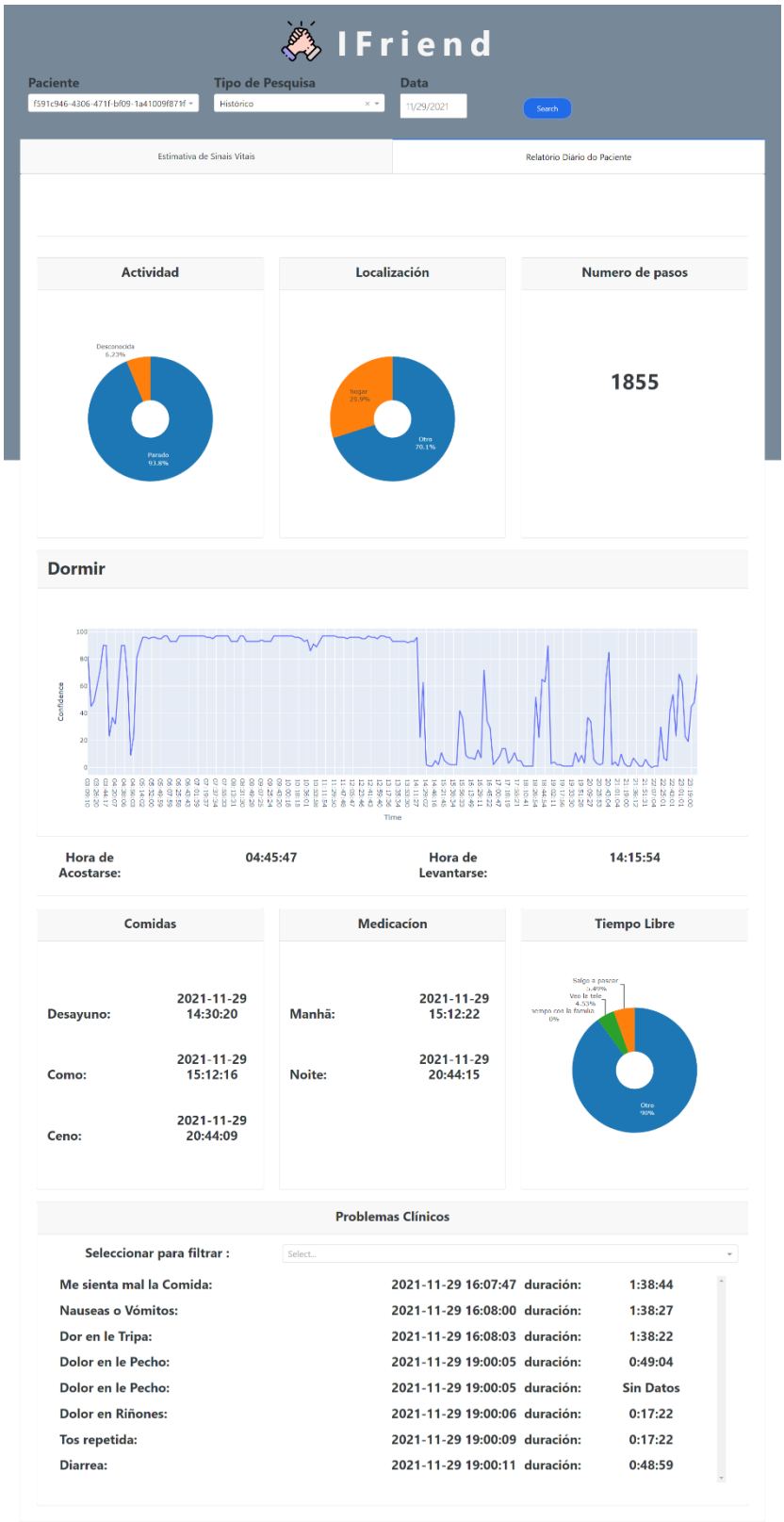

Figura 37 - Dash Web App Relatório Diário

Para obter os dados da aplicação usou-se o modulo COMET do FIWARE[29], que permite fazer um *GET Request* para se conseguir efetuar uma consulta dos dados. Este mecanismo permite recolher a informação desejada num certo intervalo de tempo e agregada de diferentes formas.

Para a informação da rotina, verifica-se que botões foram carregados, e adiciona-se o nome e a hora na tabela, de forma que faça sentido cronologicamente.

Para a atividade faz-se a contagem de cada um dos tipos de atividades, depois coloca-se no gráfico circular de maneira a mostrar que percentagem do dia cada atividade ocupou.

Do mesmo modo, a localização também segue o mesmo padrão, conta-se quantas vezes esteve em casa e quantas vezes teve fora, apresentando-se a percentagem de cada um no gráfico circular.

No caso do número de passos, o valor obtido pelo sensor do *smartphone* é o total de passos desde que se abriu a aplicação pela primeira vez. Então, para se obter o número de passos naquele dia subtrai-se o total de passos do início do dia, ao total de passos do final do dia.

Por último, o gráfico com os dados do sono, é um gráfico linear onde no eixo do x tem a hora que se recebeu dados do sono e no eixo do y a confiança. Quanto maior for a confiança, maior a probabilidade de o utilizador estar a dormir.

4.2. Considerações ao trabalho realizado

Os testes desta aplicação vão ser realizados em 3 fases, aplicados por diferentes entidades e em diferentes pessoas. Um primeiro teste, realizado pela Universidade de Coimbra, onde se vai testar em estudantes numa sala com duas pessoas, com o âmbito de verificar se o sistema funciona como o pretendido. Um segundo teste, efetuado pela Universidade de Coimbra, Hospital Príncipe de Astúrias e Universidade de Alcalá, aplicado em idosos com insuficiência renal de um lar e com o mesmo âmbito do anterior teste. E um último teste, executado no Hospital Príncipe de Astúrias e na Universidade de Alcalá, com os mesmos idosos do lar numa escala maior com o objetivo de obter dados para analisar e desenvolver um algoritmo preditivo de problemas clínicos.

Foram realizados testes de laboratório, onde a aplicação foi utilizada durante alguns dias para fazer a recolha dos dados da rotina do utilizador. Desta forma foi possível corrigir erros que apareceram, confirmar que os dados eram guardados corretamente e fazer recolha de dados, permitindo testar o *dashboard* criado para mostrar os dados dos idosos aos médicos.

Ao longo do desenvolvimento deste projeto, foram feitas reuniões com o grupo de trabalho do Prof. Dr. Jorge Sá Silva e do Prof. Dr. André Rodrigues e com os parceiros do projeto iFriend, de forma a obter diferentes opiniões e visões sobre o projeto, de maneira que o projeto esteja sempre a melhorar, assim como decidir os próximos passos a executar para o desenvolvimento do projeto e dar a conhecer o estado do projeto aos parceiros do projeto, figuras 38 e 39.

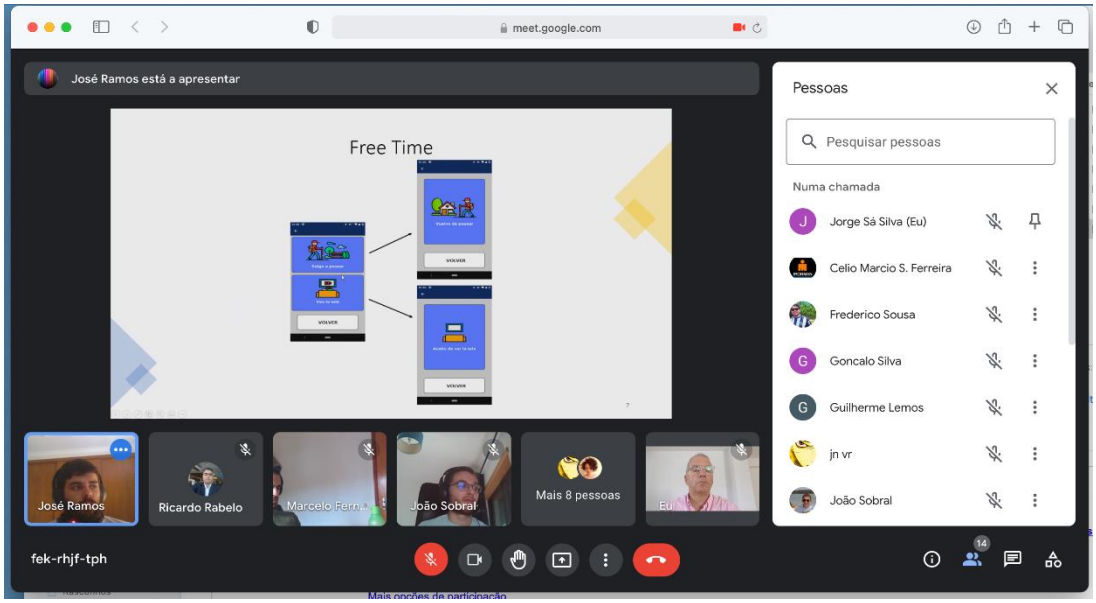

Figura 38 - Reunião com grupo de trabalho

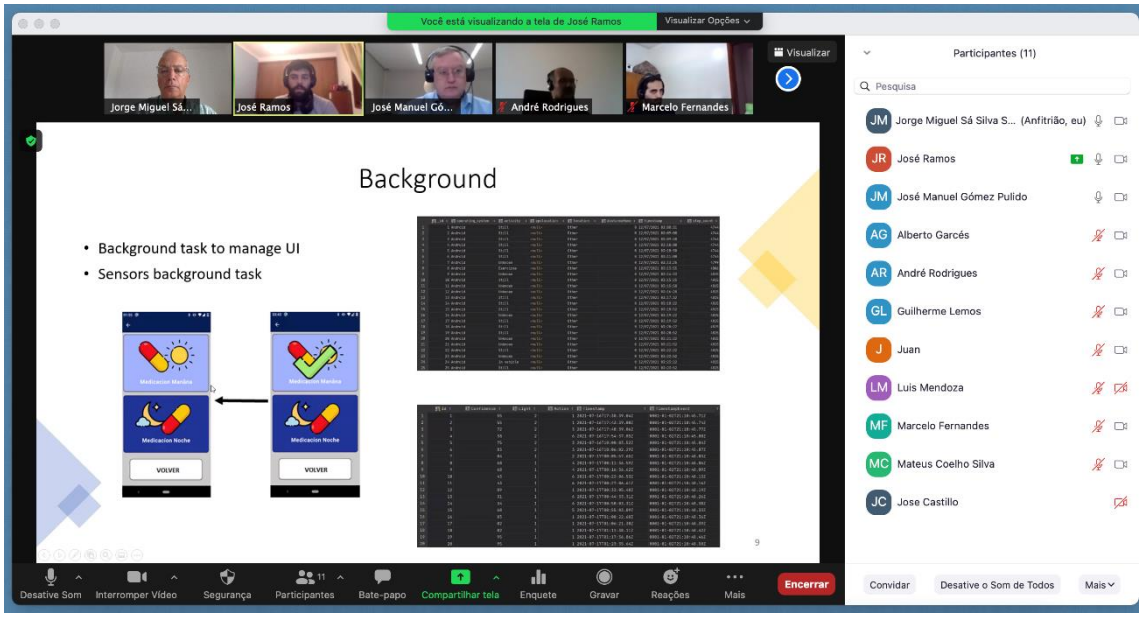

Figura 39 - Reunião com os parceiros do iFriend

5. Conclusão e trabalho futuro

O objetivo do projeto iFriend foi construir um sistema HITLCPS com o propósito de monitorizar idosos de um lar, com insuficiência renal, de maneira a conseguir avaliar o seu estado de saúde e conseguir fazer uma possível inferência de doenças que poderão estar a desenvolver-se.

Desta forma, este sistema para ser bem concebido teve de ser composto por três componentes: a aquisição de dados, a inferência de estados e a atuação. Destes, o primeiro está desenvolvido de maneira que seja possível trabalhar nas outras duas componentes.

A aquisição de dados foi conseguida pela recolha de sinais vitais, a partir de redes WiFi e com a recolha de dados de uma aplicação para *smartphone*. Esta aplicação foi desenvolvida em Xamarin, que possibilita a criação de aplicações tanto para Android como para IOS, alcançando mais utilizadores. Nesta, é recolhida a rotina dos utilizadores através do uso dos botões da aplicação e os dados dos sensores do smartphone através da criação de uma biblioteca específica para o efeito.

A inferência de estados e a atuação ainda estão num estado de desenvolvimento inicial, pois faltam fazer testes do sistema numa situação real, que seria no lar de idosos, para se adquirir dados suficientes para utilizar algoritmos, para descobrir algum tipo de doença. De qualquer modo, a obtenção da inferência de estados não fazia parte dos objetivos desta dissertação. E por estas razões, esta parte do projeto vai ser complementada pelos parceiros dos projetos.

Por fim, no futuro os nossos parceiros farão testes no lar de idosos para perceber se é possível desenvolver algum tipo de algoritmo para inferir possíveis doenças que os idosos estejam a desenvolver.

Referências

1. Grandez, K., et al. *Wearable wireless sensor for the gait monitorization of Parkinsonian patients*. IEEE.
2. Soares, M., *Metodologias Ágeis Extreme Programming e Scrum para o Desenvolvimento de Software.* Revista Eletrônica de Sistemas de Informação, 2004. **3**.
3. Routh, K. and T. Pal. *A survey on technological, business and societal aspects of Internet of Things by Q3, 2017*. in *2018 3rd International Conference On Internet of Things: Smart Innovation and Usages (IoT-SIU)*. 2018.
4. Ziegler, S., et al. *Internet of Things and crowd sourcing - a paradigm change for the research on the Internet of Things*. in *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*. 2015.
5. Leng, J., Z. Lin, and P. Wang. *Poster Abstract: An Implementation of an Internet of Things System for Smart Hospitals*. in *2020 IEEE/ACM Fifth International Conference on Internet-of-Things Design and Implementation (IoTDI)*. 2020.
6. Nunes, D.S., P. Zhang, and J.S. Silva, *A Survey on Human-in-the-Loop Applications Towards an Internet of All.* IEEE Communications Surveys & Tutorials, 2015. **17**(2): p. 944-965.
7. Jirgl, M., Z. Bradac, and P. Fiedler, *Human-in-the-Loop Issue in Context of the Cyber-Physical Systems.* IFAC-PapersOnLine, 2018. **51**(6): p. 225-230.
8. Microsoft. *O que é middleware?* ; Available from: https://azure.microsoft.com/pt-br/overview/what-is-middleware/.
9. Sarkar, A., et al. *Android Application Development: A Brief Overview of Android Platforms and Evolution of Security Systems*. in *2019 Third International conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*. 2019.
10. Thongkaew, S., et al. *Dalvik bytecode acceleration using Fetch/Decode Hardware Extension with hybrid Execution*. in *2014 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*. 2014.
11. Microsoft-Docs. *What is Xamarin?* ; Available from: https://docs.microsoft.com/en-us/xamarin/get-started/what-is-xamarin.
12. Microsoft-Docs. *What is Xamarin.Forms?* 2021; Available from: https://docs.microsoft.com/en-us/xamarin/get-started/what-is-xamarin-forms.
13. Wall, N., *Xamarin Forms - Porting a Windows Mobile App into a Cross-Platform Mobile App*. 2018.
14. Vishal, K. and A.S. Kushwaha. *Mobile Application Development Research Based on Xamarin Platform*. in *2018 4th International Conference on Computing Sciences (ICCS)*. 2018.
15. GIt. *Getting Started - About Version Control*. Available from: https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control.
16. Armando, N., et al., *WSNs in FIWARE -- Towards the Development of People-Centric Applications*. 2017. 445-456.
17. FIWARE. *What is FIWARE?* ; Available from: https://www.fiware.org/about-us/.
18. Plotly. *Introducing Dash*. Available from: https://medium.com/plotly/introducing-dash-5ecf7191b503.
19. Plotly. *Introduction to Dash*. Available from: https://dash.plotly.com/introduction.
20. Plotly. *Manufacturing SPC Dashboard*. Available from: https://dash.gallery/dash-manufacture-spc-dashboard/.
21. Goggle. *Firebase*. Available from: https://firebase.google.com/.

22. Rajappa, A., et al. *Implementation of PingER on Android Mobile Devices Using Firebase*. in *2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*. 2020.
23. Postman. *What is Postman?* ; Available from: https://www.postman.com/product/what-is-postman/.
24. Fernandes, J.M.d.S.L., *ISABELA: IoT Student Advisor and BEst Life Analyser*. 2017, Universidade de Coimbra.
25. Pan, S., et al. *A Sensor Glove for the Interaction with a Nursing-Care Assistive Robot*. in *2019 IEEE International Conference on Industrial Cyber Physical Systems (ICPS)*. 2019.
26. Barbosa, R., et al. *An architecture for emotional smartphones in Internet of Things*. in *2016 IEEE Ecuador Technical Chapters Meeting (ETCM)*. 2016.
27. FIWARE. *Welcome to Orion Context Broker*. Available from: https://fiware-orion.readthedocs.io/en/master/index.html.
28. FIWARE. *CYGNUS*. Available from: https://fiware-cygnus.readthedocs.io/en/latest/#welcome.
29. FIWARE. *WELCOME TO THE FIWARE SHORT TIME HISTORIC (STH) - COMET DOCUMENTATION.*; Available from: https://fiware-sth-comet.readthedocs.io/en/latest/index.html.
30. Liu, J., et al., *Tracking Vital Signs During Sleep Leveraging Off-the-shelf WiFi*. 2015. 267-276.
31. Chowdhury, T., *Using Wi-Fi Channel State Information (CSI) for Human Activity Recognition and Fall Detection the thesis can also be accessed at: https://open.library.ubc.ca/cIRcle/collections/ubctheses/24/items/1.0365967*. 2018.
32. Plotly. *Basic Dash Callbacks*. Available from: https://dash.plotly.com/basic-callbacks.
33. Wondershare. *Mockitt*. Available from: https://mockitt.wondershare.com/.
34. Microsoft-Docs. *Xamarin.Essentials*. Available from: https://docs.microsoft.com/en-us/xamarin/essentials/.
35. Microsoft-Docs. *Foreground Services*. 2022; Available from: https://docs.microsoft.com/en-us/xamarin/android/app-fundamentals/services/foreground-services.
36. Microsoft-Docs. *Xamarin.Forms Device Class*. 2021; Available from: https://docs.microsoft.com/en-us/xamarin/xamarin-forms/platform/device.
37. Google. *ActivityRecognitionClient*. 2022; Available from: https://developers.google.com/android/reference/com/google/android/gms/location/ActivityRecognitionClient.
38. Microsoft-Docs. *DateTime Struct*. Available from: https://docs.microsoft.com/en-us/dotnet/api/system.datetime?view=net-6.0.
39. Microsoft-Docs. *Android.Hardware Namespace*. Available from: https://docs.microsoft.com/en-us/dotnet/api/android.hardware?view=xamarin-android-sdk-12.
40. Microsoft-Docs. *SensorType Enum*. Available from: https://docs.microsoft.com/en-us/dotnet/api/android.hardware.sensortype?view=xamarin-android-sdk-12.
41. Microsoft-Docs. *SensorManager Class*. Available from: https://docs.microsoft.com/en-us/dotnet/api/android.hardware.sensormanager?view=xamarin-android-sdk-12.
42. Microsoft-Docs. *KeyguardManager Class*. Available from: https://docs.microsoft.com/en-us/dotnet/api/android.app.keyguardmanager?view=xamarin-android-sdk-12.
43. Microsoft-Docs. *UsageStatsManager Class*. Available from: https://docs.microsoft.com/en-us/dotnet/api/android.app.usage.usagestatsmanager?view=xamarin-android-sdk-12.

44. Microsoft-Docs. *Android Audio*. 2021; Available from: https://docs.microsoft.com/en-us/xamarin/android/app-fundamentals/android-audio.
45. Microsoft-Docs. *Xamarin.Android*. Available from: https://docs.microsoft.com/en-us/xamarin/android/.
46. Microsoft-Docs. *Location services on Android*. 2022; Available from: https://docs.microsoft.com/en-us/xamarin/android/platform/maps-and-location/location.
47. Microsoft-Docs. *WifiManager Class*. Available from: https://docs.microsoft.com/en-us/dotnet/api/android.net.wifi.wifimanager?view=xamarin-android-sdk-12.
48. Microsoft-Docs. *Broadcast Receivers in Xamarin.Android*. 2020; Available from: https://docs.microsoft.com/en-us/xamarin/android/app-fundamentals/broadcast-receivers.
49. xabre, s., janusw. *Plugin.BLE*. Available from: https://www.nuget.org/packages/Plugin.BLE#readme-body-tab.
50. Microsoft-Docs. *Xamarin.Essentials: Preferências*. Available from: https://docs.microsoft.com/en-us/xamarin/essentials/preferences?context=xamarin%2Fandroid&tabs=android.
51. Microsoft. *Store data in a local SQLite.NET database*. Available from: https://docs.microsoft.com/en-us/xamarin/get-started/quickstarts/database?pivots=macos.
52. Google. *Firebase Authentication*. Available from: https://firebase.google.com/docs/auth?hl=pt-br.